Infor Mongoose Creating and Customizing Reports

Release 9.03.x

Important Notices

Trademark Acknowledgements

Publication Information

Release: Infor Mongoose 9.03.x
Publication Date: January 22, 2017
Document code: lib_9.03.x_mongooseolh_mongoose_en-us

Contents

About this guide

This guide provides conceptual and procedural information related to creating and customizing reports in *Infor Mongoose*-based applications.

Use this guide as a reference at your site. This guide does not teach you about practices specific to particular applications. This guide shows the general information on how to create and customize reports.

Intended audience

This guide is designed primarily for these users:

- System administrators who are responsible for the production and scheduling of the various types of reports that can be generated in Mongoose-based applications
- Administrators and form developers who are responsible for designing and creating reports in Mongoose-based applications
- Users of any type who are tasked with (or interested in) customizing existing reports for their own uses

Prerequisite knowledge

To fully understand and use the information presented in this guide, you should first:

- Be familiar and comfortable with navigating and using your Mongoose-based application.
- Have at least some knowledge of and experience with creating or modifying forms in your Mongoose-based application.

Related documents

Much of the content of this guide, along with additional related information can be found in the online help for your Mongoose-based application. You can always find the latest edition of the help on the Infor Xtreme support site, as well as the Mongoose portal.

Contacting Infor

If you have questions about Infor products, go to the Infor Xtreme Support portal.

If we update this document after the product release, we will post the new version on this website. We recommend that you check this website periodically for updated documentation.

If you have comments about Infor documentation, contact documentation@infor.com.

Reports Overview

1

Reports are available in a variety of forms and formats in Mongoose. Generally, they fall into one of these categories:

- Custom reports generated using a Report type form
- Custom reports generated using DataViews
- System standard form reports
- Reports generated using Microsoft SSRS

Custom reports from Report type forms

Mongoose now has a form type called "Report." The Report type of form allows you to create Mongoose forms that are designed to generate templates for custom reports. This is an improvement over other types of custom reports in Mongoose, in that you can both run and publish the report without having to use third-party reporting engines. This kind of report is ideal for reports that you want to generate on a regular basis, either on demand (by running the form) or automatically (as a background task).

Custom reports from DataViews

For certain kinds of reports, the DataViews functionality in Mongoose can be a valuable and easy tool to generate them. These are most commonly used for reports that you might want to generate "on the fly", without wanting to invest a lot of time and effort in formatting the results.

These reports also do not require the use of a third-party report generation engine, but they do require that you be licensed to use DataViews.

System standard form reports

The system includes a number of standard report forms created for the purpose of generating certain reports. These reports can be identified by the use of the word "Report" in the form name/caption (for example, User Authorization Report).

Reports using the Microsoft SSRS reporting engine

While you can still create and use these kinds of reports (and we still support them for backward compatibility), SQL Server Reporting Services (SSRS) reports are now deprecated in Mongoose. Instead, we recommend that you use either the new Report type of form or a DataView-based report.

Publishing reports

Depending on the type of report, reports can be published to PDF or Excel files, which can then be delivered to printers, email addresses, or fax machines. Most reports also allow you to set various options and preview the output before actually publishing them.

Customizing/Modifying reports

If you have the required editing permissions, you can customize reports of any type to your needs. The requirements for customizations and modifications depend on what type of report you want to customize/modify.

Previewing Reports

Most reports allow you to preview the output before generating it.

To preview a report:

1 Open a report criteria form and define the report parameters.

2 Click **Preview**.

Note: After you click the **Preview** button, a preview window displays. Closing the preview window closes the message window and prevents the preview from opening. However, the preview background task continues to run and preview files are generated. If you believe there is a problem with a preview, and you would like to cancel the background task, contact your system administrator.

Report previews behave the same as generated reports, with these exceptions:

- Previews are not sent to a printer.
- Previews cannot be configured from the Report Options form. Previews use global settings that are defined on the Intranets form.
- For previews, the format depends on the type of report:
  - Reports created from Report type forms display in a separate preview display window.
  - DataView reports and standard form reports display in a DataView Results form.
  - Reports created using SSRS display in a separate preview display window.
- Email notification is not needed or supported.

Generating Reports

To generate a report means to output it to a printer, to an email recipient, or to a fax destination. When you click the **Print** button on a report form, you generate the report.

In most cases, we recommend previewing the report before generating it. This gives you a chance to see what the output will look like and make any necessary changes before actually outputting the report.

When you generate a report, the report is sent to the background task queue, where it is then routed to the report generator. From there, it is routed either to a printer, to an email recipient, or to a fax destination, depending on the options for that report as set on the Report Options form.

Where do report outputs go?

When a report is previewed or generated, the output can go to any of a number of different locations, depending on various settings within Mongoose. The Infor Task Manager (or TaskMan, and not to be confused with Windows Task Manager) is set to send report outputs according to these rules:

- If the process default **Report output obfuscation** is set to **1**, then all report output goes to a single system folder. The filename for each consists of the report name appended with a session ID number.

  If the **Report output obfuscation** process default is not set or is set to **0** (which is the default), then TaskMan looks in the next location.
- The next location TaskMan looks at is the Report Options form.
  - If a report profile has been created there for the report, then TaskMan checks to see if a destination for report outputs has been set in the **Output Directory** field. If a destination has been set in the **Output Directory** field, then TaskMan uses that output path.
  - If no **Output Directory** has been specified, then TaskMan uses the output setting specified on the Intranets form.
- If no report profile for a report exists on the Report Options form, then TaskMan checks next to see if the Sites or Sites/Entities form has a **Report Output Folder** designated for report outputs. If so, TaskMan uses that folder as the destination for report outputs. If not, then TaskMan uses the output setting specified on the Intranets form.
- Finally, if no other output destination has been defined elsewhere, then TaskMan uses the output path as set in the **TaskMan Path** field of the Intranets form.

In all cases where the **Report output obfuscation** process default is **0** (zero), TaskMan uses the designated report output path and looks for a subfolder labeled with the user ID of the user who initiated the report generation. If that subfolder does not exist already, then TaskMan creates it before placing the output in that subfolder.

Printing Reports

When you print a report, the report data is first sent to the report generator and then to the printer. The system uses printing options as set on the Report Options form or the Intranets form. You can select a printer other than your default printer. You can also preview the report before you actually print it.

Note: Before reports can be printed, TaskMan must be set up as described in the *System Administration Guide*.

1. Verify that the Report Options form is set up for the report you are running.

   Make sure the **Output Format** field for this report is set to **Printer**.

If no option record exists for your report, the report output defaults to the format as set on the Intranets form. The default output format on the Intranets form is **Acrobat Format** (PDF). Other output formats are available for your use, but only the PDF format is supported by Infor.

Note: If you do not select **Printer** as the output format on either the Report Options form or the Intranets form, the output does not go to a printer.

2 Optionally, to be notified by email that the report has been printed, verify this information:

- On the Report Options form, the **Email Notification** field is set to **Yes** for your user ID or for the report you are printing, or both.
- Your email address is listed for your record on the Users form.
- On the Intranets form, the **Send Email Notification** field is selected.

3 Open the report form and define the report parameters.

4 Optionally, to send output to a printer other than your default printer, use these steps:

a Click **Select Printer** to open the Report Options form.

Note: In this step, the Report Options form opens with limited functionality. This functionality is restricted to allow you only to select a printer. To make other changes to the report options for this report, you must perform Step 1.

b In the Report Options form, if necessary, scroll to the right to locate the **Printer Name** column.

c Specify the printer you want.

d Save and close the Report Options form.

Note:

You can select different printers for successive print jobs, but you must be careful: TaskMan uses the printer you have designated at the time it runs the print job.

Suppose, for example, that you queue up one print job to use Printer A and then a second print job to use Printer B. If TaskMan already had a queue and did not pick up the first job before you switched printers for the second job, then TaskMan uses Printer B for both print jobs.

To be safe, it is best to let one print job finish, if possible, before initiating the second print job.

5 Optionally, preview the report output.

6 Click **Print**.

Emailing Reports

When you send the report as an email, the system sends it to the email address that is specified on the Report Options form, or on the Intranets form if no address is specified on the Report Options form).

Note: Before reports can be printed, TaskMan must be set up as described in the *System Administration Guide*.

To email a report:

1 Optionally, verify that the Report Options form has been set up for the report you are running:

- Make sure the **Output Format** field for this report is not set to **Printer**.

  Note: If no option record exists for your report on the Report Options form, the report output defaults to the format as set on the Intranets form. The default output format on the Intranets form is **Acrobat Format** (PDF). Other output formats are available for your use, but only the PDF format is supported by Infor.

- Verify that the **Email Notification** and **Attach Report** fields are both set to **Yes**.

  If you do not want to attach a copy of the report to your email notification, you can leave the **Attach Report** field set to **No**. If you later want to view the report, you can find the output file at: *TaskMan path*\Report\Output Files\YourUserID

2. Open the report form and define the report parameters.
3. Optionally, preview the report output.
4. Click **Print**.

Faxing Reports

For information to configure fax functionality, see "Fax Configurations" on page 11.

To fax a report:

Note: Before reports can be printed, TaskMan must be set up as described in the *System Administration Guide*.

1. Optionally, verify that the Report Options form has been set up for the report you are running.

   Make sure the **Output Format** field for this report is not set to **Printer**.

   Note: If no option record exists for your report on the Report Options form, the report output defaults to the format as set on the Intranets form. The default output format on the Intranets form is **Acrobat Format** (PDF). Other output formats are available for your use, but only the PDF format is supported by Infor.

2. Verify that the fax-related fields have been configured correctly.
3. Open the report form and define the report parameters.
4. Optionally, preview the report output.
5. Click **Print**.

Fax Configurations

The system supports several configurations for faxing reports.

Windows Fax

By default, the system uses the Fax capability built into the Windows operating system. Windows Fax runs on the TaskMan machine and uses a standard fax modem. The TaskMan service calls an executable that initiates faxing of a report through the modem. This configuration is recommended for light-duty faxing.

Infor Framework Fax Service

The Infor Framework Fax Service can be installed on a fax server machine rather than the TaskMan machine (see the Infor Mongoose Installation Guide for instructions). The Infor Framework Fax Service monitors a directory on the TaskMan machine and initiates faxing through the built-in Windows Fax capability on the fax server, which is configured with a standard fax modem. This configuration might be preferable to the Windows Fax configuration where fax loads are higher.

Third-Party Fax Services

The system supports some third-party fax services. A fax service can be installed on a separate fax server machine with an appropriate modem or fax board. The service monitors a directory on the fax server, which is shared with the TaskMan machine.

Configurations and Report Options

A fax configuration applies to all reports that can be faxed and to all users. Administrators can override configuration options for a specific report or user.

Creating reports

2

Mongoose offers the capabilities to produce four types of reports:

- Custom reports generated from Report type forms
- Custom reports generated from DataViews
- Standard reports, packaged with Mongoose
- Custom reports generated from SQL Server Reporting Services (SSRS) templates

Creation of custom reports is a two-stage process:

- Create the report object that will be used to define the source and potential contents of the generated report.

  Depending on the type of report, this object might be a Report type (template) form, a predefined DataView, or an SSRS report definition file.

- Create a report criteria form that is used to generate the actual report.

  Regardless of the object used to generate the form, each report type uses an associated report criteria form. This form is accessible to end users.

  Note: This includes the standard reports packaged with Mongoose. The forms that are used to generate these reports are actually report criteria forms for predefined DataViews that we created.

  Typically, this form provides options that users can select to determine what data is generated by and included in the report. This form also usually includes both a **Preview** button and a **Print** button.

Report type forms

About Report type forms

Report type forms are specialized Mongoose forms that you can create to produce templates for your own custom reports. Although they are true Mongoose forms, they behave a little differently than the more common types of forms.

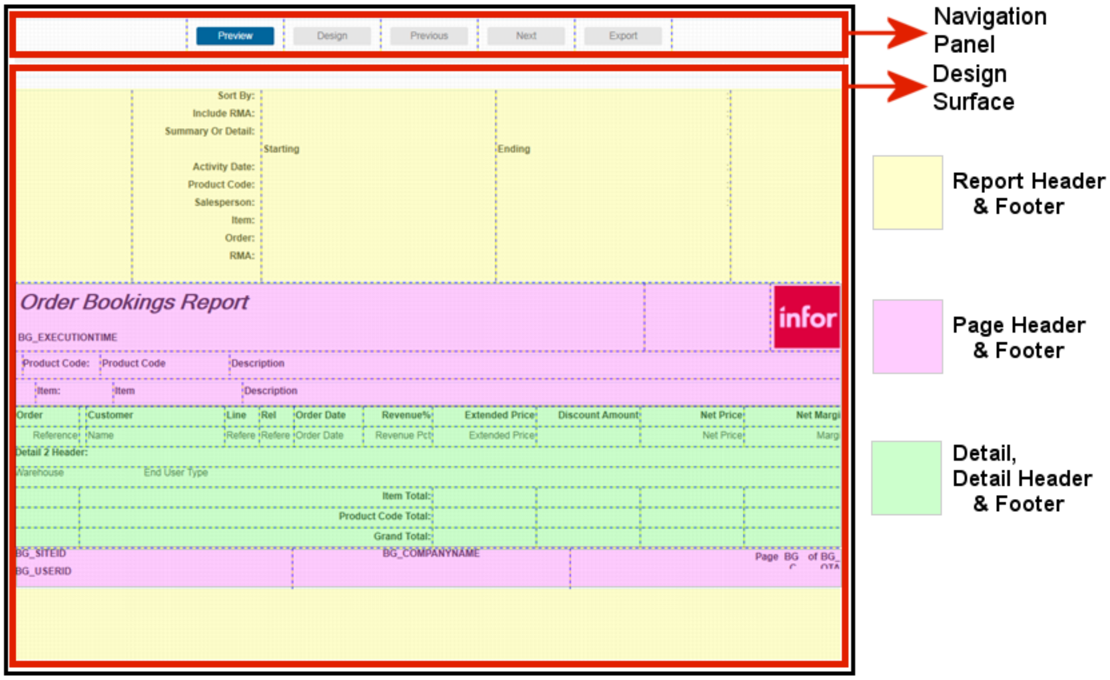
A report-template type of form actually contains only the layout and definition of a report. While it is possible to run a report from this form, the normal practice is to create an associated report criteria form, similar to those used to generate other types of reports.

As with other types of reports, you can generate this report on a regular basis, as a background task in Mongoose, or manually, on demand.

Note: In the web client, you can view a report-template type form as a template, but you cannot use the runtime preview function or generate a report from it.

The structure of a report-template type form

The basic report-template type form has two panes, the second of which is subdivided into five basic regions, as in this example.



See "About Report type form regions" on page 15.

Collections on report-template type forms

Report-template forms support the use of both primary collections and secondary collections, but not subcollections. This is because the Detail sections are designed to be repeatable sections, and are designed to handle only one row (record) of data at a time.

This means that you cannot use components that display more than one row of data at a time. This includes grids, DataViews, list boxes, trees, and so on. Because subcollections require the use of a grid to display their data, this also means that subcollections cannot be used in report-template type forms.

You can, however, use multiple secondary collections to access and display the data you want for your report. You can also establish multiple parent/child(/grandchild, /sibling, etc.) relationships between collections.

The hierarchy between collections is established by means of a Parent Collection property. Available for secondary collections only, this property allows you to specify a parent collection for each secondary collection. This hierarchy is used when processing rows for the report, to determine when each secondary collection is to be processed.

