Infor LN Studio Application Development Guide

Copyright © 2016 Infor

Important Notices

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

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

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

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

Trademark Acknowledgements

The word and design marks set forth herein are trademarks and/or registered trademarks of Infor and/or related affiliates and subsidiaries. All rights reserved. All other company, product, trade or service names referenced may be registered trademarks or trademarks of their respective owners.

Publication Information

Release: 10.5

Publication Date: June 22, 2016

Document number: devstudioappldg (U8921)

Contents

About this Guide....13
Contacting Infor....13

Chapter 1: Upgrade notice....15

Chapter 2: Infor LN Studio introduction....17

Chapter 3: Overview....19
LN Studio overview....19
Functions and features....19
User roles....22
Current scope and limitations....22
Architecture....22
Troubleshooting....23
LN Studio workbench....23
Perspectives, editors and views....23
Opening and switching perspectives....25
Running sessions....26
Preferences....26
More information....26
Applications....26

Chapter 4: Installation and configuration....29
First installation....29
Prerequisites....29
Installing LN Studio....30
Selecting the workspace....32
Defining connectivity settings....32
Defining dynamic connection points....34
Setting user preferences....36
Installing LN Studio updates....37

Chapter 5: Activity based development....39
Overview....39
Configuration Management....40
Usage of SCM....40
Software components....42

Development with SCM for project VRCs and applications....42
VRC numbering....44
CM Actions....44
Development without SCM....46
VRC numbering....47
CM Actions....48
Activity context....50
Cleaning an activity....51
Recovering an activity....52
Reassigning an activity....53
Linking run configurations to an activity....54

Chapter 6: Development procedure....55
Developing software components....55
Verifying software components....58
Verifying software components....59
Handling warnings....59
Identifying problems in your code....61
Sample exercise....61
Running and debugging software components....62
Running LN sessions....62
Debugging LN sessions....64
Suspending sessions....68
Resuming the execution of suspended sessions....68
Stepping through the execution....68
Inspecting values....69
Evaluating expressions....69
Using breakpoints....70
Running BFlow tests....74

Chapter 7: InContext Modeling....75
Context messages....75
InContext model....75
Related topics....76

Chapter 8: LN Studio Limitations....77
General limitations....77
Script Editor Limitations....78
Configuration Management (CM) limitations....78
Build limitations....78
Debugger limitations....79
Debug View....79

Variables view....79
Expressions view....79
Script Editor....80
Watchpoint....80
Web UI Limitations....80

Chapter 9: Wizards and Dialogs....81
Activity....81
Activity Information....81
Activity Overview, End Activity....82
Clean activity....82
Open an Activity....83
Reassign Activity....84
Recover activity....84
Recover activity....85
Link Run Configurations....86
Create IEX Patch....87
Browse and Select....87
Browse Command Sets....87
Browse component....88
Browse domain....89
Browse Default Sizes....90
Browse label....90
Browse library....91
Browse product ID....91
Browse Queries....91
Browse session....92
Browse Input Fields....92
Browse text field....93
Browse time Format / Browse date format....93
Select a Chart....93
Select Component(s)....94
Select Component Types....96
Select a Report....96
Select Table Fields....97
Create a New Software Component Wizard....97
Creating an additional file....97
Create a New Infor LN Software Component....98
Creating a domain....100
Creating a function....102
Creating a label....102
Creating a library....104

Creating a menu....105
Creating a message....105
Creating a question....106
Creating a report....107
Creating a session....109
Creating a table....111
Debugging....112
Execute Assignment Expression....112
Line Breakpoint properties....112
Method Breakpoint properties....113
Watchpoint properties....113
Preference Pages....114
Preferences - Editor....114
Preferences - Code Assist....116
Preferences - Folding....117
Preferences - Syntax....117
Preferences - Templates....118
Preferences - Launching....119
Preferences - Multipage....120
Preferences - Workbench....121
Preferences - Infor LN Configuration Management....122
Verify Software Components....123
Verify Components View Filter....123
Verify Warning Details....124
Verify Warning Help....127
Miscellaneous....128
Check In comment....128
Configuration Management....128
Configurations....129
Copy from other layouts....132
Create a new Application....132
Create a New Form Command....136
Duplicate an Infor LN Software Component....137
Application Search....139
Generate Library from WSDL file....143

Chapter 10: Multipage Editors....145

Editor tabs....146
Preferences....147
Starting a multipage editor....148
Additional File Editor....148
Introduction....148

Overview....148
TDE Documentation....149
BFlow Editor....150
Introduction....150
General....150
Variables....151
Actions....151
XML....152
TDE Documentation....152
Business Object Editor....152
Introduction....152
Overview....152
TDE Documentation....153
Domain Editor....154
Introduction....154
Overview....154
Enum - Set Data....163
Documentation....165
TDE Documentation....165
Function Editor....166
Introduction....166
Overview....166
Source....167
Documentation....167
TDE Documentation....167
Label Editor....168
Label Editor....168
Details....173
Library Editor....174
Introduction....174
Overview....174
Source....178
Documentation....178
TDE Documentation....178
Menu Editor....178
Introduction....178
Overview....178
Preview....182
Documentation....182
TDE Documentation....182
Message Editor....183
Message Editor....183

Details....186
Question Editor....187
Question Editor....187
Details....190
Report Editor....190
Overview....191
Input Fields....196
Layouts....200
Source....205
Documentation....205
TDE Documentation....205
Session Editor....205
Session Editor....205
Details....228
Session Model Editor....236
Introduction....236
Session....236
Context Messages....237
TDE Documentation....238
Table Editor....238
Table Editor....238
Details....252
Table InContext Model Editor....257
Introduction....257
Table....257
Context Messages....259
TDE Documentation....261

Chapter 11: Script Editor....263

Editing source code....264
Script editor features....265
Script Editor Preferences....266
Source Tab....267
Markers and Marker bar....268
Overview ruler....271
Editor shortcut menu....272
Keyboard shortcuts....274
Details....275
Comparing Source Code Versions....275
Application Search....275
Code Assist....276
Comments....279

Folding....281
Incremental Find....283
Keyword Search....283
Mark Occurrences....284
Open Declaration....285
Quick Outline view....288
Text Hovering....289
ToDo Comments....290
Toggle Breakpoints....290
Generating Source from WSDL....291

Chapter 12: Perspectives....293
Application perspective....293
Debug perspective....294

Chapter 13: Views....297
Activity Explorer view....297
Toolbar....298
Shortcut menu....299
Icons and decorators....303
Breakpoints view....306
Toolbar....306
Shortcut menu....307
Icons and decorators....308
BFlow Palette view....308
Component Explorer view....309
Toolbar....310
Shortcut menu....310
Icons and decorators....311
Debug view....312
Toolbar....312
Shortcut menu....313
Icons....314
Limitations....314
Dependent InContext Models view....314
Toolbar....315
Shortcut menu....315
Expressions view....315
Toolbar....316
Shortcut menu....317
Icons and decorators....317
Limitations....318

History view....318
Columns....319
Toolbar....320
Shortcut menu....321
Outline view....321
Toolbar....322
Shortcut menu....323
Icons....323
Problems view....324
Toolbar....325
Shortcut menu....325
Icons....325
Progress view....326
Toolbar....326
Icons....327
Tasks view....327
Toolbar....328
Shortcut menu....328
Icons....329
Variables view....329
Toolbar....330
Shortcut menu....331
Icons and decorators....331
Limitations....332
Verify Components view....332
Toolbar....333
Shortcut menu....334
Icons....334

Chapter 14: Infor LN Project Server....335

Infor LN Project Server introduction....335
Procedures....335
Defining a software project....335
Delivering software components....336
Dialogs....339
Create a Software Project....339
Create a new Activity....340
Compile activity overview....341
Open a Delivery Activity....341
Create a Solution....344
Add development activity....346
Browse Solutions....347

Browse Solution Types....349
Select Component(s) for delivery....350
Project Server View Filter....353
Multipage Editors....353
Solution Editor....353
Perspectives....357
Project Server perspective....357
Views....357
Software Project Explorer view....357
Delivery Explorer view....360

Glossary....363

About this Guide

Document Summary

This document describes how developers must use Infor LN Studio to develop new Infor LN applications, or to customize existing software components.

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 Administration Guide*

Contacting Infor

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

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

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

Upgrade notice

1

Important note for users upgrading from Application Studio 8.4.2 to Infor LN Studio 10.5

If you have implemented Application Studio 8.4.2 already, the following additional steps must be performed before you can start to use version 10.5:

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 using the same workspace.

   To define those development servers:

   a. Install LN Studio 10.5 (note that the development server and project server must have been upgraded already to Enterprise Server 8.5 or higher). See "Installing LN Studio" on page 30.
   b. Start LN Studio, using a new workspace.
   c. Configure the Project Server connection. See "Defining connectivity settings" on page 32.
   d. Define the development environment(s) in the Application Explorer view. See "Defining a development environment".

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 32.
   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" on page 52.

   Note that LN Studio will ask to define the dynamic connection points when they are needed. For details on dynamic connection points, see "Defining dynamic connection points" on page 34.

Infor LN Studio introduction

2

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 Global Solutions 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.

Overview

3

LN Studio overview

This overview describes the following:

- Functions and features of the LN Studio
- Current scope and limitations
- Architecture of the LN Studio solution

Functions and features

This section describes some key characteristics of LN Studio.

Activity based development

The software development process in LN Studio is activity based. Each modification done on the software must be linked to an *activity*. Each activity is part of a *software project*. A software project is linked to an *application*. You can create new software projects and activities from the LN Studio, or select existing projects and activities. Administrators can create new applications. For more information, see "Activity based development" on page 39 and "Applications" on page 26.

Configuration Management (CM)

Currently the configuration management in LN Studio is based on the *Software Configuration Management (SCM)* module in Enterprise Server. SCM is based on RCS, a freeware third party tool available for the UNIX and Windows platforms, and supports the *Check out* and *Check in* of various types of software components. A more extensive CM solution will be offered in a future release. For details on the usage of SCM, and information on how to set up an SCM environment on the LN server, see "To use the Software Configuration Management system (SCM)" in the LN Web Help.

Workbench

The LN Studio workbench is based on Eclipse.

This table shows the features of the LN Studio workbench:

Feature	Description
Wizards	LN Studio contains various wizards that you can use to create software components, such as sessions, menus, labels, libraries and messages. See "Create a New Infor LN Software Component" on page 98. You can edit the new components using component-specific Multipage Editors.
Multipage Editors	LN Studio contains various multipage editors that you can use to edit software components, such as sessions, menus, labels, libraries and messages. See "Multipage Editors" on page 145.
Graphical Editors	Editors for modeling Business Interfaces and their implementations.
Script editor	An editor that supports the 4GL programming language. A content assistant is available, supporting syntax highlighting and completion of keywords. The editor also supports an outline of the scripts or libraries. See "Script Editor" on page 263.
Activity Explorer view	Shows a tree view of the software components in the activity that you have opened, and their relation to each other. Each activity is part of a software project and is assigned to a user (software engineer). Activities and software projects are stored on the *Project Server*. The Activity Explorer shows these components: • Additional File • Business Object • Domain • Function • Label • Library • Menu • Message • Question • Report • Session • Table The view's shortcut menu enables you to perform Configuration Management related actions, such as check in and check out, on these components. See "Activity Explorer view" on page 297. **Note:** • To create a new software component, you must use the Create a New Infor LN Software Component wizard. • To link a software component, which already exists on the LN server, to the activity that you have opened in the Activity Explorer, complete one of these steps:

Feature	Description
	• Use the **Get** command in the Component Explorer view. • Use the **Select a Software Component** command in the LN Studio toolbar. • Use the **Open Declaration** command in the Script Editor. • Project Managers can create new activities through the Software Project Explorer view.
Problems view	Shows errors and warnings from a project build (compilation). See "Problems view" on page 324.
Tasks view	Shows *Eclipse tasks*. Each Eclipse task represents an action that needs to be done, for example modify a particular line in a script. If a task is associated to (a line in) a script or library, you can double-click the task to edit the script or library. The editor opens the script or library at the involved line. You can create Eclipse tasks through the Tasks view and from the LN Studio script editor. For more information, see "Tasks view" on page 327, "Source Tab" on page 267, and "ToDo Comments" on page 290.
Verify Components view	Shows the *warnings* that are generated when you verify software components. The verification process performs various checks, based on the LN design principles. For more information on the verification of software components, see: • "Verifying software components" on page 58 • "Verify Software Components (VSC)" in the Web Help on the LN server.
Build option	Synchronizes the modified sources with the server, and starts the compilation using the generators and compilers on the server. Any problems that occur during the build process are displayed in the Problems view.
Run / Debug options	The **Run** command enables you to run sessions from the LN Studio. The **Debug** command runs a session in debug mode. You can set breakpoints before you start the **Debug** command. From the debugger you can execute various debug commands, for example: you can suspend a running session, so that you can inspect it more closely. See "Running LN sessions" on page 62 and "Debugging LN sessions" on page 64.

For details on the these features, see the "LN Studio workbench" on page 23 topic.

Note: To use LN Studio for customizations, such as changing reports and forms, you do not need an additional license. However, if you want to use LN Studio for development activities, such as creating new tables, editing UI scripts, functions, or libraries, you need a development license, and if necessary source codes.

Some LN Studio functionality depends on the Enterprise Server version of the LN server. For an example, see "Application Search" on page 275.

User roles

This table shows the user roles that can be distinguished in LN Studio:

Role	Description
Administrator	The administrator creates and maintains the *applications* for which the software projects are defined. For more information on the administrator's tasks, see "Defining an application".
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. For more information on the project manager's tasks, see "Defining a software project" on page 335.
Software Engineer	Software engineers develop software components through the LN Studio. Before editing a component, an engineer must first open an activity that the *Project Manager* assigned to him/her. For more information on the tasks of a software engineer, see "Developing software components" on page 55.

Current scope and limitations

Currently the configuration management in LN Studio is based on the SCM module in Enterprise Server.

In future versions, the configuration management will be enhanced. In addition, database modeler and Workflow tools will be added.

For detailed information on the limitations of the LN Studio, see "LN Studio Limitations" on page 77.

Architecture

The LN Studio is implemented in the Eclipse 4.3 framework.

The LN Studio workbench runs on the user's client PC (desktop).

The software components, and the *Package VRCs* to which they belong, reside on the LN server. The development projects, to which the components are linked, can reside on the same LN server. However, you can also configure a dedicated server to store the project data.

The connection between the client PC and the server(s) is based on JCA: the LN Studio connects through a JCA connectivity plug-in on the client PC to the server.

For details on Eclipse, see the Eclipse online help and http://www.eclipse.org/.

Troubleshooting

If you encounter any error messages, for example on connectivity, read these log files for detailed information:

- The Eclipse log files in your local workspace. To find the location of your workspace, select **File > Switch Workspace** in the Eclipse workbench. The Eclipse log file is located in the `.workspace\.log` file in the workspace. If LN Studio is running, you can also view the log messages in the Error Log view in LN Studio.
- The log file of the Connectivity plug-in. To find the location of this log file:
  1. Select **Windows > Preferences**. The Preferences dialog box is displayed.
  2. In the tree that is displayed in the left pane of the dialog box, expand the **Infor LN JCA Connectivity** node. Then, click **Log configuration**. The log file location is displayed.

LN Studio workbench

The *LN Studio* workbench is a desktop development environment that is based on the *Eclipse* framework.

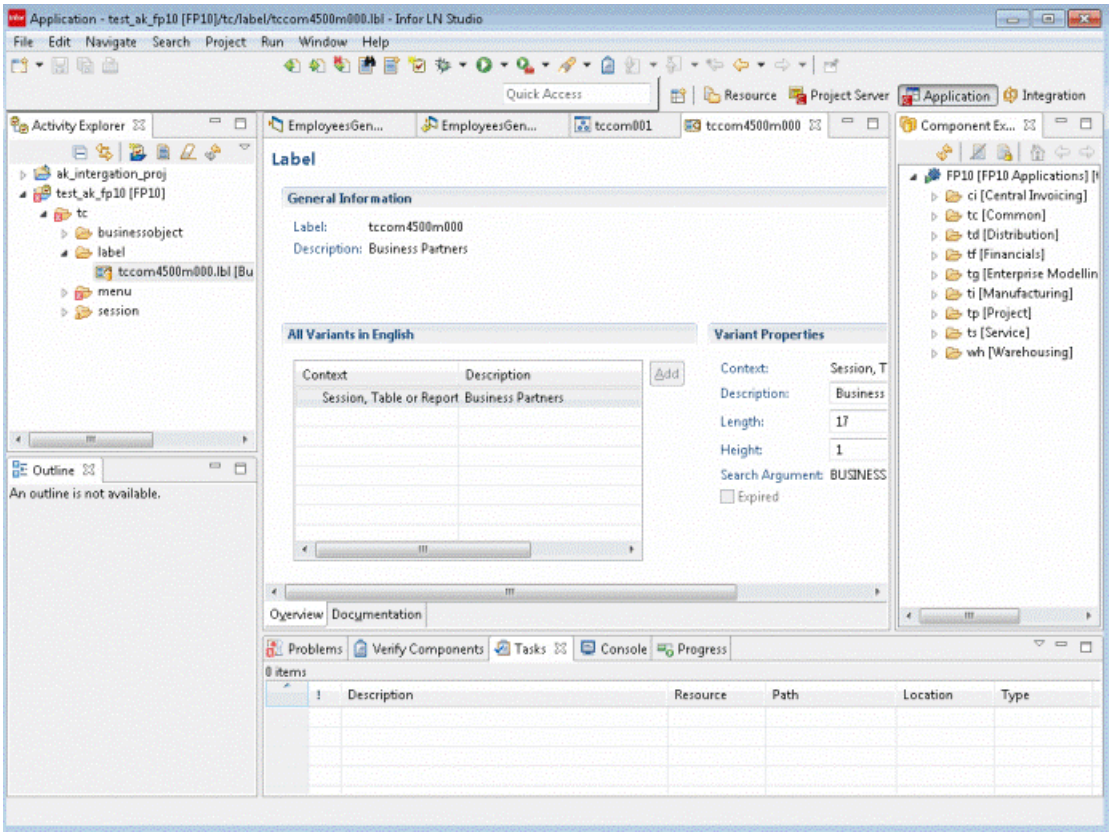
You can use the workbench to perform these tasks:

- Define *software projects* and *activities* in *Project Server*.
- Open an activity in the Activity Explorer view.
- Edit components such as *Sessions*, *Reports*, and *DAL* scripts.
- Run software components. For example, run a session after you changed the corresponding UI script.
- Debug software components.
- Edit Business Interfaces.
- Generate runtime code for Business Interfaces.
- Test Business Interfaces.

