Reporting Guide for Microsoft Dynamics CRM

Version: 8.1

View the latest version of this document online.

Contents

Report writing with CRM 2016 for online and on-premises

Applies To: Dynamics CRM 2016, Dynamics CRM Online

Microsoft Dynamics CRM includes reports, dashboards, and support for Power BI for Office 365 that provides useful business information and visualizations to the user.

Paginated reports

 Microsoft Dynamics CRM includes a Report Wizard that can be used to easily create reports in just a few steps without using XML or SQL-based queries. For more information about the Report Wizard, see CRM Help & Training: Create, edit, or copy a report using the Report Wizard.

However, to create more complex reports, you can either create your own custom reports from scratch, or use an existing Microsoft Dynamics CRM report as a template. The topics in this guide show you how to create new reports or change existing reports using Microsoft Visual Studio as the report writing tool and Microsoft SQL Server Reporting Services as the report engine.

Dashboards

There are two types of dashboards in Microsoft Dynamics CRM, user dashboards and system dashboards. Any user can create a dashboard visible only to them in their work area such as Sales, Service, or Marketing. An administrator or customizer creates or customizes system dashboards that, when published, are visible to everyone in the organization. A user can choose to set their user dashboard as their default dashboard and override the system dashboard. More information: Work with, create, or customize dashboards

Power BI

Power BI is a self-service business intelligence (BI) platform used to discover, analyze, visualize data and share or collaborate these insights with colleagues. Power BI provides information workers and everyday business users with excellent data analysis and visualization capabilities to get better business insights. More information: Referenced topic '48997010-a47c-4e16-b7d2-f55d7a52ba19' is not in the TOC.

There are several ways you can use Power BI with Microsoft Dynamics CRM.

- Load a Microsoft Dynamics CRM content pack and start using the Power BI service to display CRM insights.

-     Use Power BI Desktop to modify and customize your reports and visualizations.

- Embed a Power BI tile in to a Dynamics CRM personal dashboard.

- Use Power BI and Microsoft Office Excel together.

 Note

In most cases report features are the same with either Microsoft Dynamics CRM Online or Microsoft Dynamics CRM (on-premises); however there are the following differences.

- SQL-based reports can only be used with Microsoft Dynamics CRM (on-premises). Working with SQL-based reports (CRM on-premises only)

- The Schedule Report feature in the Reports area of Microsoft Dynamics CRM lets users schedule report snapshots at certain intervals. This feature is currently available only with Microsoft

Dynamics CRM (on-premises). For more information about creating and editing reports in Microsoft Dynamics CRM, see Customize and organize reports.

- The reporting infrastructure in Microsoft Dynamics CRM Online has RDL sandboxing enabled. Therefore, custom code in report definitions will not work.   RDL sandboxing for Microsoft Dynamics CRM Online

In This Section

Get started writing reports

Working with Microsoft Dynamics CRM reports

Working with SQL-based reports (CRM on-premises only)

Test and troubleshoot reports

Publish reports

Report considerations and best practices

Example reports

Copy reports between Microsoft Dynamics CRM on-premises deployments

Related Sections

Referenced topic 'd0d49a86-6297-4431-8b30-1f477bca2bad' is not in the TOC.

Referenced topic '9f201f30-245a-458e-b15f-961a9d049ea7' is not in the TOC.

Referenced topic '633e9a2d-cba1-4700-ba18-01487767109c' is not in the TOC.

Referenced topic '605bb886-116a-4275-83fe-e9fdc83d2f2f' is not in the TOC.

Get started writing reports

Applies To: CRM 2016 on-prem, CRM Online

Microsoft Dynamics CRM uses Microsoft SQL Server Reporting Services report definition language (RDL) reports to query CRM data and return refined results back to the report user. For more information about RDL, see TechNet: Report Definition Language (SSRS).

To create or modify existing RDL reports that can be used with Microsoft Dynamics CRM, use either T-SQL or FetchXML, which is then converted to RDL by using report authoring tools. The following table lists the differences between SQL-based and Fetch-based reports in Microsoft Dynamics CRM.

Area	SQL-based report	Fetch-based report
Supported Microsoft Dynamics CRM Versions	Microsoft Dynamics CRM (on-premises)	Microsoft Dynamics CRM Online and Microsoft Dynamics CRM (on-premises)

Area	SQL-based report	Fetch-based report
Report Query Language	Uses Transact-SQL (T-SQL), a set of programming extensions that provide comprehensive transaction control by using Structured Query Language (SQL). More information:[TechNet: Transact-SQL Reference (Database Engine)](#)	Uses FetchXML, an extensible markup language (XML) designed specifically for Microsoft Dynamics CRM queries. More information: [MSDN: FetchXML schema](#)
Requires Report Authoring Extension?	No	Yes
.RDL file Data Provider	The `<DataProvider>` element value in the .rdl file is set to **SQL**. For example: `<DataProvider>SQL</DataProvider>`	The `<DataProvider>` element value in the .rdl file is set to **MSCRMFETCH**. For example: `<DataProvider>MSCRMFETCH</DataProvider>`

.RDL file Report Query	The query specified for retrieving data is in the `<CommandText>` sub-element under the `<Query>` element in the report definition (.rdl file) and is a SQL query. For example, the query for retrieving all account names for a SQL-based report will be:
	
	`<CommandText>SELECT name FROM FilteredAccount;</CommandText>`
		The query specified for retrieving data is in the `<CommandText>` sub-element under the `<Query>` element in the report definition (.rdl file) and is a FetchXML query. For example, the query for retrieving all account names for a Fetch-based report will be:
		
		`<CommandText><fetch version="1.0"`
		
		`output-format="xml-platform"`
		
		`mapping="logical">     <entity`
		
		`name="account">        <attribute`
		
		`name="name" />`
		
		`</entity></fetch></CommandText>`

If you want, you can use a third-party tool, SQL2FetchXML, to convert your SQL scripts to FetchXML, and then use the FetchXML query in your Fetch-based reports. More information: [SQL2FetchXML Help](#)

In This Section

This section covers what you need to create a new Microsoft Dynamics CRM report.

[Report writing environment using SQL Server Data Tools](#)
[Create a new report using SQL Server Data Tools](#)

Related Sections

[Report writing with CRM 2016 for online and on-premises](#)
[Working with SQL-based reports (CRM on-premises only)](#)

Report writing environment using SQL Server Data Tools

Applies To: CRM 2016 on-prem, CRM Online

This topic describes what's needed to author Microsoft Dynamics CRM reports. For Microsoft Dynamics CRM (on-premises), this topic assumes you already have a functioning deployment of Microsoft Dynamics CRM Server. For information about Microsoft Dynamics CRM (on-premises) requirements for reporting, see Referenced topic '6d0d42e2-0ad0-4dfa-aa42-72ab4e92b001' is not in the TOC..

You must have the required development tools and appropriate privileges in Microsoft Dynamics CRM to write and publish a report. Also, you should be familiar with:

- Microsoft Visual Studio.

- Creating Microsoft SQL Server Reporting Services reports using SQL Server Data Tools (SSDT).

- To write Fetch-based reports, the Microsoft Dynamics CRM FetchXML language.

- To write SQL-based reports, the Transact-SQL language for Microsoft SQL Server.

Required tools

The following are required to write a custom report for Microsoft Dynamics CRM:

- **Microsoft SQL Server Reporting Services**. Microsoft Dynamics CRM uses Microsoft SQL Server Reporting Services as the report engine.

- **Microsoft Visual Studio 2012** or **Microsoft Visual Studio 2010**.

- **SQL Server Data Tools**. This is a report authoring environment that is used as a plug-in Microsoft Visual Studio.

  - For Microsoft Visual Studio 2012: You must download and install Download: Microsoft SQL Server Data Tools - Business Intelligence for Visual Studio 2012.

  - For Microsoft Visual Studio 2010: You must select and install the SQL Server Data Tools (SSDT) feature that is included with Microsoft SQL Server 2012 or Microsoft SQL Server 2012 Express on the computer that is running Microsoft Visual Studio 2010. Download: Microsoft SQL Server 2012 Express

- **Microsoft Dynamics CRM Report Authoring Extension**. This is required if you are writing custom Fetch-based reports. Notice that Microsoft Dynamics CRM Online only supports Fetch-based reports. Microsoft Dynamics CRM Report Authoring Extension must be installed on the computer where Microsoft Visual Studio and SQL Server Data Tools are installed. Download: CRM 2016 Report Authoring Extension. For installation instructions, see Install Microsoft Dynamics CRM Report Authoring Extension.

Note

Microsoft Dynamics CRM Report Authoring Extension is available only in a 32-bit version.

Required privileges

To deploy custom reports to Microsoft Dynamics CRM, you must have a Microsoft Dynamics CRM account and a security role assigned to you that includes the **PublishReport** privilege. By default, the System Customizer and System Administrator security roles include these privileges.

Report development process

The following lists the steps for developing custom Microsoft Dynamics CRM reports. You may have to repeat some steps while you develop a report:

1. Develop a report concept or specification based on what business information is to be displayed.

2. Decide on the type of report you want to create: Fetch-based or SQL-based. Microsoft Dynamics CRM Online users can only create custom Fetch-based reports. More information: Report writing with CRM 2016 for online and on-premises

3. Create a custom report or use an existing report to modify using SQL Server Data Tools in Microsoft Visual Studio.

    *   Create a new (custom) report. More information: Create a new report using SQL Server Data Tools

    *   Download an existing Microsoft Dynamics CRM report definition language (.rdl) file. You can do this in the Microsoft Dynamics CRM web application. More information: Modify an RDL file

        Alternatively, for Microsoft Dynamics CRM (on-premises), reports are located in the C:\Program Files\Microsoft Dynamics CRM Reporting Extensions\LangPacks\<lcid>\Reports\MSCRM\ folder where Microsoft Dynamics CRM Reporting Extensions is installed. More information: Modify an existing SQL-based report using SQL Server Data Tools

4. Create basic report parameters. More information: Use parameters in reports

5. Specify datasets and filtering criteria for retrieving data:

    *   For SQL-based reports, create datasets that contain Microsoft Dynamics CRM data obtained from the filtered views. More information: Filtered views in Microsoft Dynamics CRM

    *   Enable pre-filtering on the primary entities. More information: Improve report performance by using filters

6. Define the basic layout of the report, including headers and footers.

7. Add report items as required based on the report specification. More information: Add report navigation

8. Preview the report in Microsoft Visual Studio, and resolve any errors. More information: Test and troubleshoot reports

9. Deploy the report to the reporting server by using Microsoft Dynamics CRM. More information:
    Publish reports

10. Run the deployed report to verify.

See Also

Use SQL and filtered views to retrieve data for reports
Report writing with CRM 2016 for online and on-premises
Create a new report using SQL Server Data Tools
Getting Started with Custom Reports in the Cloud
Business Intelligence Development Studio
Report Designer and Business Intelligence Development Studio

Create a new report using SQL Server Data Tools

Applies To: CRM 2016 on-prem, CRM Online

SQL Server Data Tools is a report authoring environment that lets you create or edit Microsoft SQL Server Reporting Services reports in Microsoft Visual Studio. The end result is a report definition .rdl file that contains the report definition that you can publish in Microsoft Dynamics CRM to view reports.

Alternatively, you can author reports by using a common text editor. To reduce the effort to create a custom report, use an existing .rdl file that provides most of the desired functionality and modify it. For more information about the format of the XML elements in an .rdl file, see Report Definition Language Reference. The modified report XML can be verified by using the specified XML schema. Reporting Services will also verify the report definition and reject a report if the definition is invalid when you try to upload the report in Microsoft Dynamics CRM.

🖉 Note

If the .rdl file contains a FetchXML query, the query in the RDL is validated by Microsoft Dynamics CRM Report Authoring Extension, which internally validates it against the FetchXML schema. For more information, see MSDN: Fetch XML Schema.

In This Topic

Create a custom Fetch-based report (CRM Online and CRM on-premises)
Create a custom SQL-based report (CRM on-premises only)

Create a custom Fetch-based report (CRM Online and CRM on-premises)

To create a custom Fetch-based report:

1. Make sure that you have a supported version of Microsoft Visual Studio, SQL Server Data Tools, Microsoft Dynamics CRM Report Authoring Extension and the necessary privileges. More information: Report writing environment using SQL Server Data Tools

2. Open Microsoft Visual Studio, and create a report server project.

3. In Solution Explorer, right-click the **Reports** folder, and then click **Add New Report**.

4. Click **Next**.

5. On the **Select the Data Source** page, click **New Data Source**, and specify the following details:

    *   **Name**: Type a name for the data source.

    *   **Type**: Select **Microsoft Dynamics CRM Fetch**.

    *   **Connection String**: Specify the connection string. The connection string must be specified in the following format:

        ServerURL;*OrganizationName*;*HomeRealmURL*

        In this connection string, only *ServerURL* is mandatory. If *OrganizationName* isn't specified, the first organization that the user running this query belongs to is used. *HomeRealmURL* is the Home Realm URL of the Identity Provider used by your organization and is needed when your organization uses Federation for identity management. Contact your network administrator to determine the Home Realm URL.