Orientation and paper sizes

The New Form Wizard determines the initial form size and orientation when generating reports. Both the orientation and paper size can be changed later, however. By using Glue properties along with the main FlexLayout region, you can adapt the form design to different paper sizes.

About Report type form regions

One specialized aspect of the report-template type of form is that it is comprised entirely of FlexLayout components and regions. This gives you a great deal of flexibility when designing and laying out your report-template form.

At the most basic level, the entire report-template form is contained within two FlexLayout components, by default named **NavigationFlexLayout** and **MainFlexLayout**.

The NavigationFlexLayout component

The **NavigationFlexLayout** component hosts the various buttons used during runtime, when you are designing, developing, and testing the report-template. This part of the form does not print or publish when you actually generate and publish the report.

The buttons, which are created by default as part of the report-template form creation, include these:

- **Run** - This button allows you to actually run the report in a kind of "preview" mode. When you click this button, the system displays the report exactly as it will appear in the published output.
- **Design** - This button allows you to view the basic template design on which the report output is built. This is useful especially when you are designing the report, as it shows you how each region in the report is designed to display.
- **Previous** - When active, this button displays the page of the report that comes before the currently displayed page.
- **Next** - When active, this button displays the page of the report that comes after the currently displayed page.
- **Export** - This button actually publishes the report to a PDF file, after prompting you for the location to which it should be created.

  Note: Although you can use this button to manually publish the file, it publishes all information exactly as presented in the preview. You cannot select what options, ranges, etc., to include. For this reason, it is generally more useful to use a report criteria form, even for manual publication of the report.

The MainFlexLayout component

The **MainFlexLayout** component contains regions for all the content that is actually published when the report is generated. This component contains these basic regions that are used to control the way the report information displays:

- Report regions - These include the **Report Header** and **Report Footer** regions. The Report Header region typically displays on just the first page of the report, before any records are processed. It is typically used to show what options were selected/used when generating the report.
- Page regions - These include the **Page Header** and **Page Footer** regions. These regions display on each page and are configured on a page-by-page basis. Examples of information presented in these regions includes page numbers, site information, user information, company logo, report title, and so on.
- Group regions - These include the **Group Header** and **Group Footer** regions. These are included only if you selected at least one group property in the New Form Wizard. They are used to order and group together data according to the properties they are grouped by.
- Detail regions - These include the **Detail Header**, **Detail**, and **Detail Footer** regions. These regions are generated repeatedly. The **Detail Header** displays before the first row in the **Detail** collection. Then the **Detail** subregion displays once for each row in the collection. Finally, the **Detail Footer** displays once after the last **Detail** row in the collection.

Each of these regions can itself be a container for whatever additional components are required to display the information on the report as desired.

Note: Any FlexLayout components you add to the Report type form are normal FlexLayout components and do not include the special attributes built into the FlexLayout components that were created as part of the original form.

All of these basic regions must specify a character-based length (height). The heights of these subregions must be a fixed value, as opposed to a flexible value, so that the system can accurately determine which regions can fit on each page as the report is generated.

About Collection Hierarchies in Report Type Forms

Report type forms can include data from several different IDO collections in a single report. This is done by adding secondary collections to the Report type form and then retrieving specific data from each collection.

This secondary collection data can also be retrieved in a hierarchical fashion, by specifying the primary collection or another secondary collection as the "parent" of a given collection. This is done by setting the Parent Collection property of a secondary collection to specify the collection you want as the "parent" collection. The parent collection might or might not be the primary collection.

This hierarchy determines the order in which each collection's data is processed in the report output.

For example, suppose you wanted to produce a report of customer orders that includes any notes, order lines, and shipment information, similar to this:

```
Order
    Order Notes
    Order Lines
```

```
        Order Line Notes
        Order Shipments
            Order Shipment Notes
```

In this example, the Order IDO collection would be the primary collection. There would be five secondary collections, one for each "descendant" of Order.

As we have set it up, two of these secondary collections--Order Notes and Order Lines--would be set up to use the primary collection, Order, as the "parent" collection. The Order Line Notes collection and Order Shipment collection would both be set up with the Order Lines collection as their parent collections. And the Order Shipment Notes collection would be set up with Order Shipments as its parent collection.

```
Order
    Order Notes (Parent collection: Order)
    Order Lines  (Parent collection: Order)
        Order Line Notes (Parent collection: Order Lines)
        Order Shipments (Parent collection: Order Lines)
            Order Shipment Notes (Parent collection: Order Shipments)
```

Creating a Report type form

The Report type of form is designed to make it possible for you to create your own custom report-template forms that you can then use to generate your custom reports. Create the report form using the New Form Wizard.

Note: In addition to this form, you should plan to create a second form, called a report criteria form, to allow a user to select report options for, preview, and print the report itself.

To create a report-template type form:

1. In Design Mode, launch the New Form Wizard.
2. On the first page of the wizard, select the **Create a new form** option and then click **Next**.
3. On the next page, complete these required fields:

   - In the **Name** field, perform one of these actions:

     - From the drop-down list, select the name of an existing form to use as the basis for the new form.
     - Type a unique name for the new form.

     Note: Because the basic purpose of this form type is to serve as the template for the actual custom report, we recommend that you use a naming convention in which you append the word "Template" to the end of the form name; for example: **MyReportReportTemplate**, where *MyReport* is the actual name of the report that will be output.

     Later, then, you should create the report criteria form for users to use and generate the custom report.

   - From the **Form Type** drop-down list, select **Report**.

- From the **Data Source** drop-down list, select an IDO to use as the primary collection.

4 Optionally, specify a default **Device Type** and any comments.

Comments are stored with the form definition and do not appear anywhere on the actual form.

Note: The **Form Layout** setting has no effect on a Report type form.

5 Click **Next**.

6 In the next page of the wizard (**Select Properties**), specify the properties of the IDO that are to be used in the report generated by the new form.

7 Optionally, in the next page of the wizard (**Properties**), specify:

- The order in which the properties are to be displayed
- The captions for properties/components
- What component class, if any, is to be applied to each property/component, along with any parameters to be passed

8 Click **Next**.

WinStudio displays the **Select Regions** page of the wizard.

9 In the **Select Regions** section, select which "boilerplate" regions to include in the report-template form.

10 In the **Set Orientation** section, specify whether to use a **Portrait** orientation or a **Landscape** orientation for the report form and its published output.

11 Optionally, in the **Select Group Properties** section, select which properties to use to group returns.

For example, suppose you want to group a report on Quarterly Sales by Salesperson. You would select the Salesperson property in this section, and the system would then group the sales figures by salesperson.

You can select multiple properties to group by. If you do, the system processes the data in the order in which you list them here. To change the order, use the up and down arrows.

As with other property selection sections, the right panel indicates which properties are to be included.

12 Click **Next**.

13 Optionally, to create a template from the new form, select the **Save Template** option.

14 Click **Finish**.

WinStudio then creates the form and displays a rough-draft version in Design Mode.

15 Use Design Mode to finish crafting your report form.

You can now modify and customize all regions of your report, replace the corporate logo image with one of your own, and so on. Because the Report type of form is based on the FlexLayout component type, you should be familiar with that component type before attempting to modify your report form.

In addition, you can employ secondary collections and set the hierarchies between them (and the primary collection). To set those hierarchies, use the **Parent Collection** property on the **Collections** property sheet.

Note: Once you finish designing your report template, you should create an accompanying report criteria form. Once that form is created, come back to the template form and set up the required associations between the two forms.

DataView reports

About DataView reports

DataView reports can be produced from any type of DataView result. In most cases, you can prepare and generate DataView reports directly from the DataView Results form or the DataView Form Results form. These report outputs are most useful when you need to create a quick *ad hoc* report.

You can, however, also create standard-type reports based on DataViews. These standard-type DataView reports must be based on predefined DataViews. To produce the standard-type report output options, then, you must also create a report criteria form.

Creating a DataView report

DataView reports must be based on predefined DataViews. The DataView defined for report output is much like any other predefined DataView, but with a few specific requirements for the output.

To create a DataView report:

1 Create a predefined DataView that contains all the data fields/columns you want to display on the report output.

When creating the predefined DataView, make sure you include these specifications, in addition to whatever other specifications you want to make on the DataViews Setup form:

- Select the specific IDO properties to include in the output.

  This is done by clicking the **IDO Setup** button on the **General** tab, and then by selecting the desired properties in the **Properties** section of the DataView IDO Setup form.

  You can also set additional IDO parameters using the DataView IDO Setup form.

- Optionally, create input parameters for the DataView to use when creating the output display.

  Note: Although this is optional, we strongly recommend that you create input parameters, especially for the starting and ending values of ranges.

  This is done by selecting the **Input Parameters** tab on the DataViews Setup form, and then creating the necessary parameters. These input parameters require an IDO property name, an operator, and optionally, a description consisting of a string.

- Optionally, set user permissions to determine who has the ability to access and generate the report.

  Note: Although this is optional, we strongly recommend that you set access permissions for the report. Otherwise, anybody has the ability to access and generate the report.

  This is done by selecting the **User Permissions** tab and then setting the individual user and/or the group permissions.

2 Launch the DataView, and verify that the data you want is displayed correctly:

a In the DataView Results form, click **Launch**.

If you created input parameters, the system displays DataView Input dialog boxes that prompt you for the parameters by which to filter the results. If you leave these prompt fields blank and click **Next** or **Finish**, the system returns all applicable data for each field.

b Optionally, in the DataView Results form, rearrange the layout, select what data to include, etc.

c When you have the report display set to your requirements, save the layout.

Note: Although it is not required that you save a layout, we recommend that you do so. If you try to create a background task without a saved layout, it can cause problems later with report generation.

3. Close the DataView Results form, return to the DataViews Setup form, and refresh it.
4. Select the **Layouts** tab and verify that your layout displays in the list of layouts.
5. Save and close the DataView Results form.
6. To have the report generated automatically, according to a schedule, create a background task definition for it, using the Background Task Definitions form. At a minimum, specify these values:
   - For the **Executable Name** field, specify the DataView name and the name of the layout to use. Use this syntax:

     DataViewName-LayoutName

     where:

     - *DataViewName* is the name of the DataView exactly as it appears on the DataViews and DataViews Setup forms.
     - *LayoutName* is the name of the layout you want the system to use when generating the report.
     - The two names are separated by a hyphen and no spaces.
   - For the **Executable Type**, specify **RPT**.
   - For the **Report Type**, specify **DATAVIEW**.
7. Create the report criteria form.

Setting up a new DataView

To set up (create) a new predefined DataView:

1. On the DataViews Setup form, execute Filter In Place.
2. Click in the Grid View, and then click in the auto-insert (last, empty) row.
3. In the **DataView** field, specify a name that best describes the data to be presented.
4. Optionally, to prevent modifications to the DataView by users other than those with Vendor Developer permissions, select the **System Record** option.

   This option allows you to protect the structure of DataViews that you deliver to customers/users. It prevents others from deleting or modifying any content you provide, while also allowing customers to add their own content.
5. Optionally, to designate a report caption/title, in the **Caption Override** field, specify the caption.

   This can be a translatable string or a literal value.

When specified, this string replaces the default system-generated caption on DataView report outputs.

Note: You can also specify a layout-specific caption override, by setting the **Caption Override** field on the **Layouts** tab. If both that and this field have caption overrides, the layout-specific override takes precedence.

6 Optionally, use the **Report Orientation** field to set the orientation (**Landscape** or **Portrait**) for the DataView report.

This setting is used for both the layout of the PDF that is generated and for printing (when sent directly to a printer).

7 On the **General** tab, specify the required and optional information for each IDO that is to be included in the DataView queries.

See "Specifying DataView setup information - General tab" on page 22.

8 Optionally, after specifying each IDO, to make additional IDO specifications for the DataView, click **IDO Setup** and follow the procedure in the topic "Setting additional IDO specifications for a DataView" on page 24.

9 Optionally, use the options on the **Input Parameters** tab to add input values or ranges.

See "Specifying DataView setup information - Input Parameters tab" on page 25.

10 Optionally, use the options on the **User Permissions** tab to control who has access to this DataView.

See "Specifying DataViews setup information - User Permissions tab" on page 26.

11 Optionally, view information about or make copies of layouts on the **Layouts** tab.

See "Specifying DataView setup information - Layouts tab" on page 27.

12 Save your work.

Note: If the intended use for this DataView is to be used for report output, you should create an associated report criteria form.

Setting Up Predefined DataViews

Using the DataViews Setup form, you can create and maintain predefined DataViews, select the IDOs to use when results are displayed to the user, designate user and group permissions, and select data layout options.

The definition for a predefined DataView consists of up to four basic sets of options:

- The general settings are the fundamental settings that define the DataView. These settings are made on the **General** tab:
  - The DataView name
  - The IDOs that the DataView is set to query
  - How many records the query is limited to
  - The definitions of any filters to be used in the query
  - Other related information

  For more information, see "Setting up a new DataView" on page 20.

- Input parameters are considered optional and are set using the **Input Parameters** tab. Input parameters are used as filtering mechanisms to limit the amount and type of data retrieved. These parameters can be especially useful in setting up DataViews as sources for formal reports.
- You can control who has access to what DataViews by setting user and group permissions on the **User Permissions** tab.
- You can view information about and make copies of layouts, using the **Layouts** tab.

Specifying DataView setup information - General tab

When you set up a predefined DataView, specify this information on the **General** tab of the DataViews Setup form.

Note: You can specify multiple data sources to be queried in the DataView. Specify individual information for each data source.

Option	Required or Optional?	Description/Comments
IDO Setup	Optional	Use this button to launch the DataView IDO Setup form and specify exactly what data you want to use from a specified IDO. You can use this button/form for each IDO that you specify for a DataView, to define and limit exactly what data from each IDO is to be queried. See "Setting additional IDO specifications for a DataView" on page 24.
Source Type	Required	Specify whether the data source is to be an **IDO Collection** or a custom load **IDO Method**.
IDO	Required	Specify what IDO is to be the source for the data to be returned.
IDO Alias	Required	Verify that the IDO alias is what you want. This value is generated automatically. You can modify the recommended alias.
Source Name	Required when enabled	Specify the name of the custom load IDO method being used as the data source. This field is enabled only if **IDO Method** is selected as the **Source Type**.
Parent IDO	See Description/Comments.	Use this field to specify an IDO collection that is to be considered the "parent" of the subcollection specified in this row. This field is enabled only if **IDO Collection** is selected as the **Source Type**. It is required only if the specified IDO collection is a subcollection.

Option	Required or Optional?	Description/Comments
Record Cap	Optional	Use this field to specify the maximum number of records to return when the DataView query is performed. Select from these options: • **Use System Setting** - The system record cap setting determines the maximum number of records that can be returned. • **Use Specified Max** - This option allows you to set your own maximum for the number of records to be returned. When selected, this option makes visible and enables the **Record Cap Value** field. • **Retrieve All** - This option instructs the system to retrieve all available records, regardless of any record cap settings elsewhere on the system.
Record Cap Value	See Description/Comments.	This column is visible and enabled only when the **Record Cap** setting is **Use Specified Max**. By default, this field contains the system setting for the record cap. You can change it to whatever value you choose.
Filter	Optional	Specify a filter for the system to use when performing the query. This option is enabled only when **Source Type** is **IDO Collection**. To filter data when **Source Type** is **IDO Method**, you must specify the filter parameters within the method itself.
Order By	Optional	Optionally, specify which property is to be used to sort the retrieved results. This option is enabled only when **Source Type** is **IDO Collection**.
Link Type	Optional	Specify whether the parent-child link is to be a multi-level or single-level link. **Note:** This option is enabled only if a **Parent IDO** is specified. • The **Multi Level** option (default) nests and indents the child IDO information under the parent information. • The **Single Level** option links and presents the parent and child IDO information at the same level.
Link By	Optional	Specify how the subcollection is linked to the parent IDO collection. **Note:** This option is enabled only if a **Parent IDO** is specified.