Perspectives, editors and views

You can open multiple Workbench windows simultaneously.

This figure shows a sample Workbench window:



In a Workbench window, you can open multiple *perspectives*, and you can switch between the perspectives that you have opened. The name of the active perspective is displayed in the title of the window. See the previous figure for an example.

A perspective contains editors and views, and controls what appears in certain menus and tool bars.

An editor is a visual component within the LN Studio workbench that is typically used to edit or browse a resource, such as a session 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 an LN Studio workbench window.

A view is a visual component within the LN Studio workbench. It is typically used to perform these actions:

- Navigate a hierarchy of information. For example, browse the resources in the development repository.
- Open an editor.
- Show properties for the active editor.

Modifications made in a view are saved immediately. Normally, only one instance of a particular type of view may exist within a workbench window.

These views are examples of typical LN Studio views:

- Activity Explorer
- Software Project Explorer
- Component Explorer

- Verify Components

In LN Studio, these perspectives are used:

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

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 that you would commonly use while editing scripts and libraries. The Debug perspective contains the views that you would use while debugging programs. The Integration perspective is used when developing Business Interfaces. As you work in the workbench, you will probably switch perspectives frequently.

For details on these perspectives, see:

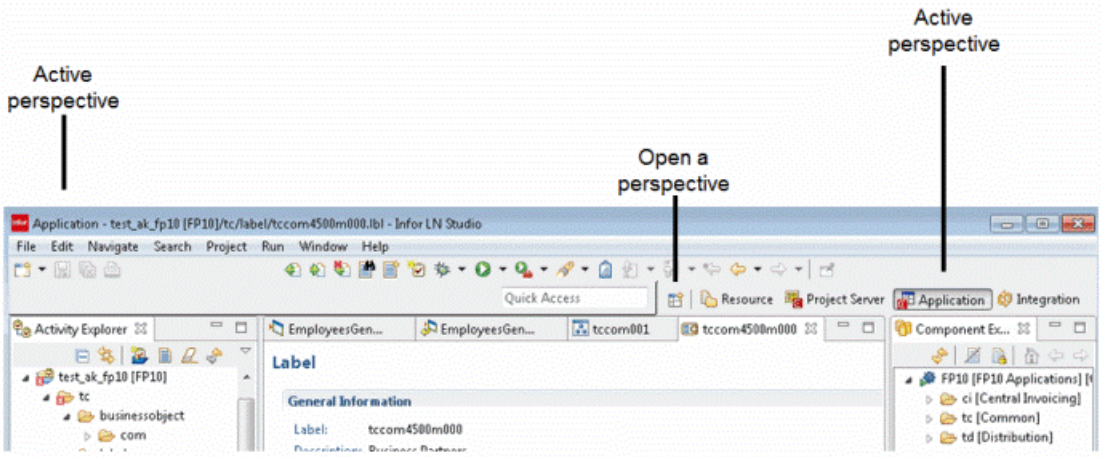
- "Application perspective" on page 293
- "Debug perspective" on page 294
- "Project Server perspective" on page 357
- Infor LN Studio Integration online help
- *Infor LN Studio Integration Development Guide*

Opening and switching perspectives

You can open a new perspective through the **Open Perspective** command on the **Windows** menu.

Alternatively you can use the shortcut bar in the top right corner of the window. See the following figure. This shortcut bar allows you to open new perspectives and switch between ones already open.

The name of the active perspective is shown in the title of the window and its item in the shortcut bar is highlighted. In this example, the LN Studio perspective is in use.



Running sessions

From the **Run** menu, you can run any session on the LN server.

See "Running LN sessions" on page 62.

Preferences

You can set various user preferences for the LN Studio workbench. For example: connectivity settings, preferences for the editor, and label decorations.

See "Setting user preferences" on page 36.

More information

The LN Studio workbench is based on Eclipse.

For details on the standard Eclipse functionality, see the *Workbench User Guide* in the Eclipse online help. This guide contains useful information and tutorials on various topics.

For information on the extra functionality developed by Infor, see "LN Studio overview" on page 19.

To access the *Workbench User Guide* and the LN Studio online help, select **Help > Help Contents**.

Applications

This topic describes the concept of *applications*.

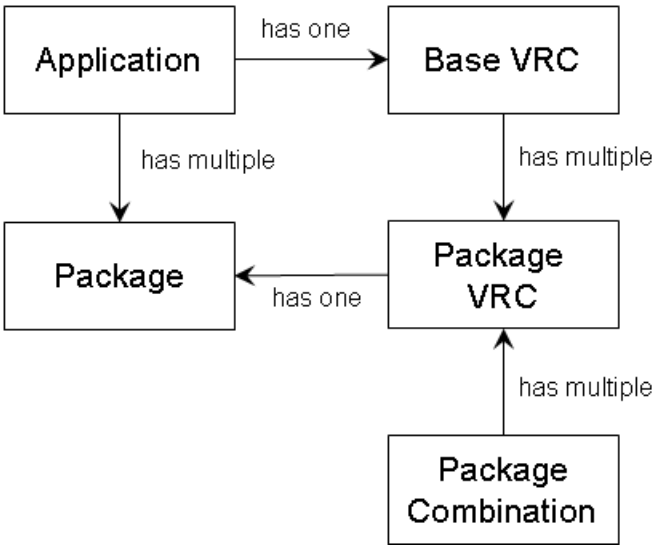
LN Studio uses *applications* to group software components created or modified during a software development project. This is done by bundling one or more LN packages.

An LN Studio application comprises a consistent set of packages that functionally belong together. Each application is linked to a *Base VRC*. This determines the environment on the LN server in which the application's software components are stored.

For each application, you can define one or more projects, in which the software engineers can develop their software components.

The Base VRC of an LN Studio application corresponds with 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. Therefore, all packages assigned to an application are linked to the *Export VRC* of the *Base VRC* of the PMC module and the other way around. Therefore, all packages that have the same VRC as the PMC Export VRC represent one application.

This diagram illustrates the relationships between applications, packages, and VRCs:



Note: You can only delete, modify or create components belonging to the packages linked to the application. Therefore, the Component Explorer shows the components from the packages linked to the application. The Create a New Infor LN Software Component wizards only allow you to create new components in these packages. This restriction also applies to the Duplicate an Infor LN Software Component wizard.

Installation and configuration

4

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 higher.
- Infor Enterprise Server AddOn 8.5 or higher.

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 higher.

- Infor Enterprise Server AddOn 8.5 or higher.

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 higher
- Java Runtime Environment (JRE) 1.7 (Java 7) or higher

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

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 29.
- Before you can install LN Studio, the system administrator must give you access to the zip file with the LN Studio software.

  See "Preparing the installation".

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

2. Extract the LN Studio zip file.

   Open the zip file that contains the LN Studio software, and extract all contents of the file to the new installation folder.

   For LN Studio 10.5 these zip files can be available:

   - `InforLNStudio_10.5.0.nnn-x86.zip` (32-bits)
   - `InforLNStudio_10.5.0.nnn-x86_64.zip` (64-bits)

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

   Note: The selection of the zip file depends on the JRE version. If you use a 32-bits JRE on a 64-bits machine, you must select the 32-bits zip file.

Note: If you upgrade from Application Studio 8.4.2 to LN Studio 10.5, you must perform the actions described in the "Upgrade notice" on page 15.

Post installation steps

1 Select the workspace.

See "Selecting the workspace" on page 32.

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 32.

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 32.

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 will be 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 Activation Type **BW**, **Rexec**, or **BaanLogin**, and specify the corresponding settings. 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 connec-

Connection point	Points of attention
	tion points. As a result, only one bshell is used to connect to the server, instead of two, which improves the performance.
Debug	Select Activation Type **Socket-In**. Select a free local port number, for example 7900.
RuntimeRepositoryConnection	Select Activation Type **BaanLogin** (recommended), **Rexec**, or **BW**, and specify the corresponding settings. The company number for this connection point must be **000**.
TestServerConnection	Select Activation Type **BaanLogin** (recommended), **Rexec**, or **BW**, 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".

If you want to run or debug sessions after creating or modifying the Debug connection point, you must restart LN Studio.

Defining dynamic connection points

Dynamic connection points

LN Studio uses the following dynamic *Connection Points*:

Dynamic connection point	Description	Configuration
Administrator	Enables you to perform administrative tasks on the LN server. The connection point is used, for example: • When an *administrator* defines a new *application* from the Application Explorer. • When a *Project Manager* links a project to the corresponding application (through the Project Properties dialog).	Configure this connection point for each LN server, on which you want to perform administrative tasks or development tasks.
Development Address	Is used to connect to the development repository on the LN server. This connection point is used, for example, when a software engineer: • checks out, or checks in a software component.	Configure these connection points for each *software project*, in which you develop software.

Dynamic connection point	Description	Configuration
	• edits a script or library. • builds (compiles) a session.	
Runtime Address	Sends information to the LN server when you run or debug a session from the LN Studio.	

You are prompted to configure these connection points, the first time you open an *activity* linked to the software project. See the following section.

Configuring dynamic connection points

To configure the dynamic connection points mentioned, complete the following steps:

1 Select the project

Complete these steps:

a Open the Application perspective: On the **Windows** menu, select **Open Perspective**, and subsequently click **Other**. Select **Application** from the list and click **OK**.

b In the Activity Explorer view, click **Open an Infor LN Studio Activity**. The Open an Activity dialog box is displayed.

c Select the *software project* from the list. A question window is displayed: you are prompted to configure the corresponding Administrator connection point.

Note: This question is displayed only the first time you open an activity for the project.

2 Configure the Administrator connection point

Complete these steps:

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

b Specify the properties for the connection point.

Points of attention:

- Select Activation Type **BW**, **Rexec**, or **BaanLogin**, and specify the corresponding settings.
- 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 1 bshell is used to connect to the server, instead of 2. This improves the performance.

See the wizard's online help.

c To save the connection point and to close the wizard, click **Finish**. A question window is displayed: you are prompted to configure the Development Address and Runtime Address connection points.

3 Configure the Development Address and Runtime Address connection points

Complete these steps:

a In the question window, click **Yes**. The Configure Connection point wizard starts and automatically generates these connection points:

- <project name>_dev (Development Address)
- <project name>_rt (Runtime Address)

b Specify the properties for these connection points.

Points of attention:

- **Development Address**
  - Select Activation Type **BW** and specify the corresponding settings.
  - The company number for this connection point must be **000**.
- **Runtime Address**
  - Select Activation Type **BW** and specify the corresponding settings.
  - A runtime address is always linked to a particular company number, such as company 100, on the LN server.

If the development repository and the application data accessed through these connection points reside in the same LN environment, you can share these connection points. As a result, only 1 bshell is used to connect to the server, instead of 2. This improves the performance.

See the wizard's online help.

c To save the connection point and to close the wizard, click **Finish**.

You can now open an activity for the project.

Modifying existing connection points

To modify the configuration properties of existing connection points:

1. Select **Windows > Preferences**. The Preferences dialog box is displayed.
2. In the tree in the left pane of the dialog box, select **Infor LN JCA Connectivity**. The Infor LN JCA Connectivity dialog box is displayed. The dialog box shows the connection points. See the dialog box's online help.
3. To configure a connection point, select the connection point and click **Edit**. The Configure Connection point wizard starts. Modify the properties for the connection point and close the wizard.

Setting user preferences

You can use the Preferences dialog box to set various user preferences.

To set preferences:

1. Select **Window > Preferences.**

   The Preferences dialog box is displayed. This dialog box contains several preference pages.

2 Specify your preference settings. To open a preference page, select the corresponding item in the tree structure in the left pane of the Preferences dialog box.

This table shows the preference pages that are relevant for the Application perspective:

Item in menu structure in the left pane	Preference page
Infor LN JCA Connectivity	Infor LN JCA Connectivity This page is part of the connectivity plug-in. See the page's online help.
	Log configuration This page is part of the connectivity plug-in. See the page's online help.
Infor LN Studio Application	"Preferences - Editor" on page 114
	"Preferences - Code Assist" on page 116
	"Preferences - Folding" on page 117
	"Preferences - Syntax" on page 117
	"Preferences - Templates" on page 118
	"Preferences - Launching" on page 119
	"Preferences - Multipage" on page 120
	"Preferences - Workbench" on page 121
Team	"Preferences - Infor LN Configuration Management" on page 122

Other preferences

The Preferences dialog box also contains preference pages for the Integration perspective in LN Studio. See the documentation for the Integration perspective.

For information on the other preference pages, see "Preferences" in the "Reference" section in the *Workbench User Guide*, or use the online help function (F1) in the pages.

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. See "Preparing the installation".

To install updates:

1 Open the LN Studio zip file

Open the zip file that contains the new software.

For LN Studio 10.5 these zip files can be available:

- `InforLNStudio_10.5.0.nnn-x86.zip` (32-bits)
- `InforLNStudio_10.5.0.nnn-x86_64.zip` (64-bits)

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

Note: The selection of the zip file depends on the JRE version. If you use a 32-bits JRE on a 64-bits machine, you must select the 32-bits zip file.

2 Extract the LN Studio zip file

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!

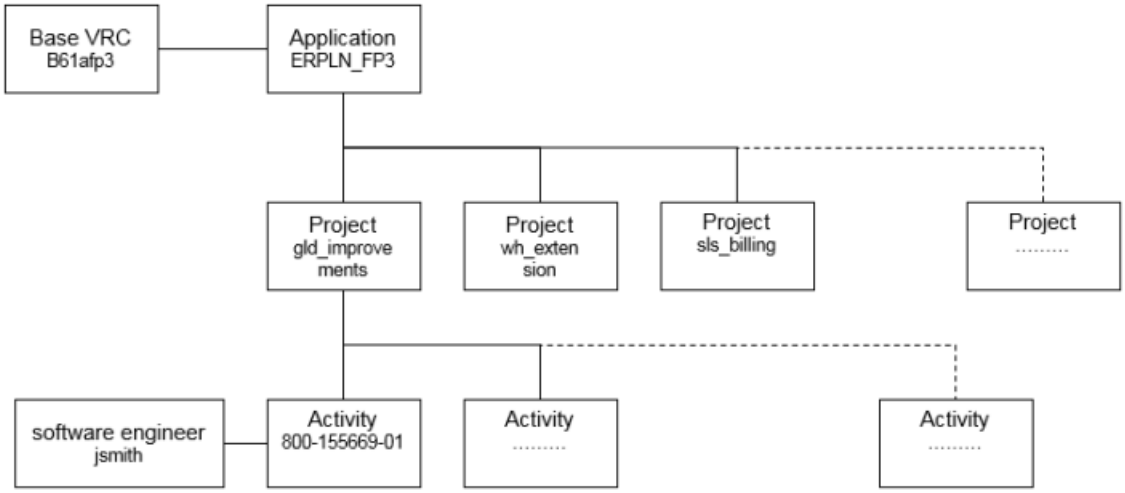
Activity based development

5

Overview

The software development process in LN Studio is activity based. Before a software component can be modified, a software engineer must link the component to an *activity*. Each activity is part of a *software project* and each software project belongs to an *Application*.

This diagram shows an application, software projects, and activities:



A software application is a consistent set of packages that functionally belong together. An application is linked to a *Base VRC*, which determines the environment on the LN server in which the application's software components are stored. *Administrators* can create applications through the "Application Explorer view". See "Defining an application".

For each high level software requirement (business requirement), a *software project* is created on the *Project Server*. Multiple projects can be defined per application. A software project is split up into activities. *Project managers* can create software projects and activities. They can also view the modified LN software components per activity, through the Software Project Explorer. A separate activity is created for each individual software requirement, such as a bug fix or a new feature within the project. Each activity is assigned to a *software engineer*. See "Defining a software project" in the Project Server documentation.

Software engineers can open their activities through the Activity Explorer view.

See "Developing software components" on page 55.

Note:

- Components in an activity are only accessible for the software engineer to which the activity is assigned. However, other software engineers can access these components if they put this activity in the activity context of their own activity.

  See "Activity context" on page 50.

- Activities can be shared over multiple software engineers when their names are added as Assignee in the Activity properties. In this case they also can modify the components.

Configuration Management

The configuration management in LN Studio is based on the *Software Configuration Management (SCM)* module in Enterprise Server. SCM is based on RCS, a freeware third party tool available for the UNIX and Windows platforms. SCM supports the *Check out* and *Check in* of various software components. A more extensive CM solution will be offered in a future release. For details on the usage of SCM, and information on how to set up an SCM environment on the LN server, see "To use the Software Configuration Management system (SCM)" in the LN Web Help.

Note: Because of SCM limitations, the LN Studio configuration management is only available for main component types such as forms, sessions, table definitions, reports, and functions. The 4GL component types, such as labels, questions, and messages, are not supported.

Usage of SCM

The usage of SCM has an impact on the VRC structure on the LN server. It also impacts the way configuration management actions, such as check in and check out, work.

The usage of SCM is optional. If you use SCM, you must at least activate SCM for the project VRC(s) on the LN server. Additionally, you can enable SCM for your LN Studio application(s).

The following situations can be distinguished:

Scenario	Description
SCM for project VRCs and applications	SCM is activated for the project VRC(s) on the LN server and for the applications in LN Studio. This is the preferred scenario if your LN server supports SCM. Each time you check in a component, LN Studio generates a new revision. Other software engineers can use (not modify) your checked in components. See "Development with SCM for project VRCs and applications" on page 42.

Scenario	Description
No SCM	SCM is disabled for the project VRC(s) on the LN server and applications in LN Studio. LN Studio does not generate any revisions. Other software engineers can use (not modify) your checked in components and your checked out components. See "Development without SCM" on page 46.

To activate SCM for project VRCs and applications

Activate SCM for a project VRC

To activate SCM for a *Project VRC*:

1. Log on to the LN server.
2. Start the Package VRCs (ttadv1511m000) session, and create a new (SCM) VRC. The new VRC must be derived from the project VRC.
3. Start the (De)activate SCM and Component Management by Package VRC (ttscm0501m000) session. The session shows a list of VRCs. 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.
4. Select the new SCM VRC and click **OK**.
5. To make the changes effective, log off and log on again.

