ENESYS
13, rue Camille Desmoulins
92130 ISSY-LES-MOULINEAUX

Telephone: +33 (1) 58 04 27 93
Fax : +33 (1) 58 04 23 00
E-mail: contact@enesys.fr

Enesys RS Data Extension 1.4.0

User Manual

Table of Contents

About Enesys RS Data Extension

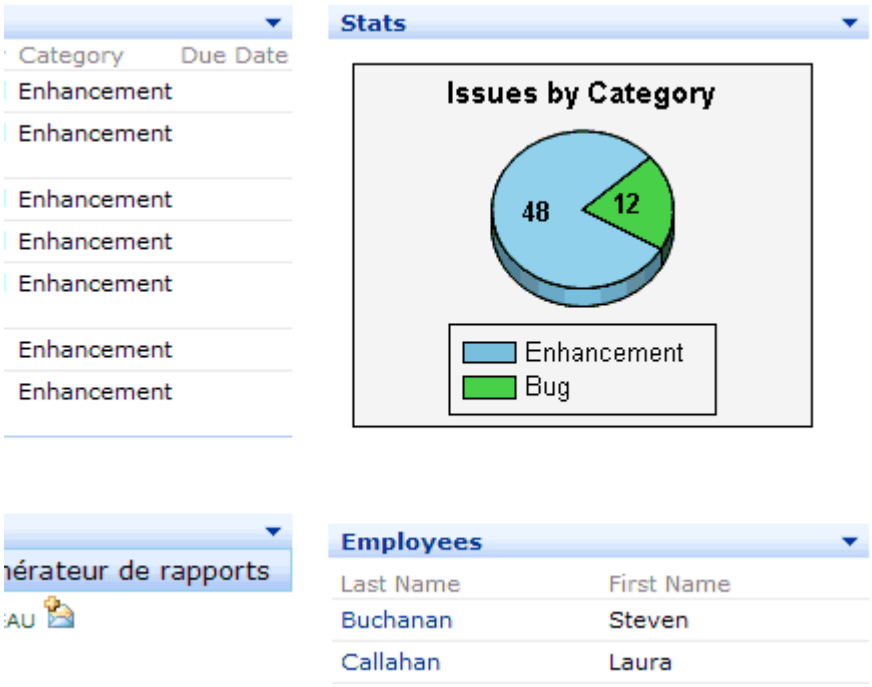
Introduction

Enesys Data Extension Reporting Services is a data extension for Microsoft SQL Reporting Services that makes it possible to create reports using data in SharePoint lists.

In addition to offering a printing tool that was missing in SharePoint, the use of Enesys RS Data Extension opens a particularly interesting field of application:

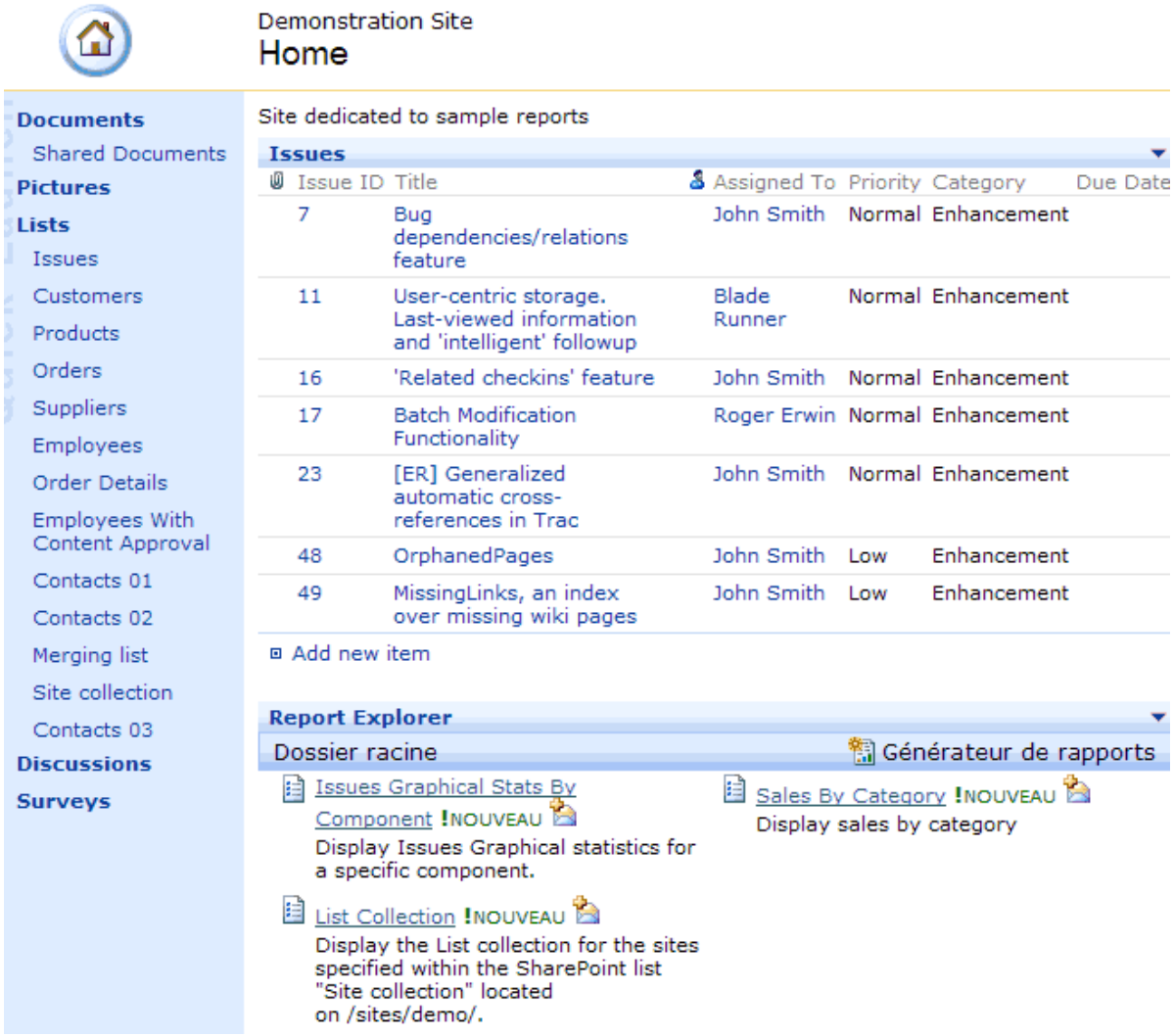
Display graphics on your SharePoint Sites

The Service Pack 2 of Reporting Services provides a report viewer web part that lets you display a report located on your report server. You can use this web part to display graphics directly into your SharePoint pages.



Make your reports available on your SharePoint Site

Using the Report Explorer web part provided with the Service Pack 2 of Reporting Services, you can easily make your reports available to users on their team sites:



Provide limited access to specific lists

It might be desirable for some specific lists to provide an access limited to some fields. For example, you might have a list of contacts whose "home phone numbers" should only be viewable by some authorized users. It's not currently possible to address this situation in SharePoint without duplicating the contact list.

Reports can be executed in the user security context of your choice. Thus you can restrict permissions on a SharePoint list and still provide specific views on that list using reports built with Enesys RS Data Extension.

Merge lists from various sites for mail-merge purpose

Using the <multiList> feature of **Enesys RS Data Extension**, you can merge an unlimited number of SharePoint lists. One of the possible uses of this feature is to merge contact lists from various sites and to export the merged data to excel for mail-merge purpose.

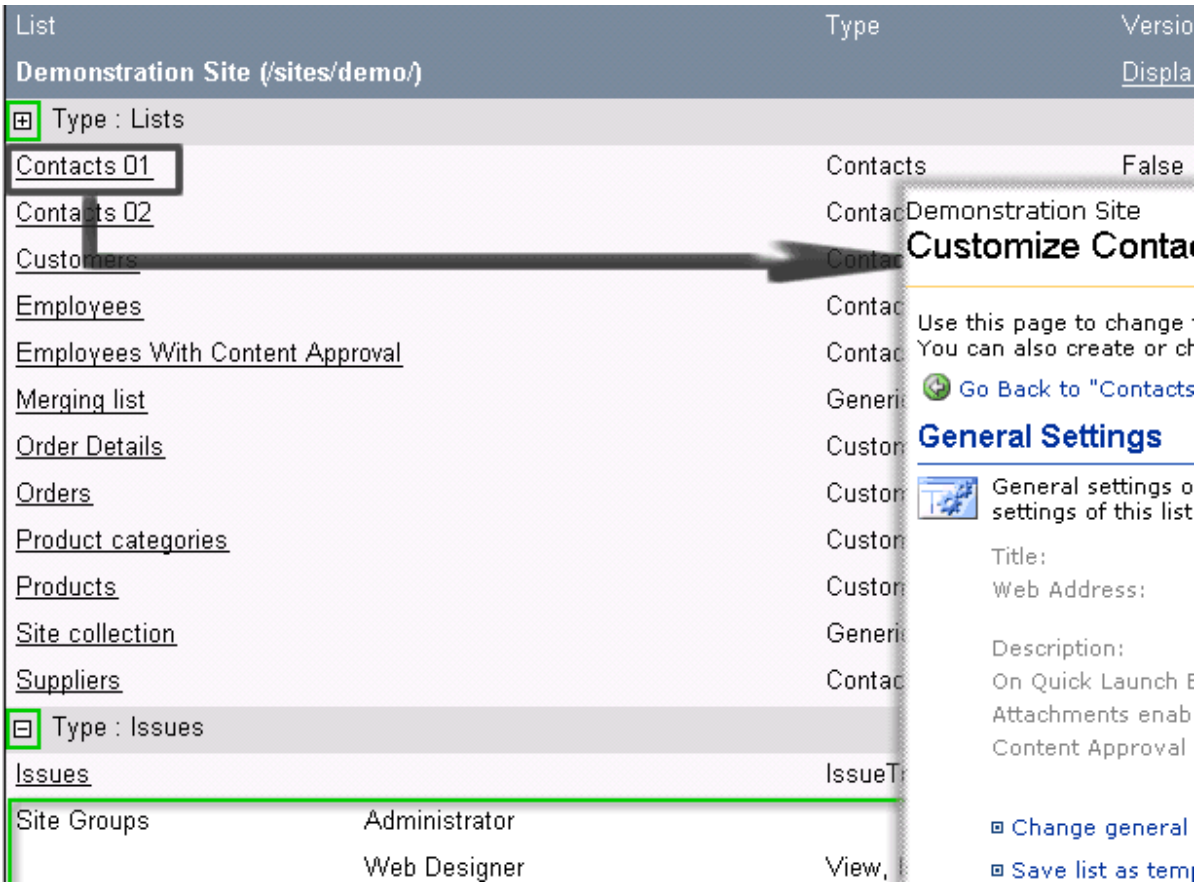
There's no need for it to be complicated for the users and just one click on a link can build the report and export it automatically to excel.

Get an overall vision of your list permissions

SharePoint is not really famous for being very user friendly regarding permissions management. Checking list permissions rapidly becomes a tiedous task.

Not only you can get an overall vision of your list and sites permissions with clean formatted reports, you can also use Reporting Services navigation features to build highly interactive reports that lets you jump from you report to SharePoint setting pages or to other more detailed reports.

So that you do not have to start from scratch we have included several samples that might be used "as is" with just minor modifications:



Deliver summary reports through email

By using Reporting Services subscription features, you can deliver reports based on SharePoint data through email on a regular schedule.

Roll up list items into one view

Using the "union" operator or the <multiList> feature, you can roll-up list items from various lists located in various sites into one view.

License agreement

By installing Enesys RS Data Extension (herein the "Software") developed by Enesys, you are accepting the following License Agreement.

IMPORTANT: this license is a legal agreement between you (either an individual or a single entity) and ENESYS. By installing and using the software you are agreeing to be bound by the terms of this license agreement. Read it carefully before installing and using the software. if you do not agree to the terms of this license agreement, then do not install the software.

License grants

This license grants you the right to install the SOFTWARE on one (and only one) server where Microsoft SQL Reporting Services is installed.

The SOFTWARE may be installed on an unlimited number of workstation for the purpose of building reports.

License limitations

You may not disassemble, decompile, reverse engineer, or attempt in any manner to reconstruct or discover any source code of the SOFTWARE.

You may not rent or provide hosting services using the SOFTWARE.

Limited warranty

Enesys warrants that the SOFTWARE will perform substantially in accordance with the accompanying documentation for a period of thirty days from the date of receipt.

Neither Enesys nor its suppliers shall be liable to you or any third party for any indirect, special, incidental, punitive, cover or consequential damages (including, but not limited to, damages for the inability to use equipment or access data, loss of business, loss of profits, business interruption or the like), arising out of the use of, or inability to use, the software and based on any theory of liability including breach of contract, breach of warranty, product liability or otherwise.

Enesys's total liability to you for actual damages for any cause whatsoever will be limited to the amount paid by you for the SOFTWARE that caused such damage.

Support

To obtain technical support, you may use the forums available on our web site http://www.enesyssoftware.com.

Installation

Enesys RS Data Extension is provided as a zip file which contains the following files:

File	Description
Enesys.ReportingServices.SpExtension.dll	Reporting Services Data Processing Extension for SQL Server 2000 Reporting Services.
Enesys.ReportingServices.SpExtension2005.dll	Reporting Services Data Processing Extension for SQL Server 2005 Reporting Services.
EnesysRSDataExtensionManual.pdf	English manual.
ConfigFiles.zip	Zip file containing the configuration files that must be updated when installing Enesys RS Data Extension on your server and/or workstation as explained in this manual. Those files are not intended to replace your own configuration files. Their purpose is to highlight where the configuration elements should be added.
ReportSamples.zip	Zip file containing report samples for both SQL Server 2000 and 2005. The zip file also includes a backed up SharePoint site containing the lists necessary to run the sample reports. Please refer to the chapter dedicated to the samples for more information.