Option	Required or Optional?	Description/Comments
		The syntax is the same as described in "IDO Link By Editor" on page 25.
System Record	Optional	To prevent modifications to the setup for the selected IDO by users other than those with Vendor Developer permissions, select this option. This option is slightly different from the System Record option for the DataView itself, in that it protects only the selected IDO setup. The exceptions to this rule are the Record Cap settings and the Show Notes settings.
Show External Notes	Optional	To display any external notes that might be attached to the DataView, select this option. When this option is selected, external notes are displayed as child layers in the DataView, underneath the IDO.
Show Internal Notes	Optional	To display any internal notes that might be attached to the DataView, select this option. When this option is selected, internal notes are displayed as child layers in the DataView, underneath the IDO.

Setting additional IDO specifications for a DataView

When you create a predefined DataView, you can set up the associated IDO so that only the desired data is retrieved when the DataView is displayed. This ensures that you minimize the retrieval time and get only the data that you really need.

Note: This information applies only to predefined DataViews. The data for other types of DataView displays is controlled by the source from which the display was launched.

Also, the information in this topic is closely related to the information in the topic "Specifying DataView setup information - General tab" on page 22. In general, when information is presented in that topic, it is not repeated here. Only additional information and the expanded capabilities of the DataView IDO Setup form is presented here.

To configure additional IDO specifications for a DataView, use these guidelines:

- **IDO** section - Depending on the **Source Type**, you can change or modify several of the settings in this section. You can also set or modify these settings on the General tab of the DataViews Setup form.
- **Link By** section - As with the **Link By** option on the DataViews Setup form, these options are enabled only when a **Parent IDO** is specified on that form.

  This form, however, provides additional specification options. You can specify which properties are to be used to link and display data. Do this by specifying a **Parent Property** and a **Child Property** and then clicking **Add**.
- **Order By** section - As with the **Order By** option on the DataViews Setup form, these options are enabled only when the **Source Type** is **IDO Collection**, as specified on that form.

This form, however, provides additional specification options. You can specify properties by which to sort the data that is displayed. Do this by specifying one or more properties by which to sort the data and whether the data is to be displayed in **Ascending** or **Descending** order; and then clicking **Add**.

Note: If a record cap is in use, the **Order By** setting determines which records are selected for retrieval first.

- **Properties** section - You can specify exactly which properties are to be included in the data query by selecting them in this section. Properties that are not "**Selected**" are not included in the return results for the data query.

  In addition to selecting which properties to include in the data query, you can also view and/or specify these settings for each property:

Setting	Description
Description	Read-only. This field displays any description provided for the property, as specified on the IDO Properties form.
Default Caption	Read-only. This field displays any default caption currently defined for the property, as specified on the IDO Properties form. You can override the default caption, using the **Caption Override** field.
Caption Override	Use this field to override the default caption for a property. Whatever you provide as a caption override is what displays as the column header for that property in the DataView Results form.
System Record	When selected, this option prevents anyone other than a Vendor Developer from making modifications to or deleting a property specification for the DataView.

IDO Link By Editor

Use this form to specify how a subcollection is linked to the parent collection. For example, you might specify that the parent collection's CoNum property must equal the subcollection's CoNum property.

To add the specified pair of linked properties to the Link By specification, click **Add**.

To add multiple pairs, do not use AND between the pairs. Instead, separate them with commas and no spaces (for example:**CoNum=CoNum,CoLine=CoLine**).

To remove a selected pair from the Link By specification, click **Remove**.

Specifying DataView setup information - Input Parameters tab

When you set up a predefined DataView, you can specify this information on the **Input Parameters** tab of the DataViews Setup form.

Note: The options on this tab can be especially helpful when setting up a DataView to be used to create reports. You can use input parameters to specify exactly what data can be used for query ranges or other return results, using a report criteria form associated with the DataView.

Column	Description
Sequence	These integers indicate the order in which the specified input parameters are queried. You can assign this number at the time of the parameter's creation. Once the parameter has been saved, you cannot change it.
Property Name	Use the drop-down list to specify the IDO property to be used for the input parameter. When you select a property in this field, the **Description** field is automatically populated with the default value for the property, if one exists. You can modify the value in the **Description** field, if desired. **Note:** If the **Source Type** on the **General** tab is set to **IDO Method**, this field is disabled.
Operator	When used, this field specifies an operation to use in setting a range for the data to be retrieved. When two such operators are used in tandem, you can set an upper range limit and a lower range limit. For example, suppose you want to retrieve the values of all customers whose names begin with the letter 'C'. You could create one parameter using the operator **Greater Than Or Equal To** with a **Description** value of **C**. You could then create a second parameter using the operator **Less Than**, with a **Description** value of **D**. By applying both of these parameters to the **Property Name** of **CustName**, you effectively set a range for customers whose names begin with the letter 'C'. **Note:** If the **Source Type** on the **General** tab is set to **IDO Method**, this field is disabled.
Description	Specify the value to be used for the input parameter. If a **Property Name** is specified, this value is specified automatically but can be changed. This is the value on which the **Operator**, if any, performs.
End of Day	This option is enabled only when the specified property is a date/time property. When selected, this option causes the time on a Date/Time property value to be set to the "end of day," which is defined in the system as **11:59:59.99 PM**. When cleared, the system uses the system date/time that the user performed the query. The use of this option allows you to set very precise date/time ranges. **Note:** If the **Source Type** on the **General** tab is set to **IDO Method**, this field is disabled.

Specifying DataViews setup information - User Permissions tab

Use the **User Permissions** tab on the DataViews Setup form to specify who can access the DataView that is being created or modified. You can specify permissions at the user level, the group level, or a combination of the two.

Column	Description
User Name	Specify users who are to have access to the DataView. The list is populated from the list of users in the system.
[User] **Description**	(Read-only) This field displays the description associated with the specified user, as specified on the Users form.
Group Name	Specify user groups that are to have access to the DataView. The list is populated from the list of user groups in the system.
[Group] **Description**	(Read-only) This field displays the description associated with the specified user group, as specified on the Groups form.

Specifying DataView setup information - Layouts tab

When you set up a predefined DataView, you can specify this information on the **Layouts** tab of the DataViews Setup form.

Note: You can also use this tab to create copies of layouts. These duplicate layouts can each have a different scope and can be designated as the default version of the layout if desired.

Field/Option	Description
Copy Layout	To create a copy of a layout that you can use as the basis of a new layout, click this button. When you copy a layout, the copy initially has all the same field values, except for the **Scope Type**, which defaults to **Site**.
Layout	Use this field to name and identify a new layout. For layouts that have already been saved, this field is read-only and shows the name of the layout as it was saved.
Scope Type	Use this field to specify the level of scope to which the layout is available. The default scope for a copied layout is **Site**. If you specify a scope of either **Group** or **User**, you must also specify a **Group Name** or **User Name**, respectively. If you have Vendor Developer editing permissions, and you want to define the scope type as **Vendor Default**, enter **0** (zero) in this field.
Scope Name	If you specify either **Group** or **User** in the **Scope Type** field, you must specify the name of the applicable user group or the individual user in this field. **Note:** If **Group** is specified as the **Scope Type**, this column's header displays as **Group Name**. If **User** is specified as the **Scope Type**, this column's header displays as **User Name**.
Default	When there are multiple layouts with the same **Scope Type** and **Scope Name**, this option specifies the default layout. The **Default** layout option is set using the DataView Layouts form. On this form, the option is read-only.

Field/Option	Description
Caption Override	Optionally, to designate a caption/title to use for the selected layout, specify the caption. This can be a translatable string or a literal value. When specified, this string replaces the default system-generated caption on DataView report outputs. **Note:** You can specify a DataView-level caption override, by setting the **Caption Override** field at the top of the form. If both fields have caption overrides, the layout-specific override here takes precedence.
Report Orientation	Use this field to set the orientation (**Landscape** or **Portrait**) for the selected layout. **Note:** You can specify a DataView-level orientation, by setting the **Report Orientation** field at the top of the form. If both fields have orientations specified, the layout-specific setting here takes precedence.

SSRS reports

About SSRS reports

Infor Mongoose can be used to create and generate SQL Server Reporting Services (SSRS) reports.

Note: Although Mongoose still supports the creation and production of SSRS reports, they are now deprecated. Instead, we encourage the use of the Report type form or DataViews to create reports within Mongoose.

As with other types of reports, SSRS reports are typically made up of different sections. Most reports have these sections:

- Page header
- Report header
- Data section
- Page footer

Some reports include additional features, such as notes.

SSRS has four container classes. Three of these are subclasses of the Tablix class: Lists, Tables, and Matrices. The fourth container class is the Rectangle. Data stored in a Tablix object can be filtered, sorted, and grouped. Tablix objects also change their size and shape to best fit the space provided by a particular output format. Rectangles are used to contain other objects, like tablix objects, textboxes, images, and so on. Rectangles are useful to restrain the movement of objects, such as tables, as they grow. This is especially important for reports like "label reports," where precise control of object positioning is required.

In general, most reports use a rectangle for the report header, page header, page footer, and one or more tables for detail data. In some instances, where there is a parent-child relationship to data, a list format can be used for the parent data, and one or more tables for the child data.

Virtually any element or component of an SSRS report can be formatted independently of the others. You can format the backgrounds of each section separately, and the text fonts and colors as well.

Example: Creating an SSRS report based on a stored procedure

Note: This example procedure is not guaranteed to be current or accurate with the SSRS tools and processes now available. Since SSRS reports are now deprecated in Mongoose-based applications, we are including this and related topics only for legacy support purposes.

This example, which you can also use as a short tutorial, creates a report based on a stored procedure that displays data from a "UserTask" table. This simple report will be designed to list tasks, grouped by user.

This example assumes that you are using Visual Studio 2008 or SQL Server Business Intelligence Development Studio. Note that the steps would be very similar if you were to use Report Builder 3.0.

Although this example provides the procedural steps for using these tools, this is not to be considered a tutorial on how to use these tools. For information on how best to use these tools, see their respective product documentation.

To create an SSRS report based on a stored procedure, perform these procedures:

1. Create the report project.

   See "Creating an SSRS report project" on page 30.

2. Create the report definition file.

   See "Creating an SSRS report definition file from a template" on page 31 or "Creating an SSRS report definition file without a template" on page 31.

3. Create the stored procedure.

   See "Creating a stored procedure for an SSRS report" on page 32.

4. Set the data source properties.

   See "Setting data source properties for a stored procedure-based report" on page 35.

5. Modify the report parameters.

   See "Modifying SSRS report parameters" on page 36.

6. Set the report title.

   See "Setting an SSRS report title" on page 37.

7. Test the report configuration.

   See "Testing the SSRS report configuration" on page 38.

8. Populate the report with data.

See "Populating an SSRS report with data" on page 38.

9 Configure the report layout.

This includes such considerations as these:

- Adding fields and labels to the report to help with the data being displayed
- Formatting backgrounds of report sections
- Formatting the fonts and font colors being used to display the data
- Adding images, such as corporate logos

10 Reset the data source connection string.

See "Resetting the data source connection string" on page 38.

11 Deploy the report to the report server.

See "Deploying an SSRS report to the Report Server" on page 39.

Although the SSRS report definition file has been created and deployed, you still cannot generate the report, either manually or automatically. Before you can do that, you must:

- Create the report criteria form that allows users to specify the data they want to include in the report output.
- If you want to have the report generated automatically, create a background task definition and schedule a background task.

Creating an SSRS report project

To create an SSRS report project:

1. Run Visual Studio 2008 or SQL Server Business Intelligence Development Studio.
2. From the **File** menu select **New > Project**.
3. In the New Project dialog box, select:
   - **Project types: Business Intelligence Projects**
   - **Templates: Report Server Project**
4. For the **Name**, specify:
   - For a stored procedure-based report, **ReportSPDemo**
   - For an IDO-based report, **ReportIDODemo**
5. For the **Location**, specify **C:\Reports**.
6. Click **OK**.

Visual Studio creates the report definition project with a set of default folders.

Report definition files

About report definition files

The foundational phase of creating an SSRS report is the creation of the report definition file. Report definition files are special XML files that contain all the data and formatting specifications for a named report that can be generated. The system uses this file, which must have the extension ".rdl", to generate and render reports when requested.

Report definition files are created using tools provided by Microsoft. It is beyond the scope of this topic to specify and explain those tools. We recommend that you search online for the current set of report building tools from Microsoft.

It is possible to create a report definition from scratch but it is much easier if you begin with one of the report definition templates provided by Infor. These report templates are typically available in this directory:

C:\Program Files (x86)\Infor\Mongoose-based application\Report\Reports\Templates

where *Mongoose-based application* is the name of your Mongoose-based application.

The report templates can also be obtained from the Infor Xtreme Support site. They are contained in a zip file named **SSRS_Report_Templates.zip**, which you can download and unzip to the directory specified above. To find the zip file, you can search on "SSRS Report Templates". This and related topics discuss only the techniques for creating report definition files using these templates.

Creating an SSRS report definition file without a template

Note: For the working examples, we use and modify one of the report templates Infor provides. The procedure in this topic is included only to provide an example of how to create a report definition file without using a template. If you are using the working example as a tutorial, you need not complete this procedure.

To create an SSRS report definition file without a template:

1. With the report project open in Visual Studio, in the Solution Explorer, right-click **Reports** and, from the context menu, select **Add > New Item**.
2. In the Add New Item dialog box, **Templates** field, select **Report**.
3. In the **Name** field, specify a name for your report definition file, and then click **Add**.

Visual Studio creates the RDL file and displays a **Design** workspace in which to design and build your report definition.

Once the basic report definition file has been created, you can proceed to design the report output with the items, components, and data available in Visual Studio 2008.

Creating an SSRS report definition file from a template

To create an SSRS report definition file from a template:

1. Locate the template you want to use as the basis for your report.
   In these examples, we use these files:

- **Template_Report_Landscape.rdl** for a stored procedure-based report
- **Template_Report_Landscape_IDO.rdl** for an IDO-based report.

2. Copy the template file into your report project folder.
3. Rename your copy of the template file as either **UserTaskSP.rdl** or **UserTaskIDO.rdl** (for the stored procedure report or the IDO report, respectively).
4. In Visual Studio, in the Solution Explorer, right-click **Reports**.
5. From the context menu, select **Add > Existing Item**.
6. Select your *.rdl file and then click **Add**.

   Visual Studio adds your *.rdl file to the list of reports.
7. To open your *.rdl file, double-click its name in the Solution Explorer.

   Visual Studio displays a workspace with the report definition file in Design Mode.

   Notice that the report template already has report headers and footers configured. Many of the entries in the headers and footers are preconfigured to pick up variable values from within the system and need not be modified.

Creating a stored procedure for an SSRS report

Before creating the report definition, we must create a stored procedure that will be used as the data source for the report. This stored procedure is to be called **Rpt_UserTaskSp**.

Note: When creating stored procedures and other system elements, we recommend strongly that you follow the naming conventions already in place. For example, with this stored procedure, we identify it as a stored procedure to use in conjunction with reports using the prefix **Rpt**. We identify it as a stored procedure with the suffix **Sp**.

To create the stored procedure for this example:

1. Open SQL Server Management Studio and connect to your database server.
2. Select your application database and click **New Query**.

