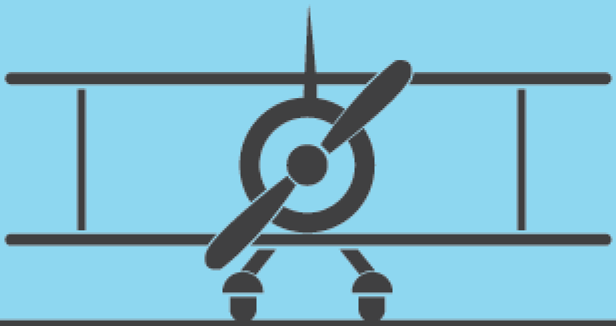


Windows Azure SQL Reporting Succinctly

by Stacia Misner

Syncfusion | Technology Resource Portal

Windows Azure SQL Reporting Succinctly

By
Stacia Misner

Foreword by Daniel Jebaraj

Technical Reviewer: Clay Burch, Ph.D., director of technical support, Syncfusion, Inc.

Copy Editor: Courtney Wright

Acquisitions Coordinator: Marissa Keller Outten, director of business development, Syncfusion, Inc.

Proofreader: Graham High, content producer, Syncfusion, Inc.

Table of Contents

The Story behind the *Succinctly* Series of Books

Daniel Jebaraj, Vice President
Syncfusion, Inc.

Staying on the cutting edge

As many of you may know, Syncfusion is a provider of software components for the Microsoft platform. This puts us in the exciting but challenging position of always being on the cutting edge.

Whenever platforms or tools are shipping out of Microsoft, which seems to be about every other week these days, we have to educate ourselves, quickly.

Information is plentiful but harder to digest

In reality, this translates into a lot of book orders, blog searches, and Twitter scans.

While more information is becoming available on the Internet and more and more books are being published, even on topics that are relatively new, one aspect that continues to inhibit us is the inability to find concise technology overview books.

We are usually faced with two options: read several 500+ page books or scour the web for relevant blog posts and other articles. Just as everyone else who has a job to do and customers to serve, we find this quite frustrating.

The *Succinctly* series

This frustration translated into a deep desire to produce a series of concise technical books that would be targeted at developers working on the Microsoft platform.

We firmly believe, given the background knowledge such developers have, that most topics can be translated into books that are between 50 and 100 pages.

This is exactly what we resolved to accomplish with the *Succinctly* series. Isn't everything wonderful born out of a deep desire to change things for the better?

The best authors, the best content

Each author was carefully chosen from a pool of talented experts who shared our vision. The book you now hold in your hands, and the others available in this series, are a result of the authors' tireless work. You will find original content that is guaranteed to get you up and running in about the time it takes to drink a few cups of coffee.

About the Author

Stacia Misner is a Microsoft SQL Server MVP, SQL Server Analysis Services Maestro, Microsoft Certified IT Professional-BI, and Microsoft Certified Technology Specialist-BI with a Bachelor's degree in Social Sciences. As a consultant, educator, author, and mentor, her career spans more than 25 years, with a focus on improving business practices through technology. Since 2000, Stacia has been providing consulting and education services for Microsoft's business intelligence technologies, and in 2006 she founded Data Inspirations. During these years, she has authored or co-authored multiple books and articles as well as delivered classes and presentations around the world covering different components of the Microsoft SQL Server database and BI platform.

Chapter 1 Introduction to SQL Reporting

Maybe you've heard of Windows Azure and SQL Server, and even SQL Server Reporting Services. But what exactly is Windows Azure SQL Reporting? In this introduction, I start by providing an answer to this question. Next, I provide a comparison of SQL Reporting to Reporting Services, so that you can better understand the similarities and differences between a cloud-based and an on-premises reporting solution. At this point, you might still wonder why you would need SQL Reporting instead of using a perfectly good Reporting Services solution, so I also explain some of the advantages of SQL Reporting and describe some common scenarios for which it's well suited.

Note: Although familiarity with SQL Server Reporting Services is helpful, it is not required. You will learn the basics necessary to successfully create, deploy, manage, and view reports in this e-book.

What Is SQL Reporting?

A simple broad explanation is that SQL Reporting is a version of Reporting Services…in the cloud. But what does it mean to be "in the cloud?" Conceptually, an application in the cloud relies on a hardware, security, application, and networking infrastructure that you access over the Internet and is built, maintained, monitored, and tuned as a commercial service by a third party. Typically, you pay a subscription to use the computing power delivered by the cloud provider. In the case of SQL Reporting, that third party is Microsoft.

Like its on-premises predecessor, Reporting Services, SQL Reporting is a platform that supports three different types of activities that we call the reporting life cycle:

- Report development
- Report management
- Report access

Although SQL Reporting does not itself provide tools for **report development**, it supports the same tools commonly used by professional report developers and less technical business users who create reports for Reporting Services. SQL Server Data Tools, Business Intelligence Development Studio, and Report Builder allow you to create a report locally. You can then later connect to your SQL Reporting server to save it.

After you have saved one or more reports to SQL Reporting, you use a portal to perform a variety of **report management** tasks. You can organize reports into folders, secure reports, and monitor usage.

SQL Reporting supports **report access** to users through a portal, direct access by using a URL, or a custom application using a Web service method. Users can view a report online, or export it to a variety of other formats, such as Excel and PDF.

How Does SQL Reporting Compare to Reporting Services?

Although SQL Reporting is similar to Reporting Services, it doesn't have parity with the features in Reporting Services. Table 1 summarizes the features supported in each product.

Table 1: Feature Comparison

Feature	Reporting Services	SQL Reporting
Design tools (SSDT, BIDS, Report Builder)	Yes	Yes
URL Access	Yes	Yes
Export to multiple formats	Yes	Yes
Support for multiple types of data sources	Yes	SQL Database only
Web application for report portal	Yes	
Export as data feed	Yes	
Caching	Yes	
Subscriptions	Yes	
Snapshots	Yes	
Report history	Yes	
External images	Yes	
Extensions	Yes	

Feature	Reporting Services	SQL Reporting
Custom code in reports	Yes	
SharePoint integration	Yes	
Windows integrated security	Yes	
Custom authentication	Yes	Yes
Role-based security	Yes	Yes
URL Access	Yes	Yes
Embed reporting in custom application	Yes	Yes

Reporting Services is a mature, full-featured, on-premises reporting platform capable of providing access to reports in a variety of ways to people inside or outside your organization through on-demand execution or scheduled delivery processes. To ensure optimal performance, it includes options for background processing and cache management.

By contrast, SQL Reporting is a component of the Windows Azure platform that is used as a cloud-based service for a portion of the reporting life cycle. In essence, you exchange your on-premise server for a comparable server, or farm of servers, in the cloud. Although you continue to build reports using the same tools available for Reporting Services, you cannot customize SQL Reporting reports through the use of embedded code or external assemblies. Furthermore, you cannot use external images or export a report as a data feed.

After you publish the reports to SQL Reporting, you can use an administration portal to organize and secure content in the cloud. However, many of the administration features available in Reporting Services are missing. For example, you cannot configure background processing options such as caching, snapshots, or subscriptions, nor can you manage report history.

When you access reports from SQL Reporting, you can access them from a reporting portal, but it lacks the interface that you find in the Report Manager web application available in Reporting Services. Instead of a polished user interface, you see a list of links to folders and reports. Once you open a report, the experience between SQL Reporting and Reporting Services is very similar. One exception is the inability to export a SQL Reporting report to a data feed. All other export options found in Reporting Services are available in SQL Reporting.

Why Use SQL Reporting?

Why use SQL Reporting if it doesn't have feature parity with Reporting Services? For some situations, SQL Reporting is definitely not the right solution. However, there are many scenarios for which it is a perfect fit.

Before considering such scenarios, let's consider some advantages you gain by having a cloud-based reporting solution. First, you can add a new server in just a few minutes, whenever you need it, and for as long as you need it. You simply can't make that happen in your own organization as quickly and cost-effectively considering the time and money needed to acquire hardware and software and then to get it configured, secured, installed with software, and so on.

Furthermore, if you need to support a lot of users, you need to create an environment that scales. An on-premises solution requires more time, money, hardware, and resources to achieve scale, but with SQL Reporting, you can add more servers as needed very easily, and you can take them away just as easily. Like other services available for the Windows Azure Platform, SQL Reporting is designed to support elastic scale. You only pay for what you use and can add or release excess capacity, depending on your needs. Based on your day-to-day, month-to-month, or seasonal requirements, you can easily and quickly add subscriptions and report servers to extend the capacity of your SQL Reporting environment or delete subscriptions and servers to decrease it.

A cloud-based reporting solution frees you from maintenance overhead activities such as applying patches to a server and ensuring it has been updated with the latest service packs. At this point, the extent of management for your SQL Reporting Server is limited to adding users and granting them permissions to the server resources. SQL Reporting includes a management portal where you can perform these tasks manually, or you can develop your own security management application using the Web service API.

SQL Reporting offers enough flexibility to adapt to a variety of needs. Just as no two Reporting Services implementations are identical, there are a variety of ways to use SQL Reporting. Let's consider four possible use case scenarios for SQL Reporting:

- Small company
- Large company
- Limited project life span
- Cloud application developer

Small Company

Even a small company has questions about how well the business is doing. You might have one or two business applications collecting data, but you need a better way to use that data. When you don't have a formal IT department, you might not have anyone in your business with the right skills to set up and maintain a report server. In this case, SQL Reporting can deliver the capabilities you need without adding staff.

SQL Reporting allows you to start gaining insights into your data even without a formal set of reports in place. You can set up a variety of report parts that people can use to construct their own reports as part of SQL Reporting's self-service business intelligence features. That is, people can build their own reports in response to specific questions using pre-constructed report parts and datasets without knowing how to write queries or configure data regions. Even better, when your organization has a centralized place to store the resulting reports, everyone in the company can take advantage of what someone else has created, and that's how new insights are developed. The bottom line is that business intelligence is not just for big companies.

Large Company

In a large company, you have an IT staff and a technical infrastructure in place to support reporting needs. However, the need to support mobile workers has become increasingly more common. These mobile workers need web access to corporate data when they're away from the office whether that's through a browser or a mobile device such as a smartphone or tablet. Although this scenario is supported by Reporting Services, it's not trivial to set up. Security can be challenging to configure, any time you open access to the corporate network to the Internet, there is a real risk that this opening can be exploited.

You can isolate data that people need for reporting by putting it in SQL Database, a cloud-based relational database platform. Then build the reports your mobile users require and deploy those reports to SQL Reporting. That way, they can retrieve the data they need securely without putting your corporate network at risk.

Limited Project Lifespan

Regardless of the size of your company, another scenario that lends itself well to a SQL Reporting solution is a project that has a limited lifespan but requires you to scale a solution quickly for thousands of users. As an example, let's say you have a quarterly survey for which there is a lot of activity spanning a few weeks, during which time you gather the data, and then during the next few weeks, you share the aggregate data with these users.

When you use an on-premises solution based on SQL Server and Reporting Services for a project such as this, there are many steps to take before your solution is ready for production. You must acquire the hardware, install the software, perform thorough load testing, and so on. Considerable time and money is spent to support a requirement whose lifetime is measured in weeks.

On the other hand, with a cloud solution based on SQL Database and SQL Reporting, you can add servers when you need them, enable your application, gather the data, and deliver reporting with minimal effort. It takes only minutes to get a scalable and highly available infrastructure in place. Later, when you're ready to retire the project, you can archive the data back to an on-premises server.

Cloud Application Developer

Yet another scenario to consider is integration with a Windows Azure cloud application. Let’s say you’re a developer of an application that performs a service and, like a traditional on-premises solution, gathers data in the process. Further, you want to include reporting as part of your solution. There’s no need to develop a reporting mechanism yourself. You can simply add SQL Reporting to your application architecture and easily deliver reporting capabilities to your users.

Chapter 2 Getting Started

Before you can start delivering reports with SQL Reporting, you need to have data available for those reports. SQL Reporting can use only SQL Database as a data source, so you need to plan some time for establishing a database to use for reports. To do this, you need a Windows Azure subscription to which you add the SQL Database and SQL Reporting services. Then you create a database in SQL Database and migrate data into it.

Windows Azure Setup

SQL Database and SQL Reporting are separate services available on the Windows Azure platform. Windows Azure is a subscription-based, cloud-computing platform hosted by Microsoft. Prior to activating a subscription, you must create a Microsoft account (formerly known as a Windows Live ID) at http://go.microsoft.com/fwlink/p/?LinkID=238657. Next, sign up for Windows Azure at http://windows.azure.com.

Rather than pay licensing and maintenance fees for software, and investing in hardware, you can pay a monthly bill for provisioned services. At the time of this writing, Windows Azure is available as one of the following types of subscriptions:

- **3-Month Free Trial**. You can sign up for a full-featured, free trial version of Windows Azure, which includes 750 hours of SQL Database and 100 hours per month of SQL Reporting, in addition to other services.
- **Pay-As-You-Go**. You pay for the services you use. SQL Database pricing is currently based on database size. SQL Reporting pricing is based on the number of clock hours per deployed instance. In addition, if the number of reports executed with a clock hour exceeds a specified threshold, an additional hour is billed. Refer to Pricing Details for current pay-as-you-go rates for SQL Database, SQL Reporting, and other services.
- **6-Month Plan or 12-Month Plan**. With these plans, you make a minimum monthly commitment to receive a discount against the base pay-as-you-go rate. A higher monthly commitment earns a higher discount. If you don't provision enough services to meet the minimum commitment, the balance rolls over to a future month, and is forfeited at the end of the subscription term if never used.

SQL Database Setup

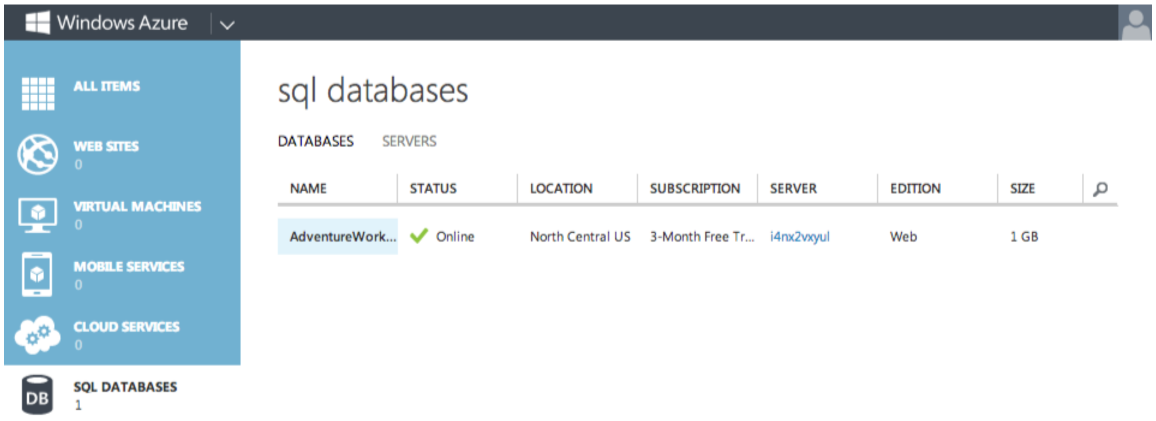
When you create a new database on the SQL Database server, you must decide which edition to use and specify the maximum size of the database. You have the following choices for your database:

- Web edition: 1 GB or 5 GB
- Business edition: 10 GB, 20 GB, 30 GB, 40 GB, 50 GB, 100 GB, or 150 GB.

To add the SQL Database service to your subscription, use the **Management Portal** link on the Windows Azure home page. Then in the Management Portal, click **New** in the bottom left corner, point to **Data Services**, and then point to **SQL Database**. You have the following three options:

- **Quick create**. Use this option to create a database by providing a name, assigning it to a database server (or creating a new database server if none exist yet in your subscription), and creating a login name and password. The database is automatically created as a 1-GB Web edition database and assigned to a server and a data center location.
- **Custom create**. Use this option to create a database by providing a name, selecting an edition, a maximum database size, a collation method, and a server.
- **Import**. Use this option to copy a database from another SQL Database server or from an on-premises SQL Server database instance. Before you select this option, however, you must export the database that you want to copy to a BACPAC file and store it in a Windows Azure blob storage account. I explain more about how to do this in the Data-Tier Application Export and Import section later in this chapter. To perform the import, you supply the URL and a name for the database, select the server, and create a login name and password.

After creating your database, it appears in the **SQL Databases** list in the Windows Azure Management Portal, as shown in Figure 1. Here you can see its status, the data center location, the Windows Azure subscription with which it is associated, the server name, the database edition, and its size.



NAME	STATUS	LOCATION	SUBSCRIPTION	SERVER	EDITION	SIZE
AdventureWork...	✓ Online	North Central US	3-Month Free Tr...	i4nx2vxyul	Web	1 GB

Figure 1: SQL Databases in Windows Azure Management Portal

To view more information about the database, click its name in the **SQL Databases** list. At the top of the page is a dashboard containing one chart that displays database activity for the past 24 hours and another chart comparing the current size of the database to the maximum size defined. (If you later decide you want to change the edition type or the database size, click the **Configure** link at the top of the page.)

Scroll through the dashboard page to view additional information about the database in the lower portion of the page. Here you can find the fully qualified domain name of the server and to locate the URL for online management of the database. This URL opens the Management Portal for SQL Database, which allows you to execute queries or design tables, views, or stored procedures for your database. You might prefer to use SQL Server Management Studio instead if you already have an on-premises SQL Server instance available.

You cannot access the server through the online Management Portal or from SQL Server Management Studio until you add your computer's IP address to the server configuration. To do this, click the **SQL Databases** icon in the navigation panel on the left, click the **Servers** link above the list of databases, and then click **Configure**. You can add your current IP address with a single click and you can set up additional rules to grant access to a range of IP addresses if necessary. Be sure to click **Save** at the bottom of the page when you have finished creating rules.

Data Migration Options

Once your database server is in place, you are ready to add a database and load data. If you are moving a database from an on-premises database server, you have several options:

- Deploy Database Wizard
- Database script
- Windows Azure SQL Data Sync
- Data-Tier Application Export and Import
- SQL Server Integration Services
- Bulk Copy Utility

SQL Database Migration Wizard

You can download an open source application that analyzes your database for compatibility issues and optionally performs a database migration for you from Codeplex. Because it is an open source application, it is not supported by Microsoft.

Deploy Database Wizard

If you are running an on-premises SQL Server 2012 instance, you can use the Deployment Wizard in SQL Server Management Studio to deploy a SQL Server 2008, 2008 R2, or 2012 database. To do this, right-click the database to migrate in Object Explorer, point to **Tasks**, and click **Deploy Database to SQL Azure**. On the **Deployment Settings** page of the wizard, click **Connect** to define the connection to the server by providing the server name and login information. You can also change the database name if you like. You have the option here to specify the database edition and maximum size of the database.

Of all the data migration options available, the Deploy Database Wizard is the most straightforward. However, there are potential issues with your database that can cause problems with the migration. Review the guidelines and limitations at MSDN to understand what you need to address prior to migrating your database.

Database Script

In SQL Server Management Studio, you can use the Generate Scripts Wizard to produce T-SQL scripts for migrating your on-premises database to your SQL Database server. In Object Explorer, right-click the database, point to **Tasks**, and click **Generate Scripts**. Step through the wizard to select the objects to migrate, either the entire database or specific database objects, and save the scripts to a single file. Before you save the file, click **Advanced** on the **Set Scripting Options** page of the wizard to change some of the options for compatibility with SQL Database as shown in Table 2.

Table 2: Scripting Options for Generate Scripts Wizard

Option	Setting
Convert UDDTs to Base Types	True
Script for the database engine type	SQL Azure Database
Types of data to script	Schema and data

Although you can open and execute script files in the Management Portal for SQL Database, a database script file is likely too large. The portal accepts input of 250,000 characters or less. To run your script, use SQL Server Management Studio to connect to your SQL Database server. In the **Connect To Server** dialog box, you need to provide the entire server name found on the database's dashboard page of the Windows Azure Management Portal. For example, if your server is **xyz123**, then you type in **xyz123.database.windows.net** as the server name. Switch to SQL Authentication and type in the login name and password that you specified when creating the database. Then open the saved script file, select the applicable database in the **Available Databases** drop-down list in the toolbar, and then execute the script.

SQL Database does not fully support T-SQL. Consequently, you might need to modify your script before it can complete successfully. As with the Deploy Database Wizard, you should be familiar with the list of guidelines and limitations at MSDN to understand what you might need to change in your script.

The advantage of using a database script is that it is relatively straightforward. You can use a script to migrate a subset of your database or make changes to the schema where necessary. However, a potential problem with this approach is that the script performs single row inserts of your data. If you have a high volume of data, this approach to data migration can perform poorly.

Windows Azure Data Sync

Another option for migrating data is to use synchronization between an on-premises SQL Server 2005 SP2 (or later) and a SQL Database server. In the Windows Azure management portal, go to the **SQL Databases** page. Point to **Add Sync** on the ribbon at the bottom of the page, and click **New Sync Agent**. If this is the first time you have added a sync agent, you must click the link to download it and then execute the downloaded installer, but leave the **New SQL Data Sync Agent** dialog box open.

When installation of the agent is complete, return to the **New SQL Data Sync Agent** dialog box and type a name for the agent and then select a region and subscription for the agent. On the ribbon at the bottom of the page, click **Manage Key** and then click **Generate** to create the agent access key. Click the icon to the right of the key to copy it to your clipboard, and then run the newly installed application on your computer. Click **Submit Agent Key** on the ribbon and then paste the agent access key into the dialog box that is displayed.

Next, click **Register** on the ribbon to register the database. Click the applicable type of authentication, SQL or Windows, for the on-premises database and supply the server and database names in the **SQL Server Configuration** dialog box.

Back in the Windows Azure management portal, click **SQL Databases** in the navigation pane, point to **Add Sync** on the ribbon, and then click **New Sync Group**. Provide a name, and select the region and subscription for the sync group on the first page of the wizard. On the second page of the wizard, select a database in the **Hub Database** drop-down list, and then provide credentials for a connection to this database. In addition, you need to specify whether the hub or the client wins in the event of a conflict. On the third page, you specify a reference database, credentials if it uses SQL authentication, and a sync direction. You can choose one of the following options for sync direction:

- Bi-Directional
- Sync to the Hub
- Sync from the Hub

You can define the tables and columns to synchronize by establishing sync rules. To do this, click **SQL Databases** in the Windows Azure management portal, click the **Sync** link at the top of the page, click the **Sync group**, click the **Sync Rules** link, and then click the **Define Sync Rules** link. Next select either the hub or reference database in the drop-down list. Then click the arrow next to a table to display the columns that you want to synchronize. Click **Select** on the ribbon at the bottom of the page to select all columns in all tables. Another option is to select a single column in a table and then click **Select** to select all columns from the current table. When you finish making your selections, click **Save**.

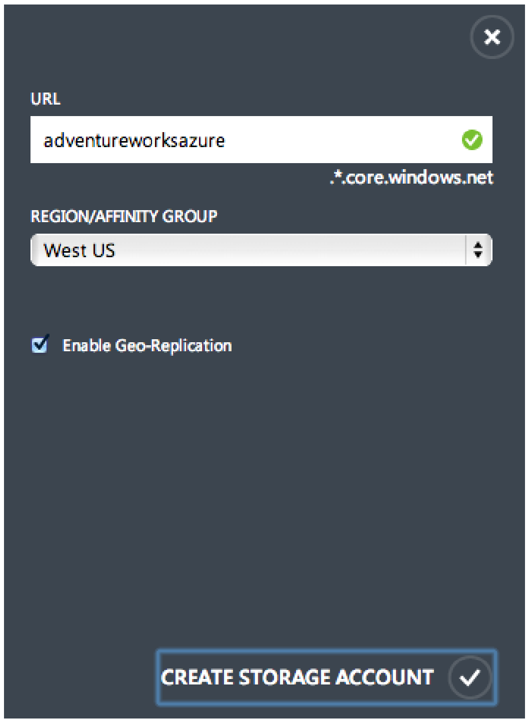
You can use the Windows Azure management portal to synchronize the databases on demand or on a schedule. Click **SQL Databases**, click the **Sync** link, and then click the **sync group**. For on-demand sync, click **Sync** on the ribbon. Otherwise, click **Configure**, and then click **On** for Automatic Sync. You can then set a **Sync Frequency** ranging from five minutes to one month. Be sure to click **Save** before exiting the page.

Data-Tier Application Export and Import

You can use the Data-Tier Application (DAC) export process to copy a database's schema definition and data into an export file (BACPAC). You then store this file in a Windows Azure blob storage account from which you use the DAC import process to migrate the data to a database in your SQL Database server. An advantage of using this approach is the ability to use source control and versioning to manage your database.

Windows Azure Blob Storage Account

Before you create the BACPAC file, you must create a blob storage account. In the Windows Azure Management Portal, click **New** in the lower left corner, point to **Data Services**, point to **Storage**, and then click **Quick Create**. In the URL box, type a subdomain name using a combination of lowercase letters and numbers with a minimum length of three and a maximum length of 24 characters. If the subdomain you type in is already in use on the Windows Azure servers, a warning message is displayed and prevents you from creating your account until you supply a unique name. When the name is valid, a green icon is displayed in the URL box, as shown in Figure 2.


URL: adventureworksazure .*.core.windows.net

REGION/AFFINITY GROUP: West US

☑ Enable Geo-Replication

Figure 2: Storage account creation

You must also select a data center location in the **Region/Affinity Group** drop-down list. The **Enable Geo-Replication** check box is selected by default. The **geo-replication** option ensures that your data persists in multiple locations, at no additional cost, to mitigate an outage in a single data center. Click **Create Storage Account** to start the process, which can take several minutes to complete.

After the account is created, you must next add a Blob storage container to it. On the **Storage** page of the Windows Azure Management portal, click the account in the list, then click the **Containers** link at the top of the dashboard page, and then click **Create a Blob Container**. To create a new container, you assign it a name that conforms to the following rules:

- Between 3 and 63 characters
- Contain only letters, numbers, or -
- Cannot start with - or use - twice in succession

You also choose one of the following access methods:

- Private. Only you as the account owner can use the container.
- Public Blob. Anyone can access the blobs in the container, but not the container properties and metadata.
- Public Container. Anyone has full access to the container.

When the container is successfully created, you will have a URL like this: http://adventureworksazure.blob.core.windows.net/database-export.

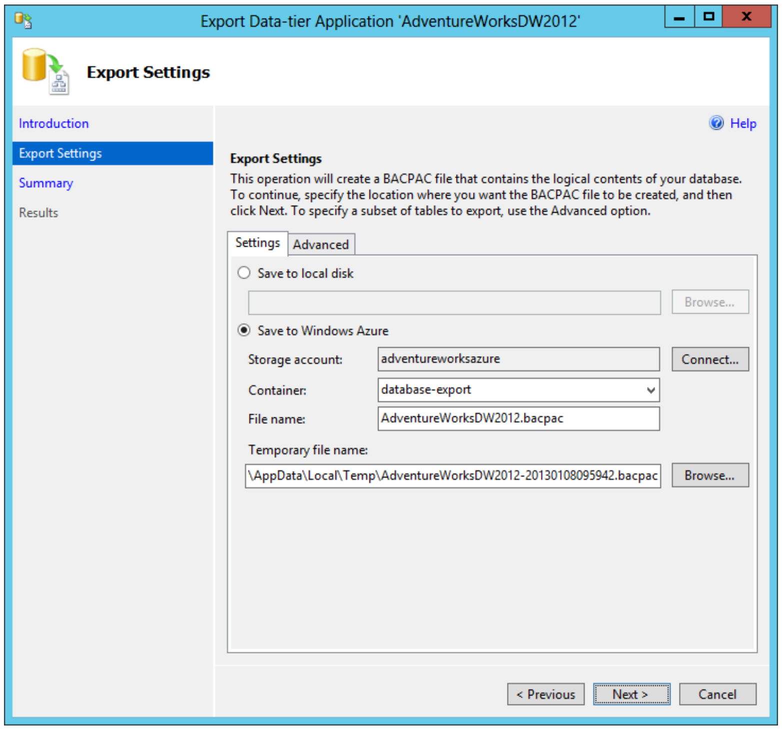
BACPAC Export

SQL Server Management Studio includes the Export Data-Tier Application Wizard to create a BACPAC file that you can upload to the blob storage in Windows Azure. Right-click the database in **Object Explorer**, point to **Tasks**, and click **Export Data-tier Application**. On the **Export Settings** page of the wizard, you have two options for saving the BACPAC file. One option is to save the file locally and then use a custom application or tools like Azure Storage Explorer or Cloud Storage Studio 2 to upload the file to your blob storage. The other option is to save the file to Windows Azure, in which case you need to click **Connect** to define the connection settings.

Before you can define the connection settings in the wizard, you need the **Storage Account** and **Account Key**. You can find these values by opening the dashboard page for the storage account in Windows Azure Management Portal. Click **Manage Keys** at the bottom of the page to view the storage account name, primary access key, and secondary access key. You can click a button next to each of these fields to place the value in your computer's clipboard and then paste the value into the corresponding field in the wizard.

Note: Be sure to paste the storage account first to clear the account key.

When you make a successful connection, you complete the **Export Settings** page of the wizard by selecting a container, as shown in Figure 3. You can set the file name and the temporary location for the file.


Export Data-tier Application 'AdventureWorksDW2012'

Export Settings

Introduction
Export Settings
Summary
Results

Help

Export Settings
This operation will create a BACPAC file that contains the logical contents of your database. To continue, specify the location where you want the BACPAC file to be created, and then click Next. To specify a subset of tables to export, use the Advanced option.

Settings | Advanced

○ Save to local disk

Browse...

● Save to Windows Azure

Storage account: adventureworksazure — Connect...