The extension must be installed and configured on the server on which Microsoft SQL Report Server is installed, as well as on the development machine on which the reports will be built using the Report Designer.

The installation instructions assume a default installation. Adjust the installation paths depending on your installation. Rather than copying the entries from the documentation, we encourage you to copy them from the sample configuration files.

SQL Server 2005

Report Server

Copy **Enesys.ReportingServices.SpExtension2005.dll** in the directory *<drive>\Program Files\Microsoft SQL Server\MSSQL.3\Reporting Services\ReportServer\bin*.

Modify the configuration file *<drive>\Program Files\Microsoft SQL Server\MSSQL.3\Reporting Services\ReportServer\bin\RSReportServer.config*.

Locate the **Data** element in the config file:

```
<Data>
   <Extension Name="SQL"
   Type="Microsoft.ReportingServices.DataExtensions.SqlConnectionWrapper,Microsoft.
   ReportingServices.DataExtensions"/>
   …
</Data>
```

And add the following entry:

```
   <Extension Name="ENESYSSPS"
   Type="Enesys.ReportingServices.SpExtension.Connection,Enesys.ReportingServices.S
   pExtension2005"/>
```

In order to grant the rights needed to execute the extension, modify the configuration file *<drive>\Program Files\Microsoft SQL Server\MSSQL.3\Reporting Services\ReportServer\rssrvpolicy.config* in order to add the following entry:

```
<CodeGroup
      class="UnionCodeGroup"
      version="1"
      PermissionSetName="FullTrust">
  <IMembershipCondition
        class="UrlMembershipCondition"
        version="1"

   Url="c:\program files\Microsoft SQL Server\MSSQL.3\Reporting Services\ReportServ
er\bin\Enesys.ReportingServices.SpExtension2005.dll"/>
</CodeGroup>
```

The CodeGroup entry should be placed directly below the existing entry for the URL membership "$CodeGen$/*".

Development machine - Report designer

Copy **Enesys.ReportingServices.SpExtension2005.dll** in the directory *<drive>\Program Files\ Microsoft Visual Studio 8\Common7\IDE\PrivateAssemblies*.

Modify the configuration file *<drive>\Program Files\ Microsoft Visual Studio 8\Common7\IDE\PrivateAssemblies\RSReportServer.config*.

Locate the **Data** element in the config file:

```
<Data>
   <Extension Name="SQL"
   Type="Microsoft.ReportingServices.DataExtensions.SqlConnectionWrapper,Microsoft.
   ReportingServices.DataExtensions"/>
   …
</Data>
```

And add the following entry:

```
   <Extension Name="ENESYSSPS"
   Type="Enesys.ReportingServices.SpExtension.Connection,Enesys.ReportingServices.S
   pExtension2005"/>
```

Locate the **Designer** element in the config file:

```
<Designer>
   <Extension Name="SQL"
   Type="Microsoft.ReportingServices.QueryDesigners.VDTQueryDesigner,Microsoft.Repo
   rtingServices.QueryDesigners"/>
   <Extension Name="OLEDB"
   Type="Microsoft.ReportingServices.QueryDesigners.VDTQueryDesigner,Microsoft.Repo
   rtingServices.QueryDesigners"/>
   …
</Designer>
```

And add the following entry:

```
   <Extension Name="ENESYSSPS"
   Type="Microsoft.ReportingServices.QueryDesigners.GenericQueryDesigner,Microsoft.
   ReportingServices.QueryDesigners"/>
```

Modify the configuration file *<drive>\Program Files\ Microsoft Visual Studio 8\Common7\IDE\PrivateAssemblies\RSPreviewPolicy.config*. and add the following entry:

```
<CodeGroup
  class="UnionCodeGroup"
  version="1"
  PermissionSetName="FullTrust">
 <IMembershipCondition
       class="UrlMembershipCondition"
       version="1"
       Url="C:\Program Files\Microsoft Visual
Studio 8\Common7\IDE\PrivateAssemblies\Enesys.ReportingServices.SpExtensio
n2005.dll" />
</CodeGroup>
```

SQL server 2000

Report Server

Copy **Enesys.ReportingServices.SpExtension.dll** in the directory *<drive>\Program Files\Microsoft SQL Server\MSSQL\Reporting Services\ReportServer\bin*.

Modify the configuration file *<drive>\Program Files\Microsoft SQL Server\MSSQL\Reporting Services\ReportServer\RSReportServer.config*.

Locate the **Data** element:

```
<Data>
   <Extension Name="SQL"
   Type="Microsoft.ReportingServices.DataExtensions.SqlConnectionWrapper,Microsoft.
   ReportingServices.DataExtensions"/>
```

```
   <Extension Name="OLEDB"
   Type="Microsoft.ReportingServices.DataExtensions.OleDbConnectionWrapper,Microsof
   t.ReportingServices.DataExtensions"/>
   <Extension Name="ORACLE"
   Type="Microsoft.ReportingServices.DataExtensions.OracleClientConnectionWrapper,M
   icrosoft.ReportingServices.DataExtensions"/>
   <Extension Name="ODBC"
   Type="Microsoft.ReportingServices.DataExtensions.OdbcConnectionWrapper,Microsoft
   .ReportingServices.DataExtensions"/>
</Data>
```

And add the following entry:

```
   <Extension Name="ENESYSSPS"
   Type="Enesys.ReportingServices.SpExtension.Connection,Enesys.ReportingServices.S
   pExtension"/>
```

In order to grant the rights needed to execute the extension, modify the configuration file *<drive>\Program Files\Microsoft SQL Server\MSSQL\Reporting Services\ReportServer\rssrvpolicy.config* in order to add the following entry:

```
<CodeGroup
      class="UnionCodeGroup"
      version="1"
      PermissionSetName="FullTrust">
  <IMembershipCondition
        class="UrlMembershipCondition"
        version="1"
        Url="c:\Program Files\Microsoft SQL Server\MSSQL\Reporting
Services\ReportServer\bin/Enesys.ReportingServices.SpExtension.dll"/>
</CodeGroup>
```

The CodeGroup entry should be placed directly below the existing entry for the URL membership "$CodeGen$/*".

Development Machine – Report designer

Copy **Enesys.ReportingServices.SpExtension.dll** in the directory <drive>\Program Files\Microsoft SQL Server\80\Tools\Report Designer.

Modify the configuration file <drive>\Program Files\Microsoft SQL Server\80\Tools\Report Designer\RSReportDesigner.config.

Locate the **Data** element:

```
<Data>
   <Extension Name="SQL"
   Type="Microsoft.ReportingServices.DataExtensions.SqlConnectionWrapper,Microsoft.
   ReportingServices.DataExtensions"/>
   <Extension Name="OLEDB"
   Type="Microsoft.ReportingServices.DataExtensions.OleDbConnectionWrapper,Microsof
   t.ReportingServices.DataExtensions"/>
   <Extension Name="ORACLE"
   Type="Microsoft.ReportingServices.DataExtensions.OracleClientConnectionWrapper,M
   icrosoft.ReportingServices.DataExtensions"/>
   <Extension Name="ODBC"
   Type="Microsoft.ReportingServices.DataExtensions.OdbcConnectionWrapper,Microsoft
   .ReportingServices.DataExtensions"/>
</Data>
```

And add the following entry:

```
<Extension Name="ENESYSSPS"
Type="Enesys.ReportingServices.SpExtension.Connection,Enesys.ReportingServices.SpE
xtension"/>
```

Locate the **Designer** element in the config file:

```
<Designer>
   <Extension Name="SQL"
   Type="Microsoft.ReportDesigner.Design.VDTQueryDesigner,Microsoft.ReportingServic
   es.Designer"/>
   <Extension Name="OLEDB"
   Type="Microsoft.ReportDesigner.Design.VDTQueryDesigner,Microsoft.ReportingServic
   es.Designer"/>
   <Extension Name="ORACLE"
   Type="Microsoft.ReportDesigner.Design.VDTQueryDesigner,Microsoft.ReportingServic
   es.Designer"/>
   <Extension Name="ODBC"
   Type="Microsoft.ReportDesigner.Design.VDTQueryDesigner,Microsoft.ReportingServic
   es.Designer"/>
</Designer>
```

And add the following entry:

```
<Extension Name="ENESYSSPS"
Type="Microsoft.ReportDesigner.Design.GenericQueryDesigner,Microsoft.ReportingServ
ices.Designer"/>
```

Sample configuration files

The installation package includes a zip file named ConfigFiles.zip. This file contains sample configuration files for both SQL Server 2000 and SQL Server 2005.

Be aware that those files are not intended to replace your own configuration files. Their purpose is to highlight where the configuration elements should be added.

To make it easier, the elements that must be added into your own configuration files have been surrounded by XML comments as in the following examples:

```
<!-- Enesys RS Data Extension - BEGIN -->
<Extension Name="ENESYSSPS"
 Type="Enesys.ReportingServices.SpExtensi
 Enesys.ReportingServices.SpExtension2005
<!-- Enesys RS Data Extension - END -->
```

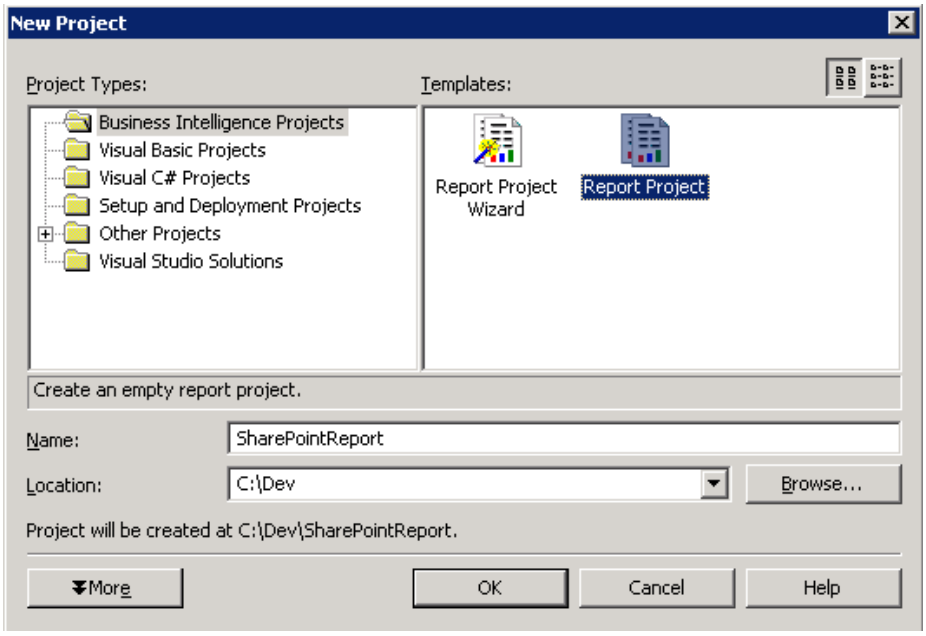
We recommend that you copy the various entries from those sample configuration files rather than from the current documentation.

Using Enesys RS Data Extension

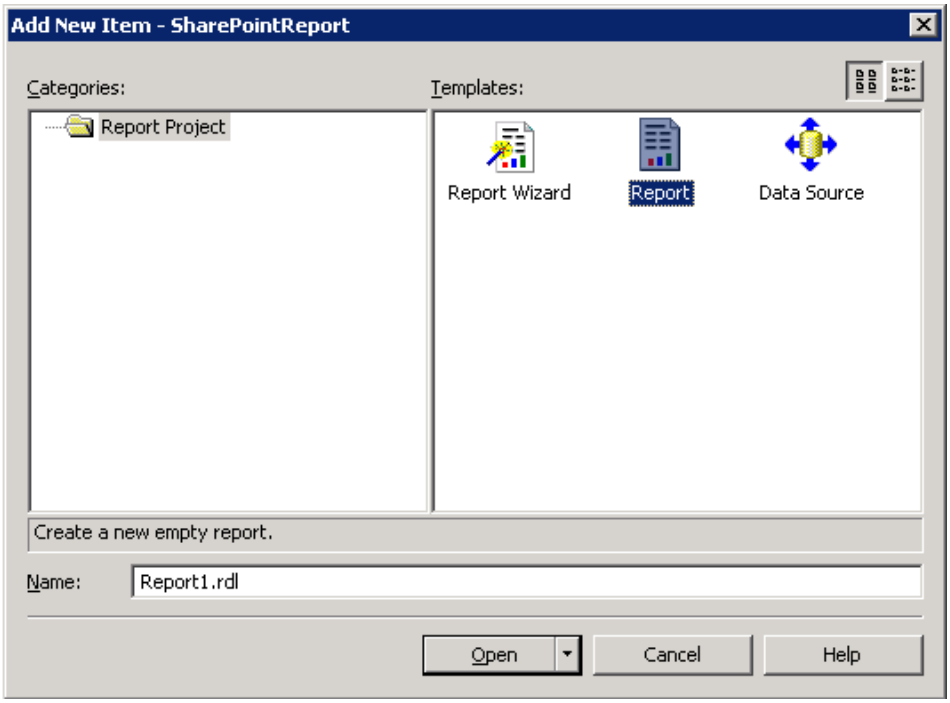
Quick start

This chapter shows how to quickly create a report using data from a SharePoint list.

