Infor Enterprise Server Plug-in for Microsoft SQL Server Reporting Services Development Guide

Release 10.7.x

Important Notices

The material contained in this publication (including any supplementary information) constitutes and contains confidential and proprietary information of Infor.

By gaining access to the attached, you acknowledge and agree that the material (including any modification, translation or adaptation of the material) and all copyright, trade secrets and all other right, title and interest therein, are the sole property of Infor and that you shall not gain right, title or interest in the material (including any modification, translation or adaptation of the material) by virtue of your review thereof other than the non-exclusive right to use the material solely in connection with and the furtherance of your license and use of software made available to your company from Infor pursuant to a separate agreement, the terms of which separate agreement shall govern your use of this material and all supplemental related materials ("Purpose").

In addition, by accessing the enclosed material, you acknowledge and agree that you are required to maintain such material in strict confidence and that your use of such material is limited to the Purpose described above. Although Infor has taken due care to ensure that the material included in this publication is accurate and complete, Infor cannot warrant that the information contained in this publication is complete, does not contain typographical or other errors, or will meet your specific requirements. As such, Infor does not assume and hereby disclaims all liability, consequential or otherwise, for any loss or damage to any person or entity which is caused by or relates to errors or omissions in this publication (including any supplementary information), whether such errors or omissions result from negligence, accident or any other cause.

Without limitation, U.S. export control laws and other applicable export and import laws govern your use of this material and you will neither export or re-export, directly or indirectly, this material nor any related materials or supplemental information in violation of such laws, or use such materials for any purpose prohibited by such laws.

Trademark Acknowledgements

Publication Information

Release: Infor LN 10.7.x
Publication Date: October 1, 2020
Document code: ln_10.7.x_lnrptsqlserverdg__en-us

Contents

About this guide

This document describes how to install and configure the Infor ES plug-in for Microsoft SQL Server Reporting Services on your computer.

You can use this plug-in in Microsoft SQL Server Data Tools (SSDT).

SSDT is an Integrated Development Environment (IDE) and is part of Microsoft SQL Server 2012 and Microsoft SQL Server 2016. You can install SSDT as part of the Microsoft SQL Server installation. SSDT runs either as part of Visual Studio, or is installed as Visual Studio Integrated Shell if Visual Studio has not been installed.

The document also explains how to design reports for Infor LN.

Intended audience

This document is intended for developers that want to develop Infor LN reports for Microsoft SQL Server Reporting Services (SSRS).

Related documents

You can find the documents in the product documentation section of the Infor Support Portal, as described in "Contacting Infor".

- *Infor Enterprise Server Plug-in for Microsoft SSRS Administration Guide*
- *Infor LN - Development Tools Development Guide*

Contacting Infor

If you have questions about Infor products, go to Infor Concierge at https://concierge.infor.com/ and create a support incident.

The latest documentation is available from docs.infor.com or from the Infor Support Portal. To access documentation on the Infor Support Portal, select **Search > Browse Documentation**. We recommend that you check this portal periodically for updated documentation.

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

Chapter 1: Introduction

Microsoft SQL Server Reporting Services provides a complete server-based platform. This platform is designed to support a wide variety of reporting requirements, enabling organizations to deliver information across their entire enterprise.

The Infor ES plug-in for Microsoft SQL Server Reporting Services is a plug-in for Microsoft SQL Server Data Tools (SSDT). With this plug-in, you can use the power of SSRS in an Infor LN environment.

You can use the Infor ES Reporting plug-in to redesign LN session reports, in SSDT, as SSRS report designs. Therefore, you can give the reports a modern layout and use features such as images, indicators, and graphs.

You can deploy these reports to a central report server. To use a report design that is stored on the report server, LN users can print the corresponding LN report to a Microsoft Reporting Services device.

The Infor ES Reporting plug-in consists of these components:

- ES Data Extension

  This SSDT extension is used to design and preview Infor LN reports.
- ES Report Configurator

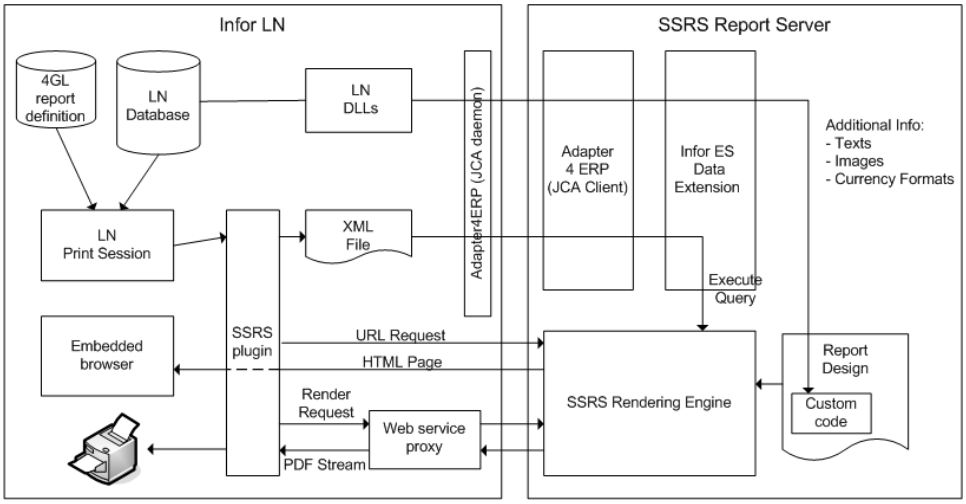
  This SSDT Add-in automates various steps when you create a report. Without this configurator, you must perform these steps manually.

Note: LN reports for SSRS must be developed in SSDT. Usually, reports for SSRS can be developed in SSDT and in the Stand Alone SQL Server Report Builder. The Report Builder does not support extensions, so you cannot create LN Datasources. Therefore, you cannot develop LN reports in the Report Builder.

Runtime architecture

This section describes the components of the integration between LN and Microsoft Reporting Services (SSRS).

This diagram shows the runtime architecture of the MS reporting solution for LN:



Note: Currently only push reporting from LN is supported.

The solution for LN contains these parts:

- LN Print Session / 4GL report definition / LN Database

  The print session in LN queries the LN database and sends the data to the 4GL report. The 4GL report definition contains the so-called report input fields. These fields are available for the Microsoft report.

- SSRS Plug-in

  The SSRS Plug-in is a set of components in the LN server that handles the printing of data by using a Microsoft report. The plug-in stores the available Report Servers and handles the flow of data when the user prints a Microsoft report. Also, the plug-in provides functions that can be called from the SSRS side to retrieve the (meta)data of the reports.

- XML file

  An XML file is created by using the data to be printed on the report. This file contains Report Properties, Labels, and Data Rows. This file is used as the datasource on the SSRS side.

- Embedded Browser / URL Request

  When the user views the report directly, a URL Request is built up and activated in the embedded browser of Web UI. SSRS then renders the report and the HTML is sent to the browser.

- Render Request / Web service proxy / Printer

  The user can also print the report directly. Therefore, a Web service renders the report in PDF format. This PDF file is sent to the printer.

- Report Design / Custom Code

  The report design is stored on the SSRS side in a SQL server database. This design specifies the datasource that is to be used.

- SSRS Rendering Engine

  The rendering engine processes URL requests or Web service requests. The engine runs the query, which must be defined in the report design. For the LN push reports, this query retrieves the XML file from the server by means of the Infor ES Data Extension.

- Infor ES Data Extension

The Infor ES Data Extension is an extension to the standard datasource that SSRS can handle. This data extension retrieves the XML file from the LN server, through the Adapter4ERP, and fills the internal dataset. SSRS uses this dataset to render the report.

- Adapter4ERP

  Adapter4ERP is a common LN component that is used to connect a .NET environment to LN. Adapter4ERP logs on to the server, starts a bshell, and runs BDE- and DLL-calls in LN.

- LN DLLs

  During the rendering of the report, it may be necessary to retrieve additional data from the server. You can retrieve additional data from custom code in the report.

Chapter 2: Installing the Infor ES plug-in for Microsoft SQL Server Reporting Services

This chapter describes how to install the Infor ES plug-in for Microsoft SQL Server Reporting Services (SSRS).

To design reports, you must install these components on your computer:

- Infor ES Data Extension
- Infor ES Report Configurator

  The Report Configurator is not strictly required.

  See Infor ES Report Configurator on page 47.

Prerequisites

Before you start the installation, ensure that these prerequisites are fulfilled:

- You must have access to a Microsoft SQL Server 2012, or a Microsoft SQL Server 2016 environment. SQL Server is usually installed on a central machine. Ask your administrator for details, such as the location of the database and the login credentials.
  Note: For development purposes, you can install a SQL server environment on your PC. The device that is set up in LN must refer to the SQL server. See the *Infor Enterprise Server Plug-in for Microsoft SSRS Administration Guide*.
- Microsoft SQL Server Data Tools (SSDT) must be installed on your PC. SSDT is used to create Report Server Reports. See the Microsoft documentation.
- If you use Microsoft SQL Server 2016, you must install these client tools:
  1. Visual Studio 2015
  2. SQL Server Data Tools (SSDT) 2015
  3. Visual Studio extensibility tools update 3
  4. Infor ES plug-in for Microsoft SQL Server Reporting Services

  You must install the tools in the indicated order.

Note:

- Follow the sizing guidelines for using Infor LN with the Microsoft Reporting Solution. See the Infor LN Sizing Guidelines Microsoft Reporting Solution.

- The Report Server supports Native or SharePoint Integrated mode with Windows Authentication. In future releases claims-based authentication, for example ADFS, will also be supported.

Installation

To install the Infor ES Reporting plug-in:

1. Download the Infor ES plug-in for Microsoft SQL Server Reporting Services installer.
2. Unzip the file and double-click the setup.exe file in the start folder.
3. The Infor Installation wizard starts.
4. On the Welcome screen, click **Next**.

During the installation process, several dialog boxes are displayed. The dialog boxes are self-explanatory. Specify the required information and click **Finish** to complete the installation.