Container: database-export

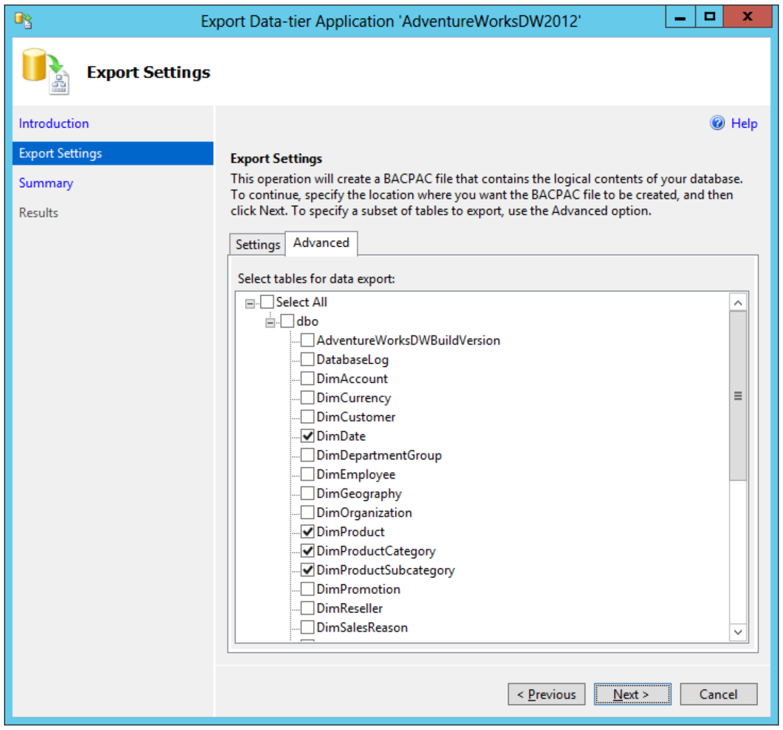
File name: AdventureWorksDW2012.bacpac

Temporary file name:
\AppData\Local\Temp\AdventureWorksDW2012-20130108095942.bacpac — Browse...

< Previous | Next > | Cancel

Figure 3: Export Data-tier Application Wizard

On the **Advanced** tab, you can select specific tables to export if you prefer not to migrate the entire database, as shown in Figure 4.


Export Data-tier Application 'AdventureWorksDW2012'

Export Settings

Introduction
Export Settings
Summary
Results

Help

Export Settings
This operation will create a BACPAC file that contains the logical contents of your database. To continue, specify the location where you want the BACPAC file to be created, and then click Next. To specify a subset of tables to export, use the Advanced option.

Settings | Advanced

Select tables for data export:

- ☐ Select All
  - ☐ dbo
    - ☐ AdventureWorksDWBuildVersion
    - ☐ DatabaseLog
    - ☐ DimAccount
    - ☐ DimCurrency
    - ☐ DimCustomer
    - ☑ DimDate
    - ☐ DimDepartmentGroup
    - ☐ DimEmployee
    - ☐ DimGeography
    - ☐ DimOrganization
    - ☑ DimProduct
    - ☑ DimProductCategory
    - ☑ DimProductSubcategory
    - ☐ DimPromotion
    - ☐ DimReseller
    - ☐ DimSalesReason

< Previous | Next > | Cancel

Figure 4: Individual table selection for export

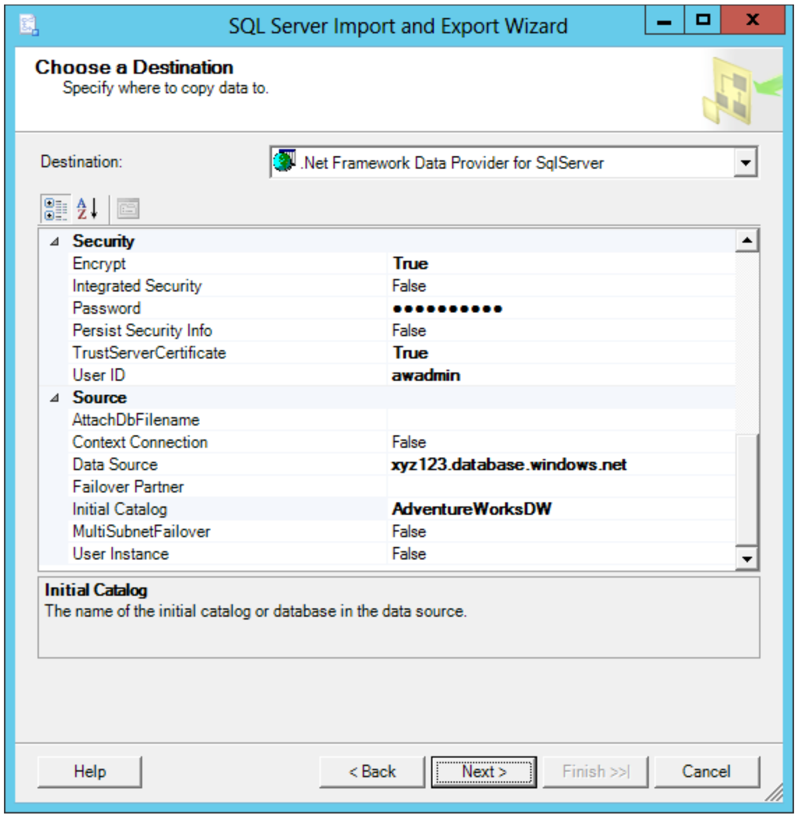
Migration into SQL Database

In the Windows Azure Management Portal, use the navigation pane to open the list of databases on your SQL Database server. Click **Import** on the ribbon at the bottom. You need to supply the URL for the BACPAC file. When you click the URL box, a dialog box opens to allow you to navigate the storage account and its containers to locate the BACPAC file.

You must name the database, select a database server or create a new one, and provide the login credentials. You have the option to select a check box to configure advanced database settings. Specifically, you set the edition and size of the database.

SQL Server Integration Services

You can use the SQL Server Import and Export Wizard in SQL Server Management Studio as a simple way to export data from your on-premises server to a SQL Database server. In **Object Explorer**, right-click the database, point to **Tasks**, and then click **Export Data** to launch the wizard. The **Choose a Data Source** page of the wizard automatically populates with the correct data source provider, server, authentication, and database selections. On the **Choose a Destination** page, select **.NET Framework Data Provider for SqlServer**. Then change the **Encrypt** property to **True**, type in the login password, set **TrustServerCertificate** to **True**, and type in the login User ID. The fully qualified name for the server and the name of the database goes into **Initial Catalog**, as shown in Figure 5.


SQL Server Import and Export Wizard
Choose a Destination
Specify where to copy data to.
Help
< Back
Next >
Finish >>|
Cancel

Destination: .Net Framework Data Provider for SqlServer

Security	
Encrypt	True
Integrated Security	False
Password	●●●●●●●●●●
Persist Security Info	False
TrustServerCertificate	True
User ID	awadmin
Source	
AttachDbFilename	
Context Connection	False
Data Source	xyz123.database.windows.net
Failover Partner	
Initial Catalog	AdventureWorksDW
MultiSubnetFailover	False
User Instance	False

Initial Catalog
The name of the initial catalog or database in the data source.

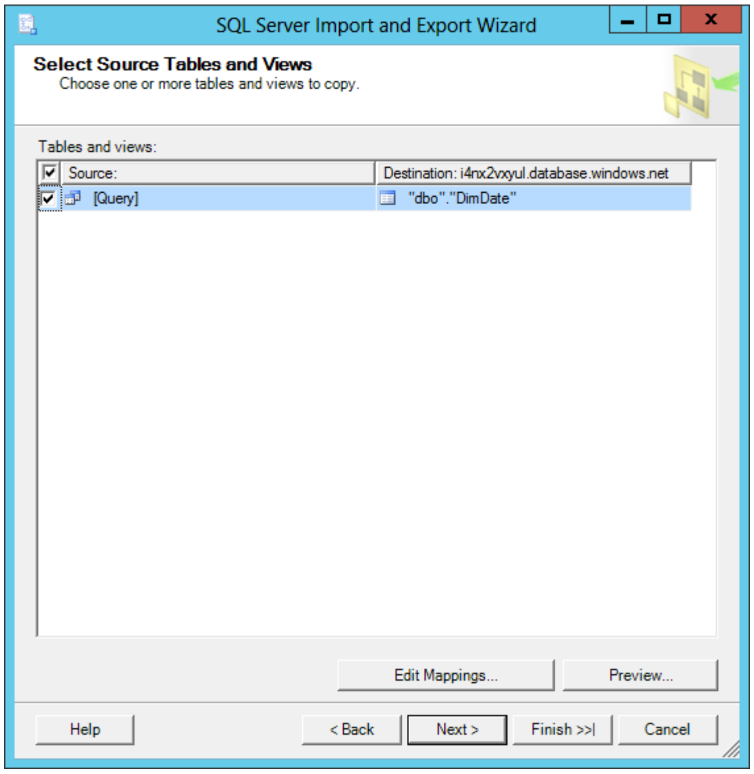
Figure 5: Properties for .NET Framework Data Provider for SqlServer

On the **Specify Table Copy or Query** page of the wizard, you have the option to select specific tables or write a query to define the data you want to migrate.

On the **Review Data Type Mapping** page, you might see warnings, but you can continue as long as no errors appear. You can then run the package built by the wizard. However, it's possible that the package will fail. In spite of this failure, the wizard copies the schemas for each table to the target database in SQL Database. For example, if the source tables do not have clustered indexes, the wizard fails. You can leave the wizard open and then add a clustered index by using the SQL Server Management Studio interface or by running a T-SQL command like this (for each table to migrate):

```
create clustered index idxDateKey on dbo.DimDate (DateKey)
```

After adding the clustered indexes, click **Back** to return to the **Specify Table Copy or Query** page and resume the wizard. You must switch to the query mode because the table copy mode creates tables in the target database during package execution, but the tables already exist if you try to run the wizard again after the initial failure. If you switch the wizard to query mode, you can write a SELECT statement to copy all or part of the data for a table, select the target table on the **Select Source Tables and Views** page, as shown in Figure 6, and then execute the wizard. You need to perform this set of steps for each table individually.


SQL Server Import and Export Wizard

Select Source Tables and Views
Choose one or more tables and views to copy.

Tables and views:

	Source:	Destination: i4nx2vxyul.database.windows.net
✓	[Query]	"dbo"."DimDate"

Figure 6: Mapping source query to destination table

Note: For a one-time migration, the wizard should suffice, but if you are regularly migrating databases to SQL Database, consider building a more robust Integration Services package. In this package, you can separate the steps of copying schemas, adding clustered indexes, and copying data.

Bulk Copy Utility

The SQL Server bcp utility is yet another way to move data, although it is limited to only data. The advantage of this approach is higher performance for high data volumes as compared to using single row inserts with the Database Script method. The bcp utility cannot migrate the database schema, so you must use a separate process to prepare your database in SQL Database. For example, you can use the Generate Scripts Wizard in SQL Server Management Studio to define the database objects you want to copy, and set the **Types of Data to Script** option to export the schema only. You run the bcp utility once to export data from an on-premises SQL Server to store the data in data files, and then run it again to transfer the data from the data files into an empty database on your SQL Database server.

Note: You must use the bcp utility from the client tools that install with SQL Server 2008 R2 or later.

Export from Source Database

Run the **bcp** utility for each table that you want to migrate. To export data from your SQL Server database, use a command like this in a **Command Prompt** window:

```
bcp AdventureWorksDW2012.dbo.FactResellerSales out c:\your
path\FactResellerSales.dat -S localhost -T -n -q
```

Note that the second argument is the database, schema, and table name. The fourth argument is the output path for storing the exported data file. You can replace the argument following the `-S` parameter with a server or database instance name if you are not running the bcp utility on the source server. The remaining parameters are optional. The `-n` parameter keeps the native database types of the data while the `-q` parameter executes the `SET QUOTED_IDENTIFIERS ON` statement when connecting to the source database.

Import into Target Database

Next run the bcp utility to import the data into the target database on your SQL Database server. Let’s say you have a table that you want to process in two steps. You can use the following commands to define the rows to process in each execution of the bcp utility:

```
bcp AdventureWorksDW.dbo.FactResellerSales in c:\your
path\FactResellerSales.dat –n –U awadmin –S tcp:xyz123.database.windows.net –
P YourPassword –b 500 –L 29999 –h"TABLOCK"


bcp AdventureWorksDW.dbo.FactResellerSales in c:\your
path\FactResellerSales.dat –n –U awadmin –S tcp:xyz123.database.windows.net –
P YourPassword –b 500 –F 30000 –L 61000 –h"TABLOCK"
```

The `-S` parameter now references the SQL Database server with `tcp:` as a prefix. Notice also the `-U` and `-P` parameters that require the login name and password you set up for your database. The `-b` parameter specifies the number of rows to process per batch during execution of the bcp utility. The `-F` and `-L` parameters specify the first and last rows in the file to process, although in the first bcp command the `-F` is omitted to indicate processing starts with the first row. Last, the `-h` parameter is a hint to apply a table-level lock during processing.

Note: If your data contains Unicode characters, be sure to add the –w parameter to ensure these characters are retained and not converted.

SQL Reporting Setup

At the time of this writing, you perform the setup of the SQL Reporting server on a separate portal. To access this portal, click your user name in the top right corner of the Windows Azure Management Portal and click **Previous Portal**. In the lower left corner of the old portal, click **Reporting** and then in the center of the browser window, click **Create a New SQL Reporting Server**. In the **Create Server** dialog box, select a subscription and a region for hosting the new server. To minimize latency issues, consider selecting the same region that you use for the SQL Database that you plan to use as a data source for reporting. You must also establish an administrator name and password to use for managing the report server.

In the left panel, you can expand the **Subscriptions** folder to view the subscriptions associated with your account. Expand the subscription to view your SQL Reporting servers. Click on a server to see its Web service URL and manage its contents, which you will learn how to do in Chapter 4, Report Management.

Chapter 3 Report Development

Now that you have data available in a SQL Database and a server set up to host reports in SQL Reporting, the next step is to develop reports. In this chapter, I introduce you to the tools you can use to create reports and explain the basic features that you can use in reports. In addition, I describe some differences you encounter if you decide to develop reports in Report Builder rather than BIDS or SSDT. Last, I show you how to work with the **ReportViewer** control if your goal is to develop a custom application for reporting.

Tools

The authoring process for SQL Azure Reporting uses the same on-premises tools that are available for SQL Server Reporting Services:

- Report Designer
- Report Builder
- ReportViewer control

If you already have Reporting Services in-house, then you can use **Report Designer** in Business Intelligence Development Studio (BIDS) if you are working with SQL 2008 R2 or SQL Server Data Tools (SSDT) if you are working with SQL Server 2012. If you do not have Reporting Services already, that's okay, too. You can download SQL Server 2012 Express with Advanced Services at no cost and use Report Designer in SQL Server Data Tools to develop reports.

Another option is to download **Report Builder**, also at no cost. Like BIDS and SSDT, it provides support for all Reporting Services features. The difference between these products is that Report Builder allows you to work with only one report at a time, whereas BIDS and SSDT allow you to work with multiple reports as part of a project. On the other hand, if you decide to implement report parts, as I explain how to do later in this chapter, you must use Report Builder to create new reports based on those report parts.

One more option is to use the **ReportViewer** control in Visual Studio as part of a custom Windows or Web application. Just like BIDS, SSDT, and Report Builder, you can create an interactive, dynamic, and visually appealing report using a variety of layout options.

Report Development Fundamentals

Report development is a multistep process. In this section, I describe how to use the Report Designer. Most of the functionality in Report Builder and ReportViewer is similar, but I explain the differences in those tools later in this chapter should you need to use one of those tools instead. After you select a tool, your next step is to acquire data for your report, so you create data sources and datasets to define where the data is and what you want to retrieve. Once you have the data, you can design the report layout. During this step, you specify how the data is structured in the report, add titles, header and footer text, and images. You apply formatting and configure any interactive features to control not only the look and feel of the report, but also the behavior of the report.

As you work, you can preview the report to ensure you get the desired results. When you finish, you deploy the report to SQL Reporting. Because report development is a very iterative process, you are likely to preview the report and then return to working on the report layout to fine-tune the report design multiple times until you get the results you want.

Report Projects

As already mentioned, the report designer is the report development tool of choice for most professional report developers. The advantage of working with it rather than Report Builder is the ability to work with multiple reports as part of a project. When you create a new project, you can choose to create one of the following project types:

- Report Server Project Wizard
- Report Server Project

For quick development of a basic report, you can use the Report Server Project Wizard. You use the wizard interface to step through the process of defining the data source, defining a query, and selecting a layout from a few options. Because you can make modifications to a report after using the wizard, it’s an easy way to start simple reports.

The other project type is the Report Server Project. This is the project type that you select most frequently for report development. In this chapter, I use the Report Server Project type to explain report development fundamentals.

Project Items

After you create a project, you then add items to it. As you add items, they are visible in the **Solution Explorer** window, as shown in Figure 7. The report server project supports three different types of items:

- Data Source
- Dataset
- Report

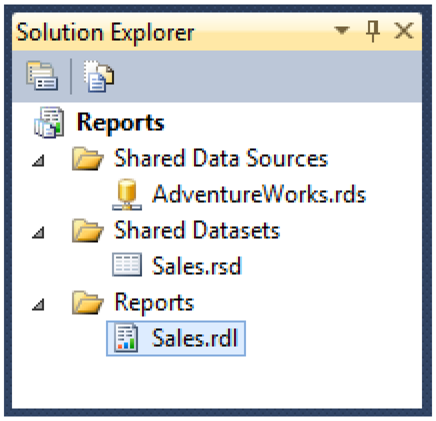


Figure 7: Report project items in Solution Explorer

The **data source** is stored as an RDS file in the project directory on your computer and is uploaded separately to SQL Reporting when you deploy the project. You can create one or more data sources for use with a report. When a data source is created independently of a report, it is known as a **shared data source** and available for you to use with any report. That way, any changes to data source settings can be made once for multiple reports.

The **dataset** is stored as an RSD file, and is also uploaded separately. Like data sources, you can use one or more datasets with any report and a change made to the shared dataset affects all reports.

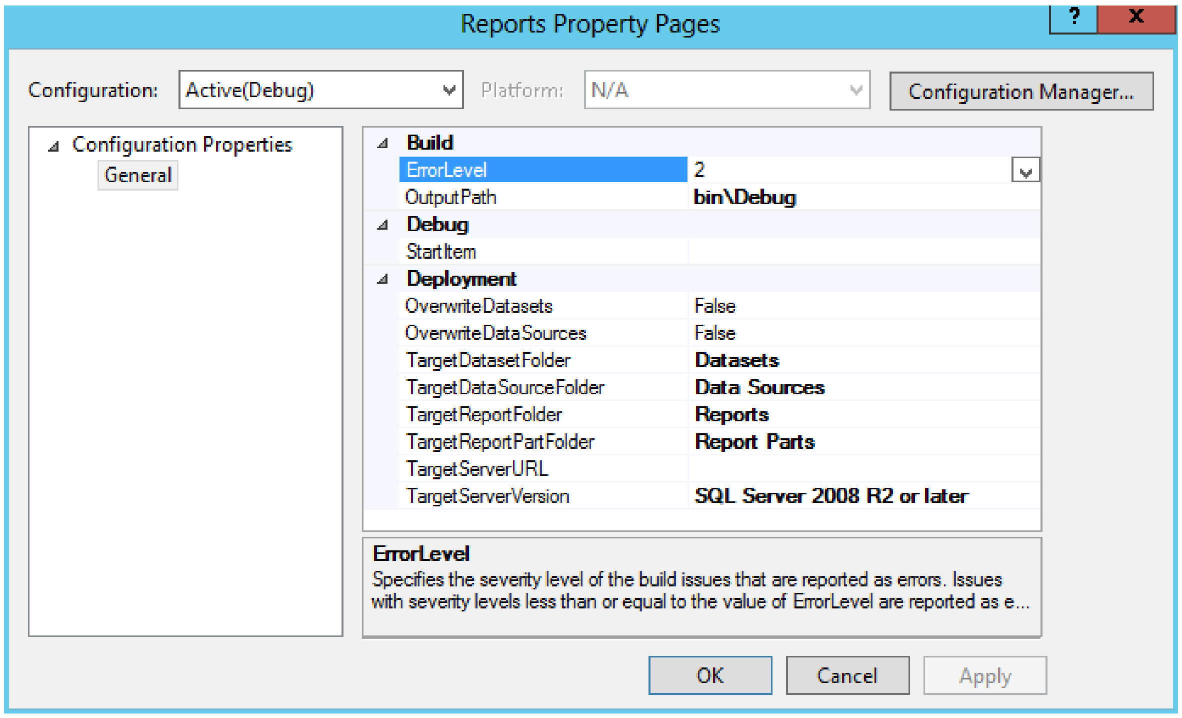
The third item type is the **report,** which is stored as an RDL file. Although it's possible to have a project without data source and dataset files, you will always have at least one report file in your project.

To add a project item, right-click on the applicable folder in **Solution Explorer**, and use the applicable command, such as **Add New Data Source** or **Add New Dataset**. However, if you choose **Add New Report** from the **Reports** folder, you launch a wizard. To bypass the wizard and create a blank report, point to **Add** after you right-click the **Reports** folder and select **New Item**. In the **Add New Item** dialog box, select **Report** and replace the default name, such as **Report1.rdl**, with a more suitable name. Be sure to retain the RDL file extension in the name.

Project Properties

You must configure project properties before deployment if you plan to use Report Designer to deploy your reports to SQL Reporting. These properties are used only by the **Deploy** command. At deployment, these properties identify target locations for your various project files on the SQL Reporting server. Many of the project properties have default values and refer to folders to use for storing project items. For example, your shared datasets go to the **Datasets** folder, and shared data sources go to the **Data Sources** folder. You can change these target folders to an alternate path if you like.

Right-click the project name in **Solution Explorer** and select **Properties** to open the project properties, shown in Figure 8. Notice the **TargetReportFolder** defaults to the name of the current project, which in this case is **Reports**. You might prefer a different folder name for your reports than the project name, so be sure to change this value if necessary. Also, you must specify the **TargetServerURL** for the report server before you can deploy reports. It will be empty in a new project, and the project won't deploy until you provide a value for this property. Use the Web service URL that is displayed in the SQL Reporting portal, such as https://xyz123.reporting.windows.net/ReportServer.



Build	
ErrorLevel	2
OutputPath	bin\Debug
Debug	
StartItem	
Deployment	
OverwriteDatasets	False
OverwriteDataSources	False
TargetDatasetFolder	Datasets
TargetDataSourceFolder	Data Sources
TargetReportFolder	Reports
TargetReportPartFolder	Report Parts
TargetServerURL	
TargetServerVersion	SQL Server 2008 R2 or later

ErrorLevel
Specifies the severity level of the build issues that are reported as errors. Issues with severity levels less than or equal to the value of ErrorLevel are reported as e...

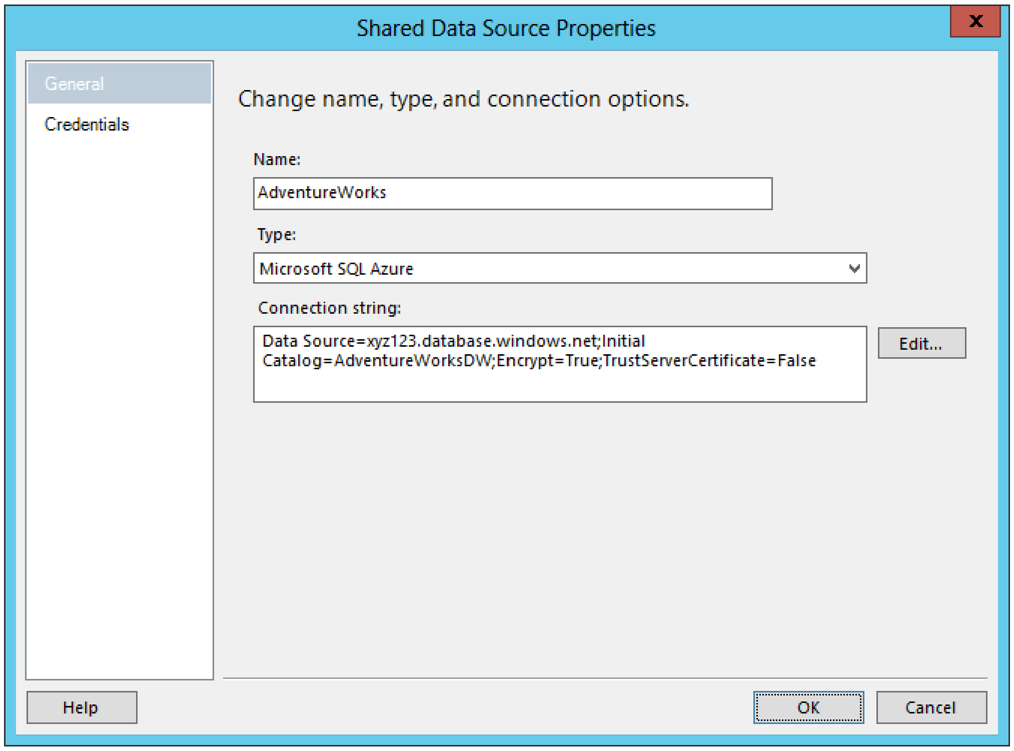
Figure 8: Report project properties

Data for Reports

As part of the data acquisition process, you define data sources to provide instructions to the SQL Reporting server about where to locate data to be used in the reports and how to authenticate the requests for the data. You also need to create datasets to define the columns to retrieve from the data sources for placement in your report.

Data Sources

To display data in your report, you need at least one data source. A Reporting Services data source includes a data provider, a connection string, and credentials to use for authentication. Because SQL Reporting only supports SQL Database as a source, you must select **Microsoft SQL Azure** in the **Type** drop-down list, as shown in Figure 9. The example in Figure 9 illustrates a shared data source, which you create by right-clicking the **Shared Data Sources** folder in **Solution Explorer**, but you can also create an embedded data source in the **Report Data** pane as I explain later in this chapter.



Change name, type, and connection options.

Name: AdventureWorks

Type: Microsoft SQL Azure

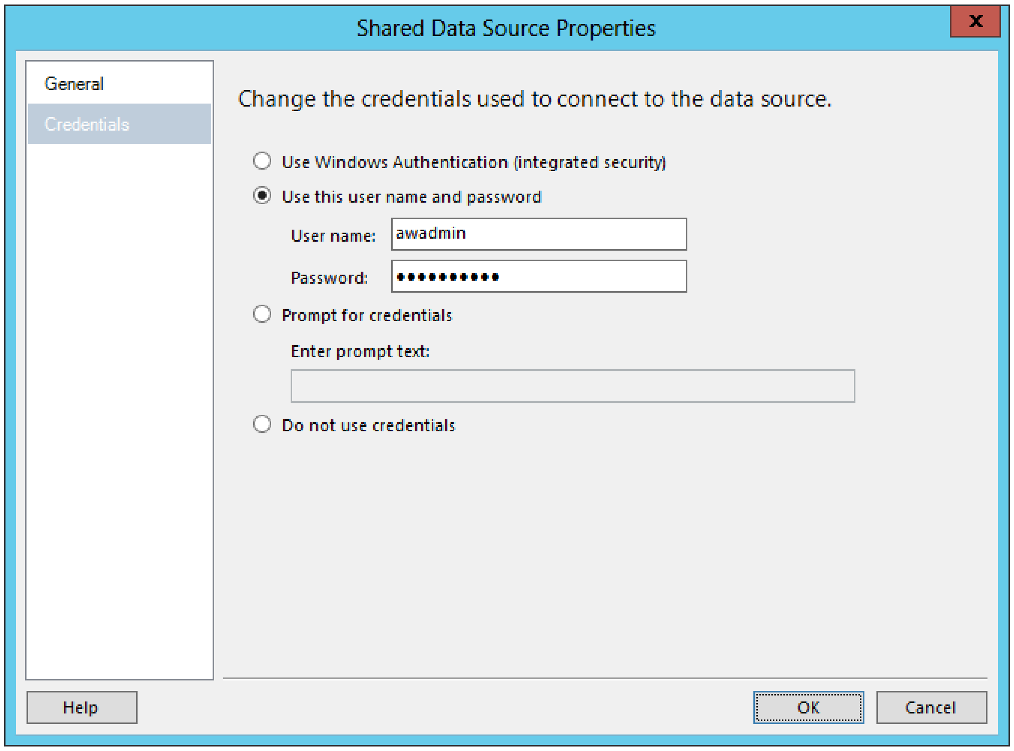
Connection string:
Data Source=xyz123.database.windows.net;Initial Catalog=AdventureWorksDW;Encrypt=True;TrustServerCertificate=False

Figure 9: Shared data source for SQL Database

The format of the connection string to SQL Database looks like this:

```
Data Source=xyz123.database.windows.net;Initial
Catalog=AdventureWorksDW;Encrypt=True;TrustServerCertificate=False
```

You enter the login name and password on the **Credentials** tab of the **Shared Data Source Properties** dialog box, shown in Figure 10.



Change the credentials used to connect to the data source.