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

Activate SCM for an application

Note: You can only activate SCM for an application if SCM is active for all project VRCs in the application.

To activate SCM for an application:

1. Start LN Studio. Open the Application perspective.
2. Go to the Application Explorer view.
3. Expand the correct development environment.
4. Expand the 'Applications' tree structure. Double-click the application to start the Application properties dialog.
5. Select the **Use SCM** check box.
6. Save the changes. Close the dialog.

Software components

When a personal activity in the LN Studio is open, you can link software components from the LN dictionary to the project. You can also create new components via the LN Studio.

You can link or create these components:

- Additional File
- Domain
- Function
- Label
- Library
- Menu
- Message
- Question
- Report
- Session
- Table

These software components are the starting point of development for a software project.

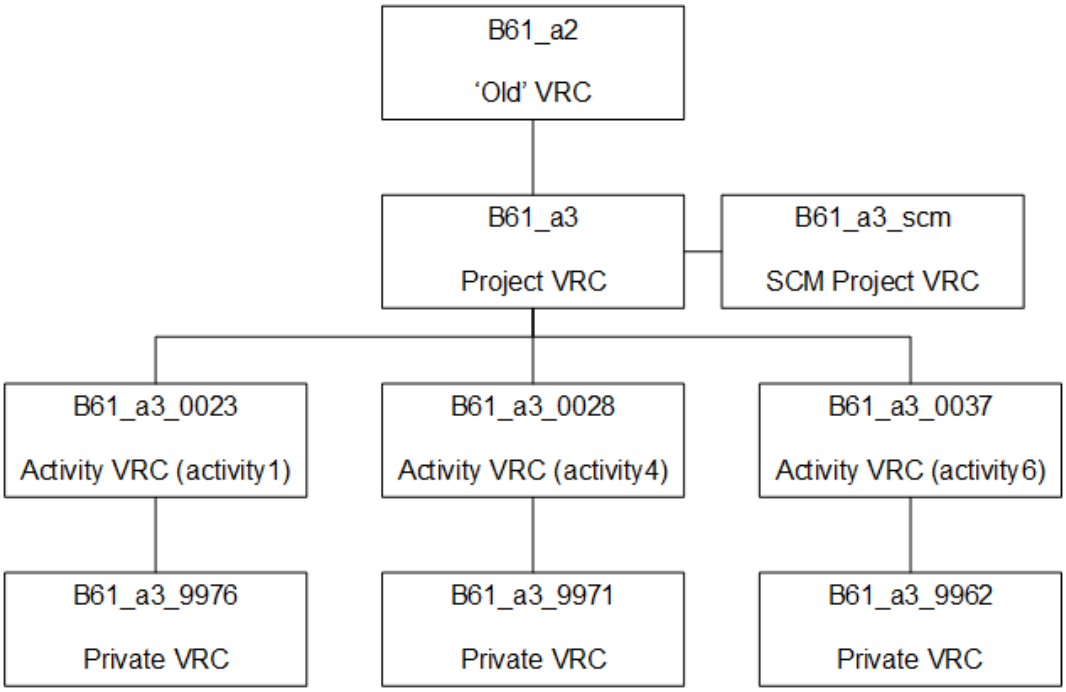
The LN Studio shows a tree of software components on the LN server and a tree of components in your current activity. For details on the layout of the LN Studio, see the "LN Studio workbench" on page 23 section.

Development with SCM for project VRCs and applications

VRC structure

If SCM is enabled for project VRCs and applications, LN Studio generates two VRCs on the LN server for each activity: an *Activity VRC* and a *Private VRC*. In these VRCs, a software engineer can modify software components without disturbing other software engineers.

This diagram shows an example:



These VRCs are used:

- The 'old' VRC contains software components developed in a previous project.
- The *Project VRC* contains all finished software components for the project. From here, deliveries to customers are done, once the project is completed.

  Note: The project VRC is the *Export VRC* linked to the *Base VRC* of the *Application* to which the project belongs. Therefore, the project VRC contains finished software components for other projects defined for the same application.
- The SCM Project VRC is derived from the project VRC. It is used to store checked out software components.
- The Activity VRC contains checked in software components for an activity. The activity VRC is generated automatically when you open the activity for the first time. The components in this VRC are only accessible for the software engineer to which the activity is assigned. However, other software engineers can access these components if they put this activity in the activity context of their own activity.

  See "Activity context" on page 50.
- The Private VRC is derived from the activity VRC. It contains the 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.

Note: Various software engineers can work at the same activity. In that case, multiple private VRCs exist for the same activity VRC.

VRC numbering

The numbers in the customer extension of the activity and private VRCs are generated automatically:

- Activity VRCs: Numbering starts with 0001. The number is increased for each new activity VRC. The maximum number is 4999.
- Private VRCs: Numbering starts with 9999. The number is decreased for each new private VRC. The minimum number is 5000.

Example

LN Studio generates these VRCs:

- For the first activity: activity VRC B61_a3_0001 and private VRC B61_a3_9999.
- For the second activity: activity VRC B61_a3_0002 and private VRC B61_a3_9998.
- For the third activity: activity VRC B61_a3_0003 and private VRC B61_a3_9997.

Note: The numbers of an activity VRC and the corresponding private VRC do not always have to be complementary. This is because multiple private VRCs can exist for the same activity.

When you end an activity, LN Studio performs a roll-up of all component revisions and revision texts. The resulting components and texts are moved to the project VRC. The empty activity VRC and private VRC are released, allowing these VRCs to be reused in the future.

CM Actions

Check out

You must check out a component before it can be modified. LN Studio checks whether the component is already present in your activity VRC (for example because of an earlier check out and check in action).

- LN Studio does not support parallel development for applications. Therefore, if the component is not present in your activity VRC, LN Studio locks the component in the *Project VRC*, so that other software engineers cannot check out the component, and copies the component to your private VRC where it can be modified. The component stays locked until you end your activity.
- If the component is already present in the *Activity VRC*, LN Studio copies the component from your activity VRC to your Private VRC. The component in the project VRC stays locked.

To check out a software component in LN Studio:

1 Right-click the component in the Activity Explorer view.

2 Select **Team > Checkout**.

Note: If you try to change a component which is not checked out, you are prompted whether you want to check out the component.

Check in

You can check in a component when you finish the changes to the component.

LN Studio moves the checked out component from your private VRC to your activity VRC. A historical version (revision) is generated, for which you must enter a revision text.

Note:

- The checked in component stays in the activity VRC. The component is not copied to the project VRC. The component in the project VRC stays locked and not accessible to all software engineers. To unlock the component in the project VRC and to make it accessible for other software engineers, end or cancel the activity, or delete the component from the activity.
- Other software engineers can use (not modify) your checked in components. This is possible if your activity is inserted into the activity context of their own activity.

  See "Activity context" on page 50.

To check in a software component in LN Studio:

1. Right-click the component in the Activity Explorer view.
2. Select **Team > Checkin**. The Check In comment dialog box starts. Enter a comment and click **OK** to check in the component.

Note:

- The revision text is stored in the configuration management system on the server. A single text is used for all revisions of a component. This text is displayed each time you check in a new revision of the component, so you can modify or extend it. If you check in multiple components together, you can use the same revision text for all components.

  See "Check In comment" on page 128.

- If Verify Software Components (VSC) is active and blocking VSC warnings are present, *Check in* is not possible.

Undo checkout

You can undo the check out of a component. LN Studio removes the component from your private VRC and cancels all changes you made since the last check in of the component. The component in the activity VRC is not removed.

Note:

The component in the project VRC stays locked and is not accessible for other software engineers. To unlock the component in the project VRC and to make it accessible for other software engineers, end or cancel the activity, or delete the component from the activity.

To undo the check out of a software component in LN Studio:

1. Right-click the component in the Activity Explorer view.
2. Select **Team > Uncheckout**.

End Activity

When you finish all changes in an activity, you can end the activity so the changed components are released to the project VRC.

When you end an activity, LN Studio performs these actions:

- Creates a roll-up of all component revisions and revision texts in the activity VRC.

- Moves the resulting software components and texts from the activity VRC to the project VRC. Therefore, components become available to other software engineers.
- Unlocks the components in the project VRC.
- Releases the private VRC and the activity VRC, so they can be reused in the future.
- Disconnects the corresponding run configurations from the activity. You can link these configurations to another activity. See "Linking run configurations to an activity" on page 54.

Note: You can only end an activity if all components in that activity are checked in.

To end an activity in LN Studio:

1 Right-click the activity in the Activity Explorer view.
2 Select **Team > End activity**.

Note: If not all components are checked in and the **Partial End Allowed** check box in the software project properties is selected, you can partially end the activity. To do this, in the Activity Overview, End Activity dialog box, select a subset of the components and click **OK**. See the dialog box's online help.

Cancel Activity

To undo all changes in an activity, cancel the activity.

LN Studio performs these actions:

- Removes all components from the activity.
- Unlocks the components in the project VRC.
- Releases the private VRC and the activity VRC, so they can be reused in the future.

To cancel an activity in LN Studio:

1 Right-click the activity in the Activity Explorer view.
2 Select **Team > Cancel Activity**.

Cancel a range of activities

To undo all changes in a range of activities, run the Global Cancel Activity (ttadv1223m000) session on the LN server.

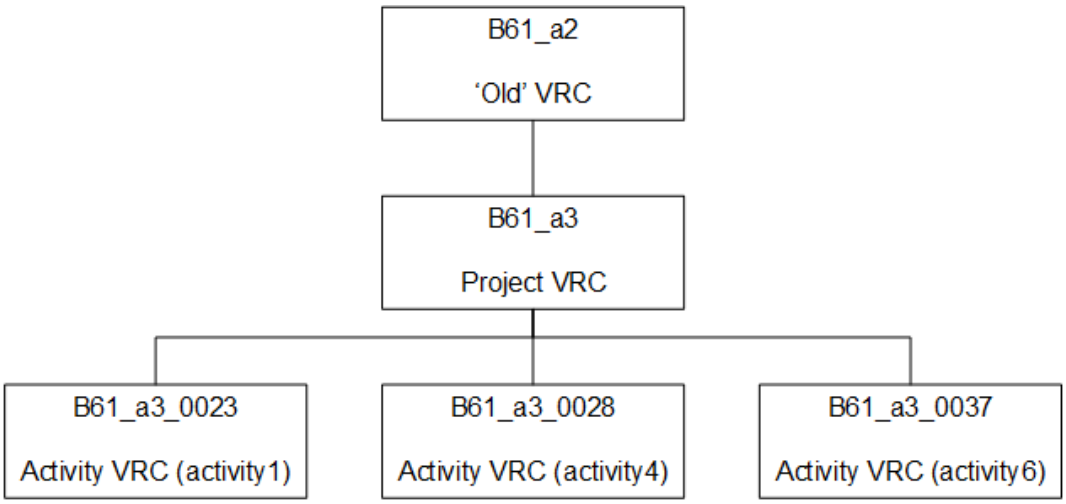
Note: This session does not update the status of the canceled activities in the Project Server. See the session help.

Development without SCM

VRC structure

If SCM is disabled for an application, LN Studio generates an *Activity VRC* on the LN server for each activity. In this VRC a software engineer can modify software components without disturbing other software engineers.

This diagram shows an example:



These VRCs are used:

- The 'old' VRC contains software components developed in a previous project.
- The *Project VRC* contains all finished software components for the project. From this VRC, deliveries to customers are carried out, once the project is completed.

  Note: The project VRC is the *Export VRC* linked to the *Base VRC* of the *Application* to which the project belongs. Therefore, the project VRC contains finished software components for other projects defined for the same application.
- Activity VRC. Contains checked in and checked out software components for an activity. The activity VRC is generated automatically when you open the activity for the first time. The components in this VRC are only accessible for the software engineer to which the activity is assigned. However, other software engineers can access these components if they put this activity in the activity context of their own activity. See "Activity context" on page 50.

VRC numbering

The numbers in the customer extension of the activity VRCs are generated automatically. The numbering starts with 0001 and increases for each new activity VRC. The maximum number is 4999.

Example

LN Studio generates these VRCs:

- First activity: activity VRC B61_a3_0001.
- Second activity: activity VRC B61_a3_0002.
- Third activity: activity VRC B61_a3_0003.

Note: When you end an activity, the corresponding components are moved to the project VRC. The empty activity VRC is released, so it can be reused.

CM Actions

Check out

You must check out a component before it can be modified. LN Studio checks whether the component is already present in your activity VRC (for example because of an earlier check out and check in action).

- If the component is not yet present in your activity VRC, LN Studio locks the component in the *Project VRC*, so that other software engineers cannot check out the component, and copies the component to your activity VRC where it can be modified. The component in the project VRC stays locked until you end your activity.
- If the component is already present in the activity VRC, the component gets the "checked out" status, so that it can be modified. Note: The component in the project VRC stays locked until you end your activity.

To check out a software component in LN Studio:

1. Right-click the component in the Activity Explorer view.
2. Select **Team > Checkout**.

Check in

You can check in a component when you finish the changes to the component.

LN Studio changes the status of the component in the activity VRC to "checked in". To modify the component, perform a new check out.

Note: The checked in component stays in the activity VRC. The component is not copied to the project VRC. The component in the project VRC stays locked. Therefore, it is not accessible for all software engineers. To unlock the component in the project VRC and to make it accessible for other software engineers, end your activity, or cancel the activity or the component.

To check in a software component in LN Studio:

1. Right-click the component in the Activity Explorer view.
2. Select **Team > Checkin**.

Note: If Verify Software Components (VSC) is active and blocking VSC warnings are present, *Check in* is not possible.

Undo checkout

You can undo the check out of a component. LN Studio cancels all changes made since the last check in of the component. The component stays in the activity VRC.

Note: The component in the project VRC stays locked. Therefore it is not accessible for other software engineers. To unlock the component in the project VRC and make it accessible for all software engineers, end your activity.

To undo the check out of a software component in LN Studio:

1. Right-click the component in the Activity Explorer view.
2. Select **Team > Uncheckout**.

End activity

When you finish all changes in an activity, you can end the activity so the changed components are released to the project VRC.

When you end an activity, LN Studio performs the following actions:

- Moves all software components in the activity VRC to the project VRC. Therefore, components become available to other software engineers.
- Unlocks the components in the project VRC .
- Releases the activity VRC, so it can be reused.
- Disconnects the corresponding run configurations from the activity. You can link these configurations to another activity. See "Linking run configurations to an activity" on page 54.

Note: You can only end an activity if all components in that activity are checked in.

To end an activity in LN Studio:

1. Right-click the activity in the Activity Explorer view.
2. Select **Team > End activity**.

Note: If not all components are checked in and the **Partial End Allowed** check box in the software project properties is selected, you can partially end the activity. To do this, in the Activity Overview, End Activity dialog box, select a subset of the components and click **OK**. See the dialog box's online help.

Cancel Activity

To undo all changes in an activity, cancel the activity.

LN Studio performs these actions:

- Removes all components from the activity.
- Unlocks the components in the project VRC.
- Releases the activity VRC, so it can be reused.

To cancel an activity in LN Studio:

1. Right-click the activity in the Activity Explorer view.
2. Select **Team > Cancel Activity**.

Cancel a range of activities

To undo all changes in a range of activities, run the Global Cancel Activity (ttadv1223m000) session on the LN development server.

Note: This session does not update the status of the canceled activities in the Project Server. See the session help.

Activity context

The components in an *Activity VRC* are only accessible for the software engineer to which the activity is assigned.

Sometimes, it is necessary to use components developed by another software engineer.

For example:

- In your UI script, you want to use a new DLL developed by another software engineer.
- You want to open components from another software engineer through the **Open Declaration** command in the Script Editor.

To access software components developed by another software engineer, put the activity that contains the desired components in the activity context of your own activity.

To add an activity to the activity context of your own activity:

1. Right-click your activity in the Activity Explorer, and select **Properties**.
2. In the left pane in the properties dialog box, select **LN Studio Activity Information**.
3. Click the **Context** tab.
4. Click **Add** and select the project to which the desired activity belongs.
5. Click **Add Activity** and select the desired activity from the list.
6. Click **OK**.

Note: When you add multiple activities, use the **Up** and **Down** buttons to change the order in which the activity VRCs will be searched at runtime. The activity at the top of the list is searched immediately after your own activity. Subsequently, the second activity in the list is searched, and so on.

To delete a project or an activity from the activity context, right-click the project or activity and select **Delete**.

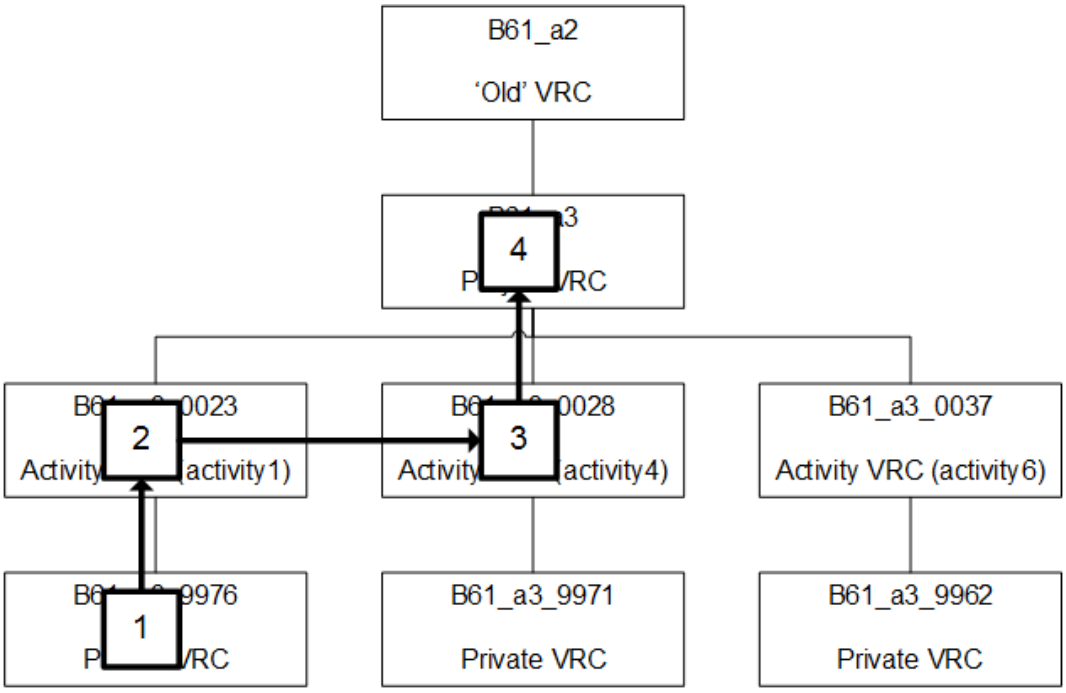
Example- search order

Activity 1 has activity 4 in its activity context.