Points of attention

- In the **Setup type** dialog box, specify **`Custom`**.
- On the **Custom Setup** page, select **`Development Environment`**.

  The Infor ES Data Extension and the Infor ES Report Configurator will be installed on your computer.

Chapter 3: Creating and modifying reports

You can create report designs that are based on existing reports in LN.

Creating a report design - procedure summary

To create and deploy a report design that is based on an LN report:

1. Generate a file with report metadata and runtime data.
2. Create a Microsoft Reporting report.
3. Configure the report.
4. Modify the layout of the report.
5. Add LN-specific features.
6. Preview the report.
7. Prepare the report for deployment.
8. Deploy the report to a Report Server.
9. Optionally, import the report in the LN dictionary.

See the following sections.

Note: This chapter describes how you can create a report design. To customize a deployed report design, you must first prepare the report for development.

Creating a report design - procedure details

To create and deploy a report design that is based on an LN report:

Step 1 - Generating a file with report metadata and runtime data

You must generate an XML file with report metadata and runtime data. This file is used to create a datasource with the fields, properties, and labels for the report.

To generate a file with report metadata and runtime data:

1. Log on to LN.
2. Start the session to which the LN report is linked.
3. Complete the session's form. Ensure that sufficient data is included in the selection ranges. The data that you specify is used only for development and previewing. At runtime, you can specify different selections.
4. Print the report. In the **Select Device (ttstpsplopen)** session, click the **Display** tab and select a device that is configured to generate a file with report metadata and runtime data.

Note:

- Ask your administrator for the name of the appropriate device.
- The device must be of device type "Microsoft Reporting Services" and must have the "-DESIGNER" argument.
- To create such a device, see the *Infor Enterprise Server Plug-in for Microsoft SSRS Administration Guide*.

When you print the report, an XML file named `${BSE_TMP}\ier_`*`[your name]_[report name]`*`.xml` is generated.

The file contains this information:

- Fields, properties, and labels of the LN report.
- The data that falls within the specified selection ranges.

The XML file is used when you design a report in SSDT and when you preview the report.

Step 2 - Creating a Microsoft Reporting report

This section describes how to create a report using the report wizard. Alternatively, you can create an empty report or copy an existing report. See the Microsoft documentation.

To create a report using the report wizard:

1. Start the SQL Server Data Tools:
   - For Microsoft SQL Server 2012, select **Start > All Programs > Microsoft SQL Server 2012 > SQL Server Data Tools**. If you have Microsoft Visual Studio 2010 installed, select **Start > All Programs > Microsoft Visual Studio 2010**.
   - For Microsoft SQL Server 2016, select **Start > All Programs > Microsoft SQL Server 2016 > SQL Server Data Tools**. If you have Microsoft Visual Studio 2015 installed, select **Start > All Programs > Microsoft Visual Studio 2015**.
2. Create a Report Server project or open an existing report project.
3. In the **Solution Explorer**, right-click the Reports folder and select **Add New Report**.

4 In the Report Wizard, click **Next**.
5 Select **New data source** and specify a name.

The usage of shared data sources is not supported for LN reports.

6 From the **Type** combo box, select Infor ES Reporting Service.
7 Specify the connection string for the data source. This string is a semicolon-separated list of key-value pairs, which must include values for the host and the bse.

These are the optional keys:

- Bshell
- Protocol
- Port

If omitted, the keys will have these default values:

- bshell=bshell
- protocol=rexec
- port=512.

If protocol=BaanLogin is specified, the default value for the port is 7150.

This is an example of a static connection string:

```
host=server1;bse=/erpln/bse;bshell=bshell;protocol=baanlogin
```

Caution: Static connection strings are only meant to be used at development time. At runtime, a "dynamic" connection string is required.

See Step 7 - Preparing the report for deployment on page 17.

For subreports for LN a part of the connection string is dynamic.

See Converting stand-alone reports to subreports on page 60.

8 Click **Credentials**.
9 Select **Use a specific user name and password**.
10 Specify your user name and password to log on to your LN development system.
11 Click **OK**.
12 Click **Next**.
13 Click **Query Builder**.

If the Enterprise Server Query Designer is started:

- Select Fields in the Dataset Type.
- To find the report in the tree, expand the desired package and module node.
- To select the desired report, click on the node.

When the Generic Query Designer is started:

- In the Query Designer, specify the report that you generated in Step 1. Alternatively, specify the LN report code, for example tdsls440501200.
- Click the exclamation mark.
- The input fields with runtime data are displayed.

14 Click **OK**.
15 Click **Next** and click **Next** again.
16 Design the table by dragging the fields on the report layout.

17 Click **Finish** and click **Finish** again.
18 To run the report, right-click the newly created report and select **Run**.

Step 3 - Configuring the report

To perform these configuration actions, you can use the Infor ES Report Configurator:

- Set the Language property of the report.

  Setting this property will derive the language from the LanguageISO report parameter from the LN datasource. The Language property determines how certain values such as currencies, dates, and numerics are displayed on the report.
- Create a dataset for properties. These properties are required for the LN-specific parameters.
- Create a dataset for labels. These labels are used to include labels from the LN report on the report that is being designed.

To configure the report:

1 On the **Tools** menu, select **ES Report Configurator**. The Infor ES Report Configurator starts.
2 On the **Report Selection** page, select the report(s) you want to configure:
   - Select the report in the **Not selected reports** box.
   - Click the left arrow button. The report is displayed in the **Selected reports** box.

   Note: To select or deselect all reports simultaneously, you can use the double-arrow buttons.
3 In the left pane, select **Configuration Actions** to open the **Configuration Actions** page.
4 Select these check boxes/options:
   - Set report language
   - Create properties dataset
   - Create labels dataset
5 Click **Execute Actions**.

Step 4 - Modifying the layout of the report

In the report layout, you can add table columns and various other components. SSDT has various features to modify the layout of the report. See the Microsoft documentation.

Step 5 - Adding LN-specific features

You can add various LN-specific features to the report, such as labels, text fields, and messages.

See LN-specific features on page 25.

Step 6 - Previewing the report

To generate a preview of the report, click the **Preview** tab in the report editor.

Note: Do not use the **Run** command on the shortcut menu of the **Solution Explorer**. After you run this command from the **Solution Explorer**, the **Preview** tab in the editor can no longer show the results from LN-specific features. To restore the functionality of the **Preview** tab, you must close SSDT and start SSDT again.

Step 7 - Preparing the report for deployment

Currently, the connection string is a fixed string pointing to a development system.

See Step 2 - Creating a Microsoft Reporting report on page 14.

To start the report from LN, you must change the report's connection string.

When a report is previewed for development purposes in SSDT, the report retrieves the data from the XML file that was generated in LN. When you start the report from LN, dynamic files, which contain a different selection of data, are generated. The information to retrieve the data from these dynamic files is sent with parameters in the URL request. Therefore, at runtime, a different connection string, which contains these parameters, is required.

To prepare the report for deployment:

1. On the **Tools** menu, select **Infor ES Configurator**. The Infor ES Report Configurator starts.
2. On the Report Selection page, select the report(s) you want to prepare for deployment:
   - Select the report in the **Not selected reports** box.
   - Click the left arrow button. The report is displayed in the **Selected reports** box.

   Note: To select or deselect all reports simultaneously, you can use the double-arrow buttons.
3. In the left pane, select **Development/Deployment** to open the Development/Deployment page.
4. Click **Prepare for Deployment**.

   The report's connection string is changed to a generic string. This string picks up the parameters that are passed on the URL that is used to start the report. These parameters on the URL specify the URL of the JCA daemon running in the bshell on the runtime system, and the location of the datafile on the runtime system.

   Note: The report's fixed 'development' connection string is stored in the report's preferences. To modify or preview the report again in SSDT, you must restore the 'development' connection string.

   See Preparing a report for development on page 18.

Step 8 – Deploying the report to a Report Server

To view the designed report directly from LN, the report design must be deployed to a Report Server. To deploy a report design:

1. In the **Solution Explorer**, right-click your project and select **Properties**.

2 On the project **Property Pages**, specify this information:

OverwriteDatasets
Specify **`True`**.

OverwriteDataSources
Specify **`True`**.

TargetReportFolder
Specify the folder where the report designs must be stored. The folder depends on the setup of your Report Server in LN. By default, this folder is Infor Reporting/LN/SessionReports/[LN server name]/[package combination]. See the *Infor Enterprise Server Plug-in for Microsoft SSRS Administration Guide*.

TargetServerURL
Specify the URL that is used to access the Report Server. See the *Infor Enterprise Server Plug-in for Microsoft SSRS Administration Guide*.

Step 9 – Importing the report in the LN dictionary

This step is optional. Perform this step if you want the LN export and import mechanisms to distribute the report designs to other environments. See “Import SSRS Designs” in the *Infor Enterprise Server Plug-in for Microsoft SSRS Administration Guide*.

Modifying a deployed report design

In LN, an administrator can export reports to report design files that can be customized in SSDT. See the *Infor Enterprise Server Plug-in for Microsoft SSRS Administration Guide*.

To modify a deployed report design:

1 Prepare the report for development.
See Preparing a report for development on page 18.

2 Modify the report design and prepare the report for deployment.
See Creating a report design - procedure details on page 13.

Preparing a report for development

A deployed report has a generic connection string. To modify or preview a deployed report design, you must change the connection string to a fixed string pointing to a development system.

To prepare a report for development:

1. On the **Tools** menu, select **Infor ES Configurator**. The Infor ES Report Configurator starts.
2. On the **Report Selection** page, select the report you want to prepare for deployment:
   - Select the report in the **Not selected reports** box.
   - Click the left arrow button. The report is displayed in the **Selected reports** box.

   Note: To select or deselect all reports simultaneously, you can use the double-arrow buttons.
3. In the left pane, select **Development/Deployment** to open the **Development/Deployment** page.
4. Click **Prepare for Development**.

   The report's connection string is changed to the fixed 'development' connection string, which is stored in the report's preferences.