Creating a new report project



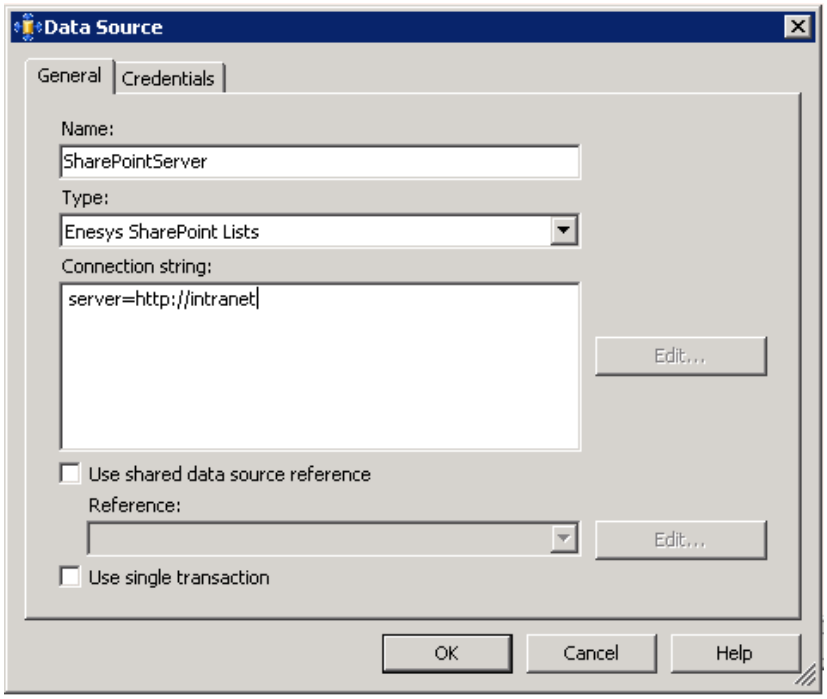
Adding a new report



Creating a dataset

Develop the **Dataset** scroll-down list and select **New Dataset**.

When creating the first dataset, you must define a data source:



Enter the information, taking into account the following instructions:

Field	Description
Name	Indicate a name for each data source; for example, the name of the SharePoint server whose data will be used.
Type	Select "Enesys SharePoint Lists" from the scroll-down list.
Connection string	Enter server=<url SharePoint>. Replace <url SharePoint> with the URL of your SharePoint server (e.g. http://intranet).

Select the **Credentials** and use the desired authentication method.

Click OK to create the dataset. Now you must create the query that will make it possible to define the SharePoint data associated with the dataset.

Creating a query

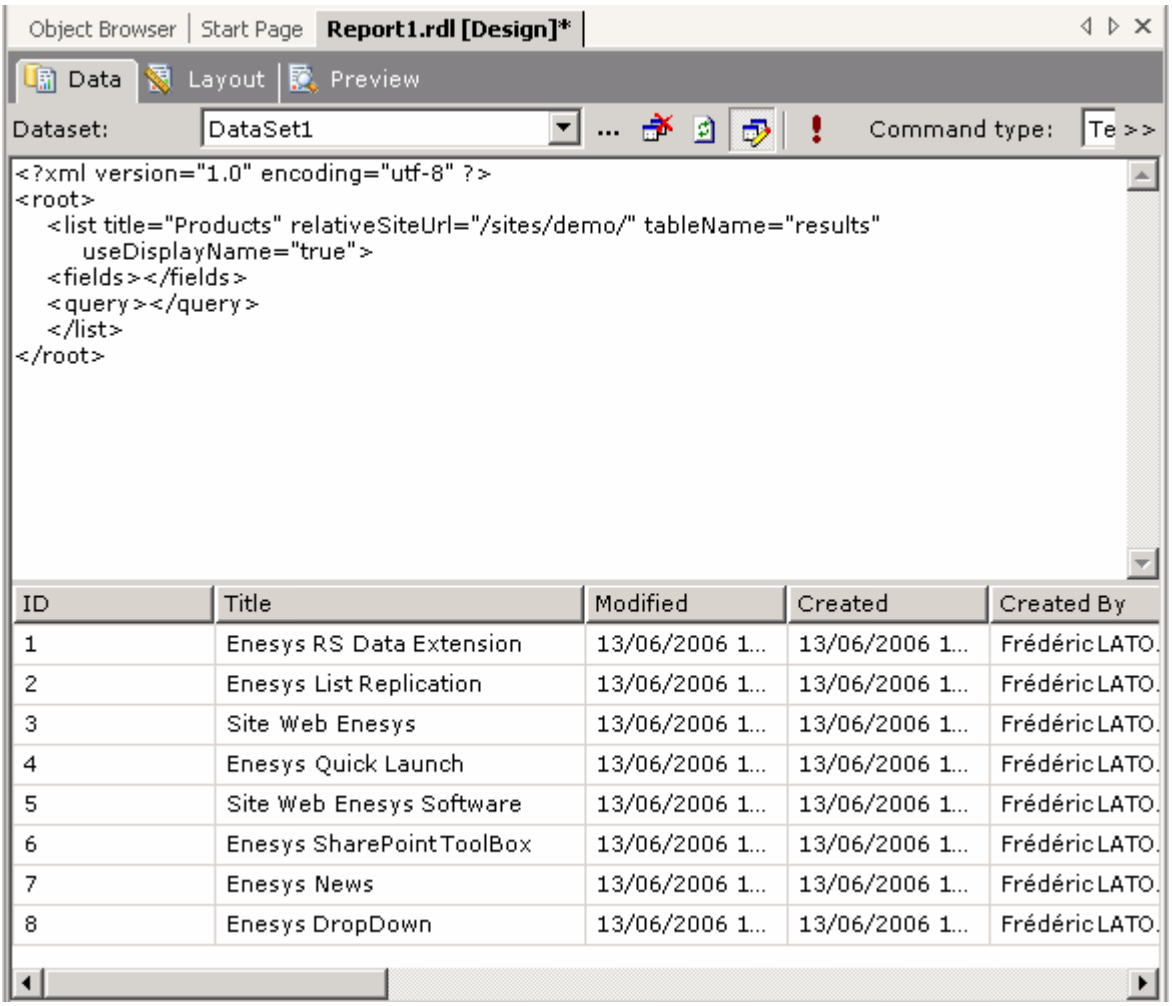
Click on the Generic Query Designer button  .

Enter (or copy) the following text:

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
   <list title="Products" relativeSiteUrl="/sites/demo/" tableName="results" useDisplayName="true">
   <fields></fields>
   <query></query>
   </list>
</root>
```

Replace "Products" and "/sites/demo/" with the names of a list and a SharePoint site located on your server.

Click on the run button to execute the query. You obtain all the elements in the SharePoint list:



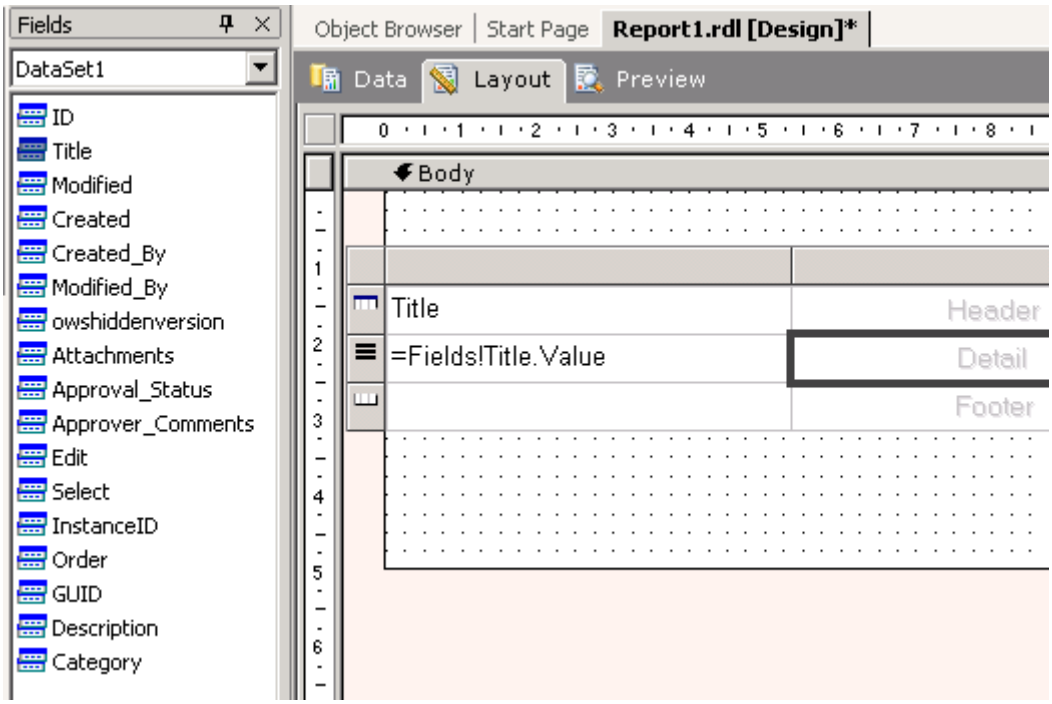
Once the query has been tested, the logical approach is, of course, to design a layout for the data.

Layout

Note that if you have just created the dataset, you may have to click the "refresh fields" button to display the list of fields that can be used in the formatting.

Drag the **Table** button from the toolbox.

Drag the desired fields from the SharePoint list into the table:



Change the style of the various cells to make the report more visually appealing and select Preview to see the result.

Conclusion

For the most part, the information presented in this quick start guide is not specific to Enesys RS Data Extension. The approach is similar regardless of the type of data used.

What you should remember about the use of data from SharePoint lists:

- The **data source** used must be of the type Enesys SharePoint Lists.
- The **connection string** for the **data source** must be in the form "*server=<url>*" where *<url>* corresponds to the URL of the SharePoint server whose data is being used.
- The **dataset** that will contain the SharePoint data is built from a specific **query string** in XML format that is explained in depth in this documentation.

Obtaining data from a SharePoint list

You can retrieve data from any SharePoint list using Enesys RS Data Extension specific query syntax which is based on XML.

In its simplest form, a query string appears as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
   <list title="Products" relativeSiteUrl="/sites/demo/" tableName="Products">
      <fields></fields>
      <query></query>
   </list>
</root>
```

The <list /> element makes it possible to obtain data from a SharePoint list.
The attributes and the child elements make it possible to specify the desired list, as well as any selection criteria.

The previous query retrieves all the items from the SharePoint list "Products" located on the site "/sites/demo".

All the columns are returned because nothing was specified in the element <fields></fields>.

All the lines are returned because no criteria was specified in the element <query></query>.

Specifying SharePoint list

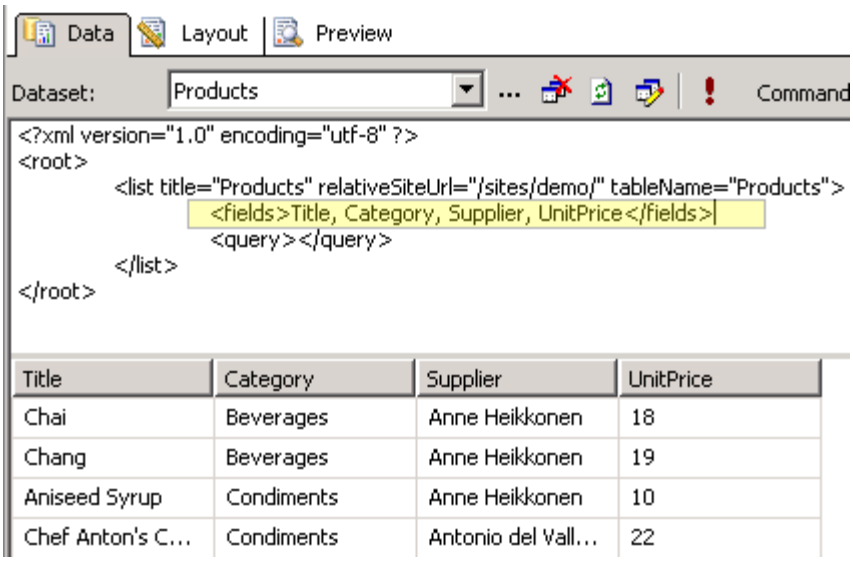
A SharePoint list title can easily be modified by a user. To avoid this situation you may specify the SharePoint list to retrieve items from, by specifying its ID (also called name) using the listID attribute rather that specifying its title using title attribute.

The SharePoint list ID will not change over time. It's still can be deleted.

Note that if you are using both attributes, listID will take the precedence over title. Even if you rely on list id to specify a SharePoint list, we encourage you to set title attribute as a meaningful reminder of list content.