- ( ) Use Windows Authentication (integrated security)
- (●) Use this user name and password

User name: awadmin

Password: ••••••••••

- ( ) Prompt for credentials

Enter prompt text:

- ( ) Do not use credentials

Figure 10: Credentials for data source connection

There are two different types of data sources:

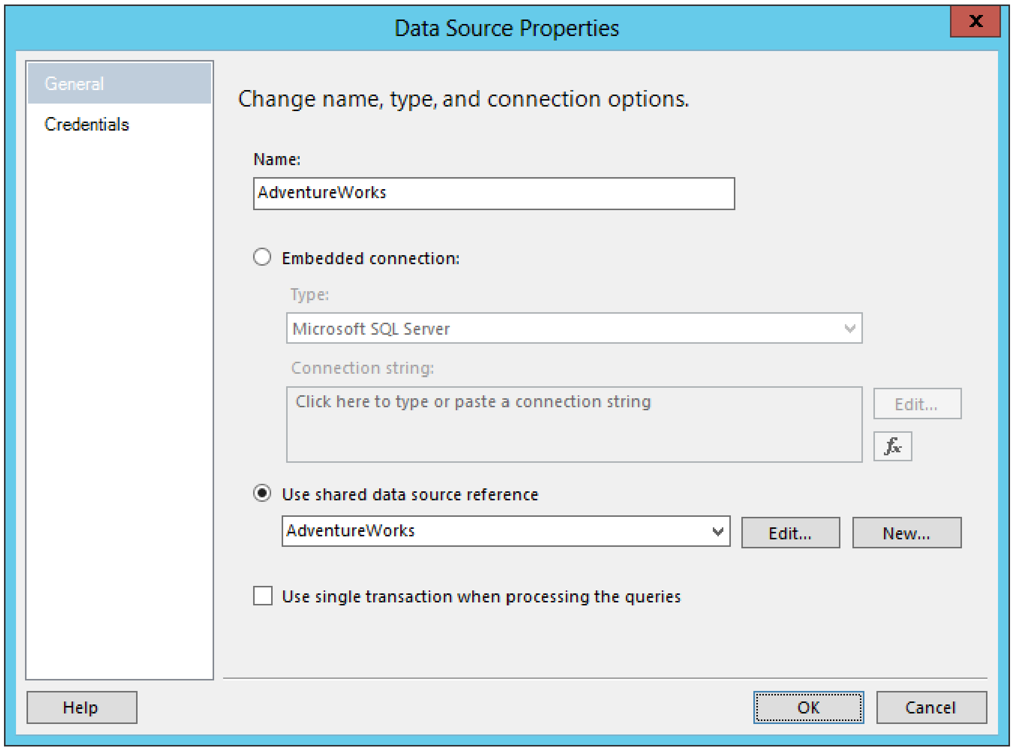
- Shared data source
- Embedded data source

Typically, you create a **shared data source** to define the connection information once for use with many reports. If connection information changes later, such as when you move a source database to a different SQL Database server, you can update the data source definition once to update all associated reports.

Your other option is an **embedded data source**. In this case, you don't create a separate data source file for the project, but instead create a data source definition inside of each report's RDL. The advantage of using an embedded data source is that you can use dynamic connection strings. The disadvantage is that if you need to make a change to the connection string, you must open every report and edit each connection string, which can be a very tedious process if you have a lot of reports.

Note: If you decide to use an embedded data source, be aware that the only way you can change credentials is to open the report in the report designer or Report Builder, edit the credentials, and then redeploy the report. The management portal does not support changes to embedded data sources.

Regardless of the type of data source you decide to use, you must add a reference to the data source in your report. To do this, open the report and then, in the **Report Data** pane, right-click the **Data Source** folder and click **Add Data Source**. Here you can choose the **Embedded Connection** option and then define the connection string and credentials, or the **Shared Data Source Reference** option and then select the applicable data source in the drop-down list, as shown in Figure 11.



Change name, type, and connection options.

Name: AdventureWorks

Embedded connection:

Type: Microsoft SQL Server

Connection string: Click here to type or paste a connection string

(●) Use shared data source reference: AdventureWorks

[ ] Use single transaction when processing the queries

Figure 11: Shared data source reference added to report

Tip: Consider naming the data source to match the shared data source name or the name of the database in an embedded connection to make the data source more easily identifiable in your report.

Datasets

After creating a data source definition, your next step in the report development process is to create a dataset. In simple terms, a dataset is the query used to get data for the report. However, a closer look reveals that it contains a description of the fields or table columns to retrieve from the data source, among other information. During report development, you use these collections of fields as placeholders for data in your report.

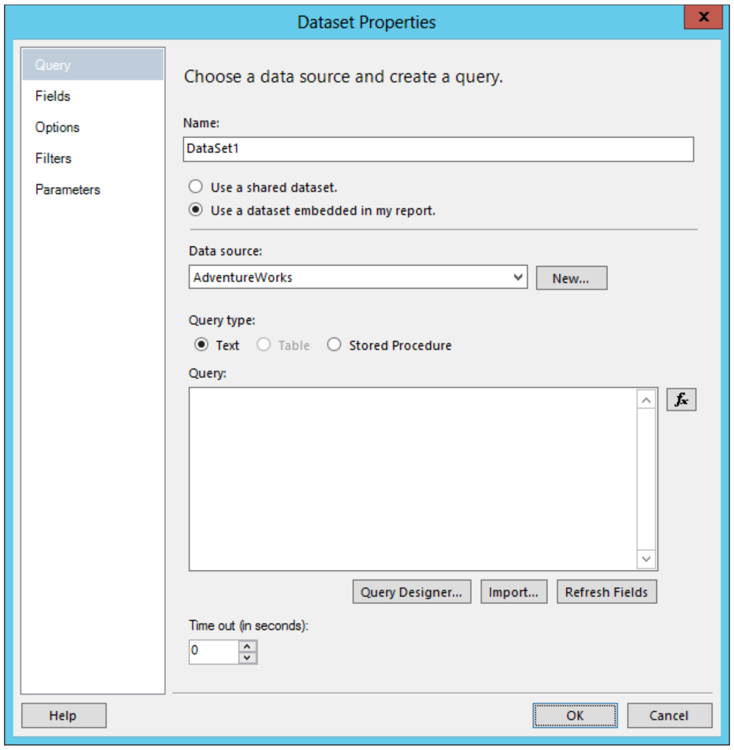
There are two types of datasets you can create:

- Shared dataset
- Embedded dataset

If you need to use the same query repeatedly with many reports, then a **shared dataset** is the better option, much like you can use a shared data source with many reports. That way, you can make any necessary changes to one file only. When you create a shared dataset, the output is an RSD file that is added to your project.

An **embedded dataset**, on the other hand, is associated with one report only. The dataset definition, although similar to the contents of an RSD file, is embedded directly within the report's RDL file.

Whether you use a shared or embedded dataset, you must include it in your report. To do this, open the report and then, in the **Report Data** pane, right-click the **Dataset** folder and click **Add Dataset**. Here you can choose the **Embedded Connection** option as shown in Figure 12, and then define the query type and query string, or select the shared dataset option and then select the applicable dataset in the drop-down list.



Choose a data source and create a query.

Name: DataSet1

Use a shared dataset. [✓ Use a dataset embedded in my report.]

Data source: AdventureWorks

Query type: [✓ Text] Table Stored Procedure

Query:

Time out (in seconds): 0

Figure 12: Report dataset properties

If you select **Text** as the **Query Type** option, you can type the query string directly into the query box or click the **Query Designer** to open a simple query designer. Even if you open the query designer, you can click a button to switch to a text editor. Either way, you will find it helpful to open this separate dialog box because it allows you to execute a query to confirm both that it's valid and that you get the results you need. At this point, the **Fields** collection for your dataset is generated to match the data columns that you see in the query results.

You can also use the **Stored Procedure** option for the **Query Type**. When you choose this option, the query box is replaced with a drop-down list from which you select one of the available stored procedures.

As an alternative to providing the query text or selecting a stored procedure, you can click **Import** and locate a SQL script file or an existing RDL. With this option, you can reuse an existing query by importing it from another file. However, any changes made to the original query will not update the dataset into which you imported it.

Another important property to configure in the dataset is the **Time Out** property. By default, it is set to 0, which means the query will wait indefinitely for the data source to return data. If you want to put a time limit on the query, and fail the report if the data source is taking too long, you can put in the maximum number of seconds that you want to wait as the time-out value.

Report Items

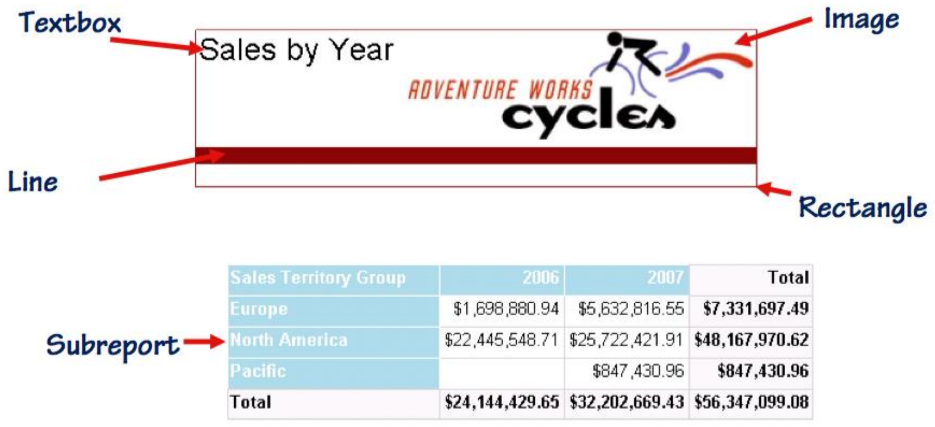
You spend the majority of your report development time working with report items in the design layout. Report items are the objects that you position on reports, such as text boxes and images. Some report items are independent of data, while others, known as data regions, require an association with a dataset in the report.

To add a report item to your report, open the **Toolbox** window and drag the item to a specific position on the report. You can drag edges of the item to resize it if necessary, or select the item and then set the **Location** and **Size** properties in the **Properties** window for greater precision. Of course, there are other properties available to configure, but these vary by report item type.

Independent Report Items

Each of the following independent report items is shown in Figure 13:

- **Text Box**. Use for report titles or freestanding labels.
- **Line**. Create visual separation between items on a report.
- **Rectangle**. Put a border around one or more objects or use as a container to keep objects together on a page.
- **Image**. Display a logo on a report, use as a watermark on a page, or display in a data region if the image is part of a dataset, such as product images stored with product data.
- **Subreport**. Build a separate report and then embed it in other reports as a subreport.



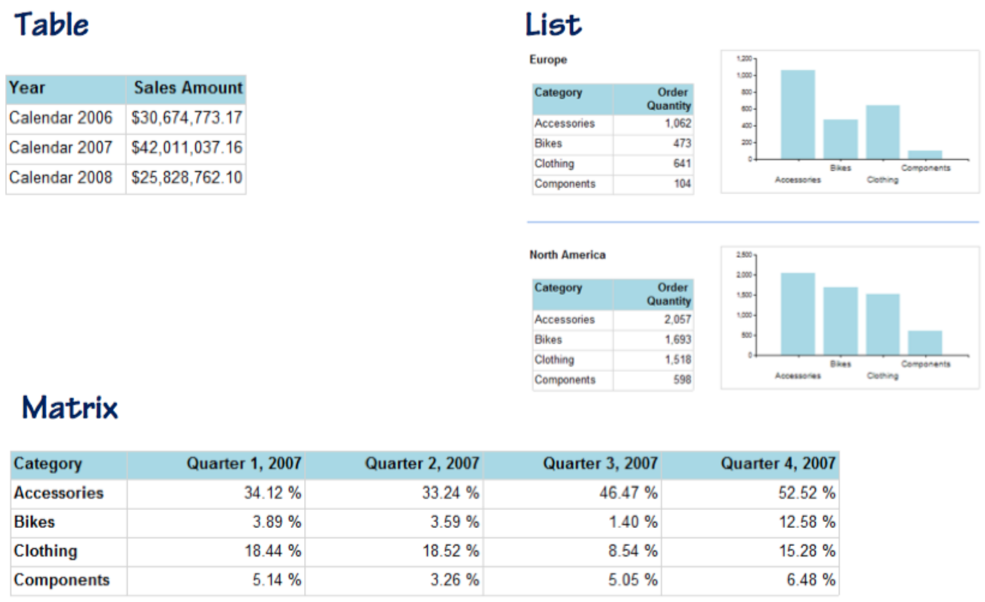
Sales Territory Group	2006	2007	Total
Europe	$1,698,880.94	$5,632,816.55	**$7,331,697.49**
North America	$22,445,548.71	$25,722,421.91	**$48,167,970.62**
Pacific		$847,430.96	**$847,430.96**
Total	**$24,144,429.65**	**$32,202,669.43**	**$56,347,099.08**

Figure 13: Independent report items

Data Regions

The following data regions, shown in Figure 14, present the data just as you see it in the query results with optional summarization:

- **Table**. A table has a fixed number of columns. The number of rows will vary according to your query results. Although this table doesn't show it, you can also add grouping and subtotals to a table.
- **Matrix**. A matrix has a variable number of rows and columns. You can also add grouping and subtotals to a matrix layout.
- **List**. A list is a free-form area that repeats its contents according to a grouping that you've defined. In Figure 14, you can see a table and a chart inside of a list, grouped by sales territory. In other words, there is one instance of the table and chart for Europe and another instance of the table and chart for North America due to the grouping configured for the list.



Table

Year	Sales Amount
Calendar 2006	$30,674,773.17
Calendar 2007	$42,011,037.16
Calendar 2008	$25,828,762.10

List

Europe

Category	Order Quantity
Accessories	1,062
Bikes	473
Clothing	641
Components	104

North America

Category	Order Quantity
Accessories	2,057
Bikes	1,693
Clothing	1,518
Components	598

Matrix

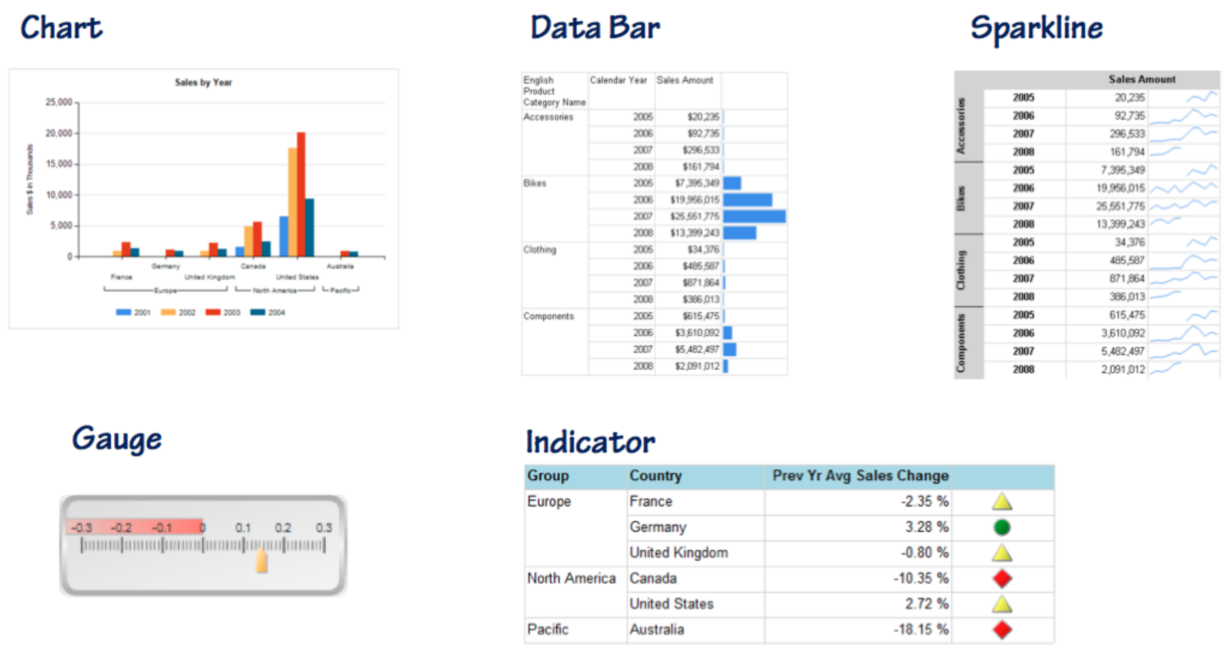
Category	Quarter 1, 2007	Quarter 2, 2007	Quarter 3, 2007	Quarter 4, 2007
Accessories	34.12 %	33.24 %	46.47 %	52.52 %
Bikes	3.89 %	3.59 %	1.40 %	12.58 %
Clothing	18.44 %	18.52 %	8.54 %	15.28 %
Components	5.14 %	3.26 %	5.05 %	6.48 %

Figure 14: Data regions

- **Chart**. Many different types of charts are supported, such as column, bar, pie, and scatter charts, to name a few. You have the ability to configure the properties of each chart element, such as the axis titles; whether to include a legend and how it appears; colors for the series; and so on.
- **Data bar**. Although a data bar is not required to be placed inside a data region, more often it is added to a table or matrix as an inline chart to display a single data point in chart form.
- **Sparkline**. A sparkline is similar to a data bar in that it is commonly used as an inline chart. However, the sparkline plots multiple data points, such as sales by month of year.

- **Gauge**. A gauge is useful for visual comparisons of a value, as shown by a pointer, to a range such as a red zone.
- **Indicator**. You can use an indicator as another way to compare values to goals, using a variety of shapes or colors, and even your own images if you prefer not to use the available icons.

Graphical data regions use data visualization techniques to summarize the data for comparative purposes or pattern discovery, as shown in Figure 15.



English Product Category Name	Calendar Year	Sales Amount	
Accessories	2005	$20,235	
	2006	$92,735	
	2007	$296,533	
	2008	$161,794	
Bikes	2005	$7,396,349	
	2006	$19,956,015	
	2007	$25,551,775	
	2008	$13,399,243	
Clothing	2005	$34,376	
	2006	$485,587	
	2007	$871,864	
	2008	$386,013	
Components	2005	$615,475	
	2006	$3,610,092	
	2007	$5,482,497	
	2008	$2,091,012	

		Sales Amount	
Accessories	2005	20,235	
	2006	92,735	
	2007	296,533	
	2008	161,794	
Bikes	2005	7,395,349	
	2006	19,956,015	
	2007	25,551,775	
	2008	13,399,243	
Clothing	2005	34,376	
	2006	485,587	
	2007	871,864	
	2008	386,013	
Components	2005	615,475	
	2006	3,610,092	
	2007	5,482,497	
	2008	2,091,012	

Group	Country	Prev Yr Avg Sales Change	
Europe	France	-2.35 %	▲
	Germany	3.28 %	●
	United Kingdom	-0.80 %	▲
North America	Canada	-10.35 %	◆
	United States	2.72 %	▲
Pacific	Australia	-18.15 %	◆

Figure 15: Graphical data regions

Another special type of data region is the map report item that you use for spatial data visualization, as shown in Figure 16. You can create a map containing any or all of the following types of spatial data:

- **Point**. Each point represents a coordinate on a map. You can optionally use Bing integration to bring in street-level detail.
- **Line**. Lines are pairs of points. As an example, you might use multiple lines to show routing information.
- **Polygon**. A polygon is a series of points for which the first and last points are the same.

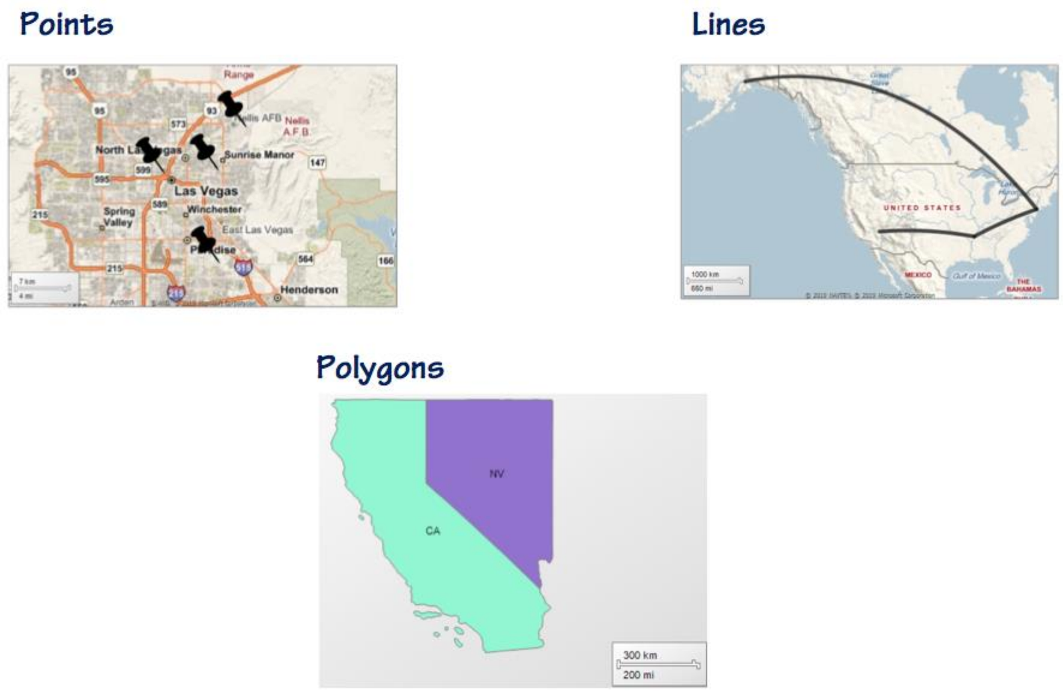


Figure 16: Spatial data types in maps

Now let's take a closer look at working with data regions. Although the data regions are listed separately in the **Toolbox** window as **Table**, **Matrix**, and **List**, they are actually the same structure, known as **Tablix**. The difference between them lies in the definition of detail rows, row groups, and column groups.

On the other hand, the placement of fields in each type of data region is similar. You can drag a field from the dataset list in the **Report Data** pane and drop it into a text box in the data region. Another way to place fields is to point to a text box, click the **Field List** icon that is displayed, and select a field.

When working with data regions, you can also group data. In fact, the matrix allows you to work only with groups instead of detail rows. You can drag the field to use for grouping from the **Report Data** pane to the **Row Groups** or **Columns** group pane that is displayed at the bottom of the report designer, as shown in Figure 17.

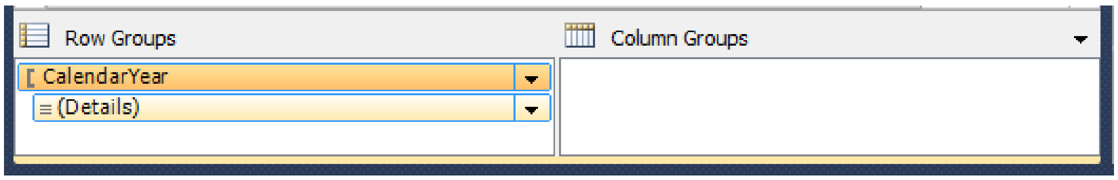
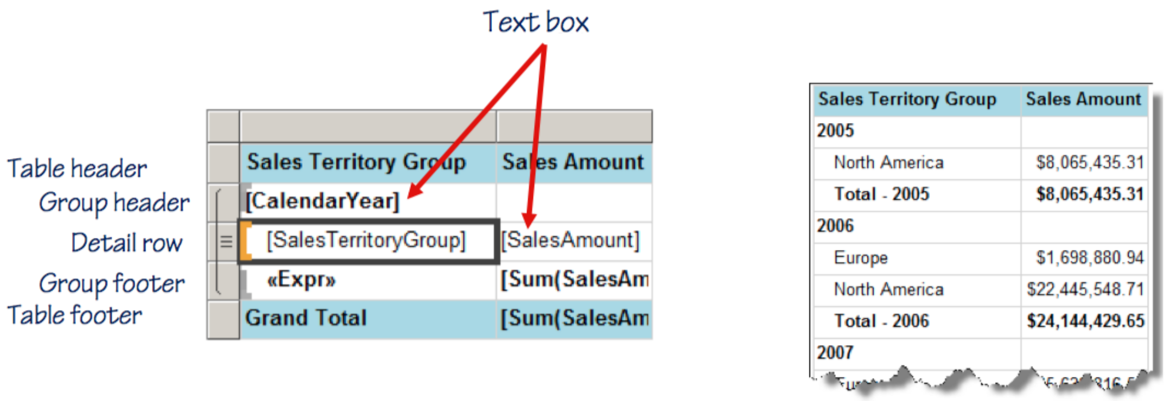


Figure 17: Groups

Table

A table is simply a tabular collection of text boxes. The number of columns it contains always remains static, but the number of rows in the table will vary by the number of rows that are returned by the query. These are called the detail rows, as shown in Figure 18. When you work in preview mode, you can see the detail row in the center of the table data region with three horizontal lines in the gray row handle on the left edge of the table.



Sales Territory Group	Sales Amount
[CalendarYear]	
[SalesTerritoryGroup]	[SalesAmount]
«Expr»	[Sum(SalesAm
Grand Total	[Sum(SalesAm

Sales Territory Group	Sales Amount
2005	
North America	$8,065,435.31
Total - 2005	$8,065,435.31
2006	
Europe	$1,698,880.94
North America	$22,445,548.71
Total - 2006	$24,144,429.65
2007	

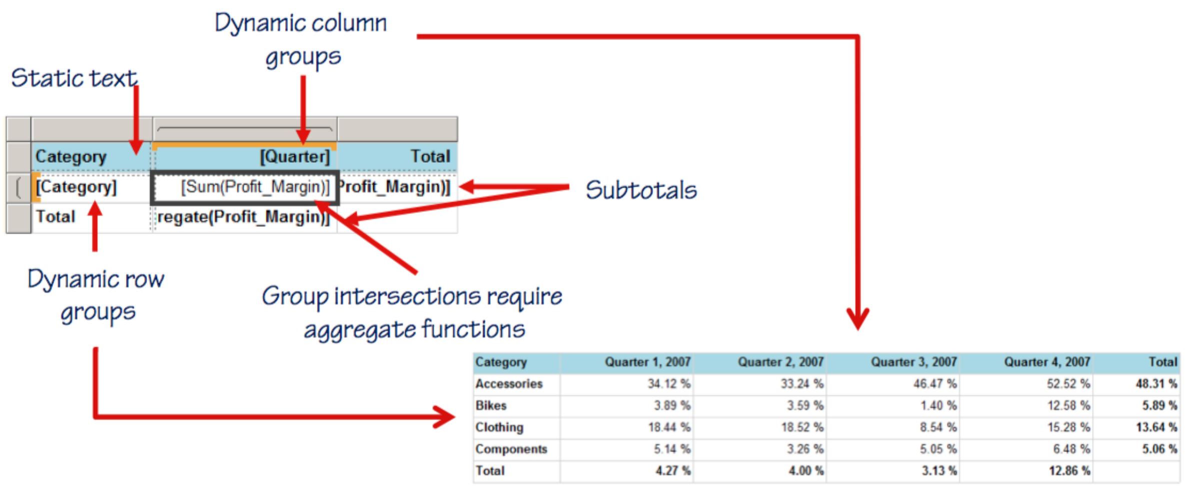
Figure 18: Table layout

For each row in your query results, a corresponding detail row appears in the rendered table when you switch to the **Preview** tab. Optionally, you can include header and footer rows for the table and for groups. In fact, you always have a table header row at the top of the table, unless you decide to delete it, and it appears only once. A table footer is an optional addition to your table, and like the table header, it appears only once after all detail rows and any group rows have rendered.

If you add a group to the table, you can include a group header, which appears once for each instance of the group before the detail rows belonging to that instance. Similarly, we can add a group footer row, which is displayed after the detail rows belonging to the respective group. In the example shown in Figure 18, the detail rows group by year, with a group header displaying the year and a group footer displaying the subtotal of sales for the year.

Matrix

A matrix is a crosstab collection of text boxes displayed in a dynamic arrangement of columns and rows, like a pivot table in Excel. The arrangement is dynamic because the number of columns and rows can vary in each rendering of the matrix when your dataset returns different results, either because the data has changed in the source, or because you have applied a filter. For dynamic row groups, the matrix will display one row per instance of the group defined. Similarly, for dynamic column groups, you'll get one column per instance defined for column groups. The matrix shown in Figure 19 has only one group on columns and one group on rows, but you can create multiple groupings on both columns and rows.



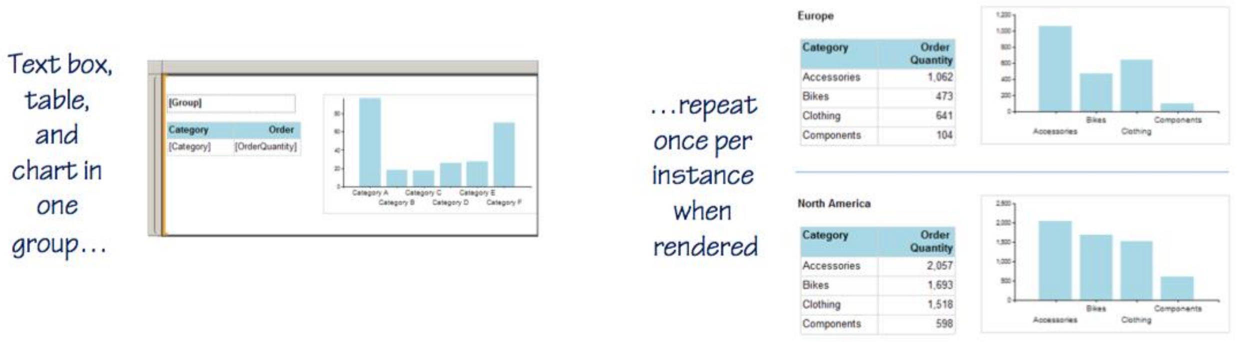
Category	Quarter 1, 2007	Quarter 2, 2007	Quarter 3, 2007	Quarter 4, 2007	Total
Accessories	34.12 %	33.24 %	46.47 %	52.52 %	**48.31 %**
Bikes	3.89 %	3.59 %	1.40 %	12.58 %	**5.89 %**
Clothing	18.44 %	18.52 %	8.54 %	15.28 %	**13.64 %**
Components	5.14 %	3.26 %	5.05 %	6.48 %	**5.06 %**
Total	**4.27 %**	**4.00 %**	**3.13 %**	**12.86 %**	

Figure 19: Matrix layout

When working with a matrix, you must use an aggregate function with the field you place in the text box at the intersection of a row and column group. Incidentally, the text box in the top left corner of the matrix can contain static text, but its use is optional. Another optional feature is the addition of subtotal rows or columns by group.

List

The list is a freeform collection of any report item that you want to repeat as part of a group. This collection is usually a combination of text boxes and data regions. A list supports one group only, but you can put a list inside of a list if you need to create a freeform layout with multiple groupings. Figure 20 illustrates a list that contains a text box, a table, and a matrix that are all part of one group. In the rendered report, each combination of text box, table, and matrix repeats once per instance of the defined group, which in this example is sales territory.