The search order for activity 1 is as follows:

1. Activity 1 private VRC (only if SCM is active)
2. Activity 1 activity VRC
3. Activity 4 activity VRC
4. Project VRC
5. Lower VRCs

This diagram shows an example:



The context of your activity can change. The context changes, for example, in these situations:

- You add another activity to the context of your activity.
- A new component is added in an activity that is in the context of your activity.
- An existing component is modified and compiled in an activity that is in the context of your activity.

If your activity context changes, complete the following steps to incorporate the changes in your activity:

1 Select **Project > Clean**. The Clean dialog box starts.
2 Use this dialog box to invoke a full build for your activity. All previous build results are discarded.

   Note: Ensure the **Build Automatically** option in the **Project** menu is selected.

3 Recover your activity.
   See "Recovering an activity" on page 52.

Cleaning an activity

An activity can contain a lot of components that were retrieved from the server but never changed, such as when you have frequently used the **Open Declaration** command.

Clean an activity to remove these unchanged components.

To clean an activity:

1 In the Activity Explorer, select the activity you want to clean.
2 Click **Clean activity** in the Activity Explorer 's toolbar. The Clean activity dialog starts.

3 Optionally, select **Clean all activities** in the dialog to clean all activities displayed in the Activity Explorer.

4 Click **OK** to start the cleaning process. The process removes all components that are not checked out, created, or modified from the selected activities. All other components are ignored.

For details, see "Clean activity" on page 82.

Recovering an activity

You must recover an activity if, for some reason, the corresponding local workspace is not in sync with the LN server.

The recovery process restores the components in the activity from the Configuration Management System on the server to your local workspace. Optionally, unchanged components are removed from the workspace.

There are various situations in which you must recover an activity, such as:

- After a PC crash. On your new PC, you can start LN Studio, open your activity, and restore your local workspace from the server.
- After new LN Studio software is installed on your PC and on the server. The structure of the old files in your workspace is possibly not compatible with the new software. Recover your activity to retrieve new files from the server.
- After an intermediate code freeze. Your local workspace is not valid anymore.

  To solve this, you must recover your activity.

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

- After a final code freeze. Your activities have moved to a different software project. The directory names in your local workspace are not valid anymore.

  To solve this:

  1 Delete your activity from the local workspace. In the Activity Explorer, right-click the activity and, on the shortcut menu, select **Delete**.

  2 Open the activity again. In the Activity Explorer, click **Open an Infor LN Studio Activity**. The Open an Activity - Select Project dialog appears. Select the new software project from the list and click **Next**. In the second page of the dialog, select your activity from the list and click **Finish**.

  3 Recover your activity.

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

- After other developers checked in components you use in your components. Recover your activity to get the latest versions of these components.

  For example, you call another developer's DLL in one of your UI scripts. Recover your activity after the other developer checked in a new version of the DLL.

  Note: To use components of other developers, put the corresponding activities in the context of your activity.

See "Activity context" on page 50.

- If the activity is shared with other developers. Recover the activity to display the components created by the other developers.
- You have unchanged components in your activity. After a while, the server may contain new versions of these components, changed by other developers. Recover your activity to get the latest versions of these components.

Note: Recover you activity regularly to make sure the components in your workspace are up-to-date.

To recover an activity:

1. In the Activity Explorer, select the activity you want to recover.
2. Click **Recover activity** in the Activity Explorer 's toolbar. The Recover activity dialog starts.
3. Specify the settings for the recovery process. See "Recover activity" on page 84.
4. Click **OK**. LN Studio displays the recovery actions that will be executed. See "Recover activity" on page 85.
5. Click **OK** to start the recovery process. The recovery results are displayed.
6. Click **OK** to close the dialog.

Reassigning an activity

To reassign an activity to another user:

1. In the Activity Explorer, right-click the activity and, on the shortcut menu, select **Team**. Then, click **Reassign Activity**. The Reassign Activity dialog starts.
2. In the **Reassign to** field, select the user, to which you want to assign your activity, from the list.
3. Click **OK** to reassign the activity and close the dialog.

Note: After you have assigned an activity to another user:

- The other user has full permissions for the activity.
- You have only limited permissions for the activity. You can finish the components you were working at, but you cannot check out other components, cancel the activity, or end the activity.
- You can reassign the activity to your self again.

  If the other user has reassigned the activity to you, your permissions are still limited. To get full permissions, you must refresh your activity properties. To do this:

  1. In the Activity Explorer, right-click the activity and, on the shortcut menu, select **Properties**. A properties dialog starts.
  2. In the left pane, click **Activity Information**. The Activity Information dialog starts.
  3. Click **Refresh** to update the permissions.
  4. Click **OK** to close the dialog.

Linking run configurations to an activity

This section describes the procedure to link one or more run configurations to an activity.

Use this procedure, for example, after you end an activity. When an activity is ended, the corresponding run configurations are no longer linked to an activity. You can link these activities to another activity.

To link run configurations to an activity:

1. In the Activity Explorer, right-click the desired activity and select **Link Run Configurations...**.
   The Link Run Configurations dialog is displayed.
2. Select the desired run configuration(s) and click **Link**.

Development procedure

6

Developing software components

To develop a software component:

1 Open an activity

To open an *activity* in the LN Studio:

a Open the Application perspective: Select **Windows > Open Perspective > Other**. Select **Application** from the list and click **OK**.

b In the Activity Explorer, right-click and select **New > Infor LN Studio Application Project (Open Activity)** . The Open an Activity dialog is displayed.

c Select your *software project* from the list and click **Next**. The Select Activity page is displayed.

Note: The first time you open an activity for the selected project, you are prompted to configure the project's dynamic *Connection Points*. See "Defining dynamic connection points" on page 34.

d Select your activity from the list and click **Finish**.

The activity is now displayed in the Activity Explorer. A tree structure of the packages and the corresponding software components on the LN server is displayed in the Component Explorer. You can use this tree structure to link software components to your activity.

2 Create a new component or link an existing component to your personal activity

Once you have opened a personal activity in the LN Studio, you can create a new software component, or link an existing component to your activity.

To create a new component, in the Activity Explorer, right-click and select **New > Infor LN Component** . The Create a New Infor LN Software Component wizard starts. Use this wizard to define the basic properties for the new component. The created component appears in the Activity Explorer of the Application perspective. See "Create a New Infor LN Software Component" on page 98.

To link an existing component to your activity, complete one of these steps:

- Expand the relevant folders in the tree structure in the Component Explorer and select one or more software components. Then double-click the selected components, or right-click and select **Get**. The components are now displayed in the Activity Explorer.
- In the Activity Explorer, select an activity and, on the LN Studio toolbar, click **Select a Software Component**. The Select Component(s) dialog starts. Find the desired components and click **OK** to add the components to your activity. See "Select Component(s)" on page 94.

Note When you add a component to an activity, the corresponding editor can start automatically for the component. This depends on your workbench preference settings. See the description of the "Preferences - Workbench" on page 121 dialog.

Note: You can also add a component to an activity, and open it in the corresponding editor, while editing another component. For example, you can do one of the following:

- Use the **Open Declaration** command in the Script Editor to open the component you selected in the source code. See "Open Declaration" on page 285.
- Click a hyperlink in a multipage editor to open the corresponding component. For example, click the Label Code hyperlink in the Session Editor to open the label that contains the session description.

3 Check out the component

The *check out* process locks the software component for other developers. During the check out phase, the component can be updated and tested.

To check out a component:

a Right-click the component in the Activity Explorer.

b Select **Team > Checkout**.

Alternatively, in the Application perspective's toolbar, click **Check Out a Component (Alt+C, Alt+O)**.

4 Edit the component

Double-click the component in the Activity Explorer to start the corresponding editor.

For example:

- Double-click a session to edit the session in the Session Editor.
- Double-click a table to edit the table in the Table Editor.

For details on the LN Studio editors, refer to "Multipage Editors" on page 145 and "Script Editor" on page 263.

Perform the required changes and, when finished, save the changes and close the editor.

Note: If you try to change a component, which is not yet checked out, you are prompted to check out the component.

5 Build the component

By default, the component is built (compiled) automatically every time it is saved. Error messages and warnings generated by the compilation process, if any, are displayed in the Problems view.

To disable/enable the automatic compilation, turn off/turn on the **Build Automatically** command in the **Project** menu.

If automatic compilation is disabled, you must start the compilation process in the following way:

a Select the component in the Activity Explorer.

b Select **Project > Build All**.

6 Solve error messages

Solve all errors that are detected by the compilation process.

Note: Double-click an error message in the Problems view, to open the component involved. If you double-click a source code error, the source is opened at the corresponding line.

See "Identifying problems in your code" on page 61.

7 Run/debug the component

To test the changes to a component, you can launch a session, in which the component is used, from the workbench. For example:

- To test the changes to a report, launch the session to which the report belongs.
- To test the changes to a table, start a session that operates on the table. For example, to test changes in the Employees - General (tccom001) table, launch the Employees - General (tccom0101m000) session.

You can launch sessions in either run or debug mode:

- In run mode, the session executes, but you cannot suspend or examine the execution. Refer to "Running LN sessions" on page 62 for details.
- In debug mode, you can suspend and resume the execution, inspect variables, and evaluate expressions.

  See "Debugging LN sessions" on page 64.

Note: You can only launch sessions that belong to the packages displayed in the Component Explorer

8 Check in the component

The *check in* process releases the checked out software component and also stores a historical version.

To check in a component:

a Right-click the component in the Activity Explorer.

b Select **Team > Checkin**.

Alternatively, in the Application perspective's toolbar, click **Check In a Component (Alt+C, Alt+I)**.

9 End Activity

To end an activity:

a Right-click the activity in the Activity Explorer.

b On the shortcut menu, select **Team**, and subsequently click **End Activity**. The "Activity Overview, End Activity" on page 82 dialog starts.

c To end the entire activity, select all components and click **OK**. LN Studio moves the components from the *Activity VRC* to the *Project VRC*, so that the components become available to other software engineers. The selected components are compiled automatically. The activity disappears from the Activity Explorer.

Note:

- The automatic compilation fails, if a component in your activity calls an uncompiled component in another activity. The activity is closed anyway. When the problem that caused the compilation error is solved, you can recompile the closed activity in the Software Project Explorer. See the following 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.
    - To recompile the closed activity A, the owner of activity A goes to the Software Project Explorer and runs the **Compile closed activity** command.

- If the **Partial End Allowed** check box in the software project properties is selected, you can partially end the activity. To do this, in the Activity Overview, End Activity dialog box, select a subset of the components and click **OK**. See the dialog box's online help.
- If you run and debug sessions in Web UI, the corresponding form definitions are stored in the form cache on the Web UI server. To recover disk space on the Web UI server, it is recommended to remove these old forms after the activity is closed. The Web UI administrator can remove these forms via the Remove Obsolete Forms page in the Web UI Administration Console. For details, see the *Infor Enterprise Server Web UI - Installation and Configuration Guide*.
- You can only end an activity, if the previous activity is delivered. Otherwise the End Activity action is blocked.

10 Deliver the software components

The components in the activity are now ready for delivery.

If the PMC integration is enabled, you are prompted to deliver the activity and create a PMC solution. Complete one of these steps:

- To generate and deliver a solution in one go, click **Yes** in the question window.
- To manually create and deliver a solution, click **No**.

If the PMC integration is not enabled, manually create and release a solution through the PMC sessions on the LN server.

For details on the delivery process, see "Delivering software components" on page 336.

Verifying software components

Introduction

LN Studio enables you to perform quality control on the software components that you develop: you can use the **Verify Component** command to perform various checks, based on the LN design principles. When you verify software components, a list of *warnings* is generated. Per warning you can decide to accept the warning, or to solve the problem.

Among other things, the **Verify Component** command performs these actions:

- Checks whether software development is done according to the LN coding and programming standards.
- Searches for inconsistency in the Enterprise Server dictionary.
- Searches for suspicious constructions.
- Searches for constructions the compiler will let through. For example:
    - (Dynamic) function implementation.
    - A message code is used in a script or library, but the message is expired or not present.

Note: The **Verify Component** command in LN Studio is based on the *Verify Software Components (VSC)* functionality on the LN server. See "Verify Software Components (VSC)" in the Web Help on the LN server.

Verifying software components

To verify one or more software components:

1 Switch to the Application perspective.

2 Select the desired components in the Activity Explorer. Right-click and select **Verify Component**. The verification process starts. The resulting warnings are displayed in the Verify Components view.

3 Handle the generated warnings. See the following section.

Note: If the **Build Automatically** option in the **Project** menu is selected, the verification can also start automatically each time you save a component. To enable automatic verification, you must select the **Automatically check components after a change** check box in the Preferences - Workbench page.

Handling warnings

To handle warnings:

1 View the list of warnings and the warning details

View the generated warnings in the Verify Components view. To view detailed information for a warning, right-click the warning, and select **View Details**. The Verify Warning Details dialog starts.

Each warning has various attributes, such as:

- *verification code*.
- Warning ID and message text.
- Component name
- Component type.
- Component subtype
- Location where the error occurs in the component. For example, a line number in a script.
- Priority: high, normal, suspicious, or low.

Note: You can click **Help** in the Verify Warning Details dialog to view detailed information and examples, which will help you to solve the warning.

2 Display the "blocking" warnings

Depending on the verification filter settings, the component verification generates the following types of warnings:

- "Blocking" warnings. These warnings block the check in of software components. You must either solve or accept these warnings immediately.
- Other warnings that you do not have to handle immediately.

By default, the Verify Components view displays all generated warnings. You can apply a filter, so that the view displays "blocking" warnings only.

To display "blocking" warnings only:

a Click **Verify Warning Filter** in the view's toolbar. The Verify Components View Filter dialog appears.

b Select the **Enabled** check box and the **Only blocking** check box. Clear all remaining check boxes.

c Click **OK** to apply the filter.

3 Accept or solve the warnings

For each "blocking" warning, you must decide whether you want to accept it or solve it.

For example, *VSC* has generated the following warning: "Possible missing break in case statement".

- If this break is not missing, but left out on purpose, you can accept the warning.

  To accept a warning:

  a Right-click the warning in the Verify Components view, and on the shortcut menu click **View Details**. The Verify Warning Details dialog starts.

  b Select the **Accept Warning** check box and enter the reason for acceptance in the **Accept Text** field.

  c Click **OK**.

  Note: As a result of the verification filter settings that are specified in the Verification Filter Defaults (tlvsc2110m000) session:

  - VSC can block the acceptance of a warning. You cannot accept such a warning, but you must solve it.
  - Some warnings can block the check in procedure for the component. You must solve these warnings, before you can check in the components.

- If this break was left out accidentally, you must solve the warning.

  To solve a warning:

  a Right-click the warning in the Verify Components view, and on the shortcut menu click **View Details**. The Verify Warning Details dialog starts.

  b Click **Help** to view detailed information and examples that can help you to solve the warning. Usually this information is displayed:

  - A description, a motivation and a possible solution for the warning.
  - A wrong example.
  - A correct example.

  c Close the Verify Warning Details dialog and return to the Verify Components view.

  d Double-click the warning. The appropriate editor for the software component starts automatically at the location where the error occurs. Edit the software component and fix the problem.

  For example: VSC has generated the following warning "Possible missing break in case statement". When you double-click the warning, the script editor starts automatically at the relevant line number in the script or library, where you can add the missing statement.

e Verify the component again. If no further errors are found, the warning automatically disappears from the warnings list.

Identifying problems in your code

This topic describes the indicators used to identify problems in your code.

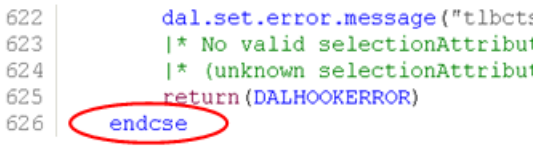
Sample exercise

To get familiar with the problem indicators, try the following simple exercise:

1 Add a syntax error

To add a syntax error, open a source that contains a case statement and remove a letter from the endcase statement.

This figure shows an example:



2 Save changed code and view problem indicators

On the Workbench toolbar, click **Save** to save the changed code. Do not close the editor. The source is compiled automatically and the problem is indicated in various ways.

Note: Automatic compilation only works if the **Build Automatically** option in the **Project** menu is turned on.

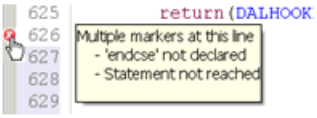
The problem is indicated in, for example, these ways:

- In the Activity Explorer view, problem decorators appear on the affected component and its parent elements.
- An error marker appears in the marker bar, to the left of the line that contains the syntax error.
- Another error marker appears in the vertical ruler. This marker represents the syntax error and is shown relative to the error's position in the document. The marker does not move as you scroll the document. Click the marker to navigate quickly to the corresponding location in the source code.
- The problem is displayed in the Problems view.

3 View problem description via hovering

To view the description of the problem, hover with the mouse over the error markers in the marker bar and vertical ruler.

This figure shows an example:



For more information on hovering, see "Text Hovering" on page 289.

4 Close the editor

Click the **Close** button on the editor's tab to close the editor.

5 Use **Go To** command

In the Problems view, complete one of the following steps:

- Right-click a problem and, on the shortcut menu, select **Go To**.
- Double-click a problem.

The editor starts and opens the affected source at the location of the problem.

6 Correct the problem

To solve the problem, in the editor, add the missing letter in the endcase statement.

On the Workbench toolbar, click **Save** to save the changed code. The source is compiled automatically and the problem indicators disappear.

Note: Automatic compilation only works if the **Build Automatically** option in the **Project** menu is turned on.

Running and debugging software components

Running LN sessions

You can run LN sessions from the *Eclipse* workbench.

For each session that you want to run, you must generate, or manually create, a run configuration. In this configuration you can specify execution parameters.

To run a session for the first time, use one of these procedures:

- "Automatically generate a run configuration and then run the session" on page 63
- "Manually create a run configuration and then run the session" on page 63

When you run a session, the session executes, but you cannot suspend or examine the execution.

