Infor LN Studio Administration Guide

Release 10.6.x

Publication Information

Release: Infor LN 10.6.x
Publication Date: October 19, 2018
Document code: ln_10.6.x_lnstudioag__en-us

Contents

About this Guide

Document Summary

This document describes how administrators can install and configure Infor LN Studio and Infor LN Project Server.

Related documents

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

- *Infor LN Studio Application 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.

If we update this document after the product release, we will post the new version on the Infor Support Portal. To access documentation, 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: Upgrade notice

Important note for users upgrading from Application Studio 8.4.2 to Infor LN Studio 10.6.x

If you have implemented Application Studio 8.4.2 already, the following additional steps must be performed before you can start to use version 10.6.x:

1. Define the development environment(s). This is a one-time action for the system administrator. In version 8.4.2 you could only use one development server per workspace. This was changed in version 8.5: you can now connect to multiple development servers while using the same workspace.

   To define those development servers:

   a. Install LN Studio 10.6.x. The development server and project server must have been upgraded already to Enterprise Server 8.5 or later.

      See Installing LN Studio on page 9.

   b. Start LN Studio, using a new workspace.

   c. Configure the Project Server connection.

      See Defining connectivity settings on page 11.

   d. Define the development environment(s) in the **Application Explorer** view. See the *Infor LN Studio Administration Guide*.

2. The system administrator must update the projects in the **Software Project Explorer**: For each project, fill the **Development Environment** field.

3. All developers must perform the following steps:

   a. Start in a new workspace.

   b. Specify Project Server and Debug *connections*.

      See Defining connectivity settings on page 11.

   c. Open the activities they were working on in the new workspace.

   d. Use the recover workspace functionality to retrieve the modified components from the development server.

      See Recovering an activity.

   You are prompted to define the dynamic connection points when they are required.

   For details on dynamic connection points, see Defining dynamic connection points.

Chapter 2: LN Studio overview

LN Studio is a development platform for LN and is implemented in the *Eclipse* framework.

Eclipse is a Java based development environment with many plug-ins available. LN Studio adds several plug-ins to Eclipse that supply additional functionality such as editors, views, wizards, and perspectives.

You can use LN Studio to perform these actions:

- Create and edit LN software components through the Eclipse workbench. You create components through the **Create a New Infor LN Software Component** wizard. You can edit components through various multipage editors.
- Run and debug sessions through the Eclipse workbench.
- Create and test Infor Business Interfaces.

Caution: The Java Connector Architecture (JCA) `JCAAdapter4ERPln.jar` library that is embedded in this product is copyright and proprietary to Infor and contains interfaces and/or APIs that are strictly private to Infor. These interfaces and/or APIs cannot be used by external applications, devices, and/or software libraries. Usage of the JCA library will be monitored and, if used illegal, will cause a non-compliancy situation.

Note: A number of screenshots in the documentation may be based on previous application releases. They can differ slightly from your application screens. However, the described functionality is similar.

Chapter 3: Installation and configuration

First installation

Prerequisites

This section describes the prerequisites to run *LN Studio*.

Licenses

To use LN Studio, you need these licenses:

- A license for the Adapter for LN. Take care of the following:
  - If you already have an Adapter for LN license for product ID 7013, you do not need a new license.
  - If you do not have a license yet, obtain a license for product ID 7056.
- An LN Development license, product ID 10146. This is required if you want to create or modify tables, domains, UI scripts, functions, or libraries. This license is also required if you want to modify and generate Business Interface implementations.

To obtain a license, add the corresponding product ID in the Infor Solution License Manager (SLM) and request a license for it.

Prerequisites for the LN server(s)

For the LN server the following is required:

- Infor Enterprise Server 8.5 or later.
- Infor Enterprise Server AddOn 8.5 or later.

Note: The development repository and the development projects and activities, that are defined in *Project Server*, can reside on the same LN server. However, you can also configure a dedicated server to store the project data. The prerequisites for such a dedicated project server are:

- Infor Enterprise Server 8.5 or later.
- Infor Enterprise Server AddOn 8.5 or later.

Note: On the LN server(s), the `FGL_STUDIO_ENABLED` environment variable must be set to 1.

Prerequisites for the client PC

Prerequisites for the client PC are as follows:

- 2 GB RAM memory (recommended)
- Microsoft Windows 7 or later
- Java Runtime Environment (JRE) 1.7 (Java 7) or later, 64-bits

  You can download the JRE software from the http://java.com/download website.

Preparing the installation

Note: This section is intended for system administrators.

Preparing the first installation

Before users can install LN Studio, the system administrator must complete these steps:

1. Download or copy the `InforLNStudio_<version number>.nnn-x86_64.zip` file, which contains the LN Studio software.

   You can download this file from solution 1561993 on the http://www.infor.com/inforxtreme site.

   `nnn` represents a build number, such as "0023".
2. Ensure users can access the zip file. For example, put the file on a network share.

Preparing the installation of updates

Before users can install updates of the LN Studio software, the system administrator must complete these steps:

1. Download a new LN Studio zip file from the http://www.infor.com/inforxtreme site.
2. Ensure the users can access the zip file.
3. Notify the users that new updates are available.

Installing LN Studio

This section describes how you can install LN Studio.

Note:

- For information on the prerequisites for the installation of LN Studio, see Prerequisites on page 8.
- Before you can install LN Studio, the system administrator must give you access to the zip file with the LN Studio software.

To install LN Studio:

1. Create an installation folder.

   Create a new folder on your machine.

The recommended installation directory is: `C:\Infor\LN\LNStudio`.

Note: On Windows 7 or later, do not install LN Studio in a Windows protected folder, such as `C:\Program Files` or `C:\Program Files (x86)`.

2 Open the `InforLNStudio_<version number>.nnn-x86_64.zip` file, which contains the LN Studio software, and extract all contents of the file to the new installation folder.

`nnn` represents a build number, such as "0023".

Note: If you upgrade from Application Studio 8.4.2 to LN Studio 10.6.x, you must perform the actions described in the Upgrade notice on page 6.

Post installation steps

1 Select the workspace.

See Selecting the workspace on page 11.

2 Configure static connection points.

When you have installed LN Studio, you must configure these static *Connection Points*:

- Project Server
- Debug

See Defining connectivity settings on page 11.

3 Configure dynamic connection points.

For each LN server, on which you want to perform administrative tasks, you must configure the following dynamic connection point:

Administrator

For each *software project* you develop software in, you must configure these dynamic connection points:

- Development Address
- Runtime Address

Note: You cannot define these connection points in advance. When working in LN Studio, a prompt is displayed whenever such a connection is required, but not yet configured. At that moment a wizard to configure the required connection point is started.

4 Configure connection points for Business Interface development.

To use interface projects, configure these connection points:

- RuntimeRepositoryConnection
- TestServerConnection

See Defining connectivity settings on page 11.

Note: These connection points are not required if the interface project is using "related software projects".

5 Define LN Studio preferences for Application development.

a Select **Window > Preferences**.

b In the left pane of the **Preferences** window, select **Infor LN Studio Application**. For details about the preferences, see the "Application Preferences" section in this guide or the dialog's online help.

6 Define LN Studio preferences for Business Interface development.
   a Select **Window > Preferences**.
   b In the left pane of the **Preferences** window, select **Infor LN Studio Integration**. For details about the preferences, see the "Integration Preferences" section in this guide or the preference page's online help.

Selecting the workspace

To select the workspace:

1 To start LN Studio, run the `eclipse.exe` executable in the LN Studio installation folder.

   The **Workspace Launcher** dialog box is displayed.

2 Specify this information:

   Workspace
   The location of the workspace. The workspace is the directory, usually on your own PC, where your work is stored.
   It is discouraged to locate the workspace in the directory where the LN Studio software or the Eclipse software is installed. It is better to select a different directory, for example under `C:\data`. This results in a better overview of the directory structure on your desktop, and makes the workspace independent of the versions of the LN Studio and Eclipse software.

   Use this as the default and do not ask again
   Select this check box to prevent that the **Workspace Launcher** dialog box is displayed again.

3 Click **OK**.

4 Allow installation of BW and Dynamic Form Editor.

   LN Studio requires Infor LN BW activation for the development and runtime addresses. LN Studio also requires the Infor LN Dynamic Form Editor (DFE) to develop session forms.

   If BW and DFE are not installed on your computer, or older versions are installed, an automatic installation of these components is performed when you start LN Studio for the first time.

   If User Account Control (UAC) is enabled, confirmation dialog boxes are displayed during the BW and DFE installations: you are prompted to allow the BW and DFE setup programs to make changes to your computer.

   Click **Yes** in both dialog boxes.

A single Workbench window is displayed.

Note: If you want to select another workspace after the Workbench is started, select **File > Switch Workspace > Other**.

Defining connectivity settings

This section describes the procedure to define connectivity settings for the LN Studio.

Infor LN Studio uses *Connection Points* to connect to the LN server that contains the LN *applications* and the development repository, and to the server on which the software projects are stored.

This table shows the static connection points you must configure to define the connectivity settings:

Connection point	Description
ProjectServer	This connection point is used to connect to the Project Server. The Project Server contains projects and activities that are used in the Infor LN Studio. The Project Server connection point is used, for example: • When a *project manager* creates a new project, or a new activity in the **Software Project Explorer** view. • When a *software engineer* starts the **Open an Activity** wizard from the **Activity Explorer** view.
Debug	When a software engineer debugs a session from the LN Studio, the debugger on the LN server uses this connection point to send messages to the LN Studio on the client PC.
RuntimeRepositoryConnection	This connection point is used for exchanging business metadata with the application server. This connection is not required if the interface project is using "related software projects".
TestServerConnection	This connection point is used when testing Business Interface implementations. This connection is not required if the interface project is using "related software projects".

Configuring static connection points

To configure the connection points:

1 Display the standard connection points

a On the Eclipse **Windows** menu, select **Preferences**. The **Preferences** dialog appears.

b In the tree that is displayed in the left pane of the dialog, select **Infor LN JCA Connectivity**. The **Infor LN JCA Connectivity** dialog is displayed. The dialog shows the standard Debug and Project Server connection points, which were defined by LN Studio by default.

The **Infor LN JCA Connectivity** dialog is part of the connectivity plug-in. For details, see the dialog's online help and to the online help of the Infor Connectivity plug-in.

2 Configure the connection points

To configure a connection point, select the connection point and click **Edit**. The **Configure Connection point** wizard starts. Specify the properties for the connection points and close the wizard.

This table shows points of attention per static connection point:

Connection point	Points of attention
ProjectServer	Select the appropriate activation type: • To connect to an on-premises installation of LN, select the **BW**, **Rexec**, or **BaanLogin** activation type, and specify the corresponding settings. • To connect to a multi-tenant cloud installation of LN, specify the **HTTP-OAuth2** activation type after setting up a "Cloud Environment". To connect to a single-tenant cloud installation of LN, specify the **HTTP-Basic** activation type. See "Specifying additional settings for cloud development". The company number for this connection point must be **000**. The application and project data accessed through these connection points can reside in the same LN environment. If so, you can share these connection points. As a result, only one bshell is used to connect to the server, instead of two, which improves the performance.
Debug	Select the **Socket-In** activation type. Select a free local port number, for example 7900. **Note:** In a multi-tenant environment you cannot debug with LN Studio. You can use the Debug Workbench, see the **Debug and Profile 4GL (ttadv1123m000)** session.
RuntimeRepositoryConnection	Select the **BaanLogin** (recommended), **Rexec**, or **BW** activation type, and specify the corresponding settings. The company number for this connection point must be **000**.
TestServerConnection	Select the **BaanLogin** (recommended), **Rexec**, or **BW** activation type, and specify the corresponding settings.

The **Configure Connection point** wizard is part of the connectivity plug-in. For details, see the wizard's online help and the online help of the Infor Connectivity plug-in.

Note:

- You can use the **Preferences** dialog box to set various other user preferences for the LN Studio workbench. See "Application preferences", "Integration preferences", and "Connectivity preferences" in the *Infor LN Studio Administration Guide*.
- To run or debug sessions after creating or modifying the Debug connection point, you must restart LN Studio.
- In a multi-tenant environment you cannot debug with LN Studio. You can use the Debug Workbench, see the **Debug and Profile 4GL (ttadv1123m000)** session.

Specifying additional settings for multi-tenant cloud development

You can connect to an LN installation in the cloud. To achieve this, you must set up an HTTPS connection to an LN UI server where the LN Client Service is enabled. The required authorization settings are usually stored in a `.ionapi` file. Your administrator can distribute this file in multiple ways:

- Send the file by email.
- Publish the file on a network share or on an HTTP server in your local intranet.

The settings in the `.ionapi` file are used to define a "Cloud Environment".

Adding a cloud environment

1. Select **Window > Preferences**.
2. In the tree in the left pane of the dialog box, expand **Infor LN JCA Connectivity** and select **Cloud Environments**.
3. Click **Add**. The **Configure Cloud Environment** wizard starts.
4. If the `.ionapi` file is present on your file system or on a network share, select **Import from file**. Then click **Browse** to locate the `.ionapi` file. Alternatively, specify the path to the `.ionapi` file directly in the text box.