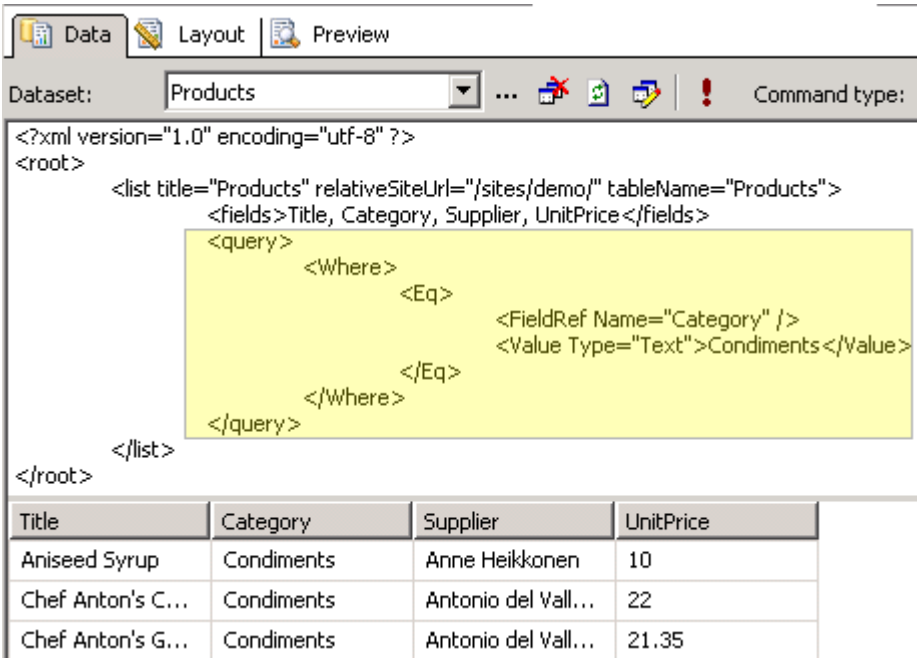
Specifying list columns

f you don't need all the columns of the SharePoint list, you can specify the columns you would like to retrieve using the <fields> element:



Filtering list data

A filter can be applied to a list using Collaboration Application Markup Language query format. It doesn't take a long time to understand the basic principles. The CAML Query must be placed within the <query> element. The following query example will retrieve items where the **Category** column equals to "Condiments":

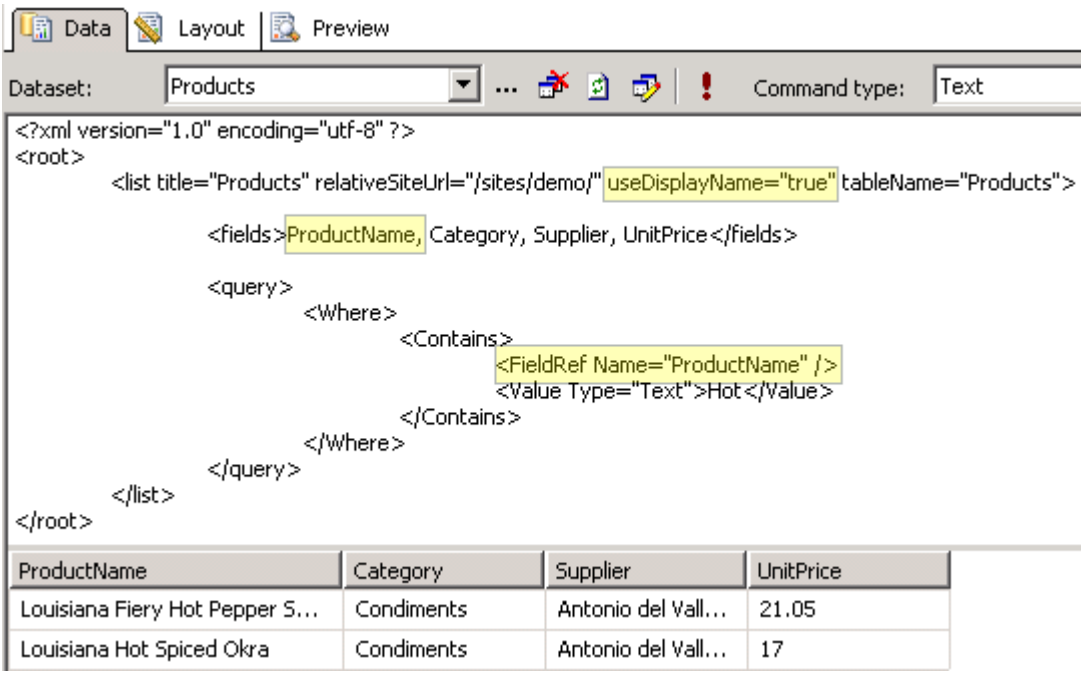


You can also apply a filter at the report level using reporting services features. However, in that case, the whole list data is retrieved before the filter is applied. CAML filter is applied at SharePoint server side and it will improve the performance considerably if you just need a subset of the list. You can also use a mix of those filtering options when necessary.

Using column display names or internal names

SharePoint list columns have an internal name and a display name. When a column is initially created, the display name and the internal name of the column are the same (except if the name contains space or accent marks). When you modify the name of a column, it will only modify the display name. The internal name is never modified and you may end up with columns whose internal names no longer have any connections to the display names.

Though using internal names has the advantage of not breaking your report when a column name is changed, you can choose to use display names when specifying columns to retrieve <fields> element) and filtering the list <query> element) by setting the useDisplayName attribute to true.



Using report parameters

Reporting Services lets you define parameters at the report level so that the user may be proposed several options for running the report.

Parameters may be used within the <query></query> element. A parameter is composed of a name surrounded by the characters @ and ! **(e.g. @product!)**.

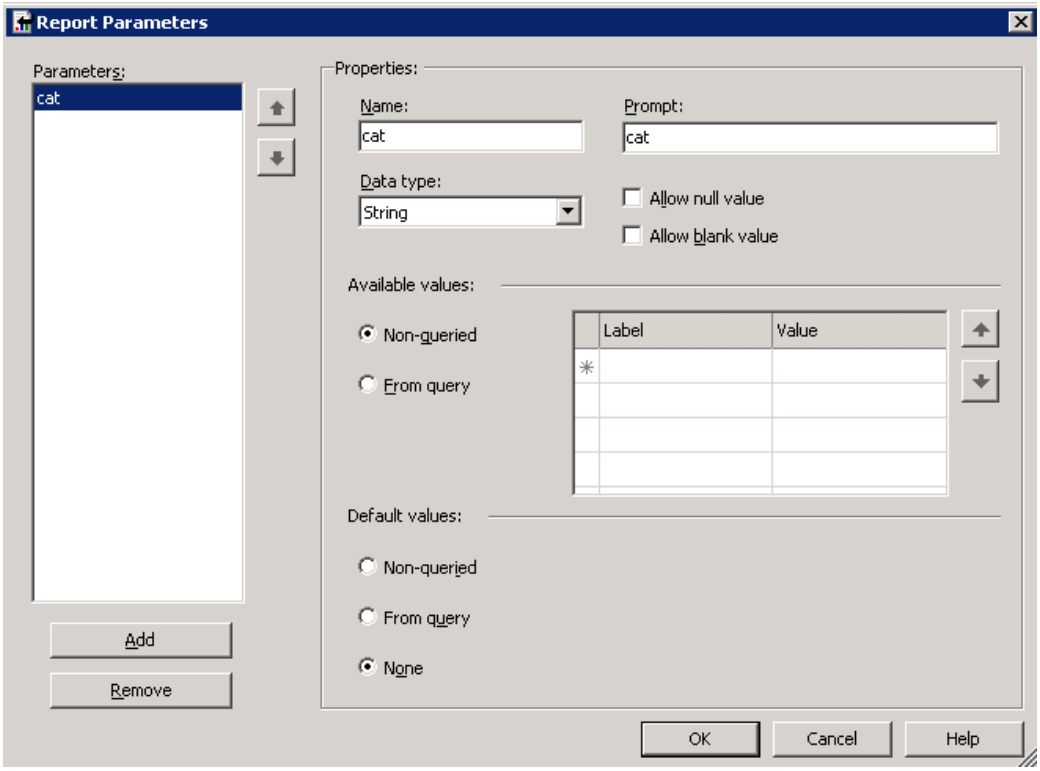
When you use a parameter in the query string and it is not defined at the report level, it will be automatically created when you select the Layout tab.

Also note that the query execution in the data window will ask you to enter a value for the parameter.

The following example shows how to return the data from the SharePoint "Products" list whose category is equal to the value of the "cat" parameter. The parameter's value will be entered when the report is executed and inserted into the query string.

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
   <list title="Products" relativeSiteUrl="/sites/demo/" tableName="Products">
      <fields>Title, Category, Supplier, UnitPrice</fields>
      <query>
         <Where>
            <Eq>
               <FieldRef Name="Category" />
               <Value Type="Text">@cat!</Value>
            </Eq>
         </Where>
      </query>
   </list>
</root>
```

As previously indicated, entering the parameter in the query string will automatically create the corresponding report parameter, as shown in the following image:



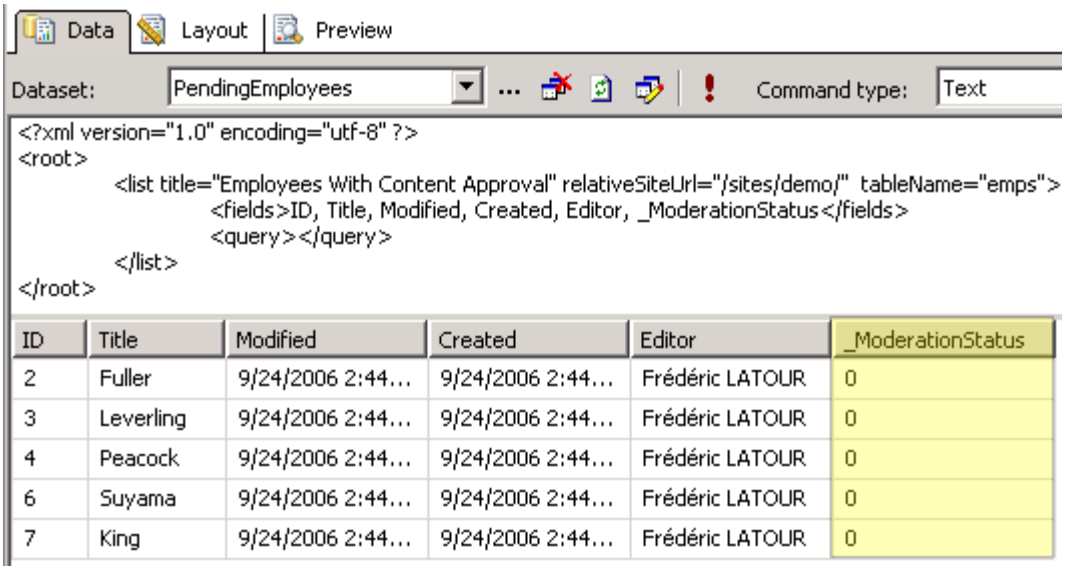
Rather than letting the category be entered in full text, you can create a new dataset that will be used to define the available values, from which the report user shall make a selection.

If you have a distinct SharePoint list containing the list of categories, you can simply create a dataset returning the list data.
To create a dataset from the list of "products" while eliminating repeats (a category may be associated with several products and thus appear several times in the list), you must apply a "distinct" operation to the list as it will be explained later.

Note that you can use any other data source (e.g. SQL) for the available values.

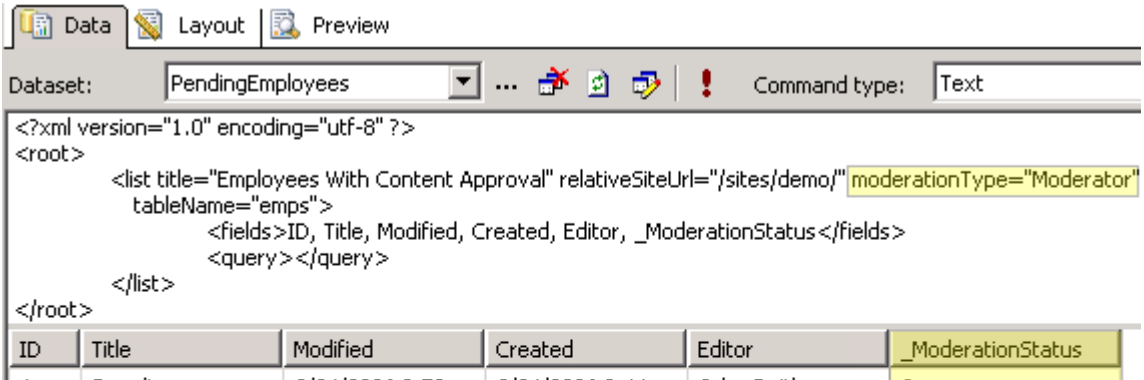
Pending items

For lists for which content approval is required, only approved items are retrieved by default as shown by the **_ModerationStatus** column equals to 0:



ID	Title	Modified	Created	Editor	_ModerationStatus
2	Fuller	9/24/2006 2:44...	9/24/2006 2:44...	Frédéric LATOUR	0
3	Leverling	9/24/2006 2:44...	9/24/2006 2:44...	Frédéric LATOUR	0
4	Peacock	9/24/2006 2:44...	9/24/2006 2:44...	Frédéric LATOUR	0
6	Suyama	9/24/2006 2:44...	9/24/2006 2:44...	Frédéric LATOUR	0
7	King	9/24/2006 2:44...	9/24/2006 2:44...	Frédéric LATOUR	0

You can use the moderationType attribute to retrieve pending items for a specific list:



ID	Title	Modified	Created	Editor	_ModerationStatus
1	Davolio	9/24/2006 2:59...	9/24/2006 2:44...	John Smith	2
2	Fuller	9/24/2006 2:44...	9/24/2006 2:44...	Frédéric LATOUR	0
3	Leverling	9/24/2006 2:44...	9/24/2006 2:44...	Frédéric LATOUR	0
4	Peacock	9/24/2006 2:44...	9/24/2006 2:44...	Frédéric LATOUR	0
5	Buchanan	9/24/2006 2:58...	9/24/2006 2:44...	John Smith	2
6	Suyama	9/24/2006 2:44...	9/24/2006 2:44...	Frédéric LATOUR	0
7	King	9/24/2006 2:44...	9/24/2006 2:44...	Frédéric LATOUR	0
8	Callahan	9/24/2006 3:10...	9/24/2006 2:44...	Roger Erwin	2
9	Dodsworth	9/24/2006 2:59...	9/24/2006 2:44...	John Smith	2

Expanding recurring events

The expandRecurrent attribute let's you expand recurring events for an event list.

You will appreciate this feature as even SharePoint object model does not include this possibility. Besides building reports displaying recurring events, a possible scenario is to export recurring events as an xml file using Reporting Services subscription features in order to feed another data source or a business process.

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
   <list title="Events" relativeSiteUrl="/sites/demo/" expandRecurrent="true"
         expandFirstDate="@firstDate!" expandLastDate="@lastDate!"
         tableName="Events">
      <fields></fields>
      <query>
      </query>
   </list>
</root>
```

The expandRecurrent attribute goes along with expandFirstDate and expandLastDate optional attributes that lets you define the range of dates for which recurring events will be expanded. Parameters may be used to set expandFirstDate and expandLastDate values.

It is important to note that the CAML Query will not filter the expanded events. Recurring events are expanded after they have been retrieved from the SharePoint list.

Let's see an example to clarify this point:

```
<root>
   <list title="Events" relativeSiteUrl="/sites/demo/" expandRecurrent="true"
         tableName="Events">
      <fields></fields>
      <query>
         <Where>
            <Geq>
               <FieldRef Name="EventDate" />
               <Value Type="DateTime">@StartDate!</Value>
            </Geq>
         </Where>
      </query>
   </list>
</root>
```

One may think that the previous query would retrieve all events (including recurring events) starting from "StartDate" parameter. This is not exactly the case. Recurring events starting before "StartDate" parameter won't be retrieved at all though after being expanded some events may indeed start after "startDate" parameter. Only recurring events starting from "StartDate" will be expanded.