        Click **Credentials** to specify the credentials to connect to Microsoft Dynamics CRM or Microsoft Dynamics CRM Online, and click **Next**.

6. On the **Design the Query** page, type the FetchXML query in the **Query** box. To get this query, you can do one of the following:

    *   Get the FetchXML from an Advanced Find query. To do this, open Microsoft Dynamics CRM, click **Advanced Find**, create the query that you want, and then on the **Advanced Find** tab, click **Download Fetch XML**. Copy the FetchXML into the **Query** box of the Dataset Properties in Microsoft Visual Studio.

    *   Manually enter the FetchXML query. The following example shows how to create a report that displays all accounts with 5000 or more employees.

        ```
        <fetch version="1.0" output-format="xml-platform" mapping="logical" distinct="false">

          <entity name="account">

            <attribute name="name" />

            <attribute name="numberofemployees" />

            <attribute name="accountnumber" />
        ```

```
       <order attribute="name" descending="false" />

       <filter type="and">

         <condition attribute="numberofemployees" operator="gt" value="5000" />

       </filter>

     </entity>

   </fetch>
```

More information: MSDN: Build queries with FetchXML in the Microsoft Dynamics CRM SDK.

Click **Next**.

7. Verify the fields that will be included in the report. Click **Next**.

8. Select a style to apply to the report, and then click **Next**.

9. Verify the fields that will be included in the report, and enter a name for the report, such as *Accounts With More Than 5000 Employees*. Click **Finish**.

10. If you'd like to see how the report will appear when it's run, click the **Preview** tab.

This generates an .rdl file with the specified report name. You can use this file to publish your custom report in Microsoft Dynamics CRM Online using the Report Wizard. More information: Publish reports.

Create a custom SQL-based report (CRM on-premises only)

To create a custom SQL-based report using SQL Server Data Tools:

1. Make sure that you have a supported version of Microsoft Visual Studio and SQL Server Data Tools and the necessary privileges. More information: Report writing environment using SQL Server Data Tools

2. Open Microsoft Visual Studio, and create a report server project.

3. In Solution Explorer, right-click the **Reports** folder, and then click **Add New Report**.

4. Click **Next**.

5. On the **Select the Data Source** page, click **New data source**, and specify the following details:

    - **Name**: Type a name for the data source.

    - **Type**: Select **Microsoft SQL Server**.

    - **Connection String**: Specify the connection string to connect to the instance of the Microsoft SQL Server database. To build the connection string, and click **Edit** to type the SQL Server name and *organizationName*_MSCRM database. To supply credentials, select **Credentials**. Click **Next**.

6. On the **Design the Query** page, type the SQL query to use for the report, and click **Next**. For example, to create a report that displays all accounts with 5000 or more employees, where *OrgName*_MSCRM is the name of the organization database, use this query.

```
Use OrgName_MSCRM Select Name, AccountNumber, NumberofEmployees from AccountBase

where NumberofEmployees > 5000 order by NumberofEmployees desc
```
Alternatively, you can design a query by choosing **Query Builder**.

7. On the **Select the Report Type** page, select a **Tabular** report or a **Matrix** report, and click **Next**.

8. Verify the fields that will be included in the report. Click **Next**.

9. Select a style to apply to the report, and then click **Next**.

10. Verify the fields that will be included in the report, and enter a name for the report, such as *Accounts With More Than 5000 Employees*. Click **Finish**.

11. If you'd like to see how the report will appear when it's run, click the **Preview** tab.

This generates an .rdl file with the specified report name. You can use the .rdl file to publish your custom report in Microsoft Dynamics CRM. More information:Publish reports.

For more information about how to create a report by using the Report Designer, see Create a Basic Table Report (SSRS Tutorial).

See Also

Report writing environment using SQL Server Data Tools
Modify an existing SQL-based report using SQL Server Data Tools
Blog: Getting Started With Custom Reports In The Cloud

© 2016 Microsoft. All rights reserved. Copyright

Working with Microsoft Dynamics CRM reports

Applies To: CRM 2016 on-prem, CRM Online
This section describes the different features that are available to determine how reports display in Microsoft Dynamics CRM.

In This Section

Format report content
Add report navigation
Categorize and display reports in different languages
Use parameters in reports

See Also

Report writing with CRM 2016 for online and on-premises
Working with SQL-based reports (CRM on-premises only)
Publish reports

Format report content

Applies To: CRM 2016 on-prem

You can obtain the Microsoft Dynamics CRM organization formatting values for date, time, number, and currency by using the **fn_GetFormatStrings** SQL function that is available in the Microsoft Dynamics CRM database. This function returns a single row data table that contains the formatting values. To view the contents of the returned table, execute the following SQL query on the database server.

```
USE <organization>_MSCRM

SELECT * FROM dbo.fn_GetFormatStrings()
```

Use formatting values in reports

1. Create a dataset to contain the formatting data. For information about how to create a dataset, see [Create a Shared Dataset or Embedded Dataset (Report Builder and SSRS)](#).

    a. Name the dataset DSNumandCurrency or use another name. DSNumandCurrency is the dataset name that is used in reports that are included with Microsoft Dynamics CRM.

    b. Use the SQL **SELECT** statement described above to fill the dataset.

2. Reference the format field, such as date and time, number, or currency (described below) from the dataset in the **Format** property of the report item that you want to format.

📝 Note

- For date, time, and currency formatting, set the **Language** property of the report item or the report to `"=First(Fields! NumberLanguageCode.Value, "DSNumandCurrency")"`.

Date and time values

For date and time, you also set the **Calendar** property of the report item to `"=First(Fields!CalendarType.Value, "DSNumandCurrency")"`.

Formatting string	**Report item** Format **property value**
Date	`=First(Fields!DateFormat.Value, "DSNumandCurrency")`
Time	`=First(Fields!TimeFormat.Value, "DSNumandCurrency")`

Number values

The **fn_GetFormatStrings** function returns the number of format strings with precision values between 0 and 5. You can specify a precision by putting a decimal value between the underscore characters in the field name.

Formatting string	Report item Format property value
Integer	`=First(Fields!NumberFormat_0_Precision.Value, "DSNumandCurrency")`
Decimal with 2-decimal points precision	`=First(Fields!NumberFormat_2_Precision.Value, "DSNumandCurrency")`

Base currency value

The **fn_GetFormatStrings** function returns base currency format strings with precision values between 0 and 5. You can specify a precision by putting a decimal value between the underscore characters in the field name.

Formatting string	Report item Format property value
Base currency with 2-decimal points precision	`=First(Fields!CurrencyFormat_2_Precision.Value, "DSNumandCurrency")`

Transaction currency

When you create a report on an entity with the transaction currency information, you can retrieve the transaction currency format string from the crm_moneyformatstring column of the filtered view for an entity. After a column has been added to the dataset, you can reference the column on the **Format** property of the report. For information about how to add columns to a dataset, see MSDN: How to: Add, Edit, or Delete a Field in the Report Data Pane. For example, to retrieve a price unit and the transaction currency formatting information from the quote detail filtered view, use the following **SELECT** statement:

```
SELECT priceperunit, crm_moneyformatstring FROM FilteredQuoteDetail
```

You can reference a new column on the **Format** property of the report item as follows:

```
=Fields.crm_moneyformatstring.Value
```

See Also

Working with SQL-based reports (CRM on-premises only)
Add report navigation

Add report navigation

Applies To: CRM 2016 on-prem, CRM Online

Report navigation enables a dynamic and interactive reporting experience. By using various types of actions, reports let the user navigate to detailed reports, Microsoft Dynamics CRM records, or other websites.

 Note

For more information about report navigation, see Drillthrough, Drilldown, Subreports, and Nested Data Regions (Report Builder and SSRS) in the Microsoft SQL Server documentation. This topic describes report navigation that is specific to Microsoft Dynamics CRM reports.

Dynamic drill through to Microsoft Dynamics CRM

You can drill through a report to navigate to a Microsoft Dynamics CRM web form. A drill-through report is implemented in the following steps:

1. An image or value item (such as a text box) is added to a report. The **Value** property of this item contains code that builds a URL by using the base address of Microsoft Dynamics CRM plus parameters that refer to a specific record.

2. When the user selects the report item, a new browser window is opened by using the constructed URL passed as the target web address.

3. Microsoft Dynamics CRM loads the information for the specified entity into a web form that is displayed in the browser window.

To set up a drill-through report in Microsoft Dynamics CRM

1. Create a hidden parameter of type string in the report that has the name CRM_URL. For more information about adding parameters, see Use parameters in reports. When the report is run, this parameter is automatically set to the web address of Microsoft Dynamics CRM.

2. Add a report item, such as a **Textbox**.

3. Right-click the drill-through report item and select **Properties** from the shortcut menu.

4. Click **Advanced**.

5. In the **Navigation** tab, click **Jump to URL** and enter an expression in the following format:

    ```
    = Parameters!CRM_URL.Value & "?ID={"& GUID &"}&LogicalName=entity logical name"
    ```

    The entity GUID and entity logical name have to be added to the URL to be able to drill through. For example:

    ```
    = Parameters!CRM_URL.Value &

    "?ID={"&Fields!Opportunityid.Value.ToString()&"}&LogicalName=opportunity"
    ```

6. Click **OK**.

In this example code, the value of a dataset field that contains the GUID of an **Opportunity** object is converted to a string and used as an ID parameter in the URL. A parameter that contains the LogicalName value for an opportunity entity is also appended.

The GUID of a record can be obtained from the appropriate filtered view, for example, **FilteredOpportunity**.

See Also

Report writing with CRM 2016 for online and on-premises
Categorize and display reports in different languages

Categorize and display reports in different languages

Applies To: CRM 2016 on-prem, CRM Online

You can enable additional languages in Microsoft Dynamics CRM by enabling Language Packs. This lets you display text in the user interface, Help, and reports in different languages. For more information about how to enable Language Packs, see Enable Languages. For Microsoft Dynamics CRM (on-premises), you must download and install Language Packs before you enable them. More information: Install and enable a Language Pack

To categorize the reports by language, use the **Report.LanguageCode** attribute. You can set the attribute to a specific locale ID (for example, 1033 for US English) to make the report visible to the users of that language. For example, the English out-of-the-box Account Summary report appears in the Reports grid in the English user interface, but not in the Spanish or German user interfaces in the same organization.

You can also set the **Report.LanguageCode** attribute to -1 (minus one) to make the report visible to all users in the base language user interface, which is installed during the original Microsoft Dynamics CRM server installation, and in the user interfaces in other languages. For information about the valid Locale ID values, see MSDN: Microsoft Locale ID Values.

You can use the report language information in combination with information that is contained in the report entity, report category, and report visibility entities to determine the areas and categories in Microsoft Dynamics CRM where the report is shown in different user interfaces languages.

 Note

The **Language** element inside the report definition language (RDL) file does not determine where the report is shown in Microsoft Dynamics CRM. It contains an expression that evaluates to a language code as defined in the Internet Engineering Task Force (IETF) RFC1766 specification. The language code is used mainly for formatting numbers, dates, and times for a specified language. More information:Language Element (Report) (RDL).

See Also

Use parameters in reports

Applies To: CRM 2016 on-prem, CRM Online

You use parameters in reports to control the data retrieved by prompting the user for a value or a set of values when the user runs the report. The dataset query retrieves only the data that is requested by the user. You can also add hidden and special parameters in the reports that do not prompt the user for input, but can be used for operations such as data filtering and dynamic drill-through.

📝 Note

The maximum length of the parameter values that is passed in from Microsoft Dynamics CRM is 2,000 characters. For example, if you run a report and create a data filter through the **Advanced Find** user interface, the resulting filter expression that is passed to a filter parameter cannot exceed 2,000 characters. There is no maximum limit on the number of parameters that you can specify. However, you might have to limit the length of the string on the URL line and number of parameters to meet the requirements of a particular browser.

In this topic

Adding parameters
Hidden parameters

Adding parameters

You can add parameters to a report to define a report's individual parameters, pass information through a query, or provide access to user settings, such as **CRM_CurrencySymbol** and **CRM_CurrencyPositivePattern** parameters.