   If the `.ionapi` file is published on an HTTP server, select **Import from URL** and specify the URL to the file.
5. Click **Import**. The remaining fields on the wizard page are filled with settings from the `.ionapi` file.
6. Optionally, change the **Environment Name**. Leave the rest of the settings unchanged. Tip: copy the value of the **Gateway URL** to the clipboard; you need this later on.
7. Click **Finish**.

Adding a server certificate to the trust store

Because you are creating a Secure HTTPS connection to LN UI, LN Studio must know whether the server is trusted. Therefore, you must register the trusted server certificates in a trust store. The trust store is a file in "Java Key Store" format. This file contains a "global" configuration that is valid for all workspaces. Therefore, the file is located outside of the LN Studio installation folder, and outside of the workspace folder. A good location is your "home folder".

To add a server certificate to the trust store:

1. Select **Window > Preferences**.
2. In the tree in the left pane of the dialog box, expand **Infor LN JCA Connectivity** and select **Trusted SSL Certificates**.
3. If you do not yet have a trust store, complete these steps:
   a. Click **Browse**. Browse to the folder where you want to create the trust store, such as `C:\Users\<YourName>` .

b Specify the file name. Use a `.jks` extension for the file name. For example, specify **`LNStudioTrustStore.jks`**.

c To complete the browse dialog box, click **Open** or press **Enter**.

4 Optionally, specify a password for the trust store. If you leave the field blank, a default password is used.

5 Click **Apply**. If the trust store did not yet exist, you are prompted whether you want to create a new one. Click **Yes**.

6 The certificates in the chosen trust store are listed in the **Certificates** table, and the **Add** button is enabled now. If you have a copy of the certificate in a file, you can click **Add** to add that certificate. Usually, this is not the case, so you must use another way.

7 In the **Check Trusted URL** field, specify the HTTPS URL of the LN UI server you want to use. You can go back to the **Cloud Environments** preference page, select the configured cloud environment, click **Edit**, and copy the **Gateway URL**. The **Gateway URL** is the first part of the URL to the LN UI server.

8 Click **Check**. Usually, a "Check failed" dialog, which suggests to import the certificate presented by the server, is displayed. Click **Yes**. The **Import Certificate** dialog box is displayed.

9 In the **Import Certificate** dialog box, the **Server Certificates** drop-down list may contain more than one certificate.

The HTTPS server presents a chain of certificates, starting with its own certificate. Next is the certificate of the Certification Authority (CA) that issued the server certificate, followed by the issuer of the CA certificate, and so on. The last certificate in the chain is usually a "root CA certificate", issued by itself.

You must import the last certificate. This certificate is already selected in the drop-down list. Details of the selected certificate are displayed in the dialog box.

10 In the **Alias** field, specify a short name for the certificate. This alias is stored in lower case in the trust store. Click **Finish**. The certificate is now present in the **Certificates** table.

11 Click **Check** again. A "Check succeeded" dialog box is displayed.

12 Click **OK**. The **Preferences** dialog box is closed.

Configuring connection points

You must configure these connection points:

- The static connection point for the Project Server
- The dynamic connection points for Administrator, Development Address, and Runtime Address.

These connection points must use an HTTPS connection to an LN UI server that provides access to an LN environment in the cloud.

Example - Project Server connection point

For example, to configure the Project Server connection point:

1 Select **Windows > Preferences**.

2 Select **Infor LN JCA Connectivity**.

3 In the table with connection points, select **`ProjectServer`** and click **Edit**. Alternatively, double-click the row with **`ProjectServer`**. The **Configure Connection point** wizard starts.
4 In the **Activation Type** drop-down list, select **`HTTP-OAuth2`** and click **Next**.
5 Select the correct **Cloud Environment** from the drop-down list.
6 The **Service Endpoint URL** is automatically filled with the URL for the LN UI server.
7 For a project server connection, ensure the **Company** field contains `0`.
8 To complete the configuration, click **Finish**.
9 To test the connection, select the connection point and click **Ping**. A dialog box containing a login page is displayed. During normal usage of LN Studio, this login page is displayed the first time a connection to the cloud is established.

   Follow the instructions on the page. These instructions are provided by the authorization server of the cloud. If everything is okay, the connection test succeeds. If the test fails, errors are reported: the configuration is incorrect, or you specified invalid credentials on the login page.

Specifying additional settings for single-tenant cloud development

You can connect to an LN installation in the cloud. To achieve this, you must set up an HTTPS connection to an LN UI server where the LN Client Service is enabled.

Configuring connection points

You must configure these connection points:

- The static connection point for the Project Server
- The dynamic connection points for Administrator, Development Address, and Runtime Address

These connection points must use an HTTPS connection to an LN UI server that provides access to an LN environment in the cloud.

Example - Project Server connection point

For example, to configure the Project Server connection point:

1 Select **Windows > Preferences**.
2 Select **Infor LN JCA Connectivity**.
3 In the table with connection points, select **`ProjectServer`** and click **Edit**. Alternatively, double-click the row with **`ProjectServer`**. The **Configure Connection point** wizard starts.
4 In the **Activation Type** drop-down list, select **`HTTP-Basic`** and click **Next**.
5 In the **Service Endpoint URL** field, specify the URL for the LN UI server. Your administrator can supply the correct value.

   The LN Client Service should be enabled. For details on how to enable the LN Client Service, see the *Infor LN UI Administration Guide*.
6 In the **LN UI Environment** field, specify the name of the environment as defined on the LN UII server. Your administrator can supply the correct value.

7 For a project server connection, ensure the **Company** field contains `0`.
8 Leave the **Command** field blank, unless your administrator has instructed otherwise.
9 In the **User** and **Password** fields, specify the correct values.
10 To complete the configuration, click **Finish**.
11 To test the connection, select the connection point and click **Ping**.

Installing LN Studio updates

This section describes how you can install updates for LN Studio.

Note: Before you can install updates, the system administrator must provide a new LN Studio zip file.

To install updates:

1 Open the `InforLNStudio_<version number>.nnn-x86_64.zip` file, which contains the new software.

`nnn` represents a build number, such as "0023".

2 Extract all contents of the file to your existing LN Studio installation folder, such as `C:\Infor\LN\LNStudio`. Ensure the **Overwrite existing files** check box is selected!

Chapter 4: Administrative tasks

Preparing for multi-tenant cloud development

To connect to an LN environment that runs in the multi-tenant cloud, you must perform some special actions. You cannot use BW, Rexec, or BaanLogin to connect to the LN system. Instead you must use HTTPS to connect to an LN UI server in the cloud. The LN UI server provides access to the LN environment. The HTTPS connection requires OAuth 2.0 authentication. All LN Studio users must know the OAuth 2.0 related settings, otherwise they cannot establish a successful connection.

To prepare for multi-tenant cloud development:

1. Register LN Studio as authorized app in the cloud. To perform the registration, use the ION API application within the Infor Ming.leTM portal. See the *Infor ION API Administration Guide*.
   a. Add a non-Infor authorized app of the Web Application type.
   b. Add a description. Specify **`LN Studio`**.
   c. Set the **Redirect URL** to **`oob://localhost/lnstudio`**.
   d. You can set the **Authorized Javascript Origins** to **`http://localhost`**.
   e. Download the credentials.

      This results in a file with `.ionapi` extension. All LN Studio users require this file to create a connection to the cloud.
2. Verify that the settings in the `.ionapi` file are correct. To do that, configure your Project Server connection point.

   See Specifying additional settings for multi-tenant cloud development on page 14.
   a. Create a cloud environment using the `.ionapi` file.
   b. Start the Configuration wizard for the Project Server connection point.
   c. On the second page, specify the correct LN UI environment name.
   d. Verify that the wizard specified the correct **Service Endpoint URL**. If not, you must instruct the LN Studio users to specify a correct value here.
   e. Finish the wizard and verify that the ping succeeds.
3. If the `.ionapi` settings are correct, distribute this file to all LN Studio users. You can distribute the file through email, or you can publish the file on a network share or on a local HTTP server. The `.ionapi` file contains sensitive information, so ensure it is not visible to unauthorized persons.
4. Supply the LN Studio users with the information they require to configure a cloud connection. As a minimum, they must know the location of the `.ionapi` file and the name of the LN UI environment.

Defining a development environment

This topic describes how to define a development environment. For each development environment, you can define one or more *Applications*.

To define a development environment:

1. Start LN Studio
2. Open the **Project Server** perspective.

   Select **Window > Open Perspective > Other**. A list of perspectives is displayed. Select **Project Server** and click **OK**.

   The **Application Explorer** is displayed.
3. Specify the development environment properties

   On the view's toolbar, click **Create a new development environment**. The **New Development Environment** dialog is displayed. In this dialog, specify the development environment properties, such as Hostname and BSE path. See the dialog's online help.

   Save the environment and close the dialog. The new environment is displayed in the **Application Explorer** view.
4. Configure the Administrator connection point for the new environment

   Complete these steps:

   a Click on the triangle icon of the new development environment. Two sub folders, `Applications` and `Localizations`, are displayed.

   b Click on the triangle icon of the `Applications` or `Localizations` folder. You are prompted to configure an Administrator *Connection Point*.

   c In the question window, click **Yes**.The **Configure Connection point** wizard starts.

   d Specify the properties for the connection point.

   Points of attention

   - Select the appropriate activation type:
     - To connect to an on-premises installation of LN, select the **`BW`**, **`Rexec`**, or **`BaanLogin`** activation type, and specify the corresponding settings.
     - To connect to a multi-tenant cloud installation of LN, specify the **`HTTP-OAuth2`** activation type.
     - To connect to a single-tenant cloud installation, specify the **`HTTP-Basic`** activation type.
   - Select company number **`000`**.
   - The application and project data can reside in the same LN environment. If so, you can share the Administrator and ProjectServer connection points. As a result, only one bshell is used to connect to the server, instead of two. This improves the performance.

   For details, see the wizard's online help.

Defining an application

This topic describes how to define an *Application*. For each application, you can define a project in which the software engineers can develop their software components.

Note: Before you can define an application, you must define one or more development environments and the corresponding Administrator Connection Points.

See Defining a development environment on page 19.

To define an LN application:

1. Start LN Studio
2. Open the **Project Server** perspective.

   Select **Window > Open Perspective > Other**. A list of perspectives is displayed. Select **Project Server** and click **OK**.

   The **Application Explorer** is displayed.
3. Specify the application properties

   Complete these steps:

   a. In the view, right-click the development environment for which you want to define an application.
   b. On the shortcut menu, select **New Application**. The **New Application** dialog is displayed.
   c. Specify the application properties, such as application name, description, and release.

      For details, see the dialog's online help.

Note: You can now define one or more projects for the new application. See Defining a software project on page 43.

Code freeze procedure

A code freeze is performed when the development process reaches a certain milestone. Development is stopped in the current VRC and started in a new derived VRC. Certain activities might still be open and developers may want to continue working in the new VRC, on the components in these activities.

The code freeze procedure handles the move of these components and the update of the involved activities and configuration management settings.

Two types of code freeze are distinguished:

- Intermediate code freeze.

  An existing software project is linked to a new VRC, which is used as the export VRC for the application.

  See Intermediate code freeze on page 21.
- Final code freeze.

  A new application and project and a corresponding VRC are created. All activities are moved to the new project.

  See Final code freeze on page 23.

Intermediate code freeze

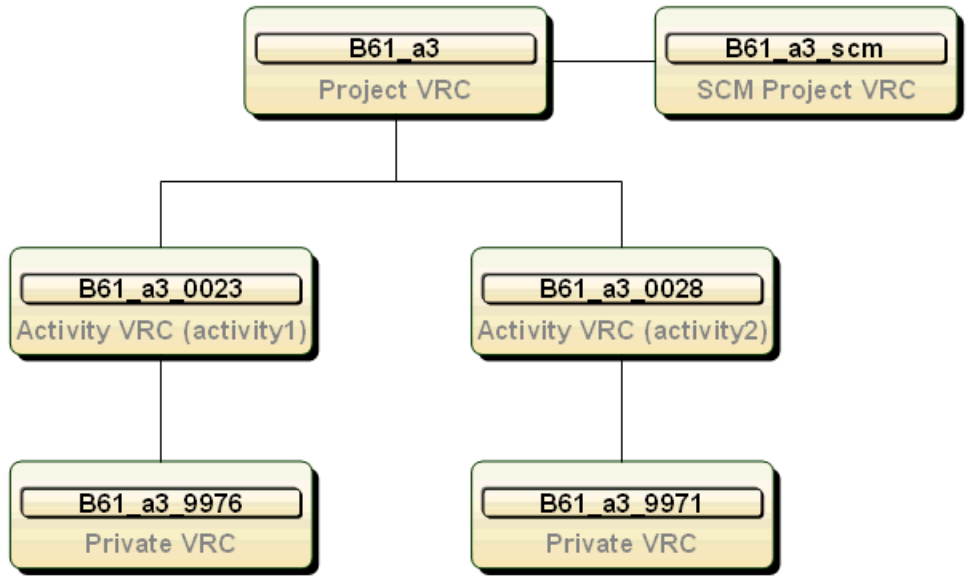
When you perform an intermediate code freeze, the current project is linked to a new project VRC, which is used as the export VRC for the application.