To make the query as easy to write would have made it necessary to write our own caml interpreter in order to filter the expanded events with the relevant part of the caml query.

Nevertheless, there is a solution to obtain events between two dates including the recurring events by writing the following query:

```
<root>
   <list title="Events" relativeSiteUrl="/sites/demo/" expandRecurrent="true"
         expandFirstDate="@firstDate!" expandLastDate="@lastDate!"
         tableName="RecEvents">
      <fields></fields>
      <query>
         <Where>
            <And>
               <And>
                  <Leq>
                     <FieldRef Name="EventDate" />
                     <Value Type="DateTime">@lastDate!</Value>
                  </Leq>
                  <Geq>
                     <FieldRef Name="EndDate" />
                     <Value Type="DateTime">@firstDate!</Value>
                  </Geq>
               </And>
               <Eq>
                  <FieldRef Name="fRecurrence" />
                  <Value Type="Boolean">1</Value>
               </Eq>
            </And>
         </Where>
      </query>
   </list>
```

```
<list title="Events" relativeSiteUrl="/sites/demo/" expandRecurrent="false"
      tableName="Events">
   <fields></fields>
   <query>
      <Where>
         <And>
         <And>
            <Geq>
               <FieldRef Name="EventDate" />
               <Value Type="DateTime">@firstDate!</Value>
            </Geq>
            <Leq>
               <FieldRef Name="EventDate" />
               <Value Type="DateTime">@lastDate!</Value>
            </Leq>
         </And>
            <Eq>
               <FieldRef Name="fRecurrence" />
               <Value Type="Boolean">0</Value>
            </Eq>
         </And>
      </Where>
   </query>
</list>

<sqlOp op="union">
   <dstTableName>MergedEvents</dstTableName>
   <parentTableName>RecEvents</parentTableName>
   <childTableName>Events</childTableName>
   <labelColumn>true</labelColumn>
   <parentLabelValue>Recurrent</parentLabelValue>
   <childLabelValue>Non Recurrent</childLabelValue>
</sqlOp>

<resultSet>MergedEvents</resultSet>
</root>
```

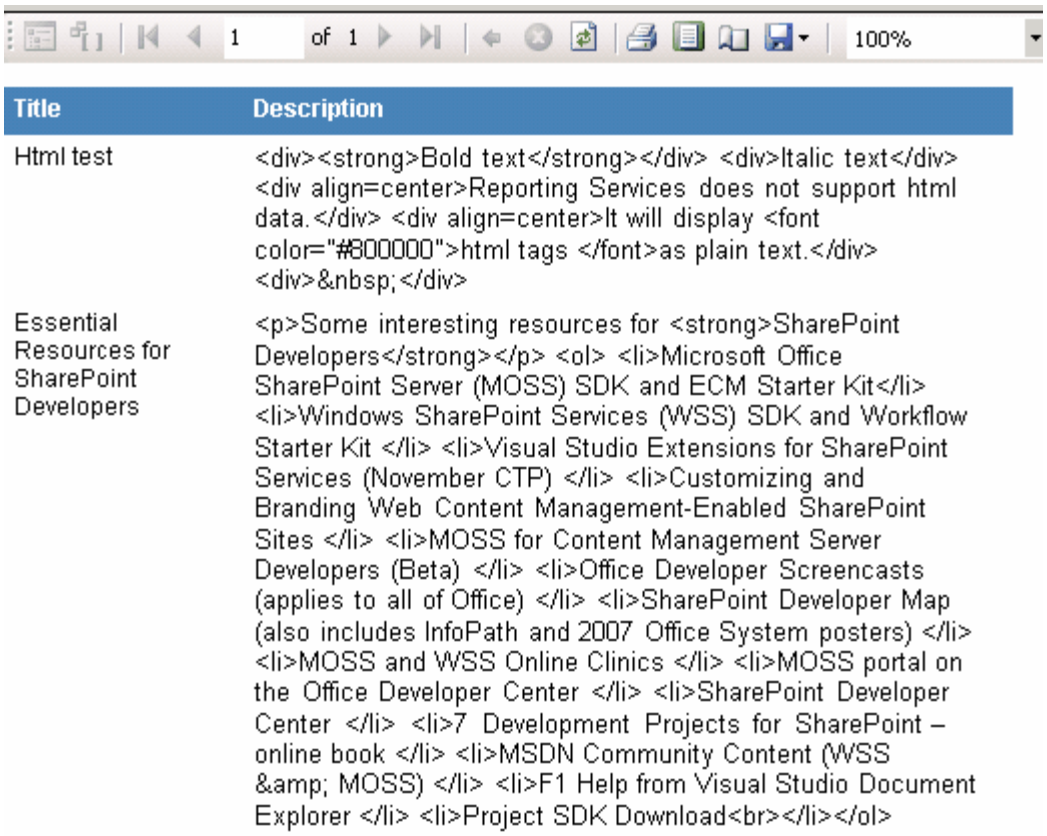
The first <list> query element will retrieve recurring items only (fRecurrence) and expand them using the date range defined by “firstDate” and “lastDate” parameters.

The second <list> query element will retrieve non recurring events where EventDate is between “firstDate” and “lastDate” parameters.

At last, <sqlOp op=”union”> operation will merge both result sets into one dataset suitable for Reporting Services.

Stripping Html Tags

It is a well known limitation that Reporting Services is not able to handle html tags within a specific field. Thus, html data within a field will be displayed as plain text as shown in the following image:



Title	Description
Html test	Bold text Italic text Reporting Services does not support html data. It will display html tags as plain text.
Essential Resources for SharePoint Developers	Some interesting resources for SharePoint Developers - Microsoft Office SharePoint Server (MOSS) SDK and ECM Starter Kit - Windows SharePoint Services (WSS) SDK and Workflow Starter Kit - Visual Studio Extensions for SharePoint Services (November CTP) - Customizing and Branding Web Content Management-Enabled SharePoint Sites - MOSS for Content Management Server Developers (Beta) - Office Developer Screencasts (applies to all of Office) - SharePoint Developer Map (also includes InfoPath and 2007 Office System posters) - MOSS and WSS Online Clinics - MOSS portal on the Office Developer Center - SharePoint Developer Center 7 Development Projects for SharePoint – online book - MSDN Community Content (WSS & MOSS) - F1 Help from Visual Studio Document Explorer - Project SDK Download

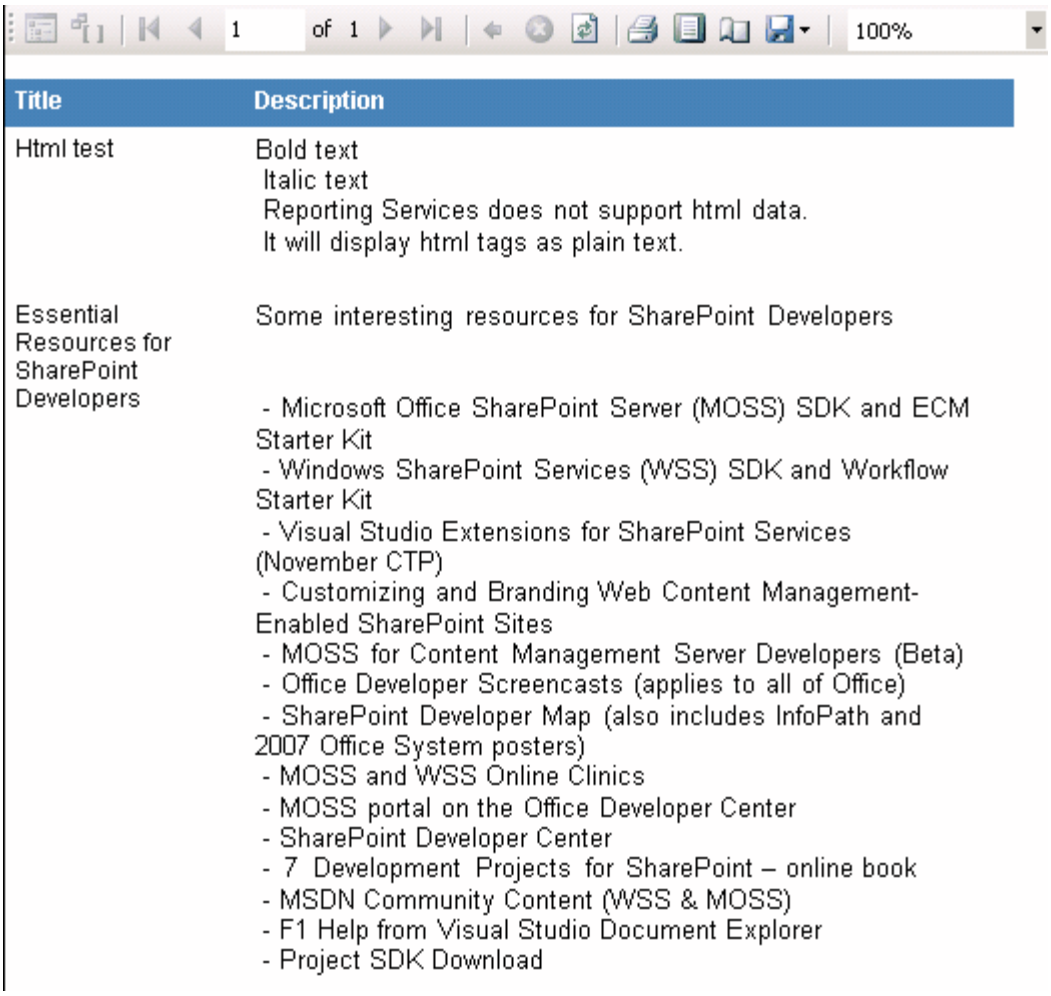
Though, you may use your own approach to remove html tags by using Reporting Services embedded code features, we have added the ability to strip html tags for a specific SharePoint list using the stripHtml attribute:

```
<root>
   <list title="Html Sample" relativeSiteUrl="/sites/demo/" stripHtml="true"
         tableName="html">
      <fields></fields>
      <query>
      </query>
   </list>

   <resultSet>html</resultSet>
</root>
```

The stripHtml attribute is optional and defaults to false if it's not defined.

Though not really appealing, stripping html makes text at least readable :



Title	Description
Html test	Bold text Italic text Reporting Services does not support html data. It will display html tags as plain text.
Essential Resources for SharePoint Developers	Some interesting resources for SharePoint Developers - Microsoft Office SharePoint Server (MOSS) SDK and ECM Starter Kit - Windows SharePoint Services (WSS) SDK and Workflow Starter Kit - Visual Studio Extensions for SharePoint Services (November CTP) - Customizing and Branding Web Content Management-Enabled SharePoint Sites - MOSS for Content Management Server Developers (Beta) - Office Developer Screencasts (applies to all of Office) - SharePoint Developer Map (also includes InfoPath and 2007 Office System posters) - MOSS and WSS Online Clinics - MOSS portal on the Office Developer Center - SharePoint Developer Center - 7 Development Projects for SharePoint – online book - MSDN Community Content (WSS & MOSS) - F1 Help from Visual Studio Document Explorer - Project SDK Download

Using Running Values

Though Reporting Services makes it possible to calculate running values on the fly, it will not let you use aggregate functions on running values (max, min, etc).

To overcome this limitation and being able to meet specific scenarios, Enesys RS Data Extension lets you specify columns for handling running values. In that case, the running value being seen as a column on Reporting Services side, it is possible to apply aggregation function on it.

Columns containing running values are added using a <customFields> child element of the <list> element as shown in the following example:

```
<root>
   <list title="Order Details" relativeSiteUrl="/sites/demo/" tableName="Order Details">
      <fields>Product, Quantity</fields>
      <query>
         <OrderBy>
            <FieldRef Name="Product" />
         </OrderBy>
      </query>
      <customFields>
         <field name="RunningTotal" dataType="System.Int32" op="RunningSum"
                groupColumnName="Product" param="Quantity"></field>
      </customFields>
   </list>

   <resultSet>Order Details</resultSet>
</root>
```

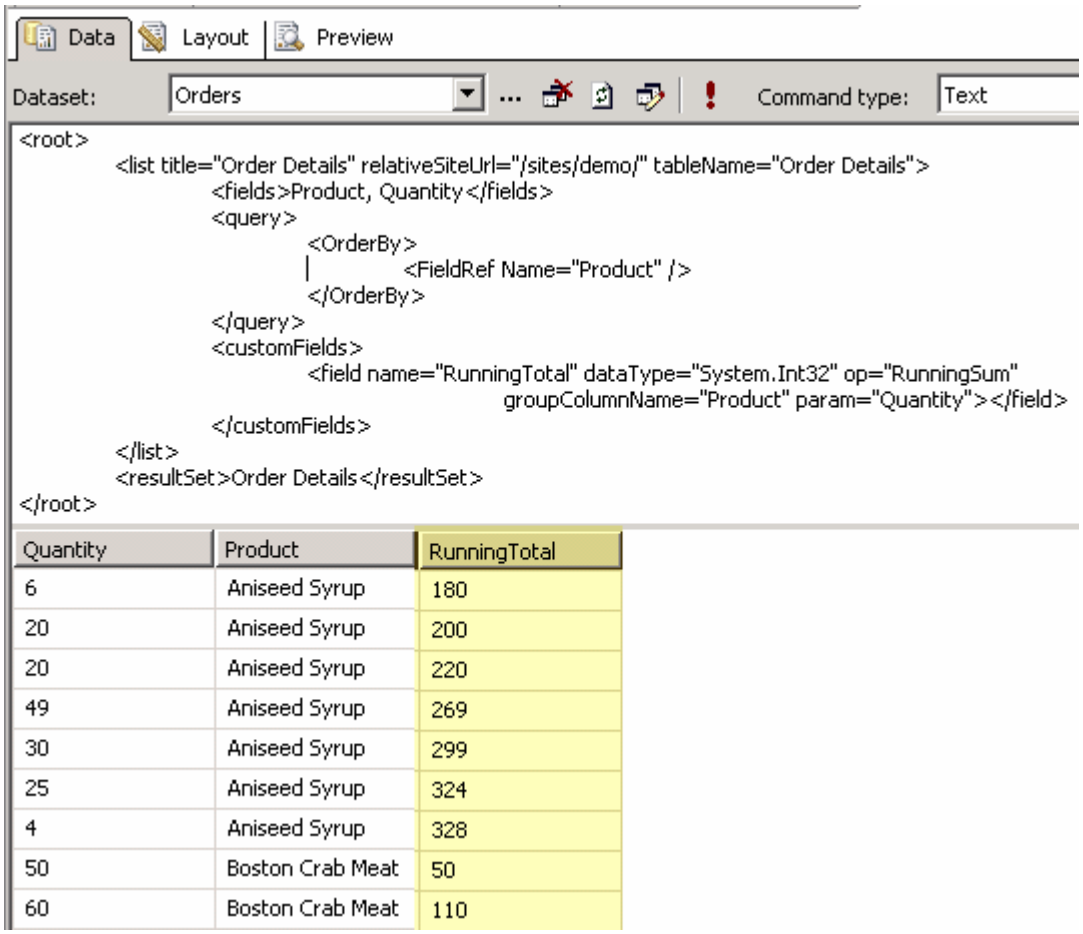
Each <field> child element of <customFields> element represents a specific column holding a running value. The attributes let's you specify the name of the new column that will be holding the running value, the data type of the column, the type of running value, the grouping column and the column holding the value.