The <ReportParameter> is an element in the report definition (RDL) file that is used to describe an individual parameter in the report. The <QueryParameter> contains information about an individual parameter that is passed to the data source as part of a query. The following XML code taken from the Account Summary report's RDL file demonstrates how to use the **ReportParameter** and **QueryParameter** parameters.

```
<

ReportParameter
```

```xml
            Name
            ="CRM_FilteredAccount"
            >
  <DataType>String</DataType>
    <Nullable>true</Nullable>
    <DefaultValue>
      <Values>
        <Value>select * from FilteredAccount</Value>
      </Values>
    </DefaultValue>
    <AllowBlank>true</AllowBlank>
    <Prompt>CRM_FilteredAccount</Prompt>
</ReportParameter>


          <
          Query
          >
  <rd:UseGenericDesigner>true</rd:UseGenericDesigner>
  <CommandText>declare @sql as nVarchar(max)
     set @sql = '
     SELECT top 10 CAST(accountid as nvarchar(100)) as AccountID,
     name, '''' as None
     FROM (' + @FilteredAccount + ') as fa'
     exec(@sql)
  </CommandText>
  <QueryParameters>
    <QueryParameter Name="@FilteredAccount">
      <Value>=Parameters!FilteredAccount.Value</Value>
    </QueryParameter>
  </QueryParameters>
  <DataSourceName>CRM</DataSourceName>
</Query>
```

The following examples show how to use the **QueryParameter** and **ReportParameter** parameters in a **Fetch** based report:

```
<
ReportParameter
 Name
="FilteredAccount"
>
<DataType>String</DataType>
<Prompt>Filtered Account</Prompt>
      <DefaultValue>
            <Values>
                  <Value>
                        <fetch version="1.0" output-format="xml-platform"
mapping="logical" distinct="false">
                        <entity name="account">
                        <all-attributes/>
                        </entity>
                        </fetch>
                  </Value>
            </Values>
      </DefaultValue>
</ReportParameter>


<
Query
>
<DataSourceName>DataSource1</DataSourceName>
<CommandText>
      <fetch>
      <entity name="account" enableprefiltering="true"
prefilterparametername="FilteredAccount">
            <attribute name="accountid" />
```

```
                    <attribute name="name" />

                </entity>

            </fetch>

        </CommandText>

    <QueryParameters>

        <QueryParameter Name="FilteredAccount">

            <Value>=Parameters!FilteredAccount.Value</Value>

        </QueryParameter>

    </QueryParameters>

    <rd:UseGenericDesigner>true</rd:UseGenericDesigner>

    </Query>
```

Hidden parameters

The Report Designer in Microsoft Visual Studio has built-in support for hidden parameters. In addition, you can hide parameters by adding a CRM_ prefix to the parameter name in a Microsoft Dynamics CRM report. By default, the parameters with a CRM_ prefix are hidden when the report is published through Microsoft Dynamics CRM. When you run the report, you aren't prompted to enter parameter values for the hidden parameters.

Special parameters

The following table shows the special hidden parameters that you can use in your reports.

Parameter	Description
CRM_FilterText	Contains the value of the filter text that a report user interactively creates in the Report Viewer when the user runs a report. The parameter is in a filter summary text box that is located in the report header. The initial value is set to the default filter.
CRM_URL	Set to the URL of the Microsoft Dynamics CRM Web application. Use this parameter when drilling through to Microsoft Dynamics CRM.
CRM_FilteredEntity	Use in a query expression to enable data pre-filtering (through **Advanced Find**).

You must create all parameters in a report before you can refer to them. The values of these special parameters are filled in by Microsoft Dynamics CRM when you run the report.

Additional format parameters

The following table contains additional parameters that you can use in the reports. Among them are parameters that provide access to the user Number settings information. You can use these values to format and display the numeric values. These parameters are similar to values specified in the NumberFormatInfo Class. Use these parameters in custom reports to format the data according to the user settings.

Parameter	Description
CRM_FullName	The full name of the user on whose behalf the report is running.
CRM_UserTimeZone	User's time zone name, for example, Pacific Standard Time.
CRM_UILanguageId	Current locale (LCID) of the user.
CRM_YearStartWeekCode	The first week of the year that's used in Microsoft Dynamics CRM.
CRM_WeekStartDayCode	The first day of the week that is used in CRM.
CRM_FiscalCalendarStart	The start date for the fiscal year that is used in CRM.
CRM_FiscalPeriodType	Specifies how the fiscal year is divided, Quarterly, Monthly, Annually and so on.
CRM_FiscalYearDisplayCode	Specifies whether the fiscal year name is displayed based on when the fiscal year starts or when it ends.
CRM_FiscalYearPeriodConnect	Specifies how the fiscal year and fiscal period are connected when displayed together.
CRM_FiscalYearFormat	Specifies how the name of the fiscal year will be displayed.
CRM_FiscalPeriodFormat	Specifies how the fiscal period will be displayed.
CRM_FiscalYearPrefixFormat	Specifies whether a prefix is attached to the fiscal year when it is displayed.
CRM_FiscalYearSuffixFormat	Specifies whether a suffix is attached to the fiscal year when it is displayed.
CRM_CurrencyDecimalPrecision	The currency decimal precision.
CRM_CurrencySymbol	The organization's currency symbol.
CRM_CurrencyPositivePattern	The format pattern for positive currency values.
CRM_CurrencyNegativePattern	The format pattern for negative currency values.
CRM_NumberDecimalDigits	The number of decimal places to use in numeric values.
CRM_NumberDecimalSeperator	The string that is used as a decimal separator in

Parameter	Description
	numeric values.
CRM_NumberNegativePattern	The format pattern for negative numeric values.
CRM_NumberGroupSizes	The number of digits in each group to the left of the decimal in numeric values.
CRM_NumberGroupSeperator	The string that separates groups of digits to the left of the decimal in numeric values.
CRM_DateSeperator	The string that separates the components of a date, such as year, month, and day.
CRM_TimeSeperator	The string that separates the components of time, such as hour, minutes, and seconds.
CRM_AMDesignator	The string that separates the components of time, such as hour, minutes, and seconds
CRM_PMDesignator	The designator for hours that are "post meridiem" (PM).
CRM_ShortDatePattern	The format pattern for a short date value that is associated with the "d" format pattern.
CRM_LongDatePattern	The format pattern for a long date value that is associated with the "D" format pattern.
CRM_ShortTimePattern	The format pattern for a short time value that is associated with the "t" format pattern.
CRM_MonthDayPattern	The format pattern for month and day values that are associated with the "m" and "M" format patterns.

See Also

Report writing with CRM 2016 for online and on-premises
Working with SQL-based reports (CRM on-premises only)
Publish reports

Working with SQL-based reports (CRM on-premises only)

Applies To: CRM 2016 on-prem

This section contains information about how to modify reports that use Transact-SQL.

In This Section

See Also

© 2016 Microsoft. All rights reserved. Copyright

Modify an existing SQL-based report using SQL Server Data Tools

Applies To: CRM 2016 on-prem

This topic provides information about modifying your existing Report Definition Language (RDL) file using SQL Server Data Tools. However, make sure that your modified RDL file conforms to the RDL schema and specifications. More information:MSDN: Report Definition XML Elements

In this topic

Work with complex SQL queries

When you create or modify a SQL-based report by using the Report Wizard in SQL Server Data Tools, you have to type some complex SQL queries into the Generic Query Designer because of SQL query limitations in Query Builder. Use Query Builder to generate an initial simple SQL query, and then switch to Generic Query Designer to add more complex query logic.

 Note

New or existing SQL queries are limited to 260 table joins. The table join limitation includes your own table joins plus any table joins that are executed within the filtered views that are referred to.

When you add many string concatenations to an SQL query by using Query Designer or Query Builder, SQL Server Data Tools takes more time to refresh report items bound to the query's dataset. This results in reduced user productivity when you edit a report. For improved report writing productivity, you can bypass the report item refresh by manually editing the code for the SQL query in the Report Definition Language (RDL) file.

Modify an RDL file

1. In Microsoft Dynamics CRM, go to **Sales** > **Reports** and then select the report that you want. Click **Edit** on the command bar, and on the **Actions** menu, select **Download Report**.
2. Open SQL Server Data Tools, and create a report server project.
3. In Solution Explorer, right-click the **Reports** folder, select **Add**, and then click **Existing Item**. In the file dialog box, select the RDL file you downloaded in the previous step.
4. To view the XML code of the RDL file, in the Solution Explorer pane, right-click the RDL file, and then click **View Code**. Make the required changes, and save the file.

Add elements by using the Report Designer

1. Perform steps 1 through 3 as specified in Modify an RDL file.
2. Right-click the RDL file, and then click **View Designer**. The report element is displayed on the **Design** tab.
3. Use the **Report Data** pane to add datasets, select table fields, define queries, and add parameters to a report.
4. Save the changes. This adds the required XML code for these report elements in the RDL file.

Test the report

After you finish editing the RDL file, save the changes, and switch back to the report **Preview** tab in SQL Server Data Tools to test the report. Any XML schema errors or SQL errors are reported in SQL Server Data Tools.

When the report is ready, Publish reports.

See Also

Report writing with CRM 2016 for online and on-premises
Use SQL and filtered views to retrieve data for reports
Create a new report using SQL Server Data Tools

Use SQL and filtered views to retrieve data for reports

Applies To: CRM 2016 on-prem

Microsoft Dynamics CRM data and metadata are stored in a Microsoft SQL Server database named *<organization_name>*_MSCRM on the server that is running Microsoft SQL Server in the Microsoft Dynamics CRM (on-premises) deployment. SQL-based reports in Microsoft Dynamics CRM use the filtered views provided for each entity to retrieve data for the reports. Filtered views are fully compliant with the Microsoft Dynamics CRM security model. When you run a report that obtains data from filtered views, the Microsoft Dynamics CRM security role determines what data you can view in the report. Data in filtered views is restricted at these levels: the organization, the business unit, the owner, and at the field level.

Filtered views exist for all Microsoft Dynamics CRM entities, including custom entities. Your custom SQL-based reports cannot read data directly from the Microsoft Dynamics CRM database tables. Instead, you must use the filtered views to retrieve data for your custom SQL-based reports.

The following SQL query returns all columns from the filtered view for the **Account** entity:

```
SELECT * FROM dbo.FilteredAccount
```

Filtered views also provide a way to pull Microsoft Dynamics CRM report data into Microsoft Office applications, such as Microsoft Office Excel and Microsoft Access. For a complete listing of all the standard filtered views organized by product area, see Filtered views in Microsoft Dynamics CRM.

In this topic

Custom and customized entities
Entity schemas for creating custom SQL-based reports
Naming conventions in the Microsoft Dynamics CRM database

Custom and customized entities

When you create a new custom entity in the Microsoft Dynamics CRM database, a new filtered view for the entity is automatically created. Further, if you add or change an attribute in a custom entity or customizable system entity, the change is automatically included in the associated filtered view.

Entity schemas for creating custom SQL-based reports

To find schema information about any filtered view, entity, or attribute in the Microsoft Dynamics CRM database, use one of the following methods:

* In Microsoft Visual Studio, use **SQL Server Object Explorer** to connect to the SQL Server where the Microsoft Dynamics CRM organization database is located. Expand the **Databases** node and then expand the *<organization_name>*_MSCRM database. The filtered views, such as

FilteredContact, can be accessed under the **Views** node. Right-clicking the filtered view displays a shortcut menu that enables you to explore the design of the filtered view and the data it returns.

- Log on to Microsoft Dynamics CRM Web application by using an account that has the System Administrator security role. In Microsoft Dynamics CRM, choose **Settings**, and then select **Customizations**. Next, choose **Customize the System**, expand **Entities**, and double-click an entity name to view its fields (attributes) and relationships.

  - Choose **Fields** to show all the attributes that include the display name and a description for each attribute. To see the dependencies for the attributes, select an attribute, choose **More Actions**, and then select **Show Dependencies**.

  - Choose **1:N Relationships**, **N:1 Relationships**, and **N:N Relationships** to show the entities that have a relationship with the current entity, and the attributes that are used to define the relationships.

- Use Microsoft SQL Server Management Studio to view the database contents directly.

All the methods that are listed here let you access schema information for custom or customized entities and attributes.

Naming conventions in the Microsoft Dynamics CRM database

The following are the attribute naming conventions in the Microsoft Dynamics CRM database.

Attributes obtained through filtered views

Although field names in Microsoft Dynamics CRM are case-sensitive and in mixed case, the attribute names obtained through filtered views are in lowercase.

Drop-down list attributes

All drop-down lists (option sets) have two associated fields for every string in the list. For each string, there is a value (code) field and a label (name) field, such as, **leadsource** and **leadsourcename**. For example, the filtered view for Leads returns two fields related to the **LeadSource** attribute of type **Picklist**: LeadSource = 1 and LeadSourceName ="Advertisement". Reports display the label field and use the value field for numeric comparisons.

DateTime attributes

The **DateTime** attributes are represented by two fields in the filtered view: DateTime and UTC DateTime. The first field contains the date and time value for the appropriate time zone and the second field contains the date and time value in Coordinated Universal Time (UTC).

Entity field

For an entity table in the database, the primary key field is in the name format `EntityId`, for example, `AccountId`. Each `EntityId` field has an associated field that contains the value that should be displayed in reports. For example, for the account entity, it is the `Name` field that contains the name of the account.

In this section

[Filtered views in Microsoft Dynamics CRM](#)

See Also

[Working with SQL-based reports (CRM on-premises only)](#)
[Format report content](#)

Filtered views in Microsoft Dynamics CRM

Applies To: CRM 2016 on-prem

This topic lists many of the filtered views available in on-premises Microsoft Dynamics CRM. You can use these filtered views to securely retrieve data for your custom SQL-based reports, and display the reports to a user based on their security role in Microsoft Dynamics CRM.