Note:

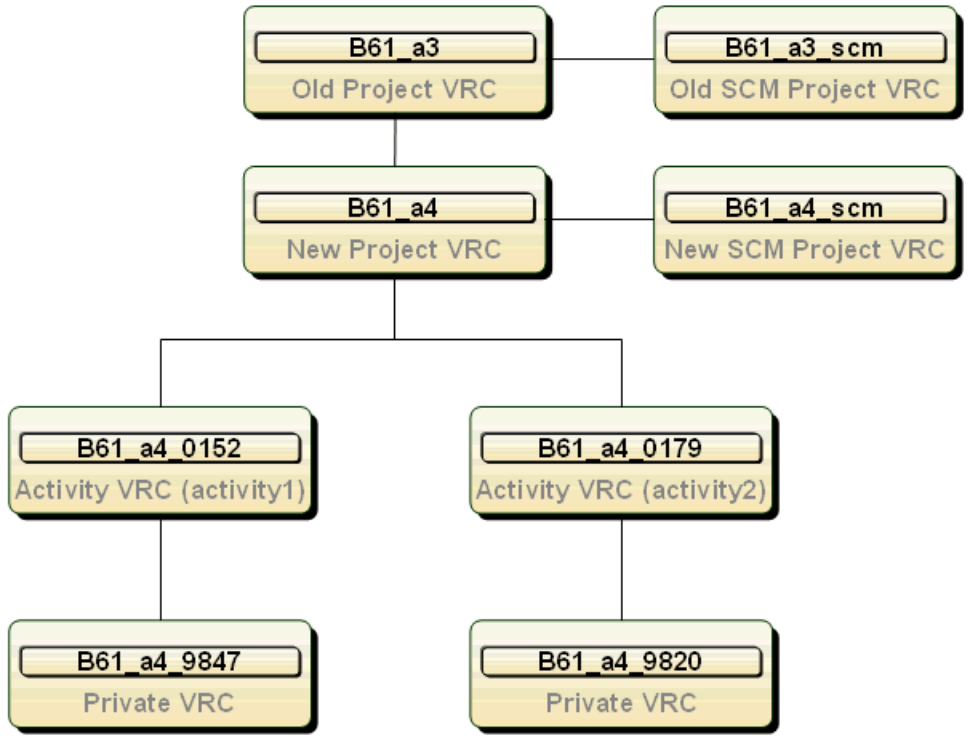
- The code freeze creates new activity VRCs derived from the new project VRC and, if SCM is active, new private VRCs derived from the new activity VRCs.
- All involved activities stay linked to the current project.
- The application and its *Base VRC* do not change.

Example

This diagram shows a VRC structure before an intermediate code freeze. SCM is active.



This diagram shows the VRC structure after an intermediate code freeze.



The new VRC structure contains:

- A new project VRC and a new SCM project VRC.
- New activity VRCs derived from the new project VRC.
- New private VRCs derived from the new activity VRCs.

Intermediate code freeze procedure

1 Create new project VRC.

Start the **Package VRCs (ttadv1511m000)** session and create a new project VRC. Then, start the **Directories of Software Components (ttadv1115m000)** session and specify the directories to store the software components of the new VRC.

2 If SCM is active for the old project VRC, activate SCM for the new project VRC.

a Start the **(De)activate SCM and Component Management by Package VRC (ttscm0501m000)** session. The session shows a list of VRCs.

b Select the project VRC. On the *appropriate menu*, select **Activate SCM (and CM)**. The **Activate SCM and Component Management by Package VRC (ttscm0110s000)** session starts.

c Select the new SCM VRC and click **OK**.

See "To use the Software Configuration Management system (SCM)" in the LN Web Help.

3 Generate new VRCs for the involved activities.

a Start the **Code Freeze (ttadv1222m000)** session.

b Clear the **Final Code Freeze** check box.

c In the **Source VRC** field, select the old project VRC.
d In the **Target VRC** field, select the new project VRC.
e Select the full range of activities.
f Select the **Update Export VRC in Base VRC's** check box, so the new project VRC becomes the export VRC of the application's base VRC.

If this check box is cleared, you must use the **Base VRC's (ttpmc0110m000)** session to assign the new project VRC as the export VRC.

Note: LN Studio users must not recover their activities before the new project VRC is assigned as the export VRC.

g Click **Move**. The session performs these actions:
- Creates new activity VRCs, derived from the new project VRC.
- Creates, if SCM is active, new private VRCs derived from the new activity VRCs.
- Moves components from the old activity and private VRCs to these new VRCs.
- Removes the old activity and private VRCs.

4 Inform the LN Studio users the code freeze is finished.

The users must recover their workspace.
- **Note:** This recovery must be performed after the new project VRC is assigned as the export VRC for the application's base VRC. See step 3 in this procedure.
- The users must select the **Recover existing activity files** check box in the **Recover activity** dialog box.

See Recovering an activity.

Final code freeze

When you perform a final code freeze, you must create a new application and a new project. The new project is linked to a new project VRC, which is used as the export VRC for the application.

Note:
- The code freeze creates new activity VRCs derived from the new project VRC and, if SCM is active, new private VRCs derived from the new activity VRCs.
- All involved activities are moved to the new project.

Final code freeze procedure

1 Create new project VRC.

Start the **Package VRCs (ttadv1511m000)** session and create a new project VRC. Then, start the **Directories of Software Components (ttadv1115m000)** session and specify the directories to store the software components of the new VRC.

2 If SCM is active for the old project VRC, activate SCM for the new project VRC.

a Start the **(De)activate SCM and Component Management by Package VRC (ttscm0501m000)** session. The session shows a list of VRCs.

b Select the project VRC. On the *appropriate menu*, select **Activate SCM (and CM)**. The **Activate SCM and Component Management by Package VRC (ttscm0110s000)** session starts.
c Select the new SCM VRC and click **OK**.

See "To use the Software Configuration Management system (SCM)" in the LN Web Help.

3 Define new base VRC.

Start the **Base VRCs (ttpmc0110m000)** session and create a new *Base VRC*. In the **Export VRC** field, select the new project VRC.

4 Start LN Studio and define a new application.

Note: Take care of the following when you specify the properties of the new application in the **New Application** dialog box:

- In the **Release** field, select the new base VRC created in the previous step.
- If SCM is active for the new project VRC, select the **Use SCM** check box. If SCM is not active, clear the check box.
- In the lower part of the dialog box, specify the appropriate packages and languages.

See Defining an application on page 20.

5 Start LN Studio and define a new software project.

Note: In the **Application** field in the **Create a Software Project** dialog box, select the new application created in the previous step.

See Defining a software project on page 43.

6 Move activities to the new project.

a Log on to the project server and start the **Final Code Freeze (tlprj1220m000)** session.
b In the **Source Project** field, select the old project.
c In the **Target Project** field, select the new project.
d Select the full range of activities.
e Click **Move**. The session moves the activities to the new project.

7 Generate new VRCs for the involved activities.

a Start the **Code Freeze (ttadv1222m000)** session.
b Select the **Final Code Freeze** check box.
c In the **Source VRC** field, select the old project VRC.
d In the **Target VRC** field, select the new project VRC.
e Select the full range of activities.
f Click **Move**. The session performs these actions:
  - Creates new activity VRCs derived from the new project VRC.
  - Creates, if SCM is active, new private VRCs derived from the new activity VRCs.
  - Moves components from the old activity and private VRCs to these new VRCs.
  - Removes the old activity and private VRCs.

8 Inform the LN Studio users the code freeze is finished.

For each activity involved, users must complete these steps:

a Delete the activity, including all files, from the local workspace. To do this, right-click the activity in the **Activity Explorer** and select **Delete**. A confirmation question appears. Select **Also delete contents** and click **Yes**.

b Open the activity again.

Note: In the **Open an Activity - Select Project** dialog box, the users must first select the new software project from the list, and then select the activity.

c Recover the activity.

Note: The users must select the **Recover existing activity files** check box in the **Recover activity** dialog box.

See Recovering an activity.

Chapter 5: Application preferences

Under **Infor LN Studio Application** in the preference pages, you can specify preferences related to application development. To open the preference pages, in Eclipse, select **Windows > Preferences**.

Preferences - Editor

Use this dialog window to specify your script editor preferences.

These settings only apply to the LN Studio **Script Editor**.

Marker Identifier
Determines the text that is added in the source code, when you add maintenance comments.
Allowed values

Activity Name	The name of the current *Activity*.
Activity Requirement ID	The ID of the business requirement to which the current activity belongs.
Project Name	The name of the current *Software Project*.
Project Requirement ID	The ID of the business requirement to which the current project belongs.

Example

Marker Identifier = "Activity Name", and your current activity is "myactivity".

In the **Script Editor**, you mark lines as new. The following maintenance comments are added:

- Behind the first new line: " `|#myactivity.sn` ".
- Behind the last new line: " `|#myactivity.en` ".

See "Comments" in the *Infor LN Studio Application Development Guide*.

Show Mark Occurrences
If this check box is selected, Mark Occurrences is enabled, so occurrences of the selected element are marked in the source code. See "Mark Occurrences" in the *Infor LN Studio Application Development Guide*.

Highlight Matching Brackets
If this check box is selected, the script editor can highlight matching brackets in the source code.

Usage: Place the cursor directly behind a bracket. The corresponding bracket is highlighted automatically.

This functionality applies to the following types of brackets:

- Parentheses: (,).
- Braces: {, }.

Remove Trailing Whitespaces on Save
If this check box is selected, the script editor removes trailing whitespaces (space and tab) of each line in the source code during save actions.

Delay key manager
The number of milliseconds after which the editor's parser will react to the last key-event.

This only works if the editor is in asynchronous mode.

Max LOC of syn. behaviour
The maximum number of lines of code for which the editor can run in synchronous mode. For the remaining lines, the editor switches to asynchronous mode.
Note:

In synchronous mode, the parser directly reacts to each key-event. As a result, all changes are colored immediately.

If you enter 0 in this field, the editor always runs in asynchronous mode.

Max LOC of parser
The maximum number of lines of code the parser can handle.

If a script has more lines, the parser does not react anymore. Therefore, all editor functionality that requires the parser's output drops out. This enhances the performance when editing very large scripts.

Max LOC of syntax highlighting
The maximum number of lines of code to which syntax highlighting can be applied.

If a script has more lines, the parser does not react anymore. Therefore, all editor functionality that requires the parser's output drops out. This enhances the performance when editing very large scripts.

Background Color
Use this field to set the background color for your editor.

Choose one of the following options:

`System Defaults`	The default background color is white.
`Custom`	Choose the color from a basic color schema or define custom colors.

Note:

The editor only uses this color for source code of checked out components. If a component is not checked out, the source code is read-only and the editor uses the **`ReadOnly`** color.

ReadOnly
Use this field to set the background color for read-only source code, such as source code of components that are not checked out.

Choose the color from a basic color schema or define custom colors.

Preferences - Code Assist

Use this dialog window to specify your *Code assist* preferences.

These settings only apply to the LN Studio **Script Editor**.

See "Code Assist" in the *Infor LN Studio Application Development Guide*.

Code Assist

Enable Code Assist
Select this check box to enable the code assist feature.

Present proposals in alphabetical order
If this check box is selected, *Code assist* presents the list of suggested completions in alphabetical order.

Automatically insert
If this check box is selected, and if there is only one completion proposal for the string you entered in the **Script Editor**, code assist will automatically insert the proposal at the cursor position.
Example

In the **Script Editor**, you type " `she` " and press CTRL+SPACE. Code assist automatically inserts the following text, because this is the only completion proposal available: " `shell(command,mode)` ".

Completion proposal background
The background color for the completion proposal.

Completion proposal foreground
The foreground color (text color) for the completion proposal.

Auto Activation

Enable auto activation
If this check box is selected, Code Assist starts automatically when you type the auto activation trigger character.

Auto activation delay
The delay in milliseconds between the moment you type the auto activation trigger character, and the moment *Code assist* starts automatically.

Auto activation trigger Char for Infor LN
The character that triggers the auto activation of *Code assist*. The default is a dot (.).

Preferences - Folding

Use this dialog window to specify your *Folding* preferences.

These settings only apply to the LN Studio **Script Editor**.

See "Folding" in the *Infor LN Studio Application Development Guide*.

Enable folding
If this check box is selected, you can fold and unfold the selected region types when you open a new editor.

If this check box is cleared, you cannot fold and unfold any region type in the script editor.

Collapse all selected region types when opening the Editor
If this check box is selected, all selected region types are collapsed automatically when you open the script editor.

Folding these region types:

Functions
If this check box is selected, folding is enabled for functions.

Header
If this check box is selected, folding is enabled for header comments.

Macro's
If this check box is selected, folding is enabled for macros.