Europe

Category	Order Quantity
Accessories	1,062
Bikes	473
Clothing	641
Components	104

North America

Category	Order Quantity
Accessories	2,057
Bikes	1,693
Clothing	1,518
Components	598

Figure 20: List layout

Expressions

Expressions provide added flexibility in your reports by allowing you to modify data from your dataset or to adjust the setting of a property at the time of report execution. In this section, I start with an overview of the possibilities that expressions enable, explain how you can use placeholders in simple expressions to more easily interpret the report design layout, and review the global collections that you can reference in expressions. I also show you how to use the Expression Editor to create more complex expressions and explore some of the functions that you can use in expressions.

Expression Usage

There are two types of expressions in Reporting Services:

- **Simple**. A simple expression references an object and usually a property of that object. For example, placement of a field into a text box automatically generates an expression to display the **Value** property of a particular field in the **Fields** collection of a dataset looks like this:

```
=Fields!CalendarYear.Value
```

- **Complex**. Complex expressions perform a calculation using Visual Basic .NET syntax. For example, an expression that returns a total of the value of a field for each detail row in a dataset looks like this:

```
=Sum(Fields!SalesAmount.Value)
```

As you develop reports, you are likely to use expressions in many different ways:

- **Dynamic connection string**. Although this is not the most common way to use expressions, this approach does allow you to generate a connection string at run time so that you can specify the conditions under which one connection string might be used versus another. However, the downside of using dynamic connection strings is that you can use it only with an embedded data source, which means you could have some administrative overhead when you need to make changes to the expression.
- **Dynamic dataset**. You can use an expression to create a query string at run time, perhaps based on parameters provided in the report. A dynamic dataset might use completely different sources in the FROM clause or modify the JOINS to the query in different ways depending on the conditions at runtime.
- **Calculated field**. You can create scalar calculations with expressions. A scalar calculation resolves as a separate value for each row in the dataset. The calculated field is displayed in the **Report Data** pane along with fields generated by the dataset. You can then add a calculated field to a data region as if the field came from the data source. That way, when you use the same calculation multiple times in your report, the definition is stored in one place and you can reference the field as many times as needed. Furthermore, if you need to make a change to the expression, you have only one place to make the change.

Tip: Generally it's better to put a scalar expression in the query. However, sometimes you might not have access to the query directly, such as when you use a stored procedure in your dataset.

- **Text box calculation**. The most common reason to use expressions is to use the results of a calculation to display a value in a text box.
- **Filter**. The use of an expression in a filter is often used in combination with a parameter, although a filter does not require a parameter. For example, you can use a filter expression that calculates today's date and then filters the data from a query to display only values for today.

Placeholders

There are many expressions that you use in reports that are considered simple expressions. Simple expressions refer to a single item and display in a text box in design mode as a placeholder so you can easily see what the text box contains during report development. Prior to SQL Server 2008, only a portion of the expression in a text box was visible. To see the entire expression, you were required to open the Expression Editor. Now you can use placeholders with the following collections:

- **Fields**. The Fields collection is part of a dataset and stores the values retrieved by a query. When you drag and drop a field from the **Report Data** pane into a text box, the report designer automatically creates a placeholder that includes the field name and encloses it in brackets. In other words, when you add the **CalendarYear** field to a text box, you see `[CalendarYear]` display in the text box instead of the actual expression stored in the RDL, `=Fields!CalendarYear.Value`. You can also use placeholders with aggregate functions, which the report designer creates when you use the **Add Total** command in a tablix.
- **Parameters**. The Parameters collection allows you to reference report parameter selections in an expression. I explain more about parameters later in this chapter – specifically, report parameters. If your report includes a report parameter named BusinessType, you can refer to it by using a placeholder that uses the `@` symbol as a prefix like this: `[@BusinessType]`.
- **Globals**. The Globals collection contains a variety of fields that are commonly used in reports, such as execution time or page number. You use the **&** symbol as a prefix like this: `[&PageNumber]`.

Any other type of expression is a complex expression, and the report designer displays a generic placeholder, `<<Expr>>`. The only way for you to know what it represents is to open the Expression Editor to view the underlying expression.

Expression Editor

It's usually not practical to type expressions directly into a text box or to set a property value. Instead you use the Expression Editor to use as a graphical interface. You can right-click a text box and click **Expressions** to open the editor, click the **Expression** button, or select **Expression** in a property's drop-down list, as shown in Figure 21.

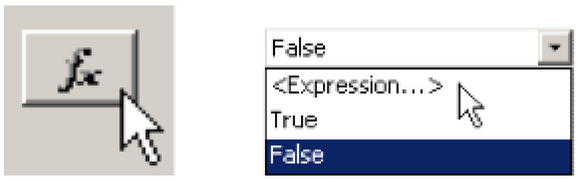
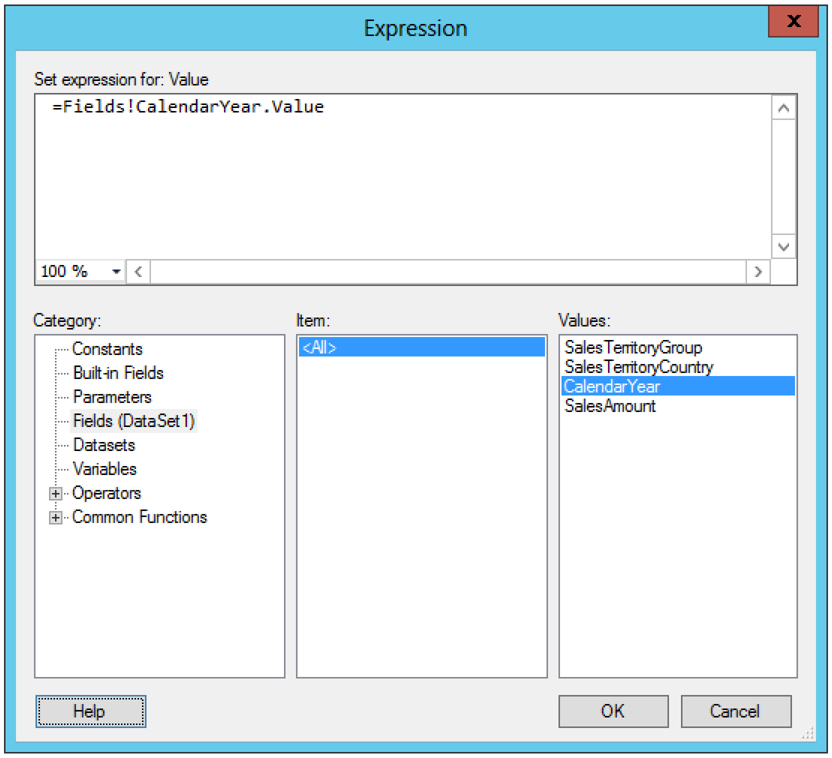


Figure 21: Opening the Expression Editor

In the box at the top of the Expression Editor, shown in Figure 22, you can type your expression directly. You can also use the following lists in the bottom part of the editor to display items that you can use to build an expression:

- **Constants**. This list contains values only when you are editing an expression for a property that has constants. For example, if you are adding an expression for the **Color** property, this list includes a set of colors, such as blue, red, or green.
- **Built-in Fields**. This list contains items found in the Globals collection, such as `ReportName`, `PageNumber`, and `TotalPages`.
- **Parameters**. If your report contains report parameters, their names appear in this list.
- **Fields**. Here you find the fields in the dataset associated with the text box or property to which you're adding the expression. This list is empty otherwise.
- **Variables**. If you create report variables, you see them in this list. Group variables, on the other hand, do not display here.
- **Operators**. This list displays all operators available for mathematical, logical or string operations, among others.
- **Common Functions**. Having a list of functions is helpful, especially if you're not familiar with the available functions or can't remember a function's syntax. The functions are grouped logically by operation. For example, string manipulation function, arithmetic operations, and aggregate functions are organized as separate groups.



```
=Fields!CalendarYear.Value
```

Figure 22: Expression Editor

Expression Examples

There are many ways to use expressions. In this section, I explain a few of the more common examples.

First, if you prompt the user to make a selection by using a report parameter, it's considered a best practice to display the label of the selection somewhere in the report. As you learn later, you can set up a report parameter to allow the user to select multiple values. In that case, you need a way to make one string out of all the values. To do this, you can use the **`Join`** function, reference the Parameters collection, and specify a delimiter to combine values that are stored in an array, like this:

```
=Join(Parameters!SalesTerritoryGroup.Label, ", ")
```

When you use a multi-value parameter, the values and labels are stored in a zero-based array, and you must reference a single item by using an index, like this:

```
=Parameters!SalesTerritoryGroup.Label(0)
```

Another common operation is concatenation of string values. For example, you might combine a field value with a constant string to use as a label in a text box, like this:

```
=Fields!CalendarYear.Value + " Total"
```

Tip: When concatenating values, you need to be sure that all the items are strings. Use conversion functions if necessary.

Conditional formatting is a great way to create dynamic and visually informative reports. Any property that accepts expressions is a candidate for conditional formatting. For example, you can set the **Color** property of a text box based on whether the sales amount for the group of rows in scope is above a specified value like this:

```
=Iif(sum(Fields!SalesAmount.Value)<1000000,"Red", "Black")
```

Collections

The **Datasets** collection allows you to access information about the datasets that are included in a report. From a user perspective, this isn't a particularly useful collection, but it can be for developers who need to troubleshoot report problems, particularly when using dynamic datasets. You can add a text box that displays the command text and see exactly what query is being sent to the data source by using the command text property as shown here:

```
=Datasets!Dataset1.CommandText
```

Likewise there's a **DataSources** collection that's not commonly used except for troubleshooting purposes. You can use it to check the reference to a data source like this:

```
=DataSources!AdventureWorksDW.DataSourceReference
```

ReportItems is a collection that is not displayed in the **Report Data** pane or in the Expression Editor. However, it can be useful when you need to reference the value in a particular text box that in turn may be the result of a calculation. Rather than copy and paste that calculation multiple times, you can define it once and then use the ReportItems collection to reference it. Another important use for the ReportItems collection is access to a text box value after a page renders. SQL Reporting renders a page by preparing the report body first and then the page header and footer. After the report body renders, the dataset is no longer available. However, you can use the ReportItems collection in an expression to access a text box value that you want to display in the page header or footer, like this:

```
=ReportItems!Textbox1.Value
```

The **User** collection is a special collection that you use to set an expression based on the current user. You can use `User!UserID` in an expression to get the current user's Windows login when you want to apply data-level security. When you need to create multilingual reports, you can set report properties based on the user's regional settings by using `User!Language` in an expression.

You can create a variable to store a value at the report level or at the group level and then use the **Variables** collection to retrieve the current value of the variable at report execution time. The value you store in a variable can be a constant value or an expression. You might use a report variable as part of an expression that you use to display the execution time because an expression in the report evaluates only when the page renders, but a report variable doesn't change. That is, if you put the expression `Globals!ExecutionTime` in a page footer of a multiple page report, you find a different value on each page of the rendered report. However, if you store `Now()` in a report variable called `ReportExecutionTime`, and then put the expression `Variables!ReportExecutionTime.Value` in the page footer, you see the same value on each page of the rendered report.

You can also create a variable with group scope. A group variable evaluates when the group instance changes, so it can potentially change multiple types on the same page of a report. You use a group variable when you need to have a value that is unique to the current group.

The **Globals** collection is useful for displaying information that is specific to your report, such as the execution time or the report server URL. You can use an object in this collection in any text box in your report, but you can use the page-related objects, such as `Globals!PageNumber` or `Globals!TotalPages` in the page header or footer only. In the **Report Data** pane, the **Built-in Fields** folder contains eight fields that you can use to quickly add a Globals object to a text box. If you need to change the appearance of your report based on the current rendering, such as hiding a report title if you render a report to Excel, you can assign an expression like this to the **Hidden** property of a text box:

```
=Iif(RenderFormat.Name = "EXCELOPENXML", True, False)
```

Aggregate Functions

You use aggregate functions primarily to summarize many records of numerical data and display a value, such as a total or average. You can also use aggregate functions to get the first or last string value in a group of records. An aggregate function uses a scope argument to determine which detail rows to include in the aggregation. It can refer to group scope, the scope of a data region, or a dataset.

If you omit the scope argument when using an aggregate function, the scope is inferred by the location of the expression. If the expression appears in a text box in a group or a property of that text box, the scope is limited to the dataset rows associated with that group. However, if you want to use a scope other than the inferred scope, you must specify the scope explicitly in the aggregate function by placing the scope name in double quotes, like this:

```
=Sum(Fields!SalesAmount.Value, "Tablix1")
```

Any reference to a group, data region, or dataset must match the actual case used in the name, or the expression fails. You only know if the expression succeeds or fails due to a case mismatch if you attempt to preview the report, since no warning is displayed in the Expression Editor to alert you to the mistake.

Report Parameters

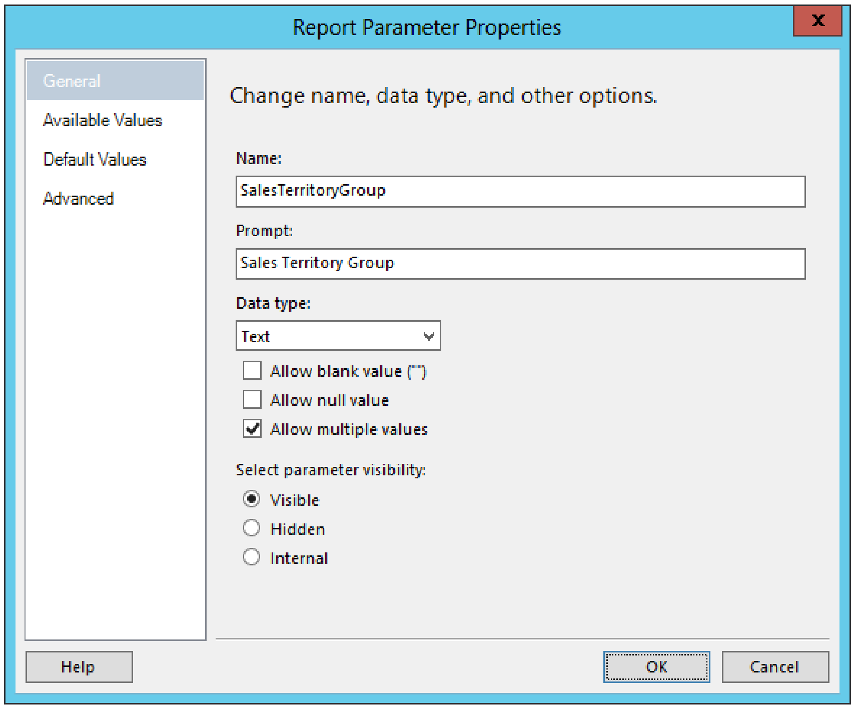
You use report parameters to accept input from a user. At report execution time, the user is prompted to provide the report parameter value. You can supply default values so that the report executes first but allows the user to execute the report again after changing a report parameter value. You can also leave the value empty and force the user to make a selection or type a value before executing the report.

A common way to use this input is to filter data by filtering one or more data regions or by passing the user's selection to a dataset's query parameter. You can also use report parameters in expressions. For example, you can use parameters to prompt the user for information that determines how the report should appear, such as sorting of rows or visibility of columns.

There are three ways that a user can provide input for a report parameter value:

- **Available values**. With this option, the user makes a selection from a drop-down list. You can restrict the user to one selection, or you can allow the user to select multiple values.
- **No value**. You can require the user to type a value. However, apart from data type checking, there's nothing in Reporting Services that you can use to validate the input. If you prompt the user for input in this way, you should not use the value directly within a SQL query because of the potential for a SQL injection attack. If you must use an input value like this as input for a query parameter, such as when you have a value like an invoice number that is not well suited for a drop-down list, you should always use a stored procedure for the dataset so that you can perform validation on the input value before using it in a query.
- **Date value**. When you set the date data type for a parameter, the user can either type the value directly or use the calendar control to select a date.

To add a report parameter to your report, right-click the **Parameters** folder in the **Report Data** pane, and then click **Add Parameter**. At minimum, you need to specify a name for the report parameter, provide a prompt that the user will see in the report viewer, and specify a data type, as shown in Figure 23.



Change name, data type, and other options.

Name: SalesTerritoryGroup

Prompt: Sales Territory Group

Data type: Text

☐ Allow blank value ("")
☐ Allow null value
☑ Allow multiple values

Select parameter visibility:

[✓ Visible] Hidden Internal

Figure 23: Report parameter properties

By default, the report parameter is configured without available values, which means that the user must directly enter a value. You also have the option to allow a blank value or a null value. A blank value is interpreted as an empty string.

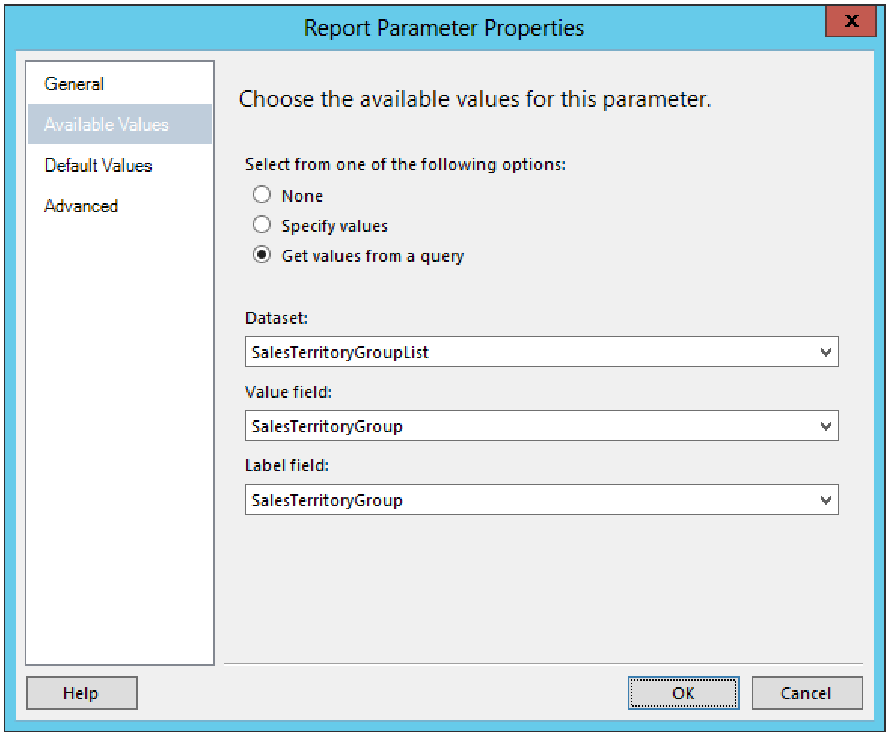
If you do provide available values, the default setting is to allow the user to select only one value at a time from the list, but you can enable the option to allow the user to select multiple values as shown in Figure 23. However, if you enable the multi-value option, you cannot allow null values.

Typically, report parameters are visible to users. Use the **Hidden** option if you don't want the prompt to display when users view the report, but need the ability to set the parameter value on the report server using URL access or when creating a subscription. Use the **Internal** option when you need to use a parameter value in a report, but set the parameter value without user interaction. For example, you can get a parameter value by evaluating an expression or retrieving a value from a dataset, and bypass the need to prompt the user.

When you want the user to select values from a drop-down list, you must configure the **Available Values** for the report parameters. On the **Available Values** page of the report parameter properties, you have the following three options:

- **None**. By default, the **None** option is selected, which forces the user to directly enter a value.
- **Specify Values**. You can enter a static list of valid values for the report parameter. **Value** is the default passed to a parameter expression and to a query parameter, whereas **Label** is the text that is displayed in the report parameter's drop-down list when the user opens the report. For both Label and Value, you can specify an expression rather than a static value.

- **Get Values From a Query**. Whenever possible, you should use a dataset to provide a list of available values for a report parameter. Ideally, your dataset should reference a view in your data source. Then, if you need to add or remove items in the list, you can manage the change in the data source and can thereby avoid the need to edit the report parameter list in each report using the same list. When you use a dataset to supply report parameter values, you typically have two columns in the dataset, one for the parameter value and one for the parameter label. You can, however, use a single column as both value and label, as shown in Figure 24.



Choose the available values for this parameter.

Select from one of the following options:

- None
- Specify values
- [✓ Get values from a query]

Dataset: SalesTerritoryGroupList

Value field: SalesTerritoryGroup

Label field: SalesTerritoryGroup

Figure 24: Report parameter available values from a query

On the **Default Values** page of the report parameter properties, you again have three options:

- **No Default Value**. By default, this option is selected, which means the report does not render until the user selects a value.
- **Specify Values**. You can also provide a default value to allow the report to execute before the user selects a parameter value. Just as with the available values, you can use expressions to specify default values instead of using static text.
- **Get Values From a Query**. You can create a dataset that sets the default value. If the report parameter is configured to accept only a single value, the dataset must return only a single row. A multi-value report parameter can accept multiple rows. Either way, the field that you reference must contain a valid value as defined by the value column on the **Available Values** page.

Filters

One way to change the contents of a report dynamically at report execution time is to use a filter. Although you can create a filter without a report parameter, the most common way to configure a filter is by using a report parameter that prompts the user for the filter value. When you use a report parameter with a filter defined for a report element, the report filter has no effect on the data retrieved from the source. That means any change to the filter value resulting from a new report parameter value selection affects what you see in the report, but does not cause a new execution of the dataset's query to retrieve different data.

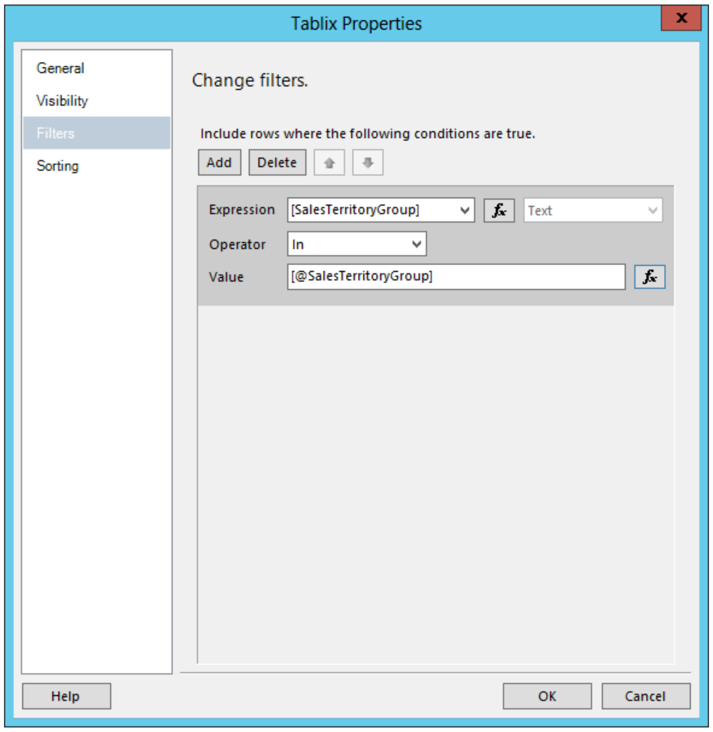
When you configure a filter, you define an expression that determines the condition that must be met in order for data to be visible in the report. If you're creating a dynamic filter based on user input, you can use a report parameter to store the user's selection and then reference that parameter in the filter condition.

There are different ways to add a filter to your report:

- **Dataset**. Because a filter is applied to the dataset rows after the query executes, even if you apply the filter to the dataset, the time to render the report after selecting a new report parameter value is quite low. When you want to both minimize the number of query executions against the source database and maximize report performance when changing report parameter values, use a report filter on the dataset. To add a filter to a dataset, double-click the dataset in the **Report Data** pane, and then open the **Filters** page of the **Dataset Properties** dialog box.
- **Data region**. When you need access to all data in some data regions and filtered data in other data regions on the same report, use a report filter on the applicable data regions instead of filtering the dataset. To add a filter to a data region, select the object in the drop-down list at the top of the **Properties** window, click the **Filters** property box, and then click the ellipsis button that is displayed.
- **Group**. A less commonly used option is applying a filter to a group. To do this, double-click the group in the **Row Groups** or **Column Groups** pane, and open the **Filters** page of the **Group Properties** dialog box.

Regardless of where you apply the filter, the process is similar. You compare one value, such as a field from the dataset, to another value, such as a report parameter value. However, any valid expression can be used for the comparison.

Most of the time, the operator you use for comparison is an equal sign, but when you have a multi-value report parameter, you use the **In** operator instead, as shown in Figure 25.



Change filters.

Include rows where the following conditions are true.

Expression [SalesTerritoryGroup] Text

Operator In

Value [@SalesTerritoryGroup]

Figure 25: Filtering a data region

You can use other operators for comparison, as shown in Table 3.

Table 3: Filter Operators

Operator	**Action**
=, <>, >, >=, <, <=, *Like*	Compares the expression to the value
Top N, Bottom N	Compares expression to Top (Bottom) set of N values (N = integer)
Top %, Bottom %	Compares expression to Top (Bottom) N percent of values (N = integer or float)
Between	Determines whether the expression is between two values, inclusive
In	Determines whether the expression is found in a list of values

Query Parameters

You can use a query parameter to filter data at the source. You might want to do this when it's important to reduce the amount of network traffic caused by returning a dataset's query results to a report. Other reasons include a need to reduce the amount of time required to render a report or security requirements.

You start by adding a query parameter to the WHERE clause of your dataset query like this:

```
SELECT
 st.SalesTerritoryGroup,
 st.SalesTerritoryCountry,
 d.CalendarYear,
 SUM(s.SalesAmount) SalesAmount
FROM
 dbo.FactResellerSales s
 JOIN dbo.DimDate d on d.DateKey = s.OrderDateKey
 JOIN dbo.DimSalesTerritory st on st.SalesTerritoryKey =
   s.SalesTerritoryKey
WHERE st.SalesTerritoryGroup = @SalesTerritoryGroup
GROUP BY
 st.SalesTerritoryGroup,
 st.SalesTerritoryCountry,
 d.CalendarYear
ORDER BY
 st.SalesTerritoryGroup,
 st.SalesTerritoryCountry
```

By default, the report designer adds a matching report parameter to your report and associates the query parameter with the report parameter. You can see this association on the **Parameters** page of the **Dataset Properties** dialog box. You are not required to use a report parameter, though. You can delete the report parameter and then associate the query parameter with an expression, such as the date or the current user.

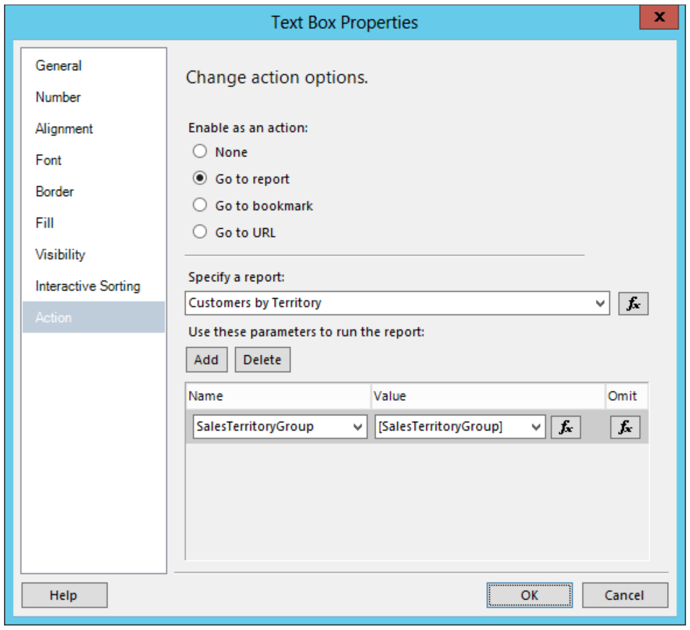
Tip: It's possible that a query parameter value produces a filter that does not return any data from the source. When that happens, the data region associated with the dataset is not displayed in the report, and it can be confusing to the user to see a report title without anything else. To clarify the situation, you can use the NoRowsMessage property of a data region to display a message.

Subreports and Drillthrough

Another way to use parameters is to share data between reports. You can create relationships between reports, either by using a subreport to display one report inside of another report or by using a drillthrough report to allow the user to click on a text box in one report to open another report.

To use this feature, you can configure a report to pass the current context to the associated subreport or drillthrough report. The current context comes from data available for the current scope. It might be a field from the dataset, a custom expression based on a field, or a report parameter. In the case of a drillthrough report, the current context could also come from a text box that the user clicks on.

To implement this feature, the subreport or drillthrough report must have a parameter already defined in the report. After you add a subreport to your report, right-click it and then select **Subreport Properties**, select the report to add, and go to the **Parameters** page of the dialog box. If you are building a drillthrough action instead, right-click the text box to use as a link to the report, click **Text Box Properties**, go to the **Action** page of the dialog box, select the **Go To Report** option, and select the report to use for drillthough. In either case, click **Add** to add a row for defining the parameter to pass, and then select that parameter in the **Name** drop-down list. In the **Value** field, you configure the expression that passes into the parameter in the target report. In Figure 26, a field in the current report will pass to the drillthrough report.



Change action options.

Enable as an action:
None
[✓ Go to report]
Go to bookmark
Go to URL

Specify a report: Customers by Territory

Use these parameters to run the report:

Name	Value	Omit
SalesTerritoryGroup	[SalesTerritoryGroup]	

Figure 26: Configuring a parameter for drillthrough

The **Omit** option for the parameter shown in Figure 26 is applicable only to drillthrough reports. Use it to define an expression that must evaluate to **True** when you want to disable passing the parameters.