Chapter 4: Creating a Main report with Subreports

This chapter describes how to create a main report with one or more subreports for an LN session. Creating a main report with subreports is necessary for these sessions:

- Sessions for which one 4GL report is opened and closed multiple times within a single print batch. An example is the **Print Contract Quotation Documents (tsctm2400m000)** session.
- Sessions for which multiple 4GL reports are printed within a single print batch. For example, the **Print Production Order Documents (tisfc0408m000)** session can print various reports within a single print batch, such as these:
  - **Order Covering Note (tisfc040801000)**
  - **Routing Sheet (tisfc040802000)**
  - **Operation Note (tisfc040803000)**
  - **Material List (tisfc040804000)**

Procedure overview

To create a main report with subreports:

1. Generate datafiles.
2. Create subreports for the reports that are linked to the print session.
3. Create a main report for the print session and link the subreports to this main report.
4. Prepare the reports for deployment.

Procedure details

Complete these steps:

Step 1 - Generating datafiles

Generate XML files with report metadata and runtime data for the main report and the subreports. These files are used to create datasources with the fields, properties, and labels for the reports.

To generate datafiles:

1. Log on to LN.
2. Start the session for which to create the main report and subreports.
3. Complete the session's form and select the reports for which to create subreports.
4. Print the reports to a device of type "Microsoft Reporting Services." The device must have the "-DESIGNER" argument.

   These files are generated:

   - `${BSE_TMP}\ier_[your name]_[report name]_main.xml`

     For example: `${BSE_TMP}\ier_jdoe_tisfc040801000_main.xml`

     This file contains data for the main report.
   - `${BSE_TMP}\ier_[your name]_[report name].xml`

     For example: `${BSE_TMP}\ier_jdoe_tisfc040801000.xml`

     This file contains data for the subreports.

If you run the session with a selection that does not open multiple 4GL reports, the main file is not created. Subreports can also be activated as stand-alone reports.

Step 2 - Creating subreports

Create subreports for the reports for which you generated metadata and runtime data. Subreports are intended to be used within a main report.

To include subreports in a main report, see Step 3 - Creating a main report and linking the subreports to this main report on page 22.

To create a subreport:

1. Start the SQL Server Data Tools (SSDT) and create a stand-alone report.

   See Creating and modifying reports on page 13.

   Points of attention:

   - In the query designer, select the appropriate report in the tree. The report codes for the Routing Sheet report are in the "REPORT CODE" tags in the `${BSE_TMP}\ier_[your name]_[report name].xml file`.

     See this sample code:

```
</ROW>
</ROWS>
</REPORT>
<REPORT CODE="tisfc040802000" GUID="ba2b2906-4f0f-11e0-b560-e226adcce2cf">
<PROPERTIES>
<PROPERTY NAME="CompanyNumber">660</PROPERTY>
```

- Ensure that the report has a normal page header and page footer, because it can also be activated as a stand-alone report.
- If a subreport is included in a main report, the page header and page footer of the subreport are not printed. You must place header information, such as the report description, in the body of the subreport.

2 Convert the stand-alone report to a subreport:
   a In SSDT, on the **Tools** menu select **Infor ES Configurator**. The Infor ES Report Configurator starts.
   b Select **Report Specific Actions** to open the **Report Selection** page.
   c Select the new report in the **Not selected reports** box. Click the left arrow button to display the report in the **Selected reports** box.
   d In the left pane, select **Configuration Actions**.
   e Select **Convert to Subreport**.
   f Click **Execute Actions**.

   Two additional report parameters, guid and datafile, are generated for the report. The connection string for the report's data source is updated in accordance with these parameters.

   Instead of using the Report Configurator, you can convert the report manually.

   See Manual configuration and deployment steps on page 53.

3 Preview the subreport:
   a Open the report in the report editor and click the **Preview** tab. Two input fields, **guid** and **datafile**, are displayed.
   b Specify the datafile in this format:

   ```
   ${BSE}/tmp/ier_[your_userid]_[your_report].xml
   ```

   c Complete one of these steps:
      - If the datafile contains multiple reports, clear the **NULL** check box for the **guid** field and specify the guid of the report. You can copy the guid from the datafile.
      - If the datafile contains only one report, select the **NULL** check box for the **guid** field. If the **NULL** check box is selected, while the datafile contains more reports, you will be asked to create a main report with subreports for previewing.
   d Click **View Report**.

Step 3 - Creating a main report and linking the subreports to this main report

Main reports are reports with a tablix that uses multiple rows. Each row is linked to a subreport.

To create a main report and link the subreports to this main report:

1 Start the SQL Server Data Tools (SSDT).
2 In the **Solution Explorer**, right-click your report project and select **Add > New Item**.
3 In the **Templates** pane in the **Add New Item** window, select the "Report" template.
4 In the **Name** field, specify the report name in this format: [report name]_main.rdl.

For example, if the name of the main report's data file is `ier_jdoe_tisfc040801000_main.xml`, specify **`tisfc040801000_main.rdl`**.

5 Click **Add** to open the main report in the **Design** view.

6 Add these components to the report:
- Header
- Footer
- Tablix

For details, see the Microsoft documentation.

7 To include subreports in the main report, insert rows in the tablix and link subreports to these rows.

For each subreport to be included in the main report, complete these steps:

a Insert an empty row in the tablix. To insert a row, right-click a row in the tablix and select **Insert Row > Inside Group – Below**.

b Right-click the first and only cell in the new row, and select **Insert > Subreport**. The text "<Subreport>" is displayed in the cell.

c Right-click the "<Subreport>" cell and select **Subreport Properties**.

d In the left pane in the **Subreport Properties** dialog box, select **General** and specify this information:

Name
Specify a name for the subreport.

Use this report as a subreport
Specify one of the subreports created in the previous section.

e In the left pane in the **Subreport Properties** dialog box, select **Parameters**.

f To add the "guid" parameter, click **Add** and specify this information:

Name
Specify **`guid`**.

Value
Specify **`=Fields!guid.Value`**.

g To add the "datafile" parameter, click **Add** and specify this information:

Name
Specify **`datafile`**.

Value
Specify **`=Fields!datafile.Value`**.

h Click **OK**.

i Right-click the row and select **Row Visibility**. Specify this information:

When the report is initially run

Select "Show or hide based on an expression" and specify this expression:

=Fields!report.Value <> "[report code]"

[report code] must correspond with the subreport for the selected row. The report code is in the <report> tag of the main report's row data. See this example:

```
<ROWS>
   <ROW>
      <report>tisfc040801000</report>
      <guid>77252722-2954f-11e0-89ce-0050569b7088</guid>
      <language>2</language>
      <languageISO>en</languageISO>
      <datafile>/prod/toolsdev/Latest/bse/tmp/ier_jdoe_tis
fc040801000.xml</datafile>
   </ROW>
<ROWS>
```

j Click **OK**.

8 When all tablix rows are completed, click the **Preview** tab to preview the main report.

Step 4 - Preparing the reports for deployment

Use the Infor ES Report Configurator to prepare main reports and subreports for deployment.

See Creating and modifying reports on page 13.

When a main report is prepared for deployment, note these actions:

- These subreport parameters are added for all subreports:

Parameter name	Value
jca_uri	=Parameters!jca_uri.Value
jca_ticket	=Parameters!jca_ticket.Value

- The report parameters are updated accordingly.
- The connection string expression is updated accordingly.

When a subreport is prepared for deployment, note these actions:

- The report parameters are updated accordingly, the "guid" parameter becomes hidden.
- The connection string expression is updated accordingly.

When a main report or subreport is prepared for development to modify and preview it again, the above actions regarding the parameters and connection string expression are reversed.

Instead of using the Report Configurator, you can manually prepare main reports and subreports for deployment or development.

See Manual configuration and deployment steps on page 53.

Chapter 5: LN-specific features

This chapter describes how you can use these LN components in your reports:

- Labels
- Messages
- Text fields
- Currency formats
- Images
- Barcodes
- Other parameters, such as report descriptions
- LN dlls

Note: Before you can use these components, you must generate custom code.

See Configuration Actions on page 48.

Using labels in reports

Each report has a Labels dataset that can contain many labels. To use labels in a report, use one of these methods:

- Manually assign labels to fields in the report.

  See Manual configuration and deployment steps on page 53.
- Use the Label Assignment page in the Infor ES Report Configurator to assign labels to report fields.

  See Infor ES Report Configurator on page 47

  .

Using messages in reports

Using messages is similar to using labels:

- Use an expression of the form

```
=Code.tt_getMessage([messagecode])
```

- Ensure that the report contains this block of code:

```
Function tt_getMessage(ByVal code As String) As String
return Infor.ReportingServices.Utilities.LNUtils.GetMessage(code, _
        Report.Parameters!LanguageISO.Value, _
        Report.Parameters!ConnectionString.Value)
End Function
```

Note: Adding this code can be automated with the Report Configurator. See Infor ES Report Configurator on page 47.

The messagecode to be used in the expression is the code as defined in LN, including the package code. For example, specify "tccom00034" to use message "com00034" in package "tc."

Using data fields of type Text in reports

For performance optimization reasons, report datasets only contain the codes of Text fields instead of the actual text. Therefore, the actual text must be retrieved from the LN server.

To use data fields of type Text in a report:

1. Open a report and click the design tab.
2. Right-click anywhere on the design tab, but outside of any control.
3. Select **Report Properties...**.
4. Select the **References** page.
5. If needed, add a reference to the `Infor.ReportingServices.LNDataExtension.dll` assembly.

   See the previous section.
6. In the **Report Properties** dialog box, select the **Code** page.
7. Specify this custom code:

```
Function tt_getText(field as Field) as String
return
Infor.ReportingServices.Utilities.LNUtils.getTextFieldValue(field.Value,
 _
        field("language"), _
        field("company"), _
        Report.Parameters!ConnectionString.Value)
End Function
```

8. Click **OK** to close the **Report Properties** dialog box.
9. Drag a data field of type Text to the report.
10. Right-click the newly created text field.
11. Select **Expression**.
12. Enter this expression with the real text field in it, for example:

    `Code.tt_getText(Fields!your_text_field)`

Note: You must pass the field object, not the value property.