Section
If this check box is selected, folding is enabled for script sections, such as the declaration section, the before.program section, and field sections in a UI script.

Preferences - Syntax

Use this dialog window to specify syntax coloring preferences.

These settings only apply to the LN Studio **Script Editor**.

Limitations

The script editor does not recognize syntax patterns because no syntax parser is available in this release. Therefore, if a variable has a name identical to a keyword such as "domain", "function", or "case", the variable is colored as a keyword. For example:

- You add the following code to the declaration section of a UI script: `long case`.
- `case` is colored as a keyword, because the editor does not recognize it as a variable.

Syntax Coloring
Specify the syntax coloring settings for various elements in the program text.
To specify the syntax coloring settings for an element:

1. Select the element from the list.
2. Specify whether the element must be bold and/or italic.
3. Click the color button.
4. Select the color from a basic color schema or define custom colors.

Preview
Shows a preview of your syntax coloring settings.

Preferences - Templates

Use this dialog window to create, edit, and remove templates.

These settings only apply to the LN Studio **Script Editor**.

To insert a template in the script editor, use the Code Assist feature.

Overview window

Use the check boxes to enable and disable templates. If a check box is selected, you can use Code Assist to insert the corresponding template in your source code.

Use these buttons to manipulate and configure templates:

Button	Description
New	Opens a dialog to create a new template.
Edit	Opens a dialog to edit the currently selected template.
Remove	Removes all selected templates.
Restore Removed	Restores any predefined templates that have been removed.
Revert to Default	Restores any predefined templates to their default configuration. This action does not modify user-created templates.
Import	Imports templates from an XML file in the file system.
Export	Exports all selected templates to an XML file in the file system.

Edit Template dialog window

Name
The name of the template.

Context
This is a standard Eclipse option that is not used by LN Studio.

Automatically insert
If this check box is selected, code assist automatically inserts the template at the cursor position, if the template is the only completion proposal for the string you entered in the **Script Editor**.

Description
The description of the template.

Pattern
The pattern of the template. This is the text inserted in the editor when you select the template.

In the template pattern, you can use predefined template variables. Variables are resolved to their concrete value when the template is evaluated in its context. You can specify variables using simple or full syntax. If there are multiple possible matches for a variable, they are presented as proposals to the user.

To add a variable to the template pattern, click **Insert Variable** and select the desired variable from the list.

You can select general and LN-specific template variables.

This table shows the general template variables:

Variable	Description
`${cursor}`	Specifies the cursor position when you leave the template edit mode. This is useful if the cursor must jump to another place than to the end of the template, on leaving template edit mode.
`${date}`	The current date.
`${dollar}`	The dollar symbol '$'. Alternatively, you can use two dollar signs: '$$'.
`${line_selection}`	The content of all currently selected lines.
`${time}`	The current time.
`${user}`	The user name.
`${word_selection}`	The content of the current text selection.
`${year}`	The current year.

This table shows the LN-specific template variables:

Variable	Description
`${name}`	The name of the function which is used above this variable
`${input_arg}`	The arguments of the function which is used above this variable.
`${returnType}`	The return type of the function which is used above this variable.
`${output_arg}`	Place holder

Preferences - Launching

Use this dialog window to specify your launching preferences.

When you launch a software component, the LN Studio checks whether the release of the corresponding *application* matches the package combination in which the component is executed. If they do not match, you may encounter problems during debugging.

In this dialog, you can specify whether/how the launch should continue if release and package combination do not match.

Note:

- The application release is specified in the **Create a new Application** dialog. The release is identical to the *Base VRC* on which the software development is based.
- The package combination depends on the runtime address *connection points* of the application's *software projects*. A runtime address connection point is linked to a `.bwc` file. You can find the package combination in the user data of the user account specified in this `.bwc` file. To view the user data, run the **User Data (ttaad2500m000)** session. Alternatively, look in the **Command** field of the **BW Configuration Properties** dialog.

  Default Launch Mode
  Specify how sessions are started by default.

  Select an option. In the **Configurations** dialog, the corresponding launch mode is selected automatically.

  Allowed values
  - Launch in BW
  - Attach to running Web UI

  Continue if release does not match package combination
  Use this field to specify whether/how the launch will continue if release and package combination do not match.

  Select one of the following options:

Always	The session is launched anyway.
Never	The session is not launched.
Prompt	A warning is displayed and you can choose whether you want to launch the session.

  The default option is Prompt.

  BFlow - Clear console before launch
  If this check box is selected, the LN Studio **Console** is cleared before a BFlow run or debug configuration is launched.

Preferences - Multipage

Use this dialog window to specify preferences for the Multipage Editors.

Display the TDE overview page for all editors
If this check box is selected, a **TDE Documentation** tab is displayed in all multipage editors. This tab shows an XML tree with the technical information on the software component's Technical Data Entity.

The information displayed in the **TDE Documentation** tab is primarily intended for support and troubleshooting purposes. Therefore it is recommended to display the tab only in case of problems.

Path for Additional File attachments
The directory where LN Studio stores the additional files that you edit in the **Additional File Editor**.

Preferences - Workbench

Use this dialog window to specify whether the editor is opened automatically each time you add a new resource to your workspace.

The script editor can start automatically, for example:

- When you use the **Get** command in the **Component Explorer** view to add a component to your workspace.
- When you use the **Open Declaration (source)** command in the script editor to view detailed information on a function that is called in your source code and not present in your workspace.
- During debugging, when you step into a function that is called in your source code, but not already present in your workspace.

Automatically open the editor for a new resource
Use this field to specify whether the corresponding editor is opened automatically for a new resource.

This table shows the available options:

`Always`	The editor starts automatically each time you add a component to your workspace.
`Never`	The editor does not start automatically when you add a component to your workspace.
`Prompt`	When you add a component to your workspace, the following question is displayed: "Open editor for the new software component?" Click **Yes** or **No** to answer the question. Optionally, click the **Remember my decision** check box in the question window. Depending on your answer to the question, this will change the **Automatically open the editor for a new resource** preference setting to "Always" or "Never".

The default option is **`Prompt`**.

Automatically check components after a change
If this check box is selected, LN Studio automatically verifies software components each time a component is saved.

If this check box is cleared, components are not verified automatically. You must start the verification manually through the **Verify Component** command in the shortcut menu in the **Activity Explorer** view.

For details, see "Verifying software components" in the *Infor LN Studio Application Development Guide*.

Preferences - Infor LN Configuration Management

Note: This page is not located under **Infor LN Studio Application** preferences, but under **Team** preferences.

Use this dialog window to specify your Configuration Management preferences.

Checkout while editing
Indicates whether LN Studio must prompt you to checkout a software component when you edit the component in the corresponding editor.

Select one of the following options:

Never	You cannot check out a component while editing. To change a component, you must first check out the component and then start the corresponding editor.
Prompt	You can check out a component while editing. If you try to change a component, which is not yet checked out, you are prompted whether you want to check out the component. If you click **Yes**, the component is checked out so you can change it. If you click **No**, the component is not checked out so you cannot change it.

The default option is Prompt.

Decorations

Append version information to resource name
If this check box is selected, LN Studio appends the version and release information of the *application* to the resource names displayed in the **Activity Explorer** view.

For example: 7.6_a4

Show image decoration for dirty resources

If this check box is selected, LN Studio displays a  decorator in the **Activity Explorer** view for each resource not in sync with the LN server.

A resource is dirty if the version stored in your local workspace differs from the version on the LN server.

Resources can become dirty if the **Build Automatically** option is turned off: When you save changes you make to a resource, the changes are saved in your local Eclipse workspace. However, the software component on the LN server is not updated.

You must build the component to synchronize the local resource and the LN server. The  decorator disappears automatically.

Text decoration for dirty resources
The character entered here is displayed as a prefix to the names of dirty resources in the **Activity Explorer** view.

A resource is dirty if the version stored in your local workspace differs from the version on the LN server.

Resources can become dirty if the **Build Automatically** option is turned off: When you save changes you make to a resource, the changes are saved in your local Eclipse workspace. However, the software component on the LN server is not updated.

You must build the component to synchronize the local resource and the LN server. The prefix disappears automatically.

End Activity Behavior

By default, close the activity
If this check box is selected, the activity is ended completely when you run the **End Activity** command in the **Activity Explorer**. The activity disappears from the **Activity Explorer**.

If this check box is cleared, the activity is ended partially when you run the **End Activity** command in the **Activity Explorer**. The activity is still present in the **Activity Explorer**.

Chapter 6: Integration preferences

Configuration

This section describes the configuration of Infor LN Studio related to the development of Business Interfaces.

These preference pages are available:

- **Infor LN Studio Integration**

  This page contains some general preferences.
- **Business Interface Test Tool**

  On this page you can set the configuration for testing Business Interface Implementations.
- **Colors**

  With the Colors preferences of LN Studio Integration, you can set the colors of the graphical modelers and textual editors that you use for Business Interface modeling.
- **Default Libraries**

  On this page you can specify some libraries that will automatically be included in new Interface projects.
- **Default Related Software Projects**

  On this page you can specify a set of related software projects.
- **Generators - Infor LN Implementation**

  With the **Infor LN Implementation** preferences of the Generators, you can set the VRC of the LN server for which you generate the Business Interface Implementation.

To navigate to the preference pages of LN Studio, select **Windows > Preferences**. In the Preferences window, expand the **Infor LN Studio Integration** folder and select one of the preference options.

Note: To generate proxies for old BDEs (version 6.1 only!)

The Business Interfaces you develop with LN Studio do not use proxies. However, to remain backwards compatible, LN Studio still contains functionality to generate BDE Proxies for the business interface definitions (BIDs) that you created with Integration Studio 6.1.

You can configure the involved Proxy Generators using the .NET Proxy and Java Proxy preferences of the LN Studio Generators. For detailed information, see the online Help of LN Studio.

General preferences

Use this page to specify the location of the "output" and "model" folder in the Integration project. The default values should suffice for most users.

The page contains these additional settings:

Automatically save dirty editors before refactoring
If this check box is selected, editors with unsaved changes are saved automatically when performing a refactoring action. If this check box is cleared, a warning window is displayed when you try to perform a refactoring action.

Validate on differences between base and specialized Interface Definitions and generate warnings
If this check box is selected, an automatic check is performed when you save a specialized interface definition. This reduces the performance. Optionally, to improve performance, clear this check box.

Prefer objects in libraries when automatically fixing references during import
During an import, duplicate datatypes can be found because the same datatypes can exist in the importer and in the libraries. If this check box is selected, in case of duplicates, the importer creates references to the datatypes from the library, rather than to the datatypes in the importer.

Number of import from folders in history
The number of entries in the list in the Import page.

Business Interface Test Tool preferences

Use this page to set the configuration for testing Business Interface Implementations.

Allow a manual change of the request xml for a Test
If this check box is selected, you can manually edit the XML documents, which are used as a request for a Business Interface, such as a SyncBOD or a Request BDE message.

If this check box is cleared, you cannot edit the XML request documents LN Studio generated from the specified test parameters.

Note: We do not recommend that you manually edit a generated request. because this can result in an incorrect request and might cause problems when you execute the test. Therefore, this check box is cleared by default.

Color preferences

Use this page to customize the colors that are used in the graphical modelers and text editors used for Business Interface modeling. Select the **Graphical Modelers** tab to define the colors of the graphical elements in the modelers. Select the **Editors** tab to define the colors of the text-based editors. You

can change any color by clicking on the colored button. A **Color** dialog box, in which you can select the desired color, is displayed.

Defining a set of default libraries

When creating an interface project, default libraries are automatically included in the project.

To define default libraries:

1. In Eclipse, select **Windows > Preferences**.
2. In the left-hand navigation pane of the **Preferences** window, select **Infor LN Studio Integration > Default Libraries**.
3. To add a library, click **Add**. A file selection dialog box, in which you can navigate to an existing library, is displayed.
4. Select the library and click **Open**.

You can include the default libraries in two different ways. If you select the **Link the Libraries to the project** option, the new Interface project will contain links to the default libraries. If you select **Copy the Libraries to the project**, the new Interface project will contain copies of the libraries.

To remove a library from the set of default libraries, select the library in the list, and click **Remove**.

Defining a default set of related software projects

You can define a default set of related software projects. When you create a new interface project, these default projects will be linked as related software projects. The related software projects are relevant when developing business interface implementations. When you generate Infor LN implementations, you can generate the libraries for those implementations in activities of these related software projects.

Note: The following procedure is optional. We recommend that you define related projects, but this is not mandatory.

To define the default set of related software projects:

1. In Eclipse, select **Windows > Preferences**. The **Preferences** window is displayed.
2. In the left-hand navigation pane of the **Preferences** window, select **Infor LN Studio Integration > Default Related Software Projects**. The **Default Related Software Projects** dialog box is displayed.
3. To add a related software project, click **Add**.
4. Specify this information:

   Name
   Select a software project from the list.

Is Integrated
If this check box is selected, you can use components of the software project when you are editing hooks in a business interface definition. For example, in the hook editor, you can use the **Open Declaration** command to open components of the related software project.