Note: If you want to examine the execution, you must run the session in debug mode. In debug mode, you can suspend and resume the execution, inspect variables, and evaluate expressions.

See "Debugging LN sessions" on page 64.

Automatically generate a run configuration and then run the session

Complete these steps:

1. In the Activity Explorer, right-click the session.
2. Select **Run As > Infor LN Session** . The session starts. A run configuration is generated automatically.

You can modify a generated run configuration in various ways. For example:

- In the Activity Explorer, right-click the session and select **Run As > Run Configurations**.
- Select **Run > Run Configurations**.
- Click the **Run Configurations** command on the **Run** button pull-down menu.

The Run - Create, manage, and run configurations dialog box is displayed.

For more information, see "Run - Create, manage, and run configurations" on page 129.

Manually create a run configuration and then run the session

1. Add a new configuration
   a. Select **Run > Run Configurations**, or select **Run Configurations** on the **Run** button pull-down menu. The Configurations dialog box is displayed.
   b. In the left pane, right-click **Infor LN Session** and select **New**. A new configuration is added in the left pane.
   c. In the **Sessions** tab, select an activity from the list.
   d. Select the session you want to run. The name of the configuration is automatically filled with the description of the selected session.

      To select a session, perform one of these actions:

      - Select a session linked to your activity from the list.
      - Click **Browse** to select a session that is not linked to your activity. The Browse session dialog box starts. In this dialog box you can select a session on the LN server.
      - Specify a session code.

      Note: You can only select sessions that belong to the packages displayed in the Component Explorer.

   e. Specify the launch mode for the session.

      See "Configurations" on page 129.

   f. Select the company in which you want to start the session.
   g. Optionally: Specify additional execution parameters, such as profiling parameters and environment variables, in the other tabs. See "Configurations" on page 129.
   h. Click **Apply** to save the settings.

Note: You can also use run configurations to start sessions in debug mode. See "Debugging LN sessions" on page 64.

2 Run the session

Click **Run** in the Run dialog box. The session is started in run mode.

Note: To start sessions through Web UI you must start Web UI first.

Restarting a session in run mode

The workbench keeps a history of all configurations launched.

To restart the most recent launch, click the **Run** button in the toolbar.

You can restart any previous launch in various ways. For example:

- Select a previous launch on the **Run** button pull-down menu.
- Select **Run > Run History** and select a previous launch from the sub-menu.
- Select **Run > Run Configurations**, or select **Run Configurations** on the **Run** button pull-down menu. The Run - Create, manage, and run configurations dialog box is displayed. Select the desired configuration and click **Run**.
- In the Activity Explorer, right-click the session and select **Run As > Infor LN Session**.

Select Activity command in Web UI

The **Options** menu in Web UI contains the **Select Activity** command. You can use this command to test software components that are checked out to an LN Studio activity.

For details, see the Web UI online help.

Debugging LN sessions

Debugging LN sessions through BW or Web UI

From the *Eclipse* workbench you can run LN sessions in *debug* mode. In debug mode, you can suspend and resume the execution, inspect variables, and evaluate expressions.

For each session that you want to run in debug mode, you must generate, or manually create, a run configuration. In this configuration you can specify execution parameters.

To run a session in debug mode for the first time, use one of these procedures:

- "Manually create a run configuration and then run the session in debug mode" on page 65
- "Automatically generate a run configuration and then run the session in debug mode" on page 66

Note: To debug sessions, a debug *Connection Point* is required. For more information, see "Defining connectivity settings" on page 32.

Manually create a run configuration and then run the session in debug mode

To debug a session for the first time:

1 Set breakpoints

Edit the involved script or library, and set breakpoints before you start the debug process. For example, set breakpoints in the session's UI script, or in a library called by the session.

You can set these types of breakpoints:

- Line breakpoints
- Method breakpoints
- Watchpoints

For details, see "Using breakpoints" on page 70.

Note:

- Before you can set breakpoints in a UI script, you must add the corresponding session to your activity.
- Before you can set breakpoints in a library, you must add the library to your activity.

2 Add a new configuration

a On the **Run** menu, select **Debug Configurations**, or click the **Debug Configurations** command on the **Debug** button pull-down menu. The Debug - Create, manage, and run configurations dialog is displayed.

b In the left pane, right-click **Infor LN Session** and select **New**. A new configuration is added in the left pane.

c In the **Sessions** tab, select an activity from the list.

d Select the session that calls the script or library you want to debug. The name of the configuration is automatically filled with the name of the selected session.

To select a session, perform one of these actions:

- Select a session linked to your activity from the list.
- Click **Browse** to select a session that is not linked to your activity. The Browse session dialog starts. In this dialog you can select a session on the LN server.
- Specify a session code.

Note: You can only select sessions that belong to the packages displayed in the Component Explorer

e Specify the launch mode for the session.

See "Configurations" on page 129.

f Select the company in which you want to start the session.

g Optionally: Specify additional execution parameters, such as profiling parameters and environment variables, in the other tabs. For details, refer to the dialog help.

h Click **Apply** to save the settings.

Note: Debug configurations can also be used to start sessions in run mode.

See "Running LN sessions" on page 62.

3 Debug the session

Click **Debug** in the Debug dialog. The session is started in debug mode.

Use the views and commands in the Debug perspective to examine the execution and debug the session:

- Use the Debug view and the **Run** menu to control the execution of the session. For example, you can terminate or suspend executing sessions, resume the execution, and step through the execution. For details, see:
  - "Suspending sessions" on page 68
  - "Resuming the execution of suspended sessions" on page 68
  - "Stepping through the execution" on page 68
- Use the Variables view to inspect the values of the variables of a *stack frame*.
  See "Inspecting values" on page 69.
- Use the Expressions view to evaluate expressions in the context of a stack frame.
  See "Evaluating expressions" on page 69.
- Use the Script Editor and the Outline view to toggle breakpoints.
- Use the Breakpoints view to enable, disable, or remove breakpoints.
  See "Using breakpoints" on page 70.

Note:

- The Variables view displays all variables that are in scope. If a session contains a lot of variables, the performance can be affected. To boost the performance, display local variables only, or close the Variables view and use the Expressions view.
- Try to keep the list of expressions in the Expressions view limited. This is, because all expressions are evaluated each time a session suspends, which can affect the performance adversely.
- The session in the previous example is started through Web UI. To start sessions through Web UI you must start Web UI first.

Automatically generate a run configuration and then run the session in debug mode

Complete these steps:

1. Set breakpoints.
   See "Manually create a run configuration and then run the session in debug mode" on page 65.
2. In the Activity Explorer, right-click the session and select **Debug As > Infor LN Session** . A run configuration is generated automatically. The session starts in debug mode.
   See "Manually create a run configuration and then run the session in debug mode" on page 65.

You can modify a generated run configuration in various ways. For example:

- In the Activity Explorer, right-click the session and select **Debug As**. Then select **Debug Configurations**.
- On the **Run** menu, select **Debug Configurations**.
- Click the **Debug Configurations** command on the **Debug** button pull-down menu.

The Run - Create, manage, and run configurations dialog is displayed.

For more information, see "Configurations" on page 129.

Restarting a session in debug mode

The workbench keeps a history of all configurations launched.

To restart the most recent launch, click the **Debug** button in the toolbar.

You can restart any previous launch in various ways. For example:

- Select a previous launch on the **Debug** button pull-down menu.
- On the **Run** menu, select **Debug History** and select a previous launch from the sub-menu.
- On the **Run** menu, select **Debug Configurations**, or click the **Debug Configurations** command on the **Debug** button pull-down menu. The Debug - Create, manage, and run configurations dialog is displayed. Select the desired configuration and click **Debug**.
- In the Activity Explorer, right-click the session and select **Debug As > Infor LN Session** .

Debugging through LN UI

LN Studio can attach to a running LN UI and can debug sessions started in LN UI. Unlike debugging through BW or Web UI, the sessions are started from LN UI and not from LN Studio.

Note: To debug sessions, a debug Connection Point is required.

See "Defining connectivity settings" on page 32.

To debug a session:

1 Connect to LN UI

In the Activity Explorer, right-click the activity and select **Debug As > LN UI Connection** . A run configuration is generated automatically. LN Studio is now connected to the LN server.

2 Connect to LN Studio from LN UI

Start LN UI and select **Options > Debug and Profile 4GL** . The Debug and Profile 4GL (ttadv1123m000) session starts. Select your activity and select the **Debug Mode** check box. Click **OK** to save the settings.

3 Set breakpoints

See "Manually create a run configuration and then run the session in debug mode" on page 65.

4 Debug a session

Start a session in LN UI. The session is started in debug mode.

See "Manually create a run configuration and then run the session in debug mode" on page 65.

Step 1 and 2 are only required at the beginning, before you start the first session. Subsequent sessions automatically start in debug mode using the established connection.

Suspending sessions

When a session is executed in debug mode, you can suspend a running process, so that you can inspect it more closely.

To suspend an executing session:

1. Select the session in the Debug view.
2. Click **Suspend** in the Debug view toolbar or use the shortcut menu. The session suspends its execution. The current call stack for the session is displayed, and the current line of execution is highlighted in the editor in the Debug perspective.

When a session suspends, the top *stack frame* of the session is automatically selected. The Variables view shows the stack frame's variables and their values. You can further examine complex variables by expanding them to show the values of their members.

When a session is suspended and the cursor is hovered over a variable in the editor, the value of that variable is displayed.

Resuming the execution of suspended sessions

To resume the execution of a suspended session:

1. Select the session or its stack frame in the Debug view.
2. Click **Resume** in the Debug view toolbar or use the shortcut menu or press F8. The session resumes its execution, and stack frames are no longer displayed for the session. The Variables view is cleared.

Stepping through the execution

When a session is suspended, you can use the following commands to step through the execution of the program line-by-line:

Command	Usage
Step Into	This command steps into the current statement: **1** Select a *thread* or a *stack frame* in the Debug view. The current line of execution in the selected frame is highlighted in the editor. **2** Run the **Step Into** command, or press F5, to invoke the next instruction to be executed. The execution suspends at the next executable line in the method that is invoked. **Note:** Ensure that the **Reuse editor when displaying source code** option in the Run/Debug preferences dialog box is disabled. If this option is enabled, and you use the **Step Into** command while the script is not in the workspace, an error page is displayed in some situations. This command is equivalent to the **s** action in the classic 4GL Debugger.

Command	Usage
Step Over	This command steps over the current statement: **1** Select a thread or a stack frame in the Debug view. The current line of execution in the selected frame is highlighted in the editor. **2** Run the **Step Over** command, or press F6, to execute the next instruction. The execution suspends on the next executable line. This command is equivalent to the **S** action in the classic 4GL Debugger.
Step Return	Run the **Step Return** command, or press F7, to continue the execution until the end of the current function. This command is equivalent to the **return** action in the classic 4GL Debugger.
Run to Line	Run the **Run to Line** command, or press CTRL+R, to move the execution pointer, without executing code, to the line that you selected in the Editor area. This command is equivalent to the **Go to Line** (**g**) action in the classic 4GL Debugger.

Note:

- You can start the step commands from the toolbar and the shortcut menu in the Debug view, and from the **Run** menu in the Eclipse Workbench.
- You can start the **Run to Line** command only from the **Run** menu in the Eclipse Workbench.
- If a breakpoint is encountered while performing a step operation, the execution will suspend at the breakpoint and the step operation is ended.

Inspecting values

When you select a *stack frame*, the visible variables in that stack frame are displayed in the Variables view.

The Variables view shows the value of primitive types. To examine complex variables, such as arrays or strings, expand them to show their members.

XML nodes are displayed in an XML tree, which you can collapse and expand.

Evaluating expressions

LN Studio enables you to create watch expressions.

When the Virtual Machine suspends a session (due to hitting a breakpoint or stepping through code), you can evaluate these expressions in the context of a *stack frame*.

Creating watch expressions

To create a watch expression that is executed each time you step through the execution, complete one of these steps:

- Right-click in the Expressions view. On the shortcut menu, select **Add Watch Expression** and type the name of the expression you want to evaluate.
- In the LN Studio Script Editor, select the expression you want to evaluate. Right-click and, on the shortcut menu, select **Watch**.
- When you are debugging a session, go to the Variables view. Right-click a variable you want to evaluate and, on the shortcut menu, click **Create Watch Expression**.
- Specify a condition or modification watchpoint; a watch expression is added automatically.

The watch expressions you enter are stored automatically. So, the next time you debug a session, the same expressions are available in the Expressions view.

Executing Assignment Expressions

To define an expression that must be executed only once, right-click in the script editor or right-click in the Expressions view and, on the shortcut menu, select **Execute Assignment Expression**. The Execute Assignment Expression dialog starts. For details, refer to the dialog's online help.

Note: If you select a variable name in the script editor before you start the **Execute Assignment Expression** command, this expression is displayed in the dialog: `<variable name>:=`

For example, you select the `task.child.process` variable in the script. Then, on the shortcut menu, you click **Execute Assignment Expression**. The Execute Assignment Expression dialog starts and contains this expression: `task.child.process:=`

Evaluating expressions

To evaluate expressions:

1. Switch to the Debug perspective and open the Expressions view.
2. Start a session in debug mode.
3. In the Debug view, select the stack frame in which you want to perform an evaluation. The expressions in the Expressions view are evaluated automatically.
   - For the detail pane of the Expressions view, the evaluation context will be a selected variable.
   - If you do not select a variable, the selected stack frame will be the context.

Note: When you edit a watch expression in the Expressions view, you can use the **Reevaluate Watch Expression** command in the view's shortcut menu to evaluate the watch expression again.

Using breakpoints

A breakpoint is a point in a program that, when reached, triggers some special behavior useful to the process of debugging.

A breakpoint causes the execution of a session to suspend at the location where the breakpoint is set.

This table shows the breakpoint types you can define in LN Studio:

Line breakpoints	You can set a line breakpoint on an executable line in the source code. During the debugging, the session execution suspends before that line of code is executed. The debugger selects the session that has suspended and displays the *stack frames* on that session's stack. The line where the breakpoint was set is highlighted in the editor in the Debug perspective.
Method breakpoints	You can set method breakpoints on the functions displayed in the Outline view. The session execution suspends when the execution reaches a function for which a breakpoint was set.
Watchpoints	You can use watchpoints to monitor variables. There are two types of watchpoints: • Modification watchpoint. This is the default type. For this type of watchpoint, session execution suspends each time the variable being monitored is modified. • Condition watchpoint. For this type of watchpoint, session execution suspends when the variable gets a certain value. You can change the watchpoint type through a Properties dialog in the Breakpoints view. You cannot set a watchpoint on array elements.

You can set line breakpoints and watchpoints in the LN Studio script editor. You can set method breakpoints in the Outline view.

Breakpoints are displayed in the vertical script editor ruler and in the Breakpoints view.

You can enable and disable breakpoints via their context menus in the Breakpoints view.:

- When a breakpoint is enabled, it causes a session to suspend whenever the breakpoint is reached. Enabled breakpoints are indicated with a blue circle. Resolved breakpoints are shown with a check mark overlay.
- When a breakpoint is disabled, it does not cause sessions to suspend. Disabled breakpoints are indicated with a white circle.

Note: You cannot define breakpoints in report scripts. For a report, breakpoints must be defined in the report's debug script. To open the debug script, in the **Overview** tab in the Report Editor, click **View Debug Script**. The debug script is displayed in a separate tab in the editor area.

Adding line breakpoints

To set a line breakpoint:

1. In the script editor, go to the line where you want to set the breakpoint.
2. Double-click on the marker bar directly to the left of the line.

   Alternatively, right-click on the marker bar and, on the shortcut menu, select **Toggle Breakpoint**.

A new breakpoint marker appears on the marker bar. Also, the new breakpoint appears in the Breakpoints view.

Adding method breakpoints

To set a method breakpoint, complete one of these steps:

- In the Outline view, right-click a function and, on the shortcut menu, select **Toggle Method Breakpoint**.
- In the script editor, double-click the marker bar next to a line that is part of a function header.
- In the script editor, right-click the marker bar next to a line that is part of a function header and, on the shortcut menu, select **Toggle Breakpoint**.

A new method breakpoint marker appears on the marker bar in the script editor. Also, the new breakpoint appears in the Breakpoints view.

Adding watchpoints

You can define watchpoints for local and global variables via the LN Studio script editor.

Adding a modification watchpoint for a local variable

To set a modification watchpoint for a local variable:

1. In the script editor, select the variable (or table field) you want to monitor.
2. Right-click and, on the shortcut menu, select **Modification Watchpoint**. A modification watchpoint marker appears in the editor's marker bar, and the new watchpoint is displayed in the Breakpoints view.

Adding a condition watchpoint for a local variable

To set a condition watchpoint for a local variable:

1. In the script editor, select the variable (or table field) you want to monitor.
2. Right-click and select **Modification Watchpoint**. A modification watchpoint marker appears in the editor's marker bar, and the new watchpoint is displayed in the Breakpoints view.
3. Use the Properties dialog in the Breakpoints view to change the new watchpoint into a condition watchpoint. See "Watchpoint properties" on page 113.

Note: Alternatively, complete these steps:

1. Right-click in the script editor and, on the shortcut menu, click **Condition Watchpoint**. The New Condition Watchpoint dialog is started.
2. Specify the name of the variable (or table field) you want to monitor.
3. Select **Variable Value** and indicate when the session must suspend. See "Watchpoint properties" on page 113.

Adding a watchpoint for an external or global variable

To set a condition or modification watchpoint for an external or global variable that does not occur in the source code you are editing:

1. Right-click in the script editor and click **Condition Watchpoint**. The New Condition Watchpoint dialog is started.

2 Specify the name of the variable (or table field) you want to monitor, and indicate when the session must suspend. See "Watchpoint properties" on page 113.

Note: When you define a watchpoint in the script editor, you must be sure that the selected text really is a variable or table field: the editor does not check this. In the editor you can select any piece of text, and subsequently define a watchpoint for it, which is completely useless.

When you define a condition or modification watchpoint, a watch expression is added automatically. For details on watch expressions, see "Evaluating expressions" on page 69.

Removing breakpoints

You can easily remove breakpoints when you no longer need them.

To remove a line breakpoint or a watchpoint:

1 In the script editor, go to the line where the breakpoint is defined.

2 Double-click on the breakpoint icon, or right-click the breakpoint icon and, on the shortcut menu, select **Toggle Breakpoint**.