13 Click **OK** to close the **Expression** dialog box.

14 To show the text field content, save and run the report.

Note:

- Alternatively, you can generate the custom code and the expression through the Infor ES Report Configurator. See Infor ES Report Configurator on page 47.
- If the LN session report uses the lattr.textexpand option, you cannot use this text expression. Therefore, you must use the Value property of the text field, for example, =Fields!your_text_field.Value

Filtering lines of text

This table shows the functions that you can use to filter lines of text:

Function	Syntax, function definition, and description

tt_filterTextLinewise	tt_filterTextLinewise(text, includePattern, replacementString)
	`Function tt_filterTextLinewise(inputText as String, _`
	`  includePattern as String, _`
	`  replaceString as String) as String`
	`  return Infor.ReportingServices.Utilities.LNUtils.FilterTextLinewise( _`
	`  inputText, _`
	`  includePattern, _`
	`  replaceString)`
	`End Function`
	Filters the lines of text, replacing lines that match the given includePattern with the given replacementString.
	Lines that do not match are deleted. The return value contains the filtered text.
	The replacementString may contain back-references, such as "\1" and "\2", to capturing groups in the includePattern. A replacementString "\0" prints the entire original line without any changes.

Function	Syntax, function definition, and description

tt_filterTextBlockwise	tt_filterTextBlockwise(text, includeStart, includeEnd, excludeStart, excludeEnd, includeUnmarkedLines)
	`Function tt_filterTextBlockwise(inputText as String, _`
	`  includeStart as String, includeEnd as String, _`
	`  excludeStart as String, excludeEnd as String, _`
	`  includeUnmarkedLines as Boolean) as String`
	`return Infor.ReportingServices.Utilities.LNUtils.FilterTextBlockwise( _`
	`  inputText, _`
	`  includeStart, _`
	`  includeEnd, _`
	`  excludeStart, _`
	`  excludeEnd, _`
	`  includeUnmarkedLines)`
	`End Function`
	
	Filters the lines of text that are organized in "blocks", and returns the lines that pass the filter.
	A line that matches the regular expression "includeStart" marks the beginning of an "include block".
	Similarly, "includeEnd" marks the end of an "include block", and "excludeStart" / "excludeEnd" mark the beginning / end of an "exclude block".
	All lines within an "include block" are returned by this filter; lines within an "exclude block" are deleted. The start/end marker lines are also deleted.
	If "includeUnmarkedLines" has value 1, lines outside any block are returned as well; if "includeUnmarkedLines" has value 0, those lines are deleted.

Example - filtering text lines

A multiline text may contain lines that are marked as only for internal reports or only for external reports. When developing an external report, you want to filter the text so that the internal-only lines are excluded, and the external-only lines are included. To accomplish this, you can use one of the tt_filterText functions.

These functions are available, for two different types of mark-up used on the text:

- tt_filterTextLinewise

  This function is used for text on which each line is individually marked as external or internal, for example by using a special character at the beginning of the lines. For example, you can use these special characters at the beginning of the lines:

  - A "<" character to mark a line as "internal-only"
  - A ">" character to mark a line as external-only.

For example, you want to filter this text:

```
Normal line

The empty line above, and this line, are also normal
<This is an internal-line
Normal again
>External line
Final line is normal text.
```

You want the external-only report to have this text:

```
Normal line

The empty line above, and this line, are also normal
Normal again
External line
Final line is normal text.
```

- tt_filterTextBlockwise

  This function is used for text consisting of "blocks" that are marked by special begin and end markers. For example, you can use these marker lines:

  - "<<<" to mark the beginning and end of an "internal block of text"
  - ">>>" to mark the beginning and end of an "external block of text"

  For example, you want to filter this text:

```
<<<
This is internal text
consisting of several lines.
<<<
>>>
This is external text
Not to be printed in internal reports
>>>
```

  You want the external-only report to have this text:

```
This is external text
Not to be printed in internal reports
```

The tt_filterTextLinewise function uses these arguments:

- The text to be filtered. This can be a fixed string, but usually this is a tt_getText expression.
- A regular expression that matches lines to be included. In the example, the expression must match "normal" and "internal" lines, but should not match "external" lines. This regular expression meets these conditions: "^(>|(?!<))(.*)$". This expression uses "negative look-ahead" functionality to exclude lines beginning with "<".

- A replacement string for the matching lines. This string may contain "backreferences" to capturing groups in the regular expression. For example, "\2" is substituted by the contents of the second capturing group in the regular expression. "\0" prints the entire matching line.

The tt_filterTextBlockwise function uses these arguments:

- The text to be filtered. This can be a fixed string, but usually this is a tt_getText expression.
- A regular expression that matches lines that are used as begin markers for a block of text that must be included.
- A regular expression that matches lines that are used as end markers for a block of text that must be included. In the example, identical begin and end markers are used, but they may be different.
- A regular expression that matches lines that are used as begin markers for a block of text that must be excluded.
- A regular expression that matches lines that are used as end markers for a block of text that must be excluded.
- A flag that indicates whether text that is outside the two blocks must also be included in the output.

To use this functionality, specify an expression similar to the following code as the expression for a field on your report:

```
=Code.tt_filterTextLinewise(Code.tt_getText(Fields!your_text_field),
"^(>|(?!<))(.*)$", "\2")
```

This code prints these lines:

- Lines that start with ">". The ">" character itself is not included in the output.
- All other lines, except those that start with "<".

The same can be accomplished with this code:

```
=Code.tt_filterTextLinewise(Code.tt_getText(Fields!your_text_field),
"^>(.*)$|^([^<].*)$|^$", "\1\2")
```

In this case the regular expression consists of these alternatives:

- Matching lines that start with ">"
- Lines that start with anything else than "<"
- Empty lines

The replacement string is the concatenation of the first and second capturing groups. At least one of those groups will be empty.

This code is an example for using the blockwise filtering:

```
=Code.tt_filterTextBlockwise(Code.tt_getText(Fields!your_text_field),
"^>>>$", "^>>>$", "^<<<$", "^<<<$", False)
```

This code prints text inside blocks marked by a start marker ">>>" and an end marker ">>>". All other lines are suppressed.

The report must contain function declarations for the tt_filterTextLinewise and tt_filterTextBlockwise functions. Use the LN Configurator to generate the custom code in your report.

Using currency formats in a report

To use a currency format expression in the Format property of a text field that contains a currency amount:

1. Click the text field containing the currency amount you want to format.
2. Go to the **Properties** window.
3. If the properties of **Selected Text** are displayed instead of the properties of the **Text Box**, press Esc once.
4. Click the **Format** property and select **<Expression...>** from the drop-down box.
5. Specify an expression such as the following:

   =Code.tt_getCurrencyFormat("006", Fields!your_currency_field.Value)

   The first argument must be the format to be used (defined in LN as %A006). The second argument must be the field that contains a currency, for example Fields!tdsls400_ccur.Value.
6. Click **OK** to close the **Expression** dialog box.
7. To get this expression working:
   a. Verify that the report contains a reference to the `Infor.ReportingServices.LNDataExtension.dll` assembly. See the instructions for using labels.
   b. On the **Code** page of the **Report Properties**, add the following block of custom code if it is not present:

```
Function tt_getCurrencyFormat(format as String, currency as String)
 as String
return Infor.ReportingServices.Utilities.LNUtils.GetCurrencyFormat(
 _
        format, _
        currency, _
        Report.Parameters!ConnectionString.Value)
End Function
```

   Note: Adding this code can be automated with the Report Configurator.

   See Infor ES Report Configurator on page 47.

   c. Preview the report. The currency amount should be correctly formatted.

Using images in reports

Four types of images can be used on a report:

Company logo

The company logo is an image that is linked to the company record in the **Companies (ttaad1100m000)** session. If no image is linked to the company, the image that is linked to company 000 is used as company logo. If no image is linked to company 000, the company logo from the Additional File in the Tools AddOn (ta) package is used. The code of this Additional File is `tagencompany_logo.gif`. The company logo can be placed on a report by adding an image to the report.

To add an image to the report:

1. Open the design page of the report.
2. Right-click the report design page and select **Insert > Image**.
3. Give the image a name.
4. In the **Select the image source** field, specify the **`Database`** option.
5. Click the **Expression** button (**fx**) next to the **Use this field** drop-down box.
6. Specify an expression such as the following:

   `=Code.tt_getCompanyLogo()`
7. In the **Use this MIME type** field, specify the desired option. For example, 'image/gif'. The MIME type depends on the image format.
8. Ensure that the report contains this block of code:

```
Function tt_getCompanyLogo() as String
return
Infor.ReportingServices.Utilities.LNUtils.GetCompanyLogo(Report.Param
eters!ConnectionString.Value)
End Function
```

Note: Adding this code can be automated with the Report Configurator.

See Infor ES Report Configurator on page 47.

Application image

An application image is linked to an LN application record and identified by a unique id (guid). An application image can be placed on a report by adding an image to the report.

To add an image to the report:

1. Open the design page of the report.
2. Right-click the report design page and select **Insert > Image**.
3. Give the image a name.
4. In the **Select the image source** field, specify the **`Database`** option.
5. Click the **Expression** button (**fx**) next to the **Use this field** drop-down box.
6. Specify an expression such as the following:

   `=Code.tt_getApplicationImage(Fields!your_field.Value, Clng(Parameters!CompanyNumber.Value))`

The first argument must be the field that contains a guid, representing the application image. The second argument must contain the company number converted to a value of datatype long.

7 In the **Use this MIME type** field, specify the desired option. For example, 'image/gif'. The MIME type depends on the image format.

8 Ensure that the report contains this block of code:

```
Function tt_getApplicationImage(ByVal guid as String, ByVal company as
 Long) as String
return Infor.ReportingServices.Utilities.LNUtils.GetApplicationIm
age(guid, _
 company, _
 Report.Parameters!ConnectionString.Value)
End Function
```

Note: Adding this code can be automated with the Report Configurator.

See Infor ES Report Configurator on page 47.

Enum image

An enum image is linked to an LN enumerated domain.