5 Click **OK**.

To remove a software project from the list, right-click the entry in the list and select **Delete**.

Preferences of Infor LN Studio Implementation Generator

This section describes the preferences of the LN Studio Implementation Generator. With this generator, you can generate the runtime code for an implementation on an LN application server.

Before you can generate Business Interface runtime code, you must specify the version, in other words, the VRC in which the Business Interface runtime will be created. You must set the version on the Preferences page of the LN Studio Implementation Generator. To access this page, in the **Preferences** dialog box, select **Infor LN Studio Integration > Generators > Infor LN Implementation**. To open the **Preferences** dialog box, select **Window > Preferences**.

The **Major Version**, **Minor Version** and optional **Subordinate Version** refer to what is called the Base VRC in LN. To view the available base VRCs, run the **Base VRCs (ttpmc0110m000)** session in LN. If no base VRC exists that corresponds to the VRC in which you must generate the Business Interface runtime, you must create a base VRC in LN.

To determine the actual package VRC that is used, the LN Studio Implementation Generator uses these criteria:

- Takes the package from the value specified for the **Implementation Identifier** property of the Business Interface Implementation.
- Takes the VRC from the Export VRC, as defined in the **Base VRCs (ttpmc0110m000)** session.

Note: To develop in the resulting package VRC, you must have sufficient authorizations. In Infor development, the VRC that you must use is a VRC that is open for development or maintenance. In a customer environment, the VRC is a customization VRC, such as B61C_a_cust.

This table shows the preferences of the LN Studio Implementation Generator:

Preference property	Description
Major Version	Version number of the Base VRC.
Minor Version	Release code of the Base VRC.
Subordinate Version	Customer code of the Base VRC.

Chapter 7: Connectivity preferences

Under **Infor LN JCA Connectivity** in the preference pages, you can specify preferences related to JCA connectivity. To open the preference pages, in Eclipse, select **Windows > Preferences**. The main preference page contains the configuration of the connection points. There is a sub-page for configuring the logging preferences.

Connection Points Configuration

This page contains the connection points that are required to connect to one or more LN servers. Some connection points are static and are always present in the list: ProjectServer, Debug, RuntimeRepositoryConnection, and TestServerConnection. Other connection points are dynamic. Dynamic connection points are added, by starting a wizard, at the time they are required. You cannot add new connection points on this preference page.

To change the settings of a connection point:

1. Select the connection point in the list.
2. Click **Edit**. The configuration wizard starts.
3. On the first wizard page, you can either choose to share the connection and specify the shared connection point, or choose an activation type. The available activation types depend on the connection point you are configuring.
4. Depending on the choices on the first page, you may need to click **Next** and complete a second page.
5. Specify the required data for the selected activation type.
6. On the final wizard page, click **Finish**.

To test a connection point:

1. Select the connection point in the list.
2. Click **Ping**. A dialog box shows whether the test succeeded.

Log Configuration

Use this page to specify the settings for logging the connection-related information to a log file. In case of problems, a support engineer may ask for this information.

You can specify these preferences:

Enable logging
If this check box is cleared, no logging takes place. Select this check box to activate logging.

Location
Specify the complete path of the log file. The file name must end in "`.log`". Use the browse button to select the folder in which the log file must be stored.

Application log level
Select the log level. This influences the amount of messages that are logged. These levels are supported:

- Error

  Only errors are logged.

- Info

  Errors and informational messages are logged.

- Debug

  Errors, informational messages, and debug messages are logged.

Limit file size
If no limit is specified, the log file grows when new messages are logged. If you specify a limit, by size or by time, the log file does not grow indefinitely.

On size
If this check box is selected, the log file does not grow beyond a maximum size. If the maximum size is reached, the existing log file is renamed by appending ".1" to the name, and a new log file is opened. Existing log files that have previously been renamed are renamed to the next higher number: the ".1" file is renamed to ".2", the ".2" file is renamed to ".3", etc. If the maximum number of log files is reached, the oldest one is deleted.

- Maximum size

  Specify the maximum size of the log in MB.

- Maximum number

  Specify the maximum number of log files that must be preserved. Example: if you specify "2" in this field, at most 3 log files will be present in the log file location: the current log file and two older log files with the extensions ".1" and ".2".

On time
If this check box is selected, the log file is renewed periodically. If the specified interval has elapsed, the existing log file is renamed by appending a timestamp to the name, and a new log file is opened. Select one of these intervals:

- Daily
- Half-daily

- Hourly

Chapter 8: Infor LN Project Server

Infor LN Project Server

Infor LN Project Server is used to maintain *Software Projects* and *Activities* for *Software Engineers* that develop software in *LN Studio*.

Project Server functionality is available in the **Project Server** perspective in LN Studio.

Note: A number of screenshots in the documentation may be based on previous application releases. They can differ slightly from your application screens. However, the described functionality is similar.

Procedures

Defining a software project

This topic describes how to define *software projects* and *activities*. The projects and activities are stored in *Project Server*.

To define a software project:

1. Start LN Studio
2. Open the **Software Project Explorer** view

   On the **Windows** menu, select **Open Perspective**, and then click **Project Server**.

   The **Software Project Explorer** view is displayed in the Eclipse workbench.
3. Specify the project data

   Complete these steps:

   a. On the view's toolbar, or on the shortcut menu, select **New Software Project**. The **Create a Software Project** dialog box is displayed.
   b. Specify the project properties. See the online help of the dialog box.
   c. Save the project and close the dialog box.
4. Add one or more activities to the new project

   Complete these steps:

a In the **Software Project Explorer** view, right-click the new project and select **New Activity**. The **Create a new Activity** dialog box is displayed.
b Specify the activity properties. See the online help of the dialog box.
c Save the activity and close the dialog box.

The users, to which the activities are assigned, can now start developing software in the new project. They can select the assigned activities in the **Application** perspective in the LN Studio workbench.

Dialogs

New Development Environment

Use this dialog to define a new development environment.

Name
The name of the development environment.

Hostname
The TCP/IP hostname or IP address of the LN server.

BSE path
The directory on the application server where the LN Software Environment resides.

Bshell
The logical name of the bshell, as specified in the `${BSE}/lib/ipc_info` configuration file.

Create a new Application

You can use the **New Application** dialog box to create a new *application*.

New Application window

To open the **New Application** dialog box, complete one of the following steps:

- Right-click the **Application Explorer** view of LN Studio and, on the shortcut menu, select **New Application**.
- Click the **New Application** button on the **Application Explorer**'s toolbar.

To create a new LN Studio application, in the **New Application** dialog box, fill in the required fields and click **OK**. The new application is created in the currently connected LN server as indicated by the Administrator *Connection Point* in the **Infor LN JCA Connectivity** window.

The following section describes the fields of the **New Application** dialog box.

Note: To view or modify the properties of an existing application, use the **Application Properties** dialog box. This dialog box is identical to the window described here, except for the **Application name** field. When you create a new application, specify an application name. When you modify an existing application, you cannot change the application name (read-only field).

Application

Use the upper part of the dialog to specify basic application properties.

Application name
The name of the application.

When you create a new application, it is mandatory to specify an application name.

Allowed values

The application name identifies the application. Therefore, the application name must be unique in the context of the connected LN server. The name can be any ASCII text with a maximum length of 30 characters.

Description
The description of the application.

Specifying an application description is optional.

Allowed values

The description is a (multi byte) text with a maximum length of 50 characters.

Release
The release of the application.
For LN applications, the release is identical to the *Base VRC* on which the software development is based.
See Applications.
Specifying a release is mandatory.
Note: An LN Studio application is linked to the Base VRC of a PMC module. Within the PMC module, the package VRCs are linked to the export VRC of the PMC Base VRC.
Allowed values

To specify a value for this field, select a release from the drop-down list.

Base
This field is not yet implemented.
Note: You can access the field and select a value from the list. However, this value is ignored by the LN Studio. Therefore, you are recommended to leave this field empty.

InContext Reference Model
The application's InContext Reference Model.

Related topics

- "InContext Modeling" in the *Infor LN Studio Application Development Guide*
- *Infor Enterprise Server InContext Modeling Development Guide*

Use SCM
If this check box is selected, the software components developed or modified in LN Studio are managed with the *Software Configuration Management* (SCM) tool of the connected LN environment.

Use SCM to perform version management for packages linked to the application, or for the individual activities of the application. To use SCM for the version management of the packages, SCM must be enabled for the associated Project VRC on the LN server. If SCM is not enabled, you cannot add a package from this VRC to the application.

See "Activity based development" in the *Infor LN Studio Application Development Guide*.

Packages

Use this tab to link packages to the application.

Package
A package linked to the application.

In LN Studio, you can link packages to an application. These packages must have a VRC that corresponds with the Export VRC of the Base VRC of the PMC module. This must also match the Base VRC of the application. This Export VRC contains the application data with the linked packages. LN Studio will automatically create such an Export VRC, if no such VRC is available.

The **Packages** field displays a table with the packages that have been linked to the application. These packages have the same Base VRC as the application. The table shows the name of the packages and the package description. An additional field indicates whether the package is used.

To link packages to the application:

1. Click **Add**. An empty package record is added to the **Packages** grid. Then, in the **Package** field, select a package from the drop-down list. This list contains all the packages linked to the application's Base VRC. After you select a package, the package name and description are displayed in the grid.
   Note: To add all packages at the same time, click **Add All**.
2. To indicate whether a package is used in the application, select or clear the corresponding **Active** check box in the grid.

Note: You require special authorization to create or modify applications linked to a Base VRC of a standard LN release. To make a customization for a particular customer, copy the application to the customer's application.

Description
The description of the package.

This field is read-only. The specified description is derived from the selected package.

Active
If this check box is selected, the package of the application is changed or modified in LN Studio.
Allowed values

The check box is cleared.

Languages

Use this tab to link software languages to the application.

Language
A software language linked to the application during development time.
Important

If you use the Language Translation Support functionality, to translate labels or other user interface components, do not add secondary software languages to an application. Only use secondary software languages for small translations performed by a software developer, outside the framework of the LTS export/import mechanism.

The term "software language" refers to a language used by the application to communicate with the application user. Therefore, when you run the application, the text on the buttons, commands, menus, messages and so on, are expressed in one of the software languages linked to the application.

You can link multiple languages to an application, depending on the language packs installed on the application's LN environment.

An LN Studio application can have two types of languages: primary and secondary. Each application must have only one primary language. This is the development language and used to display the editable components throughout the user interface of LN Studio. If any references are made to another component, this reference is expressed in the primary language. By default, the primary language is `2`. If you do not specify a primary language, English is used.

Apart from the primary language, an application can possess multiple secondary languages. These can be considered as translations of the expressions in the primary language. You can view and modify these translations in the editor of the concerned software component. The translations are used when running the application in a mode for a particular secondary language, determined by the language of the user.

The **Languages** field displays a table with all languages linked to the application. These languages must be installed on the LN server connected to LN Studio through the Administrator connection point. The table shows the name of the languages and the language descriptions. An additional field indicates whether the language is a primary or secondary language.

To link a new language to the application, complete the following steps:

1. Click **Add**. An empty language record is added to the **Languages** grid.
2. In the **Language** field, select a language from the drop-down list. This list contains all the packages installed on the connected LN server. After you select a language, the language code and language description are entered in the grid.
3. To indicate whether the language acts as primary language, select or clear the **Primary** check box.

Description
The description of the language.

This field is read-only. The specified description is derived from the selected language.

Primary
If this check box is selected, the current language will act as the primary language of the application.
If this check box is cleared, the current language is a secondary language.
Note:

If you set the primary language of the application, any language that was previously the primary language will automatically be marked as a secondary language.

Allowed values

The check box is cleared.

Documentation

Documentation
Use this field to specify additional information.

Create a Software Project

Use this dialog box to create a *Software Project*.

Project

Project name
The project name.

Description
The project description.

Requirement ID
The ID of the business requirement to which the project belongs (requirements are usually stored in a requirement management or defect tracking system).

Partial End Allowed
If this check box is selected, you can partially end the activities linked to the project. When you end an activity, you can select the components for which the activity is ended. These components are checked in to the project VRC, while the other components stay available in the activity VRC. See Developing software components.

If this check box is cleared, you can only end activities in their entirety.

Sharing Activities Allowed
If this check box is selected, each activity linked to the project can be assigned to multiple users.

If this check box is cleared, each activity can be assigned to only one user.

Note:

When you open a shared activity, the components developed by other users are not displayed automatically. To display these components, recover the activity. See the online help of the **Recover activity** dialog.

If sharing activities is used in combination with activity context, all assignees of an activity must use the same context. Otherwise, the assignees will get different results when they recover an activity, or when they launch or debug a session.

Integration With PMC
If this check box is selected, you can create delivery activities in the project, and deliver components from Project Server. When you end a development activity, you are prompted to deliver the activity and create a PMC solution.
See Delivering software components.

If this check box is cleared, the PMC integration is disabled. You cannot deliver the components from Project Server. You can perform the PMC delivery from the LN server instead.

Repository

Development Environment
The LN environment in which the software for the project is developed. Select the environment from a drop-down list.