Entity schema name	Entity type code	Report view name	Description
Account	1	FilteredAccount	Business that represents a customer or potential customer. The company that is billed in business transactions.
AccountLeads	16	FilteredAccountLeads	Represents the intersect table for the accountleads_associationrelationship.
ActivityMimeAttachment	1001	FilteredActivityMimeAttachment	Multipurpose Internet Mail Extensions (MIME) attachment for an email activity.
ActivityParty	135	FilteredActivityParty	Person or group associated with an activity. An activity can have multiple activity parties.
ActivityPointer	42	FilteredActivityPointer	Task performed, or to be

Entity schema name	Entity type code	Report view name	Description
	00		performed, by a user. An activity is any action for which an entry can be made on a calendar. This view rolls up all the different activities into one view.
Annotation	5	FilteredAnnotation	Annotation (note) that is attached to one or more objects, including other notes.
AnnualFiscalCalendar	2000	FilteredAnnualFiscalCalendar	Year long fiscal calendar of an organization. A span of time during which the financial activities of an organization are calculated.
ApplicationFile	4707	FilteredApplicationFile	For internal use only.
Appointment	4201	FilteredAppointment	Commitment representing a time interval with start/end times and duration.
AsyncOperation	4700	FilteredAsyncOperation	Process whose execution can proceed independently or in the background.
Attachment	1002	FilteredAttachment	For internal use only.
BulkDeleteOperation	4424	FilteredBulkDeleteOperation	User-submitted bulk deletion job.
BulkOperation	4406	FilteredBulkOperation	System operation used to perform lengthy and asynchronous operations on large data sets, such as distributing a campaign activity or quick campaign.
BulkOperationLog	4405	FilteredBulkOperationLog	Log used to track bulk operation execution, successes, and failures.
BusinessUnit	10	FilteredBusinessUnit	Business, division, or department in the Microsoft Dynamics CRM database.
BusinessUnitNewsArticle	13	FilteredBusinessUnitNewsArticle	Announcement associated with

Entity schema name	Entity type code	Report view name	Description
	2		an organization.
Calendar	40 03	FilteredCalendar	Calendar used by the scheduling system to define when an appointment or activity is to occur.
CalendarRule	40 04	FilteredCalendarRule	For internal use only.
Campaign	44 00	FilteredCampaign	Container for campaign activities and responses, sales literature, products, and lists to create, plan, execute, and track the results of a specific marketing campaign through its life.
CampaignActivity	44 02	FilteredCampaignActivity	Task performed, or to be performed, by a user for planning or running a campaign.
CampaignActivityItem	44 04	FilteredCampaignActivityItem	Represents the intersect table for the following relationships: • campaignactivitysalesliterature_association • campaignactivitylist_association
CampaignItem	44 03	FilteredCampaignItem	Represents the intersect table for the following relationships: • campaignproduct_association • campaigncampaign_association • campaignsalesliterature_association • campaignlist_association
CampaignResponse	44 01	FilteredCampaignResponse	Response from an existing or a potential new customer for a campaign.
Competitor	12	FilteredCompetitor	Business competing for the sale

Entity schema name	Entity type code	Report view name	Description
	3		represented by a lead or opportunity.
CompetitorAddress	1004	FilteredCompetitorAddress	For internal use only.
CompetitorProduct	1006	FilteredCompetitorProduct	Represents the intersect table for the competitorproduct_association relationship.
CompetitorSalesLiterature	26	FilteredCompetitorSalesLiterature	Represents the intersect table for the competitorsalesliterature_associationrelationship.
Connection	3234	FilteredConnection	Relationship between two entities.
ConnectionRole	3231	FilteredConnectionRole	Role describing a relationship between a two records.
ConnectionRoleAssociation	3232	FilteredConnectionRoleAssociation	Represents the intersect table for the connectionroleassociation_association relationship.
ConstraintBasedGroup	4007	FilteredConstraintBasedGroup	Group or collection of people, equipment, and/or facilities that can be scheduled.
Contact	2	FilteredContact	Person with whom a business unit has a relationship, such as customer, supplier, or colleague.
ContactInvoices	17	FilteredContactInvoices	Represents the intersect table for the contactinvoices_association relationship.
ContactLeads	22	FilteredContactLeads	Represents the intersect table for the contactleads_associationrelationship.
ContactOrders	19	FilteredContactOrders	Represents the intersect table for the contactorders_associationrelationship.

Entity schema name	Entity type code	Report view name	Description
ContactQuotes	18	FilteredContactQuotes	Represents the intersect table for the contactquotes_associationrelationship.
Contract	10 10	FilteredContract	Agreement to provide customer service during a specified amount of time or number of cases.
ContractDetail	10 11	FilteredContractDetail	Line item in a contract that specifies the type of service a customer is entitled to.
ContractTemplate	20 11	FilteredContractTemplate	Template for a contract containing the standard attributes of a contract.
CustomerAddress	10 71	FilteredCustomerAddress	Address and shipping information. Used to store additional addresses for an account or contact.
CustomerOpportunityRole	45 03	FilteredCustomerOpportunityRole	**Deprecated.** Relationship between an account or contact and an opportunity.
CustomerRelationship	45 02	FilteredCustomerRelationship	**Deprecated.** Relationship between a customer and a partner in which either can be an account or contact.
Discount	10 13	FilteredDiscount	Price reduction made from the list price of a product or service based on the quantity purchased.
DiscountType	10 80	FilteredDiscountType	Type of discount specified as either a percentage or an amount.
DuplicateRule	44 14	FilteredDuplicateRule	Rule used to identify potential duplicates.
Email	42 02	FilteredEmail	Activity that is delivered by using email protocols.
Equipment	40 00	FilteredEquipment	Resource that can be scheduled.

Entity schema name	Entity type code	Report view name	Description
Fax	4204	FilteredFax	Activity that tracks call outcome and number of pages for a fax and optionally stores an electronic copy of the document.
FieldSecurityProfile	1200	FilteredFieldSecurityProfile	Profile which defines access level for secured attributes
FixedMonthlyFiscalCalendar	2004	FilteredFixedMonthlyFiscalCalendar	Fixed monthly fiscal calendar of an organization. A span of time during which the financial activities of an organization are calculated.
Goal	9600	FilteredGoal	Target objective for a user or a team for a specified time period.
GoalRollupQuery	9602	FilteredGoalRollupQuery	Query that is used to filter the results of the goal rollup.
Import	4410	FilteredImport	Status and ownership information for an import job.
ImportFile	4412	FilteredImportFile	File name of file used for import.
ImportLog	4423	FilteredImportLog	Failure reason and other detailed information for a record that failed to import.
ImportMap	4411	FilteredImportMap	Data map used in import.
Incident	112	FilteredIncident	Service request case associated with a contract.
IncidentResolution	4206	FilteredIncidentResolution	Special type of activity that includes description of the resolution, billing status, and the duration of the case.
IntegrationStatus	3000	FilteredIntegrationStatus	For internal use only.
InternalAddress	1003	FilteredInternalAddress	For internal use only.
Invoice	1090	FilteredInvoice	Order that has been billed.

Entity schema name	Entity type code	Report view name	Description
InvoiceDetail	1091	FilteredInvoiceDetail	Line item in an invoice containing detailed billing information for a product.
KbArticle	127	FilteredKbArticle	Structured content that is part of the knowledge base.
KbArticleComment	1082	FilteredKbArticleComment	Comment on a knowledge base article.
KbArticleTemplate	1016	FilteredKbArticleTemplate	Template for a knowledge base article that contains the standard attributes of an article.
Lead	4	FilteredLead	Prospect or potential sales opportunity. Leads are converted into accounts, contacts, or opportunities when they are qualified. Otherwise, they are deleted or archived.
LeadAddress	1017	FilteredLeadAddress	For internal use only.
LeadCompetitors	24	FilteredLeadCompetitors	Represents the intersect table for the leadcompetitors_associationrelationship.
LeadProduct	27	FilteredLeadProduct	Represents the intersect table for the leadproduct_associationrelationship.
Letter	4207	FilteredLetter	Activity that tracks the delivery of a letter. The activity can contain the electronic copy of the letter.
List	4300	FilteredList	Group of existing or potential customers created for a marketing campaign or other sales purposes.
ListMember	4301	FilteredListMember	Represents the intersect table for the following relationships: • listaccount_association

Entity schema name	Entity type code	Report view name	Description
			• listcontact_association • listlead_association
MailMergeTemplate	9106	FilteredMailMergeTemplate	Template for a mail merge document that contains the standard attributes of that document.
Metric	9603	FilteredMetric	Type of measurement for a goal, such as money amount or count.
MonthlyFiscalCalendar	2003	FilteredMonthlyFiscalCalendar	Monthly fiscal calendar of an organization. A span of time during which the financial activities of an organization are calculated.
Opportunity	3	FilteredOpportunity	Potential revenue-generating event, or sale to an account, which needs to be tracked through a sales process to completion.
OpportunityClose	4208	FilteredOpportunityClose	Activity that is created automatically when an opportunity is closed, containing information such as the description of the closing and actual revenue.
OpportunityCompetitors	25	FilteredOpportunityCompetitors	Represents the intersect table for the opportunitycompetitors_associationrelationship.
OpportunityProduct	1083	FilteredOpportunityProduct	Association between an opportunity and a product.
OrderClose	4209	FilteredOrderClose	Activity generated automatically when an order is closed.
Organization	1019	FilteredOrganization	Top level of the Microsoft Dynamics CRM business hierarchy. The organization can be a specific business, holding company, or corporation.

Entity schema name	Entity type code	Report view name	Description
			This view contains the format that is used in Microsoft Dynamics CRM for currency, date and time format, number format, negative number handling, pricing decimal precision, and start day of the week. It also defines the fiscal period for the organization.
PhoneCall	4210	FilteredPhoneCall	Activity to track a telephone call.
PluginAssembly	4605	FilteredPluginAssembly	Assembly that contains one or more plug-in types.
PluginType	4602	FilteredPluginType	Type that inherits from the **IPlugin** interface and is contained within a plug-in assembly.
PluginTypeStatistic	4603	FilteredPluginTypeStatistic	Plug-in type statistic.
PriceLevel	1022	FilteredPriceLevel	Entity that defines pricing levels.
Privilege	1023	FilteredPrivilege	Permission to perform an action in Microsoft Dynamics CRM. Microsoft Dynamics CRM checks for the privilege and rejects the attempt if the user does not hold the privilege.
ProcessSession	4710	FilteredProcessSession	Information that is generated when a dialog is run. Every time that you run a dialog, a dialog session is created.
Product	1024	FilteredProduct	Information about products and their pricing information.
ProductAssociation	1025	FilteredProductAssociation	Represents the intersect table for the productassociation_associationrelationship.
ProductPriceLevel	1026	FilteredProductPriceLevel	Information about how to price a product in the specified price

Entity schema name	Entity type code	Report view name	Description
			level, including pricing method, rounding option, and discount type based on a specified product unit.
ProductSalesLiterature	21	FilteredProductSalesLiterature	Represents the intersect table for the productsalesliterature_associationrelationship.
ProductSubstitute	1028	FilteredProductSubstitute	Represents the intersect table for the productsubstitute_associationrelationship.
Publisher	7101	FilteredPublisher	A publisher of a Microsoft Dynamics CRM solution.
PublisherAddress	7102	FilteredPublisherAddress	Address and shipping information. Used to store additional addresses for a publisher.
QuarterlyFiscalCalendar	2002	FilteredQuarterlyFiscalCalendar	Quarterly fiscal calendar of an organization. A span of time during which the financial activities of an organization are calculated.
Queue	2020	FilteredQueue	A list of records requiring action, such as accounts, cases, and activities.
QueueItem	2029	FilteredQueueItem	A specific item in a queue, such as a case record or an activity record.
Quote	1084	FilteredQuote	Formal offer for products and/or services, proposed at specific prices and related payment terms, which is sent to a prospective customer.
QuoteClose	4211	FilteredQuoteClose	Activity generated when a quote is closed.
QuoteDetail	1085	FilteredQuoteDetail	Product line item in a quote. The details include such information as product ID, description,

Entity schema name	Entity type code	Report view name	Description
			quantity, and cost.
RecurringAppointmentMaster	4251	FilteredRecurringAppointmentMaster	The master appointment of a recurring appointment series.
RelationshipRole	4500	FilteredRelationshipRole	**Deprecated.** Relationship between an account or contact and an opportunity.
RelationshipRoleMap	4501	FilteredRelationshipRoleMap	**Deprecated.** Mapping of the primary associated objects between which the relationship role is valid.
Report	9100	FilteredReport	Data summary in an easy-to-read layout.
ReportCategory	9102	FilteredReportCategory	Categories related to a report. A report can be related to multiple categories.
ReportEntity	9101	FilteredReportEntity	Entities related to a report. A report can be related to multiple entities.
ReportLink	9104	FilteredReportLink	Links and dependencies between reports. A report may drill through to another report, or it may have another report as a sub-report.
ReportVisibility	9103	FilteredReportVisibility	Area in which to show a report. A report can be shown in multiple areas.
Resource	4002	FilteredResource	User or facility/equipment that can be scheduled for a service.
ResourceGroup	4005	FilteredResourceGroup	Resource group or team whose members can be scheduled for a service.
ResourceSpec	4006	FilteredResourceSpec	Selection rule that allows the scheduling engine to select a number of resources from a pool of resources. The rules can be associated with a service.
RibbonCommand	11	FilteredRibbonCommand	For internal use only.