   SQL Server Management Studio displays a blank query window.
3. Copy and paste this stored procedure code into the query window:

```
-- =================================================================

-- Stored Procedure: Rpt_UserTaskSp
--
-- This is a report stored procedure used by the UserTask report demo.
-- =================================================================

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

IF EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Rpt_UserTaskSp]') AND type IN (N'P', N'PC'))
DROP PROCEDURE [dbo].[Rpt_UserTaskSp]
GO
CREATE PROCEDURE Rpt_UserTaskSp
(
  @UsernameStarting             UsernameType       = NULL
, @UsernameEnding               UsernameType       = NULL
, @TaskNameStarting             MessageSubjectType = NULL
, @TaskNameEnding               MessageSubjectType = NULL
, @RemindDateTimeStarting       DateTimeType       = NULL
, @RemindDateTimeEnding         DateTimeType       = NULL
, @RemindDateTimeStartingOffset DateOffsetType     = NULL
, @RemindDateTimeEndingOffset   DateOffsetType     = NULL
) AS

-- Transaction management.
BEGIN TRANSACTION
SET XACT_ABORT ON

-- Set the isolation level specified for the background task
-- or use the system default.
IF dbo.GetIsolationLevel(N'UserTaskReport') = N'COMMITTED'
   SET TRANSACTION ISOLATION LEVEL READ COMMITTED
ELSE
   SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

DECLARE
  @RptSessionID  RowPointerType
, @LowDate       DateType
, @LowCharacter  HighLowCharType
, @HighCharacter HighLowCharType;

-- A session context is created so session variables can be used.
EXEC InitSessionContextSp
  @ContextName = 'Rpt_UserTaskSp'
, @SessionID   = @RptSessionID OUTPUT;

-- Set the low and high values used for defaulting.
SET @LowDate       = dbo.LowDate();
SET @LowCharacter  = dbo.LowCharacter();
SET @HighCharacter = dbo.HighCharacter();

, @RemindDateTimeStarting       DateTimeType       = NULL
, @RemindDateTimeEnding         DateTimeType       = NULL
, @RemindDateTimeStartingOffset DateOffsetType     = NULL
, @RemindDateTimeEndingOffset   DateOffsetType     = NULL
) AS

-- Transaction management.
BEGIN TRANSACTION
SET XACT_ABORT ON

-- Set the isolation level specified for the background task
-- or use the system default.
IF dbo.GetIsolationLevel(N'UserTaskReport') = N'COMMITTED'
   SET TRANSACTION ISOLATION LEVEL READ COMMITTED
ELSE
   SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

DECLARE
  @RptSessionID  RowPointerType
, @LowDate       DateType
, @LowCharacter  HighLowCharType
, @HighCharacter HighLowCharType;

-- A session context is created so session variables can be used.
EXEC InitSessionContextSp
  @ContextName = 'Rpt_UserTaskSp'
, @SessionID   = @RptSessionID OUTPUT;

-- Set the low and high values used for defaulting.
SET @LowDate       = dbo.LowDate();
SET @LowCharacter  = dbo.LowCharacter();
SET @HighCharacter = dbo.HighCharacter();

-- Declare variables used to create the temp table.
DECLARE
  @UserId          TokenType
, @Username        UsernameType
, @TaskName        MessageSubjectType
, @RemindDateTime  DateTimeType
, @TaskDescription NoteType
, @RowPointer      RowPointerType;

-- Create an empty temp table for the report output.
SELECT
  @UserId          AS UserId
, @Username        AS Username
, @TaskName        AS TaskName
, @RemindDateTime  AS RemindDateTime
, @TaskDescription AS TaskDescription
, @RowPointer      AS RowPointer
INTO #ReportOutput
WHERE 1=0;

-- Insert data into the temp table.
INSERT INTO #ReportOutput
SELECT
  t.UserId
, n.Username
, t.TaskName
, t.RemindDateTime
, t.TaskDescription
, t.RowPointer
FROM UserTask t
INNER JOIN UserNames n ON t.UserId = n.UserId
```

```
WHERE n.Username BETWEEN @UsernameStarting AND @UsernameEnding
AND   t.TaskName BETWEEN @TaskNameStarting AND @TaskNameEnding
AND   ISNULL(t.RemindDateTime, @LowDate) BETWEEN
@RemindDateTimeStarting AND @RemindDateTimeEnding;

-- Return the report data.
SELECT
  UserId
, Username
, TaskName
, RemindDateTime
, TaskDescription
, RowPointer
FROM #ReportOutput
ORDER BY Username, RemindDateTime, TaskName;

COMMIT TRANSACTION
EXEC CloseSessionContextSp @SessionID = @RptSessionID;
GO
```

4 Click **Execute** and verify that the stored procedure executes successfully.

5 Save the stored procedure with the name **Rpt_UserTaskSp.sql**.

Setting data source properties for a stored procedure-based report

Before you can produce an SSRS report, you must have a source defined for the data to populate the report fields. The procedure to do this is different, depending on whether you are creating a report based on a stored procedure or on an IDO.

To set data source properties for a stored procedure-based SSRS report:

1 In Visual Studio, locate the **Data Source** option in the **Report Data** panel.

   If the **Report Data** panel is not displayed, you can view it by selecting **View** menu **> Report Data**.

2 Expand the **Data Source** option and locate or create the data source for your environment.

   Typically, this data source has the same name as the application. If you see the data source you are looking for, double-click the name of the data source and go on to substep c.

   If you do not see the data source you are looking for, you must create it, using substeps a and b, before going on to substep c:

   a In the **Report Data** panel, right-click **Data Sources** and select **Add Data Source**.

   b In the Data Source Properties dialog box, specify a **Name** for the data source (for example, **UserTaskDemo**).

   c In the **Type** field, specify **Microsoft SQL Server**.

   d (Optional, but strongly recommended) On the **Credentials** tab, specify user login credentials for your application database in the **Use this user name and password** option.

   Make sure you enter your Mongoose login credentials and not the SQL Server login credentials, because this is for an IDO-based report. This allows you to preview report layouts without having to enter login credentials every time.

Later, you will change this.

e Click **OK**.

The **Report Data** panel should now display the data source you just specified, if it did not already.

3 Add and assign the dataset:

a Right-click the name of your data source and, from the context menu, select **Add Dataset**.

b In the Dataset Properties dialog box, **Name** field, specify an appropriate name for your dataset (for example, **UserTaskDataset**).

c Verify that the **Use a dataset embedded in my report** option is selected.

d Verify that the data source defined in Step 2 is specified in the **Data Source** field.

e For the **Query type** option, select **Stored Procedure**.

f In the **Select or enter stored procedure** name field, specify the name of the stored procedure you created (for example, **Rpt_UserTaskSp**).

g Click **Refresh Fields**.

Visual Studio displays a list of automatically created query parameters in the Define Query Parameters dialog box.

h Verify that all **Parameter Value** fields specify **<Null>**.

i Click **OK**.

j In the Dataset Properties dialog box, click **OK**.

The **Report Data** panel displays the new dataset under the **Datasets** entry.

k In the **Properties** panel field drop-down list (at the top of the panel), select **Tablix 1**.

l In the **General > Dataset Name** field, specify the dataset you just created.

Modifying SSRS report parameters

To make sure that the report form that is to be created later can make use of the various report parameters correctly, you must make sure that the parameters are defined with the correct data types, properties, and default values, if any.

To modify the SSRS report parameters as required for the desired output:

1 In the **Report Data** panel, double-click **Parameters** to expand the list of parameters.

2 Double-click each of the parameters specified in this table and modify the values appropriately, using the Report Parameter Properties dialog box for each parameter:

Name	Prompt	Data Type	Allow Null	Default Value
StartingUsername	Start with user	Text	Selected	(null)
EndingUsername	End with user	Text	Selected	(null)
StartTaskName	Starting task	Text	Selected	(null)
EndTaskName	Ending task	Text	Selected	(null)
StartReminder	Start reminder	Date/Time	Selected	(null)

Name	Prompt	Data Type	Allow Null	Default Value
EndReminder	End reminder	Date/Time	Selected	(null)
StartReminderOffset	Start reminder offset	Integer	Selected	(null)
EndReminderOffset	End reminder offset	Integer	Selected	(null)

Note: For the **Default Value**, you must select the **Default Values** tab and then select **Specify values**. Click **Add** and then set the **Value** field to **(null)**.

Setting an SSRS report title

As it exists at this point, the report title for our example refers to a non-existent report parameter called **ReportName**. We must change this to refer to a parameter that will correctly define the report title we want to use.

Note: Before doing this procedure, it would be a good idea to create the name string (**fUserTaskReport**) for the new report title. This allows you, in the Expressions Editor, to select the string from a list. To do this, while in Design Mode, from the **Edit** menu, select **String** and use the Strings dialog box to create the string.

To set the SSRS report title parameter:

1 Change the report title element reference:

a In the**UserTask.rdl**[**Design**] workspace view, click to select the placeholder element in the upper left corner of the report.

This placeholder element is displayed as **[@ReportName]**.

b In the **Properties** sheet for this element, locate the **General** > **Value** field, which should display as: **=Parameters!ReportName.Value**

c Change this value to: **=Parameters!fUserTaskReport.Value**

The report title now displays in the **Design** workspace view as **[@fUserTaskReport]**.

2 Create the report parameter to associate with the reference:

a In the **Report Data** panel, right-click **Parameters** and then select **Add Parameter**.

b In the Report Parameter Properties dialog box, set the general properties like this:

- **Name: fUserTaskReport**
- **Prompt: fUserTaskReport_**

  Note: The trailing underscore (_) is required for translatable string parameters.

- **Data type:** Leave as **Text**

c On the **Default Values** tab, select **Specify values**.

d Click **Add**.

e In the **Value** field, specify **fUserTaskReport**.

f Click **OK**.

Visual Studio adds fUserTaskReport to the list of available **Parameters**.

Testing the SSRS report configuration

At this point in the SSRS report creation process, you should be able to test the report configuration, to make sure that the output is going to have the layout configuration you want. You will not yet be able to test the data, only the configuration.

To test the SSRS report configuration:

1. In Visual Studio, inside the workspace area, click the **Preview** tab for the report.
2. If prompted for the database login credentials, supply the user ID and password.

   Visual Studio renders a preview of the report without any data populating the various fields. At this point, the variable elements, such as the report name still appear as variables.

Populating an SSRS report with data

Previewing the report at this point confirms only that it is configured to process correctly. But you still have not populated it with any data. To ensure that the report is properly populated with data at the time of report generation, you can check to verify that the data is being properly queried.

To verify that the SSRS report query is correct:

1. In the **Report Data** panel, double-click the dataset you created earlier.
2. In the Dataset Properties dialog box, click **Query Designer**.
3. In the Query Designer dialog box, click the exclamation mark icon to query the data from the database.
4. In the Define Query Parameters dialog box, if the **Parameter Values** column displays **<Blank>** for any or all parameters, change them to **<Null>**.
5. Click **OK**.
6. Verify that the query returned the expected data from the UserTask table.
7. Click **OK** repeatedly to exit all dialog boxes.

   Note: If you have not yet created any user tasks, there is no data with which to populate this display. If this is the case, the query simply returns a single row containing the column headers. To see actual data displayed, first create some data using the Task List form and then run the query again.

Resetting the data source connection string

In an earlier procedure of the process, you should have changed the "connection string" for the data source to a hard-coded value for testing purposes. Before you can run the report successfully, though, you must reset the connection string back to its original value.

To reset the data source connection string:

1. In Visual Studio, locate and double-click the name of your data source.

2 In the Data Source Properties dialog box, **General** tab, change the hard-coded value in the **Connection string** field to:

- **[@pConnectionString]** for a report based on a stored procedure
- **[@pConnectionStringIDO]** for a report based on an IDO

You can use the Expression Editor to do this.

3 Select the **Credentials** tab and then select the **Prompt for credentials** option.

4 In the **Enter prompt text** field, enter some text to use as a prompt.

Note: This text never really displays anywhere in our system, but without something in this field, Visual Studio does not let you save your changes to the data source.

Deploying an SSRS report to the Report Server

The final step in the creation of an SSRS report is to deploy the report definition file to the Report Server from which it is to be called and run.

To deploy the report definition file:

1 Verify that your project is set up to use the correct deployment settings:

a In the Solution Explorer, right-click the name of your project.

(Be aware that this is not the *.rdl file, but rather the top-level project entry.)

b From the context menu, select **Properties**.

c In the Project Property Pages dialog box, verify that you are using the correct settings.

If necessary, change those that are not set correctly. The fields that you are particularly concerned with are:

- **TargetReportFolder –** Set this to the name of the report folder on your report server in which report definition files are deployed.
- **TargetServerURL –** Using a standard HTTP path, specify the name and location of the report server to which reports are deployed.

  Typically, this follows the format: **http:// serverName/ReportServer**, where *serverName* is the DNS or IP address of your Report Server.
- **TargerServerVersion –** Verify that this is SQL Server 2008 R2. Other versions do not work.

2 Deploy the report definition file:

a Right-click the name of your RDL file.

b From the context menu, select **Deploy**.

Example: Creating an SSRS report based on an IDO

Note: This example procedure is not guaranteed to be current or accurate with the SSRS tools and processes now available. Since SSRS reports are now deprecated in Mongoose-based applications, we are including this and related topics only for legacy support purposes.

This example, which you can also use as a short tutorial, creates a report based on an Intelligent Data Object (IDO) that displays data from a "UserTask" table. This simple report will be designed to list tasks, grouped by user.

This example assumes that you are using Visual Studio 2008 or SQL Server Business Intelligence Development Studio. Note that the steps would be very similar if you were to use Report Builder 3.0.

Although this example provides the procedural steps for using these tools, this is not to be considered a tutorial on how to use these tools. For information on how best to use these tools, see their respective product documentation.

To create an SSRS report based on an IDO, perform these procedures:

1. Create the report project.

   See "Creating an SSRS report project" on page 30.

2. Create the report definition file.

   See "Creating an SSRS report definition file from a template" on page 31 or "Creating an SSRS report definition file without a template" on page 31.

3. Create a connection string parameter.

   See "Creating a connection string parameter for an SSRS report" on page 43.

4. Set the data source properties.

   See "Setting data source properties for an IDO-based report" on page 43.

5. Modify the report parameters.

   See "Modifying SSRS report parameters" on page 36.

6. Set the report title.

   See "Setting an SSRS report title" on page 37.

7. Test the report configuration.

   See "Testing the SSRS report configuration" on page 38.

8. Populate the report with data.

   See "Populating an SSRS report with data" on page 38.

9. Configure the report layout.

   This includes such considerations as these:

   - Adding fields and labels to the report to help with the data being displayed
   - Formatting backgrounds of report sections
   - Formatting the fonts and font colors being used to display the data
   - Adding images, such as corporate logos

10. Reset the data source connection string.

See "Resetting the data source connection string" on page 38.

11 Deploy the report to the report server.

See "Deploying an SSRS report to the Report Server" on page 39.

Although the SSRS report definition file has been created and deployed, you still cannot generate the report, either manually or automatically. Before you can do that, you must:

- Create the report criteria form that allows users to specify the data they want to include in the report output.
- If you want to have the report generated automatically, create a background task definition and schedule a background task.

Creating an SSRS report project

To create an SSRS report project:

1. Run Visual Studio 2008 or SQL Server Business Intelligence Development Studio.
2. From the **File** menu select **New > Project**.
3. In the New Project dialog box, select:
    - **Project types: Business Intelligence Projects**
    - **Templates: Report Server Project**
4. For the **Name**, specify:
    - For a stored procedure-based report, **ReportSPDemo**
    - For an IDO-based report, **ReportIDODemo**
5. For the **Location**, specify **C:\Reports**.
6. Click **OK**.