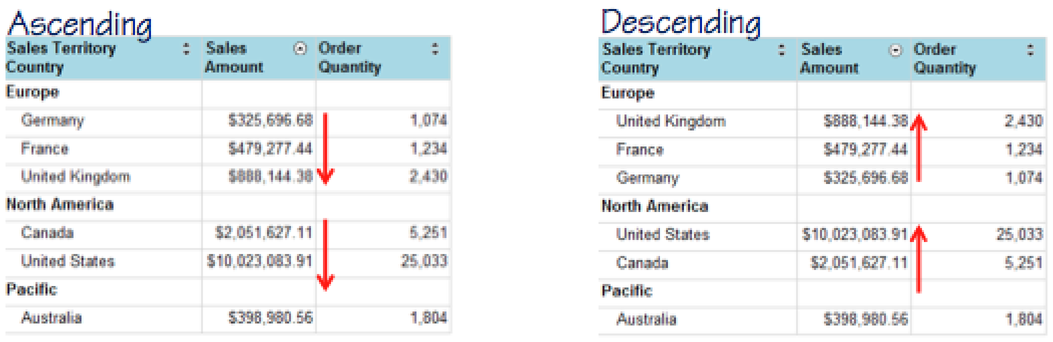
Interactivity

Reporting Services has many interactive features that you can add to reports to enhance the online viewing experience for users. Some of these features affect the report layout such as interactive sorting, fixed headers, tooltips, and visibility. Other features are useful as navigation aids, such as a document map, actions, and embedded HTML.

Interactive Sort

When you design a report, you can define the sort order of the data by including an ORDER BY clause in the dataset or by setting the Sorting property on the tablix or group. You can also add a parameter to the report to prompt the user for the sort direction and use the input to set the **Sorting** property. However, another option is to configure a text box for interactive sort. To do this, right-click the text box, select **Text Box Properties**, go to **Interactive Sorting**, and select the check box to enable this feature.

Usually, the text box to which you add interactive sorting is a column header or row header in a data region. As shown in Figure 27, you can configure each column header in a tablix to use interactive sort, as indicated by the arrow icons in each text box. The user can sort one column at a time. The first time that a user clicks on a column, the data sorts in ascending order with the smallest numbers in the top rows, and the highest numbers in the bottom rows. If the data is organized in groups, the detail rows are sorted in groups. A second click on the column sorts the data in descending order.



Ascending

Sales Territory Country	Sales Amount	Order Quantity
Europe		
Germany	$325,696.68	1,074
France	$479,277.44	1,234
United Kingdom	$888,144.38	2,430
North America		
Canada	$2,051,627.11	5,251
United States	$10,023,083.91	25,033
Pacific		
Australia	$398,980.56	1,804

Descending

Sales Territory Country	Sales Amount	Order Quantity
Europe		
United Kingdom	$888,144.38	2,430
France	$479,277.44	1,234
Germany	$325,696.68	1,074
North America		
United States	$10,023,083.91	25,033
Canada	$2,051,627.11	5,251
Pacific		
Australia	$398,980.56	1,804

Figure 27: Interactive sorting

The interactive sort feature is not limited to placement in column header rows or to sorting only detail rows. Rather than sort the entire tablix, you can also define the sort to apply only within a specific group by adding interactive sort to a text box within the data region and then limiting the scope to the detail rows within that group. Or you can apply the sort to entire groups rather than to the detail rows.

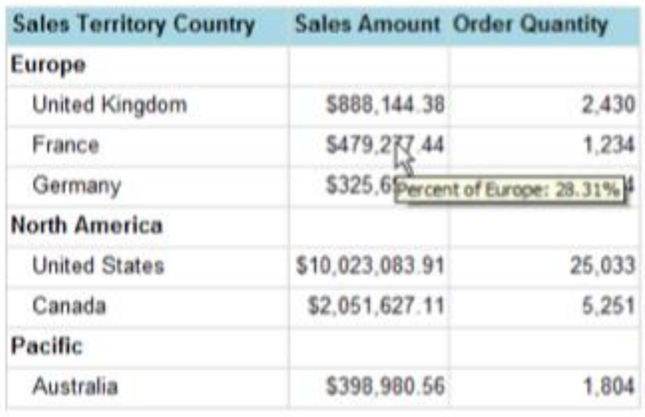
Fixed Headers

If you have a report that doesn't fit completely on the screen, the user must scroll to view the information at the bottom of the page. However, a user can lose track of which information is contained in which column by the time the bottom of the page is reached. To address this problem for vertical scrolling, you can select the tablix in the **Properties** window, and then set the **FixedColumnHeaders** property to **True**. You can change the **FixedRowHeaders** property when your report requires horizontal scrolling.

Tip: When you use either of the fixed headers properties, it's important to change the BackgroundColor property of the row or column. If no color is defined, the background is transparent, and the user will see the data scrolling behind the text in the fixed headers. When you add a color, the data will no longer be visible during scrolling.

Tooltips

You can use a tooltip to display a message when the user positions the cursor over an item in the report such as a text box, as shown in Figure 28. You can use the tooltip to display other data from the dataset that isn't visible in the report or the result of a calculation, like the percentage in this example. Select the item in the report, and then click **Expression** in the drop-down list for the **Tooltip** property in the **Properties** window. You can then type **static text** in double quotes or add an expression that resolves as a string value.



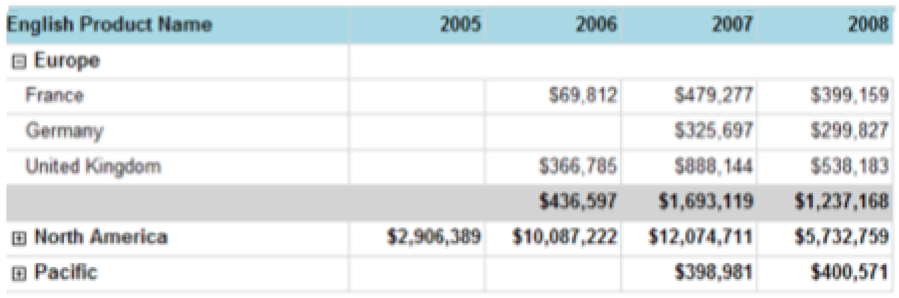
Sales Territory Country	Sales Amount	Order Quantity
Europe		
United Kingdom	$888,144.38	2,430
France	$479,277.44	1,234
Germany	$325,6	
North America		
United States	$10,023,083.91	25,033
Canada	$2,051,627.11	5,251
Pacific		
Australia	$398,980.56	1,804

Percent of Europe: 28.31%

Figure 28: Tooltip

Visibility

The visibility feature is not limited to online viewing, but the interactive aspect of it is. A common reason to use this feature is to include both summary and detail data in a report. Figure 29 is an example of a tablix that contains a group row for Sales Territory Group and child rows for Sales Territory Country.



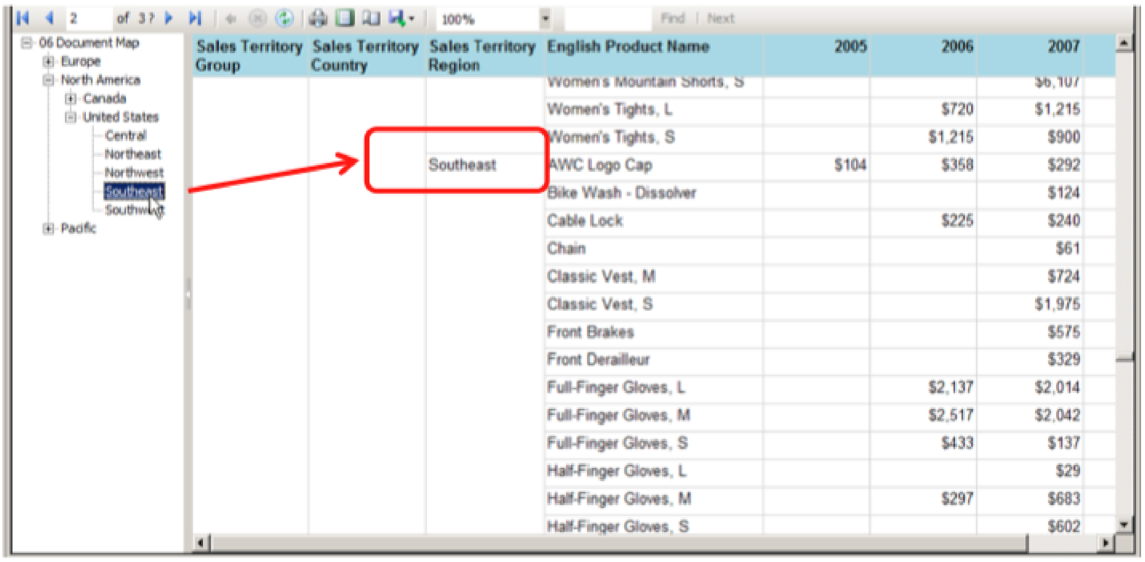
English Product Name	2005	2006	2007	2008
⊟ Europe				
France		$69,812	$479,277	$399,159
Germany			$325,697	$299,827
United Kingdom		$366,785	$888,144	$538,183
		$436,597	$1,693,119	$1,237,168
⊞ North America	$2,906,389	$10,087,222	$12,074,711	$5,732,759
⊞ Pacific			$398,981	$400,571

Figure 29: Using visibility properties to toggle the display of detail rows

To create the appearance of a drilldown report in which only the summary data is displayed when the user opens the report, use the visibility properties **Hidden** and **ToggleItem**. Visibility can apply to an entire row, an entire column, or a specific text box. Select the text box or click the column or row handle and then, in the **Properties** window, set the **Hidden** property to **True**. In the **ToggleItem** property's drop-down list, select the name of the text box in which you want the toggle switch icon to display.

Document Map

If you have more than one group defined in a tablix report item, you can add the document map feature as a navigation aid for a long report. It works like an interactive table of contents for the report that allows the user not only to see the group instances at a glance, but also to jump to the location of a specific instance. When the report includes a hierarchy of groups, the document map displays the list of bookmarks in a tree form. The user must expand a specific group instance to view the instances of groups contained by the selected group, as shown in Figure 30.



Sales Territory Group	Sales Territory Country	Sales Territory Region	English Product Name	2005	2006	2007
			Women's Mountain Shorts, S			$6,107
			Women's Tights, L		$720	$1,215
			Women's Tights, S		$1,215	$900
		Southeast	AWC Logo Cap	$104	$358	$292
			Bike Wash - Dissolver			$124
			Cable Lock		$225	$240
			Chain			$61
			Classic Vest, M			$724
			Classic Vest, S			$1,975
			Front Brakes			$575
			Front Derailleur			$329
			Full-Finger Gloves, L		$2,137	$2,014
			Full-Finger Gloves, M		$2,517	$2,042
			Full-Finger Gloves, S		$433	$137
			Half-Finger Gloves, L			$29
			Half-Finger Gloves, M		$297	$683
			Half-Finger Gloves, S			$602

Figure 30: Document map

For each instance in a group, you define the expression to display in the document map. This expression becomes the equivalent of a bookmark and is displayed next to the report. To define this expression, double-click the group in the **Row Groups** pane and go to the **Advanced** page.

Note: The document map works not only for reports rendered in HTML, but also for reports rendered in Excel and PDF.

Actions

Each text box and image has an **Action** property that you can use to define a target location for display when the user clicks the text box or image. Select the item in the report body, click the **Actions** property box in the **Properties** window, and then click the ellipsis button. You can implement the following three types of actions:

- **Go To Bookmark**. A bookmark is useful for jumping from one section of a report to another. Consider a scenario in which you have a tablix on one page of a report, and a list containing charts spread out over multiple pages. You can set the Bookmark property of the chart using an expression that produces a unique name for each chart in the list. Then, configure an action on a text box in the tablix that uses an expression based on the value in the text box the user clicks to generate the bookmark target. At run time, clicking the text box causes the report to jump to the applicable chart.

- **Go To Report**. To configure this action, you define a target report. Optionally, you can map expressions in the parent report to parameters in the target report. In that case, the target report can display information using the same context as the source report if the target report is designed to filter based on report parameter values.
- **Go To URL**. This action requires the user to click on a text box in the report and that in turn causes Reporting Services to open the default browser using a target URL as the address. You can provide a static URL or use an expression that resolves as a URL.

Embedded HTML

Another way to work with URLs and other types of HTML markup is to configure text as HTML markup, also known as embedded HTML. To use this technique, you start by using an expression in a text box that resolves as a valid hyperlink. This expression can come directly from the dataset, or can be derived as a complex expression in the text box.

You can use HTML markup tags listed in Table 4, such as FONT, DIV, LI, or B, to control the report layout, but the A HREF tag in particular is useful for navigation. For example, you might have a report that displays email addresses. When you include the HTML tag in the string expression for the text box, the report renders the HTML tag as text. To render the expression as HTML, you must select the text in the text box rather than select the text box, and change the **MarkupType** property to **HTML** in the **Properties** window. The report then renders the string as a link, and the user’s email client opens when the user clicks the link and creates a new message addressed to the selected recipient.

Table 4: HTML Markup Tags

Tag Type	HTML Markup Tag
Font	<FONT>
Header, Style, and Block Elements	1.1.1.1.1 <DIV> 1.1.1.1.2 <H{n}> 1.1.1.1.3 <HN> 1.1.1.1.4 <LI> 1.1.1.1.5 <P> <SPAN>
Hyperlink	<A Href>

Tag Type	HTML Markup Tag
List	<LI> <OL> <UL>
Text Format	<B> <I> <S> <U>

Pagination

You can configure properties in your report to define page size, margin, and orientation. By default, the page size is set to 8.5 inches wide by 11 inches high, which produces a portrait orientation, but you can change it to landscape by setting the **Width** property to 11 inches and the **Height** property to 8.5 inches. To access these properties, select **Report** in the drop-down list at the top of the **Properties** window.

Although you can configure the height and width of the report, there are other properties that affect the distribution of report items on each page of the report. Margins, body size, and the presence or absence of page headers and page footers each have an effect on the overall page size, as shown in Figure 31. The renderer determines the page size by subtracting the margin sizes from the height and width defined for the report. Then within that area, the render allocates space to the page header and page footer, if you use them, and the remainder is available to the body of the report.

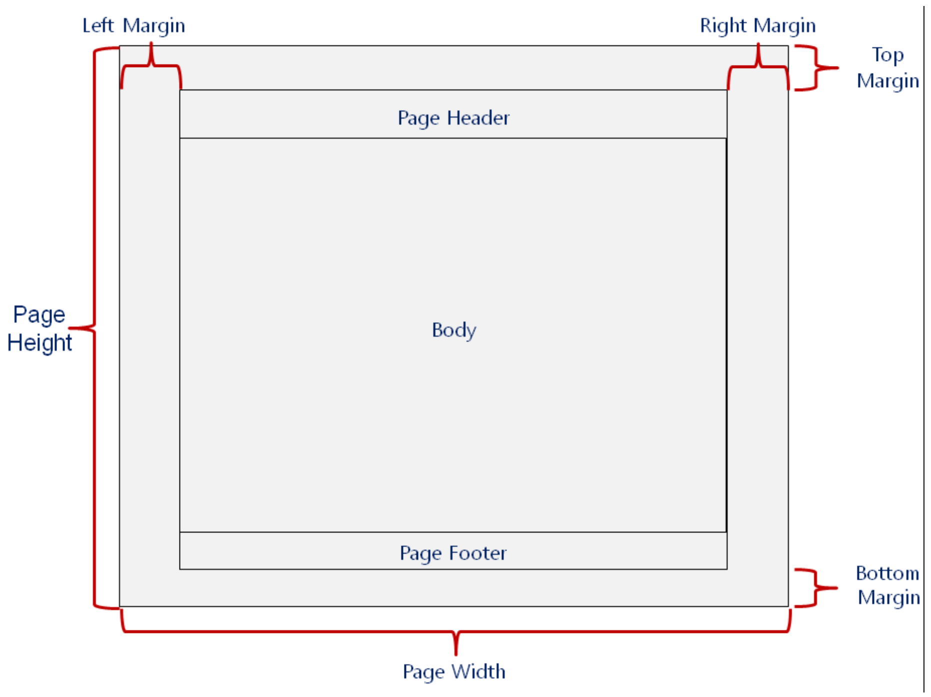


Figure 31: Page sizing

When you select **Report** in the **Properties** window drop-down list, you can set the **Width** and **Height** properties as well as the properties for the margins: **Left**, **Right**, **Top**, and **Bottom**. If you right-click in the report designer outside the report body, you can click the command to add a page header or page footer and then select these items in the **Properties** window to adjust the **Height** property. Additionally, you can select **Body** to set Width and Height properties.

Tip: If the body size plus the margins exceeds the page size, the report contains an extra blank page between pages containing report content when you print it or export to PDF or TIFF format.

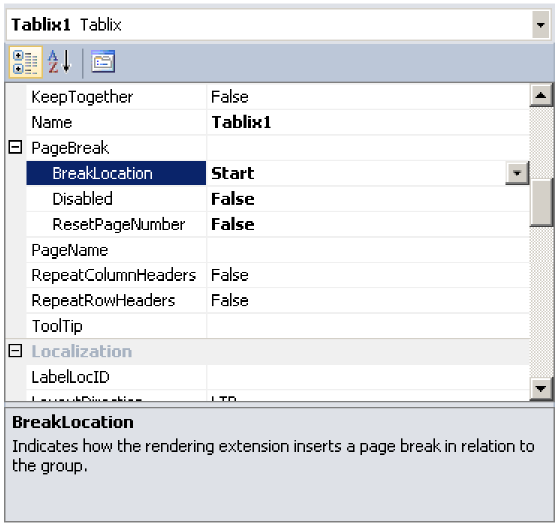
Another factor affecting page size is the white space in the body of your report. If the body size is wider than the report size, blank pages appear in your printed report. This problem might not be evident on the report design surface. For example, in a matrix, where columns are dynamic, the white space to the right of a column group is included in the report after all the columns are rendered resulting in blank pages, even when the matrix itself fits on one page. To disable this behavior, select **Report** in the **Properties** window and change the value of the **ConsumeContainerWhiteSpace** property to **True**.

By default, SQL Reporting attempts to fit as much of a data region onto a page as it can before creating a new page, known as a soft page break. However, you can insert hard page breaks relative to a data region or when a group instance changes within a data region. A rendering extension might continue to insert soft page breaks if the location of the page break that you add results in a page size larger than the report properties allow.

Page Breaks

You can create page breaks on most report items. To do this, select the data region to which you want to add a page break, and then expand the **PageBreak** category in the **Properties** window, as shown in Figure 32, to set the following properties:

- **BreakLocation.** Set the location of the page break relative to the selected report item: None, Start, End, or Start And End. You can set the break location at the start of a report item to force a page break before the report item renders. Or you can set the break location at the end of a report item. This might not make sense if nothing appears in the report after the report item that you set the page break on, because you'll end up with an extra blank page at the end. You can also have a page break before and after the report item renders.
- **Disabled.** Use a conditional expression to determine whether to apply a page break. For example, you might set it to **True** when the render format is HTML to prevent a page break, but set it to **False** when the render format is PDF to add the page break.

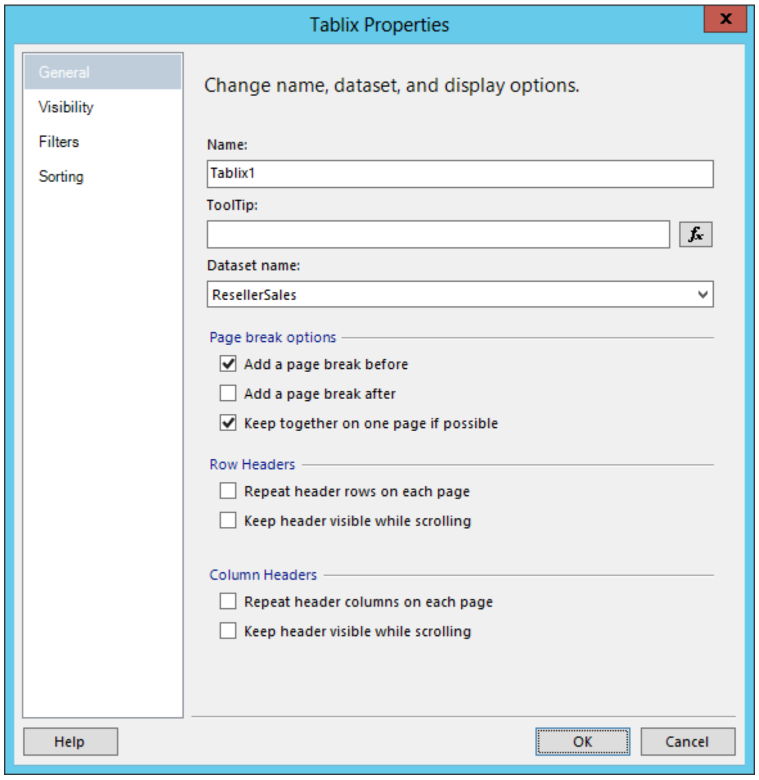

Tablix1 Tablix

KeepTogether	False
Name	Tablix1
PageBreak	
BreakLocation	Start
Disabled	False
ResetPageNumber	False
PageName	
RepeatColumnHeaders	False
RepeatRowHeaders	False
ToolTip	
Localization	
LabelLocID	

BreakLocation
Indicates how the rendering extension inserts a page break in relation to the group.

Figure 32: Page break properties

You can also configure a page break in the **Properties** dialog box for a data region, such as the **Tablix Properties** dialog box shown in Figure 33. In the **Page Break Options** section, you can add a break before or after the data region or both before and after it renders. For the tablix, you can also select the **Keep Together On One Page If Possible** check box to set the **KeepTogether** property to **True**.



Change name, dataset, and display options.

Name: Tablix1

ToolTip:

Dataset name: ResellerSales

Page break options

- [✓] Add a page break before
- [ ] Add a page break after
- [✓] Keep together on one page if possible

Row Headers

- [ ] Repeat header rows on each page
- [ ] Keep header visible while scrolling

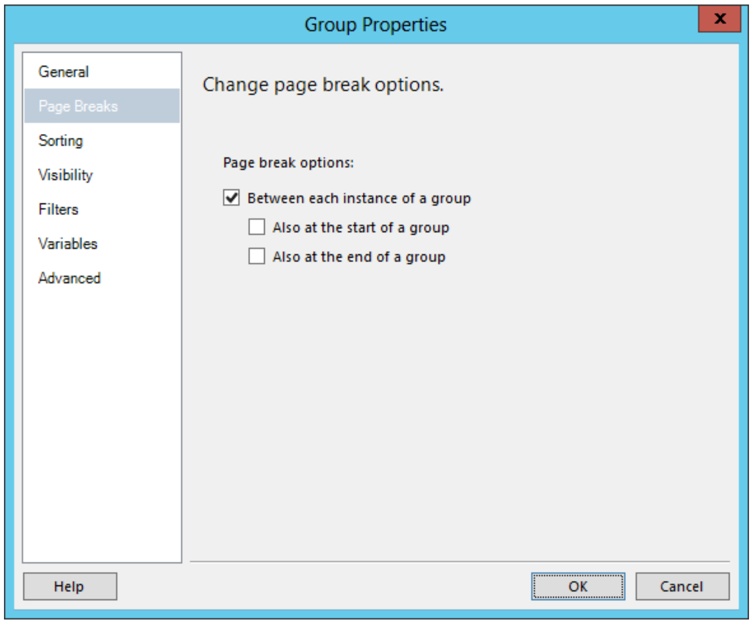
Column Headers

- [ ] Repeat header columns on each page
- [ ] Keep header visible while scrolling

Figure 33: Tablix pagination properties

When you set the **KeepTogether** property to **True**, the rendering engine pushes the tablix to a new page if it doesn't fit on the same page as the preceding items. If it's not possible to render the tablix on a single page, the rendering engine renders as much of the table as possible on the first page and then continues on subsequent pages.

You can also set page breaks by group within a data region. The same properties that you have for setting page breaks by report item are applicable to groups: **BreakLocation** and **Disabled**. However, for **BreakLocation**, you define a page break between group instances in addition to breaking at the start or end of a group. That way, you don't create an unnecessary page break at the beginning or end of a report item and create a new page only when a new group starts. To do this, select the group in the **Groupings Pane**, and then expand the **Group** category in the **Properties** window to locate the **PageBreak** options. Another way to do this is to right-click the group and open the **Page Breaks** page of the **Group Properties** dialog box, as shown in Figure 34.



Change page break options.

Page break options:

[✓] Between each instance of a group

[ ] Also at the start of a group

[ ] Also at the end of a group

Figure 34: Group page break options

Repeating Headers

By default, the column headers appear only once when the table renders, so when a table spans multiple pages, you lose the context of each column on subsequent pages. To make the tablix easier to read, you can configure repeating column headers. You can do this by setting the **RepeatColumnHeaders** property to **True** in the **Properties** window for the tablix.

If your tablix doesn't have column groups in the tablix, you won't achieve the desired effect. First, you need to view the **Row Groups in Advanced Mode**. To do this, click the arrow to the right of the **Column Groups** and select **Advanced Mode**. Notice that **Static Groups** now appear in the **Row Groups**, as shown in Figure 35.

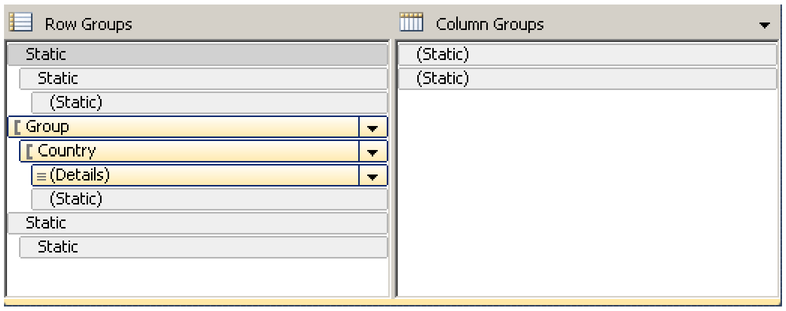


Figure 35: Viewing groups in Advanced Mode

You can set the column headers by selecting the static group that highlights the leftmost column header. This is generally the first static group listed. In the **Properties** window, set the **RepeatOnNewPage** property to **True** and the **KeepWithGroup** property to **After**. This latter property specifies which group the static member needs to stay with as follows:

- **After**. The static member stays with the group after or below it, acting as a group header.
- **Before**. The static member stays with the group before or above it, acting as a group footer.
- **None**. The rendering engine automatically decides where to put the static member.

Page Numbering

When a report spans multiple pages, you should add page numbers in a page header or page footer by using one of the following global variables in an expression:

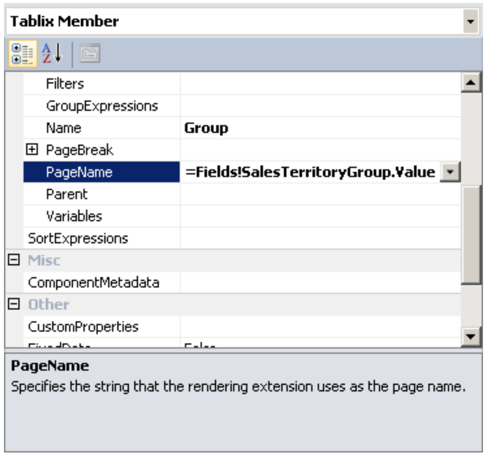
- **Globals!PageNumber**. The current page number relative to the last page number reset.
- **Globals!TotalPages**. The total number of pages in the current page numbering group.
- **Globals!OverallPageNumber**. The current page number in the report.
- **Globals!OverallTotalPages**. The total number of pages in the report.

If you do not reset page numbers, the **OverallPageNumber** variable will be the same as the **PageNumber** variable, and the **TotalPages** variable will be equal to the **OverallTotalPages** variable. You can use the **ResetPageNumber** property in the set of page break properties for a report item or group to reset the numbering.

Page Naming

In addition to numbering pages, you can also assign each page a unique name. The first page of a report has a blank name by default. If you decide to assign a page name, you assign a constant value or expression to the report property **InitialPageName**.

You can then reset the page name when you create a page break for a report item or group. For example, you might use the name of a group to reset the page name. You can set the page name property only by using the Properties window. Select the item that you want to use to control the page name, such as the tablix group, and then set the **PageName** property, as shown in Figure 36.

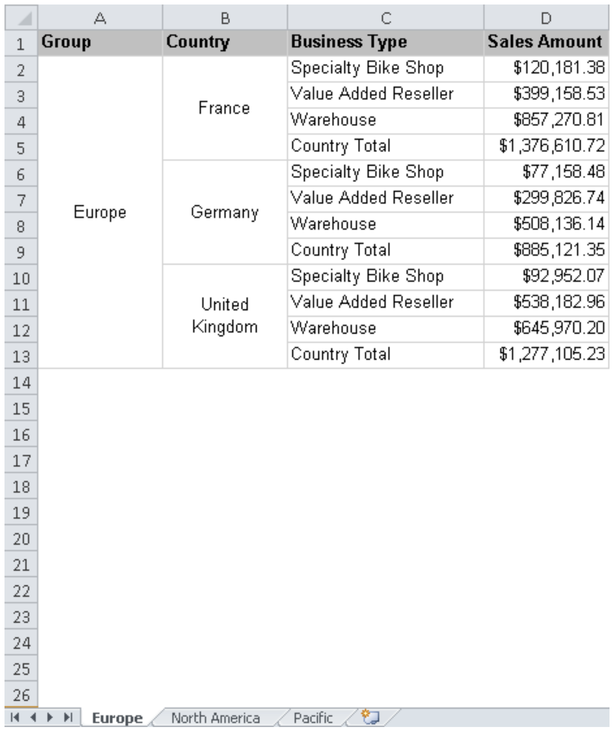

Tablix Member

Filters	
GroupExpressions	
Name	**Group**
⊞ PageBreak	
PageName	**=Fields!SalesTerritoryGroup.Value**
Parent	
Variables	
SortExpressions	
⊟ Misc	
ComponentMetadata	
⊟ Other	
CustomProperties	

PageName
Specifies the string that the rendering extension uses as the page name.

Figure 36: PageName property

When you export a report to Excel, the page name becomes the sheet name, as shown in Figure 37. If you set an initial page name for your report but do not set page names for the other pages in the report, all the sheets in Excel will have the initial page name.