To place the image on the report:

1 Open the design page of the report.

2 Right-click the report design page and select **Insert > Image**.

3 Give the image a name.

4 In the **Select the image source** field, specify the **`Database`** option.

5 Click the **Expression** button (**fx**) next to the **Use this field** drop-down box.

6 Specify an expression such as the following:

`=Code.tt_getEnumImage("your_domain", Fields!your_field.Value)`

The first argument must be the LN enumerated domain. The second argument must contain the enum value.

7 In the **Use this MIME type** field, specify the desired option. For example, 'image/gif'. The MIME type depends on the image format.

8 Ensure that the report contains this block of code:

```
Function tt_getEnumImage(ByVal domainValue as String, ByVal enumValue
 as Long) as String
return Infor.ReportingServices.Utilities.LNUtils.GetEnumImage(domain
Value, _
 enumValue, _
 Report.Parameters!ConnectionString.Value)
End Function
```

Additional file

An additional file is an image that is stored in the LN server. To place the file on the report:

1. Open the design page of the report.
2. Right-click the report design page and select **Insert > Image**.
3. Give the image a name.
4. In the **Select the image source** field, specify the **`Database`** option.
5. Click the **Expression** button (**fx**) next to the **Use this field** drop-down box.
6. Specify an expression such as the following:

   `=Code.tt_getAdditionalFile("tt", "gbf", "abort1.gif")`

   The first argument must contain the package code. The second argument must contain the module code. The third argument must contain the file name of the additional file.
7. In the **Use this MIME type** field, specify the desired option. For example, 'image/gif'. The MIME type depends on the image format.
8. Ensure that the report contains this block of code:

```
Function tt_getAdditionalFile(ByVal packageCode as String, ByVal mod
uleCode as String, ByVal fileCode as String) as String
return
Infor.ReportingServices.Utilities.LNUtils.GetAdditionalFile(packageCode,
 _
 moduleCode, fileCode, _
 Report.Parameters!ConnectionString.Value)
End Function
```

Using company formats in a report

Company numbers can have one to four digits. If no company numbers greater than 999 are used, the standard is to print company numbers with three characters, such as 050, or 100. If company numbers greater than 999 are used, all company numbers are printed with four characters. For example, 0050, 0100, and 1210. To ensure consistency, company numbers should be printed using the defined company format.

To use company formats in a report:

1. Ensure that the tt_getCompanyFormat method is defined in the report's custom code. To define this method, complete one of these steps:
   - Start the Infor ES Report Configurator and generate custom code for the report.

     See Infor ES Report Configurator on page 47.
   - In the report's custom code, add this code:

```
Function tt_getCompanyFormat() as String
return Infor.ReportingServices.Utilities.LNUtils.GetCompanyFormat
```

```
(Report.Parameters!ConnectionString.Value)
End Function
```

- For each company field, complete these steps:
  - **a** Right-click the field and select **Text Box Properties** to open the **Text Box Properties** dialog box.
  - **b** In the left pane, select **Number**.
  - **c** In the **Category** field, specify **`Custom`**.
  - **d** In the **Custom format** field, specify this expression:

    `=code.tt_getCompanyFormat()`

The company number will now always be printed in three or four characters.

Using Barcodes in reports

Use the custom code of an SSRS Report to include barcode images in reports.

To include a barcode bitmap image in an SSRS Report:

1. Open a report.
2. Click the **Design** tab.
3. In the **Design** view, right-click a report item.
4. Select **Insert > Image**.
5. Specify a name for the image.
6. Specify **`Database`** in the **Select the image source** field.
7. Click **Expression** (**fx**) next to the **Use this field** drop-down box
8. Specify an expression that calls a function in the custom code to construct the barcode, for example:

   =Code.getBarcodeForItem(Fields!tcibd001_item.value)
9. Specify image/bmp in the **Use this MIME type** field.
10. Close the image properties dialog box.
11. Select **Report Properties** and click the **Code** tab.
12. Add the function to construct the barcode in the custom code. See the API specifications and example sections.

API specifications

The API provides one method to create either 1D or 2D barcode images in the `Infor.Reporting Services.Utilities` assembly:

- Method name: GetBarcodeImage(Barcode.Type barCodeType, string barCodeValue, ref ArrayList valuePairOptions)
- This method returns: Base 64 encoded bitmap image representing the barCodeValue

- Arguments:
  - barCodeType: 1D or 2D supported barcode type. See the barcode types table later in this chapter.
  - barCodeValue: The value for which a barcode image is constructed.
  - valuePairOptions: Barcode-specific parameters. See the supported parameters that are listed in the barcode parameters table later in this chapter.

This table shows the supported Barcode Types. Types must be preceded by **`Infor.ReportingServices.Utilities.Barcode.Type`**.

1D Barcode type	1D Barcode types	2D Barcode types
AUSTRALIAPOST	CHINA_POST	AZTEC
CODABAR	CODE_11	DATABAR
CODE_128	CODE_128A	DATAMATRIX
CODE_128B	CODE_39	MAXICODE
CODE_93	CODE_B	MICRO_PDF
DATABAR_EXP	DATABAR_LIM	MICRO_QR
DATABAR_OMNI	DATABAR_TRUNC	PDF417
DEUTSCHEPOST	EAN_13	QRCODE
EAN_13_PLUS_2	EAN_13_PLUS_5	TRUNCATED
EAN_14	EAN_8	
EAN_8_PLUS_2	EAN_8_PLUS_5	
EXTENDED_39	EXTENDED_93	
FOURSTATE	GS1_128	
HIBC_128	HIBC_39	
IATA_2_OF_5	INFOMAIL_A	
INTELLIGENT_MAIL	INTERLEAVED_2_OF_5	
ISBN	ISMN	
ISSN	ITALIAP25	
ITALIAP39	ITF_14	
ITF_6	JAPANPOST	
KOREANPA	MATRIX_2_OF_5	
MSI_PLESSEY	PLANET_12	
PLANET_14	PLESSEY	
POSTNET	PZN	
RM4SCC	SSCC	

1D Barcode type	1D Barcode types	2D Barcode types
STANDARD_2_OF_5	TELEPEN	
TELEPENA	TELEPENN	
UPC_A	UPC_A_PLUS_2	
UPC_A_PLUS_5	UPC_E	
UPC_E0	UPC_E1	

This table shows the Barcode Parameters. Parameters must be preceded by **`Infor.ReportingServices.Utilities.Barcode.Parameter`**.

Barcode parameter	Barcode Type	Parameter Type	Description
Indicators	1D	boolean	Displays light margin indicators for those barcode types that support these indicators.
BothBearers	1D	boolean	Determines if both upper and lower bearer bars are displayed.
ExtendBearers	1D	boolean	Allows bearer bars to extend into light margins.
ShowText	1D	boolean	Specifies that barcode text content must be displayed under the bars.
AutoCheckdigit	1D	boolean	Specifies if check digits are calculated automatically.
ShowCheckdigit	1D	boolean	Displays an automatic check digit (for those barcode types which permit this).
Extra1	1D	boolean	Additional properties which are not always used. Additional properties provide additional functions for a limited number of specific barcode types
Extra2	1D	boolean	Additional properties which are not always used. Additional properties provide additional functions for a limited number of specific barcode types
AzFlag	2D	boolean	Switches the Aztec flag on or off. Aztec and 2D Universal only.
AzMenu	2D	boolean	Switches the Menu flag on or off. Aztec only
AzRvideo	2D	boolean	Switches the Reverse Video flag on or off. Aztec only.
Width	1D	float	Required target width of barcode image

Barcode parameter	Barcode Type	Parameter Type	Description
Height	1D	float	Required target height of the barcode image
Xunit1D	1D	float	Specifies the thickness of each barcode element in mils (1/1000 inches)
Xunit2D	2D	float	Specifies the thickness of each barcode element in mils (1/1000 inches)
Ymultiplier	2D	float	The height of a barcode element specified in Xunits
StartMode	2D	int	Starting mode for the 2D barcode: 0=Numeric; 1=Alphanumeric; 2=Byte
SecurityLevel	2D	int	Specifies the security level.
Columns	2D	int	Target number of columns for Datamatrix, PDF417 or Databar symbols

Extra barcode parameter information

Security level values

Security level values are the amount of redundancy that is built in the barcode image to allow errors to be corrected. The greater the security level the larger the barcode image.

For PDF417, the allowed values are 0-8.

Aztec codes

The allowed security level values for Aztec codes depend on the mode:

- Normal mode 0-99
- Compact mode 1-4
- Full range mode 1-32 (the value is ignored for Runes).

For the Compact mode and Full range mode, the security level is the required number of Aztec layers in the symbol.

Datamatrix symbols

Datamatrix symbols are the allowed security level values for:

- Square symbols: 0–23
- Rectangular symbols: 0-6
- QR Codes: 0-3
- Micro QR Code: 0-2

- Other barcode types: 0

VB example for a method call that retrieves a 1D barcode with text

```
Function GetBarcodeImage(value as String) as String
Dim parameters as new System.Collections.ArrayList()

parameters.add(New Object() {Infor.ReportingServices.Utilities.Barcode.Pa
rameter.Width, 100.0F} )
parameters.add(New Object() {Infor.ReportingServices.Utilities.Barcode.Pa
rameter.Height, 20.0F} )
parameters.add(New Object() {Infor.ReportingServices.Utilities.Barcode.Pa
rameter.ShowText, true} )

return Infor.ReportingServices.Utilities.LNUtils.GetBarcodeImage(Infor.Re
portingServices.Utilities.Barcode.Type.CODE_39, value,  parameters)
End Function
```

VB example for a method call that retrieves a 2D barcode.

```
Function GetBarcodeImage(value as String) as String
Dim parameters as new System.Collections.ArrayList()

parameters.add(New Object() {Infor.ReportingServices.Utilities.Barcode.Pa
rameter.Xunit2D, 40F} )
parameters.add(New Object() {Infor.ReportingServices.Utilities.Barcode.Pa
rameter.StartMode, 1} )

return Infor.ReportingServices.Utilities.LNUtils.GetBarcodeImage(Infor.Re
portingServices.Utilities.Barcode.Type.QRCODE, value,  parameters)
End Function
```

Using other parameters

You can use predefined report parameters to display additional information in a report.