To remove a method breakpoint, complete one of these steps:

- In the Outline view, right-click the involved function and, on the shortcut menu, select **Toggle Method Breakpoint**.
- In the script editor, go to the line where the breakpoint is defined and double-click on the breakpoint icon, or right-click the breakpoint icon and, on the shortcut menu, select **Toggle Breakpoint**.

You can also remove breakpoints in the Breakpoints view:

- Right-click the breakpoint(s) to be removed and, on the shortcut menu, select **Remove**.
- To remove all breakpoints in one go, on the shortcut menu of the Breakpoints view, click **Remove All**.

If you find yourself frequently adding and removing a breakpoint in the same place, consider disabling the breakpoint when you do not need it and enabling it when needed again. See "Breakpoints view" on page 306.

Enabling and disabling breakpoints

You can disable and enable breakpoints as needed. When a breakpoint is disabled, session execution is not suspended by the presence of the breakpoint.

To disable a breakpoint, complete one of these steps:

- Go to the Breakpoints view and clear the breakpoint's check box.
- In the script editor, right-click the breakpoint in the marker bar and, on the shortcut menu, select **Disable Breakpoint**.

The breakpoint image changes to a white circle.

To enable a breakpoint, complete one of these steps:

- Go to the Breakpoints view and select the breakpoint's check box.

- In the script editor, right-click the breakpoint in the marker bar and, on the shortcut menu, select **Enable Breakpoint**.

The breakpoint image changes back to a blue circle.

Note: To disable all breakpoints in one go, use the **Skip All Breakpoints** command in the Breakpoints view.

Running BFlow tests

You can run and debug BFlow test scripts and test cases in LN Studio. The procedure is equivalent to running and debugging LN sessions through BW.

To run a BFlow test, right-click the test in the Activity Explorer and select **Run As > BFlow Test**.

To debug a BFlow test:

1. Set the breakpoints.
2. Right-click the test in the Activity Explorer and select **Debug As > BFlow Test**.

   The execution is paused each time a breakpoint is hit. You can then inspect the variables, continue the execution step-by-step, or resume the execution.

   The execution log is displayed in the LN Studio Console.

InContext Modeling

7

Infor Ming.le™ - LN integration

Infor Ming.le contains various context applications that show data that is related to LN objects. See these examples:

- An LN Business Partner has an address; if a user opens a Business Partner record, the corresponding location is displayed in the Map context application.
- A Sales order can have multiple documents attached to it. The Business Partner, to which the Sales Order applies, can also have multiple documents attached to it. Both, the documents attached to the Sales Order and to the Business Partner, are displayed in the Related Information context application.

Context messages

The integration between Infor LN and the Infor Ming.le context applications is based on context messages. In these context messages, the characteristics of the LN objects, which have focus in the UI, are sent to the context applications. Context messages are generated from InContext Models.

InContext model

An InContext model describes which fields are required for the context messages and how these fields can be retrieved.

Two types of InContext models exist:

- InContext Reference Model
- InContext Implementation Model

InContext Reference Model (IRM)

The IRM is a model of tables and references that is generated from the existing table definitions and other data. This model contains a basic link between table(field)s and context messages. An IRM belongs to an application, a set of packages that belong to each other and have the same version.

InContext Implementation Model (IIM)

The IIM contains the link between the tables and sessions and the context messages that must be generated. This model can be generated initially from the IRM. In the IIM, hooks are available to influence the default behavior.

In LN Studio, you can modify the IIM on table level and session level:

- Table level

  A table level model contains the applicable context message types for a particular table, and the fields that must be used to construct the message. This model applies to all sessions where that particular table is the main table, unless the table level model is overruled by the session level model.

  You can modify the table level model in the Table InContext Model Editor.

- Session level

  A session level model contains the applicable context message types for a particular session.

  Session level models are used for these purposes:

  - To specify the context message types for sessions without main table.
  - To define session-specific extensions to the table level model for sessions that have a main table.

  You can modify the session level model in the Session Model Editor.

Related topics

- *Infor Enterprise Server InContext Modeling Development Guide*
- "Table InContext Model Editor" on page 257
- "Session Model Editor" on page 236
- "Dependent InContext Models view" on page 314

LN Studio Limitations

8

This section describes these topics:

- General limitations
- Script Editor Limitations
- Configuration Management (CM) limitations
- Build limitations
- Debugger limitations
- Web UI limitations

General limitations

Connections

Take care of the following when you use Worktop and LN Studio simultaneously.

When you close the Worktop, this message is displayed:

```
There are still running sessions including integrations.
These integrations may not work anymore after closing.
Close all sessions?
```

Click **No** if there are any connections, used by LN Studio, running on the LN server. If you click **Yes**, your connections will be closed. As a result, LN Studio cannot communicate with the server, and therefore will not respond anymore.

Script Editor Limitations

Recognition of syntax patterns

The script editor does not recognize syntax patterns as 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, since the editor does not recognize it as a variable.

Debugging

There are also some editor limitations that are related to debugging. See "Debugger limitations" later in this section.

Configuration Management (CM) limitations

Blocking

CM actions are currently performed in a foreground process, and will as such block any user actions until the CM actions have finished.

Dirty state decorator

The dirty state for sessions, tables and reports can not be determined accurately, because the modification date is not available in LN.

Build limitations

“Use checked out version of script with development vrc …” message

Building files checked out by yourself, or by someone else in the same SCM group, results in the following error message: “use checked out version of script with development vrc …”.

Failure to build without a clear reason

For some reason, the storage or build of a component on the development system may fail. In certain cases, no error is displayed to the user. It is necessary to check the LN Business Adapter log file to discover the reason why a build failed.

To find the location of this log file, start the LN Business Adapter Console. Expand the **Logging** node to display a list of machines. Select your PC from the list. The name and location of the log file, and other logging parameters, are displayed.

Debugger limitations

Debug View

“An error has occurred. See error log for more details” error message

Sometimes, while debugging, the message “An error has occurred. See error log for more details.” may be given by Eclipse. This is not a fatal error, so you can click **OK** and resume debugging.

Variables view

Filters not correctly initialized

The filters in the Variables view (**Show Global Variables** and **Show Local Variables**) are not correctly initialized. To correct the settings, click the stack frame twice or select a variable in the Variables view.

Expressions view

Limited to variables

Currently, it is only possible to evaluate expressions that are actually variables. It is not possible to evaluate other expressions, such as I=0.

Changing values not reflected in UI

The value of an expression is not updated, when the value of the contained variable is changed. To solve this, use the **Reevaluate Watch Expression** command on the expression.

Script Editor

Multiple instruction pointers

Eclipse does not always remove an existing instruction pointer. This may lead to multiple instruction pointers being displayed in a script editor. To solve this, close the editor. The editor will automatically be re-opened at the next step/suspend action.

Editing while debugging

LN does not support hot code replacement, so changes made in a script will only take effect the next time a script is run. Changing a script while debugging can also lead to incorrect breakpoint handling, as the line numbers do no longer match the lines of the (already) running object.

Run/Debug preferences

The **Reuse editor when displaying source code** option in the Run/Debug preferences dialog is not supported. You must disable this option when you use the **Step Into** command in the Debug view. If this option is enabled, and you use the **Step Into** command while the script is not in the workspace, an error page is displayed in some situations.

Watchpoint

You can not set a watchpoint on array elements.

Web UI Limitations

Labels

If you run a session in Web UI, any new or changed labels are only displayed if the session is compiled and linked to your activity.

Wizards and Dialogs

9

Activity

Activity Information

Use this dialog to view the properties of an LN Studio *Activity*, and to add other activities to the context of the selected activity.

Settings

This dialog shows detailed information on:

- The selected activity.
- The *Software Project* to which the activity belongs.
- The *Application* to which the software project is linked.

See the online help of these dialogs:

- "Create a new Activity" on page 340
- "Create a Software Project" on page 339
- "New Application" on page 132

Context

You can use this tab to put another *Activity* in the context of the current activity. This is useful if you want to use components developed in another activity by another *Software Engineer* or by yourself. For example: You want to use a DLL, that is currently modified by another software engineer, in a *UI script*.

See "Activity context" on page 50.

To add an activity:

1. Click **Add** and select the project to which the desired activity belongs.
2. Click **Add Activity** and select the desired activity from the list.
3. Click **OK**.

The **Can Open** check box is read-only. It indicates whether you are assigned to the activity.

Activity Overview, End Activity

This dialog is displayed when you end an activity.

The dialog shows the components in the activity that are new or modified.

If the **Partial End Allowed** check box in the software project properties is selected, you can partially end the activity. Use the **Selected** check box to select the components for which you want to end the activity.

If you selected all components, the activity is ended completely and disappears from the Activity Explorer.

Note:

- You can only completely end an activity if all components in that activity are checked in.
- The Project VRC contains all finished software components for the project. From this VRC, deliveries to customers are done when the project is completed.
- For more information, refer to "Activity based development" on page 39.

Selected
If this check box is selected, the component is moved from the *Activity VRC* to the *Project VRC*, and becomes available to other software engineers.

If this check box is cleared, the component stays available in the activity VRC.

Type
The component type, such as session or label.

Name
The name of the component.

User(s)
The user who modified the component.

Blocked by
The activity, which blocks the end activity process, and its owner.

The end activity process is blocked if the previous activity is not yet released and delivered.

You can only end an activity, if the previous activity is released and delivered. Otherwise the End Activity action is blocked. The name of the previous activity, which blocks the End Activity process, and its owner are displayed.

Clean activity

Use this dialog to remove unchanged components from one or all activities.

The cleaning process performs these actions:

- Removes all components that are not checked out, created, or modified from the selected activities.

  In the Activity Explorer, these components have a decorator in the bottom right corner of the software component image.

- Ignores all new, modified, and checked out components.

  In the Activity Explorer, these components have a or decorator in the bottom right corner of the software component image.

Clean all activities
If this option is selected, components are removed from all activities displayed in the Activity Explorer.

Clean selected activity
If this option is selected, components are removed from the activity you selected in the Activity Explorer.

Open an Activity

Use this wizard to open a new *activity*.

To open an activity:

1. In the Select Project page, select your *software project* from the list and click **Next**. The Select Activity page is displayed.
2. Select your activity from the list and click **Finish**.

Select Project

Use this page to select the project to which the activity belongs.

All fields, except **Project name**, are read-only.

Project name
The project to which the activity belongs.

You can select the project name from a drop-down list.

All other fields
For details on the other fields, see the online help of the Create a Software Project dialog.

Select Activity

Use this page to select the activity you want to open.

All fields, except **Activity name**, are read-only.

Activity name
The activity you want to open.

You can select the activity name from a drop-down list.

All other fields
For details on the other fields, refer to the online help of the Create a new Activity dialog.

Reassign Activity

Use this dialog to assign an activity to another user.

Note: After you have assigned an activity to another user:

- The other user has full permissions for the activity.
- You have only limited permissions for the activity.
- You can reassign the activity to yourself again.

See "Reassigning an activity" on page 53.

You can start this dialog from the shortcut menu in the Activity Explorer view.

Reassign to
Select the user, to which you want to assign your activity, from the list.

Recover activity

Use this dialog to recover an activity. You can start this dialog from the Activity Explorer view.

The recovery process restores files that belong to the selected activity from the Configuration Management System on the server, and stores them in your local workspace. This is very useful, for example, after a PC crash: On your new PC, you can start LN Studio, open your activity, and restore your local workspace from the server.

Click **OK** to confirm your selections. The Recover activity dialog starts. Here, you can view the recovery actions that will be taken and decide whether you want to proceed with the recovery process.

See "Recover activity" on page 85.

Modified components in the current activity

Recover existing components
If this check box is selected, LN Studio restores all your new, modified, and checked out components from the server and stores the corresponding files in your local workspace.

In the Activity Explorer, these components have a or decorator in the bottom right corner of the software component image.

Note: Selecting this check box is mandatory if you recover your activity after a code freeze.

Modified components of other user(s)

Process modified components of other user(s)
If this check box is selected, components of other users, assigned to the activity, are also processed. If this check box is cleared, these components are ignored.

Select one of the following options:

Repair	LN Studio corrects the administrative information about these components in your local workspace, when necessary. If the components in your local workspace are corrupt, new components are retrieved from the server.
Recover	LN Studio restores these components from the server and stores the corresponding files in your local workspace. For example: you open a shared activity. The components developed by other users are not displayed automatically. Use this option to display these components.
Delete	LN Studio removes these components from your local workspace, so they are not displayed in the Activity Explorer anymore.

Non modified components
This field indicates the action that will be performed for components that are not checked out, modified, or created.

In the Activity Explorer, these components have a decorator in the bottom right corner of the software component image.

Allowed values

Ignore	LN Studio ignores these components. The components are not removed from your activity and local workspace.
Recover	LN Studio restores these components from the server and stores the corresponding files in your local workspace.
Delete	LN Studio removes these components from your activity and local workspace. **Note:** The **Clean activity** command in the Activity Explorer view does the same.

Recover activity

This dialog starts automatically when you recover an activity. It shows the actions that will be executed when you start the recovery.

This table shows an example:

Type	Name	Action	Reason
session	tezzz3110m000	Delete	Non activity component, option 'delete' set
label	tezzzabcd1234	Recover	Recover also Existing files
library	tezzzdbl310sb0	Recover	Recover also Existing files
table	tezzz102	Recover	Recover also Existing files

Click **OK** to perform the actions displayed. Click **Cancel** if you do not want to perform these actions.

After the recovery, the results are displayed. This table shows an example:

Type	Name	Action	Result
session	tezzz3110m000	Delete	Ok
label	tezzzabcd1234	Recover	Ok
library	tezzzdbl310sb0	Recover	Ok
table	tezzz102	Recover	Ok

Type
The component type, such as label or session.

Name
The name of the software component.

Action
The action that will be executed during the recovery.

Reason
The reason for the recovery action.

This field is displayed before you start the recovery.

Result
The result of the recovery action.

This field is displayed after the recovery.

Link Run Configurations

Use this window to link existing run configurations to an activity.

You can start this dialog from the "Activity Explorer view" on page 297.

The window shows the available run configurations. Select the desired configuration(s) and click **Link**.

Create IEX Patch

Use this dialog to create an IEX patch from an activity and place it in a folder.

Options

Primary language only
If this check box is selected, language-dependent components, such as reports, are only exported in the primary language.

If this check box is cleared, language-dependent components are exported in all available languages.

Include sources
If this check box is selected, sources, such as UI scripts, are included in the patch.

Destination

Development server
If this option is selected, the patch is created on the LN development server.

Client
If this option is selected, the patch is created on your client computer.

If this option is selected, the patch is created on another remote system. Click **New** to specify a remote system.

Disconnect connection after patch creation
If this check box is selected, the connection with the destination system is automatically closed when the patch is ready.

Folder
The folder, on the destination system, to store the patch.

Filename
The name of the patch file.

Browse and Select

Browse Command Sets

Use this dialog to select a command set for a session.

A command set is a selection of standard commands. At runtime, the user can run these commands through standard buttons and menu commands.

Each combination of a session type and a form type has its own command set, such as:

- D1.O: command set for display sessions with a type 1 form.
- M1.O: command set for maintenance sessions with a type 1 form.

You can start this dialog from the Session Editor.

To select a command set, click the desired set and click **OK** to close the browse dialog.

Browse component

Use this dialog to find and select a component.

You can start this dialog from various wizards and editors, such as:

- The Create a New Infor LN Software Component wizard: You can use the dialog, for example, to select a main table for a new session.
- The Table Editor: You can use the dialog, for example, to select a reference message for a table.
- The Session Editor: You can use the dialog, for example, to select a report for a print session.

To select a component:

1. Find the component: Browse the list of components or, in the **Name** field, type the first character(s) of the component name and click .
2. Click the desired component and click **OK** to close the browse dialog.

Note: The slider bar and the navigation buttons enable you to find components in one package only. To find components in a different package, type the package code in the **Name** field, and click .

For the following component types, you can use the **New** command to create a new component:

- Session
- Library
- Table
- Domain
- Report
- Message
- Question
- Label
- Menu

See the following section.

Create a new component

To create a new component:

1. Click **New**. The appropriate page of the Create a New Infor LN Software Component wizard starts. For example, the Library Properties page, or the Menu Properties page.
2. Fill in the required component properties.

3 Click **Finish** to generate the component and to close the Create a New Infor LN Software Component wizard.

- The Browse component dialog is closed automatically.
- The name of the new component is filled in automatically in the wizard or multipage editor, where you started the Browse component dialog.

Browse domain

Use this dialog to find and select a domain.

You can start this dialog, for example, from:

- The "Create a New Infor LN Software Component" on page 98 wizard.
- The "Report Editor" on page 190.
- The "Question Editor" on page 187.

To select a domain:

1 Find the domain: take one of the following steps:

- Browse the list of domains.
- Select **Search on Name**. Then, in the **Name** field, type the first character(s) of the domain name and click .
- Select **Search on Description / Search Key** and, in the **Package** field, select a package. Then, in the **Description / Search key** field, type a part of the domain's description / search key and click .

2 Click the desired domain and click **OK** to close the browse dialog.

Note: The slider bar and the navigation buttons enable you to find domains in one package only. To find domains in a different package, select **Search on Name**. Then, in the **Name** field, type the package code and click .

You can only select domains in packages that belong to the application displayed in the Component Explorer.

Use the **New** command to create a new domain. See the following section.

Create a new domain

To create a new domain:

1 Click **New**. The Domain Properties page of the "Create a New Infor LN Software Component" on page 98 wizard starts.

2 Fill in the required domain properties.

3 Click **Finish** to generate the domain and to close the Create a New Infor LN Software Component wizard.

- The Browse domain dialog is closed automatically.

- The name of the new domain is filled in automatically in the wizard or multipage editor, where you started the Browse domain dialog.

Browse Default Sizes

Use this dialog to select a default size, for example A4, for a report.

You can start this dialog, for example, from the Create a New Infor LN Software Component wizard and from the Report Editor.

To select a default size, click the desired size and click **OK** to close the browse dialog.

Browse label

Use this dialog to find and select a label.

You can start this dialog, for example, from the "Session Editor" on page 205.

To select a label:

1. Find the label: take one of the following steps:
   - Browse the list of labels.
   - Select **Search on Name**. Then, in the **Name** field, type the first character(s) of the label name and click 🔭.
   - Select **Search on Description / Search Key** and, in the **Package** field, select a package. Then, in the **Description / Search key** field, type a part of the label's description / search key and click 🔭.
2. Click the desired label and click **OK** to close the browse dialog.