Group	Country	Business Type	Sales Amount
Europe	France	Specialty Bike Shop	$120,181.38
		Value Added Reseller	$399,158.53
		Warehouse	$857,270.81
		Country Total	$1,376,610.72
	Germany	Specialty Bike Shop	$77,158.48
		Value Added Reseller	$299,826.74
		Warehouse	$508,136.14
		Country Total	$885,121.35
	United Kingdom	Specialty Bike Shop	$92,952.07
		Value Added Reseller	$538,182.96
		Warehouse	$645,970.20
		Country Total	$1,277,105.23

Figure 37: Page names become sheet names in Excel

Report Builder

Report Builder is a click-once application that you can download from an on-premises report server on demand or install as a separate client application on your computer. Report Builder is primarily a tool for information workers and power users. Although it only allows you to work with one report at a time, it otherwise provides the same functionality as BIDS or SSDT. Regardless of which tool you decide to use, the report development process remains the same.

Getting Started Wizard

The Getting Started Wizard is displayed when you open Report Builder unless you select the **Don't Show This Dialog Box At Startup** check box in the lower left corner and close the window. This wizard prompts you to start a new task by choosing one of the following options:

- **New Report**. This selection starts another wizard that steps you through the process of building a table, matrix, chart, or map.
- **New Dataset**. Choose this option to open the Dataset Designer to create a shared dataset to publish to SQL Reporting for use in multiple reports.

- **Open**. Use this option to connect to SQL Reporting and locate a report to edit in Report Builder.
- **Recent**. You can return to a report that you opened recently or view often.

Report Development in Report Builder

The report development process in Report Builder is very similar to working in BIDS or SSDT, except that you can only work with one report at a time. Furthermore, instead of using the **Toolbox** window to add report items to your report, you use the ribbon displayed across the top of the application window. The ribbon includes the following tabs:

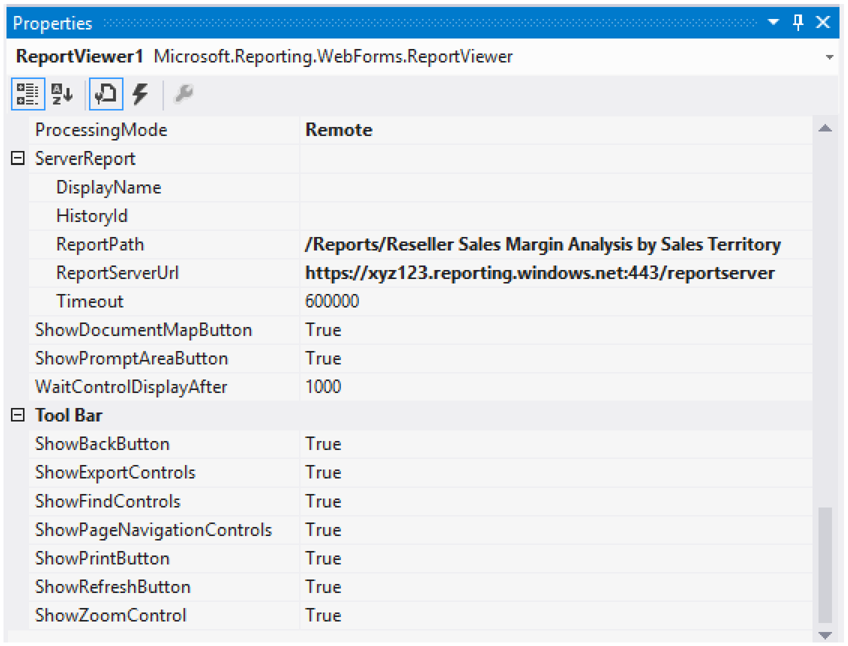
- **Home**. This tab of the ribbon contains commands to toggle between design and preview mode, copy and paste items, and format text, borders, and numbers.
- **Insert**. You use this tab to add report items to your report or to open the **Report Part Gallery**. Commands on this tab allow you to add items directly to the report or use a wizard to help you arrange dataset fields in the item.
- **View**. This tab contains check boxes that you use to hide or show the **Report Data**, **Grouping**, or **Properties**, windows or the ruler.

ReportViewer Control

By using the **ReportViewer** control in your applications, you can focus your development efforts on application logic and save the time required to build the same reporting capabilities that SQL Reporting already includes. You can create an on-premises application or host your application in Windows Azure.

On-Premises Application

To make an on-premises application, create a Windows or Web form in your application and then add the **ReportViewer** control to it. You can find it in the **Reporting** group in the **Toolbox**. Next, in the design window, click anywhere in the center of the **ReportViewer** control to set the focus of the **Properties** window. In the **Properties** window, the **ServerReport** section contains the properties that define the connection to a report on the SQL Reporting server, as shown in Figure 38.



ProcessingMode	Remote
⊟ ServerReport	
DisplayName	
HistoryId	
ReportPath	/Reports/Reseller Sales Margin Analysis by Sales Territory
ReportServerUrl	https://xyz123.reporting.windows.net:443/reportserver
Timeout	600000
ShowDocumentMapButton	True
ShowPromptAreaButton	True
WaitControlDisplayAfter	1000
⊟ Tool Bar	
ShowBackButton	True
ShowExportControls	True
ShowFindControls	True
ShowPageNavigationControls	True
ShowPrintButton	True
ShowRefreshButton	True
ShowZoomControl	True

Figure 38: ReportViewer properties

Note: Before you configure the ReportViewer control, you must deploy reports to the SQL Reporting server as described in Chapter 4, Report Management.

At a minimum, you must configure the **ReportViewer** control's **ProcessingMode**, **ReportPath**, and **ReportServerURL** properties. To use reports managed by **Reporting Services**, you must change the **ProcessingMode** property to **Remote**. When you define **ReportPath**, you provide only the folder and report name. You cannot include URL access commands. For **ReportServerURL**, you use the Web Service URL, which is the same URL that you use when deploying reports from BIDS or SSDT.

The **Toolbar** section of the **Properties** window includes several properties that allow you to configure the visibility of each item in the HTML Viewer. You can remove buttons from the HTML Viewer by setting the corresponding property to **False**. For example, you can hide the **Export** button if you want to restrict users to viewing reports online only.

You also need to modify the code in your form to pass the login user and password like this:

```
reportViewer1.ServerReport.ReportServerCredentials.SetFormsCredentials(
  null, "awadmin", "<password>", null);
reportViewer1.RefreshReport();
```

When you build your application, Visual Studio creates its executable in the **bin\Debug** folder. You can then distribute this executable file to anyone who has access to the SQL Reporting server and has permissions to the report.

Windows Azure Application

Another option is to use Windows Azure to host your application. When you create a Windows Azure application in Visual Studio, you must select the ASP.NET Web Role, assign a name to the role (such as RptVwrWebRole), and add it to your solution. You need to add a **ScriptManager** control to the design surface of your Web form above the **ReportViewer** control. Then add the **ReportViewer** control from the **Reporting** group in the **Toolbox** to the **Default.aspx form**, and then set the properties for this control as described for an on-premises application in the previous section of this chapter.

Next you need to add a class to implement and call the **IReportServerCredentials** interface like this:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Configuration;

using Microsoft.Reporting.WebForms;
using System.Security.Principal;
using System.Net;

namespace RptVwrWebRole
{
    public partial class _Default : Page
    {

        protected void Page_Init(object sender, EventArgs e)
        {
            ReportViewer1.ServerReport.ReportServerUrl = new
                Uri(String.Format("https://{0}/reportserver",
                ConfigurationManager.AppSettings["SERVER_NAME"]));
            ReportViewer1.ServerReport.ReportPath =
                ConfigurationManager.AppSettings["REPORT_PATH"];
            ReportViewer1.ServerReport.ReportServerCredentials = new
                ReportServerCredentials();
        }
    }
    public class ReportServerCredentials : IReportServerCredentials
    {
        public ReportServerCredentials()
        {
        }
        public WindowsIdentity ImpersonationUser
        {
            get
            {
                return null;
            }
        }
        public ICredentials NetworkCredentials
        {
            get
            {
                return null;
            }
        }
        public bool GetFormsCredentials(out Cookie authCookie, out string
user, out string password, out string authority)
        {
            authCookie = null;
            user = ConfigurationManager.AppSettings["USERNAME"];
            password = ConfigurationManager.AppSettings["PASSWORD"];
            authority = ConfigurationManager.AppSettings["SERVER_NAME"];
            return true;
        }
    }

}
```

You also need to update the Web.config file in your solution with the details about your report location and the credentials the application uses to access the report. To do this, you specify the application settings in the `<appsettings>` element inside the `<configuration>` element like this:

```
<appSettings>
      <add key="SERVER_NAME" value="xyz123.reporting.windows.net" />
      <add key="USERNAME" value="awuser"/>
      <add key="PASSWORD" value="<awuser_password>"/>
      <add key="REPORT_PATH" value="/Reports/Reseller Sales Margin Analysis
by Sales Territory"/>
</appSettings>
```

To run the **ReportViewer** control in a Windows Azure application, you must add the Microsoft.ReportViewer.Common.dll reference to your project. You can find this DLL in the C:\Windows\assembly\GAC_MSIL\Microsoft.ReportViewer.Common path and then navigating to the version folder that matches the version of Visual Studio you are using. The Microsoft.ReportViewer.WebForms.dll is also required, but is added to the folder when you add the control to the form. Then, for both assemblies, set the **Copy Local** property to **True**.

Now prepare the deployment package for Windows Azure. To do this, right-click the cloud service name in **Solution Explorer**, click **Package**, and then click **Package** in the dialog box. To upload the deployment package, go to the new Windows Azure portal, open the **Cloud Services** page, click **Create a Cloud Service**, give the service a name, and select a region.

After the cloud service is created, click the service name to open its dashboard. Click the **Certificates** link at the top of the page, and then click **Upload a Certificate**. Browse to the certificate on your computer and select it for upload, and then type the private key password.

Return to the dashboard for your cloud service and click the link **Upload a New Production Deployment**. Provide a deployment name and then browse to the bin\release\app.publish folder of your solution to locate the CSPKG and CSCFG files for your package. Be sure to select **Deploy Even If One Or More Roles Contain a Single Instance** if necessary.

When the deployment is complete, return to the dashboard for your deployment to get the site URL for the service. You can click the link to open your application in the browser.

Chapter 4 Report Management

When you finish report development, you deploy the report to production, which means you copy the report definition language (RDL) file to the report server. At that point, the report development process is complete and the report administration process begins. In the SQL Reporting management portal, you can upload content, manage shared data sources, and monitor SQL Reporting activity by reviewing the execution log or server usage statistics.

Report Deployment

Deployment is the process of moving files from the report developer's computer to the SQL Reporting server. Usually, the report developer creates files as part of a solution in BIDS or SSDT. A solution can contain one or more report projects, each of which can contain a set of files, such as shared data sources, shared datasets, reports, and report parts. The deployment process copies these files to the ReportServer database. Similarly, a user can save a report developed in Report Builder to the SQL Reporting server.

There are two ways that you can deploy reports and related files to the report server from BIDS or SSDT:

- Click **Deploy** on the **Build** menu.
- Right-click a project in **Solution Explorer** and click **Deploy** on the **Context** menu.

Either way, when you select the project in **Solution Explorer** before initiating the deploy process, all reports, shared data sources, shared datasets, and report parts in the project deploy after you provide credentials for a user having a Content Manager or Publisher role assignment. Likewise, you can deploy a solution with multiple projects and deploy reports for all projects. In some cases, you might prefer to deploy a specific report or set of reports. To do this, select one or more reports individually in **Solution Explorer**, and deploy only those selected reports.

You might give certain users the necessary permission to add content to the SQL Reporting server from Report Builder. These users can save reports to the server when they have the server URL.

Note: When you deploy or upload a report with a shared data source, that data source must already exist on the SQL Reporting server to allow SQL Reporting to bind the two objects together. Otherwise, the procedure will fail.

Another option is to give users access to the SQL Reporting management portal as co-administrators where they can upload reports directly to the report server using the web application. However, this interface allows users to upload one report at a time by using the **Upload** button on the toolbar.

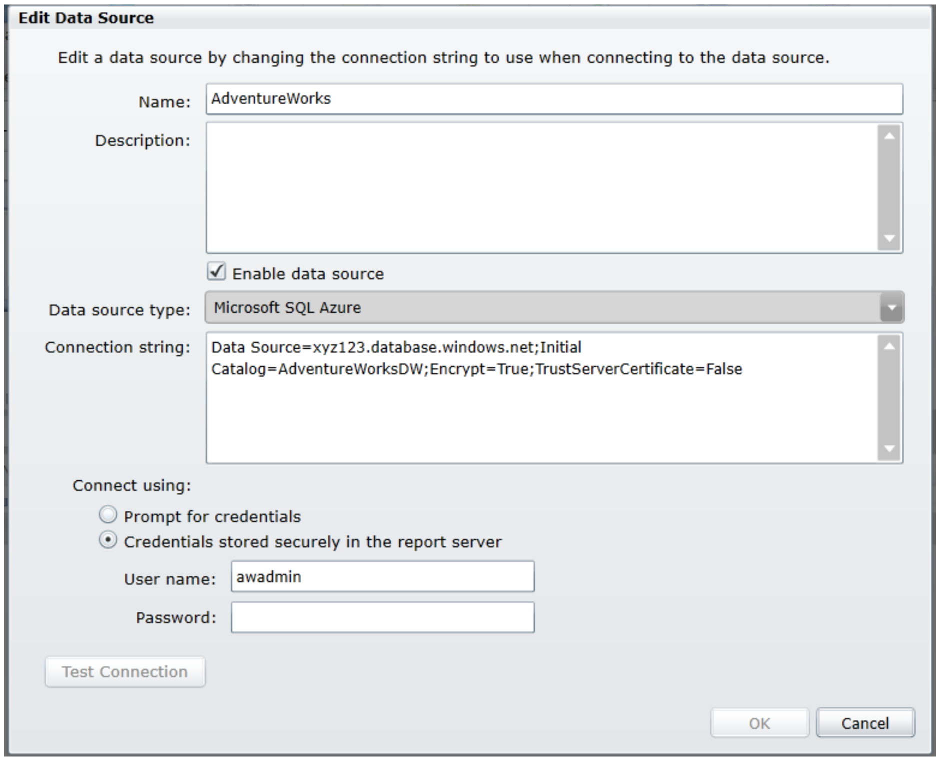
Redeployment

Sometimes report modifications are required to fix issues during testing or to accommodate changing business requirements. In that case, you need to edit the report definition file and then redeploy the report, which you can do using any of the methods described in the previous section of this chapter. However, it's important to understand what happens during redeployment.

Let's say that you've made a change to the RDL on your computer and that RDL has previously been deployed to the SQL Reporting server. The deployment process replaces the original report definition on the server with the new report definition as long as you retain the target folder setting. All existing report properties are preserved, so it's not necessary to update the data source definition or security permissions when you redeploy a report.

Data Source Management

After you deploy a data source to the SQL Reporting server, you can make changes to it by altering the connection string or by changing the credentials associated with the data source. In the SQL Reporting management portal, navigate to the folder containing the data source and click the data source link to open the **Edit Data Source** dialog box shown in Figure 39.


Edit Data Source

Edit a data source by changing the connection string to use when connecting to the data source.

Name: AdventureWorks

Description:

☑ Enable data source

Data source type: Microsoft SQL Azure

Connection string: Data Source=xyz123.database.windows.net;Initial Catalog=AdventureWorksDW;Encrypt=True;TrustServerCertificate=False

Connect using:

○ Prompt for credentials

● Credentials stored securely in the report server

User name: awadmin

Password:

Figure 39: Data source settings on SQL Reporting server

Notice the **Connect Using** options at the bottom of the **Edit Data Source** dialog box. Instead of storing credentials in the report server, you can select the **Prompt For Credentials** option to prompt the user for credentials when the user opens the report. If you select this option, the user sees the following message as a prompt: “Enter a user name and password to access the data source.” You can replace this prompt with a customized message if you like.

Execution Log

In the SQL Reporting management portal, click **Download** on the ribbon to get a copy of the execution log in CSV format. You provide a start date and end date for the data you want to download as well as a file name in which to store the data. The file includes the following information:

- **ItemPath**. The folder hierarchy in which the item is located on the SQL Reporting server.
- **UserName**. The user login name executing the report.
- **ExecutionId**. A unique identifier for a user session.
- **RequestType**. In SQL Reporting, the only valid value is Interactive.
- **Format**. The format to which the report is rendered.
- **Parameters**. The parameter values for the report execution.
- **ItemAction**. The action performed during report execution: Render, Sort, BookMarkNavigation, DocumentNavigation, GetDocumentMap, Findstring, Execute, or RenderEdit.
- **TimeStart**. The date and time the report execution started.
- **TimeEnd**. The date and time the report execution ended.
- **TimeDataRetrieval**. The number of milliseconds required to retrieve data.
- **TimeProcessing**. The number of milliseconds required to process the report.
- **TimeRendering**. The number of milliseconds required to render the report.
- **Source**. The source of the report execution, such as Live or Session, in which a subsequent request to export the report to another format occurs.
- **Status**. A successful status is logged as rsStatus. Otherwise the first error code encountered is logged.
- **ByteCount**. The byte size of a rendered report.
- **RowCount**. The total number of rows in all datasets in a report.
- **AdditionalInfo**. An XML fragment that contains more information about report execution. The information stored here can vary for each entry in the execution log.

An example of the **AdditionalInfo** value looks like this:

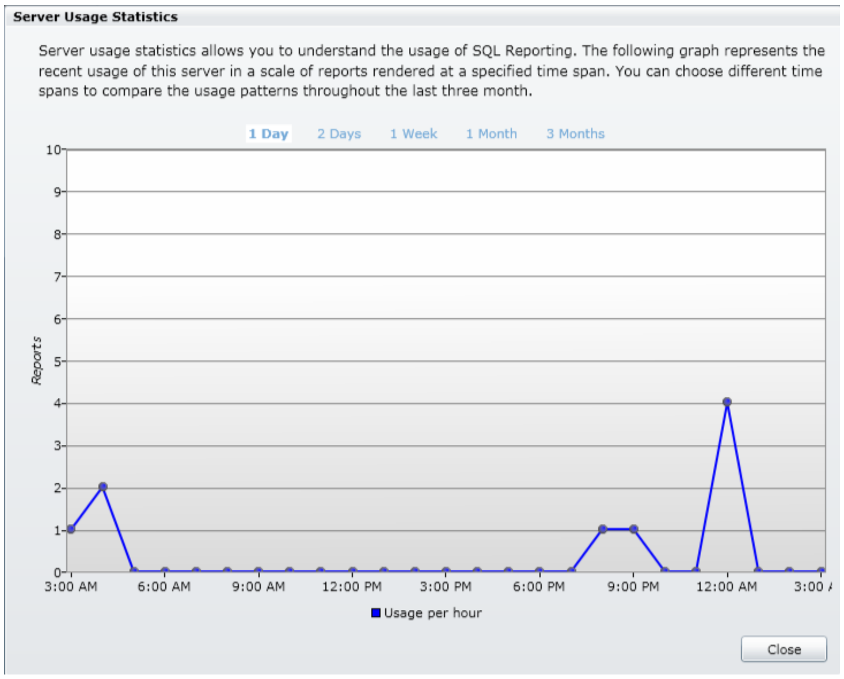
```
<AdditionalInfo>
  <ProcessingEngine>2</ProcessingEngine>
  <ScalabilityTime>
    <Pagination>0</Pagination>
    <Processing>0</Processing>
  </ScalabilityTime>
  <EstimatedMemoryUsageKB>
    <Pagination>3</Pagination>
    <Processing>19</Processing>
  </EstimatedMemoryUsageKB>
  <DataExtension>
    <SQLAZURE>1</SQLAZURE>
  </DataExtension>
  <Connections>
    <Connection>
      <ConnectionOpenTime>5</ConnectionOpenTime>
      <DataSets>
        <DataSet>
          <Name>DataSet1</Name>
          <RowsRead>18</RowsRead>
          <TotalTimeDataRetrieval>165</TotalTimeDataRetrieval>
          <ExecuteReaderTime>165</ExecuteReaderTime>
        </DataSet>
      </DataSets>
    </Connection>
  </Connections>
</AdditionalInfo>
```

The **AdditionalInfo** column can contain the following values:

- **ProcessingEngine.** The engine used to process the report. A value of 2 represents the newer on-demand processing engine (post SQL Server 2005).
- **ScalabilityTime**. The number of milliseconds required to scale operations for the pagination and processing components. If this value is 0, the request was not executed under memory pressure.
- **EstimatedMemoryUsageKB**. The estimated peak memory in kilobytes reported by the pagination and processing components.
- **DataExtension**. Each data extension or data source used in the report, including the number of occurrences per data source.
- **Connections**. An XML structure of information about the connection, including the data source name, the duration of the connection, the dataset name, the number of rows retrieved per dataset, the total retrieval time per dataset, and the *ExecuteReader* time per dataset.

Server Usage Statistics

To view server usage statistics, click **Statistics** in the SQL Reporting management portal ribbon. A dialog box like the one shown in Figure 40 displays a line chart for usage by hour. When you choose the week or month views, the chart also includes data for maximum usage per hour.


Server Usage Statistics

Server usage statistics allows you to understand the usage of SQL Reporting. The following graph represents the recent usage of this server in a scale of reports rendered at a specified time span. You can choose different time spans to compare the usage patterns throughout the last three month.

Figure 40: Server usage statistics

Report Server Properties

You can use SQL Server Management Studio (SSMS) to configure several properties for your SQL Reporting server that are not accessible in the SQL Reporting management portal. Some properties enable specific features, such as **My Reports**. Other properties define global values for report execution time-out or the amount of report history that can accumulate for a report.

When you open SSMS, select **Reporting Services** in the **Server Type** drop-down list in the **Connect To Server** dialog box. If SSMS is already open, click **Connect** in the **Object Explorer** window, and click **Reporting Services**. In the **Server Name** text box, type the URL for your SQL Reporting server, such as https://xyz123.reporting.windows.net/ReportServer. In the **Authentication** drop-down list, select **Basic Authentication**, type the user name and password for the SQL Reporting administrator account, and then click **Connect**. The report server then appears in the **Object Explorer** window, as shown in Figure 41, with separate folders available for managing report execution jobs, security, and shared schedules. To access report server properties, right-click the server node at the top of the tree, and click **Properties**.

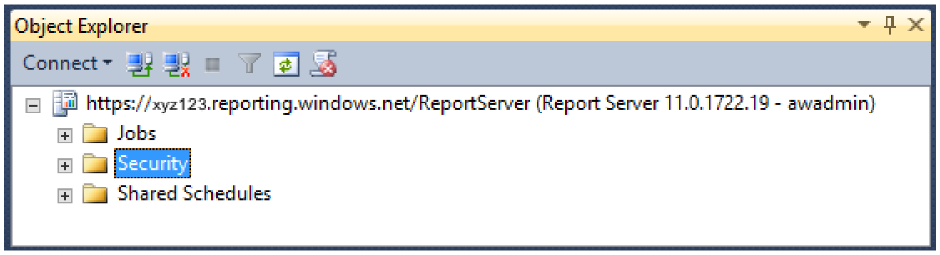
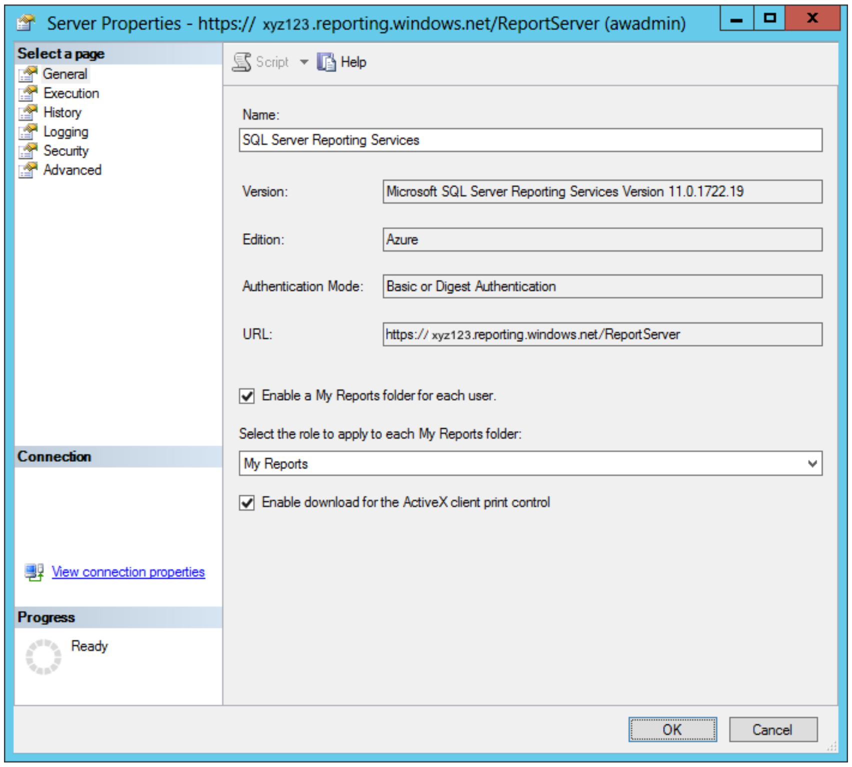

Object Explorer
Connect
https://xyz123.reporting.windows.net/ReportServer (Report Server 11.0.1722.19 - awadmin)
Jobs
Security
Shared Schedules

Figure 41: SQL Reporting server in the SSMS Object Explorer

Note: The Server Properties dialog box in SSMS displays all properties for SQL Server Reporting Services, whether the server is on-premises or hosted by the Windows Azure platform. However, most properties are applicable only to an on-premises server.

General Properties

The **General** page of the **Server Properties** dialog box shown in Figure 42 displays the current version and edition of SQL Reporting as well as the authentication mode and the URL for the Web service. By default, the **My Reports** folder feature is disabled, but the **ActiveX client** print control is enabled. The **My Reports** folder feature allows users to save reports to a personal folder on the server. The **ActiveX client** print control adds a button to the **HTML Viewer** toolbar when viewing a report in a browser. The **Print** button allows users to print the report they are viewing without first downloading the report to their computers.


Server Properties - https:// xyz123.reporting.windows.net/ReportServer (awadmin)
Select a page
General
Execution
History
Logging
Security
Advanced
Script
Help
Connection
View connection properties
Progress
Ready
OK
Cancel

Name: SQL Server Reporting Services

Version: Microsoft SQL Server Reporting Services Version 11.0.1722.19

Edition: Azure

Authentication Mode: Basic or Digest Authentication

URL: https:// xyz123.reporting.windows.net/ReportServer

☑ Enable a My Reports folder for each user.

Select the role to apply to each My Reports folder: My Reports

☑ Enable download for the ActiveX client print control

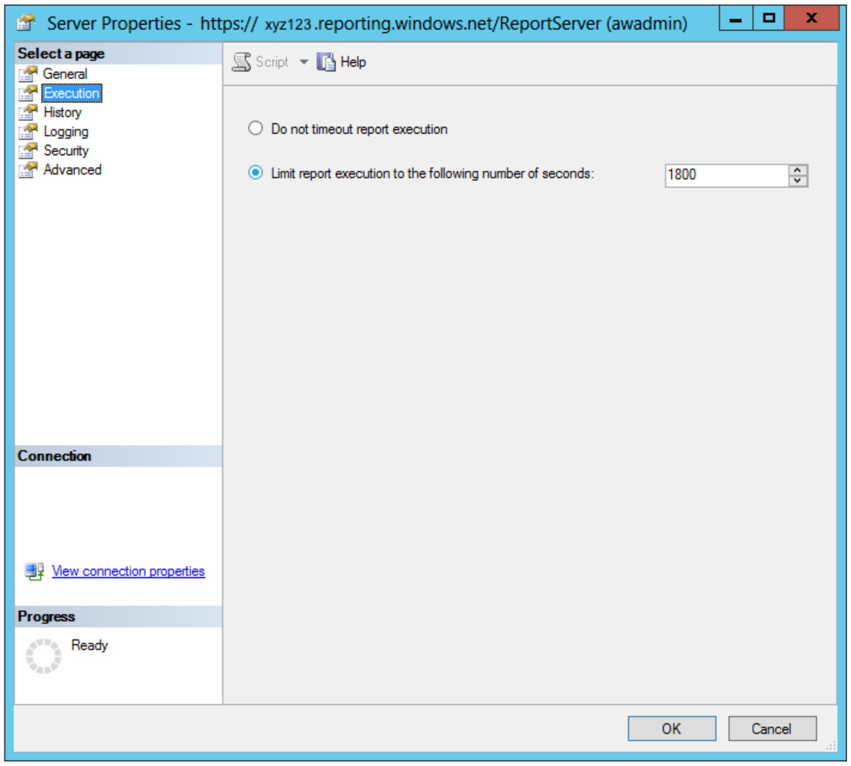
Figure 42: General SQL Reporting server properties

To enable the **My Reports** feature, select the **Enable a My Reports Folder For Each User** check box and then select a security role, such as **My Reports**, in the **Select The Role To Apply To Each My Reports Folder** drop-down list. The role you select gives users the functional equivalent of the Content Manager role, but only for the user's **My Reports** folder, as described in Chapter 7, Report Access. This permission allows users to add or delete content in a folder that only they can access.

You do have the option to disable the **ActiveX client** print control. If you disable the print control, users will not have the ability to print a report from the report page in the browser.

Execution Properties

The **Execution** page of the **Server Properties** dialog box allows you to control whether a time limit applies to report execution at the server level. The default setting is to limit report execution to 1800 seconds, as shown in Figure 43. If you expect some reports to require long execution times, you can switch to the **Do Not Timeout Report Execution** option. A report might time out because the query execution takes a long time or because a complex report requires longer processing and rendering times. You can edit a report's dataset properties to place a time-out on the query only.


Server Properties - https:// xyz123.reporting.windows.net/ReportServer (awadmin)
Select a page
General
Execution
History
Logging
Security
Advanced
Script
Help
Connection
View connection properties
Progress
Ready
OK
Cancel

Do not timeout report execution [✓ Limit report execution to the following number of seconds: 1800]

Figure 43: General SQL Reporting server properties

Chapter 5 Report Parts

Report parts are a useful way to centralize business logic and simplify maintenance for commonly used elements in a report. In this chapter, I review the benefits of using report parts, and then explain how you can create and deploy report parts from each of the report development tools. I also explain how to manage the report parts on the SQL Reporting Server, and how users interact with report parts.

Benefits of Report Parts

Often the same types of data regions are used in multiple reports, but in different combinations. Although you can use subreports to reuse the same layout, report parts offer greater flexibility for less technical users to construct their own reports in Report Builder because they can use a report part as a starting point for report development, but they can also make changes to the report part without affecting the original version.

Users can also mix and match multiple report parts to personalize their reports. For example, rather than deploy one report with a table and another report with a chart, one user can combine these two items as report parts in a single report with the table on the left and the chart on the right. Another user can create a separate report with the chart report part on the left and the table on the right.

Not only can users take advantage of report parts for report personalization, they can also use them as a way to ensure the underlying data is correct for the report. Users of report parts don’t need to know how to write T-SQL queries to retrieve data from SQL Database because the query is already built and associated with the report part.

Report Part Creation

A report part is a single report item or a collection of report items that you selectively publish to the SQL Reporting server for others to use in their reports. Not only is the report item saved, but the related data source and dataset are also saved with it. Furthermore, if the report part references a report parameter in a filter or text box expression, the corresponding report parameter is also saved with the report part. Consequently, a user does not need to know how to build these supporting elements, but can successfully create a working report simply by dropping the report part into the body of a report.

You can create a report part from any of the following report items:

- Data regions: table, matrix, list
- Graphical data regions: chart, data bar, sparkline, gauge, indicator, map
- Report items: rectangle, image
- Parameter

To create a report part, you develop a report in BIDS, SSDT, or Report Builder by creating one or more of these report items in your report. You must select the individual items you want to add to the SQL Reporting server as a report part. When you deploy the report containing these items, the report parts are created and stored on the server.

If you create a nested structure, such as a table inside a list, the entire structure becomes a single report part. You cannot publish these items as individual report parts. However, the user can delete the nested table and then add a different report item to the list. However, you might consider creating nested items as standalone report items, in which case you can publish them as separate report parts. In that case, the user can select the separate report items and then nest them if desired.

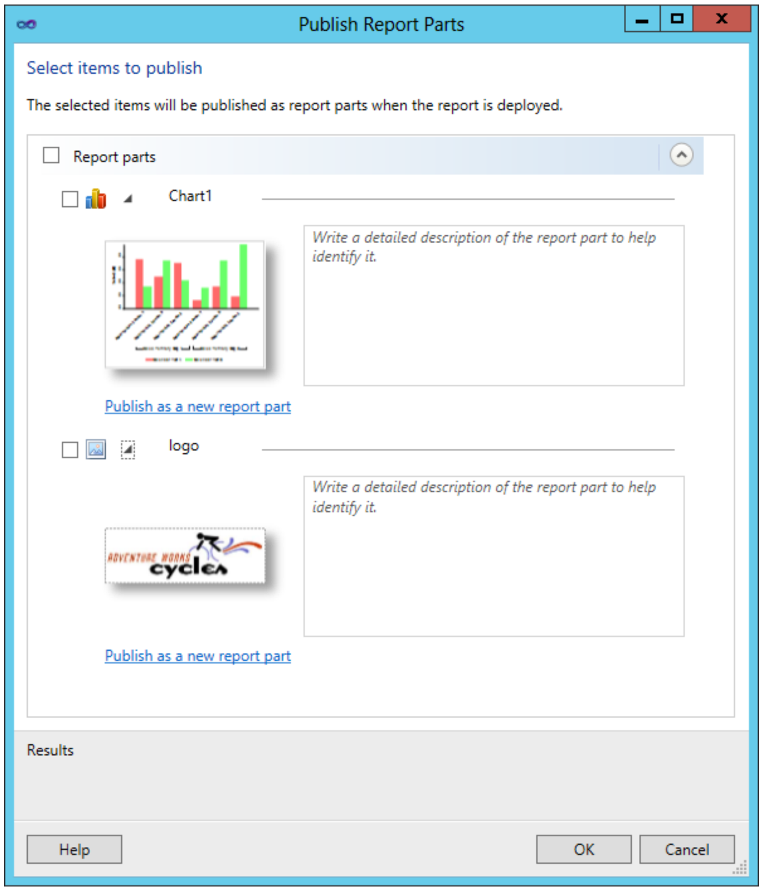
Report Part Deployment

An item in your report does not become a report part until you select it as an item to publish by using either BIDS, SSDT, or Report Builder. When publishing from BIDS or SSDT, the report parts deploy to a common folder that you define specifically for report parts in the project properties, but you can move report parts to other folders on the SQL Reporting server and configure security on them if you are a SQL Reporting administrator.

Note: To deploy report parts to the SQL Reporting server, the user name you use to publish the report parts must be assigned to the Content Manager or Publisher role, as described in Chapter 6, Security. If the user name is assigned to the My Reports role, you can also deploy report parts to your My Reports folder.

Deployment from Report Designer

In BIDS or SSDT, after you develop your report, you select report parts to publish by opening **Publish Report Parts** from the **Report** menu. You can see the report part candidates in the dialog box in a list. Expand each item to view its thumbnail image, as shown in Figure 44. You can assign a new name and a detailed description to help users locate the item when using the search feature in the **Report Part Gallery** available in **Report Builder**. For example, instead of deploying a report part as **Chart1**, you can provide a more descriptive name, such as **Sales by Territory By Year**, and you can place keywords and more information about the chart type, the fields it contains, its axes, series, and groupings in the description.


Publish Report Parts

Select items to publish

The selected items will be published as report parts when the report is deployed.

Report parts

Chart1

Write a detailed description of the report part to help identify it.

Publish as a new report part

logo

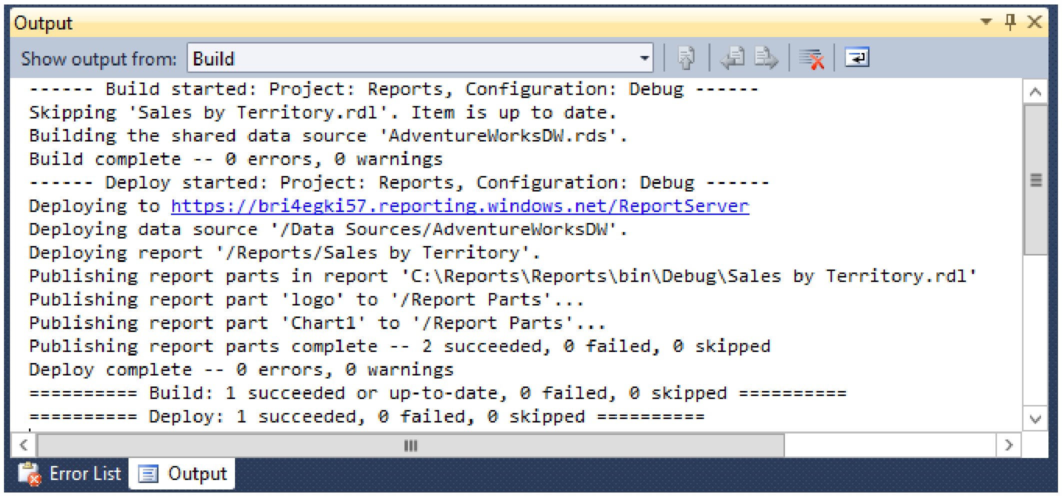
Write a detailed description of the report part to help identify it.

Publish as a new report part

Results

Figure 44: Report Part Selection

A report part deploys to the SQL Reporting server only when you select the check box for each item you want to deploy. You are not required to deploy all report items in the **Publish Report Parts** dialog box. After you select the desired report parts, you must also deploy the parent report to deploy the report parts. The output window, shown in Figure 45, confirms successful deployment of the report parts in addition to the parent report.


Output

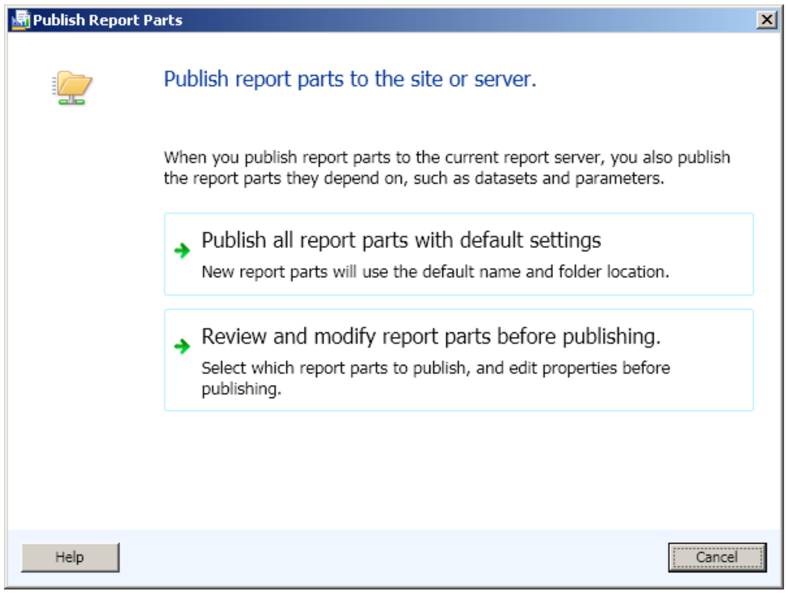
Show output from: Build