This table shows some of the report parameters that are available for use in reports:

Parameter	Description
ReportDescription	The title of the report
UserId	The ID of the LN user
UserName	The full name of the LN user

The predefined report parameters are mostly for internal use. You encounter these parameters in the custom code described earlier. It can sometimes be useful to show some parameters inside the report.

Before you can use parameters in the report design, you must make them available to the report.

Making the parameters available

Ensure that all predefined parameters are available in the report.

Complete these steps:

1. Start the Infor ES Report Configurator.
   See Infor ES Report Configurator on page 47.
2. In the **Configuration Actions** panel, select the **Create properties dataset** check box.
3. Click **Execute actions**.
4. Close the configurator.
5. Close and re-open the report. The new properties and parameters are available.

Note: You need Infor ES Report Configurator version 1.0.27.1122 or higher for this.

You can also add the extra properties and parameters manually: add a new "Query Field" to the "Properties" dataset, create a new "Report Parameter", and link the parameter to the property. However, it is highly recommended to use the Infor ES Report Configurator.

Using parameters in the report design

This example shows how you can use the ReportDescription parameter to print the title of a report.

Example

To print the report title:

1. Open the Design page of the report.
2. Find the textbox that contains the title of the report. The textbox has been generated with the name given to the report.
3. Delete the contents of the textbox.
4. Right-click the empty textbox and select **Create Placeholder**.
5. Click the **Expression** button (**fx**) next to the **Value drop-down** box.
6. In the **Category** pane, select **Parameters**.
7. In the **Values** pane, double-click **ReportDescription**.
8. The expression value should now be =Parameters!ReportDescription.Value
9. Click **OK**.
10. Specify a value in the **Label** field, such as “ReportDescription”.
11. Click **OK**.

Note: You can also drag the ReportDescription parameter from the list of parameters to the empty textbox.

Executing a method call of an LN Library

You can execute an LN Library method call in these ways:

- From within the custom code of an SSRS Report (VB)
- Using a custom assembly build with .Net Framework 3.5 (either C#, VB, or C++)

API specifications

The API provides one method to execute an LN Library method call in the Infor.ReportingServices.Utilities assembly:

- Method name: DoErpLnDllCall(string extendedConnectionString, string libraryName, string methodName, string methodReturnType, [ref object methodReturnValue], ref ArrayList methodArgumentList)
- If the call is successful, the method returns 0. If the call is unsuccessful, the method returns -1.
- Arguments:
  - extendedConnectionString: String containing connection information
  - libraryName: Name of the LN dll
  - methodName: Name of the method to call

- methodReturnType: Return type of the dll method. Only return types supported by Adapter4ERP are implemented. These are valid method return types:
  - "void"
  - "long"
  - "int",
  - "boolean"
  - "string"
  - "double"
  - "xmlstring"
- methodReturnValue: Return value of the LN Dll method. This argument should not be used to call void dll methods.
- methodArgumentList: ListArray of method arguments. A method argument contains an array of serializable argument settings. It holds the following settings in the given order:
  - scope: one of "in", "out" or "inout"
  - type: one of "string", "bool", "long", "double", "xml", "int"
  - value: required for arguments with scope "in" or "inout"
  - length: required for arguments of type "string", with scope "out" or "inout"

Examples of method calls used in a Microsoft Report (VB code)

To execute a method call from within the custom code of an SSRS Report:

1. Open a report and click the **Design** tab.
2. Right-click anywhere in the Design and select **Report Properties....**.
3. If required, add a reference to the 'Infor.ReportingServices.LNDataExtension.dll' assembly.
   See Using labels in reports on page 25.
4. Click the **Code** tab and specify one of the examples listed below.
5. Create an expression that calls the example method, for example, "=Code.IsLeapYear(2012)"

Example (VB) for a method call that checks if a given year is a leap year

```
Function IsLeapYear(year as long) as String
Dim retval as boolean
Dim ret as integer
Dim methodArgs as new System.Collections.ArrayList()

methodArgs.add(New Object() {"in", "long", year} )

ret = Infor.ReportingServices.Utilities.LNUtils.DoErpLnDllCall(Report.Pa
rameters!ConnectionString.Value, _
```

```
      "otccomdll0350", _
      "tccom.dll0350.is.leap.year", _
      "boolean", _
      retval, _
      methodArgs)

'Returns "2012 = true"
return year & " = " & retval
End Function
```

Example (VB) for a method call that converts a given number of seconds to a time in the format [hh:mm:ss]

```
Function TimeToString() as String
Dim seconds as Long
Dim ret as integer
Dim methodArgs as new System.Collections.ArrayList()

'Position of the return value for method arguments with scope "out" or
"inout".
Dim retValPos = 2

Dim retObj() as Object

seconds = 86300

methodArgs.add(New Object() {"in", "long", seconds} )
methodArgs.add(New Object() {"out", "string", "", 30} )

ret = Infor.ReportingServices.Utilities.LNUtils.DoErpLnDllCall(Report.Pa
rameters!ConnectionString.Value, _
           "otccomdll0350", _
           "tccom.dll0350.convert.time.value.to.string", _
           methodArgs)

retObj = methodArgs(1)

'Returns "86300 equals 23:58:20"
return  seconds & " equals " & retObj(retValPos)
End Function
```

Example (VB) for a method call that checks a business partner relation filter

```
Function CheckBP() as String
Dim bp as String
Dim ret as integer

'Position of the return value for method arguments with scope "out" or
```

```
"inout".
Dim retValPos = 2

Dim methodArgs as new System.Collections.ArrayList()

bp = "BPG000001"

methodArgs.add(New Object() {"in", "string", bp} )
methodArgs.add(New Object() {"out", "string", "", 30} )
methodArgs.add(New Object() {"out", "string", "", 30} )
methodArgs.add(New Object() {"out", "string",  "", 100} )

ret = Infor.ReportingServices.Utilities.LNUtils.DoErpLnDllCall(Report.Pa
rameters!ConnectionString.Value, _
      "otccomdll4000", _
      "tccom.dll4000.check.business.partner.relation.filter", _
      methodArgs)

'Returns "Business Partner BPG000001 tccom100.* tccom100(tccom100.bpid =
 "BPG000001" or tccom100.brbp = "BPG000001")
return  "Business Partner:" _
          + bp + " " _
          + methodArgs(1)(retValPos) _
          + methodArgs(2)(retValPos) _
          + methodArgs(3)(retValPos)

End Function
```

Example (VB) for a method call that reads company data, which has no method arguments

```
Function ReadCompanyData() as String
Dim ret as integer
Dim retval as String

ret = Infor.ReportingServices.Utilities.LNUtils.DoErpLnDllCall(Report.Pa
rameters!ConnectionString.Value, _
      "otcmcsdll0095", _
      "tcmcs.dll0095.read.company.data", _
      "string", _
      retval, _
      new System.Collections.ArrayList())

return  retval

End Function
```

Examples of method calls used in a custom assembly build with .Net Framework 3.5 (either C#, VB, or C++)

To execute a method call from within a custom assembly:

1. Create your own custom assembly using MS Visual Studio.
2. Add a reference to the Infor.ReportingServices.LNDataExtension.dll assembly.
3. In your custom assembly, implement one of the C# examples listed below.
4. Build your custom assembly.
5. Deploy your custom assembly.
6. In the SSRS Report, create a reference to your custom assembly. See Using labels in reports on page 25.
7. Call your custom assembly method from an SSRS Report.

Example (C#) for a method call that checks if a given year is a leap year

```
[PermissionSet(SecurityAction.Assert, Name = "FullTrust")]
        public static bool IsLeapYear(string extendedConnectionString,
long year)
        {
            int ret;
            object retval = null;
            ArrayList methodArgs = new ArrayList();


           methodArgs.Add(new object[] {"in", "long", year} );

            ret = DoErpLnDllCall(extendedConnectionString,
                "otccomdll0350",
                "tccom.dll0350.is.leap.year",
                "boolean",
                ref retval,
                ref methodArgs);

            return (bool)retval;
        }
```

Example (C#) for a method call that converts a given number of seconds to a time in the format [hh:mm:ss]

```
[PermissionSet(SecurityAction.Assert, Name = "FullTrust")]
       public static string TimeToString(string extendedConnectionString,
 long seconds)
        {
            int ret;
```

```
          //Position of the return value for method arguments with scope
"out" or "inout".
           int retValPos = 2;

           ArrayList methodArgs = new ArrayList();

           methodArgs.Add(new object[] {"in", "long", seconds} );
           methodArgs.Add(new object[] { "out", "string", "", 30 });

           ret = DoErpLnDllCall(extendedConnectionString,
               "otccomdll0350",
               "tccom.dll0350.convert.time.value.to.string",
               ref methodArgs);

           object[] retObj = (object[]) methodArgs[1];
           return seconds + " equals " + (string) retObj[retValPos];
       }
```

Chapter 6: Infor ES Report Configurator

Introduction

This chapter describes the functionality of the Infor ES Report Configurator.

The Infor ES Report Configurator is a Visual Studio Add-in for the SQL Server Data Tools (SSDT) of Microsoft Reporting. You can use this configurator to automate several aspects of creating a report for LN.

Using the Add-in

The Infor ES Report Configurator is part of the Infor ES plug-in for Microsoft SQL Server Reporting Services.

To use the Infor ES Report Configurator:

1. Start SSDT.
2. Create or open a Report Server Project.
3. From the **Tools** menu, select **Infor ES Configurator**. The ES Report Configurator window is displayed. The window consists of these components:
   - A pane that shows a tree menu where you can select the desired action.
   - A Report Selection page
4. To perform actions on one or more reports, select the desired reports on the Report Selection page. In the left pane, select the desired action. The corresponding page is opened. Perform the desired action. See Report Specific Actions on page 48.
   Note: To assign labels to fields on the report, only one report must be selected.