Application
The *application* to which the project belongs. Select the application from a drop-down list.

Documentation
Use this field to enter additional documentation about the project.

Create a new Activity

Use this dialog box to create an *activity*.

Project

Project name
The *software project* to which the activity belongs. Select a project from the drop-down list.

Description
The description of the selected project.

Activity

Name
Specify an activity name.

Description
Specify an activity description.

Type
Select an activity type from the drop-down list.
If the activity is intended for software development, select one of the following types:

- Enhancement
- Change request
- Miscellaneous
- Defect (intern)
- Defect (extern)

If the activity is intended for software delivery through PMC, select the "Delivery" type.

Note: You can only select the "Delivery" type if the **Integration With PMC** check box is selected in the properties of the project to which the activity belongs.

Requirement ID
The ID of the business requirement to which the activity belongs (requirements are usually stored in a requirement management or defect tracking system).

Owner
The *software engineer* who owns the activity. Select a software engineer from the drop-down list.

The owner can open the activity, develop components in the activity, reassign the activity, and end the activity.

Assignees

Assignee
The software engineers to which the activity is assigned. The assignees can open the activity and develop components in the activity. However, only the owner can end the activity.

To add an assignee, click **Add** and select a software engineer from the drop-down list.

The **Assignees** grid is only available if the **Sharing Activities Allowed** check box in the software project's properties is selected.

If sharing activities is not allowed, the **Add** button is disabled, and the system automatically adds the owner of the activity as assignee. This change is displayed only after you re-open the dialog box.

Documentation
Specify documentation about the activity.

Finish
Saves the new activity and closes the dialog box.

Finish and Open
Saves the new activity and closes the dialog box. Automatically switches to the **Application** perspective and opens the new activity in the **Activity Explorer** view.

Compile activity overview

This dialog box shows an overview of the components that were compiled, for example when you closed an activity.

Project Server View Filter

Use this dialog box to specify filter criteria for the **Software Project Explorer** view. To apply the filter, click **OK**. The filter criteria are stored. The next time you open the view, the filter is applied automatically.

Show Projects with status
Select one or more project statuses. The view only shows projects that have one of these statuses.

Show empty projects
Use this field to easily recognize empty projects in the view.

If this check box is cleared, a + sign is displayed before all projects in the view.

If this check box is selected, the + sign is not displayed before empty projects.

Show Activities with status
Select one or more activity statuses. The view only shows activities that have one of these statuses.

Show Activities of all users
If this check box is selected, the view shows activities of all users. If this check box is cleared, the view only shows your own activities.

Name starts with
If you select the check box, you can specify one or more characters. The view shows activities whose names start with the specified characters.
Note: The filter is case sensitive.

Perspectives

Project Server perspective

The **Project Server** perspective provides functionality for project managers and administrators.

This table shows the views in the perspective:

Software Project Explorer view	Project managers use this view to maintain *software projects* and *activities* on the *Project Server*. For details on the procedure to define projects and activities, see Defining a software project on page 43.
Application Explorer view	Administrators use this view to create and maintain the applications for which the software projects are defined. For more information on the administrator's tasks, see the *Infor LN Studio Administration Guide*.

Views

Application Explorer view

Use the **Application Explorer** view to add, maintain, and remove development environments and *applications*.

An *application* is a software application developed in the *LN Studio*. Each application is linked to a *Base VRC*. For each application, one or more projects can be defined, in which the software engineers develop their software components.

A development environment is a BSE environment on an Infor LN server, in which the software engineers develop software components.

The **Application Explorer** view is part of the **Infor Project Server** perspective.

Toolbar

The toolbar of the **Application Explorer** view includes the following buttons:

Properties	Starts a dialog box where you can view or modify the properties of the selected item.
Create a new development environment	Starts the **New Development Environment** dialog box. Here, you must enter the properties for a new development environment.
New Application	Starts the **Create a new Application** dialog box. Here, you must enter the name, release, and other properties for a new application.
Menu	Contains the **New Application** command.

Note: The remaining buttons are standard Eclipse commands. For details on these buttons, see "User interface information" in the *Workbench User Guide*.

Shortcut menu

To open the shortcut menu, right-click on a resource in the view:

This menu provides the following LN Studio-specific commands:

New Development Environment	Starts the **New Development Environment** dialog box. Here, you must enter the properties for a new development environment.
New Application	Starts the **Create a new Application** dialog box. Here, you must enter the name, release, and other properties for a new application.
Properties	Starts a dialog box where you can view or modify the properties of the selected item.

Delete	Removes the selected item.

For details on the remaining commands, see "User interface information" in the *Workbench User Guide*.

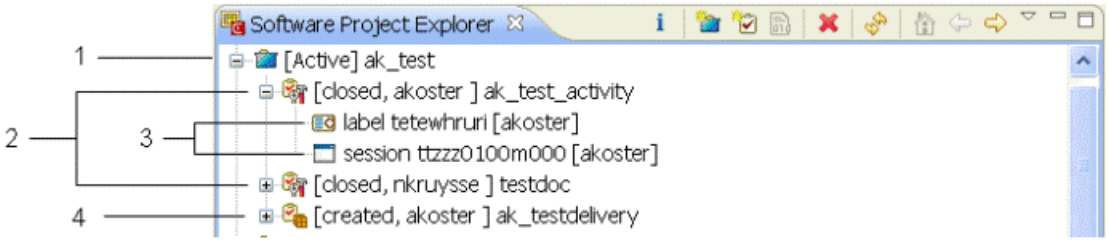
Icons

The following icons are displayed in the **Application Explorer** view:

Icon	Description
	Development Environment
	Application

Software Project Explorer view

The **Software Project Explorer** view provides a hierarchical view of the projects, activities, and modified software components in the *Project Server*. See the following figure for an example.

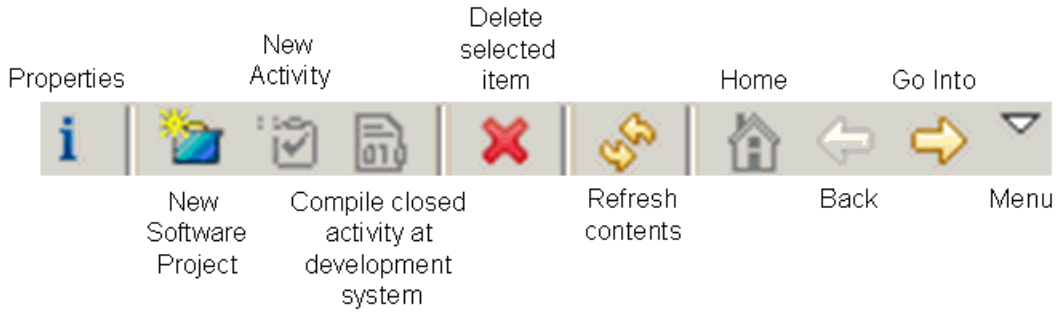


This table describes the different nodes in the previous figure:

Nr. in figure	Description
1	Project
2	Development Activity
3	Modified LN software component
4	Delivery Activity

Toolbar

The toolbar of the **Software Project Explorer** view contains the following buttons:



Properties	Shows the properties of the selected project or activity. For details on the project and activity properties, see the description of the **Create a Software Project** and **Create a new Activity** dialog boxes.
New Software Project	Adds a project on the project server.
New Activity	Adds a new activity to the selected project.
Compile closed activity at development system	Compiles the software components in the selected closed activity. Use this command if the automatic compilation during the ending of the activity has failed, and the problem that caused the compilation error is solved. Example: • Activity B is part of the context of activity A. A UI script in activity A calls a DLL that belongs to activity B. The DLL is not compiled yet. • Activity A is ended by its owner. The compilation of the UI script fails, because the DLL in activity B is not compiled. Activity A is closed anyway. • Activity B is ended by its owner. The DLL is compiled automatically and the activity is closed. • The owner of activity A can now use the **Compile closed activity at development system** option to compile the UI script in the closed activity A.
Delete selected item	Deletes the selected project or activity. **Note:** • You cannot delete a project, if the project status is `Created` or `Active`. • This option does not undo the software modifications performed within the project or activity that are already checked in.

Menu	Contains these menu items: • New Software Project • New Activity • Compile closed activity • Filters This command starts a dialog box where you can specify filter criteria for the **Software Project Explorer** view. See Project Server View Filter on page 50.

Note: The remaining buttons are standard Eclipse commands. For details on these buttons, see "User interface information" in the *Workbench User Guide*.

Shortcut menu

To open the shortcut menu, right-click on a resource in the view.

This menu contains most of the toolbar commands.

Icons

The following icons are displayed in the **Software Project Explorer** view:

Icon	Description
	Project
	Development Activity
	Delivery Activity
	Additional File
	Domain
	Function
	Label
	Library
	Menu
	Message
	Question
	Report
	Session
	Table

Glossary

3GL script

A *Program script* that can be linked to sessions without forms, or not linked to a session. 4GL statements and sections cannot be used in 3GL scripts. The entire program flow and the main function must be specified.

4GL engine

The program that provides default functionality for a session to prevent application programmers from developing a session from the beginning. The 4GL Engine, formerly called standard program (STP), is used because sessions are alike. The 4GL Engine also provides a mechanism to change the 4GL Engine's default behavior, and to program dedicated functionality for a specific session. When a session is started, a separate 4GL Engine instance is activated to handle the session.

Synonym: standard program

4GL script

An event-oriented *Program script* linked to a session. The instructions can be specified in program sections, form sections, field sections, main table input/output sections, choice sections, zoom from sections, and functions.

Activity

An activity in which software components are developed or delivered. Each activity is part of a *Software Project* and assigned to a user (software engineer). Activities and software projects are stored on the *Project Server*.

This table shows the types of activities you can define:

For software development	• Enhancement • Change Request • Miscellaneous • Defect (intern) • Defect (extern) Use these types to develop software components.
For software delivery	Delivery Use this type to group development activities and to deliver the corresponding software components through a PMC solution.

During its life cycle an activity can have these statuses:

- Created – The activity is created in Project Server.
- Opened – The activity is opened, at least once, in the **Activity Explorer**.
- Cancelled – The activity is cancelled. To re-use the activity, you can open the activity again through the **Activity Explorer**.
- Closed – The activity is closed through the **End Activity** command in the **Activity Explorer**.
- Finalized - The activity is delivered through a PMC solution.

Project Managers can create software projects and activities through the **Software Project Explorer**.

Activity VRC

A VRC that contains software components for an *activity*. The activity VRC is generated automatically when you open the activity for the first time.

By default, the components in this VRC are only accessible for the software engineer to which the activity is assigned. Optionally other software engineers can access these components if they put this activity in the activity context of their own activity.

Additional file

A generic component, such as an XML schema file, a GIF image, and so on. From LN 6.0a onwards, additional files are stored in a specific package, module, and package VRC.

Administrator
The administrator creates and maintains the applications for which the *Software Projects* are defined.

Application
A software application is a consistent set of packages that functionally belong together, and developed in the *LN Studio*. Each application is linked to a *Base VRC*, which determines the environment on the LN server in which the application's software components are stored. For each application, one or more projects can be defined, in which the software engineers develop their software components.

appropriate menu
Commands are distributed across the**Views**, **References**, and **Actions** menus, or displayed as buttons. In previous LN and Web UI releases, these commands are located in the **Specific** menu.

Array
A list of data items of the same type with the same name. An element in the array can be referenced by an expression composed of the array name and an index expression. Multilevel arrays are used for data storage in tables.

Base VRC
A way in PMC to identify products in a unique way. Updates at the distributor side are provided with the base VRC identifier. A base VRC can contain the code of the physical VRC in which the related master product is installed, for example, B61_a. However, it can also be a code unrelated to a physical VRC, for example, 7.6_a_tt. At the recipient side, every update VRC is linked to a base VRC identifier. The installation process checks if the base VRC identifier of the update matches with the base VRC identifier of the update VRC. If not, you cannot install the update in that update VRC.

Base VRC combination
A *Base VRC* combination is defined at the PMC distributor side and consists of a set of related base VRCs. A base VRC combination controls the creation of co-requisites between base VRCs. You can only define co-requisites between base VRCs that are part of the same base VRC combination. Base VRC combinations prevent the unwanted creation of co-requisites between base VRCs.

Business Object
A Business Object is an object understood by the business, such as a purchase order or an organizational unit. A Business Object has information stored in the Business Object attributes, such as the purchase order number or the organizational unit name. A Business Object also contains a set of actions, known as Business Object methods. These can manipulate the Business Object attributes, such as create purchase order and list organizational units.

From a development perspective, a Business Object is a collection of tables and functions that manipulate tables implemented simultaneously as one group during the development phase. A Business Object is identified by the combination of a package code, module code, and Business Object code.

Business Object Interface
Business Object interfaces provide a connection between partner applications, third-party applications, and the LN software. They also connect LN functional components. Business Object interfaces are developed for situations where the LN software acts principally as a server, and a client software invokes the methods in the objects.

BW Configuration
A configuration that contains data to connect a client PC to an LN server. You can maintain configurations with the Infor LN BW Environment and Configuration Selector (BECS). A configuration is stored in a file with a `.bwc` extension.