```
------ Build started: Project: Reports, Configuration: Debug ------
Skipping 'Sales by Territory.rdl'. Item is up to date.
Building the shared data source 'AdventureWorksDW.rds'.
Build complete -- 0 errors, 0 warnings
------ Deploy started: Project: Reports, Configuration: Debug ------
Deploying to https://bri4egki57.reporting.windows.net/ReportServer
Deploying data source '/Data Sources/AdventureWorksDW'.
Deploying report '/Reports/Sales by Territory'.
Publishing report parts in report 'C:\Reports\Reports\bin\Debug\Sales by Territory.rdl'
Publishing report part 'logo' to '/Report Parts'...
Publishing report part 'Chart1' to '/Report Parts'...
Publishing report parts complete -- 2 succeeded, 0 failed, 0 skipped
Deploy complete -- 0 errors, 0 warnings
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
========== Deploy: 1 succeeded, 0 failed, 0 skipped ==========
```

Figure 45: Deployment output

Deployment from Report Builder

Report Builder also allows you to deploy report parts. To do this, click **Report Builder** in the upper left corner and select **Publish Report Parts**. In the first page of the **Publish Report Parts** dialog box, shown in Figure 46, you have the option to publish all report parts by using the default settings or to review and modify report parts first. You should choose the **Review and modify report parts before publishing** option because it's better to assign the report parts clear names and add descriptions. When you choose this option, the second page of the **Publish Report Parts** dialog box is displayed, which is similar to the one you see in BIDS or SSDT. The difference in deploying from Report Builder is that you can optionally change the target folder for each report part.


Publish Report Parts

Publish report parts to the site or server.

When you publish report parts to the current report server, you also publish the report parts they depend on, such as datasets and parameters.

Publish all report parts with default settings
New report parts will use the default name and folder location.

Review and modify report parts before publishing.
Select which report parts to publish, and edit properties before publishing.

Figure 46: Report Builder's Publish Report Parts options

Report Part Redeployment

In any of the report development client tools, you can modify a report item that you have previously published as a report part. When you do this and then later open the **Publish Report Parts** dialog box, you see an additional option to deploy a new copy of the report part. In BIDS and SSDT, this option appears as a link entitled **Publish As A New Report Part**. When you click this link, a message box prompts you for confirmation to assign a new ID to the report part and displays a message that you cannot modify the prior version of the report part by continuing. If you click **Yes**, both versions of the report part are available on the SQL Reporting server, but only the last one published is available for modification in BIDS or SSDT. In Report Builder, instead of a link to click, you must select the **Publish As A New Copy Of The Report Part** check box to create a new version of the report part.

If you decide instead to redeploy your report part without creating a new version, the report part on the SQL Reporting Server is replaced with the new version. The next time a user opens a report containing that report part in Report Builder, an update notification will be displayed in the application window, as described later in this chapter in the Update Notification section.

Report Part Management

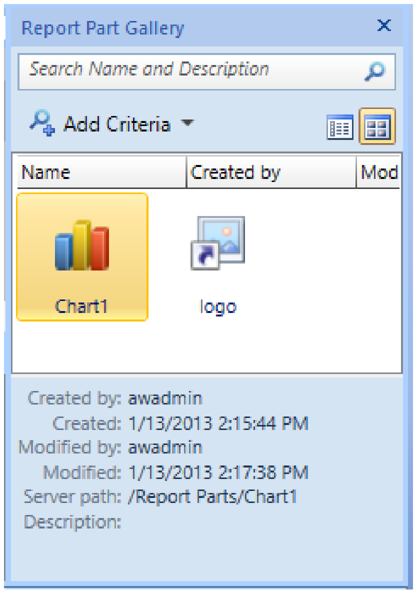
Once report parts are placed on the SQL Reporting server, you manage them like reports. You can browse the list of report parts in folders by using the Web service URL as described in Chapter 7, Report Access or by using the SQL Reporting management portal. However, you cannot click the report part link to view its contents.

You can configure security on report parts like you do for reports as described in Chapter 6, Security. A user must be assigned to the Browser or the Report Builder role to search for a report part in the Report Part Gallery and add it to a report in Report Builder.

Report Part Gallery

To add a report part to your report in Report Builder, go to the **Insert** tab of the ribbon and click **Report Parts**. The Report Part Gallery opens on the right side of the page, and includes a search box for you to locate a report part by a word or phrase in the report part name or description. You can view all available report parts by pressing **Enter** without typing a keyword, as shown in Figure 47.

Note: The Report Part Gallery is not available in BIDS or SSDT.



Created by: awadmin
Created: 1/13/2013 2:15:44 PM
Modified by: awadmin
Modified: 1/13/2013 2:17:38 PM
Server path: /Report Parts/Chart1
Description:

Figure 47: Report Part Gallery

You can further refine the scope of the search by clicking **Add Criteria** near the top of the Report Part Gallery. You can select one or more of the following criteria:

- **Created.** A date range.
- **Created By.** Original author name in domain\user format or a partial string.

- **Modified.** A date range.
- **Modified By.** Author of the most recent version in domain\user format or a partial string.
- **Server Folder.** An ellipsis button to navigate folders on the SQL Reporting server in search of report parts.
- **Type.** One or more report part types in a list, such as chart, rectangle, or tablix, used as a filter for the search.

When report parts match the criteria you specify, an icon is displayed for each report part when you use the **Thumbnail** view in the Report Part Gallery. If you prefer to see the list of report parts by name, click **Details** at the top of the Report Part Gallery. In either view, you can select a report part to view the information about the original author, the last author, the created and modified dates, the location of the report part on the report server, and the description, if one exists. To add it to your report, double-click it or drag it to the report design surface.

When you add the report part to your report, the related data source and dataset are also added to the report and appear in the **Report Data** pane. If the report part uses an embedded dataset, you can view the query and also change it if you like. However, the data source connection string is not visible because a report part requires a shared data source, which keeps the connection string secure.

You can use the report part's dataset to create other report items in your report. For example, if you add a chart from a report, you can use fields from its dataset to add a corresponding table. That way, even if users don't know how to add a data source or write a query, they can take advantage of prebuilt components to add more content to a report without relying solely on report parts.

Because the dataset is accessible in your report, regardless of whether it's shared or not, you can use it to add more report items to the report. For example, let's say you add a chart from a report part. You can resize it to make room for a matrix, and then use fields from a dataset and apply formatting. Without knowing how to add a data source or write a query, you can easily select a report part from the library, and then reuse the dataset for your own report item.

You can also change the report part's properties. For example, you can change the color palette of a chart if you like. Then you can deploy it to the server in its new form. You can replace the existing report part if you have the necessary permissions, or you can publish it as a new report part as described in the Report Part Deployment section earlier in this chapter. If you make a change to the report part, there is no obligation to update the report part on the report server. Your change can remain exclusive to your report.

Update Notification

If you have added a report part to your report that someone else subsequently modified and updated on the SQL Reporting server, you will see a notification the next time you open that report in Report Builder, as shown in Figure 48. If you already have Report Builder open and know that the report part you used has been redeployed to the server, you can click **Report Builder** and click **Check for Updates** on the menu to display the notification.

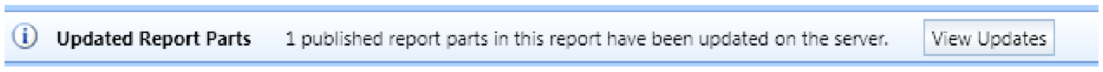

Updated Report Parts 1 published report parts in this report have been updated on the server.

Figure 48: Report parts update notification

You can click **View Updates** to see which report part is updated. However, you see only the name and description of the report part, the modified date, and the user who updated the report part. The only way to discover what type of change was made is to select the check box next to the report part name and click **Update** to apply the change to your report. If you decide you prefer the version that was in your report prior to the change, click **Undo** to restore it.

The final consideration is the difference in change management. With a report part, the user can edit the report part to personalize it or update the new report with the most current copy of the report part. It is important to note that the update can occur only when the user opens the report for editing in Report Builder, but not when rendering the report on the report server. A subreport always renders the most current version, without giving the user a choice. Furthermore, any changes to a subreport affect all reports that reference it, whereas changes to a report part affect only reports for which the report author applies the update.

Chapter 6 Security

Another important management task is implementation of security. You must add users to the SQL Reporting server and assign each user to a role. Then you map roles for report items, either at the folder level or the item level, to control what each user can see and do.

Role-Based Security

Before allowing users to access the SQL Reporting server, you should have a clear understanding of the role-based security model that SQL Reporting uses to authorize user activity on the server. There are two types of roles on the server:

- Item
- System

Item Roles

You use item roles to define permissions granting access to folders, reports, and other resources uploaded to the server such as images, shared datasets, report parts, and data sources. If you secure a folder, any content contained in that folder automatically has the same permissions. However, you can override inherited security and secure individual items by assigning different permissions.

SQL Reporting includes the following item roles:

- **Content Manager**. This role has permissions to see an item and to perform management tasks related to the item. This role can add an item to the SQL Reporting server and can configure security if the user is also a co-administrator of the subscription. Typically you set up a limited number of Content Managers on the server.
- **Publisher**. This role has permissions to see an item and add an item. However, the role does not have permissions to configure security. You usually assign this role to report developers.
- **Report Builder**. This role allows a user to open a report on the SQL Reporting server in Report Builder to edit the report.
- **My Reports**. With this role on an on-premises server, a user has permissions to save reports to a personal folder on the report server. However, the My Reports feature is not available in SQL Reporting at the time of this writing.
- **Browser**. Most report readers are assigned to the Browser role. This role can only view an item.

Note: To add a co-administrator to a subscription, go to the Settings page of the Windows Azure management portal, click the Administrators link at the top of the page, click Add on the ribbon at the bottom of the page, type the email address of the new administrator, and select a subscription.

System Roles

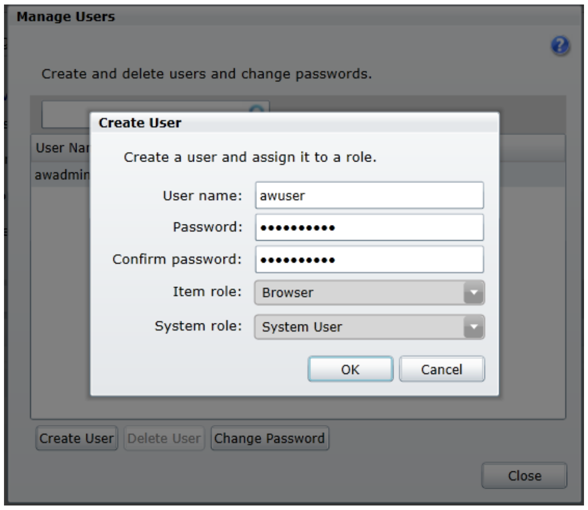
You use system role assignments to enable users to perform administrative tasks on the report server. You assign each user to one of the following roles:

- **System Administrator**. A user assigned to this role can connect to the SQL Reporting management portal to perform administrative tasks. The user can also connect to the SQL Reporting server in SSMS to view and update server properties.
- **System User**. Most users are assigned to this role. Users in this role cannot connect to the SQL Reporting management portal.

Users

After the SQL Reporting server is set up, only the server administrator has permissions to access the server's management portal and to deploy reports. Report readers initially have no access. You must create a user account for each report reader to grant access to the server and assign the user to item and system roles.

In the management portal, click **Manage** and then, in the Manage Users dialog box, click **Create User**. Type a name and password, and then select the applicable roles. Typically, you assign report readers to the Browser item role and the System User system role as shown in Figure 49. The item role assignment is the default role for the user, which you can override for individual items on the SQL Reporting server.


Manage Users

Create and delete users and change passwords.

Create User

Create a user and assign it to a role.

User name: awuser

Password: ••••••••••

Confirm password: ••••••••••

Item role: Browser

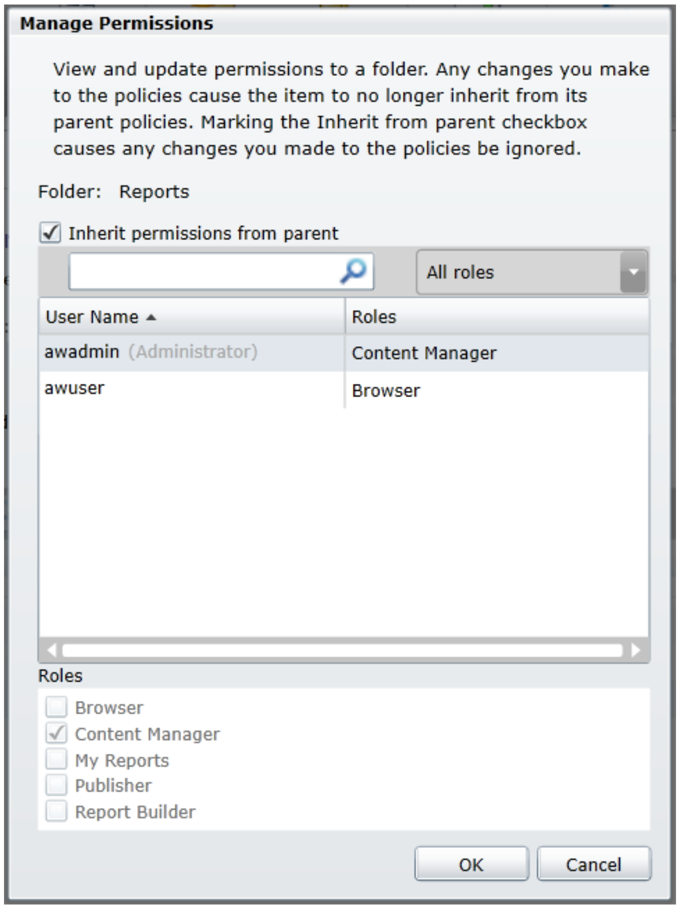
System role: System User

Figure 49: User role assignments

Report Server Item Permissions

The item role assignments for users automatically apply to each item on the SQL Reporting server. That is, a user assigned to the Browser role can view all folders and the contents of each folder. However, you can override these role assignments at the folder level or item level. To do this, click the arrow next to the item you want to secure in the SQL Reporting management folder, and select Permissions.

By default, the Inherit Permissions From Parent check box is selected, as shown in Figure 50. In the Manage Permissions dialog box, you can see the list of users already assigned to a role for the current item and the role assignment. To override the role assignment, clear the Inherit Permissions From Parent check box if necessary, select a user, and then select the new role's check box.


Manage Permissions

View and update permissions to a folder. Any changes you make to the policies cause the item to no longer inherit from its parent policies. Marking the Inherit from parent checkbox causes any changes you made to the policies be ignored.

Folder: Reports

✓ Inherit permissions from parent

All roles

User Name ▲	Roles
awadmin (Administrator)	Content Manager
awuser	Browser

Roles

- [ ] Browser
- [✓] Content Manager
- [ ] My Reports
- [ ] Publisher
- [ ] Report Builder

Figure 50: Report item permissions

You can assign users to different roles for different content on the SQL Reporting server. For example, you might have all users assigned to the Browser role on the top-level folder. Then you can assign the users in the sales department and the finance department to the Browser role for the folders for their respective departments. Then for each folder, you can assign different users to the Content Manager role. For example, as shown in Figure 51, you can designate John as the content manager of the top-level folder, Amy as the content manager of the Sales folder, and Linda as the content manager of the Finance folder.

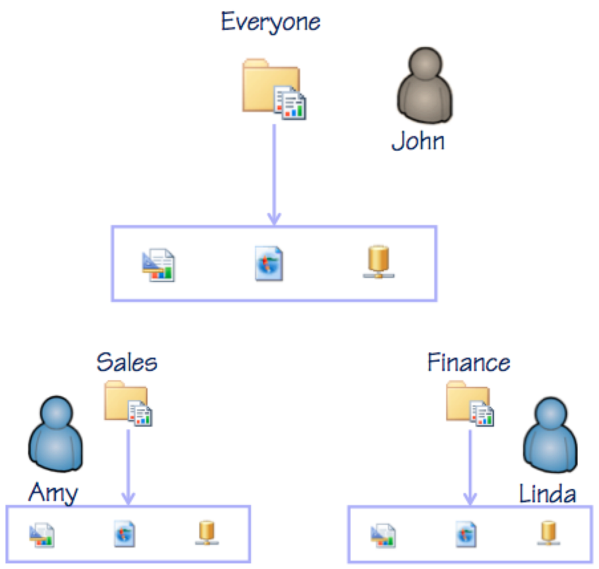


Figure 51: Report items and role assignments

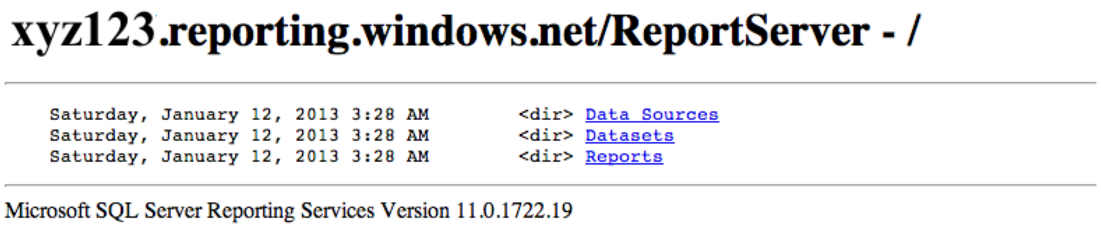
Note: Any user assigned to the Content Manager role for a data source (or its parent folder if it inherits security) can view the connection string it contains and the user account used to connect to the SQL Database. However, the password is not visible. Nonetheless, you should keep this information secure. Review permissions for each data source and limit the Content Manager role assignment to as few people as possible.

Chapter 7 Report Access

After content is uploaded, user accounts are added, and permissions are assigned, the SQL Reporting server is ready for report readers to use. Report readers are users who have the fewest privileges on a SQL Reporting server. The simplest way for this group of users to access a report is to use the SQL Reporting report server link to browse the available content and click a report's link to view it. Another option is to provide a report reader with a URL to a specific report, which can also include commands to control the rendering experience. Your third option is to have the report reader open an application that uses the ReportViewer control to display a report. In this chapter, I explain these options from the perspective of a report reader.

Web Service URL

To browse the contents of the SQL Reporting server, you must have the URL for the server and credentials to log into the server. A report server administrator provides you with the server URL which looks similar to this: https://xyz123.reporting.windows.net:443/reportserver. When you open this URL in your web browser, you are prompted for the user name and password assigned to you. After you successfully enter your credentials, the top-level items are displayed in your browser as shown in Figure 52. You can only view items for which you have been granted permissions.



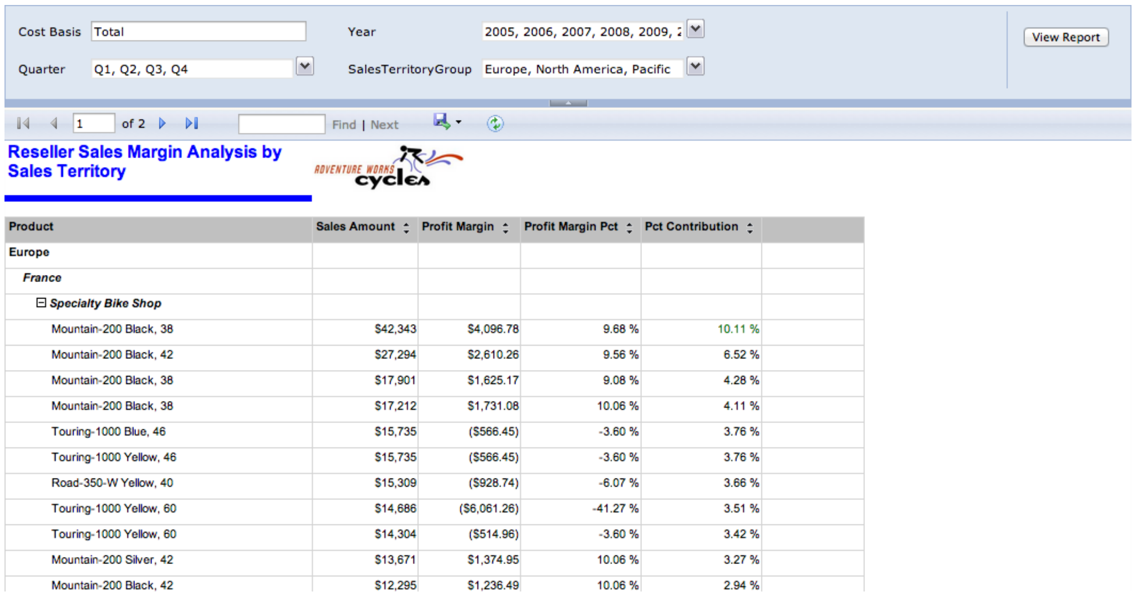
xyz123.reporting.windows.net/ReportServer - /