   See Label Assignment on page 51.
5. To modify or view preference settings, in the left pane select **Preferences**. Perform the desired action.

   See Preferences on page 52.

Report Specific Actions

After you select one or more reports on the Report Selection page, you can select these report-specific actions from the tree menu:

- Configuration Actions
- Development/Deployment
- Label Assignment

See the following sections.

Note:

In these situations, you cannot perform any of the report-specific actions:

- If you have not selected a report.
- If one or more of the selected reports are dirty. That is, the report is left open in an unsaved state.

Configuration Actions

The **Configuration Actions** panel contains actions that you can perform on the reports that you selected in the report selection panel.

To perform one or more actions:

1. Select the corresponding check boxes.
2. Click **Execute actions**.

Note: The actions are independent of each other and can be performed separately. You can repeat the actions later, although they may not have an effect if they have already been performed.

The **Configuration Actions** panel contains these actions:

Set report language

Each report has a Language property. This property determines how certain values such as currencies, dates, and numerics are displayed on the report.

On the LN reports, the language must be based on a report parameter.

If the **Set report language** check box is selected, the language property is automatically set to an expression that uses this report parameter.

Generate custom code

If you select the **Generate custom code** check box, the Infor ES Report Configurator generates these methods in the custom code:

- tt_getText

- t_filterTextLinewise
- tt_filterTextBlockwise
- tt_getLabel
- tt_getMessage
- tt_getCompanyLogo
- tt_getApplicationImage
- tt_getEnumImage
- tt_getAdditionalFile
- tt_getCurrencyFormat
- tt_getCompanyFormat

To retrieve data that is not in the generated XML report data, you can use these methods to make calls to the ES server.

For example, you can use the tt_getText method to add LN data fields of type Text in a report.

Note:
- The tt_ prefix is used to avoid conflicts with code that is added by the report designers.
- When you perform the "Generate custom code" action multiple times, existing methods in the custom code are never overwritten, but missing methods will be added.

Set textfield expressions

If the **Set textfield expressions** check box is selected, tt_getText expressions for all textfields on the report are generated in the custom code. To retrieve the text of textfields from the LN server, you can use the generated expressions.

Note: You must generate the tt_getText method through the **Generate custom code** option.

See Generate custom code on page 48.

Create properties dataset / Create labels dataset

An LN report usually has three datasets:

- A dataset for all fields
- A dataset for the properties
- A dataset for the labels

If you select the **Create properties dataset** and **Create labels dataset** check boxes, the datasets for properties and labels are generated automatically.

Before you generate these datasets, ensure that these prerequisites are met:

- The fields dataset must already exist.

  See Creating and modifying reports on page 13.
- The connection string of the datasource is a development connection string.

See Prepare for development and deployment on page 50 and Creating and modifying reports on page 13.

Note: Instead of generating the properties and labels datasets, you can create them manually.

See Manual configuration and deployment steps on page 53.

Convert to Subreport

If the **Convert to Subreport** check box is selected, the report is converted to a subreport, which can be linked to a main report. Two additional report parameters, guid and datafile, are generated. The connection string for the data source is updated in accordance with these parameters.

If multiple LN reports are printed into one single spool, you must create a main report with subreports. See Creating a Main report with Subreports on page 20.

Prepare for development and deployment

To run a report from LN, you must prepare a report for deployment.

To continue development and to view the report in the preview mode, you must prepare the report for development after it has been deployed. This action reverts the changes that were made during the deployment.

The Prepare for Development and Deployment page contains these commands:

Prepare for deployment

After you execute this command, the report can be run from within LN. You can no longer run the report in SSDT.

The **Prepare for Deployment** command automatically performs these actions:

- Adds additional parameters that are used to:
  - Ensure that the report renders data from a dynamic source. Each generated report request contains a different datafile with data.
  - Add information to the report, to make the connections secure.
- Generates a different connection string that uses the additional parameters.
- Replaces the datasource username and password by a dummy username and an empty password.
- Stores the original development connection string and the corresponding username and password. The original development connection string and the corresponding username and password are used to restore the original values when preparing for development. You can also modify the connection string on the Preferences page.

Prepare for development

After you execute this command, you can modify the report in SSDT and view the report in the preview mode. You can no longer run the report from within LN.

The **Prepare for Development** command automatically reverts all the actions that were performed to deploy the report. That is, the command removes the additional parameters and restores the original development connection string, username, and password.

Label Assignment

Each report has a Labels dataset that can contain many labels. To assign these labels to fields on the report, use the **Label Assignment** page.

The **Label Assignment** page contains two grids:

- A grid on the left with all the Textboxes, or TextRuns, of the report
- A grid on the right with all the labels, including their descriptions

To assign a label to a field on the report:

1. Select one of these options:
   - Label Reference: The label will be assigned by reference.
   - tt_getLabel() Expression: The label will be assigned by expression.
2. Drag the label from the right grid and drop it on a textbox in the left grid. The label will be assigned to this textbox.
3. Optional: If you selected to assign the label by expression, you can change the desired length of the label in the **Expression** box. While editing the expression, you can see a preview of the label.

   To get a preview for the label that is used in the expression, the Language property must be set on the Preferences page. By default, language 'en', which stands for US English, is used.

Textbox types

The **Textboxes** grid shows all the Textboxes, or TextRuns, of the report. This table shows the Textbox types:

Type	Description	Textual representation
Static	Contains static text	The static text
Label Reference	Contains a reference to a label in the labels dataset	The name of the label
Label Expression	Contains an expression to retrieve a label	The name of the label
Data Reference	Contains a reference to a field in the fields dataset	The name of the field
Other	Everything that does not belong to any of the types above	Nothing

To filter the grid by these types:

1. Click the down arrow button at the top of the grid.
2. From the pull-down menu, select the types that you want to see in the grid. All other types will be hidden.

Every Textbox can consist of multiple TextRuns. Therefore, you can assign multiple labels to different TextRuns within the same Textbox. If a Textbox has more than one TextRun, the **Textboxes** grid will show each TextRun individually.

Note:

- The label assignment functionality can only be used for one report at a time. Therefore, you cannot select multiple reports in the Report Selection panel.
- Ensure that the report uses the development connection string. Otherwise, the report will connect to the server for the label preview, and to get the labels metadata.
- The labels dataset must be available. The name of the labels dataset must be “Labels.”

Preferences

The Preferences page can contain the development connection string that is used by the “Prepare for development” action. This connection string is stored only for the current solution or project that is opened in SSDT.

See Prepare for development and deployment on page 50.

In addition, the Language property can be set. The language property is used to retrieve label variants that are displayed in the Preview text box on the Label Assignment page.

Chapter 7: Manual configuration and deployment steps

This chapter describes how you can manually perform various configuration and deployment steps for a report.

Note: You can also configure and deploy a report through the Infor ES Report Configurator. See Creating and modifying reports on page 13 and Infor ES Report Configurator on page 47.

Creating a dataset for labels

To create a dataset for labels:

1. Start the Report Data viewer.
2. Right-click the **Datasets** node and select **Add Dataset**.
3. In the **Choose a data source and create a query** dialog box, specify a name for the dataset. To use the Label Assignment functionality in the Report Configurator, you must specify **Labels**.
4. Select **Use a dataset embedded in my report**.
5. Select the new data source that you created earlier.
6. Specify **`Text`** in the **Query Type** field.
7. In the **Query** field specify the query string. The query string consists of the report code with the “/labels” suffix, for example: tdsls440501200/labels.

   Alternatively, you can use the ES Query Designer to define the query string:

   a. Click **Query Designer**.
   b. Specify **`Labels`** in the **Dataset Type** field.
   c. In the tree with packages/modules/reports, select the report.
   d. Click **OK**.
8. Click **OK**.

Creating a dataset for properties

To create a dataset for properties:

1. In the Report Data viewer, right-click the Datasets node and click **Add Dataset**.

2 Specify a name for the dataset. A recommended name is Properties.
3 Select **Use a dataset embedded in my report**.
4 Select the datasource that you created earlier.

   In the **Query** field specify the query string. The query string consists of the report code with the "/properties" suffix. For example: tdsls440501200/properties.

   Alternatively, you can use the ES Query Designer to define the query string:

   a Click **Query Designer**.
   b Specify **`Properties`** in the **Dataset Type** field.
   c In the tree with packages/modules/reports, select the report.
   d Click **OK**.
5 Click **OK**.
   Note: Several report parameters are generated in the **Parameters** folder, in **Report Data**.
6 Right-click the newly created data set and select **Dataset Properties**.
7 Select **Parameters** and select the first parameter.
8 To delete all query parameters, click **Delete** as many times as required.

Setting the language of the report

To set the language of the report:

1 Open the report and click the **Design** tab.
2 Click somewhere outside the report body. The report properties are displayed in the Properties window.

   If the report properties are not displayed, select **`Report`** from the drop-down box in the **Properties** window.
3 Under **Localization**, select the Language property.
4 Select **`<Expression...>`** from the drop-down box.
5 In the Expression editor, remove the existing value.
6 From the **Category** list, select **`Parameters`**.
7 From the **Values** list, double-click **`LanguageISO`**.
8 Click **OK**.

Using labels in reports

This section describes how to manually assign labels to fields in a report. Alternatively, you can use the Infor ES Report Configurator to assign labels to report fields.

See Infor ES Report Configurator on page 47.

To use labels in a report:

1. Open a report and click the **Design** tab.
2. Click the textbox that shows a label, for example a column header.
3. If the textbox contains fixed text rather than a placeholder with a label and an expression, remove the text from the textbox.
4. Right-click the empty textbox and select **Create Placeholder**.

   Note: Using a placeholder is optional, but can be useful. If you do not use a placeholder, you can directly specify the textbox expression. If you use a placeholder, we recommend that you use the label description as a placeholder label.
5. The **Placeholder Properties** dialog box is displayed.
6. In the **Label** field, specify the label description that you want to show, for example currency.

   Note: The **Label** field is only used at development time. The specified label description does not affect the way the label is displayed at runtime.
7. Click the **Expression** button (**fx**) next to the **Value** field. The **Expression** editor is started.