Entity schema name	Entity type code	Report view name	Description
	16		
RibbonContextGroup	11 15	FilteredRibbonContextGroup	For internal use only.
RibbonDiff	11 30	FilteredRibbonDiff	For internal use only.
RibbonRule	11 17	FilteredRibbonRule	For internal use only.
RibbonTabToCommandMa p	11 13	FilteredRibbonTabToCommand Map	For internal use only.
Role	10 36	FilteredRole	Grouping of security privileges. Users are assigned roles that authorize their access to Microsoft Dynamics CRM.
RollupField	96 04	FilteredRollupFieldItem	Field to be rolled up to calculate the actual and in-progress values against the goal.
SalesLiterature	10 38	FilteredSalesLiterature	Storage of sales literature, which may contain one or more documents.
SalesLiteratureItem	10 70	FilteredSalesLiteratureItem	Item in the sales literature collection.
SalesOrder	10 88	FilteredSalesOrder	Quote that has been accepted.
SalesOrderDetail	10 89	FilteredSalesOrderDetail	Line item in a sales order.
SalesProcessInstance	32	FilteredSalesProcessInstance	For internal use only.
SavedQuery	10 39	FilteredSavedQuery	Saved query against the database.
SdkMessage	46 06	FilteredSdkMessage	Message that is supported by the Microsoft Dynamics CRM SDK.
SdkMessageFilter	46 07	FilteredSdkMessageFilter	Filter that defines which Microsoft Dynamics CRM SDK messages are valid for each type of entity.
SdkMessagePair	46	FilteredSdkMessagePair	For internal use only.

Entity schema name	Entity type code	Report view name	Description
	13		
SdkMessageProcessingStep	4608	FilteredSdkMessageProcessingStep	Stage in the execution pipeline that a plug-in is to execute.
SdkMessageProcessingStepImage	4615	FilteredSdkMessageProcessingStepImages	For internal use only.
SdkMessageProcessingStepSecureConfig	4616	FilteredSdkMessageProcessingStepSecureConfig	For internal use only.
SdkMessageRequest	4609	FilteredSdkMessageRequest	For internal use only.
SdkMessageRequestField	4614	FilteredSdkMessageRequestField	For internal use only.
SdkMessageResponse	4610	FilteredSdkMessageResponse	For internal use only.
SdkMessageResponseField	4611	FilteredSdkMessageResponseField	For internal use only.
SemiAnnualFiscalCalendar	2001	FilteredSemiAnnualFiscalCalendar	Calendar representing the semi-annual span of time during which the financial activities of an organization are calculated.
Service	4001	FilteredService	Activity that represents work done to satisfy a customer's need.
ServiceAppointment	4214	FilteredServiceAppointment	Activity offered by the organization to satisfy its customer's needs. Each service activity includes date, time, duration, and required resources.
ServiceContractContacts	20	FilteredServiceContractContacts	Represents the intersect table for the servicecontractcontacts_associationrelationship.
ServiceEndpoint	4618	FilteredServiceEndpoint	Service endpoint that can be contacted.
SharePointDocumentLocation	9508	FilteredSharePointDocumentLocation	Document libraries or folders on a computer that is running SharePoint Server from where

Entity schema name	Entity type code	Report view name	Description
			documents can be managed in Microsoft Dynamics CRM.
SharePointSite	9502	FilteredSharePointSite	SharePoint Server site from where documents can be managed in Microsoft Dynamics CRM.
Site	4009	FilteredSite	Location or branch office where an organization does business. An organization can have multiple sites.
Solution	7100	FilteredSolution	A solution that contains Microsoft Dynamics CRM customizations.
SolutionComponent	7103	FilteredSolutionComponent	A component of a Microsoft Dynamics CRM solution.
StatusMap	1075	FilteredStatusMap	Contains valid status reasons, and the default status reason. Only certain status reasons are valid for each status value.
StringMap	1043	FilteredStringMap	Contains valid numeric values for items in a drop-down list, equivalent labels for those values, and the display order for items in the list.
Subject	129	FilteredSubject	Information regarding subjects available in Microsoft Dynamics CRM.
SystemForm	1030	FilteredSystemForm	Organization-owned entity customizations, including form layout and dashboards.
SystemUser	8	FilteredSystemUser	Person with access to Microsoft Dynamics CRM and who can own records.
SystemUserProfiles	1202	FilteredSystemUserProfiles	Represents the intersect table for the systemuserprofiles_associationrelationship.
SystemUserRoles	15	FilteredSystemUserRoles	Represents the intersect table

Entity schema name	Entity type code	Report view name	Description
			for the systemuserroles_associationrelationship.
Task	4212	FilteredTask	Generic activity representing work to be done.
Team	9	FilteredTeam	Collection of system users that routinely collaborate. Teams can be used to simplify record sharing and provide team members with common access to organization data when team members belong to different business units.
TeamMembership	23	FilteredTeamMembership	Represents the intersect table for the teammembership_associationrelationship.
TeamProfiles	1203	FilteredTeamProfiles	Represents the intersect table for the teamprofiles_associationrelationship.
TeamRoles	40	FilteredTeamRoles	Represents the intersect table for the teamroles_associationrelationship.
Template	2010	FilteredTemplate	Template for an email message that contains the standard attributes of an email message.
Territory	2013	FilteredTerritory	Territory represents sales regions.
TransactionCurrency	9105	FilteredTransactionCurrency	Currency in which a financial transaction is carried out.
UnresolvedAddress	2012	FilteredUnresolvedAddress	For internal use only.
UoM	1055	FilteredUoM	Unit of measure.
UoMSchedule	1056	FilteredUoMSchedule	Grouping of units.

Entity schema name	Entity type code	Report view name	Description
UserFiscalCalendar	1086	FilteredUserFiscalCalendar	Custom fiscal calendar used for tracking sales quotas.
UserQuery	4230	FilteredUserQuery	Saved database query that is owned by a user.
UserSettings	150	FilteredUserSettings	User's preferred settings.
WebResource	9333	FilteredWebResource	Data equivalent to files used in web development. Web resources provide client-side components that are used to provide custom user interface elements.
Workflow	4703	FilteredWorkflow	Set of logical rules that define the steps necessary to automate a specific business process, task, or set of actions to be performed.
WorkflowDependency	4704	FilteredWorkflowDependency	**Deprecated.** Dependencies for a process.
WorkflowLog	4706	FilteredWorkflowLog	Log used to track process execution.

See Also

Report writing with CRM 2016 for online and on-premises
Use SQL and filtered views to retrieve data for reports

Test and troubleshoot reports

Applies To: CRM 2016 on-prem, CRM Online

After you have created a report, test it to see if it produces the results that you want.

Test a report

1. Test the report in Microsoft Visual Studio within the **Preview** tab.
2. If any errors are reported, correct the cause of the errors, and run the test again.
3. When the report functions correctly, publish the report to the report server. To do this, in Microsoft Dynamics CRM go to **Sales** > **Reports**. Click **New** and then fill out the requested information.
4. Run the published report from Microsoft Dynamics CRM to verify the report's operation.

Suggestions for testing a report

The following list of suggestions is provided as a guide for testing your reports:

- **SQL-based reports only**: Verify that your reports access Microsoft Dynamics CRM data through filtered views to follow CRM security restrictions.
- **SQL-based reports only**: Check the number of SQL table joins. Reports might not run if there are too many table joins. After you upload a report, open the pre-filter section of your report and add some related entities, for example, accounts that have associated opportunities. Run the report with the related entities. If a max SQL joins error occurs, you may want to either simplify the report or remove some **Advanced Find** filterable entities.
- Verify that report filters are specified for the correct entities. After uploading the report, open it and check whether any entities designated for **Advanced Find** filtering should be exposed for report pre-filtering.
- If the report queries a custom entity, validate that the entity can be correctly filtered. Validate that the report returns data based on user roles with user-level security on entities, custom security roles, and other roles.
- Some reports are context-sensitive and can be run against selected records in a list. For those types of reports, verify that the report can be run against system views, custom system views, user queries, and selected records.
- Verify that you can take snapshots of the report through the Scheduling Wizard in CRM.
- Verify that you can save the report to a PDF file. Reports frequently print better with PDF formatting.
- Verify that detailed sub-reports are hidden when you publish the report, so they're not run directly by users.

Report drill-through fails in Visual Studio Report Viewer

Some parameters in the reports that are generated by the Report Wizard are marked internal (read-only). Therefore, when you upload the reports into Visual Studio, and try to drill through, the Visual Studio Report Viewer doesn't display the reports. To correct this, use Visual Studio Report Designer to change the parameters from internal to hidden. Clear the **Internal** check box, and make sure that the **Hidden** check box is selected for each of the following parameters:

- CRM_Drillthrough
- CRM_DrillthroughFilterBy
- CRM_FilterText
- CRM_URL

The parameters that are prefixed with CRM_Filtered, such as CRM_FilteredAccount or CRM_FilteredContact, also have to be marked as hidden and not internal.

📝 Note

Your report may not contain all the parameters listed here.

See Also

Report writing with CRM 2016 for online and on-premises
Publish reports
Use parameters in reports
Create a new report using SQL Server Data Tools

Publish reports

Applies To: CRM 2016 on-prem, CRM Online

To make reports available to users, publish the reports in Microsoft Dynamics CRM.

Publish a report in Microsoft Dynamics CRM by using the web application

1. Create a new report or modify a report by using SQL Server Data Tools. More information: Create a new report using SQL Server Data Tools

2. Sign in to Microsoft Dynamics CRM, and go to **Sales** > **Reports**.

3. On the command bar, click **New**.

4. In the **Report:New Report** dialog box, in the **Report Type** list, select **Existing File**, click **Browse**, and then specify the location of the .rdl file in the **File Location** box.

5. Emter appropriate data in the **Details** section of the **Report:New Report** dialog box.

6. Click **Save and Close**.

    You can test the report by selecting it from the Available Reports list. To add the report to context-sensitive lists and forms in Microsoft Dynamics CRM, follow the steps in Determine where the report will appear.

💡 **Tip**

To update the report or to update the information that is displayed in Microsoft Dynamics CRM, such as the description, in the Available Reports list, select the report, and then click **Edit** on the command bar.

Determine where the report will appear

1. Select the report in the Reports view, and then click **Edit** on the command bar.

2. In the **Report** dialog box, specify values in the **Categorization** section according to the following descriptions:

    -   **Categories**: Categorizes the report by its intended purpose. For example, a sales report can be included in context-sensitive lists and forms in the Sales area of Microsoft Dynamics CRM.

    -   **Related Record Types**: Associates the report to specific entities, such as accounts or contacts.

    -   **Display In**: Allows the report to be displayed in context-sensitive forms and lists. You can restrict where the report is displayed. If you intend to publish a hidden report, clear the **Display In** check box.

3. Click **Save and Close**.

By default, a report is uploaded as a user-owned report. To make the report visible to the whole organization, you can do either of the following when editing a report:

-   In the **Report** dialog box, click the **Administration** tab, and then in the **Viewable By** field, select **Individual**.

-   In the **Report** dialog box, on the **Actions** menu, select **Publish Report for External Use**.

Specify a default filter for the report

You can define a default filter for a report so that the filter criteria are used every time the report is run.

1. Select a report in the Reports grid, and on the More Actions (…) menu, select **Edit Default Filter**.
2. In the **Report Viewer** dialog box, define the default filter criteria that will be used every time the report runs, and then select **Save Default Filter**.

See Also

Report writing with CRM 2016 for online and on-premises
Report considerations and best practices

© 2016 Microsoft. All rights reserved. Copyright

Report considerations and best practices

Applies To: CRM 2016 on-prem, CRM Online

Make sure that your reports execute reliably and efficiently by understanding the best practices and considerations described here.

In This Section

Best practices for reports
Improve performance of reports
Improve report performance by using filters
Microsoft Dynamics CRM Online reporting considerations
RDL sandboxing for Microsoft Dynamics CRM Online

See Also

Publish reports
Example reports

© 2016 Microsoft. All rights reserved. Copyright

Best practices for reports

Applies To: CRM 2016 on-prem, CRM Online

The following best practices can help you produce reports that are easier to write, comply with established standards, and execute with improved efficiency.

General best practices

This section provides best practices for creating custom Fetch-based and SQL-based reports.

Use an existing report to create custom reports

Check whether there is an existing report in Microsoft Dynamics CRM that is close to the design you are looking for. Download the report definition for that report, and then modify the RDL file instead of creating a new one from scratch. By doing this, you will save development time and reduce report writing errors.