Note: The slider bar and the navigation buttons enable you to find labels in one package only. To find labels in a different package, select **Search on Name**. Then, in the **Name** field, type the package code and click 🔭.

You can only select labels in packages that belong to the application displayed in the Component Explorer.

Use the **New** command to create a new label. See the following section.

Create a new label

To create a new label:

1. Click **New**. The Label Properties page of the "Create a New Infor LN Software Component" on page 98 wizard starts.
2. Fill in the required label properties.

3 Click **Finish** to generate the label and to close the Create a New Infor LN Software Component wizard.

- The Browse label dialog is closed automatically.
- The name of the new label is filled in automatically in the wizard or multipage editor, where you started the Browse label dialog.

Browse library

Use this dialog to find and select a library.

You can start this dialog, for example, from the "Table Editor" on page 238.

To select a library:

1 Find the library: complete one of these steps:

- Browse the list of libraries.
- Select **Search on Name**. Then, in the **Name** field, type the first character(s) of the library name and click .
- Select **Search on Description / Search Key** and, in the **Package** field, select a package. Then, in the **Description / Search key** field, type a part of the library's description / search key and click .

2 Click the desired library and click **OK** to close the browse dialog.

Note: The slider bar and the navigation buttons enable you to find libraries in one package only. To find libraries in a different package, type the package code in the **Name** field, and click .

You can only select libraries in packages that belong to the application displayed in the Component Explorer.

Browse product ID

Use this dialog to find and select a product ID.

You can start this dialog, for example, from the "Library Editor" on page 174.

To select a product ID:

1 Find the product ID: Browse the list of product IDs.

2 Click the desired product ID and click **OK** to close the browse dialog.

Browse Queries

Use this dialog to find and select a query.

You can start this dialog from the "Menu Editor" on page 178.

To select a query:

1 Find the query: Browse the list of queries.
2 Click the desired query and click **OK** to close the browse dialog.

Note: The slider bar enables you only to browse the list of queries currently displayed. To display the remaining queries, use the navigation buttons, or type the first character(s) of a query name, and click .

Browse session

Use this dialog to find and select a session.

You can start this dialog, for example, from the "Menu Editor" on page 178.

To select a session:

1 Find the session: Browse the list of sessions or, in the **Name** field, type the first character(s) of the session name and click .
2 Click the desired session and click **OK** to close the browse dialog.

Note: The slider bar and the navigation buttons enable you to find sessions in one package only. To find sessions in a different package, type the package code in the **Name** field, and click .

Use the **New** command to create a new session. See the following section.

Create a new session

To create a new session:

1 Click **New**. The Session Properties page of the "Create a New Infor LN Software Component" on page 98 wizard starts.
2 Fill in the required session properties.
3 Click **Finish** to generate the session and to close the Create a New Infor LN Software Component wizard.
   - The Browse session dialog is closed automatically.
   - The name of the new session is filled in automatically in the wizard or multipage editor, where you started the Browse session dialog.

Browse Input Fields

Use this window to select table fields for a report.

You can start this dialog from the "Report Editor" on page 190.

Click the desired field and click **OK** to close the dialog.

Note: The dialog displays the fields of a single table. You can click **Browse** to select another table.

Browse text field

Use this dialog to find and select a text field.

You can start this dialog, for example, from the "Session Editor" on page 205.

To select a text field:

1. Find the text field: Browse the list of text fields or, in the **Name** field, type the first character(s) of the text field name and click .
2. Click the desired text field and click **OK** to close the browse dialog.

Note: The slider bar and the navigation buttons enable you to find text fields in one package only. To find text fields in a different package, type the package code in the **Name** field, and click .

Browse time Format / Browse date format

Use this dialog to find and select a time or date format.

You can start this dialog, for example, from the Domain Editor. For domains of data type "UTC Date/Time", the dialog displays a list of time formats. For domains of data type "Date", the dialog displays a list of date formats.

To select a format:

1. Find the format: Browse the list of formats.
2. Click the desired format and click **OK** to close the browse dialog.

Select a Chart

Use this dialog to link a chart to a session. Each session can have several charts.

You can start this dialog from the "Session Editor" on page 205.

Session
The session to which you link the chart.

Chart
The chart name. The name is a combination of the package, the module, and the chart code.

Click **Browse...** to start a dialog where you can select charts.

Chart Group
You can change the chart group in the UI script of the session.

Default value

Chart group 1.

Select Component(s)

Use this dialog to add software components from the LN server to your current activity.

To start the dialog:

1. In the Activity Explorer view, select an activity, or a subfolder, or a component in an activity.
2. On the LN Studio toolbar, click **Select a Software Component**, or press ALT+Q.

The dialog consists of 2 tabs, that you can use to search in 2 different ways:

- **Select on code**
- **Search on description**

Note: When you add a component to your activity, the appropriate editor can start automatically for the component. This depends on your workbench preference settings. See the description of the Preferences - Workbench dialog.

Select on code

Use this tab to search on the first part of the component code.

To add a component to the selected activity

To add a component to the selected activity:

1. Optionally: In the upper right part of the tab, click ▼. The Select Component Types dialog starts. In this dialog you can specify additional settings for the search process.
2. In the **Component Code** field, enter the first part of the component code. For example: `tdpur01`.
3. Click ···, or press CTRL+SPACE. A list of components, which match the pattern, is displayed. If you type additional characters, LN Studio automatically updates this list.
4. If you want the dialog to close automatically after you click **OK**, select the **Close Dialog after processing selection** check box.
5. Select the desired component and click **OK**. The component is now added to your current activity.

To sort search results

- To sort the search results by type, name, or description, click the corresponding column header.
- To toggle between ascending and descending sort mode, click the involved column header.

Note: As you type the package code and the module code (the first 5 characters of the component code), LN Studio automatically shows a list of matching packages and modules, and updates this list after each character typed. For example:

Component Code	Search results displayed
t	All packages starting with t.
td	Package td and all its modules.
tdr	All modules of package td that start with r. For example, tdrec and tdrpl.

If you press ENTER after you entered a pattern in the **Component Code** field, LN Studio automatically adds all components, whose names exactly match the pattern, to your current activity.

Fields

Component Code
Enter the (first part of the) component code you are searching for.

To display a list of components, which match the pattern, click ..., or press CTRL+SPACE.

Type
The component type, such as session or label.

Name
The component name.

Description
The description of the component.

Additional Information
Additional information about the component. For example, the label context.

Close Dialog after processing selection
If this check box is selected, the dialog is closed automatically after you click **OK**.

If this check box is cleared, the dialog is still displayed after you click **OK**, so you can perform another search.

Search on description

Use this tab to search on a part of the component's description.

To add a component to the selected activity:

1. Optionally: In the upper right part of the tab, click ▼. The Select Component Types dialog starts. In this dialog you can specify additional settings for the search process.
2. In the **Package** field, select a package from the list.
3. Enter the search pattern, and click 🔭. A list of components that match the search pattern is displayed.
4. If you want the dialog to close automatically after you click **OK**, select the **Close Dialog after processing selection** check box.
5. Select the desired component and click **OK**. The component is now added to your current activity.

Fields

Package
Select a package from the list.

Search Pattern
Enter the search pattern.

To display a list of components, which match the pattern, click .

Type
The component type, such as session or label.

Name
The component name.

Description
The description of the component.

Close Dialog after processing selection
If this check box is selected, the dialog is closed automatically after you click **OK**.

If this check box is cleared, the dialog is still displayed after you click **OK**, so you can perform another search.

Select Component Types

Use this dialog to specify additional settings for the search process that you want to start in the "Select Component(s)" on page 94 dialog.

To define the additional settings:

1. Use the check boxes to select the components you want to search.
2. Optionally, clear the **Expired** check box to exclude expired components from the search.
3. Specify a maximum number of components to be displayed per search.
4. Click **OK** to save the settings and to return to the Select Component(s) dialog.

Note: Click **Reset Defaults** to return to the default settings for this dialog.

Select a Report

Use this dialog to link a report to a session. A session can have several reports.

You can start this dialog from the "Session Editor" on page 205.

Session
The session to which you link the report.

Report
The report name. A report is uniquely identified by the combination of package code, module code, and report code.

Click **Browse** to start a dialog where you can select reports.

Report Group
The report groups are useful if specific reports must be used in specific situations. You can change the report group in the session's UI script. Only the report(s) of one report group can be selected at run time.

Default value

Report group 1.

Example

A session has various reports divided into 2 report groups: group 1 - "Detailed Reports" and group 2 - "Summary Reports". The session's form contains a "print what" selection field where users can select whether they want to print a "Detailed Report" or a "Summary Report". If a user selects "Detailed Report", the user can only select a report from group 1. If a user selects "Summary Report", the user can only select a report from group 2.

Select Table Fields

Use this window to select table fields for a report.

You can start this dialog from the "Create a New Infor LN Software Component" on page 98 wizard.

Click the desired fields and click **OK** to close the dialog.

Note: The dialog displays the fields of a single table. You can click **Browse** to select another table.

You can select multiple fields in one go.

Create a New Software Component Wizard

Creating an additional file

Use the Additional File Properties window of the "Create a New Infor LN Software Component" on page 98 wizard to initialize and create a new *additional file* in your local workspace. This window is the wizard's second dialog box.

For more general information about the wizard, refer to "Create a New Infor LN Software Component" on page 98.

Note: The Create a New Infor LN Software Component wizard does not really create a file, but links an existing file, for example an XML schema file or a GIF image, to a specific package, module, and package VRC. Therefore, before you start the wizard, put the additional file in a directory on your machine.

To create an additional file

1. Start the wizard and select the component type.
   a. Start the wizard.
   b. Specify the software development activity for which you want to create the new additional file. Select `Additional File` as the component type.
   c. To go to the Additional File Properties window, click **Next**.
2. Specify the required additional file properties.
   In the Additional File Properties window, specify this information:

   Name

   To enter the name, specify a package code and a module code, and click **Browse**. Subsequently, browse to the folder where the additional file is stored and select the file. The name of the selected file is appended to the package and module name. The name of the folder is displayed in the **File Path** field.

   For example, you enter package tc and module com, and you select the test01.gif file. As a result, the additional file name becomes tccomtest01.gif.

   Description
   The description of the additional file.

   All fields of the Additional File Properties window are mandatory. After you created the additional file, you cannot change its name. However, you can change the description.

   For more information about the additional file properties, refer to the "Additional File Editor" on page 148.
3. Create the additional file.
   To create the additional file in your local workspace and to exit the wizard, click **Finish**. The created additional file appears in the Activity Explorer view of the Application perspective.

   You can now edit the new additional file in the Additional File Editor.

Create a New Infor LN Software Component

Use this wizard to create a software component.

In this wizard, you can create these software components in your local workspace:

- Session
- Report
- Table
- Domain

- Library
- Function
- Menu
- Label
- Message
- Question
- Additional File

After you create the component, it is displayed in your Activity Explorer view. Edit the component using a component-specific editor.

In the first window of the wizard, specify the software development activity. Then, select the type of component you want to create for this activity.

In the second window of the wizard, fill in some component-specific properties to initialize the new component. Some of these properties, such as the name of the component, can only be set when you create the new component using the wizard, and cannot be changed afterwards.

To create a component

To create a software component:

1 Start the wizard

To start the Create a New Infor LN Software Component wizard, on the **File** menu of LN Studio, select **New > Infor LN Component** .

2 Select activity

Specify the activity in the **Activity** field. This is the software development activity in which you create the new component.

3 Select component type

Specify which type of component you want to create in the **Component Type** field.

4 Specify component properties

To continue in the following page of the wizard, click **Next**. This second window is specific to the component you previously selected and used to set some required properties for the new component.

Fill in the required properties and click **Finish** to create the component in your local workspace.

The created component appears in the Activity Explorer view of the Application perspective. You can now edit the new component in the corresponding editor.

Note: Some of these properties, such as identifying attributes and references to labels, can only be set when you create a new component. For components already created, these properties cannot be changed.

For more information, see the Help topic on the concerned component.

LN Software Component

Activity
The name of the software development activity for which you create the component.
Behind the activity, the involved software development project is indicated between square brackets.

Allowed values

You must select one of the activities from the drop-down list. This lists contains all activities that are currently open in your workspace.

Component Type
The component type.

Allowed values

This table shows the component types you can select from the list:

Component Type	Description
Session	Sessions are the basic functional elements of LN applications and are used to perform actions or tasks.
Report	LN reports are used to output data from the database to a variety of devices, such as printers, displays, and files. One or more LN reports must be linked either to an LN print session or to an LN SQL Query.
Table	Tables are data structures that are used to store data and that consists of a list of records, each entry being identified by a unique key and containing a set of related values.
Domain	A domain specifies the data type, the value range, (default) display format, length and so on. Domains can be linked to table fields, form fields, and program variables.
Library	A library is a collection of files, programs or subroutines that is used to perform specific tasks for other libraries, program scripts or sessions of LN applications.
Function	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. With the #include statement you can include a function in a *program script*.
Menu	An LN menu is a list of options from which a user can make a selection to perform a desired action, such as starting sessions, other menus, and queries.
Label	An LN label is code that is used instead of language-dependent text in forms, reports, and menus. A label consists of a name and a content description.
Message	LN messages are language independent software components used to show situation-dependent information to the user.
Question	LN questions are language independent software components that are used to ask situation-dependent questions to which the user must respond.
Additional File	A generic component, such as an XML schema file, a GIF image, and so on.

Creating a domain

Use the Domain Properties window of the Create a New Infor LN Software Component wizard to initialize and create a new Domain in your local workspace. This window is the wizard's second dialog box.

For more general information about the wizard, refer to "Create a New Infor LN Software Component" on page 98.

Note: To use the new domain in an LN application, convert the domain to runtime.

To create a domain

1 Start the wizard and select the component type.

a Start the wizard.

b Specify the software development activity for which you want to create the new domain. Select `Domain` as the component type.

c To go to the Domain Properties window, click **Next**.

2 Specify the required domain properties.

In the Domain Properties window, specify this information:

Name
This is a text value. The domain name must start with the code of the package in which the domain is used.

Description
The description of the domain.

Data Type
The data type of the domain, which is the internal representation of data, such as a string, long, double, date or UTC Date Time.

If you select a text data type (`String`, `Set`, `Enumerated` or `MultiByte String`), specify a **Length** value greater than 0.

Length
The length in characters if the data type of the domain is a (multi-byte) string, or the default display length for data types UTC Date Time, Date, Enumerated and Set.

For the latter two data types, the length defines the maximum number of characters of the descriptions of the Enumerated or Set data type. Any description with more characters will simply be cut off at the specified length.

The name, description and data type are always mandatory. The length is only mandatory for data types `String`, `Set`, `Enumerated` and `MultiByte String`.

After you created the domain, you cannot change its name anymore. You can change the other properties.

For more information about the domain properties, refer to the "Domain Editor" on page 154.

3 Create the domain.

To create the domain in your local workspace and to exit the wizard, click **Finish**. The created domain appears in the Activity Explorer view of the Application perspective.

You can now edit the new domain in the Domain Editor.

Creating a function

Use the Function Properties window of the Create a New Infor LN Software Component wizard to initialize and create a new function in your local workspace. This window is the wizard's second dialog box.

For more general information about the wizard, refer to "Create a New Infor LN Software Component" on page 98.

Note: Before you use the new function in an LN application, build the function in LN Studio. If the auto-build preference is switched on, the function will be built automatically.

To create a function

1. Start the wizard and select the component type.
   a. Start the wizard.
   b. Specify the software development activity for which you want to create the new function. Select `Function` as the component type.
   c. To go to the Function Properties window, click **Next**.
2. Specify the required function properties.
   In the Function Properties window, specify this information:

   Name
   This is a text value. The function name must start with the code of the package and module in which the function is used.

   Description
   The description of the function.

   All fields of the Function Properties window are mandatory. After you created the function, you cannot change its name anymore. You can change the description.

   For more information about the function properties, refer to the "Function Editor" on page 166.
3. Create the function.
   To create the function in your local workspace and to exit the wizard, click **Finish**. The created function appears in the Activity Explorer view of the Application perspective.

   The wizard of LN Studio will automatically create some general source code for the new function, including a header with metadata and an empty 'skeleton' function.

   You can now edit the new function in the Function Editor.

Creating a label

Use the Label Properties window of the Create a New Infor LN Software Component wizard to initialize and create a new label in your local workspace. This window is the wizard's second dialog box.

For more general information about the wizard, refer to "Create a New Infor LN Software Component" on page 98.

Note: Before you use the new label in an LN application, compile the label.

To create a label

1 Start the wizard and select the component type.

a Start the wizard.

b Specify the software development activity for which you want to create the new label. Select `Label` as the component type.

c To go to the Label Properties window, click **Next**.

2 Specify the required label properties.

In the Label Properties window, specify this information:

Name
This is a text value. The label name must start with the code of the package in which the label is used.

Description
The actual text of the label. You create the label to apply this text in various situations.

Context
The context indicates in which situations you will use the label.

You can only create labels with one of these contexts:

- General Use
- Form Commands
- External Use
- Cascading Item on Button

Labels with other contexts are generated automatically when you create the corresponding software components. For example, when you create a session, the Create a New Infor LN Software Component wizard stores the session description you entered in a label with context **Session Description**.

All fields of the Label Properties window are mandatory. After you created the label, you cannot change its name and context anymore. You can change the description. As a default, the label type is **General Use**.

For more information about the label properties, refer to the "Label Editor" on page 168.

3 Create the label.

To create the label in your local workspace and to exit the wizard, click **Finish**. The created label appears in the Activity Explorer view of the Application perspective.

The label will be created in the primary language of the application.

You can now edit the new label in the Label Editor.

Note: A new label with one variant will be created. The specified description will become the description of the label and the description of the label variant. In the label editor, you can only edit the latter.

Creating a library

Use the Library Properties window of the Create a New Infor LN Software Component wizard to initialize and create a new Library in your local workspace. This window is the wizard's second dialog box.

For more general information about the wizard, refer to "Create a New Infor LN Software Component" on page 98.

Note: Before you use the new Library in an LN application, build the library in LN Studio. If the auto-build preference is switched on, the library is built automatically.

To create a library

1 Start the wizard and select the component type.

a Start the wizard.

b Specify the software development activity for which you want to create the new library. Select `Library` as the component type.

c To go to the Library Properties window, click **Next**.