To make it simple, the previous query will retrieve the Product and Quantity columns from the "Order Details" SharePoint list, ordered by "Product". Enesys RS Data Extension will add a custom column named "RunningTotal" (name) of type Integer (dataType) and will calculate a running sum (op) based on the "Quantity" column (param). Each time the "Product" column value changes (groupColumnName), the running sum is reset to 0.

The following image is what you will obtain when running the previous query within the Data view.

Note that the Running Sum is reset when the product value changes from "Aniseed Syrup" to "Boston Crab Meat".

It is important to note also that Enesys RS Data Extension will not automatically order the data on the groupColumnName column ("Product" in that case). It's up to you to order accordingly the SharePoint list using an <OrderBy> element within the CAML query. The groupColumnName attribute will only direct ERSDE to reset the running value when the groupColumnName's column value changes.



<list> element reference

Element/Attribute	Description
title	Title of the SharePoint list from which you want to retrieve data.
listID	GUID of the SharePoint list from which you want to retrieve data. Though not specifically self describing, the list GUID has the advantage of not changing overt time even if the list title is modified. Note that if you specify both title and listID attributes, the later will take precedence.
relativeSiteUrl	URL of the SharePoint site containing the list. This URL is relative to the SharePoint site as it is defined in the data source. This approach was chosen to make it possible to easily move from a SharePoint test server to a SharePoint production server by simply modifying the data source.
tableName	This attribute makes it possible to assign a name to the list data. It is not used when you recover data from a single list.

Element/Attribute	Description
	When working with several lists, this name will make it possible to distinguish the lists to which you wish to apply the processes.
moderationType	This optional attribute makes it possible to retrieve pending items for lists for which content approval is required. By setting the value of the attribute to “Moderator”, you will retrieve pending and rejected items as well as approved items.
useDisplayName	This attribute lets you indicate that you want to use the display names of the columns rather than their internal names as part of the CAML format query and the <fields></fields> element. In effect, the columns in SharePoint lists have an internal name and a display name. When a column is initially created, the display name and the internal name are identical (except if there are spaces or accent marks). However, the internal name is never modified—even if you change the name of the column. Thus you may end up with a column whose internal name no longer has any connection to the name displayed. CAML queries use the columns' internal names. To avoid having to search for the internal name associated with the columns using a complementary tool (CAML Builder or other), you can use the column display names by setting the attribute to "true". Before performing the SharePoint Web Service query, the display names will be automatically replaced by the internal names. If you prefer using the internal names of the columns you can set the attribute to "false" or even not define the attribute.
expandRecurrent	Optional Boolean. If set to “true”, recurring events will be expanded based on their recurrence definition. This attribute will not have any effect when used with lists other than event’s ones. The default value is “false”.
expandFirstDate expandLastDate	Optional. Range of dates for which the recurring events will be expanded. The attributes have no effect if expandRecurrent is false or not defined. The date format must respect the following format : **AAAA-MM-JJTHH:MM:SS**. A report parameter may be used instead of a literal.
stripHtml	Optional Boolean. When set to “true”, specifies html tag should be stripped from SharePoint columns containing html. The default value is “false”.
<customFields> </customFields>	

Element/Attribute	Description
<fields></fields>	Limit the list data to the columns defined in the context of this element. Each column name must be separated by a "," (comma). Depending on the value of the attribute useDisplayName, the column's display name or internal name must be used. If you do not enter any columns, all of the list's columns will be returned.
<query></query>	CAML-type query used with SharePoint. Here we will not describe this type of query; please see the SharePoint documentation for information about CAML query syntax. Some additional functionalities have been added to better integrate its use with Reporting Services: If useDisplayName="true", the column display names will be used. Parameters provided by Reporting Services may be used within the query element. Each parameter takes the following form: **@name_parameter!** (a name surrounded by @ and !).

<customFields> element reference

The <customFields> element has no attributes. Each <field> child element represents a column holding a running value (future versions of ERSDE may add other types of columns).

The following table describes the attributes of the <field> element.

Element/Attribute	Description
Name	Name of the custom column to be added to the result set.
dataType	Data type of the column that will be holding the running value (System.Int32, System.Decimal,).
Op	Function used for calculating the running values. The following values may be used: • Sum • Min • Max
groupColumnName	Optional. Specify the name of the column for which the running value will be reset each time its value changes.
Param	Name of the column containing the value.

Applying operations to lists

This feature is not available with the Community Edition.

You are not limited to build reports based on one SharePoint list. Enesys RS Data Extension lets you apply specific operations between two SharePoint lists.

In fact, you can define as many <lists> element as necessary and apply as many operations as you like in order to obtain the desired Dataset from which you will build your report.

Each <list> element defined within a query returns a set of data items from a SharePoint list for which we will use the generic term "**result set**". The tableName attribute is used to give a unique name to the result set. The unique name of the result set will serve as the basis for specifying result sets involved in operations.

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
   <list title="Products" relativeSiteUrl="/sites/demo/" tableName="Products">
      <fields></fields>
      <query></query>
   </list>
</root>
```

You can apply an operation between two "result sets" by using an <sqlOp op="operation"> element. The data items resulting from an operation is considered a "result set" as well which name is given using the <dstTableName> element. Thus you can apply operations on data items resulting from other operations.

```
    <sqlOp op="join">
        <dstTableName>ProdAndCat</dstTableName>
        <parentTableName>Products</parentTableName>
        <childTableName>Product categories</childTableName>
        <parentFieldName>Category</parentFieldName>
        <childFieldName>Title</childFieldName>
    </sqlOp>

    <sqlOp op="join">
        <dstTableName>OrdersAndDetails</dstTableName>
        <parentTableName>Orders</parentTableName>
        <childTableName>Order Details</childTableName>
        <parentFieldName>OrderID</parentFieldName>
        <childFieldName>Order_x0020_ID</childFieldName>
    </sqlOp>

    <sqlOp op="join">
        <dstTableName>ProdAndOrdersAndDetails</dstTableName>
        <parentTableName>OrdersAndDetails</parentTableName>
        <childTableName>ProdAndCat</childTableName>
        <parentFieldName>Product</parentFieldName>
        <childFieldName>Title</childFieldName>
    </sqlOp>
    <resultSet>ProdAndOrdersAndDetails</resultSet>

</root>
```

Joining result sets

The **join** operation works like the SQL inner join statement. It lets you join matching items between two result sets specified by <parentTableName> and <childTableName> based on their joining columns specified respectively by <parentFieldName> and <childFieldName> elements. The <dstTableName> element lets you give a unique name to the data items resulting from the join operation. It is possible to use this data for further operations using this unique name.

The following image displays a **join** operation between the **Products** and **Product categories** SharePoint lists (bearing that a <list> element has been defined for each of them). Products that do not have a matching category won't belong to the result set.

```
<sqlOp op="join">
    <dstTableName>ProdAndCat</dstTableName>
    <parentTableName>Products</parentTableName>
    <childTableName>Product categories</childTableName>
    <parentFieldName>Category</parentFieldName>
    <childFieldName>Title</childFieldName>
</sqlOp>
```

The **outerjoin** operation works like the SQL left outer join statement. It lets you join all the items from the result set specified by <parentTableName> with matching items

from the result set specified by <childTableName> based on their joining columns specified respectively by <parentFieldName> and <childFieldName> elements. There's no difference between the **join** and **outerjoin** syntax except for the op attribute.

```
<sqlOp op="outerjoin">
   <dstTableName>ProdAndCat</dstTableName>
   <parentTableName>Products</parentTableName>
   <childTableName>Product categories</childTableName>
   <parentFieldName>Category</parentFieldName>
   <childFieldName>Title</childFieldName>
</sqlOp>
```

You are not limited to one joining column. <parentFieldName> and <childFieldName> may contain several joining columns separated by a semi colon.

Joining lists is useful (though not limited to that usage) when dealing with SharePoint lists linked with a lookup fields. For example you might want to display products grouped by category and display the category description as well.

Merging result sets

You can merge two result set by using a **union** operation. It lets you combine the items of two result sets specified by <parentTableName> and <childTableName>. Both result sets must have matching fields and data types. However, two SharePoint lists can be merge even if they are not exactly identical as long as you only use shared fields in your <list> query element that are used as part of the **union** operation.

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
   <list title="Contacts 01" relativeSiteUrl="/sites/demo/"
      tableName="Contacts01">
      <fields>Title, Company, JobTitle, WorkCountry</fields>
      <query></query>
   </list>

   <list title="Contacts 02" relativeSiteUrl="/sites/demo/"
      tableName="Contacts02">
      <fields>Title, Company, JobTitle, WorkCountry</fields>
      <query></query>
   </list>

   <sqlOp op="union">
      <dstTableName>MergedContacts</dstTableName>
      <parentTableName>Contacts01</parentTableName>
      <childTableName>Contacts02</childTableName>
      <labelColumn>true</labelColumn>
      <parentLabelValue>Contacts01</parentLabelValue>
      <childLabelValue>Contacts02</childLabelValue>
   </sqlOp>
   <resultSet>MergedContacts</resultSet>
</root>
```

If you need to distinguish the result set's origin of each item (this would be the case if you want to build a report that group items bases on their origin), you can set the <labelColumn> child element to true and set the <parentLabelValue> and <childLabelValue> with some specific value for each result set. In that case a specific column named **rstLabel** will be created in the final result set. The rstLabel column will be filled with <parentLabelValue> and <childLabelValue> element values for respectively the <parentTableName> and <childTableName> items.

Title	Company	JobTitle	WorkCountry	rstLabel
Art Braunschwe...	Split Rail Beer &...	Sales Manager	USA	Contacts01
Anabela Domin...	Tradição Hiper...	Sales Represen...	Brazil	Contacts01
Maria Anders	Alfreds Futterki...	Sales Represen...	Germany	Contacts02
Thomas Hardy	Around the Horn	Sales Represen...	UK	Contacts02
Hanna Moos	Blauer See Delik...	Sales Represen...	Germany	Contacts02
Frédérique Cite...	Blondesddsl père...	Marketing Mana...	France	Contacts02

Getting distinct values

```
<sqlOp op="distinct">
   <dstTableName>DistinctComponents</dstTableName>
   <tableName>components</tableName>
   <fieldName>Component</fieldName>
</sqlOp>
```

The **distinct** operation lets you select unique items from the result set specified by <tableName> based on the column name specified by the <fieldName> element. the <fieldName> element might contains several column names separated by a semi colon.

The **distinct** operation is especially useful for creating a report dataset used to set the available values of a parameter. Say you have a SharePoint Products list with a Category column and you would like to let the user of your report select at run time the category of products that will be displayed in the report. You can easily achieve this by creating a specific report dataset as shown in the following image :

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
   <list title="Products" relativeSiteUrl="/sites/demo/" tableName="Products">
      <fields>Category</fields>
      <query></query>
   </list>
   <sqlOp op="distinct">
      <dstTableName>Distinct categories</dstTableName>
      <tableName>Products</tableName>
      <fieldName>Category</fieldName>
   </sqlOp>
   <resultSet>Distinct categories</resultSet>
</root>
```

Getting a subset of a result set

The select operation lets select a subset of the result set specified by <sourceTableName> based on the filtering expression specified by <selectExpression>.

```
<sqlOp op="select">
   <dstTableName>testSelect</dstTableName>
   <sourceTableName>Products</sourceTableName>
   <selectExpression>UnitPrice > 100</selectExpression>
</sqlOp>
```

The filtering expression is equivalent to the datatable select method. Refer to the framework documentation for more details about expressions.

Though it is possible to filter SharePoint lists using CAML query, selecting a subset of a result set may be interesting in some complex queries involving several operations.

Report parameters may be used with the select expression as shown in the following example:

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
   <list title="Orders" relativeSiteUrl="/sites/demo/" tableName="Orders">
      <fields></fields>
      <query>
      </query>
   </list>

   <sqlOp op="select">
      <dstTableName>testSelect</dstTableName>
      <sourceTableName>Orders</sourceTableName>
      <selectExpression>OrderDate > '@orderDate!' </selectExpression>
   </sqlOp>
   <resultSet>testSelect</resultSet>
</root>
```

Operations reference

Op= "join"

This operation performs a join between two lists. The parameters are:

Element/Attribute	Description
<dstTableName></dstTableName>	Name of the destination table for the **join** operation. The result of the operation can be used as part of another operation by invoking this name.
<parentTableName></parentTableName>	Parent list used to execute the join operation. It is also possible to use the result of another operation.
<childTableName></childTableName>	Child table. It is also possible to use the result of another operation.