Use Microsoft Dynamics CRM formats for currency, number, date and time, calendar

The **FilteredUserSettings** view contains information about currency format, date and time format, number format, negative number, starting day of the week, calendars, and other formats. Microsoft Dynamics CRM provides the **fn_GetFormatStrings** database function to get the date, time, number, currency, and calendar formats from the **FilteredUserSettings** view.

Use these resources to correctly format data values in your reports.

Set the page size

Reporting Services does not provide explicit page orientation, such as portrait and landscape modes, or preset page sizes. Standard Microsoft Dynamics CRM reports were designed for 8.25 x 11 (portrait) or 11 x 8.25 (landscape) page sizes that work for both US letter and A4 paper.

Back up your reports

Make backup copies of your reports and store them on a computer other than the reporting server.

Define truncation if needed

Text wrapping is the default behavior for a text box report item in Reporting Services. This means that, unless indicated otherwise, all text will wrap at the defined width of any text area and then grow vertically. If truncation is specified, a text box will truncate text at the width of the text box within the specified padding (default is 2pt left and right). Any maximum length truncation beyond this will require custom coding.

The default Microsoft Dynamics CRM reports are set up with tool tips to show static text or values from data fields when the user hovers the mouse pointer over the report item. If you use truncation, consider setting the **ToolTip** property to the field value so that the full text will appear when the user hovers over the truncated text.

Best practices when you create reports that include date and time fields

When you create reports that use date and time fields, be aware of the following:

- To be consistent with the date and time values in Microsoft Dynamics CRM, when you create reports that use a Coordinated Universal Time (UTC)-based field (Time-Zone Independent or Date Only) don't convert the value to a regional time-zone based (User Local) value.

- If you make a change to the date and time behavior of a field in an entity, you may need to regenerate reports that use that date and time for the report to display the field correctly.

- Regenerate Report Wizard reports. Edit the report using the Report Wizard with the same parameters to regenerate the report.

- Regenerate reports based on a managed solution. For managed solutions, instead of editing the report, re-import the solution to regenerate the report.

- Regenerate custom reports created with Microsoft Visual Studio. Update the date and time fields in Visual Studio as necessary and republish the report. More information: Modify an existing SQL-based report using SQL Server Data Tools

For more information about date and time fields, see Referenced topic '73d691c7-344e-4c96-8979-c661c290bf81' is not in the TOC..

SQL-based reports

This section provided best practices for SQL-based reports only.

Writing stored procedures

Adding custom stored procedures to the Microsoft Dynamics CRM database is not supported. However, you can create a separate database and write stored procedures to that database.

Limit text length and number of items in charts

Microsoft Dynamics CRM reports use only some of the possible chart types from Reporting Services. For any chart type, limiting label length and number of items is recommended for the chart contents to be displayed correctly. When Reporting Services displays a chart with long labels, the chart itself becomes too small to be usable. There are several ways to limit text length and items in charts:

- Limit your chart label length explicitly, shortening it if necessary.

- Consider limiting the number of items displayed in charts. For more information, see Example: Limit the number of items displayed in a chart.

Use embedded images in a report

The easiest way to use images with Reporting Services is to put the images into a database. If the images are not in a database, you can use embedded images in .png, .gif, or .jpg formats in a report. The image files that are used by Microsoft Dynamics CRM are located in the C:/Inetpub/wwwroot/_imgs/ico folder on a default Microsoft Dynamics CRM installation.

See Also

Report writing with CRM 2016 for online and on-premises
Improve performance of reports

Improve performance of reports

Applies To: CRM 2016 on-prem, CRM Online

Here are some guidelines that can help you improve the performance of the report.

General

These guidelines are applicable for both Fetch-based and SQL-based reports.

- Limit a report to display information from a specified time period, instead of displaying all records in the Microsoft Dynamics CRM database.

- Pre-filter a report so that the dataset is limited.

- Calculate aggregate totals by using aggregations in a FetchXML query or a SQL statement, instead of passing raw data to Reporting Services and grouping.

- Limit the number of datasets used, if possible.

- When you compare dates, use the UTC date fields for comparisons. For example, compare the **createdonutc** fields and not the **createdon** fields in the FetchXML query or a filtered view.

SQL-based Reports

These guidelines are applicable for SQL-based reports only.

- Don't create a report that uses a large dataset or a complex SQL query available on-demand to all users.

- Don't select all columns from a Microsoft Dynamics CRM filtered view. Instead, explicitly specify the columns that you want to select in the SQL statement.

- Use SQL stored procedures instead of inline SQL.

See Also

Report writing with CRM 2016 for online and on-premises
Improve report performance by using filters

Improve report performance by using filters

Applies To: CRM 2016 on-prem, CRM Online

Reports that return large data sets can be difficult to use and can cause performance problems. To limit the data that is presented in a report, use data filters.

In addition to data filtering supported by Reporting Services, Microsoft Dynamics CRM supports data pre–filtering. You can use data pre-filtering to:

- Make reports context-sensitive by narrowing the scope of a report to return more relevant data.

- Retrieve and display a result set faster because only more relevant data is returned.

- Allow the report to be filtered using the **Advanced Find** feature.

◆ Important

Currently, report queries with hierarchical operators, such as the **Under** operator, can't be used with report filtering. When you try to run a report that uses a hierarchical operator, the report won't render.

In this topic

Enabling data pre-filtering in Fetch-based reports

Fetch-based reports support only automatic data pre-filtering. A report can have multiple data sets and multiple FetchXML queries. One data set supports one FetchXML query. To enable pre-filtering for the primary or linked entity in a Fetch-based report, you must set the value of the **enableprefiltering** parameter to "1", and specify a parameter name in the **prefilterparametername** property. The parameter name should start with "CRM_" to specify it as a hidden parameter. As with the Microsoft SQL Server-based report, this parameter specified in the FetchXML query acts as a sub query within the FetchXML query, and the sub query is constructed with the value specified by the user in the **Advanced Find** area while running a report.

The following example displays how to enable pre-filtering for the primary entity in the FetchXML query.

```
<CommandText

 <fetch distinct="false" mapping="logical">

   <entity name="account" enableprefiltering="1" prefilterparametername="CRM_FilteredAccount">

      <attribute name="name" />

      <attribute name="accountid" />

   </entity>

 </fetch>

</CommandText>

<DataSourceName>CRM</DataSourceName>
```

Similarly, you can enable pre-filtering for the linked entity. You can also specify a different pre-filtering condition for the linked entity in the FetchXML query by specify a different and unique name for the parameter name in the **prefilterparametername** property.

If you are manually modifying a Fetch-based report definition without using the Report Wizard in the Microsoft Dynamics CRM web application or SQL Server Data Tools to enable pre-filtering for primary and linked entities, make sure that you:

1. Similarly, you can enable pre-filtering for the linked entity. You can also specify a different pre-filtering condition for the linked entity in the FetchXML query by specify a different and unique name for the parameter name in the **prefilterparametername** property.

   If you are manually modifying a Fetch-based report definition without using the Report Wizard in the Microsoft Dynamics CRM web application or SQL Server Data Tools to enable pre-filtering for primary and linked entities, make sure that you:

   ```
   <
   fetch
    distinct
   ="false"
    mapping
   ="logical"
   >
       <entity name="account" enableprefiltering="1"
     prefilterparametername="CRM_FilteredAccount">
   ```

2. Create a corresponding query parameter with the same name as specified for the **prefilterparametername** property. Make sure that the parameter name starts with `CRM_` to specify it as a hidden parameter.

   ```
   <
   QueryParameters
   >
       <QueryParameter Name="CRM_FilteredAccount">
       <Value>=Parameters!CRM_FilteredAccount.Value</Value>
     </QueryParameter>
   ```

3. Create a corresponding report parameter with the same name.

   ```
   <
   ReportParameters
   >
    <ReportParameter Name="CRM_FilteredAccount">
      <DataType>String</DataType>
   ```

```
        <Prompt>CRM Filtered Account</Prompt>

    </ReportParameter>

</ReportParameters>
```

Enabling data pre-filtering in SQL-based reports (CRM on-premises only)

There are two ways that you can enable data pre-filtering on Microsoft Dynamics CRM SQL-based reports: automatic and explicit.

Automatic pre-filtering

Automatic data pre-filtering is suited for simple queries. To enable automatic data pre-filtering on a report, you can use aliases for entity tables in queries. You do this by using an alias name that starts with `CRMAF_`.

For example, the following table shows a simple query modified to enable pre-filtering on the **Account** entity.

Query without pre-filtering	Modified query with automatic pre-filtering enabled

`SELECT <column1>, <column2>, <columnN>`
 	
`FROM FilteredAccount;`
	`SELECT <column1>, <column2>, <columnN>`
	
	`FROM FilteredAccount AS CRMAF_FilteredAccount;`

When you enable automatic data pre-filtering functionality by using the `CRMAF_` prefix, Microsoft Dynamics CRM modifies the query to include a parameter (for example, **P1**) when it is uploaded to Microsoft Dynamics CRM, as shown in the following table.

Query with automatic pre-filtering	Modified by Microsoft Dynamics CRM

`SELECT <column1>, <column2>, <columnN>`
 	
`FROM FilteredAccount AS CRMAF_FilteredAccount;`
	`SELECT <column1>, <column2>, <columnN>`
	
	`FROM (@P1) AS CRMAF_FilteredAccount;`

Microsoft Dynamics CRM will pass a query to the **P1** parameter depending on how the report is being filtered. In other words, automatic data pre-filtering acts as a sub-query within the existing query.

The following examples illustrate how Microsoft Dynamics CRM passes queries to the parameter (**P1**) as per different filtering requirements. In these examples, it is assumed that you are running the report from the **Reports** area in Microsoft Dynamics CRM, and are using the data filtering option.

Example 1: If you want to view only active accounts, the resulting query would be as follows:

```
SELECT <column1>, <column2>, <columnN>

FROM (SELECT FilteredAccount.* FROM FilteredAccount WHERE statecode = 0)

AS CRMAF_FilteredAccount
```

Example 2: If you are within a specific account and run the report, the resulting query would be as follows:

```
SELECT <column1>, <column2>, <columnN>

FROM (SELECT FilteredAccount.* FROM FilteredAccount WHERE AccountId = '<CurrentAccountId>')

AS CRMAF_FilteredAccount
```

Example 3: If you are looking at a list of three selected accounts and you choose the option to run the report against the selected records, the resulting query would be as follows:

```
SELECT <column1>, <column2>, <columnN>

FROM  (SELECT FilteredAccount.* FROM FilteredAccount WHERE AccountId in ('<1stAccountId>',

'<2ndAccountId>', '<3rdAccountId>')

AS CRMAF_FilteredAccount
```

When any entity table names are aliased, the Advanced Find user interface is automatically included in the deployed report when it is run from Microsoft Dynamics CRM.

To alias an entity table name in Query Builder, right-click each table in your report, select **Properties**, and then enter the alias value in the form CRMAF_*FilteredEntity*, for example, CRMAF_FilteredAccount.

Limitation of Automatic Pre-filtering

When you use the CRMAF_ prefix to enable automatic pre-filtering, Microsoft Dynamics CRM adds a parameter in the query. With a more complex query, such as a query that uses **UNION** statements, this can lead to unexpected results because Microsoft Dynamics CRM might only add the parameter to the first query.

For example, consider the following query containing **UNION** statements:

```
SELECT <column1>, <column2>, <columnN>

FROM FilteredAccount AS CRMAF_FilteredAccount

WHERE address1_stateorprovince = 'FL'

UNION

SELECT <column1>, <column2>, <columnN>

FROM FilteredAccount AS CRMAF_FilteredAccount

WHERE address1_stateorprovince = 'CA'
```

When you upload the report, Microsoft Dynamics CRM may filter only the first query using the parameter. This leads to the filtering not being applied to the second query:

```
SELECT <column1>, <column2>, <columnN>

FROM  (@P1) AS CRMAF_FilteredAccount WHERE address1_stateorprovince = 'FL'

UNION

SELECT <column1>, <column2>, <columnN>

FROM FilteredAccount AS CRMAF_FilteredAccount

WHERE address1_stateorprovince = 'CA'
```

In the preceding example, while running the report from the **Reports** area in Microsoft Dynamics CRM and choosing the filter as annual revenue greater than 1,000,000, Microsoft Dynamics CRM will pass a query to the **P1** parameter as follows:

```
SELECT <column1>, <column2>, <columnN>

FROM  (SELECT FilteredAccount.* from FilteredAccount where AnnualRevenue > 1000000) AS

CRMAF_FilteredAccount

WHERE address1_stateorprovince = 'FL'

UNION

SELECT <column1>, <column2>, <columnN>

FROM FilteredAccount AS CRMAF_FilteredAccount

WHERE address1_stateorprovince = 'CA'
```

This implies that the query would return only those accounts in Florida with an annual revenue greater than $1,000,000 and all the accounts in California, which is not what you intended. You wanted to view all the accounts in Florida and California with annual revenue greater than $1,000,000.