Visual Studio creates the report definition project with a set of default folders.

Report definition files

About report definition files

The foundational phase of creating an SSRS report is the creation of the report definition file. Report definition files are special XML files that contain all the data and formatting specifications for a named report that can be generated. The system uses this file, which must have the extension ".rdl", to generate and render reports when requested.

Report definition files are created using tools provided by Microsoft. It is beyond the scope of this topic to specify and explain those tools. We recommend that you search online for the current set of report building tools from Microsoft.

It is possible to create a report definition from scratch but it is much easier if you begin with one of the report definition templates provided by Infor. These report templates are typically available in this directory:

C:\Program Files (x86)\Infor\Mongoose-based application\Report\Reports\Templates

where *Mongoose-based application* is the name of your Mongoose-based application.

The report templates can also be obtained from the Infor Xtreme Support site. They are contained in a zip file named **SSRS_Report_Templates.zip**, which you can download and unzip to the directory specified above. To find the zip file, you can search on "SSRS Report Templates". This and related topics discuss only the techniques for creating report definition files using these templates.

Creating an SSRS report definition file without a template

Note: For the working examples, we use and modify one of the report templates Infor provides. The procedure in this topic is included only to provide an example of how to create a report definition file without using a template. If you are using the working example as a tutorial, you need not complete this procedure.

To create an SSRS report definition file without a template:

1. With the report project open in Visual Studio, in the Solution Explorer, right-click **Reports** and, from the context menu, select **Add > New Item**.
2. In the Add New Item dialog box, **Templates** field, select **Report**.
3. In the **Name** field, specify a name for your report definition file, and then click **Add**.

Visual Studio creates the RDL file and displays a **Design** workspace in which to design and build your report definition.

Once the basic report definition file has been created, you can proceed to design the report output with the items, components, and data available in Visual Studio 2008.

Creating an SSRS report definition file from a template

To create an SSRS report definition file from a template:

1. Locate the template you want to use as the basis for your report.

   In these examples, we use these files:

   - **Template_Report_Landscape.rdl** for a stored procedure-based report
   - **Template_Report_Landscape_IDO.rdl** for an IDO-based report.

2. Copy the template file into your report project folder.
3. Rename your copy of the template file as either **UserTaskSP.rdl** or **UserTaskIDO.rdl** (for the stored procedure report or the IDO report, respectively).
4. In Visual Studio, in the Solution Explorer, right-click **Reports**.
5. From the context menu, select **Add > Existing Item**.
6. Select your *.rdl file and then click **Add**.

   Visual Studio adds your *.rdl file to the list of reports.

7. To open your *.rdl file, double-click its name in the Solution Explorer.

Visual Studio displays a workspace with the report definition file in Design Mode.

Notice that the report template already has report headers and footers configured. Many of the entries in the headers and footers are preconfigured to pick up variable values from within the system and need not be modified.

Creating a connection string parameter for an SSRS report

Before you can connect properly to a data source, you must have an appropriate **Connection String** parameter defined. This parameter should be defined as a “hidden” parameter.

To define this connection string parameter:

1 In Visual Studio 2008, in the **Report Data** panel, right-click **Parameters** and select **Add Parameter**.

2 In the Report Parameter Properties dialog box, specify these values:

- **Name: pConnectionStringIDO**
- **Prompt: pConnectionStringIDO**
- **Allow null values:** Cleared
- **Select parameter visibility: Hidden**
- **Default Values** tab, **Specify values:** Add a default connection string using this format:

  ConfigServerURL;ConfigName

  where:

  - *ConfigServerURL* is the standard URL-formatted path to the configuration server’s default page (for example, **http://MyServer/IDORequestService/ConfigServer.aspx**)
  - *ConfigName* is the name of the configuration to which you want to connect (for example, **MyMongoose**)

  Note: Take special notice of the semicolon between the URL and the configuration name.

Setting data source properties for an IDO-based report

Before you can produce an SSRS report, you must have a source defined for the data to populate the report fields. The procedure to do this is different, depending on whether you are creating a report based on a stored procedure or on an IDO.

To set data source properties for an IDO-based SSRS report:

1 In Visual Studio, locate the **Data Source** option in the **Report Data** panel.

If the **Report Data** panel is not displayed, you can view it by selecting **View** menu **> Report Data**.

2 Expand the **Data Source** option and locate or create the data source for your environment.

Typically, this data source has the same name as the application. If you see the data source you are looking for, double-click the name of the data source and go on to substep c.

If you do not see the data source you are looking for, you must create it, using substeps a and b, before going on to substep c:

a In the **Report Data** panel, right-click **Data Sources** and select **Add Data Source**.

b In the Data Source Properties dialog box, specify a **Name** for the data source (for example, **UserTaskDemo**).

c In the **Type** field, specify the name of the IDO.

Below the **Type** field, you should see a field labeled **Connection string**. The default value in that field might be blank or it might be: **[@pConnectionString]**

d Change the default value in the **Connection string** field, temporarily, to use this format:

Data Source=*serverName*;Initial Catalog=*applicationDatabaseName*

Note: You can use the **Edit** button to open a dialog box that simplifies the creation of this string.

You are changing the value of this string only temporarily, until you are done testing your report. Remember to change this string value back when you are done testing!

You will change this later to reference the pConnectionStringIDO parameter you created earlier. This allows it to work in WinStudio.

e (Optional, but strongly recommended) On the **Credentials** tab, specify user login credentials for your application database in the **Use this user name and password** option.

Make sure you enter your Mongoose login credentials and not the SQL Server login credentials, because this is for an IDO-based report. This allows you to preview report layouts without having to enter login credentials every time.

Later, you will change this.

f Click **OK**.

The **Report Data** panel should now display the data source you just specified, if it did not already.

3 Add and assign the dataset:

a Right-click the name of your data source and, from the context menu, select **Add Dataset**.

b In the Dataset Properties dialog box, **Name** field, specify an appropriate name for your dataset (for example, **UserTaskDataset**).

c Verify that the **Use a dataset embedded in my report** option is selected.

d Verify that the data source defined in Step 2 is specified in the **Data Source** field.

e For the **Query type** option, select **Text**.

f Click **Query Designer**.

In the Query Designer dialog box, verify that the IDO Name, Properties, and other fields are displayed. If not, click **Edit as Text** to toggle the view for these fields.

g From the **IDO Name** drop-down list, select the name of the IDO you are basing your report on (for example, **UserTasks**.

h From the **Properties** list, select the properties you want to include the data for in your report.

i In the **Filter** field, enter this or similar code:

```
Username BETWEEN ISNULL(@UsernameStarting,Username) AND
ISNULL(@UsernameEnding, Username)
AND TaskName BETWEEN ISNULL(@TaskNameStarting,TaskName) AND
ISNULL(@TaskNameEnding,TaskName)
AND ISNULL(RemindDateTime, '1753-01-01') BETWEEN
```

```
ISNULL(@RemindDateTimeStarting,'1753-01-01') AND
ISNULL(@RemindDateTimeEnding,'9999-12-31')
```

j Optionally, use the **OrderBy** field to specify what properties should be used to order and present the data in the report.

k To view your query in text mode, click **Edit as Text**.

l To execute and verify the query, click the Run (**!**) button.

Visual Studio displays a list of the selected query parameters in the Define Query Parameters dialog box.

m Verify that all **Parameter Value** fields specify **<Null>**, changing them if necessary.

n Click **OK**.

o In the Query Designer dialog box, click **OK**.

p In the Dataset Properties dialog box, click **OK**.

The **Report Data** panel displays the new dataset under the **Datasets** entry.

Note: Verify that the dataset you specified is the only one in the project. If there are others, delete them.

q In the **Properties** panel field drop-down list (at the top of the panel), select **Tablix 1**.

Note: The “Tablix” area is where the data populates the report when generated. For more information about a tablix and other report elements, see the Visual Studio 2008 documentation.

r In the **General** > **Dataset Name** field, specify the dataset you just created.

s Save the project.

Modifying SSRS report parameters

To make sure that the report form that is to be created later can make use of the various report parameters correctly, you must make sure that the parameters are defined with the correct data types, properties, and default values, if any.

To modify the SSRS report parameters as required for the desired output:

1 In the **Report Data** panel, double-click **Parameters** to expand the list of parameters.

2 Double-click each of the parameters specified in this table and modify the values appropriately, using the Report Parameter Properties dialog box for each parameter:

Name	Prompt	Data Type	Allow Null	Default Value
StartingUsername	Start with user	Text	Selected	(null)
EndingUsername	End with user	Text	Selected	(null)
StartTaskName	Starting task	Text	Selected	(null)
EndTaskName	Ending task	Text	Selected	(null)
StartReminder	Start reminder	Date/Time	Selected	(null)

Name	Prompt	Data Type	Allow Null	Default Value
EndReminder	End reminder	Date/Time	Selected	(null)
StartReminderOffset	Start reminder offset	Integer	Selected	(null)
EndReminderOffset	End reminder offset	Integer	Selected	(null)

Note: For the **Default Value**, you must select the **Default Values** tab and then select **Specify values**. Click **Add** and then set the **Value** field to **(null)**.

Setting an SSRS report title

As it exists at this point, the report title for our example refers to a non-existent report parameter called **ReportName**. We must change this to refer to a parameter that will correctly define the report title we want to use.

Note: Before doing this procedure, it would be a good idea to create the name string (**fUserTaskReport**) for the new report title. This allows you, in the Expressions Editor, to select the string from a list. To do this, while in Design Mode, from the **Edit** menu, select **String** and use the Strings dialog box to create the string.

To set the SSRS report title parameter:

1 Change the report title element reference:

a In the**UserTask.rdl**[**Design**] workspace view, click to select the placeholder element in the upper left corner of the report.

This placeholder element is displayed as **[@ReportName]**.

b In the **Properties** sheet for this element, locate the **General** > **Value** field, which should display as: **=Parameters!ReportName.Value**

c Change this value to: **=Parameters!fUserTaskReport.Value**

The report title now displays in the **Design** workspace view as **[@fUserTaskReport]**.

2 Create the report parameter to associate with the reference:

a In the **Report Data** panel, right-click **Parameters** and then select **Add Parameter**.

b In the Report Parameter Properties dialog box, set the general properties like this:

- **Name: fUserTaskReport**
- **Prompt: fUserTaskReport_**

  Note: The trailing underscore (_) is required for translatable string parameters.

- **Data type:** Leave as **Text**

c On the **Default Values** tab, select **Specify values**.

d Click **Add**.

e In the **Value** field, specify **fUserTaskReport**.

f Click **OK**.

Visual Studio adds fUserTaskReport to the list of available **Parameters**.

Testing the SSRS report configuration

At this point in the SSRS report creation process, you should be able to test the report configuration, to make sure that the output is going to have the layout configuration you want. You will not yet be able to test the data, only the configuration.

To test the SSRS report configuration:

1. In Visual Studio, inside the workspace area, click the **Preview** tab for the report.
2. If prompted for the database login credentials, supply the user ID and password.

   Visual Studio renders a preview of the report without any data populating the various fields. At this point, the variable elements, such as the report name still appear as variables.

Populating an SSRS report with data

Previewing the report at this point confirms only that it is configured to process correctly. But you still have not populated it with any data. To ensure that the report is properly populated with data at the time of report generation, you can check to verify that the data is being properly queried.

To verify that the SSRS report query is correct:

1. In the **Report Data** panel, double-click the dataset you created earlier.
2. In the Dataset Properties dialog box, click **Query Designer**.
3. In the Query Designer dialog box, click the exclamation mark icon to query the data from the database.
4. In the Define Query Parameters dialog box, if the **Parameter Values** column displays **<Blank>** for any or all parameters, change them to **<Null>**.
5. Click **OK**.
6. Verify that the query returned the expected data from the UserTask table.
7. Click **OK** repeatedly to exit all dialog boxes.

   Note: If you have not yet created any user tasks, there is no data with which to populate this display. If this is the case, the query simply returns a single row containing the column headers. To see actual data displayed, first create some data using the Task List form and then run the query again.

Resetting the data source connection string

In an earlier procedure of the process, you should have changed the "connection string" for the data source to a hard-coded value for testing purposes. Before you can run the report successfully, though, you must reset the connection string back to its original value.

To reset the data source connection string:

1. In Visual Studio, locate and double-click the name of your data source.

2 In the Data Source Properties dialog box, **General** tab, change the hard-coded value in the **Connection string** field to:

- **[@pConnectionString]** for a report based on a stored procedure
- **[@pConnectionStringIDO]** for a report based on an IDO

You can use the Expression Editor to do this.

3 Select the **Credentials** tab and then select the **Prompt for credentials** option.

4 In the **Enter prompt text** field, enter some text to use as a prompt.

Note: This text never really displays anywhere in our system, but without something in this field, Visual Studio does not let you save your changes to the data source.

Deploying an SSRS report to the Report Server

The final step in the creation of an SSRS report is to deploy the report definition file to the Report Server from which it is to be called and run.

To deploy the report definition file:

1 Verify that your project is set up to use the correct deployment settings:

a In the Solution Explorer, right-click the name of your project.

(Be aware that this is not the *.rdl file, but rather the top-level project entry.)

b From the context menu, select **Properties**.

c In the Project Property Pages dialog box, verify that you are using the correct settings.

If necessary, change those that are not set correctly. The fields that you are particularly concerned with are:

- **TargetReportFolder –** Set this to the name of the report folder on your report server in which report definition files are deployed.
- **TargetServerURL –** Using a standard HTTP path, specify the name and location of the report server to which reports are deployed.

  Typically, this follows the format: **http:// serverName/ReportServer**, where *serverName* is the DNS or IP address of your Report Server.
- **TargerServerVersion –** Verify that this is SQL Server 2008 R2. Other versions do not work.

2 Deploy the report definition file:

a Right-click the name of your RDL file.

b From the context menu, select **Deploy**.

Report criteria forms

About report criteria forms

Report criteria forms are specialized forms that can be used to specify how reports are to be generated. Most reports are launched and generated from these report criteria forms.

On report criteria forms, typically, you can:

- Select what data is to be included in the generated report.
- Specify how the data is to be arranged and laid out in the report output.
- Preview the report output before you actually generate the report.
- Manually generate (print) the report.
- Set up the report to be scheduled for automatic and scheduled generation.

Report criteria forms are generally intended to be used by end users, such as system administrators or others who are responsible for generating reports.

These forms usually have several editable fields for the report’s parameters. These fields can include criteria such as date ranges; selection of who to include information about, such as customers or vendors; and selection of other specific criteria either to include or not include in the report output.

For a fairly typical example of a report criteria form, take a look at the User Authorization Report.

Report criteria forms are actually the "interface," as it were, between the report object itself and the report that actually gets generated. The report object itself might be a custom Report-template type form, a predefined DataView, or an SSRS report definition file.

No matter what the report object is, though, a report criteria form is required to use it most effectively.

In addition, report criteria forms can be designed in such a way that you can schedule them to be generated automatically, either a single time or on a recurring basis.

Creating a report criteria form

Report criteria forms are specialized forms that can be used to specify how reports are to be generated. This topic presents the basic process to create such a form for a report-template form or an SSRS report.

To create a report criteria form:

1. In Design Mode, launch the New Form Wizard and create a new **Build from Scratch** form.

   The name for the form should consist of the subject of the report, appended with "Report" to identify it as a report form. For example, you might have a report on user tasks named UserTasksReport.

   You need not select a data source for this form, because it is simply a form that passes report criteria parameters to the report object.