```
Saturday, January 12, 2013 3:28 AM        <dir> Data Sources
Saturday, January 12, 2013 3:28 AM        <dir> Datasets
Saturday, January 12, 2013 3:28 AM        <dir> Reports
```

Microsoft SQL Server Reporting Services Version 11.0.1722.19

Figure 52: SQL Reporting server home directory

Click the directory links to view the item they contain or click a report link to display the report as shown in Figure 53. By default, the report renders in HTML format and includes the HTML Viewer toolbar above the report. This toolbar contains buttons that you use to navigate the report, search for text within the report, export the report to a new format, or refresh the report. If the report contains report parameters, they are displayed above the HTML Viewer toolbar. After you change parameter values, click **View Report** to update the report with the new values.



Cost Basis: Total
Year: 2005, 2006, 2007, 2008, 2009, 2
Quarter: Q1, Q2, Q3, Q4
SalesTerritoryGroup: Europe, North America, Pacific

Reseller Sales Margin Analysis by Sales Territory

Product	Sales Amount	Profit Margin	Profit Margin Pct	Pct Contribution
Europe				
France				
⊟ ***Specialty Bike Shop***				
Mountain-200 Black, 38	$42,343	$4,096.78	9.68 %	10.11 %
Mountain-200 Black, 42	$27,294	$2,610.26	9.56 %	6.52 %
Mountain-200 Black, 38	$17,901	$1,625.17	9.08 %	4.28 %
Mountain-200 Black, 38	$17,212	$1,731.08	10.06 %	4.11 %
Touring-1000 Blue, 46	$15,735	($566.45)	-3.60 %	3.76 %
Touring-1000 Yellow, 46	$15,735	($566.45)	-3.60 %	3.76 %
Road-350-W Yellow, 40	$15,309	($928.74)	-6.07 %	3.66 %
Touring-1000 Yellow, 60	$14,686	($6,061.26)	-41.27 %	3.51 %
Touring-1000 Yellow, 60	$14,304	($514.96)	-3.60 %	3.42 %
Mountain-200 Silver, 42	$13,671	$1,374.95	10.06 %	3.27 %
Mountain-200 Black, 42	$12,295	$1,236.49	10.06 %	2.94 %

Figure 53: Report in HTML Format

Note: If you are the SQL Reporting administrator, you can also navigate the folder hierarchy in the SQL Reporting management portal and click a report to open a new tab in your browser. You must type user credentials before you can view the report in the browser, as shown in Figure 53.

Rendering Formats

When users want to view reports but are unable to connect to the Internet, or when they want to use the information in a report in different ways, they can export the report to an alternate rendering format. SQL Reporting supports seven different rendering formats as alternatives to the online HTML format used to display reports in the browser. These seven formats are organized into three types of renderers:

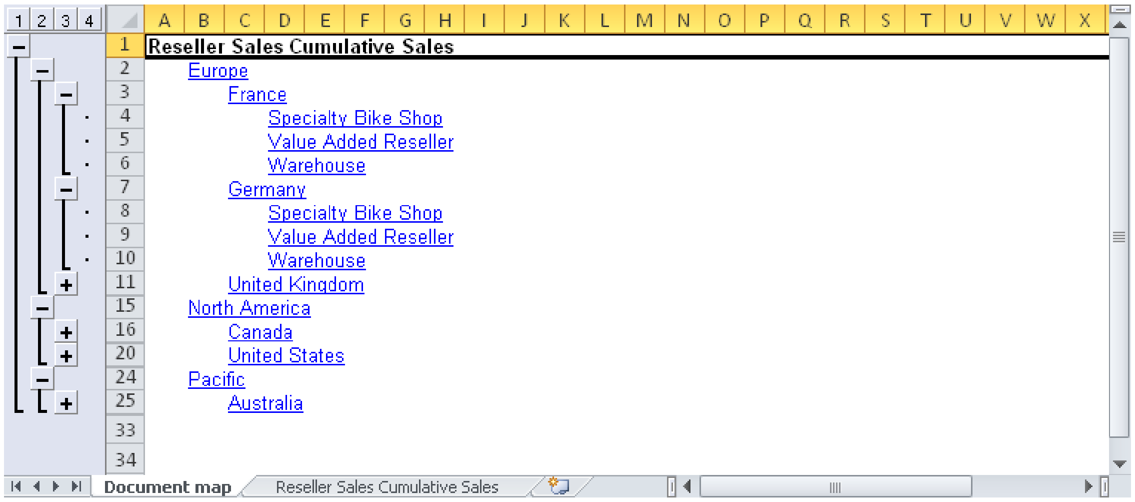
- **Soft Page-Break**. The document adheres to the report layout in the RDL and is optimized for viewing online: Excel, Word, and MHTML.
- **Hard Page-Break**. The document adheres to the report layout in the RDL and is optimized for print: TIFF and PDF. With these formats, you should check the Width, Height, and Margin properties are set correctly.
- **Data**. The output of the renderer is raw data and excludes formatting defined for the report layout: CSV, XML.

Excel

This export to Excel produces a file with an XLSX extension which is compatible with Excel 2007 and newer versions. It produces a single cell for each text box, chart, sparkline, data bar, gauge, indicator, map, and image in the report, but uses multiple cells to render data regions, subreports, rectangles, and the report body. In some cases, the render merges multiple cells when report item edges are not aligned. Background images for report items are ignored unless an image is assigned to the BackgroundImage property for the body of the report.

In most cases, formatting properties are preserved in the worksheet wherever possible. If you do not define pagination properties in the report, SQL Reporting renders the report as a single worksheet in the Excel workbook. On the other hand, if you define rules for pagination, each page break based on a group generates a separate worksheet in the XLSX file. You can name each sheet by configuring the PageName property of the group.

When an Excel workbook renders, the items contained in the page header render in the uppermost rows of a worksheet. By contrast, page footers become part of the Excel footer. A document map is added to the workbook as the first worksheet and includes document map links, as shown in Figure 54, which allows you to jump to a specific location in the Excel workbook.



1	Reseller Sales Cumulative Sales
2	Europe
3	France
4	Specialty Bike Shop
5	Value Added Reseller
6	Warehouse
7	Germany
8	Specialty Bike Shop
9	Value Added Reseller
10	Warehouse
11	United Kingdom
15	North America
16	Canada
20	United States
24	Pacific
25	Australia
33	
34	

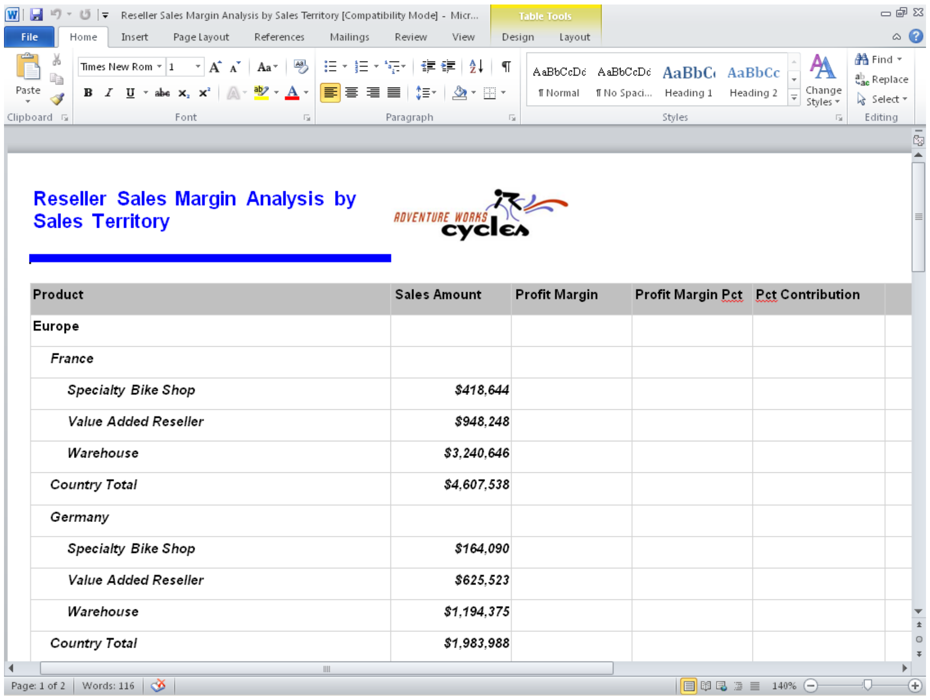
Figure 54: Document map links in Excel

Note: The Excel renderer supports a maximum of seven levels in a document map.

The Excel renderer also supports hyperlink and bookmark actions, but does not support interactive sorting. Also, expressions resolve as constant values in the workbook. The renderer produces charts as images and does not include the underlying data.

Word

This renderer produces a DOCX file, as shown in Figure 55, and preserves many interactive features, such as hyperlinks and actions. However, it does not render a document map as you might expect. Instead, it creates Word table of contents labels from the document labels, but you must use the **References** tab on the Word ribbon to create the table of contents in the document. Furthermore, the Word renderer behaves like Excel by creating static images for graphical data regions and images. However, be aware that it excludes background images in the report or the report body.



Reseller Sales Margin Analysis by Sales Territory

Product	Sales Amount	Profit Margin	Profit Margin Pct	Pct Contribution
Europe				
France				
Specialty Bike Shop	*$418,644*			
Value Added Reseller	*$948,248*			
Warehouse	*$3,240,646*			
Country Total	*$4,607,538*			
Germany				
Specialty Bike Shop	*$164,090*			
Value Added Reseller	*$625,523*			
Warehouse	*$1,194,375*			
Country Total	*$1,983,988*			

Figure 55: Viewing a report rendered as a Word document

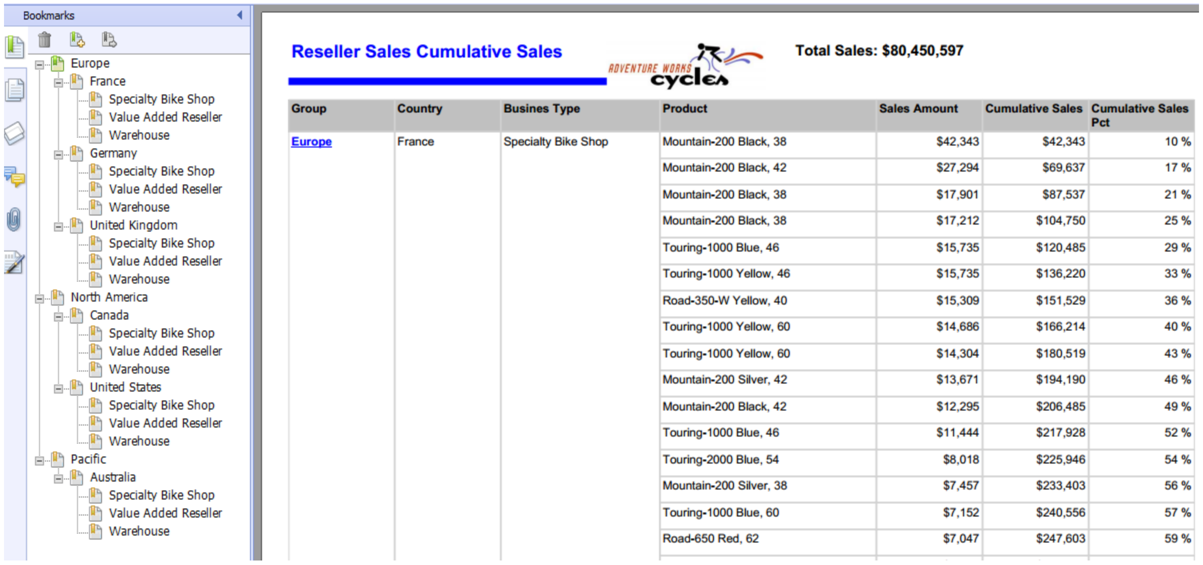
The Word renderer converts the page header and page footer in the report to a document header and footer, but ignores the **Height** property for these items. It also uses the report, body, and margin size settings that you define for the report, but the maximum width of a report is 22 inches.

MHTML

When you open a report from the browser or the SQL Reporting management portal, the report renders as HTML by default and all interactive features are enabled. If explicit page breaks are defined in the report or if the report is very large, you must use the navigation buttons in the **HTML Viewer** toolbar to move from page to page. If you want to save this report as a single page or embed it in an email message, you can save it as an MHTML file. When you do this, you lose the ability to toggle visibility. You must click the toggle item in the HTML version of the report before you perform the export to change the visibility state.

PDF

The PDF renderer produces a report that matches what you see online. That is, any report items in a hidden state are not visible in the PDF version of the report. Furthermore, the PDF file does not support interactive features, such as toggling visibility, interactive sort, bookmarks, and drillthrough. On the other hand, it does support a document map, as shown in Figure 56, and the **Go To URL** action.



- Europe
  - France
    - Specialty Bike Shop
    - Value Added Reseller
    - Warehouse
  - Germany
    - Specialty Bike Shop
    - Value Added Reseller
    - Warehouse
  - United Kingdom
    - Specialty Bike Shop
    - Value Added Reseller
    - Warehouse
- North America
  - Canada
    - Specialty Bike Shop
    - Value Added Reseller
    - Warehouse
  - United States
    - Specialty Bike Shop
    - Value Added Reseller
    - Warehouse
- Pacific
  - Australia
    - Specialty Bike Shop
    - Value Added Reseller
    - Warehouse

Reseller Sales Cumulative Sales

Total Sales: $80,450,597

Group	Country	Busines Type	Product	Sales Amount	Cumulative Sales	Cumulative Sales Pct
Europe	France	Specialty Bike Shop	Mountain-200 Black, 38	$42,343	$42,343	10 %
			Mountain-200 Black, 42	$27,294	$69,637	17 %
			Mountain-200 Black, 38	$17,901	$87,537	21 %
			Mountain-200 Black, 38	$17,212	$104,750	25 %
			Touring-1000 Blue, 46	$15,735	$120,485	29 %
			Touring-1000 Yellow, 46	$15,735	$136,220	33 %
			Road-350-W Yellow, 40	$15,309	$151,529	36 %
			Touring-1000 Yellow, 60	$14,686	$166,214	40 %
			Touring-1000 Yellow, 60	$14,304	$180,519	43 %
			Mountain-200 Silver, 42	$13,671	$194,190	46 %
			Mountain-200 Black, 42	$12,295	$206,485	49 %
			Touring-1000 Blue, 46	$11,444	$217,928	52 %
			Touring-2000 Blue, 54	$8,018	$225,946	54 %
			Mountain-200 Silver, 38	$7,457	$233,403	56 %
			Touring-1000 Blue, 60	$7,152	$240,556	57 %
			Road-650 Red, 62	$7,047	$247,603	59 %

Figure 56: Viewing a report rendered as a PDF document

TIFF

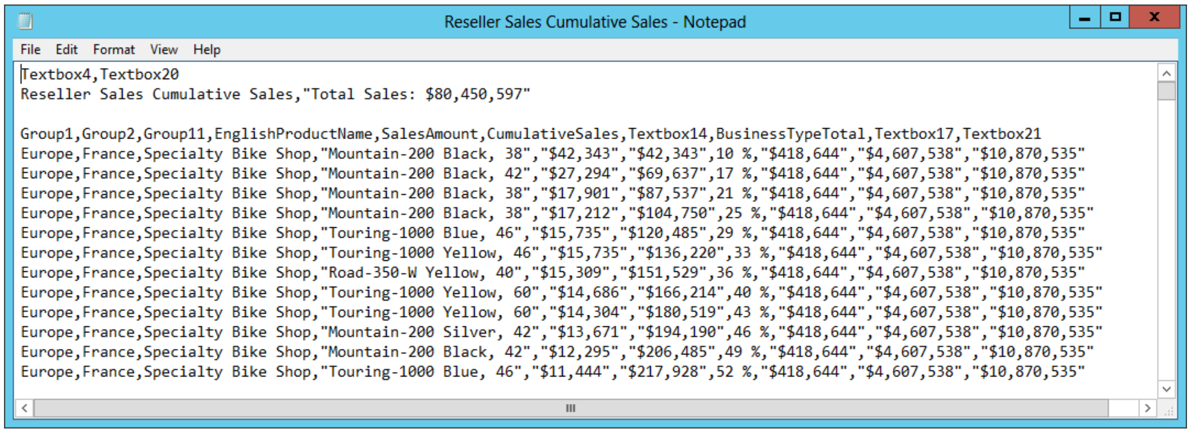
You can render a report as a multiple page image file in the Tagged Image File Format (TIFF). Like the PDF renderer, the TIFF renderer does not support interactive features and renders only the current view of the report.

CSV

The CSV renderer converts the data in your report data to the lowest level of detail. It also adds a comma delimiter between each field and a carriage return and line feed at the end of each record. All detail rows are included in the file, including rows that are hidden in the HTML version of the report. The column names are listed in a header row of the file. Be aware that the following items do not render: page header, page footer, lines, images, rectangles, or totals calculated by using expressions in a group or tablix footer. Likewise, interactive features are excluded.

Each item in the report renders as a separate group of rows and columns in the output file, as shown in Figure 57. Each report item type has its own rendering behavior, as follows:

- **Freestanding text box**. A freestanding text box renders with a column header and a row value of its contents.
- **Table**. A table includes a row and column at the lowest level of detail it contains, which is not necessarily the level of detail visible in the report. Total rows or columns in footers are not included.
- **Matrix**. A matrix includes a row and column at the lowest level of detail it contains, which is not necessarily the level of detail visible in the report. Total rows or columns in footers are not included.
- **List**. A list includes a row for each detail row or group instance it contains. If the list contains nested items, the parent group instance value repeats in each row of the nested item.
- **Subreport**. If the subreport is nested inside another report item, the parent item values repeat for each row of the subreport contents.
- **Chart**. A chart includes a row for each combination of value and category, typically represented as a single data point in the chart. If the chart contains multiple series, the label for each series is included in the rows preceding the value for that series. Data bars and sparklines have the same rendering behavior, but do not include series labels.
- **Gauge**. A gauge includes a single row containing the scale's minimum and maximum values, the range's start and end values, and the pointer value.
- **Indicator**. An indicator includes a single row with the current indicator state, available states, and the indicator value.
- **Map**. A map includes a row for each spatial data value and its corresponding label.


Reseller Sales Cumulative Sales - Notepad
File Edit Format View Help

```
Textbox4,Textbox20
Reseller Sales Cumulative Sales,"Total Sales: $80,450,597"

Group1,Group2,Group11,EnglishProductName,SalesAmount,CumulativeSales,Textbox14,BusinessTypeTotal,Textbox17,Textbox21
Europe,France,Specialty Bike Shop,"Mountain-200 Black, 38","$42,343","$42,343",10 %,"$418,644","$4,607,538","$10,870,535"
Europe,France,Specialty Bike Shop,"Mountain-200 Black, 42","$27,294","$69,637",17 %,"$418,644","$4,607,538","$10,870,535"
Europe,France,Specialty Bike Shop,"Mountain-200 Black, 38","$17,901","$87,537",21 %,"$418,644","$4,607,538","$10,870,535"
Europe,France,Specialty Bike Shop,"Mountain-200 Black, 38","$17,212","$104,750",25 %,"$418,644","$4,607,538","$10,870,535"
Europe,France,Specialty Bike Shop,"Touring-1000 Blue, 46","$15,735","$120,485",29 %,"$418,644","$4,607,538","$10,870,535"
Europe,France,Specialty Bike Shop,"Touring-1000 Yellow, 46","$15,735","$136,220",33 %,"$418,644","$4,607,538","$10,870,535"
Europe,France,Specialty Bike Shop,"Road-350-W Yellow, 40","$15,309","$151,529",36 %,"$418,644","$4,607,538","$10,870,535"
Europe,France,Specialty Bike Shop,"Touring-1000 Yellow, 60","$14,686","$166,214",40 %,"$418,644","$4,607,538","$10,870,535"
Europe,France,Specialty Bike Shop,"Touring-1000 Yellow, 60","$14,304","$180,519",43 %,"$418,644","$4,607,538","$10,870,535"
Europe,France,Specialty Bike Shop,"Mountain-200 Silver, 42","$13,671","$194,190",46 %,"$418,644","$4,607,538","$10,870,535"
Europe,France,Specialty Bike Shop,"Mountain-200 Black, 42","$12,295","$206,485",49 %,"$418,644","$4,607,538","$10,870,535"
Europe,France,Specialty Bike Shop,"Touring-1000 Blue, 46","$11,444","$217,928",52 %,"$418,644","$4,607,538","$10,870,535"
```

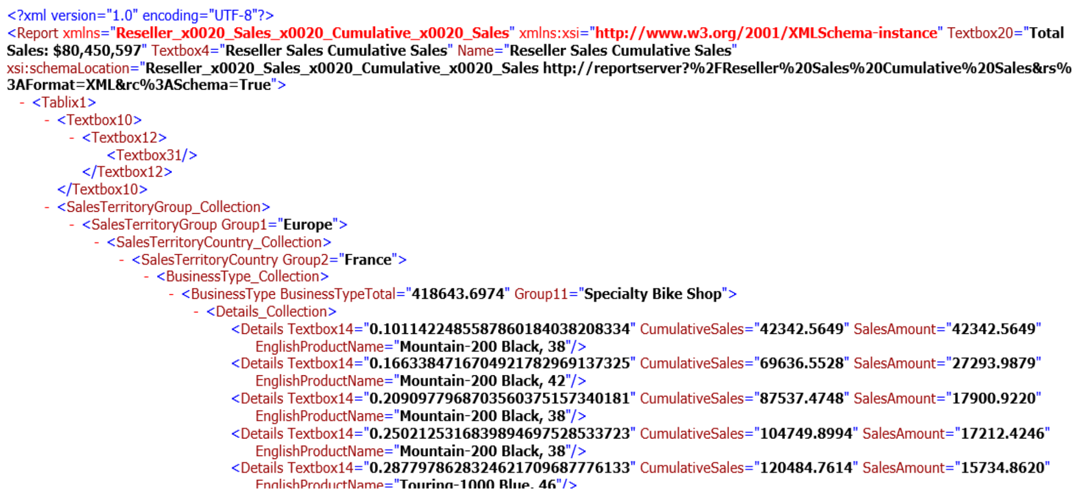
Figure 57: Viewing a report rendered as a CSV file

XML

The XML renderer exports data in a hierarchical structure, as shown in Figure 58. The names of report items in the report definition become XML elements in the exported file, so consider renaming report items from a default name like Textbox1 to a more descriptive name. The XML file includes all detail rows, including hidden data if the row can be toggled to a visible state. Each report item type has its own rendering behavior, as follows:

- **Report**. The report is the top-level element in the XML file.

- **Freestanding text box**. A freestanding text box renders as an attribute in the element of its parent container.
- **Data region and rectangle**. In this context, a data region is a table, matrix, list, chart, data bar, or sparkline. Data regions and rectangles render as child elements of their parent containers. In a chart, each series is a child element of the chart.
- **Group and details**. Each instance renders as a child element of its parent data region. Group instances include both row groups and column groups.
- **Text box inside data region**. A text box that is not freestanding renders as a child element of its parent data region.
- **Map**. The map itself renders as a child element of its parent container. Each map layer renders as a child element of the map and includes elements for each spatial member.
- **Gauge**. A gauge renders as an element of its parent container and renders the scale's minimum and maximum values, the range's start and end values, and the pointer value as attributes of the gauge element.
- **Indicator**. An indicator renders as an element of its parent container and renders the current indicator state, available states, and the indicator value as attributes of the indicator element.



```
<?xml version="1.0" encoding="UTF-8"?>
<Report xmlns="Reseller_x0020_Sales_x0020_Cumulative_x0020_Sales" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Textbox20="Total
Sales: $80,450,597" Textbox4="Reseller Sales Cumulative Sales" Name="Reseller Sales Cumulative Sales"
xsi:schemaLocation="Reseller_x0020_Sales_x0020_Cumulative_x0020_Sales http://reportserver?%2FReseller%20Sales%20Cumulative%20Sales&rs%
3AFormat=XML&rc%3ASchema=True">
 - <Tablix1>
   - <Textbox10>
     - <Textbox12>
         <Textbox31/>
       </Textbox12>
     </Textbox10>
   - <SalesTerritoryGroup_Collection>
     - <SalesTerritoryGroup Group1="Europe">
       - <SalesTerritoryCountry_Collection>
         - <SalesTerritoryCountry Group2="France">
           - <BusinessType_Collection>
             - <BusinessType BusinessTypeTotal="418643.6974" Group11="Specialty Bike Shop">
               - <Details_Collection>
                   <Details Textbox14="0.10114224855878601840382083334" CumulativeSales="42342.5649" SalesAmount="42342.5649"
                     EnglishProductName="Mountain-200 Black, 38"/>
                   <Details Textbox14="0.16633847167049217829691373325" CumulativeSales="69636.5528" SalesAmount="27293.9879"
                     EnglishProductName="Mountain-200 Black, 42"/>
                   <Details Textbox14="0.2090977968703560375157340181" CumulativeSales="87537.4748" SalesAmount="17900.9220"
                     EnglishProductName="Mountain-200 Black, 38"/>
                   <Details Textbox14="0.25021253168398946975285337223" CumulativeSales="104749.8994" SalesAmount="17212.4246"
                     EnglishProductName="Mountain-200 Black, 38"/>
                   <Details Textbox14="0.2877978628324621709687776133" CumulativeSales="120484.7614" SalesAmount="15734.8620"
                     EnglishProductName="Touring-1000 Blue, 46"/>
```

Figure 58: Viewing a report rendered as an XML file

My Reports Folder

The **My Reports** folder is a user-specific folder available for storing reports. Only the named user and report server administrators have access to an individual's **My Reports** folder. The **My Reports** feature is disabled on the server by default and must be enabled by a report administrator before use, as described in Chapter 4, Report Management. The **My Report** folder is visible when browsing the server through the Home page of the **Report Manager** if the feature is enabled and if you have been assigned to the My Reports role (or the role assigned to users of this feature).

After your **My Reports** folder is added, you can add content to it in the same way that content managers add content to other folders on the report server. That way, you can keep your favorite reports in one location rather than searching through the folder hierarchy on the SQL Reporting server.

If you are an SQL Reporting administrator, you can see the contents of any user's **My Reports** folder. You can move reports from this folder, and delete reports if necessary.

URL Access

Rather than browse the SQL Reporting server's folder directories in your browser, you might have access to a portal that includes direct links to specific reports through URL access. The link can include additional commands to change parameter default values, hide some or all of the HTML Viewer toolbar, or export the report to a different format. Role assignments remain in effect with URL access, so you cannot open a report if you do not have adequate permissions. You might store this URL in a document for quick access to a report that you reference frequently or you might be responsible for constructing a portal page with URL access links.

Report URL

You can locate a report's URL by first browsing the SQL Reporting server as described earlier in this chapter. Click the report's link and then review the URL in the browser's address bar. For example, the URL for the report shown in Figure 38 looks like this:
`https://xyz123.reporting.windows.net/ReportServer/Pages/ReportViewer.aspx?%2fReports%2fReseller+Sales+Margin+Analysis+by+Sales+Territory&rs:Command=Render`

You can copy this URL and use it in a document, email, or a webpage.

Tip: You can also abbreviate a report URL by eliminating the /Pages/ReportViewer.aspx and &rs:Command=Render portions of the URL. You can also eliminate the encoding by replacing the %2f and + characters with forward slashes and spaces, respectively.

URL Access Parameters

You are not limited to using the default settings for a report's rendering, toolbar, or parameters. Instead, you can append URL access parameters that provide additional options.

Report Parameters

For each parameter that you want to override in a report, you include a name/value pair as a string and append the string to the report's URL. If the parameter accepts multiple values, you append multiple name/value pairs like this:

```
https://xyz123.reporting.windows.net/ReportServer/Reports/Reseller Sales
Margin Analysis by Sales Territory&Quarter=1&Quarter=2
```

Note: Be sure to use the parameter's value and not its label in the name/value pair.

HTML Viewer Parameters

You can also use URL access parameters to modify the controls that are displayed in the HTML Viewer. For example, you might decide to exclude the Parameters section of the HTML Viewer because you provide parameters values in the URL. Table 5 lists the available commands to append to the URL.

Table 5: HTML Viewer Parameters

URL Access Parameter	Description	Example
DocMap	Hide a document map.	`&rc:DocMap=false`
FallbackPage	Display the specified page if a search fails.	`&rc:FallbackPage=1`
FindString	Search for a string in a report using *StartFind* and *EndFind* to specify the range of pages to search.	`&rc:FindString=Mountain-200&rc:StartFind=1&rc:EndFind=5`
Parameters	Hide parameters section of the toolbar.	`&rc:Parameters=false`
Section	Display the specified page.	`&rc:Section=5`
Toolbar	Hide the HTML Viewer toolbar.	`&rc:Toolbar=false`
Zoom	Increase or decrease the report size by the specified percentage or fit defined by *Page Width* or *Whole Page.*	`&rc:Zoom=50` or `&rc:Zoom=Page Width`

Report Server Parameters

Another way to use URL access parameters is to issue commands to the report server. You might do this when you want to export the report to a format other than HTML by using the **Format** parameter. Not all report server parameters display a report, though. For example, you can display report server contents by using the Command parameter. Table 6 lists the available parameters.

Table 6: Report Server Parameters

URL Access Parameter	Description	Example
ClearSession	Remove a report from the current report session (including all executions of the report with different parameter values).	`&rs:ClearSession=true`
Command	Perform an operation on a catalog item: • ListChildren to list child items • Render to render a report • GetSharedDatasetDefinition to show the XML for a shared dataset (if you have Read Report Definition permission) • GetDataSourceContents to show the data source properties • GetResourceContents to render a file (not a report) in an HTML page • GetComponentDefinition to display the RDL for a report (if you have Read Contents permission)	`&rs:Command=ListChildren`
Format	Render a report in the specified format (default HTML4.0)	`&rs:Format=excelopenxml`
GetNextStream	Get the next chunk of a persisted stream (default=false).	`&rs:GetNextStream=true`

URL Access Parameter	Description	Example
ParameterLanguage	Define the language for parameters independent of the browser language.	`&rs:ParameterLanguage=en-us`
PersistStreams	Render a report in a single persisted stream or one chunk at a time (default=false).	`&rs:PersistStreams=true`
SessionID	Specify an active report session.	`&rs:SessionID= 1lvwaoji1ok1sdyujn1dg5un`
ShowHideToggle	Toggle the visibility of a section of the report.	`&rs:ShowHideToggle=1`