If you download the report from Microsoft Dynamics CRM and open it in Microsoft Visual Studio, you will see the original version of the report that you uploaded into Microsoft Dynamics CRM. If you download the report directly from Microsoft SQL Server Reporting Services, you will notice that Microsoft Dynamics CRM had modified the query but did not place the parameter where you wanted it to exist.

For complex queries like this, you must use explicit pre-filtering.

Explicit pre-filtering

For complex queries such as queries using **UNION** statements, you may need to use explicit pre-filtering. Unlike automatic pre-filtering, Microsoft Dynamics CRM does not rewrite the report query by passing values to the parameters during explicit pre-filtering when such a report is uploaded to Microsoft Dynamics CRM. You must explicitly make the required changes to the report by adding the pre-filtering parameter to the report, and then referencing the parameter in the query. You can then execute the query by using dynamic SQL.

When you use dynamic SQL, filtering through Advanced Find is enabled by creating a hidden parameter named **CRM_FilteredEntity**, for example, CRM_FilteredAccount, and by using this parameter in a dynamic SQL query expression. This parameter enables filtering on the table data obtained from the specified filtered view.

Taking the same example as discussed earlier to highlight the limitation of automatic pre-filtering, the following table shows a query with automatic pre-filtering modified to use explicit pre-filtering by using dynamic SQL. It is also assumed that while running the report from the **Reports** area in Microsoft Dynamics CRM, the filter has been applied as annual revenue greater than 1,000,000.

Query with automatic pre-filtering	Query modified to use explicit pre-filtering

`SELECT <column1>, <column2>, <columnN>`
 	
`FROM FilteredAccount AS CRMAF_FilteredAccount`
 	
`WHERE address1_stateorprovince = 'FL'`
 	
`UNION`
 	
`SELECT <column1>, <column2>, <columnN>`
 	
`FROM FilteredAccount AS CRMAF_FilteredAccount`
	`DECLARE @SQL nvarchar(4000)`
	
	`DECLARE @CRM_FilteredAccount nvarchar(2000)`
	
	`Set @CRM_FilteredAccount = 'Select`
	
	`FilteredAccount.* FROM FilteredAccount where`
	
	`AnnualRevenue > 1000000'`
	
	`SET @SQL = 'SELECT <column1>, <column2>,`

Query with automatic pre-filtering	Query modified to use explicit pre-filtering

`WHERE address1_stateorprovince = 'CA'`	`<columnN>`
	
	`FROM ('+@CRM_FilteredAccount+') AS FA where`
	
	`address1_stateorprovince = ''FL''`
	
	`UNION`
	
	`SELECT <column1>, <column2>, <columnN>`
	
	`FROM ('+@CRM_FilteredAccount+') as CA where`
	
	`address1_stateorprovince  = ''CA'' '`
	
	`EXEC (@SQL)`

 Note

Most of the standard Microsoft Dynamics CRM SQL-based reports use the explicit pre-filtering option.

Passing filters in the filter summary

A filter summary displays the value of the filter that is used when a report is run. In Microsoft Dynamics CRM reports, it is displayed as a text box report item in the report header that contains the filter text value. When the user runs the report, the Report Viewer displays an **Edit Filter** button. When the button is clicked, it enables the user to define a data filter. An example of a filter summary can be found in the User Summary report that is included with Microsoft Dynamics CRM.

To add a filter summary to a report, follow these steps:

1. Create a hidden string parameter called CRM_FilterText.

2. Add a text box report item to the report and set its **Value** property as follows:

    `=Parameters!CRM_FilterText.Value.`

When the report is run, the value of the CRM_FilterText parameter will be set by the system to the text of the current filter.

Default filters

When you publish a report, you can set a default filter. For all the reports that were created by using the report wizard, if you do not set a default filter, the filter is automatically set to all records of the entity modified within the last 30 days. For the procedure to define a default report filter, see Publish reports.

See Also

Report writing with CRM 2016 for online and on-premises
Microsoft Dynamics CRM Online reporting considerations

Microsoft Dynamics CRM Online reporting considerations

Applies To: CRM Online

Microsoft Dynamics CRM Online has a number of capabilities that allow customers to surface business data that helps them drive decisions and interact with their customers more effectively. Capabilities that are available within CRM Online include views, charts, dashboards, and Microsoft SQL Server Reporting Services reports. Also included is Microsoft Office Excel integration that allows users to easily build self-service reports using the Power BI features PowerView, PowerPivot, and PowerQuery. As the volume of data held in the CRM Online database continues to grow it becomes more important than ever to think about your BI strategy and determine the most effective mechanisms for reporting and visualizing large datasets.

In a CRM Online environment, the reporting infrastructure is shared and separate from the database. In this architecture, although customers share the resources required to run the report, each report runs against the customers' individual database instances. Additionally, with CRM Online, users can run as many reports as they need whenever they want to run them to meet business goals. We do not place time restrictions on reports.

The reporting capabilities built in to CRM Online are designed to let users run reports on datasets that span shorter periods of time. Considering this, Microsoft Dynamics CRM has the following fixed settings:

- Reports and queries can execute for up to five minutes. When the maximum period is reached, the report will time out and a message is returned to the user. Within the five-minute duration, reports and queries are allowed to span large datasets that are beyond 50,000 records, which provides significant flexibility to satisfy most operational reporting needs.

- To improve query response, we recommend that detailed reports minimize the display of large numbers of records. To do this, apply suitable filtering to reduce the number of records that are returned. When you create aggregated or summarized reports, queries should push the aggregation to the query rather than fetch detailed records to perform aggregation in the report. This can be done by using Fetch XML aggregation. More information: Use FetchXML aggregation

- For charts and grids displayed in dashboards, Microsoft Dynamics CRM allows users to run queries that have a dataset that is under 50,000 rows. Should a user run a dashboard query that spans a dataset of 50,000 or more rows, Microsoft Dynamics CRM returns the message "The maximum record limit is exceeded. Reduce the number of records." The dataset practical setting helps to ensure optimal performance of the CRM Online application.

In this topic

Tips and solutions for reporting

Typically, for most organizations' reporting needs, these settings are adequate. To make sure that your users do not exceed these settings and to improve report querying performance in general, consider the following best practices.

- When you create custom reports or dashboards, design them to query smaller datasets over shorter periods of time by adding a time-based filter in the report, such as the current month or quarter, to limit the results.

- We recommend that you limit the number of entities that are needed to return the result. This helps reduce the time required to run the query and return the result set.

- We recommend that you reduce the number of records shown in detailed reports. Suitable filtering can be used to reduce the number of records returned by the query to reduce timeouts.

- For aggregated or summarized reports, queries must be used to push the aggregation to the database and not fetch detailed records and perform aggregation in the Microsoft SQL Server Reporting Services report.

- When appropriate for your business, users should run the default (out-of-the-box) reports and dashboards. These reports and dashboards are typically designed to query per user datasets, so in most cases will not exceed the dataset limit.

If CRM Online users must run reports that exceed these settings, we recommend that you review the following options for assistance with complex reporting needs. Both options effectively offload reporting workloads from CRM Online to another SQL Server datastore by using a CRM data integration solution.

- Third-party Microsoft Dynamics CRM adapters for SSIS are used in conjunction with SQL Server Integration Services (SSIS) to extend the capabilities for integration with CRM Online data.

- Extract transform load ETL tools provide a new tool set for creating analysis of CRM Online data by combining multiple data sources or extracting data to the data warehouse solution if SSIS is not in use. ETL tools provide comprehensive solutions for connecting CRM systems to move data.

 ◆ **Important**

When you use these tools, we recommend you move or synchronize data during nonbusiness hours.

If needed, there are many Microsoft Dynamics Partners who can help provide a solution for your specific reporting needs, such as creating an offline copy of the data specifically used for running large reports. These Partners are knowledgeable with the CRM data integration tools available. More information: Find the right partner

Third-party Microsoft Dynamics CRM adapters for SSIS

- CozyRoc SSIS+ component for Microsoft Dynamics CRM
- KingswaySoft SSIS Integration Toolkit for Microsoft Dynamics CRM
- RSSBus SSIS task for Microsoft Dynamics CRM

- [Team4 SSIS Connector for Microsoft CRM](#)

- [PragmaticWorks TaskFactory SSIS Source/Destination for Dynamics CRM](#)

ETL tools

- [Scribe Insight and Scribe Online with adapter for Microsoft Dynamics CRM](#)

- [Productivity tools from Informatica](#)

See Also

[Microsoft Dynamics CRM 2015 Report Authoring Extension (with SQL Server Data Tools support)](#)
[Developers guide to reports for Microsoft Dynamics CRM](#)
[Introduction to Microsoft Power Query for Excel](#)
[Dynamics CRM OData Feeds and Power Query: What's the [Record]?](#)
[Using Power View in Excel 2013 to Analyze CRM Data](#)
[Report writing with CRM 2016 for online and on-premises](#)

RDL sandboxing for Microsoft Dynamics CRM Online

Applies To: CRM Online

In Microsoft Dynamics CRM Online, the reports run in the sandbox mode. This is done by enabling Report Definition Language (RDL) Sandboxing in Microsoft SQL Server Reporting Services. The RDL Sandboxing lets you detect and restrict the usage of specific types of resources. As a result, certain features in Microsoft Dynamics CRM Online may not be available. For more information, see [MSDN: Enabling and Disabling RDL Sandboxing](#).

The current RDL Sandboxing configuration settings in Microsoft Dynamics CRM Online are described in the following sections in this topic.

In this topic

[Limits of the array result length and string result length](#)

[Allowed types and denied members](#)

[Common denied members](#)

Limits of the array result length and string result length

The maximum number of items allowed in an array return value for an RDL expression is increased from 250 to 102400. The maximum number of items allowed in a string return value for an RDL

expression is also increased from 250 to 102400. This enables you to include images and logos with sizes up to 75 KB, which will be stored in a database with Base64 encoding.

Allowed types and denied members

The RDL Sandboxing feature enables you to create a list of approved types and a list of denied members. The list of approved types is called an allow list. The list of denied members that are not permitted in the RDL expressions is called a block list.

The following table contains a list of allowed types and denied members available in sandbox mode in Microsoft Dynamics CRM Online.

Allowed Types	Denied Members
System.Array	CreateInstance
	Finalize
	GetType
	MemberwiseClone
	Resize
System.DateTime	FromBinary
	GetDateTimeFormats
	GreaterThan
	GreaterThanOrEqual
System.Object	GetType
	MemberwiseClone
	ReferenceEquals
System.DbNull	Finalize
	MemberwiseClone
	GetObjectData
	GetTypeCode
System.Math	BigMul
	DivRem
	IEEERemainder

Allowed Types	Denied Members
	E
	PI
	Pow
System.String	
System.TimeSpan	Hours
	TicksPerDay
	TicksPerHour
	TicksPerMillisecond
	TicksPerMinute
	TicksPerSecond
	Zero
	TryParse
	TryParseExact
System.Convert	ChangeType
	IConvertible.ToBoolean
	IConvertible.ToByte
	IConvertible.ToChar
	IConvertible.ToDateTime
	IConvertible.ToDecimal
	IConvertible.ToDouble
	IConvertible.ToInt16
	IConvertible.ToInt32
	IConvertible.ToInt64
	IConvertible.ToSByte
	IConvertible.ToSingle
	IConvertible.ToType
	IConvertible.ToUInt16
	IConvertible.ToUInt32
	IConvertible.ToUInt64
System.StringComparer	Create
	Finalize

Allowed Types	Denied Members
System.TimeZone	Finalize
	GetType
	MemberwiseClone
System.Uri	Unescape
	Parse
	Escape
	Finalize
System.UriBuilder	Finalize
System.Globalization.CultureInfo	ClearCachedData
System.Text.RegularExpressions.Match	Empty
	NextMatch
	Result
	Synchronized
System.Text.RegularExpressions.Regex	CacheSize
	CompileToAssembly
	GetGroupNames
	GetGroupNumbers
	GetHashCode
	Unescape
	UseOptionC
	UseOptionR
	capnames
	caps
	capsize

Allowed Types	Denied Members
	capslist
	roptions
	pattern
	factory
	IsMatch
	Matches
	Iserializable.GetObjectData
	InitializeReferences
	RightToLeft
	Options
Microsoft.VisualBasic.Constants	vbAbort
	vbAbortRetryIgnore
	vbApplicationModal
	vbArchive
	vbBinaryCompare
	vbCancel
	vbCritical
	vbDefaultButton1
	vbDefaultButton2
	vbDefaultButton3
	vbExclamation
	vbFormFeed
	vbGet
	vbHidden
	vbHide
	vbHiragana
	vbIgnore
	vbInformation
	vbKatakana