2 In the **Form** properties sheet, specify a **Caption** for the form.

3 Save the form.

4 Optionally, if you are using a source control system, check your new form in and then back out.

5 Create the report input fields and parameters that will determine what data is to be processed and included in the generated report.

   Although you can include virtually any type of component for this, there are several common types of components used on report forms. They include these:

   - To create ranges for possible return values, you can use pairs of combo boxes or date combo boxes, using one for the starting value and the other for the ending value. These are especially useful with DataView reports.
   - To create checklists of property values to include, you can use check boxes or possibly option (radio) buttons. These are more common with Report-template type forms.
   - It is also common to create "Reminder" date fields/ranges; options to increment the date when the report is generated; and options to display (or not display) the report header.

   It is beyond the scope of this topic to provide specific procedures for each of these options. To view procedures to create each of these options, however, see the Related Topics.

   Note: For each task parameter that you want to pass to the report object, you must include it as part of the BGTaskParms variable definition. The parameter specification depends on what type of parameter you are passing.

6 Create a form variable with these specifications:

   - **Name: BGTaskName**
   - **Value:** The name of the background task for this report, as defined on the Background Task Definitions form.
   - **Persistent: True**
   - **Global: False**

7 Create a form variable with these specifications:

   - **Name: BGTaskParms**
   - **Value:** The syntax to use in this field depends on the type of report object the criteria form is being created for.
     - If the report object is a Report-template type form, then use this syntax for each variable value:

       Note: You can group multiple variables into one SETVARVALUES declaration.

       SETVARVALUES(*targetVariable*=V(*sourceVariable*))

       where:

       - *targetVariable* is the name of the variable in the report object that is to receive the value of the parameter being passed.
       - *sourceVariable* is the name of the variable you are using in the report criteria form to send the value as a parameter to the report object.
     - If the report object is a DataView or an SSRS report definition file, then use this syntax for the variable values:

V(*variable1*),V(*variable2*),V(*variable3*),...

If the list of variables is for a DataView report, they must be listed in in the same sequence as they appear on the **Input Parameters** tab of the DataViews Setup form.

- **Persistent: True**
- **Global: False**

8 Create the **Print** button.

See "Creating a button to print a report" on page 54.

9 Save the form and close it.

10 Reopen the form and test the **Print** functionality.

Note: At this point, the entire contents of the template form should print, because you have not yet set up the associations between this form and the report object.

11 Create the **Preview** button.

See "Creating a button to preview a report" on page 55.

12 Save the form and close it.

13 Reopen the form and test the **Preview** functionality.

Note: At this point, the entire contents of the template form should preview, because you have not yet set up the associations between this form and the report object.

14 Make sure you go back to the report object and set up the associations between that form and this one.

The report criteria form is now ready to be used to manually generate reports. If you want or need the report to be generated automatically, probably on a recurring basis, you must perform additional steps to enable the report to be generated in the background.

Creating a set of range input fields

This topic provides the most common procedure to create a set of range input fields to include as options on a report criteria form. Variations on this procedure are possible. These options can be used to select a range of property or date values to include in a report's output.

Note: This procedure applies only to report criteria forms created in the Windows (smart) client. Because of significant differences, we do not recommend trying to perform this procedure in the web client Web Designer.

To create a set of range input fields:

1 Open the report criteria form and go into Design Mode.

2 Add a ComboBox component to the form and give it an appropriate **Name**.

Do not give the combo box a caption.

We recommend using a name that consists of the IDO property you are using to set the range, prepended with "Starting". For example, if you want to use the UserTasks property for the range, you would name this component **StartingUserTask**.

3 Bind the component to a variable that will be used to pass the parameter value:

   a Click the ellipsis (**…**) button associated with the **Data Source > Binding** property.

   b In the Edit Component Data Binding dialog box, for the **Type**, select **Variable**.

   c Click **Edit**.

   d In the Edit Variable Binding dialog box, from the **Variable** drop-down list, select the variable for the parameter you want to include.

   We recommend you name the variable the same as the component itself (for example, **StartingUserTask**).

   Note: If you have not already created the variable, you can type the name for the variable into the **Variable** field, and the variable is created when you exit the dialog box.

   e Click **OK** in each dialog box until you are returned to the form.

4 Repeat Steps 2 and 3 to create a second combo box component and variable, with the word "Ending" replacing the word "Starting".

For example (and following the examples so far), you would name this component and its variable **EndingUserTask**.

Note: You can also create a set of date range input fields, by using DateCombo components instead of ComboBox components.

5 If you are "stacking" the two combo boxes, one above the other, add these Static components:

- One above the two combo boxes, with a **Caption** that indicates what IDO property you are using to set the range; for example, **sUserTaskNumber** (which displays as **User Task Number**).
- One to the left of the top combo box, with a **Caption** of **sStarting** (**Starting**).
- One to the left of the bottom combo box, with a **Caption** of **sEnding** (**Ending**).

Note: This is only one possibility for the layout of the range combo boxes. If, for example, you are creating multiple sets of ranges, you could stack the "Starting" combo boxes in a column to the left and the "Ending" combo boxes in a column to the right. You would then use Static components to label the IDO properties to the left of each set of combo boxes, and label above the first column "Starting" and above the second column "Ending".

Creating a checklist of properties to include

This topic provides the most common procedure to create a checklist of IDO properties to include as options on a report criteria form. Variations on this procedure are possible. These options can then be used to select which properties to include in a report's output.

Note: This procedure applies only to report criteria forms created in the Windows (smart) client. Because of significant differences, we do not recommend trying to perform this procedure in the web client Web Designer.

To create the checklist, perform these steps for each IDO property you want to include as an option:

1 Open the report criteria form and go into Design Mode.

2 Add a CheckBox component to the form and give it an appropriate name and caption.

3 Bind the component to a variable that will be used to pass the parameter value:

  a Click the ellipsis (**…**) button associated with the **Data Source > Binding** property.

  b In the Edit Component Data Binding dialog box, for the **Type**, select **Variable**.

  c Click **Edit**.

  d In the Edit Variable Binding dialog box, from the **Variable** drop-down list, select the variable for the parameter you want to include.

  Note: If you have not already created the variable, you can type the name for the variable into the **Variable** field, and the variable is created when you exit the dialog box.

  e Set the **Initial Value** to either **0** (zero) or **1** (one).

  A zero here indicates that the check box is cleared. A one indicates that the check box is selected. You must indicate one value or the other here. Otherwise, if you are using the value of this check box on the report-template form, and you leave this field blank, the value of the variable is null, which can affect the way the variable value is used (or not).

  f Click **OK** in each dialog box until you are returned to the form.

4 Add the parameter/variable specification to the BGTaskParms variable.

5 Make sure that you set up the necessary associations for each check box on the report-template form.

Creating an Increment Date field for a report criteria form

Reports are often designed to be run periodically and automatically. For reports like these, there is an option to automatically increment the dates on each successive report run. The amount by which the dates are incremented is set on the Background Queue form.

To add an **Increment Date** field with the correct properties:

1 With the report criteria form open and in Design Mode, add a CheckBox component. Use these settings:

  - For the **Name** property, specify **DateFieldIncrement**, where *DateField* is the base name of your date fields.

    For example, if the base name of the date fields is **RemindDateTime**, with associated **StartingRemindDateTime** and **EndingRemindDateTime** components, then the name of this check box field should be **RemindDateTimeIncrement**.

  - For the **Component Class** property, select **DateIncrementVar**.

  - Bind it to a variable with the same name as the component (for example, **RemindDateTimeIncrement**. Give this variable an initial value of **0** (zero).

  Notice that, when you save the form, the **Increment Date** check box label displays automatically.

2 For each date field, create a hidden Edit field:

  a Name it **DateFieldOffset**, where *DateField* is the name of the starting date field.

  For example, for the **StartingRemindDateTime** date combo box, this field should be named **RemindDateTimeStartingOffset**.

b Set the **Behavior > Hidden** property to **True**.

c Bind it to a variable with the same name as the component (for example, **RemindDateTimeStartingOffset**), with no initial value.

d For the **Component Class**, select **DateOffsetVar**.

Note: If your system does not have the **DateOffsetVar** component class, you can instead set the **Data Type > Underlying Type** to **CHAR** and the **Length** to **50**.

Remember to do this step for both the **StartingRemindDateTime** and **EndingRemindDateTime** fields.

Creating a button to print a report

Report criteria forms are typically used to preview and print reports from a report object source. To generate (print) the report, you must have a **Print** button on the report criteria form.

To create a **Print** button for a report:

1 With the report criteria form open and in Design Mode, add a Button component with these specifications:

- **Name: PrintButton**
- **Caption: sPrint**

2 Create a form event and event handlers to generate the report:

a With the new **Print** button selected, select the **Events** property sheet.

b In the **Primary** field, specify **GenerateReport** as the form event name.

c Click the ellipsis (**…**) button for the **Primary** field.

d In the Event Handlers dialog box, click **New**.

e In the Event Handler Properties dialog box, specify a **Response > Type** of **Run Background Task**.

f In the **Response > Parameters** field, click the ellipsis button.

g Click **Type Specific Parameters**.

h In the Edit Background Task Name and Parms dialog box, specify these values:

- **Task Name: V(BGTaskName)**
- **Task Parms: V(BGTaskParms)**

i If you are planning to schedule the report for automatic generation later, specify these additional values:

- **Task Status: V(BGTaskStatus)**
- **Task Number: BGTaskNumber**

j In each dialog box, click **OK** until you have returned to the Event Handlers dialog box.

3 Repeat substeps 2d. through 2i. to create a second event handler with these specifications:

- **Response > Type: Run Script**

- **Response > Parameters: ReportSubmitted()**

 Note: This event simply runs a built-in script that lets you know when the report has been successfully generated. To have the system notify you when the report is not successfully generated, specify an error message in the **Error Message** field of the Event Handler Parms dialog box.

4 In the Event Handlers dialog box, click **OK**.

5 Save and close the report criteria form.

6 Reopen the form and test it to make sure the report prints.

 Note: At this point, the entire contents of the report source should print, because typically you have not yet set up the associations between this form and the report object.

Creating a button to preview a report

Report criteria forms are typically used to preview and print reports from a report object source. To preview the report, you must have a **Preview** button on the report criteria form.

To create a **Preview** button for a report:

1 With the report criteria form open and in Design Mode, add a Button component with these specifications:

- **Name: PreviewButton**
- **Caption: sPreview**

2 Create a form event and event handler to generate the report preview:

a With the new **Preview** button selected, select the **Events** property sheet.

b In the **Primary** field, specify **PreviewReport** as the form event name.

c Click the ellipsis (**…**) button for the **Primary** field.

d In the Event Handlers dialog box, click **New**.

e In the Event Handler Properties dialog box, specify a **Response > Type** of:

- **Print Preview** for a Report-template type form or an SSRS report definition file
- **Run Form** for a DataView report

f In the **Response > Parameters** field, click the ellipsis button.

g In the Event Handler Parms dialog box, for the **Error Message**, specify **mPrintPreviewError**.

h Click **Type Specific Parameters**.

i If you are creating the report criteria form for:

- A Report-template type form or an SSRS report, continue with substep **j.**
- a DataView report, skip substep **j.** and continue with substep **k.**

j In the Edit Background Task Name and Parms dialog box, specify these values:

- **Task Name: V(BGTaskName)**
- **Task Parms: V(BGTaskParms)**

Continue with substep **l.**

k In the Specify Form Run Options dialog box:

- **Form:** Select **SBDataViewResults**
- Click **Set Variables** and use the Edit Set Variable Values dialog box to set these variable values:
  - **DataViewName=nameOfDataView**

    where *nameOfDataView* is the name of the predefined DataView for which the report criteria form is being created.
  - **DataViewPromptForInputs=0**

    (This is optional, but if you do not set this variable value, and you submit the DataView report preview without any option specifications, you are prompted for the inputs when you click **Preview**.)
  - **InputParameter1=V(firstInputParameter)**

    where *firstInputParameter* is the name of the variable that corresponds with the first input parameter as set on the DataViews Setup form.
  - **InputParametern=V(nthInputParameter)**

    where *n* is the number of each succeeding input parameter and *nthInputParameter* is the name of the variable that corresponds with each succeeding input parameter as set on the DataViews Setup form.

l In each dialog box, click **OK** until you have returned to the form.

Designing a report to allow scheduling

Report criteria forms can be designed in such a way that the reports can be generated automatically, either on a one-time or a recurring basis. This, however, requires an additional procedure to be performed when creating them.

To allow a report criteria form to be used for scheduled report generation:

1 Open the report criteria form and go into Design Mode.

2 From the Toolbox, select the **MenuItem** component and then click inside the form area.

Note: When you add a menu item component to a form, it is, in effect, hidden; that is, there is no visible change to the form and the menu item component itself is not delineated on the form. If you lose focus on this type of component, to regain focus, you must select it from the **Component** drop-down list or from the Object Viewer.

3 Name the component **BackgroundQueueTaskMenuItem**.

4 For the **Caption**, specify **s&Background**.

5 Add a form event to generate the report in the background:

a With the menu item component selected, select the **Events** properties sheet.

b Click the ellipsis (**…**) button associated with the **Primary** property.

c In the Event Handlers dialog box, click **New**.

d In the Event Handler Properties dialog box, set the **Event** field to **RunBackgroundQueue**.

e For the **Response > Type**, select **Run Form as Modal Child**.

f Click the ellipsis button for **Response > Parameters**.

g Click **Type Specific Parameters**.

h In the Specify Form Run Options dialog box, from the **Form** drop-down list, select **BackgroundQueue**.

i Click **Set Variables**.

j In the Edit Set Variable Values dialog box, click **New**.

k In the Edit Set Variable Value Pair dialog box, specify these values:

- **Target** = **BGTaskName**
- **Value** = **V(BGTaskName)**

l Click **OK**.

m Repeat substeps j.-l. to create a second variable value pair with these value:

- **Target** = **RunTaskEvent**
- **Value** = **GenerateReport**

Note: "GenerateReport" is the name of the Run Background Task event that we want to schedule and be placed on the background queue. If you have been following the related topics, it should already have been created.

6 In each dialog box, click **OK**, until you have been returned to the Event Handlers dialog box.

7 If all event handlers for the form are not being displayed, click **Show All**.

8 Modify the GenerateReport event handler:

a Select the GenerateReport event handler that has the **Response Type** of **Run Background Task**.

b Click **Edit**.

c In the Event Handler Properties dialog box, click the ellipsis (**…**) button associated with the **Response > Paramters** field.

d In the Event Handler Parms dialog box, click **Type Specific Properties**.

e If the **Task Status** field is not already set, set it to **V(BGTaskStatus)**.

f If the **Task Number** field is not already set, set it to **BGTaskNumber**.

9 Click **OK** repeatedly until you are returned to the form.

10 Verify that the **Primary** field displays **RunBackgroundQueue**.

11 Save and close the report criteria form.

12 Refresh the metadata cache and then reopen the form.

13 Verify that the **Actions** menu now includes a **Background** option for this form.

14 Click the **Background** menu item and verify that the Background Queue form opens.

15 Test the automatic generation of the report by performing these actions:

- Click the **Once At** option and set the time to a few minutes from now.
- Click **OK**.

  If everything is set up correctly, you should get a "Report Submitted" confirmation message.
- After the scheduled time has elapsed, use the Background Task History form to verify that the report was generated at the correct time.

Testing the form using the background queue

Before you assign a regular schedule to a report, you should test the form to verify that it can in fact submit tasks to the background queue and that the background queue is able to subsequently process the tasks.