C
A structured programming language. C is a compiled language that contains a small set of built-in functions that are machine dependent. The rest of the C functions are machine independent and contained in libraries that can be accessed from C programs. C programs are composed of one or more functions defined by the programmer.

Chart
A graphic or diagram that displays data or the relationships between sets of data in pictorial, rather than numeric form. The data can be presented in a graph, a line, or a pie, and can include titles, legends, and footnotes.

Chart application
A program used to send data from an LN session to the Chart Manager. A chart application is linked to a package VRC to customize the attributes specified in the chart.

Chart type
The chart type determines what the chart looks like.

For example, it defines the type of graph, thickness of lines, size of bars, and colors. The following default chart types are present in LN:

- Bar
- Layer
- Line
- Pie
- Scatter
- Stacked bar

Check in
A process that releases a checked out software component within an *activity*.

Check out
A process that locks the software component for other developers. During the check out phase, the component can be updated and tested.

Child field
A table field that is part of a combined field. A combined field contains two or more table fields, which are the child fields of the same table field.

Class
In Object Oriented Programming, a class is a generalized category that describes a group of more specific items, called objects, that can exist within it. A class is a template definition of the methods and properties (variables) in a particular kind of object. Therefore, an object is a specific instance of a class that contains real values instead of variables. The class is one of the defining ideas of object-oriented programming.

The important ideas about classes are as follows:

- A class can have subclasses that can inherit all or some of the characteristics of the class.
- In relation to each subclass, the class becomes the superclass.
- Subclasses can also define their own methods and variables that are not part of their superclass.
- The structure of a class and its subclasses is called the class hierarchy.

Code assist
Code assist provides a list of suggested completions for partial character strings in the LN Studio Script Editor.

Collection
In PMC, a collection is a group of individual solutions. At the PMC distributor side, you can perform grouping in various ways. For example, manual grouping based on a functional topic, or grouping based on solutions created in a particular period and so on. You cannot define dependencies between collections. At the recipient side, the entity collection is not available. When a collection is scanned, the individual solutions are added to the PMC registry and can be processed individually.

Combined field
Two or more child fields of the same *Table*.

Connection Point
A logical endpoint in the JCA connectivity architecture. A connection point mainly contains an identifier and connection properties, such as host name, user, password, BSE, and bshell name. Client applications use the identifier to connect to a particular server.

LN Studio uses these connection points:

- ProjectServer
- Administrator
- Debug
- Development Address
- Runtime Address
- RuntimeRepositoryConnection
- TestServerConnection

Co-requisite
In general, co-requisites are defined between solutions of a standard product and derived products. Co-requisites guarantee that related products are updated simultaneously under the condition that the update VRCs of the related products are linked to the same VRC combination. The order of installation is not relevant. The solutions can have the same *Base VRC*, or different base VRCs.

Data Access Layer
A *Dynamic Link Library* linked to an LN database table.

Data Type
The internal representation of data, such as a string, long, double, date or UTC.

Debugging

The process of identifying and addressing the cause for defective behavior of a system.

These debugging techniques are available:

- Definition of conditions that suspend a running process, so you can inspect it more closely
- Navigation through a suspended process, to discover the flow and identify any problems
- Inspection, to validate the state of the process and identify any problems

Debug target

An execution context, such as a process or virtual machine, that can be debugged. In Infor 4GL, a debug target represents a session started in debug mode.

Dependency

In PMC, the relation between solutions. Dependencies are defined at the PMC distributor side and are part of the meta data of a PMC solution and guarantee that PMC solutions are installed in the correct configuration and sequence at the PMC recipient side.

The following values indicate the dependency type between solutions.

These dependency types are available:

- *Pre-requisites*
- *Co-requisites*
- *Post-requisites*

You can only install solutions that are dependent on other solutions if the other solutions are already present, or are also installed.

The same dependency types exist between *Patches*. However, to keep the descriptions readable, only solutions are mentioned, but patches are meant as well. One exception applies: the post-requisite type is not applicable to patches.

Details session

A dialog box that shows all the details (fields) of the line (record) selected in the associated overview session. Use a details session to view, enter, or change the data of one record.

A details session can contain a number of tabs to group related fields.

Development VRC

In PMC a physical VRC, derived from the *Export VRC*, in which checked-out software components are temporarily stored during a change process.

Domain

A domain describes the properties of table fields. The main property of a field is the data type, such as string, long, double, date, UTC. Other properties are the length of a string, the alignment of a string, and the date and time format.

Domains indicate the valid values for an attribute and are used to define the scale division of the axes or to verify data.

Dynamic form

A form with a dynamic form definition.

The developer does not need to determine exactly where fields must be placed, or what they must look like. Instead, the developer defines the following:

- The form contents.
- The form structure.
- The sequence of the objects on the form.

At runtime, the dynamic form displays only those fields for which the user is authorized.

Dynamic Link Library

A way to share common functions between programs. This library contains functions for common use. The library can be linked to the object as a function call at run time. Implementing a dynamic link library reduces the size of objects to a minimum, because dynamic link libraries are not included in a program's object.

Dynamic session

A session with a dynamic form definition.

Depending on settings, a dynamic session can start as one of the following:

- A details session.
- An overview session.

In the dynamic form definition, the developer does not need to determine exactly where fields must be placed, or what they must look like. Instead, the developer specifies this information:

- The form contents.
- The form structure.
- The sequence of the objects on the form.

At runtime, the dynamic form shows fields for which the user is authorized.

Eclipse
Eclipse is an open platform for tool integration built by an open community of tool providers. Operating under an open source paradigm, with a common public license that provides royalty free source code and world wide redistribution rights, the Eclipse platform provides tool developers with ultimate flexibility and control over their software technology. Eclipse is used as a platform for LN Studio.

Eclipse Task
Eclipse tasks are displayed in the **Tasks** view. Each Eclipse task represents an action that must be performed, such as to modify a particular line in a script. If a task is associated to a line in an Infor 4GL script or library, double-click the task to edit the script or library. The editor opens the script or library at the involved line.

Create Eclipse tasks through the **Tasks** view and the LN Studio script editor.

Editor
An editor is a visual component within the LN Studio workbench used to edit or browse a resource, such as a *Program script* or a library. Modifications made in an editor follow an open-save-close life cycle model. Multiple instances of an editor type may exist within a LN Studio workbench window.

Export VRC
The physical VRC from which components that belong to a PMC solution must be exported at the PMC distributor side. Each *base VRC* has an export VRC linked, so components for different products are exported from different physical VRCs.

Feature Pack
See *Service Pack*.

Field
In table definitions, a field refers to a column. In a session, a field is a specified area of a record used for a particular category of data.

Float
A data type name used to declare variables that can store floating-point numbers. A floating-point number is a number containing a decimal point, with a maximum of 15 significant digits (8 bytes).

Folding
The LN Studio script editor supports folding of code regions. If you hover over a folded element, the hidden code is displayed.

Form
A screen that appears when a session is started. A form interacts with the database, and provides the user interface used to manipulate the data on the form.

Form command
A form command starts a session, function or (sub)menu by means of which a user can carry out a particular task.

A form command, as opposed to standard menu commands such as the **Exit** command, must be especially defined for a session tab.

Form field
A field shown on a form. A form field is selected from the available fields of an input table and its reference tables.

Function
A piece of program code that makes up part of a program script.

A function is a self-contained software routine that can perform a task for the program in which the function is written, or for another program.

Group
A set of form objects grouped together, such as form fields and child groups.

Group-by field
A field on an overview form positioned above the grid. The Group-by field determines what is shown in the grid of an overview form. Only the records that belong to the Group-by fields are shown in the grid, such as all orders that belong to a specific customer. The name of the customer is shown in the Group-by field, the records are shown in the grid.

Index
One or more table fields used to sort and search records in a *table*. A table must have at least one index. The first index is always the *primary key*.

Infor 3GL
A third-generation proprietary programming language that is a mix of Basic and C.

Infor 4GL
A fourth-generation programming language designed for interacting with the programmer used with relational databases. 4GLs are event-driven.

Installation run
In PMC, a group of solutions that were installed together. This can be a range of solutions, a solution with *pre-requisites*, or a combination of both.

Integrated session

The session and the session's form are integrated into one object. The form is a subcomponent of the session.

When you perform an operation, such as copy, delete, check in, or check out, on an integrated session, you also perform the operation on the integrated session's form.

A non-integrated session's form is a separate object.

Label

A code that is used instead of language-dependent text in forms, reports, and menus. A label consists of a name and a content description. The content of a label can differ by language, but the label name remains the same for all languages.

Language number

A conversion of the language code to a number between 0 and 61. The language number eliminates problems caused by the use of uppercase and lowercase language codes.

This table shows how the language code corresponds to the language number:

Language code range	Corresponding language number range
0 to 9	0 to 9
a to z	10 to 35
A to Z	36 to 61

For example:

- Language code b = Language number 11
- Language code B = Language number 37

Library

A collection of files, computer programs, or subroutines.

LN Studio

A platform for activity based development of LN software. LN Studio is implemented in the Eclipse framework and makes use of the *Software Configuration Management* functionality on the LN server. The LN Studio workbench contains various powerful features, such as an advanced script editor, various multipage editors, an outline view, a task list, and commands to debug and run software components.

Long

A data type specified in LN as any whole number from -2147483648 to 2147483647.

Menu

A list of options from which a user can perform a desired action, such as starting sessions, other menus, and queries. A start-up menu is defined for each user. Using this start-up menu, the user can access all sub-menus attached to the start-up menu tree.

Message

A notification that informs you about something. A message attends you to an event, error, warning, and so on. Messages usually appear in a message window or are logged in a file. If displayed in a window, a message requires a confirmation: Click **OK**. Messages are different from *questions*, as questions always require a choice response.

Module

A part of a package consisting of a number of related software components, such as sessions, tables, program scripts, reports, forms and menus. For example, the General Ledger module in Financials.

A module code consists of three characters. For example, the General Ledger has the code "gld".

Obsolete solution

Obsolete solutions are an administrative aid to manage the synchronization of updates at the PMC recipient side when you install a *Service Pack*. An obsolete solution does not contain software components.

Original VRC

The VRC that contains the software components that have to be modified. If SCM is active, these components are changed in the *Private VRC*. If SCM is not active, these components are changed in the *Activity VRC*.

Overview session

A session that lists the available elements or records of one type, and some of their details (fields). Use an overview session to view, sort, add, change, copy, and remove records.

When you add or change a record a details session usually starts. Sometimes, you can add and change records directly using the overview session.

Package

A set of related modules that implements a complete part of the functionality, such as Enterprise Planning, Financials, or Warehouse Management. Packages are designed to function as independent as possible, to enable a customer to implement only particular packages.

A package code consists of two characters. For example, tt is the code of the Tools package.

Each package has a unique VRC version structure.

Package combination

A combination of several different packages with specific VRCs. A package combination represents a complete usable version of LN.

In the **User Data (ttaad2500m000)** session, each user is linked to a package combination. This determines which version of the software the user can use.

In the **Companies (ttaad1100m000)** session, each company is linked to a package combination. This indicates which version of LN is appropriate to handle the data in that company.

Package VRC

A version of a package, such as `tc B61O a cus1`. Usually, one version of a software component, such as a session, a table, or a form, is stored in one particular package VRC.

A developer can usually only modify software components in a particular package VRC.

The code of a package VRC consists of these components:

- Package code, such as `tc`
- A version (VRC) code, such as `B61O a cus1`, built up of these components:
  - Version
  - Release
  - Customer

Patch

In PMC, a patch is a collection of *Solutions*. In general a patch contains solutions created in a larger time period. The patch entity is both known at the *PMC distributor* and *PMC recipient* side. Patches are an indivisible set of solutions. You cannot install or uninstall individual solutions that belong to a patch at the PMC recipient. You can only install or uninstall patches as a whole. Yo can define dependencies between patches. Patches leave the *Base VRC* that is linked to the *update VRC* at the PMC recipient unchanged. The existing PMC registry remains and is extended with data of the newly installed patch. Patches only permit the most recent version of software components to be maintained. Patches in general mainly contain corrective solutions.

Note:

In *PMC versions* earlier than LN 6.1, the synonym Service Packs was often used for patches.

Perspective

A perspective is a group of *views* and *editors* in the *Eclipse* workbench. Each perspective provides a set of functionality aimed at accomplishing a specific type of task or works with specific types of resources. For example, the **Application** perspective combines views you would normally use while editing 4GL scripts and libraries. The **Debug** perspective contains the views you would use while debugging programs. As you work in the workbench, you will probably switch perspectives frequently.

In LN Studio, these perspectives are used:

- **Application**
- **Debug**
- **Integration**
- **Integration Test**
- **Project Server**

PMC distributor

The functional part of PMC that manages the creation of *Updates*. PMC Distributor is especially used by software vendors who create updates.

PMC recipient

The functional part of PMC that manages the installation of *Updates*. Customers, who install updates in particular use PMC recipient.