   Note:
   - If you created a placeholder, the 'Label' description is displayed between square brackets in the report design.
   - If you did not create a placeholder, the 'Label' description is displayed as follows: <<Expr>>
8. If the textbox already contains a placeholder, right-click the textbox and select **Expression...**.
9. From the **Category** list, select **`Datasets`**.
10. From the **Item** list, select **`Labels`**.
11. From the **Values** list, select the required label.
12. Double-click the selected label to display it in the expression box.
13. Click **OK** to exit the **Expression** editor.
14. If required, click **OK** to exit the Placeholder Properties.
15. Preview the report. The preview must show the correct label for the selected column header.

Using other label variants

The previous method of obtaining a label from the Labels dataset shows the longest available label of height 1.

To use another variant of that label, you must use a different expression in which you specify the available height and length.

To use other label variants:

1. Open the **Expression** editor for the textbox.
2. Delete the existing expression. Remember the field name, because you will require it in the new expression.
3. Specify an expression in the form of

   ```
   =Code.tt_getLabel([labelcode], [length], [height])
   ```

   Where:
   - [labelcode] is the fieldname as it occurs in the `Labels` dataset, surrounded by quotes.

- [length] and [height] are numbers that specify the maximum available length and height for the label.

For example, you specify this expression:

```
=Code.tt_getLabel("tctcmcs002_ccur", 20, 2)
```

4 Verify that the report contains a reference to the `Infor.ReportingServices.LNDataExtension.dll` assembly.
5 Open the **Report Properties**.
6 Open the **References** page.
7 If the assembly is not yet present, click **Add** in the **Add or remove assemblies** section.
8 Click **Browse**.
9 Click the **Browse** tab.
10 Browse to the folder that contains the Infor ES Data Extension assembly:
   - For Microsoft SQL Server 2012, the folder is named: `C:\Program Files(x86)\Microsoft Visual Studio 10.0\Common7\IDE\PrivateAssemblies`.
   - For Microsoft SQL Server 2016, the folder is named: `C:\Program Files(x86)\Microsoft Visual Studio 15.0\Common7\IDE\PrivateAssemblies`.
11 Select the `Infor.ReportingServices.LNDataExtension.dll` file.
12 Click **OK**.
13 In the **Report Properties**, select the **Code** page.
14 Specify the following block of code if it is not present:

```
Function tt_getLabel(ByVal code As String, ByVal length As Integer,
ByVal height As Integer) As String
return  Infor.ReportingServices.Utilities.LNUtils.GetLabelValue(code,
 _
        Report.Parameters!LanguageISO.Value, _
        Report.Parameters!ConnectionString.Value, _
        length, _
        height)
End Function
```

Note: Adding this code can be automated with the Report Configurator.

See Infor ES Report Configurator on page 47.

15 Preview the report. The report must show a label variant that fits the specified dimensions. If no such label exists, you will see the labelcode instead. In this case, you must specify a greater length or height, because the specified dimensions are too small to show the smallest available label variant.

Using non-report labels

To use labels that are not part of the original report, such as labels that are not present in the `Labels` dataset, you can use the same expression as described in the Using other label variants on page 55 section:

```
=Code.tt_getLabel([labelcode], [length], [height])
```

The labelcode to be used here is the complete labelcode as defined in LN, including the package code. For example, specify "tctccom000.nama" to use the "tccom000.nama" label in package tc.

Note the difference with using labels that are present in the `Labels` dataset: the names of labels in the `Labels` dataset do not contain '.' characters, but use '_' instead.

To use a label that is present in the `Labels` dataset, you must use the name with underscores.

To use a label that is not present in the `Labels` dataset, you must use the labelcode exactly as it is defined in LN.

If the label cannot be found, the labelcode is displayed when the report is rendered. In this case, check that you used a valid labelcode and used the correct size for length and height.

Preparing for deployment

To start the report from within LN, you must change some settings. For example, the connection string should not be a fixed string pointing to some development system, but it should pick up the parameters that are passed on the URL that is used to start the report. These parameters on the URL specify the URI of the JCA daemon running in the bshell on the runtime system, and the location of the datafile on the runtime system.

Note: "Dynamic data sources", that is, data sources with an expression in their connection string, must always be embedded data sources. You cannot use shared data sources with the reports.

Setting the connection string

The connection string should be changed to this expression:

```
="jca_uri=" & Parameters!jca_uri.Value & ";jca_ticket=" & Parame
ters!jca_ticket.Value & ";datafile=" & Parameters!datafile.Value
```

To set the connection string:

1. Right-click the data source in the Report Data window and select **Data Source Properties**.
2. Click the **Expression** button (**fx**) next to the **Connection string** textbox.

3 Remove the existing value (host=...) and enter this expression:

```
="jca_uri=" + Parameters!jca_uri.Value + ";jca_ticket=" + Parame
ters!jca_ticket.Value + ";datafile=" + Parameters!datafile.Value
```

4 Click **OK** to close the expression editor
5 Leave the **Properties** dialog box open for the next step.

Editing Credentials

When you run the report from within LN, you are connecting to an existing bshell. Therefore, login data, that is, your username and password, is not required. The report server requires a user name, otherwise you must set up an execution account for data sources without credentials on the report server. To set up this account, a dummy user name without a password is required.

To edit credentials:

1 Open the **Data Source Properties**.
2 Select the **Credentials** page.
3 Ensure that the **Use this user name and password** option is selected.
4 Remove your user name and password, which are required for preview.
5 Enter a dummy name, such as "x".
6 Leave the password blank.
7 Click **OK** to close the **Properties** dialog box.
8 Click **Save** to save the changes.

Removing default values of parameters

The parameters "jca_uri", "jca_ticket" and "datafile" require a non-empty value in preview mode. When running the report from within LN, these parameters must not have a default value. Instead, their value must be specified on the report URL.

To remove default values of parameters:

1 Open the **Parameters** folder in **Report Data**.
2 Right-click the 'jca_uri' parameter and select **Parameter Properties**.
3 Select the **Default Values** page.
4 Select **No default value**.
5 Click **OK**.
6 Repeat steps 1–5 for the 'jca_ticket' and 'datafile' parameters.
7 Click **Save** to save the changes.

Preparing for development

A deployed report has a generic connection string. To modify or preview a deployed report design, you must change the connection string to a fixed string pointing to a development system.

To change the connection string:

1. Right-click the data source in the **Report Data** window and select **Data Source Properties**.
2. Click the **Expression** button (**fx**) next to the **Connection string** textbox.
3. Remove the existing value (="jca_uri=") and specify this expression:

```
host=server1;bse=/erpln/bse;bshell=bshell;protocol=baanlogin
```

4. Specify a “static’ connection string for the data source. This connection string is a semicolon-separated list of key-value pairs, which must include values for the host and the bse.

   These are the optional keys:

   - Bshell
   - Protocol
   - Port

   If omitted, the optional keys will have these default values:

   - bshell=bshell
   - protocol=rexec
   - port=512

   If protocol=BaanLogin is specified, the default value for port is 7150.

   This is an example of a static connection string:

```
host=server1;bse=/erpln/bse;bshell=bshell;protocol=baanlogin
```

   Caution: Static connection strings are only meant to be used at development time. At runtime, a "dynamic" connection string is required.

5. Click **OK** to close the expression editor.

Main reports and subreports

In the Infor ES Reporting plug-in, you can create main reports and corresponding subreports. See Creating a Main report with Subreports on page 20.

This section describes how to convert a stand-alone report to a subreport, and how to prepare main reports and subreports for deployment or development. Alternatively, you can use the Infor ES Report Configurator to perform these actions. See Infor ES Report Configurator on page 47.

Converting stand-alone reports to subreports

To convert a stand-alone report to a subreport:

1. Add these parameters to the report:
   - guid
   - datafile

   Select the **Allow null value** check box for both parameters.
2. Add this text to the report's connection string:

```
;guid=" & Parameters!Guid.value & ";datafile=" & Parameters!datafile.val
ue
```

   This is an example connection string:

```
host=server1;bse=/erpln/bse;bshell=bshell;protocol=baanlogin;guid=" &
 Parameters!Guid.value & ";datafile=" & Parameters!datafile.value
```

Preparing main reports for deployment

To prepare a main report for deployment:

1. Add these parameters to each link to a subreport:

Parameter name	Value
jca_uri	=Parameters!jca_uri.Value
jca_ticket	=Parameters!jca_ticket.Value

2. Perform the steps that are described in Preparing for deployment on page 57.

Preparing subreports for deployment

To prepare a subreport for deployment:

1. Modify the general properties of the guid parameter:
   a. Clear the **Allow null value** check box.
   b. In the **Select parameter visibility** field, specify Hidden.

2 Perform the steps that are described in Preparing for deployment on page 57. For subreports, only replace the static part of the connection string. The dynamic part must be retained. The connection string of a subreport that is ready for deployment looks like this:

```
="jca_uri=" + Parameters!jca_uri.Value + ";jca_ticket=" + Parame
ters!jca_ticket.Value + ";datafile=" + Parameters!datafile.Value +
";guid=" + Parameters!guid.Value
```

Preparing main reports and subreports for development

To prepare a main report or subreport for development, reverse the actions mentioned in Preparing main reports for deployment on page 60 and Preparing subreports for deployment on page 60 regarding the parameters and connection string.

Chapter 8: Viewing and printing reports directly from LN

To print a deployed report from LN:

1. Log on to LN.
2. Start the session to which the LN report is linked.
3. Complete the session's form. Ensure that sufficient data is included in the selection ranges.
4. Print the report. In the **Select Device (ttstpsplopen)** session, click the **Display** tab and select an appropriate device.

Note:

- Ask your administrator for the names of the appropriate devices.
- The device must be of device type "Microsoft Reporting Services," and must have one of these arguments:
  - "-server [server name]"

    This device renders the report on the screen.
  - "-server [server name] -printer [printer name]"

    This device directly sends the report to the printer without rendering the report on the screen first.
- For details on how to create such a device, see the *Infor Enterprise Server Plug-in for Microsoft SSRS Administration Guide*.