To submit and test a report task on the background queue:

1. Open a form that can submit a background task to TaskMan.
2. Select **Actions > Background**.
3. On the Background Queue form, the **Daily** tab, select the option **Occurs > Once**.
4. Set the task for a few minutes from the current time.
5. Click **OK**.
6. Save your work and refresh the metadata cache (**Ctrl+U**).
7. Using the Active Background Tasks form, verify that the task is placed on the queue, and has a status of WAITING until the specified time.
8. Verify that the report is generated at the appointed time.

Scheduling a report to run on a recurring basis

Once you have verified that a report can be added to the background queue and processed correctly, you can go ahead and schedule the report to run on a recurring basis.

To schedule a report to run on a recurring basis:

1. Open a form that can submit a background task to TaskMan.
2. Select **Actions > Background**.
3. On the Background Queue form, select the options that determine how often and when the report is to be run.
4. Click **OK**.

Modifying/Customizing reports

3

Modifying or customizing reports, in many cases, is much the same as modifying or customizing any other form. It depends, however, on what type of report you want to modify/customize and whether you want to modify the report object or the report criteria form.

Modifying the report object

As discussed in other, related topics, the type of report object you might need to modify depends on what type of report you are working with:

- The report object for a report based on a Report (template) type form is the report-template form itself. Making modifications to this type of report is as straightforward as modifying any other Mongoose form. All you need to do is open the form in Design Mode, make the changes you want, and save the form.

  When modifying this type of report object, we recommend that you make a copy of the original report-template form and modify that.
- The report object for a DataView report is a predefined DataView. Making modifications to this type of report is the same as modifying any other predefined DataView.
- The report object for an SSRS report is the report definition file.T This is by far the most challenging type of report to modify, as the report definition file involves so many parts and pieces. This might involve having to modify the stored procedure, an IDO or a method used by the IDO, or the background task used by the report.

Customizing a report criteria form

Generally, a report criteria form is the easiest part of a report to customize. This is because it typically involves just adding new criteria by which to filter the report output or to removew existing criteria.

Troubleshooting Reports

A

Several common problems can occur when users try to preview or print a report. These are some of the more common ones with possible solutions:

- Reports do not print.
  See "Troubleshooting: Reports do not print" on page 62.
- Scheduled reports do not run as scheduled.
  See "Troubleshooting: Scheduled reports do not run as scheduled" on page 63.
- Report outputs do not pick up header/footer variable values.
  See "Troubleshooting: Report outputs do not pick up header/footer variable values" on page 63.
- The report generates a "File not found" error message.
  See "Troubleshooting: File not found" on page 64.
- Users outside the network are not receiving forwarded reports.
  See "Troubleshooting: Users outside network not receiving forwarded reports" on page 66.
- Labels are not being replaced with Strings table values.
  See "Troubleshooting: Labels not being replaced with Strings table values" on page 67.
- No report output is found.
  See "Troubleshooting: No report output" on page 69.
- Reports fail intermittently with errors
  See "Troubleshooting: Intermittent failures with errors" on page 69.
- Reports fail with an error code.
  See "Troubleshooting: Reports fail with error code" on page 69.
- Notes do not print with a report.
  See "Troubleshooting: Notes do not print" on page 70.
- Report gets an Error 13 - Type Mismatch.
  See "Troubleshooting: Error 13 - Type Mismatch" on page 70.
- Report gets an Error 128 - Error Running Report.
  See "Troubleshooting: Error 128 - Error Running Report" on page 70.

- Report gets an Error 534: Error Detected by Database DLL.

  This is similar to and has the same possible solutions as a "File not found" error. See "Troubleshooting: File not found" on page 64.

- Running a report produces the error: "This field name is not known."

  See "Troubleshooting: This field name is not known" on page 71.

- No users are receiving email messages about background tasks (including reports).

  See "Troubleshooting: No users are receiving email messages" on page 71.

Where to find information about error messages?

You can find error messages or additional information about an error message in these places:

- Background Task History form: Error messages for reports and report previews are displayed in the **Error Message** field on this form.
- Error log for SSRS reports: If the SQL Server Reporting Service (SSRS) generated an error message, you can view the error log for additional information.

  The error log is located in the **\Reports\Errors\userID** subfolder, in the base installation directory on the server where TaskMan resides.

- Microsoft Event Viewer: TaskMan runs as a service under Windows and generates event messages that you can view in the Microsoft Event Viewer. Some common event messages are listed in the *System Administration Guide* for your system.

  If you are having problems with a background task, you can run TaskMan in debug mode, which generates additional messages. See the section on running Infor Framework TaskMan in debug mode in the system administration guide for your system.

Troubleshooting: Reports do not print

Symptoms

Run-time users cannot print reports.

Possible solutions

- Make sure the Infor Mongoose TaskMan service is started on the application (utility) server.
- Verify that TaskMan is set up as a Windows service that logs on as with a user account created to access printers.

- Verify that the user account configured as the owner of the Infor Mongoose TaskMan service has print privileges under Windows for the default printer, plus all printers listed on the Report Options form.
- Verify that a default printer has been configured for the server on which TaskMan resides.
- If the report being printed requires a printer other than the default printer on TaskMan's server, verify that the correct printer is defined for the report on the Report Options form.
- Verify that the user account that was used to run the Infor Mongoose TaskMan service is the same user account that was used to map the printers on the TaskMan machine.
- Verify that the printer was mapped using the printer name, not the share name.
- Use SQL Profiler to verify that only one instance of TaskMan is monitoring the application database. See the section on using SQL Profiler to trace TaskMan instances in the *System Administration Guide* for your system.

Troubleshooting: Scheduled reports do not run as scheduled

Symptoms

Tasks that have been scheduled to run as background tasks using the Background Queue form are submitted but do not get processed when scheduled. The report task status remains WAITING.

Possible solution

Verify that the SQL Server Agent service is running on the report server. If necessary, reset the properties of the SQL Service Agent service to start automatically.

Troubleshooting: Report outputs do not pick up header/footer variable values

Symptoms

The report output for predefined DataView reports or reports based on a supplied SSRS template do not display the values of one or more global variables in the header or footer. Instead, they display the name of the variable (for example, BG~COMPANYNAME~) or the space is blank.

Possible solutions

The system looks for the value to replace this variable from a field on the System Parameters form. If this field is null or missing, there is no value to pick up.

Note: In some Mongoose-based applications, the System Parameters form is known as the General Parameters form.

- If your application has a System Parameters form, check and verify that there is a **Company** or **Company Name** field and that it has a value specified. If the field is there, but there is no value, specify a value for the field. If the field is missing, perform these procedures:

  1. Use the Sql Columns form to add a column to the "parms" table.

     This column must be named **company**, set to the **nvarchar** data type, be **40** characters in length, and be nullable.

  2. Starting with the IDOs form, create a new IDO to extend and replace the **Parms** IDO.
  3. Create a new property for the IDO you just created. This property must be bound to the "par.company" column you created earlier.
  4. Modify the System Parameters form by adding an **Edit** field component bound to the "company" property.

     Make sure you also add a field label (Static component).

  5. After saving the System Parameters form and exiting Design Mode, enter a value into the new field and save it.
  6. Make sure your system is set to unload IDO metadata with global objects. Then clear the metadata cache (**Ctrl+U**) and restart the report form.

- If your application has a General Parameters form, check and verify that there is a value specified in the **Company** field on the **Address** tab.

Troubleshooting: File not found

Symptoms

The report generates a "File not found" error message.

Possible solutions

This error message usually indicates an error in the programming of the report. Check for messages generated when the task ran in the Background Task History form, or check the reports error log (*TaskMan_Path*\Report\Errors\userID). Either of these might give more information about the problem.

It is also possible that the computer responsible for running reports has not been configured properly.

There are a number of things that can cause this error:

- Check the Background Task History form to make sure there are no uninterpreted **V(…)** or **P(…)** keywords in the parameters.

  Finally, subreports are linked to the main report by parameters on the subreport. When Notes subreports are linked in, the ShowInternal, ShowExternal, Rowpointer, and pConnectionString subreport parameters must be linked back to the appropriate main report fields (typically, two main report parameters).

In addition to these four parameters, any other parameters used in the subreport must also be linked in the properties. Finally, the parameters linked in the properties should allow null values or have **Null** as their default values.

If this is not done, the system expects these values to be passed from the originating form. If the wrong types of values are passed, the system generates an error. If no values are passed, the report usually "hangs" when run through TaskMan, while the report engine tries to prompt for these values on the TaskMan machine.

- The parameters passed by the report form are in the wrong order or are of the wrong data type, which can produce a T-SQL error.

  For example, if a string or date parameter is plugged into an integer parameter, SQL generates an exception. Following the remedial steps in the next option catches these errors as well.

- A Transact-SQL error occurred in the stored procedure.

  To get a more meaningful error message for a T-SQL error, run the report stored procedure through the SQL Server Management Studio. The easiest way to do this is to print the report and copy the report parameters from the Background Task History form. Paste these in as the parameters for the report stored procedure in SQL Server Management Studio, making these changes:

  - Enclose character and date parameters in single quotes.
  - Replace empty parameters with **NULL**.

  If there is a T-SQL error, this will give a line number and error description. The most common SQL errors in report stored procedures are data truncation errors. Many report stored procedures use temporary tables created using SQL data types. These tables are often populated from variables declared with user-defined data types. The lengths of many of these UDFs have been increased (for example, address fields for an internationalized site ID). This can cause a SQL exception by executing INSERT or UPDATE statements against the temporary tables using these variables, or applying the CONVERT function to them.

- The report's stored procedure was changed, but the report definition file was not resynchronized with the stored procedure.
- The stored procedure makes a call to RaiseErrorSp or raiserrorsp. RaiseErrorSp and raiserrorsp are used to generate a SQL exception with a user-specified error message. Neither of these two stored procedures should be used in report stored procedures.
- The computer responsible for running reports must have:

  - Access to the TaskMan folder (where TaskMan.exe and RunReport.exe are installed) on the server where TaskMan is running; OR
  - The TaskMan URL, if previewing over the Internet.

  The easiest way to check this is to select **Run** from the Windows **Start** menu and try to run a report.

- In rare cases, this error occurs if the RunReport.exe has become corrupted.

  If this is the case, running the report through TaskMan produces an error code of -1,073,741,819 and the report fails.

Troubleshooting: Users outside network not receiving forwarded reports

Symptoms

Users and customers outside your network are not receiving automatically forwarded reports.

Possible solutions

- If only certain users are not receiving the reports, but others are, verify that report options are set correctly for those users, by performing these actions:
  1. On the Background Task Definitions form, click **Report Options**.
  2. For the report and user you are checking, verify that the **Email Notification** option is set to **Yes**.

     If **Attach Report** is set to **Yes**, you must also set **Email Notification** to **Yes**.
- If no users outside your network are getting the reports, verify that the Exchange Server through which the system routes email is set to relay email, by performing these actions:
  1. Open the Intranets form, **Reports/TaskMan** tab, and make note of the contents in these fields:
     - **SMTP Server**
     - **SMTP Server Port**
     - **SMTP From Email**
  2. Click **Start > Run**.
  3. In the **Open** field, specify **cmd**, and then click **OK**.

     The system displays a Windows Command Prompt window.
  4. At the command prompt, enter this command: **telnet**
  5. At the Microsoft Telnet command prompt, enter this command: **set localecho**
  6. At the prompt, enter one of these commands: **o SMTPserver SMTPport** or **open SMTPserver SMTPport**,

     where:
     - *SMTPserver* is the name of the SMTP server you collected from the **SMTP Server** field
     - *SMTPport* is the port number you collected from the **SMTP Server Port** field in Step 1.
  7. At the SMTP server prompt, enter the command: **hello**

     On some systems, this command might vary; for instance, you might have to enter the command as **HELO** or some other variation. The SMTP server responds with a message that ends with `Hello`, followed by the IP address.
  8. Enter the command: **mail from:<SMTPfrom>**

     where *SMTPfrom* is the email address you collected from the **SMTP From Email** field in Step 1.

     The SMTP server responds with a message ending in `...Sender OK`.
  9. Enter the command: **rcpt to:<recipient_email>**

where *recipient_email* is the email address of the user you are trying to relay the reports to.

- If the SMTP Exchange server is not set up to relay email, the server displays a message similar to this:

  ```
  550 5.7.1 Unable to relay for recipient_email
  ```

  If you see this type of message, configure your SMTP Exchange server to relay the email. For the procedure, see your Exchange server documentation.

- If the SMTP Exchange server is set up to relay email, the server displays a message similar to this:

  ```
  250 2.1.5 recipient_email
  ```

  If you see this type of message, continue with the next step.

10 Enter the command: **data**

11 Type a test message to send to the recipient.

12 When you are finished with your message, press the **Enter** key, followed by a period (**.**), followed by the **Enter** key again.

This terminates the message and queues it for sending.

13 To exit the SMTP server session, type **quit**, followed by any other key press.

14 To exit the Microsoft Telnet session, type **quit**

15 To exit the command prompt window, type: **exit**

16 Confirm with the recipient that the test message was received.

Note: If you receive the error "Could not open connection to the host, on port *port_#*," check the firewall and virus software on the mail server to see if they are blocking the email.

Troubleshooting: Labels not being replaced with Strings table values

Symptoms

Labels on report outputs are not correct or are not being translated correctly.

Possible solutions

- Verify that the report is configured to use the correct Strings table. It might be that the system has been incorrectly configured and the report cannot find the correct Strings table.

  To select the correct String table record on the Sites form, use the **Site Name** field on the System Parameters form. Make sure that, for this site, there is a value in the **Forms Database Name** field of the Sites form (in some versions, labeled as the **Strings Table Specification** field). If the forms database is on a different server than the application database, the value for this field should also indicate the linked server name, using this format: **server_name.Forms_database**

Note: In some applications, the Sites form is known as the Sites/Entities form, and the System Parameters form is known as the General Parameters form.

TaskMan then determines the proper Strings table name by searching the specified forms database for the Strings table associated with the current session.

- Find the Strings table value using a query.

  You can check this using the SQL Server Management Studio by running this query on the Application Database:

```
SELECT i.intranet_name, p.site, i.tm_path , s.strings_table
FROM parms p
     INNER JOIN site s ON s.site = p.site
     INNER JOIN intranet i ON i.intranet_name = s.intranet_name
```

  If this SELECT action does not return any rows, then some piece of initialization data required by TaskMan is missing.

- Validate the Strings table name using a query.

  To test the Strings table name using the SQL Server Management Studio, run this statement on the Application Database to return the number of strings in the Strings table:

```
DECLARE
     @StringSQL AS NVARCHAR(255),
     @FormsDB AS NVARCHAR(255),
     @OwnerPos INT

SELECT @FormsDB = s.strings_table
FROM parms p INNER JOIN site s ON s.site = p.site

SELECT @OwnerPos = CHARINDEX( N'.dbo.', @FormsDb)
IF @OwnerPos <= 0 SELECT @OwnerPos = CHARINDEX(N'..', @FormsDb)
IF @OwnerPos > 0 SELECT @FormsDb = LEFT(@FormsDb, @OwnerPos - 1)

SELECT @StringSQL = N'SELECT COUNT(*)FROM ' + @FormsDB + N'.dbo.' +
l.StringTableName
FROM LanguageIDs l
WHERE l.LanguageID = 1033

EXEC (@StringSQL)
```

  If the Strings table specification is invalid, this SQL code returns something similar to the following:

```
Server: Msg 208, Level 16, State 1, Line 1
Invalid object name 'String_Table_Specification'.
```

- Verify that you are using the correct version of TaskMan.