Element/Attribute	Description
<parentFieldName></parentFieldName>	Columns used to perform the join in the parent table. Each column must be separated by a comma.
<childFieldName></childFieldName>	Columns used to perform the join in the child table. Each column must be separated by a comma.

Op= "outerjoin"

This operation performs an outer join between two lists. The parameters are identical to those in the "join" operation.

Op= "union"

This operation performs a union between two lists. These are the parameters:

Element/Attribute	Description
<dstTableName></dstTableName>	Name of the destination table for the **union** operation. The result of the operation can be used as part of another operation by invoking this name.
<parentTableName></parentTableName>	Parent list. It is also possible to use the result of another operation.
<childTableName></childTableName>	Child list. It is also possible to use the result of another operation.
<labelColumn></labelColumn>	Indicates if you want to use a specific column that would make it possible to distinguish the records for each table. If <labelColumn> has a "true" value, a column bearing the name "rstLabel" will be automatically created in the result of the "union" operation. This column will be filled with the value of the <parentLabelValue /> element for the parent table records and with the value of the <childLabelValue/> element for the child table records. If <labelColumn> has a "false" value, no column will be created and the <parentLabelValue /> and <childLabelValue/> elements will not be used.
<parentLabelValue></parentLabelValue>	Value used to distinguish the parent table records.
<childLabelValue></childLabelValue>	Value used to distinguish the child table records.

Op= "distinct"

This operation makes it possible to obtain the distinct elements of a list based on specific columns. The parameters are:

Element/Attribute	Description
<dstTableName></dstTableName>	Name of the destination table for the **distinct** operation. The result of the operation can be used as part of another operation by using this name.
<tableName></tableName>	List for which you want distinct values. It is also possible to use the result of another operation.
<fieldName></fieldName>	Columns making it possible to determine distinct values. Each column must be separated by a comma.

Op=”select”

Element/Attribute	Description
<dstTableName></dstTableName>	Name of the destination table (or result set) that will contain a subset of the result set represented by <sourceTableName>.
<sourceTableName></sourceTableName>	Source table (or result set).
<selectExpression></selectExpression >	Filter expression.

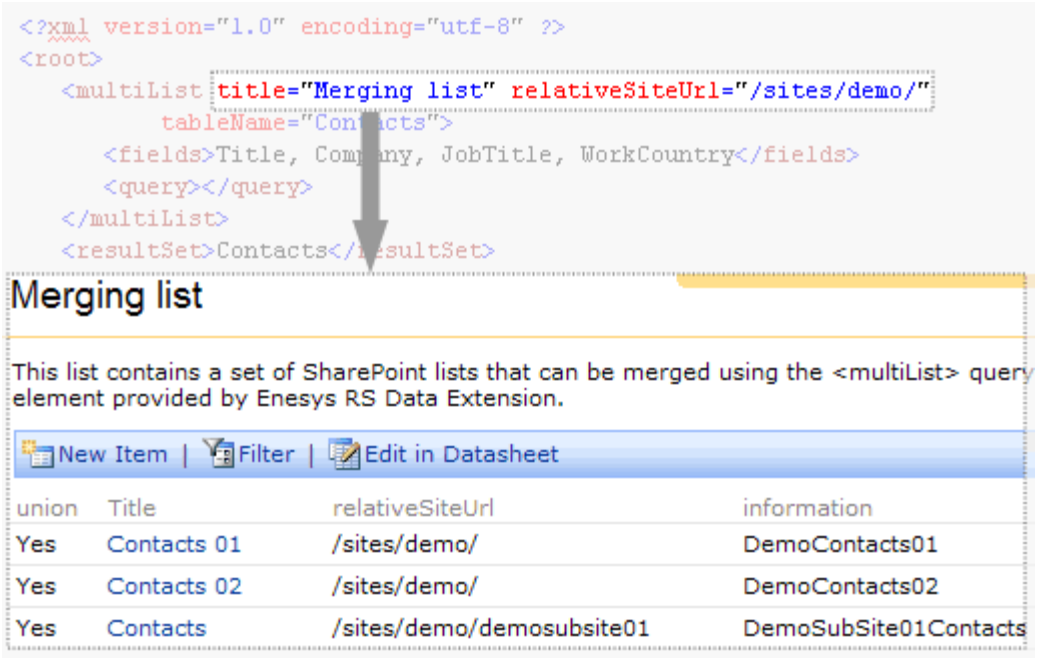
Merging multiple lists

This feature is not available with the Community Edition.

Enesys RS Data Extension provides a specific feature for merging a variable number of lists using one operation. In fact, it is a matter of applying a "union" operation to a set of SharePoint lists with no need to write a (<sqlOp/>) operation for each of the lists.

So that this operation can be dynamic and not require the modification of the report each time you want to add a list, the definition of the lists to be merged are located in a SharePoint list.

The <multiList> query element lets you specify a SharePoint list that will contain the name and relative site url of the SharePoint list you want to merge. We will use the term "merging list" to refer to this list.

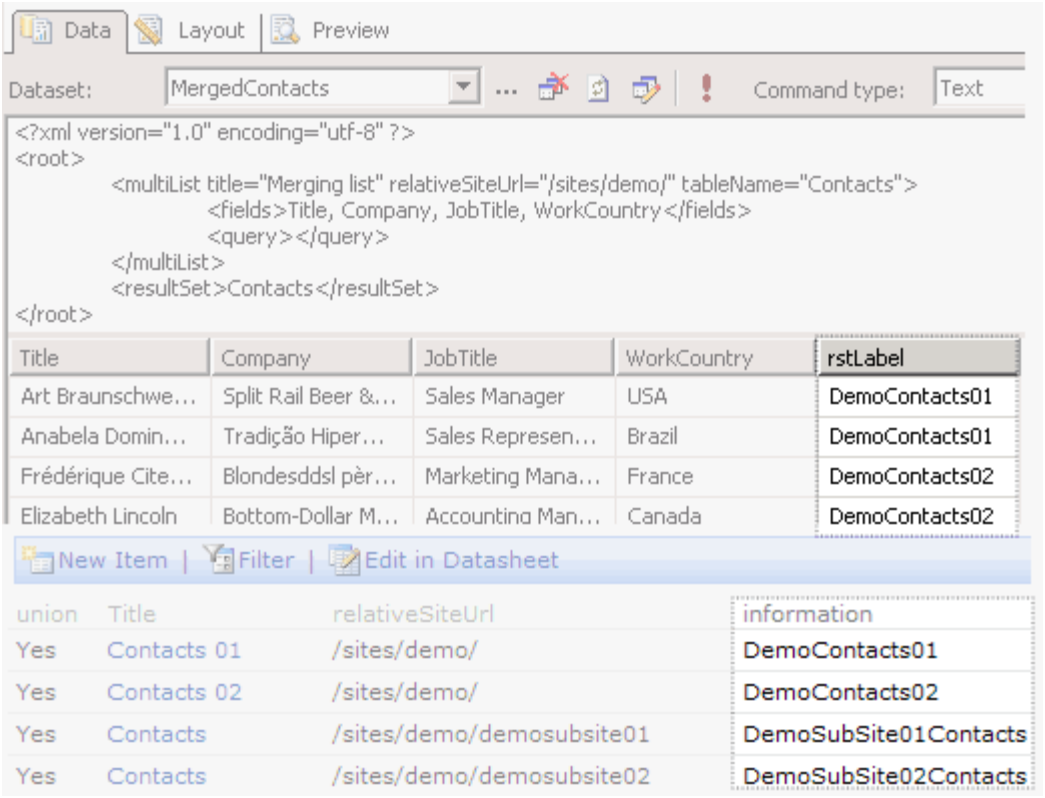


The merging list must have specific columns that will be described later.

The SharePoint lists that are merged are not necessarily completely identical as you can select the set of columns that will be merged using the <fields> element. Moreover, you can specify a filter that will be applied to each list before being merged:

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
   <multiList title="Merging list" relativeSiteUrl="/sites/demo/"
         tableName="Contacts">
      <fields>Title, Company, JobTitle, WorkCountry</fields>
      <query></query>
   </multiList>
   <resultSet>Contacts</resultSet>
</root>
```

A specific column named "**rstLabel"** is added to the result set obtained from the <multiList> query element. This column will be filled with the value of the merging list's column **Information**:



Attributes and child elements of the <multiList> element.

Element/Attribute	Description
title	Title of the SharePoint list that will contain the definition of the lists to be combined.
relativeSiteUrl	URL of the SharePoint site containing the list.
tableName	Name assigned to the data resulting from operation.
useDisplayName	This attribute lets you indicate that you want to use the display names of the columns rather than their internal names in the CAML format query and the <fields></fields> element. See the <list/> element for more information.
<fields></fields>	Limits the data from each of the lists to the columns defined in the context of this element. Each column name must be separated by a "**,**" (comma). The column's display name must be used. Note that a column named "rstLabel" is systematically added to the result obtained. This column will contain for each element of a given list the value of the "information" column that corresponds to the definition of the list. For more information, see the set of columns in the SharePoint list

Element/Attribute	Description
	that defines the lists to be combined.
<query></query>	CAML-type query that will be applied to each of the lists. See the <list/> element for more information.

Merging list

The definition of the lists to be merged must be populated in a SharePoint list. It is this list that is designated by the title attribute in the <multiList> element.

Each record in the list will represent the definition of a SharePoint list to be merged.

The list must contain the following columns:

Column - type	Description
title - text	Title of the SharePoint list that will be combined during the <multiList> operation
relativeSiteUrl - text	Relative URL of the SharePoint site containing the list (e.g. /sites/demo/).
information - text	To make it possible to distinguish the origins of the combined data, the content of this column is added to each record in each of the lists as part of the "rstLabel" column that is systematically added.
union - Yes/No	Only the lists for which this field is "Yes" shall be combined. Using such a field makes it possible to avoid deleting and re-entering the definition of lists each time you want to temporarily restrict the operation to a sub-set of the lists.

Retrieve sites data, roles and permissions

This feature is not available with the Community Edition.

Starting from version 1.3, Enesys RS Data Extension provides new query elements for retrieving sites and lists information such as list collection for a site, list permissions, web permissions, etc.

List collection

The <listCollection> query element lets you retrieve information about all the lists in a site or a set of sites.

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
   <listCollection  relativeSiteUrl="/sites/demo/" includePermissions="true"
         tableName="lists">
   </listCollection>
</root>
```

Optionally, you can retrieve the permissions as well for each list by setting the includePermissions attribute to true.

List permissions

The listPermissions query element lets you retrieve permissions from a specific list.

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
   <listPermissions listID="@List!" relativeSiteUrl="@SiteUrl!" tableName="list">
   </listPermissions>
</root>
```

Parameters may be used for the listID and relativeSiteUrl attributes values.

Web permissions

The <webPermissions> query element lets you retrieve the permissions given through site groups at the site level.

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
   <webPermissions  relativeSiteUrl="/sites/demo/" includeUsers="true"
                tableName="perms">
   </webPermissions>
</root>
```

Optionally, you can retrieve the users for each site group by setting the includeUsers attribute to true.

Cross-site groups

The <webGroups> query element lets you retrieve cross-site groups for a site or a set of sites.

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
   <webGroups  relativeSiteUrl="/sites/demo/"  includeUsers="true"
           tableName="groups">
   </webGroups>
</root>
```

Optionally, you can retrieve the users for each cross-site group by setting the includeUsers attribute to true.

Specifying a set of sites

Except <listPermissions>, All those commands share a common approach for specifying sites for which they apply.

You can specify the sites to handle in on of three ways:

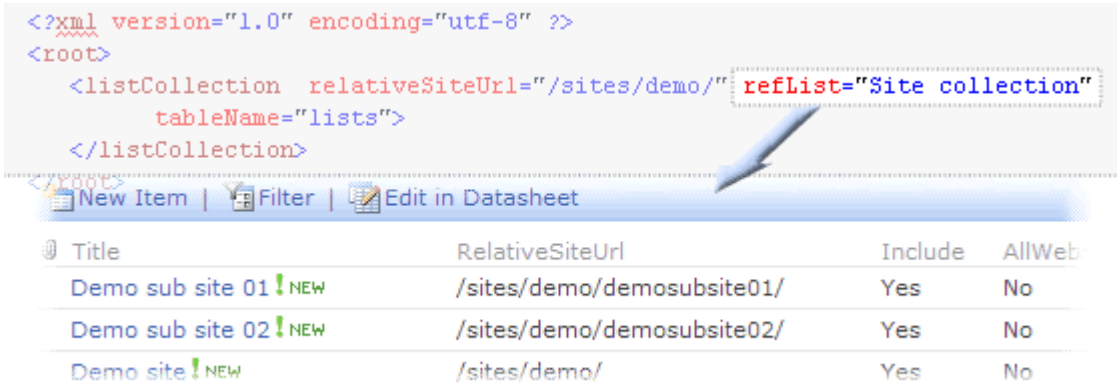
By writing directly the relative site url of the site (e.g.: /sites/demo/) within the query:

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
   <listCollection relativeSiteUrl="/sites/demo/" tableName="lists">
   </listCollection>
</root>
```

By using a report parameter so that the user might select the site when running the report:

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
   <listCollection relativeSiteUrl="@SiteUrl!" tableName="lists">
   </listCollection>
</root>
```

By specifying a SharePoint list that contains the list of sites to handle:



Sample reports

The "ReportSamples" zip files provided as part of the installation package includes report samples as well as a backed up SharePoint site containing sample lists used to run the sample reports.

Files details

After unzipping the file, you will get the following files and folders:

File/folder	Description
sites-demo.bak	Backed up SharePoint site containing sample lists used to run the sample reports. The site can be restored using the following STSADM command: *STSADM.EXE –o restore –url http://localhost/sites/demo -filename \ReportSamples\sites-demo.bak* If possible, keep demo as the restored site name so that you will not have to modify the "relativeSiteUrl" attribute for each sample report query. Please note that if you are running a non English SharePoint Server, you

	will need to install the English Language template pack before installing the demo site.
ReportSamples2000	This folder contains the sample reports in the form of a report designer solution for SQL Server 2000.
ReportSamples2005	This folder contains the sample reports in the form of a report designer solution for SQL Server 2005.

Sample reports

The following reports are provided as part of the solution:

Report	Description
IssuesGraphicalStats	Report file for the sample "Building a report showing graphical stats from a SharePoint Issues List". Complete Instructions for building this report is available on our web site.
IssuesGraphicalStats By Component	Report file for the sample "Using report parameters with SharePoint lists". Complete Instructions for building this report is available on our web site.
SalesByCategory	Report file for the sample "Joining SharePoint lists". Complete Instructions for building this report is available on our web site.
YearlySalesByCategory	Report file for the sample "Joining SharePoint lists". Complete Instructions for building this report is available on our web site.
IT - List Collection	Display the List collection for a set of sites. The sites to include in the report are specified within the SharePoint list "Site collection" located on /sites/demo/.
IT - List permissions	Display permissions for a specific list. This report is not intended (though it's possible) to be run directly. It's run from other reports .
IT - Lists and permissions for a specific site	Display lists and lists' permissions for a specific site.
IT - Lists and Permissions	Display the List collection and their permission for a set of sites located within the SharePoint list "Site collection" located on /sites/demo/.
IT - Site Groups and Users for a specific site	Display site groups as well as users belonging to the site groups for a selected site. At run time, the site is selected amongst sites specified in the SharePoint list "Site collection" located on the site "/sites/demo/"
IT - Sites Group details	Display Site groups details for a specific site. This report is not really intended to be run directly. It's used by other reports to display information details.
IT - Cross-Site Groups and	Display site groups as well as users belonging to the site groups for a selected site. At run time, the site is selected amongst sites specified in

Users	the SharePoint list "Site collection" located on the site "/sites/demo/"
Recurring Events	Simple report for testing recurring event expansion. The report will expand “Events” list recurring events for which “Test” column is true. This approach will let you test various cases independently.
Recurring Events Real World	More complex example showing how to retrieves events between 2 dates – both recurring and non recurring.
Stripping Html	Sample report showing the effect of stripping html using stripHtml attribute.
Running Values Sample	Sample report demonstrating how to add a custom column holding a running sum.
Select Operation with Parameter	This reports demonstrates how to select a subset of a result set using a report parameter.

The following reports provided as part of the report designer solution don’t have any report layout. They are intended to provide specific query syntax examples:

Report	Description
Query - Distinct values	Sample query to retrieve distinct values from a SharePoint list.
Query - Merging Lists	Sample query demonstrating the use of the "union" operator to merge two lists.
Query - Merging Multiple Contact Lists	Sample query demonstrating the use of the "multiList" query element for merging multiple lists.
Query - Pending Items	Sample query for retrieving pending items from a list.
Query - Simple Report	Sample query for retrieving products that belongs to a specific category. The category is specified at run time by the user running the report.
Query - Selecting a subset	Sample query demonstrating the use of “select” operation to get a subset of a result set.

Data source credentials

In order to retrieve SharePoint lists data, credentials information must be passed to SharePoint Web Services.

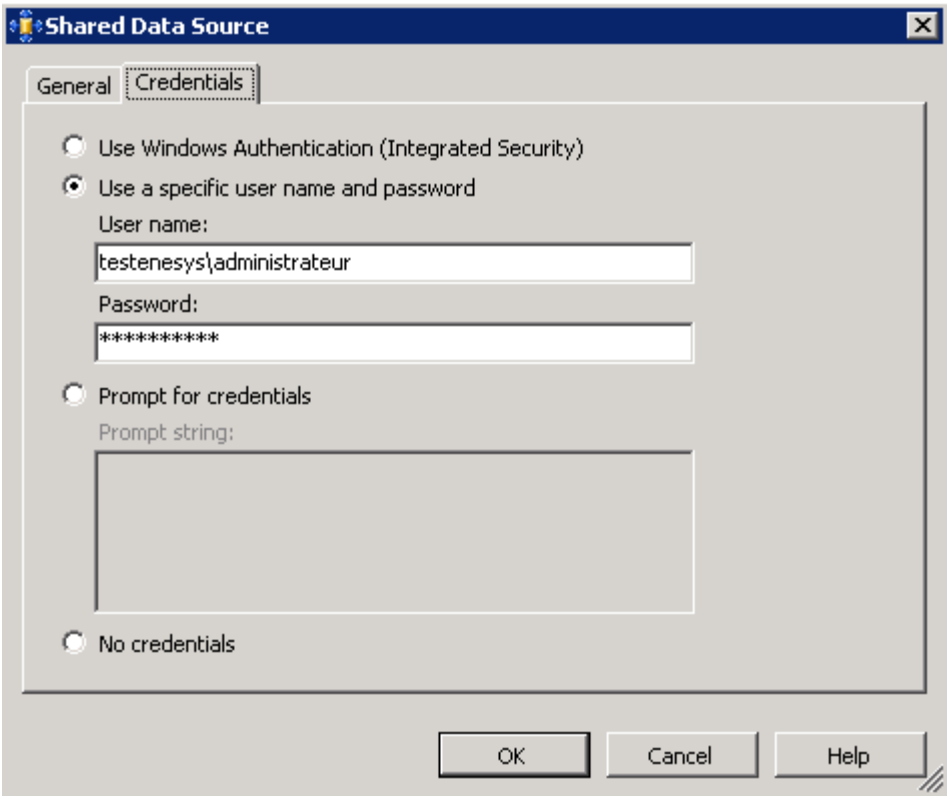
This chapter will explain the various credential options provided by Reporting Services and their effect on Enesys RS Data Extension.

For clarity, we will differentiate Report Designer and Report Server side.

Report Designer

When designing a report, you will use one or more datasets connected to a data source. Though it is possible to embed data source information within a report, we recommend using shared data sources for making future modifications easier.

For connecting to the data source, you must provide credentials information using the credential tab as shown in the following image:



Use Windows Authentication (Integrated Security)

When you use **Windows Integrated Security**, the credentials of the user currently designing the report will be passed to SharePoint Web Services. You will need the necessary rights on the SharePoint lists that will be retrieved using this data source.

Be aware that if Reporting Services is not on the same machine as SharePoint, you may need to deploy Kerberos delegation in order to pass credentials from the Report Server to SharePoint.

Use a specific name and password

When using this option, Enesys RS Data Extension will create network credentials for the account specified and will pass those credentials to SharePoint Web Services. The account specified must have the necessary rights for accessing the SharePoint lists used in the report.

Prompt for credentials

This option will only work in preview mode and will let you specify a run time the account that should be used to connect to SharePoint.

We do not recommend this option when designing a report as this will not work when you run the query in the data view.

No credentials

No credentials is not an option for Enesys RS Data Extension.

Report Server

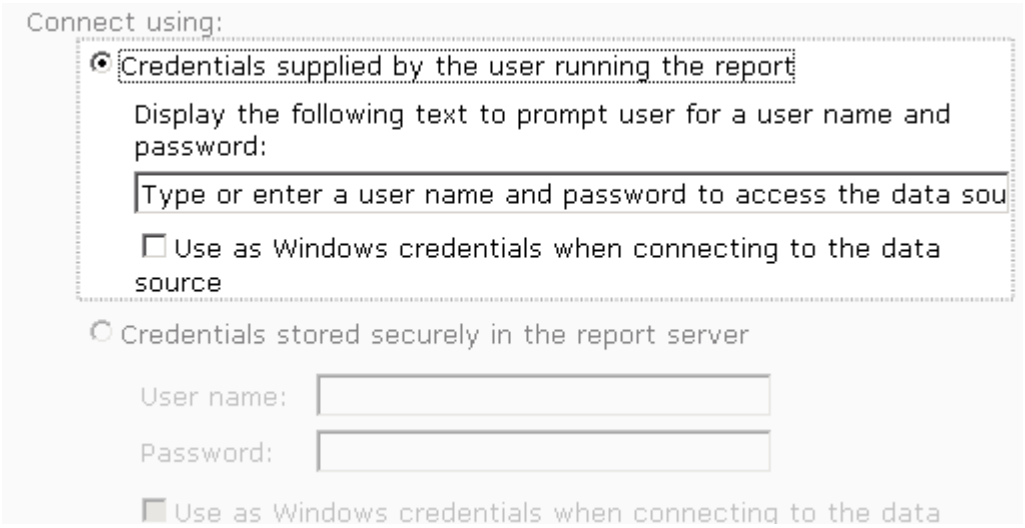
When deploying a report to the server, several cases must be considered.

If you have embedded the data source information within the report, you will end up with the same data source configuration on the server. At this stage, you may decide to use a shared data source or change the report-specific data source connection string or credentials.

If your report is using a shared data source in the report designer, the shared data source will be deployed on the server along the report unless a shared data source with the same name already exists in the server deployment path (unless you have configured your project to overwrite data sources). This is something important to note as you may end up with a completely different data source configuration once a report is deployed on the server. At this stage, you may configure the report to use a different data source or even create a report-specific data source configuration though we would not recommend this approach unless you have specific reasons to make it so.

Credentials supplied by the user running the report

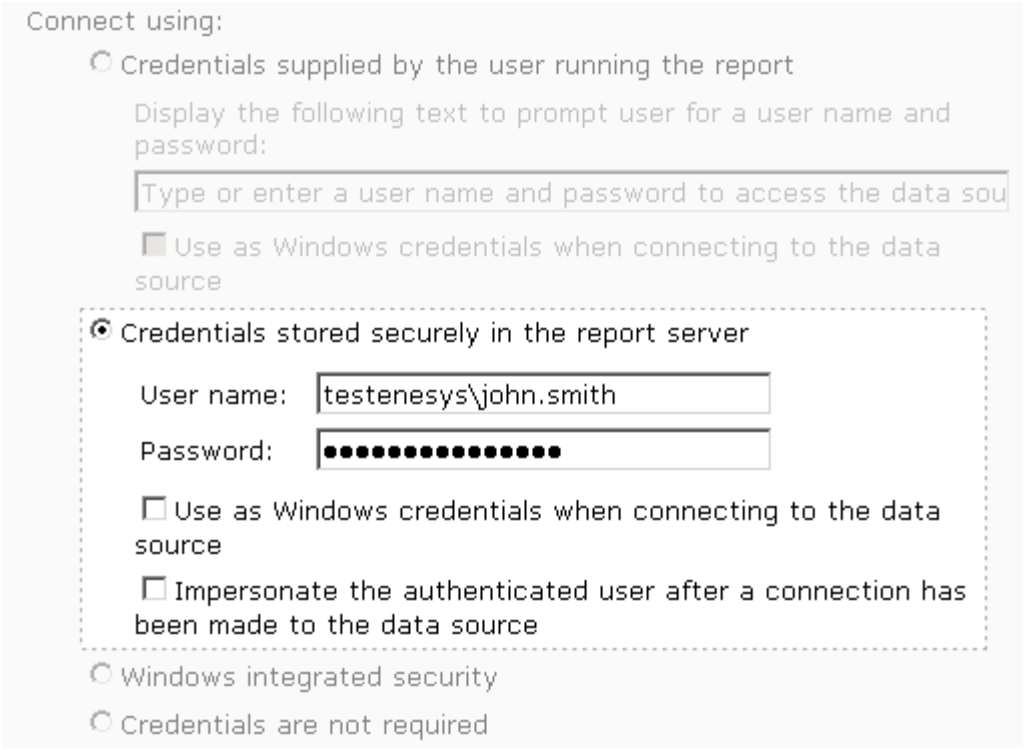
When using this option, the user running the report will be prompted to enter a user name and password. The credentials of the user account entered will be passed to SharePoint Web Services.



It is not necessary to check "Use as Windows credentials when connecting to the data source" as Enesys RS Data Extension will use network credentials anyway. This option will only makes a difference with data sources that may use different authentication schemes like SQL-Server.

Credentials stored securely in the report server

When using this option, the credentials of the user account entered and stored in the server will be passed to SharePoint Web Services.



It is not necessary to check "Use as Windows credentials when connecting to the data source" as Enesys RS Data Extension will use network credentials anyway. This option will only makes a difference with data sources that may use different authentication schemes like SQL-Server. Note however that checking this option will work properly though it will make a difference on how credentials are passed to Enesys RS Data Extension by the report server.

The "Impersonate the authenticated user after a connection has been made to the data source" is meaningless for Enesys RS Data Extension and will not be used whether you checked it or not.

Windows integrated security

When you use **Windows Integrated Security**, the credentials of the user running the report will be passed to SharePoint Web Services.

Credentials are not required

When using this option, the credentials of the unattended execution account will be used if it is configured.

This not a recommended option when using Enesys RS Data Execution.

Which credentials should you use?

Obviously, there is no definitive answer and it may depend on how you are organized.

However, from our experience on customer site, our preference goes to "credentials stored in the report server" for the following reasons:

- Stored credentials are a requirement for reports that run on a schedule (subscriptions).
- We will configure access security at the report or report server folder level so that users are not polluted by reports they won't be able to run.
- It is sometimes desirable to allow users run a report on a SharePoint list for which they don't have permissions. SharePoint doesn't offer the possibility to give read permissions on a subset of list's columns. Suppose you have a SharePoint contact list that you would like being available by everybody except for the home phone number that should only be available to managers. Unless you are ready to maintain two SharePoint lists, you may have one list for which only managers have permissions. For other users, you could build a report that would not display the home phone number column and deploy this report using stored credentials and give all users access rights to run the report.

Of course, you may use a mix of the credentials options available in order to achieve the same goals.