PMC version

The PMC version is linked to every *Update* and is intended to quarantee that various formats of PMC solution dumps are handled by the right version of the PMC software.

- PMC version 0 : does not support Service Packs
- PMC version 1 : supports Service Packs

Note:

Dumps created for a higher PMC version cannot be processed at the recipient side if PMC recipient has not been upgraded to that PMC version.

Dumps of lower PMC versions can always be processed.

Post-requisite

Post-requisites are mainly meant to prevent the installation of bad solutions. In Project Server, a post-requisite is a link from a more recent, correct, solution to an earlier, bad, solution.

Pre-requisite

Pre-requisites mainly steer the sequence in which solutions are installed. In general a pre-requisite is the link from a more recent solution to a predecessing solution.
Pre-requisites are the most common type of dependencies. A pre-requisite dependency exists between two solutions if one solution must have been installed before the other solution is installed. In that case, the first solution is a pre-requisite for the other solution. Typically, pre-requisite dependencies exist between a solution and a previous solution, if these solutions have one or more components in common. Pre-requisite dependencies can only be created to solutions in the same *Base VRC*.

Primary key

The unique identification for a record in a *table*.

Private VRC

This VRC is derived from the *activity VRC* and contains checked out components the software engineer is working on. The components in this VRC are only accessible by the software engineer to which the activity is assigned. The private VRC is generated automatically when you open the activity for the first time.

Private VRCs are only used if *SCM* is active for the *application* to which the activity belongs.

Product Maintenance and Control

Product Maintenance and Control (PMC) is a tool that helps a customer manage the updates of the Infor LN system.

With the PMC tool, you can check all patches against the customer's Infor LN system to verify their completeness, check any potential interference with the customization, and detect dependencies.

These capabilities ensure the complete and accurate installation of each software patch and Service Pack. Using the PMC tool also enhances the quality of the customer support.

PMC consists of a PMC distributor part and a PMC recipient part.

Programmer's Guide

The *Infor ES Programmer's Guide*. Access this guide through the LN Studio online help.

Program script

A sequence of instructions used to program a number of actions that must take place in addition to the standard program. The different program script types available are *3GL scripts* and *4GL scripts*.

Synonym: UI script

Project Manager

Project managers create and maintain the *Software Projects* and *Activities* in which the software components are developed. The Project manager assigns each activity to a *Software Engineer*.

The projects and activities are stored on the *Project Server*.

Project Server

The Project Server contains the *Software Projects* for *LN Studio*. Each software project contains one or more *Activities* in which the *Software Engineers* develop their software components.

Project VRC

The VRC that contains all finished software components for a *Software Project*. From this VRC, deliveries to customers are performed when the project is completed.

The project VRC is the *Export VRC* linked to the *Base VRC* of the *Application* to which the project belongs. This VRC also contains finished software components for other projects defined for the same application.

Query

The process of extracting information from a database and presenting it in a report.

Question

A notification that requires a choice response. For example, a question can prompt you to confirm or cancel a delete action. If you do not respond to a question, the process that prompted the question cannot continue. Questions are distinguished from *messages*. Messages offer no choice and do not necessarily require a response.

Reference mode

The way in which a reference restricts the contents of a table field.

A reference can have one of the following reference modes:

- Mandatory

  The field must contain a code found in the reference table.

- Mandatory unless empty

  The field can be empty. If not, it must contain a code found in the reference table.

- Not mandatory

  The field can be filled with a code not found in the reference table. The reference speeds up queries.

Reference table

The table to which some table field refers.

Example:

- One of the fields of the Items – General table is the **Country of Origin** (coor) field. This field can contain a country code. (The field can also be left empty.) LN stores country codes in the Countries table. To control this connection, the **Country of Origin** table field in the Items – General table has a reference to the Countries table.
- Items – General is the referral table and Countries is the reference table.

Referral table

The table that has a field that refers to another table.

Example:

- One of the fields of the Items – General table is the **Country of Origin** (coor) field. This field can contain a country code, but can also be left empty. LN stores country codes in the Countries table. To control this connection, the **Country of Origin** table field in the Items – General table has a reference to the Countries table.
- Items – General is the referral table and Countries is the reference table.

Report

A report is used to present data from the database, usually on paper. The report can be sent to a device, such as a physical printer, a display, or a file.

Revision Control System

A tool used by LN Tools to store revisions of scripts, libraries, includes, and report scripts.

rule

A criterion to measure the quality of software components.

The rules used by *VSC* are based on the Infor LN Software Coding Standards (SCS), the Infor LN Software Programming Standards (SPS), and the Infor LN Performance Guide.

These are examples of rules:

- A table code must have the following structure: `pp mmm s xx`, where `pp` is the package code, `mmm` is the module code, `s` is the submodule code (only when numeric), and `xx` is the sequence number.
- A message called in a script should not be expired.
- A query that performs an update must have a 'with retry' clause.

Most rules are hardcoded in the VSC software. However, some rules used to analyze scripts are implemented as *source analyze codes*.

rule database

A database with detailed information about *rules*.

The rule database contains this information:

- Detailed information about the rules implemented in VSC.
- Information about rules that might be added to future VSC versions.

This database is intended for internal use by the LN development department. It is therefore not delivered with the VSC software.

SCM group

A *Software Configuration Management* group in LN that identifies a development group with a separate development environment.

Service Pack

In PMC, a Service Pack is a collection of solutions. In general, a Service Pack contains solutions created in a larger time period. In PMC the term 'patch' is also applied for Service Packs. The patch entity is both known at the PMC distributor and PMC recipient side. A property in the patch entity makes the difference between patches and Service Packs. Service Packs are an indivisible set of solutions. You cannot install or uninstall solutions that belong to a Service Pack at the PMC recipient. You can only install or uninstall Service Packs as a whole. You can define dependencies between Service Packs. Service Packs are intended to enable you to maintain multiple *Base VRCs* in parallel. Service Packs change the base VRC that is linked to the update VRC at the PMC recipient. The existing PMC registry for the update VRC is moved to history and a new registry is started for the update VRC. This type of patch in general contains a significant amount of functional changes.

Note:

Service Packs as described in the preceding definition do not exist in PMC versions earlier than LN 6.1.

Session

A part of LN the user can start to run an application's functionality. Usually, a session is linked to a main database table and a program script. In addition, a session uses zero or more forms, reports, and charts.

The code of a session consists of a package code, a module code, four digits that indicate the main table number and the session type, an m or an s, and three additional digits, such as **Countries (tcmcs0510m000)**.

Software component

The LN software consists of these separate software components:

- Message
- Report
- Label
- Function
- Business Object
- Chart
- Integrated session
- Additional file
- Question
- Session
- Domain
- Table
- Menu
- Form
- Program script
- Library

Software Configuration Management

With software configuration management, developers can modify and test their own revision of a software component. Using a check out and check in functionality, a software component is locked for other developers. This guarantees no more than one developer can modify the same software component simultaneously.

Software Engineer

Software engineers develop software components through the LN Studio. Before editing a component, engineers must first open an *Activity* the *Project Manager* assigned to them.

Software Project

A software project for *LN Studio*. Each software project contains one or more *Activities* in which the *Software Engineers* develop their software components. Each software project is linked to an *Application*. Multiple projects can be defined per release. Software projects are stored on the *Project Server*.

During its life cycle a project can have these statuses:

- Created
- Active
- Cancelled
- Closed
- Finalized

Project Managers can create software projects and activities through the **Software Project Explorer**.

Solution

In PMC, the smallest, indivisible type of update. A solution is identified both at the distributor and recipient side by a unique solution code. The term individual solution is also frequently used and has the same meaning.
Note:

In the PMC software the term solution is often used as an alternative for the term update. A solution can then be an individual solution, which is the smallest, indivisible type of an update, or a patch.

Solution status distributor

The status describes the progress of the maintenance of solutions.

Solution status recipient

The following statuses indicate the progress of the installation or uninstallation of a solution or *Patch*.

These statuses are only used at the recipient side, and must not be confused with the *Solution status distributor* at the distributor side.

- Available

  The solution or patch is scanned and available on the system.

- To Install

  The solution or patch is checked and is ready to be installed.

- Saving

  A backup of the components is being saved.

  This status is not applicable for patches.

- Installing

  During the installing process the solution or patch has this status.

- Installed

  The solution or patch is installed.

- To Uninstall

  The solution or patch is checked to be uninstalled.

- Uninstalling

  During the uninstalling process, the solution or patch has this status.

source analyze code

A code used by *VSC* to perform user-defined checks on scripts, such as UI scripts, DLLs and DALs.

Each source analyze code is linked to an expression text and to a warning message. The expression text contains a search pattern, which is used to find errors.

See "Source Analyze Codes" in the Web Help on the LN server.

Stack frame
An execution context in a suspended session containing local variables and arguments. In Infor 4GL terms, a stack frame represents the function calls of the sources (scripts and includes) related to the debugged session.

String
A data structure that contains a number of characters that represent readable text.

Superseded solution
A superseded solution is a solution for which can be said that all software components are also contained in another so-called *Superseding solution*.

Superseding solution
A superseding solution is a solution that contains at least the same software components as contained in one or more *Superseded solutions* and can contain additional software components that are not part of any superseded solution.

A solution supersedes another solution if the following conditions are met:

- The superseding solution contains at least all the components of the other solution.
- The superseding solution contains newer versions of these components.
- The superseded solution is not yet installed. If the superseded solution is already installed, speaking of a superseding solution is illogical.

Table
A data structure used to store data that consists of a list of records. Each entry is identified by a unique key and contains a set of related values. A table contains a number of table fields that belong to a specific domain.

A table code consists of a package code, module code, and three digits.

Example

Table: tc mcs010 Countries

Code	Label	Length	Data Type
ccty	Country	3	String
dsca	Description	30	Multibyte String
meec	EU Member State	5	Enumerated
...			

Thread
A sequential flow of execution in a *debug target*. A thread contains *stack frames*. Because Infor 4GL is single-threaded, each debug target only contains a single thread that represents the debugged session.

UI script
See *Program script*.

Undo check out
A command that unlocks a checked out software component.

Universal Time Coordinated
A time/date format. LN stores dates and times in UTC format. It stores date and time in a single Long integer called the UTC long format. This integer represents the number of seconds since 0:00 hour, January 1, 1970 (in UTC).

Update
In PMC, an update is a set of changed software components, including PMC metadata, which is required to install the update in a safe and correct way. An update can contain corrective changes or functional enhancements.

Updates can be delivered in four different configurations:

- Solutions
- Collections
- Patches
- Service Packs

Update VRC
A physical VRC at the PMC recipient side in which updates are installed. Every update VRC has a *base VRC* linked.

verification code
A code that identifies a selection of *VSC* checks for software components in one or more package combinations.

A verification code is usually linked to one or more *verification filters*.

verification filter

A selection of *VSC* checks for which warnings must be generated.

A *verification code* is usually linked to one generic verification filter and one or more specific verification filters.

The generic filter defines the checks executed to verify all software components in all packages. Depending on the filter settings, the checks in the generic filter can generate two types of *warnings*:

- "Filtered to Handle" warnings. You must solve or accept these filtered warnings immediately.
- "Non-filtered" warnings you do not have to handle immediately.

Specific filters are used to reduce the number of "Filtered to Handle" warnings for a specific package, module or VRC. Each specific filter is derived from the generic filter, and therefore executes exactly the same checks as the generic filter. In a specific filter, indicate for each check whether you want to generate "Filtered to Handle" warnings or "Non-filtered" warnings.

Note: You cannot disable checks, or add extra checks in a specific filter. If you select a check which is not present in the generic filter, VSC ignores this check.

Verify Software Components

A utility to perform a quality check on the software components developed or changed in Infor LN.

Version - Release - Customer

The version - release - customer (VRC) code is an identification of a stage in the development of the LN software, such as `B61_a_ams`.

A VRC code consists of these components:

- Version

  A stage in the development in which a major part of the software is modified.

- Release

  A stage in the development in which a minor part of the software is modified.

- Customer

  An extension, localization, or customization of the software for a single customer or a small group of customers.

A VRC can be derived from a preceding VRC. Every software component contained in the preceding VRC, and not explicitly modified or set to expired in the current VRC, is available in the current VRC.

Synonym: package VRC

View

A view is a visual component within the LN Studio workbench. It is used to navigate a hierarchy of information, open an *editor*, or display properties for the active editor. Modifications made in a view are saved immediately. Usually, only one instance of a particular type of view may exist within a workbench window.

These are typical LN Studio views:

- **Activity Explorer**
- **Software Project Explorer**
- **Component Explorer**

warning

A message that explains an error found during a *VSC* check.

Depending on the *verification filter* settings, VSC generates the following:

- "Filtered to Handle" warnings. You must solve or accept these filtered warnings immediately.
- "Non-filtered" warnings you do not have to handle immediately.

Zoom session

The session in which you can browse through the available records and select a record. A zoom session is an overview session in read-only mode.

Use a zoom session to enter the code of an existing record, such as an item, order type, or warehouse in another session.

To start a zoom session, click the browse arrow button behind the field or press CTRL+B.