Allowed Types	Denied Members
	vbLet
	vbLinguisticCasing
	vbMaximizedFocus
	vbMinimizedFocus
	vbMinimizedNoFocus
	vbMsgBoxHelp
	vbMsgBoxRight
	vbMsgBoxRtlReading
	vbMsgBoxSetForeground
	vbNo
	vbNormal
	vbNormalFocus
	vbNormalNoFocus
	vbObjectError
	vbOK
	vbOKCancel
	vbOKOnly
	vbQuestion
	vbReadOnly
	vbRetry
	vbRetryCancel
	vbSet
	vbSystem
	vbSystemModal
	VbTypeName
	vbVolume
	Zero
Microsoft.VisualBasic.ControlChars	Finalize
	GetType
	MemberwiseClone

Allowed Types	Denied Members
Microsoft.VisualBasic.Conversion	Err
	ErrorToString
	Fix
Microsoft.VisualBasic.DateInterval	Finalize
	GetType
	MemberwiseClone
Microsoft.VisualBasic.Financial	Finalize
	GetType
	MemberwiseClone
	IRR
	NPV
	MIRR
Microsoft.VisualBasic.Interaction	AppActivate
	Beep
	CallByName
	Command
	CreateObject
	Environ
	Finalize
	GetAllSettings
	GetObject
	GetSetting
	GetType
	InputBox
	MemberwiseClone
	MsgBox
	SaveSetting

Allowed Types	Denied Members
	Shell
	Choose
	Switch
Microsoft.VisualBasic.Information	Erl
	Err
	IsError
	IsDBNull
	Lbound
	Ubound
	SystemTypeName
Microsoft.VisualBasic.Strings	Finalize
	GetType
	MemberwiseClone
	Lset
	Rset
Microsoft.Crm.Reporting.RdlHelper	

Common denied members

The following table contains a list of denied members common in allowed types:

DateString
Duration
Equality
Equals
Erl
Filter
GetChar

GroupNameFromNumber
GroupNumberFromName
Int
MaxValue
MinValue
Negate
Timer
TimeString
ToBinary
Finalize
GetType
MemberwiseClone

See Also

Report writing with CRM 2016 for online and on-premises
Example reports

Example reports

Applies To: Dynamics CRM 2016, Dynamics CRM Online

This section provides example reports that show how to implement common techniques in your reports for Microsoft Dynamics CRM.

Create a report using an example

1. Create a report using SQL Data Tools. More information: Report writing environment using SQL Server Data Tools

2. During the report configuration, paste the example into the report and follow any additional report configuration instructions included in the example.

    - For a Fetch XML example, see Create a custom Fetch-based report (CRM Online and CRM on-premises).

    - For a SQL-based example, see Create a custom SQL-based report (CRM on-premises only).

3. Publish the report. More information:

In This Section

See Also

Example: Limit the number of items displayed in a chart

Applies To: Dynamics CRM 2016, Dynamics CRM Online

Chart reports are limited to 15 data points. The dataset must limit the result set to the top 15 rows for the charted value. Additionally, a 16th "other" data point value is optional. This value represents an aggregation of all other rows in the domain. The 16th value is always sorted to the bottom of the result set. This adds a level of complexity to reports that contain drill-through functionality that is appropriate for some reports.

✍ Note

The number of data points and label length may have to be reduced to correctly display the contents of the chart.

Example

The following is a SQL query example that generates the top 15 data points plus a 16th other data point for a chart:

```
INSERT INTO @AcctTopTbl SELECT Top 15 accountid FROM FilteredAccount ORDER BY Revenue Desc


SELECT AcctTop.accountid, AcctTop.name, AcctTop.Revenue, 1 As SortOrder FROM (SELECT Top 15

accountid, name, Revenue FROM FilteredAccount ORDER BY Revenue Desc) AS AcctTop

UNION
```

```
SELECT Null As accountid, 'Other' As name, Sum(Revenue) As Revenue, 2 As SortOrder FROM

FilteredAccount

WHERE accountid NOT IN (Select accountid FROM @AcctTopTbl)

ORDER BY SortOrder Asc, Revenue Desc
```

Example

The following example demonstrates how to limit a number of data points shown in the table by using the FetchXML query. You have to provide two data sets with one FetchXML query per dataset. The results from the first dataset query are passed through the multi-valued parameters to the second dataset to filter the results of the query. Dataset1 retrieves the top 15 records ordered by revenue and Dataset2 retrieves the "TotalRevenue" aggregating all accounts except the accounts from DataSet1.

```
DataSet1:

<fetch count="15">

  <entity name="account" >

    <attribute name="accountid" />

    <attribute name="name" />

    <attribute name="revenue" />

    <order attribute="revenue" descending="true" />

  </entity>

</fetch>


DataSet2:

<fetch aggregate="true">

 <entity name="account">

    <attribute name="revenue" aggregate="sum" alias="TotalRevenue" />

    <filter>

       <condition attribute="accountid" operator="not-in" value="@TopAccountIds"/>

     </filter>

  </entity>

</fetch>


Report Parameter:


<ReportParameter Name=" TopAccountIds ">
```

```
        ...

        <MultiValue>true</MultiValue>

        <Hidden>true</Hidden>

        <DefaultValue>

            <DataSetReference>

                <DataSetName>DataSet1</DataSetName>

                <ValueField>accountid</ValueField>

            </DataSetReference>

        </DefaultValue>

</ReportParameter>
```

See Also

Example: Display the top X values

Applies To: Dynamics CRM 2016, Dynamics CRM Online

Microsoft Dynamics CRM includes several reports that display the top X items of an entity, where the user can specify the value of X.

To query for the top items, use dynamic SQL. Assign the SQL query string that includes the user-specified parameter to a variable. This resolves the user-specified parameter. The query string is then passed to the **SQL EXEC** function together with the user-specified parameter.

Example

In the following SQL example, the TopCount parameter contains the user-supplied value.

```
Declare @SQL nVarchar (4000)

SET @SQL = '

Select  Top ' + CONVERT(nvarchar(10), @TopCount) + ' kb.kbarticleid FROM

(' + @CRM_FilteredKBArticle + ')  kb '

Exec (@SQL)
```

If you're authoring a FetchXML-based report, this is the corresponding FetchXML query.

```
<

fetch

 count

="@TopCount"

>
```

`<entity name="kbarticle" enableprefiltering="true"`

`prefilterparametername="CRM_FilteredKbArticle" >`

`<attribute name="kbarticleid"/>`

`</entity>`

`</fetch>`

See Also

Example: Make a report context-sensitive

Applies To: Dynamics CRM 2016, Dynamics CRM Online

The following steps describe how to set up a context-sensitive report in Microsoft Dynamics CRM.

For example, if a report displays all activities for a particular account and you want to show this report on the Microsoft Dynamics CRM **Account** form, you must include **Activities** and **Accounts** in the **Related Record Types** categorization and specify **Forms for related record types** in the **Display** in field of the Report: New form.

Create and configure a context-sensitive report

1. Create a report that contains fields from a Filtered<entity> database view and establish one or more SQL JOINs with other related filtered views as required.

2. Use CRMAF_Filtered<entity> as an alias name in the report's SQL query statement. Here's an example.

   `SELECT CRMAF_FilteredActivityPointer.activitytypecodename as`

   `activitytypecodename,`

   `CRMAF_FilteredActivityPointer.regardingobjectidname as regardingobjectidname,`

```
CRMAF_FilteredActivityPointer.subject as subject,

CRMAF_FilteredAccount.name

FROM FilteredActivityPointer AS CRMAF_FilteredActivityPointer

INNER JOIN FilteredAccount As CRMAF_FilteredAccount on

CRMAF_FilteredAccount.accountid =

CRMAF_FilteredActivityPointer.regardingobjectid
```

If you have a Fetch-based report, you can use the following FetchXML query instead of the SQL query.

```
<

fetch

>

  <entity name="activitypointer" enableprefiltering="1" >

    <attribute name="activitytypecode" />

    <attribute name="regardingobjectid" />

    <attribute name="subject" alias="subject" />

    <link-entity name="account" from="accountid" to="regardingobjectid" link-
type='inner' alias="accountLink">

      <attribute name="name" alias="name" />

    </link-entity>

  </entity>

</fetch>
```

3. When you upload the report to Microsoft Dynamics CRM through the **Report: New** form, select all entities in the **Related Record Types** categorization that filtered views are referred to in the report's SQL code.

4. In the **Display In** field, select **Forms for related record types** or **Lists for related record types**. Selecting Lists for related record types lets you run a report from the entity list grid. Selecting **Forms for related record types** lets you run a report from the entity form.

See Also

Example reports
Copy reports between Microsoft Dynamics CRM on-premises deployments
Report writing with CRM 2016 for online and on-premises

Copy reports between Microsoft Dynamics CRM on-premises deployments

Applies To: CRM 2016 on-prem

The preferred method to copy a report between organizations or deployments is to include the report and any custom entities the report uses in a solution. If you do this, the entity types are mapped automatically by the system. If you choose to manually copy a report to another Microsoft Dynamics CRM deployment, you must change the entity type codes for custom entities referred to in the report.

In this topic

[Fix the type code for a custom entity used in a report](#)
[Copy a report between deployments](#)

Fix the type code for a custom entity used in a report

The type code for a custom entity may be different in different Microsoft Dynamics CRM installations. If your report contains references to a custom entity type code, such as when you perform a drill-through to a custom Microsoft Dynamics CRM entity, you must change the type code of the custom entity referred to in your report to the actual type code of the custom entity on the other system you are deploying the report to.

The entity type code for all default (non-custom) entities is predefined.

```
&etc=<entity_type_code>.
```

The recommended method to refer to the type code is to define a hidden parameter in the report and set its value to the custom entity type code on the original system. After deployment to another Microsoft Dynamics CRM installation, use a SQL statement to set the parameter's value to the type code of the custom entity on that system.

To use a CRM_OTC parameter instead of a hard-coded type code in a drill-through link to Microsoft Dynamics CRM

1. Obtain the custom entity's type code value. To do this, run the following SQL query on the Microsoft Dynamics CRM organization database. Substitute an appropriate organization_MSCRM name. Locate the type code of your custom entity in the resultset.

    ```
    use <organization>_MSCRM select Name,ObjectTypeCode from dbo.Entity order by
    ObjectTypeCode
    ```

2. In the report, create two report parameters: CRM_URL and CRM_OTC. CRM_OTC should be of type string. The default CRM_OTC value should be set to the value obtained in step 1.

3. On the report item that should be used for a drill-through link, add the following Value code:

    ```
    =Parameters!CRM_URL.Value &"?ID={" & Fields!new_custentityid.Value.ToString() &
    "}&OTC="+Parameters!CRM_OTC.Value
    ```

4. Preview and save the report.

5. Copy the report to the destination server according to the instructions in the following section.

Copy a report between deployments

After you redeploy Microsoft Dynamics CRM from one server to another, you must individually copy custom reports to the redeployed (destination) server.

To redeploy one or more reports

1. When you copy a report that references a custom entity type code, you should update the definition of the CRM_OTC report parameter to reflect the object type code of the entity in the destination system.

    From the destination Microsoft Dynamics CRM server, open a browser window and go to the source system Microsoft Dynamics CRM.

2. In the Microsoft Dynamics CRM Reports list, select the custom report. On the toolbar, click **Edit** and then click **Actions**. In the **Actions** menu, click **Download Report**.

3. Click **Save**. Specify a file name when you are prompted and save the report to the destination system. A copy of the report now exists on the destination system.

4. In the browser, go to Microsoft Dynamics CRM on the destination system.

5. On the toolbar located above the Microsoft Dynamics CRM Reports list, click **New**.

6. In the **Report:New Report** dialog box, fill in the source file location. Also, fill in the **Details** section of the form.

7. Click **Save and Close**.

8. You can now see your report in the Reports list. You may need to select an appropriate category or entity from the drop-down lists above the toolbar to see the report.

    If you don't want to define a default filter, go to step 14.

9. Select your report in the Reports list.

10. On the toolbar, point to **More Actions**, and then click **Edit Default Filter** from the menu.

11. In the Report Viewer dialog box, define the default filter criteria that will be used every time the report runs.

12. Click **Save Default Filter**.

13. Click **Run Report**.

After the report is tested and functions correctly, follow these steps to add the report to context-sensitive lists and forms in Microsoft Dynamics CRM.

To add the report to context-sensitive lists and forms in Microsoft Dynamics CRM

1. Select the report in the Reports list.

2. On the **Actions** toolbar, select **Edit**.

3. In the **Report** dialog box, fill in the **Categorization** section according to the following descriptions:

   - **Categories**: Categorizes the report according to its intended purpose. For example, a sales report can be included in context-sensitive lists and forms in the Sales area of Microsoft Dynamics CRM.

   - **Related Record Types**: Associates the report to specific entities, like an account or contact. The report can be displayed in context-sensitive lists and forms for those entities.

   - **Display In**: Allows the report to be displayed in context-sensitive forms and lists. You can restrict where the report is displayed by using this setting. If you intend to publish a hidden report, clear the **Display In** text box.

   - **Languages**: Associates the report to all or English language only.

4. Click **Save and Close**.

See Also

Report writing with CRM 2016 for online and on-premises
Publish reports

© 2016 Microsoft. All rights reserved. Copyright

Copyright