  One indicator that you might not be using the correct version of TaskMan is an error message that says "Invalid String Table Name" followed by a number.

Troubleshooting: No report output

Symptoms

Reports might be marked as **Started** and **Completed** in the Background Task History form, but there is no output in the TaskMan\Reports\OutputFiles folder.

Possible solutions

There are probably two or more instances of TaskMan monitoring the same database. To verify this, shut down the instance of TaskMan you think should be monitoring the database and resubmit the report. If the Background Task History record still gets updated, then there is at least one more instance of TaskMan monitoring the database. To find out if this is the case, launch the SQL Profiler against the database to see what hosts were polling.

Troubleshooting: Intermittent failures with errors

Symptoms

The same report is run a number of times, sometimes running successfully, and other times failing with errors. This happens for all reports. Whether it succeeds or fails appears to be random.

Possible solutions

This could occur because of the same issues described under the “No Report Output" topic.

Alternatively, there might be two instances of TaskMan: one has the privileges needed to run reports, but the other does not.

Troubleshooting: Reports fail with error code

Symptoms

Reports fail to process and return an error code of **1,073,741,819**.

Possible solutions

If you see this return code, an executable might have become corrupt. Try double-clicking on the RunReport.exe file. If it does not open dialog boxes, the executable files have been corrupted, possibly by a virus. Delete them and copy them again from the installation source.

Troubleshooting: Notes do not print

Symptoms

You print a report, but the notes do not print along with it.

Possible solution

Verify that notes are set up correctly to print with the report.

Troubleshooting: Error 13 - Type Mismatch

Symptoms

There is a data type mismatch between either what is passed from the form to the report file, or from the report file to the stored procedure.

Possible solutions

This can occur if the form passes the parameters in the wrong order, or a parameter is not defaulted correctly. For example, a form might be passing in a blank for the **Show Header** check box.

If you get this error, try selecting the check box and running it again.

If this does not work, try selecting every check box and putting a value in every field.

Finally, try running the stored procedure through the SQL Server Management Studio with the same parameters as those in BGTaskHistory, replacing blank parameters with NULL, and putting single quotes around strings, dates, and guids.

Troubleshooting: Error 128 - Error Running Report

Symptoms

Reports fail with the message: **Error running report. This field name is not known.**

Possible solutions

Verify that the report field was correctly mapped to the appropriate stored procedure field. The solution is similar to the solution for type mismatches.

For more information, see “Error 13: Type Mismatch."

Troubleshooting: This field name is not known

Symptoms

Running a report produces the error: "This field name is not known."

Possible solutions

Having a report field that was not successfully mapped to a stored procedure field, can sometimes cause this. In these cases, the unmapped stored procedure field will be part of the error message; for example:

```
This field name is not known : {Rpt_Ap01FRISp;1.TcAmtAmtPaid}
```

Make sure the report field is in the stored procedure’s result set, and set the datas ource location as described in the Error 534 topic.

Troubleshooting: No users are receiving email messages

Symptoms

No users are receiving email messages about background tasks (including reports).

Possible solutions

- Verify that TaskMan is set up as a Windows service that logs on as a user account. If TaskMan is configured as a Local System Account, email notification cannot be sent.
- Verify that the user account configured for the Infor Mongoose TaskMan service under Windows has email privileges in your mail system.
- On the Intranets form, verify that the **Send Email Notification** field is selected. TaskMan gets the value that is set here for the first database TaskMan is configured to use, and then uses that value for all databases to which it connects from the application (utility) server.

  If you do not know which database is the "first" database, you can select the **Send Email Notification** field for another database. Then, no email will be sent and no error messages appear in the event logs. You must either determine what the "first" database is and set the option there, or set the option in all databases. Conversely, to stop email notifications, either clear the option in the "first" database or clear the option in all databases.
- Verify that the recipient's user ID specifies an email address in the Users form.
- The system requires a default email profile, or a profile called "TaskMan," on the server where TaskMan resides. To verify this, if Microsoft Outlook is installed, open its properties (right-click on the Outlook icon and select **Properties**). Click **Show Profiles**. If there is no TaskMan profile, copy one that already exists, copy it, and rename it TaskMan.

About RunReport.exe

B

Infor Mongoose TaskMan launches a system process (RunReport.exe) to generate a report. The RunReport application then connects to the application database.

Once connected, RunReport queries any user options for running the report, for example, output format or printer name. Once these options are set, RunReport prints the report.

Generally, RunReport is called from the application; however, you can also run RunReport from a command line, specifying connection options, the report name, and report options. This could be used, for example, to run reports directly from an add-on product, or to run reports immediately from a workflow on a client (without having to put the report on a background queue) and then do something with the report's output in the workflow.

Keep in mind, however, that if you execute RunReport from a command line, you cannot take advantage of TaskMan's task history and management features.

Using RunReport.exe from a command line

Syntax

This is the syntax to run RunReport.exe from a command line:

```
RunReport.exe switches
```

where *switches* must be preceded by either a minus sign (-) or a forward slash (/).

If the **-m** switch is used, it must be the last switch. The order of the other switches does not matter.

Any switch that is used must be followed by the appropriate value. All switches, other than **-m**, require their values to be enclosed in parentheses if the value contains blank spaces. The value following **-m** must not be enclosed in parentheses.

Switches

These are the switches that can be used in the RunReport.exe command line:

Switch	Description
-h	(Optional) Use this switch to designate the TaskMan home directory, where taskman.exe runs. If this parameter is omitted, RunReport.exe uses its own directory.
-t	(Optional) Use this switch to designate the Task Number. For reports run through TaskMan, this is the **Task Num** field on the Active Background Tasks form, and the **Task Number** field on the Background Task History form. This number is generated by SQL when a record is inserted into the ActiveBGTasks table.
-r	(Required) Use this switch to specify the name of the report.
-d	Use this switch to designate the the DSN for the ODBC that RunReport.exe uses to connect to the database. Use this switch with the **-db** parameter if you are using either ODBC (RDO) or OLE DB (ADO) to connect to an ODBC data source to access the database. This parameter is ignored if you are using a direct OLE DB (ADO) connection to the SQL database.
-p	Use this switch to specify the password for the ODBC connection, or for the SQL database, if you are using a direct OLE DB (ADO) connection. This value defaults to an empty string.
-l	Use this switch to specify the user login for the SQL database, if you are using a direct OLE DB (ADO) connection. This value defaults to **sa**.
-u	(Optional) Use this switch to specify the user ID of the person submitting the report. This refers to the **User ID** field on the Users form. It is also displayed on the Background Task History form, for tasks submitted to TaskMan.
-db	Use this switch to specify the name of the Application Database. This is used to set the location for the main report and subreport data sources. If omitted, the data sources are left as is.
-g	(Optional) Use this switch to specify the User Group ID. If provided, this is used in a SQL WHERE clause when querying the Strings table. It is used to filter on the **ScopeName** (the user’s group) and the **ScopeType** (2, for Group).
-session	(Optional) Use this switch to designate the Session Identifier, passed when TaskMan calls RunReport.exe, to indicate the session for which variables can be accessed by the report's stored procedure.
-s	(Optional) Use this switch to specify the name of the Strings table used to translate report labels. RunReport.exe uses an ODBC connection to the application database to query the Strings table, so the table name must be qualified enough for the query to work.

Switch	Description
	If the Strings table is in a Forms Database on the same server as the Application Database, then the Strings table name must include that database name (for example, Formsdb.dbo.StringTable.) If the Strings table is in a database on a different server, then this server must be a linked server, and the Strings table name must include both the server name and the Forms Database name (for example, Server.Formsdb.dbo.StringTable). If this parameter is omitted, the report runs, but the labels are not translated and display as literal string names. In addition, a warning message is placed in the error log.
-f	(Optional) Use this switch to designate the integer code for the report output format: 1 - No longer used 2 - Acrobat Format (PDF) 3 - Comma Separated Values (CSV) 4 - Excel 8.0 (format used with Excel 97 and Excel 2000) 5 - Excel 8.0 extended (format used with Excel XP and Excel 2003) 6 - HTML 4.0 7 - Printer 8 - No longer used 9 - Word for Windows (DOC) 10 - XML 11 - No longer used 12 - MHTML 13 - TIFF
-o	(Optional) Use this switch to designate the output file name. If this is omitted, RunReport.exe uses this format: *HomeDir*\Report\OutputFiles\UserName\TaskName_Site_TaskNum.Extension where: • *HomeDir* is the root directory where reports are to be output. • *UserName* is the user ID of the individual or account from which the report is being generated. • *TaskName* is the name of the task as generated by RunReport.exe. • *Site* is the name of the site (if any). • *TaskNum* is the task number as generated by RunReport.exe. • *Extension* is the extension for the type of output being generated.
-n	(Optional) Use this switch to specify the number of copies to print, if the report output is to be sent directly to a printer (output format 7). The default value is set using the printer settings.

Switch	Description

-e	(Optional) Use this switch to specify the name of the error file that WinStudio generates for report errors.
	If a file name without a path is passed in, RunReport.exe prepends this to the path:
	`*HomeDir*\Report\Errors\*UserName*\`
	where:
	• *HomeDir* is the root directory where reports are to be output.
	• *UserName* is the user ID of the individual or account from which the report is being generated.
	If this parameter is omitted altogether, RunReport.exe uses the format:
	`*HomeDir*\Report\Errors\*UserName*\*TaskName*_*TaskNum*.txt`
	where:
	• *HomeDir* is the root directory where reports are to be output.
	• *UserName* is the user ID of the individual or account from which the report is being generated.
	• *TaskName* is the name of the task as generated by RunReport.exe.
	• *TaskNum* is the task number as generated by RunReport.exe.
	TaskMan puts the contents of this file in the **Error Message** field on the Background Task History form and then deletes the file.

-prt	(Optional) In cases where the output is to be sent directly to a printer, use this switch to designate the name of the printer. If the printer is not local, this value should be the UNC path/name. If omitted, this value defaults to the default printer of the machine where RunReport.exe is running.
-site	(Optional) Use this switch to specify the name of the site, from the Sites (or Sites/Entities) form. This value is used as part of the output file name. This parameter is used so that, if TaskMan is monitoring multiple databases, the report output from one database does not overwrite that from a different database.
-config	(Optional) Use this switch to designate the application configuration to which you are connected.
-svr	(Optional) Use this switch to designate the name of a SQL Server to override the SQL Server setting from the configuration that points to the application database.
-debug	(Optional) Use this switch to generate a debug/error file in the **Errors** directory (**Report/Errors**, rather than **Report/OutputFiles**).
-m	(Optional) Use this switch to indicate that the rest of the command line is a comma-delimited list of report-specific parameters. RunReport.exe parses this list and plugs these values into the report's parameters. This switch must be the last one used in the command line. If any other switches occur after this one, RunReport treats them as report parameters.

Switch	Description
	RunReport.exe supports the ~LIT~(...) keyword.

Examples: Command lines for reports

When TaskMan and MGCore generate reports, they format and then execute command lines like the ones in these examples.

If TaskMan is run with the debug startup parameter, this command line is printed to the Application Event log. Reports can be tested by putting this command line in a batch file and executing it. The **** -p parameter should be replaced by the appropriate password.

Example Using an ODBC (RDO) or OLE DB (ADO) ODBC Connection

```
RunReport.exe -d MyServerDSN -l sa -p **** -s
MyCompany_Forms.dbo.Strings -f 1
-r ItemQuantitiesbyABCCode -t 199 -u sa -e
"D:\TaskMan\Report\Errors\sa\ItemQuantitiesbyABCCodeReport_MI_199.txt"
 -site
"MI" -db SLMI_App -n 1 -m
~LIT~(A)~LIT~(B)~LIT~(C),~LIT~(M)~LIT~(P)~LIT~(T),~LIT~(M)~LIT~(T)~LIT~(F)~LIT
~(O),~LIT~(1),~LIT~(1),~LIT~(),~LIT~(DIS1),~LIT~(MAIN),~LIT~(AD-
10000),~LIT~(AL-10000),~LIT~(),~LIT~(),~LIT~(0)
```

Example Using an OLE DB (ADO) Direct SQL Connection

```
RunReport.exe -svr MyServer -l sa -p **** -s MyCompany_Forms.dbo.Strings
 -f 1 -
r ItemQuantitiesbyABCCode -t 199 -u sa -e
"D:\TaskMan\Report\Errors\sa\ItemQuantitiesbyABCCodeReport_MI_199.txt"
 -site
"MI" -db SLMI_App -n 1 -m
~LIT~(A)~LIT~(B)~LIT~(C),~LIT~(M)~LIT~(P)~LIT~(T),~LIT~(M)~LIT~(T)~LIT~(F)~LIT
~(O),~LIT~(1),~LIT~(1),~LIT~(),~LIT~(DIS1),~LIT~(MAIN),~LIT~(AD-
10000),~LIT~(AL-10000),~LIT~(),~LIT~(),~LIT~(0)
```

Templates for SSRS report definition files C

Infor provides these SSRS report definition templates, which help ensure uniform report layout when converting or creating reports.

- Template_Report_Landscape.rdl
- Template_Report_Landscape_IDO.rdl
- Template_Report_Portrait.rdl
- Template_Report_Portrait_IDO.rdl
- Template_SubReport_Landscape.rdl
- Template_SubReport_Portrait.rdl

Landscape report width is 10.5 inches and portrait report width is 7.77 inches. IDO report templates are for reports that retrieve data using an IDO, and subreport templates are for reports that are part of main reports.

Objects used in or by reports

D

Table 1: Executables

Executable	Description
TaskMan.exe	Runs as a service on the application (utility) server. It monitors the queue of tasks. It references MGReportProcessor to generate the task parameter (XML) file, which is passed to RunReport as its command line input parameter.
RunReport.exe	Used to launch reports. Can also be run from a command line.

Table 2: DLLs

DLL	Description
TMMsgs.dll	Used by TaskMan
MGCore.dll	Used by report previews
MGReportProcessor.dll	A .NET class library which contains the core reporting logic, which both TaskMan.exe and RunReport.exe reference.

Table 3: Tables/IDOs/Forms

Table	IDO	Form	Description
ActiveBGTasks	MGCore.ActiveBGTasks	Active Background Tasks	Queue of tasks.
n/a	n/a	Background Queue	Schedules tasks to run later.
BGTaskHistory	MGCore.BGTaskHistories	Background Task History	Created by a trigger on ActiveBGTasks, updated by TaskMan. Contains a record of when a task was submitted, started

Table	IDO	Form	Description
			and completed, plus any error messages.
BGTaskDefinitions	MGCore.BGTaskDefinitions	Background Task Definitions	Creates a record that identifies each background task to TaskMan.
ReportOptions	MGCore.ReportOptions	Report Options	Sets the output format for reports, and the UNC printer path and name if the report is sent directly to a printer.

Table 4: Stored procedures/COM methods

SP/COM method	Description
BGTaskSubmit	Used with Mongoose to submit reports to TaskMan. It returns the row pointer of BGTaskHistory (@TaskHistoryRowPointer) for a task being created.
SL.SLJobQueues.BackGroundQueueSP	Used with the Background Queue form.
MGCore.ActiveBGTasks.BackGroundQueueDeleteSP	Used with the Background Queue form.
CLM_GetBGTasksToProcessSp	TaskMan calls this at its polling time, querying the ActiveBGTasks table to retrieve background tasks to be processed.
GetTaskOptionsSp	Used to determine the report output directory, based on the Report Output Directory process default on the Process Defaults form or the **Output Directory** field on the Report Options form.
UpdateActiveBGTaskSp	While processing a requested task, TaskMan is to update the task status using the stored procedure.