2 Specify the required library properties.

In the Library Properties window, specify this information:

Name
This is a text value. The library name must start with the code of the package and module in which the library is located.

Description
The description of the library.

Type
Select a library type from the drop-down list. The following library types are used:

- General Library: General DLL - Dynamic Link Library
- Integration DLL: Dynamic Link Library that integrates between packages

All fields of the Library Properties window are mandatory. After you created the library, you cannot change its name anymore. You can change the other properties. As a default, the library type is **General Library**.

For more information about the library properties, refer to the "Library Editor" on page 174.

3 Create the library.

To create the library in your local workspace and to exit the wizard, click **Finish**. The created library appears in the Activity Explorer view of the Application perspective.

The wizard automatically creates some general source code for the new library, including a header with metadata and an empty 'skeleton' function.

If the name of the DLL consists of the table code with "ue" as suffix, such as whinh200ue, a User Exit DLL will be created. See "User Exit DLL Overview" in the *Infor ES Programmer's Guide*.

You can now edit the new library in the Library Editor.

Creating a menu

Use the Menu Properties window of the Create a New Infor LN Software Component wizard to initialize and create a new menu in your local workspace. This window is the wizard's second dialog box.

For more general information about the wizard, refer to "Create a New Infor LN Software Component" on page 98.

To create a menu

1. Start the wizard and select the component type.
   a. Start the wizard.
   b. Specify the software development activity for which you want to create the new menu. Select `Menu` as the component type.
   c. To go to the Menu Properties window, click **Next**.
2. Specify the required menu properties.
   In the Menu Properties window, specify this information:

   Name

   The name of the menu, which includes the following:

   - The package code (2 characters).
   - The module code (3 characters).
   - The menu identification code (8 characters).

   After you created the menu, you cannot change its name.

   Description
   LN Studio automatically stores the description in a label. If you want to change the description after the menu is created, edit this label in the label editor.

   All fields of the Menu Properties window are mandatory.

   For more information about the menu properties, refer to the "Menu Editor" on page 178.
3. Create the menu.
   To create the menu in your local workspace and to exit the wizard, click **Finish**. The created menu appears in the Activity Explorer view of the Application perspective.

   You can now edit the new menu in the Menu Editor.

Creating a message

Use the Message Properties window of the Create a New Infor LN Software Component wizard to initialize and create a new message in your local workspace. This window is the wizard's second dialog box.

For more general information about the wizard, refer to "Create a New Infor LN Software Component" on page 98.

To create a message

1 Start the wizard and select the component type.

   a Start the wizard.

   b Specify the software development activity for which you want to create the new message. Select `Message` as the component type.

   c To go to the Message Properties window, click **Next**.

2 Specify the required message properties.

   In the Message Properties window, specify this information:

   Name
   This is a text value. The message name must start with the code of the package in which the message is used.

   Description
   The actual text of the message. Create the message to apply this text in various situations. The message can contain codes that are substituted when the message is displayed.

   Message Type
   The message type indicates whether the message represents information, a warning or an error.

   All fields of the Message Properties window are mandatory. After you create the message, you cannot change its name, but you can change the other properties. As a default, the message type is **Warning**.

   For more information about the message properties, refer to the "Message Editor" on page 183.

3 Create the message.

   To create the message in your local workspace and to exit the wizard, click **Finish**. The created message appears in the Activity Explorer view of the Application perspective.

   The message will be created in the primary language of the application.

   You can now edit the new message in the Message Editor.

Creating a question

Use the Question Properties window of the Create a New Infor LN Software Component wizard to initialize and create a new question in your local workspace. This window is the wizard's second dialog box.

For more general information about the wizard, refer to "Create a New Infor LN Software Component" on page 98.

To create a question

1 Start the wizard and select the component type.

   a Start the wizard.

   b Specify the software development activity for which you want to create the new question. Select `Question` as the component type.

c To go to the Question Properties window, click **Next**.

2 Specify the required question properties.

In the Question Properties window, specify this information:

Name
This is a text value. The question name must start with the code of the package in which the question is used.

Description
The actual text of the question. Create the question to apply this text in various situations.

Question Type
The question type indicates the severity of the question as information, warning or error. As a default, the question type is **Warning**.

Domain
The domain that determines the possible answers to the question. Click **Browse** to select a domain. The domain must have the data type **Enumerated**.

Default Answer
The default answer to the question. Select one of the enumerate constants belonging to the specified domain.

All fields of the Question Properties window are mandatory. After you created the question, you cannot change its name anymore. You can change the other properties.

For more information about the question properties, refer to the "Question Editor" on page 187.

3 Create the question.

To create the question in your local workspace and to exit the wizard, click **Finish**. The created question appears in the Activity Explorer view of the Application perspective.

The question will be created in the primary language of the application.

You can now edit the new question in the Question Editor.

Creating a report

Use the Report Properties window of the Create a New Infor LN Software Component wizard to initialize and create a new report in your local workspace. This window is the wizard's second dialog box.

For more general information about the wizard, refer to "Create a New Infor LN Software Component" on page 98.

Note: To implement the new report in an LN application, you must synchronize with the server and compile the report.

To create a report

1 Start the wizard and select the component type.

a Start the wizard.

See "Create a New Infor LN Software Component" on page 98.

b Specify the software development activity for which you want to create the new report.

c Select **Report** as the component type.

d Click **Next**. The Select Report Mode window is displayed.

2 Select the report mode.

a In the Select Report Mode window, select how you want to create the report.

- To generate the report, select the **Generate Report** check box.
- To create the report manually, clear the **Generate Report** check box.

b Click **Next**.The Report Properties window is displayed.

3 Specify the required report properties.

The layout of the Report Properties window depends on how you create the report.

- If you create the report manually, specify this information:

  Name
  This is a text value. The report name must start with the code of the package and the module in which the report is used. After the report is created, you cannot change its name.

  Description
  This is a text value. LN Studio automatically stores the description that you enter in a label. If you want to change the description after the report is created, edit this label in the label editor.

  Report Size
  To specify a size, click **Browse** and select a value from the Browse Default Sizes dialog box.

- If you generate the report, specify this information:

  Table
  The table code of the main table. The combination of the package code, the module code, and the table number identifies a table. Enter the package code of the table, then click **Browse** and select a table from the "Browse table" on page 88 dialog box.

  Use All Table Fields
  If this check box is selected, all table fields of the main table are printed on the report. If this check box is cleared, click **Select Fields...** and select the desired fields from the Select Table Fields dialog box.

  Report Code
  The identification code of the report. A report is identified by a unique combination of package code, module code and report identifier. To regenerate an existing report, click **Browse** and select the report code from the Browse component dialog box. After the report is created, you cannot change its code.

  Report Description
  LN Studio automatically stores the description that you enter in a label. If you want to change the description after the report is created, edit this label in the label editor.

  Report Size
  To specify a size, click **Browse** and select a value from the Browse Default Sizes dialog box.

  Report Type

  Choose one of these options:

- Type 1: Single occurrence report. The fields on the report are printed vertically. Use this report type if a lot of fields, that do not fit horizontally on a report, must be printed.
- Type 2: Multi occurrence report. The fields on the report are printed horizontally.
- Type 3: Multi occurrence report with a group layout. The fields on the report are printed horizontally. In addition you can print group headers and group totals.

Use Index
Specify the number of the index. The index is used to print the report in the order according to the key fields specified in the index.

New Page On
This field is only applicable for type 3 reports.

The number of group fields which must be printed on a new page. If one of these fields changes, a page break is forced.

The total number of group fields must be less than the total number of key fields.

New Page Off
This field is only applicable for type 3 reports.

The number of group fields which must be printed on a new page.

The total number of group fields must be less than the total number of key fields.

Overwrite Existing Components
If this check box is selected, existing software components are overwritten.

For more information about the report properties, refer to the "Report Editor" on page 190.

4 Create the report.

To create the report in your local workspace and to exit the wizard, click **Finish**. The created report appears in the Activity Explorer view of the Application perspective.

You can now edit the new report in the Report Editor.

Note: When you create a report in a new application, these labels for general use are automatically created in the `label` folder in the Activity Explorer:

- `[package code]ttgen.h.comp.lbl` - Company
- `[package code]ttgen.h.date.lbl`- Date
- `[package code]ttgen.page.lbl` - Page

Creating a session

Use the Session Properties window of the Create a New Infor LN Software Component wizard to initialize and create a new session in your local workspace. This window is the wizard's second dialog box.

For more general information about the wizard, refer to "Create a New Infor LN Software Component" on page 98.

Note: To implement the new session in an LN application, synchronize with the server and compile the session.

To create a session

1 Start the wizard and select the component type.

a Start the wizard.

b Specify the software development activity for which you want to create the new session. Select `Session` as the desired component type.

c To go to the Session Properties window, click **Next**.

2 Specify the required session properties.

In the Session Properties window, specify the following properties:

- **Name**
- **Description**

The **Name** and **Description** have text values. The session name must start with the code of the package and the module in which the session is used. After the session is created, you cannot change its name.

The wizard stores the session description in a label. If you want to change the session description, edit this label in the Label Editor.

To specify one of the other properties is optional. If you do not specify a value, the properties are unset or get a default value (as displayed in the window). This table shows the used default values:

Property	Default value	Remarks
Name	-	Mandatory and read-only after the session has been created
Description	-	Mandatory
Standard Script	No	Mandatory
Script Type	UI Script with Database Handling	Mandatory if **Standard Script** is **No**
UI Script	-	Read-only field. Equal to the session's **Name** and **Description**
Session Type	Display	Mandatory
Window Type	Dialog	Mandatory
Main Session	True	Optional
Main Table	-	Optional

For more information on the session properties, see the "Session Editor" on page 205.

3 Create the session.

To create the session in your local workspace and to exit the wizard, click **Finish**. The created session appears in the Activity Explorer view of the Application perspective.

You can now edit the new session in the Session Editor.

Creating a table

Use the Table Properties window of the Create a New Infor LN Software Component wizard to initialize and create a new table in your local workspace. This window is the wizard's second dialog box.

For more general information about the wizard, refer to "Create a New Infor LN Software Component" on page 98.

To create a table

1 Start the wizard and select the component type.

   a Start the wizard.

   b Specify the software development activity for which you want to create the new table. Select `Table` as the component type.

   c To go to the Table Properties window, click **Next**.

2 Specify the required table properties.

   In the Table Properties window, specify the following properties for the table.

   Name
   The name of the table.

   Table names must have this format: `<package code><module code><table number>`. For example:

Package Code	tt	(tools)
Module Code	adv	(application development)
Table number	001	(sequence number of table)

   After you created the table, you cannot change its name.

   Description
   LN Studio automatically stores the description in a label. If you want to change the description after the table is created, edit this label in the label editor.

   All fields of the Table Properties window are mandatory.

   For more information about the table properties, refer to the "Table Editor" on page 238.

3 Create the table.

   To create the table in your local workspace and to exit the wizard, click **Finish**. The created table appears in the Activity Explorer view of the Application perspective.

   You can now edit the new table in the Table Editor.

Debugging

Execute Assignment Expression

Use this dialog to specify an expression. The expression will be executed only once.

You can start this dialog from the shortcut menu in the Expressions view.

Expression
Specify the expression in this format: `<variable>:=<value>`.

For example: `curr.compnr:=299`

Note:

- You can select expressions specified earlier from the list.
- If you select a variable name in the script editor before you start the **Execute Assignment Expression** command, this expression is displayed in the dialog: `<variable name>:=`

  For example, you select the `task.child.process` variable in the script. Then, on the shortcut menu, you click **Execute Assignment Expression**. The Execute Assignment Expression dialog starts and contains this expression: `task.child.process:=`

Add Watch Expression
If this check box is selected, the name of the variable is displayed in the Expressions view.

For example, if you enter `curr.compnr:=299`, the `curr.compnr` variable is displayed in the Expressions view.

If this check box is cleared, the name of the variable is not displayed in the Expressions view.

Line Breakpoint properties

Use this dialog to view or modify the properties of a line breakpoint.

Note: You cannot use this dialog to create new breakpoints.

For details on how to set new breakpoints, see "Using breakpoints" on page 70.

Location
The component, such as a script or library, where the breakpoint is set.

Line Number
The line number in the source code where the breakpoint is set.

Enabled
If this check box is selected, the breakpoint is enabled. The program execution suspends when the breakpoint is reached.

If this check box is cleared, the breakpoint is disabled. The breakpoint is ignored during the program execution.

Method Breakpoint properties

Use this dialog to view or modify the properties of a method breakpoint.

Note: You cannot use this dialog to create new breakpoints.

For details on how to set new breakpoints, see "Using breakpoints" on page 70.

Location
The component, such as a script or library, where the breakpoint is set.

Function
The name of the method for which the breakpoint is set.

Enabled
If this check box is selected, the breakpoint is enabled. The program execution suspends when the breakpoint is reached.

If this check box is cleared, the breakpoint is disabled. The breakpoint is ignored during the program execution.

Function Entry
If this check box is selected, the program execution suspends when the execution reaches the function for which the breakpoint is set.

Watchpoint properties

Use this dialog to view or modify the properties of a watchpoint.

Note: For details on how to set new watchpoints, see "Using breakpoints" on page 70.

Location
The component, for example a script or library, where the watchpoint is set.

Variable
The name of the variable for which the watchpoint is set.

Enabled
If this check box is selected, the watchpoint is enabled. The program execution suspends, based on the setting specified in the **Suspend on** field.

If this check box is cleared, the watchpoint is disabled. The watchpoint is ignored during the program execution.

Suspend on
Indicates when the program execution will suspend.

Allowed values

Variable Modification	The program execution suspends when the value of the variable changes.
Variable Value	The program execution suspends when the variable gets the indicated value. See the previous figure for an example. The session will suspend when the `curr.key` variable gets the value 2.

Preference Pages

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 will be 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. As a result, all editor functionality that needs the parser's output drops out. This is done to enhance 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. As a result, all editor functionality that needs the parser's output drops out. This is done to enhance 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 will be 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 will be 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 will be 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 will be 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.

Verify Software Components

Verify Components View Filter

Use this dialog to specify a filter for the Verify Components view.

To start the dialog, click **Verify Warning Filter** in the Verify Components view's toolbar.

Enabled
Indicates whether the filter is enabled. You must select this check box if you want to define a filter.

Warning level
This field indicates the warning levels that will be displayed. You can select multiple levels. The selected levels are supplementary.

For example: You select **High (1)** and **Normal (2)**. Warnings of level 1 and 2 will be displayed.

Warning status
This field determines the warning statuses that will be displayed. You can select both check boxes. The options are supplementary.

Allowed values

Only blocking	Select this check box to display blocking warnings.
Exclude accepted	Select this check box to exclude accepted warnings from the list.

Note: The selections in the filter are supplementary.

For example, in the filter properties you select the **Normal (2)** warning level and the **Only blocking** status. As a result the following is displayed:

- All "Normal" warnings, regardless of their warning status.
- All blocking warnings, regardless of their warning level.

Resources
This field determines the resources, for which warnings will be displayed.

Allowed values

On any resource	LN Studio displays warnings for all resources in the Activity Explorer.
On any resource in same project	LN Studio displays warnings for the resource that you selected in the Activity Explorer, and for all resources that belong to the same project.
On selected resource only	LN Studio only displays warnings for the resource that you selected in the Activity Explorer.

Verify Warning Details

Use this dialog to perform these actions:

- View the details of *warnings* generated when you verify software components.
- Accept warnings.

This dialog is important in the procedure to handle warnings. See "Verifying software components" on page 58.

VSC Warning

Warning Id
The unique code that identifies the message assigned to the warning.

For example: the message ID for a warning is 5053. The corresponding message text is `Function 'sleep' is not supported by Web UI .`

Note: This field corresponds with the **VSC ID** column in the Verify Components view.

Message
The text of the message assigned to the warning.

For example: `Function 'sleep' is not supported by Web UI`

Detailed Information

Resource
The LN Studio resource that represents the software component for which the warning is generated.

For example:

- tfgld1101m000.ses (session)
- tfgld1101m000.sessrc (program script)

Project
The *software project* to which the software component belongs.

VSC Verification Code
The *verification code* for which the warning was generated.

Component name
The code that identifies the software component for which the warning was generated.

Component type
The type of the main component for which the warning was generated.

For example:

A warning is generated because of an error in a session's UI script:

- The component type is "session".
- The component subtype is "script".

Component subtype
The subcomponent type for which the warning was generated. Each software component type has its own types of subcomponents.

Some examples of subcomponents are:

- A form command in a form.
- A field in a form.
- An index in a table.
- A text field in a table.
- A field section in a UI script.
- A choice section in a UI script.

Component location
The location in the software component where the problem occurs.

For example:

- A line number in a script or library.
- A form or report in a session.

Component element
The component element that causes the problem.

For example:

A warning is generated because an expired label is used in a form:

- The component type is "session".
- The component subtype is "form".
- The component element is the expired label, such as `ttadv4122sDF1`.

VSC Status

Priority

The priority of the *warning*: high, normal, suspicious, or low.

Priorities are automatically assigned by the verification process:

- The priorities for warnings generated based on *source analyze codes*, are defined in the Source Analyze Codes (tlvsc3511m000) session.
- The priorities for warnings generated based on all other types of checks are hard coded in the *VSC* software on the LN server.

Blocking

This field indicates whether the warning blocks the check in of a component.

If a warning is "blocking", either solve or accept the warning immediately.

For more information, see "Verifying software components" on page 58.

Accepted

This field indicates whether the warning is accepted.

Accept

Acceptance allowed

This field indicates whether it is allowed to accept the warning.

Allowed values

True	It is allowed to accept the warning. The **Accept Warning** field is enabled.
False	You cannot accept the warning. The **Accept Warning** field is disabled.

Note: This field is read-only. The value is determined by the *rules* used by *VSC*.

Accept Warning

To accept the warning, select this check box. In the **Accept Text** field, enter the reason for acceptance.

To solve the warning, clear this check box.

For example, the following warning is generated: "Possible missing break in case statement".

- If this break is not missing, but left out on purpose, select this check box and accept the warning.
- If this break was left out accidentally, clear this check box and solve the warning.

For more information, refer to "Verifying software components" on page 58.

Accept Text
If you select the **Accept Warning** check box to accept a warning, enter the reason in this field.

Help
Starts the Verify Warning Help dialog, which displays information to solve the warning.

Verify Warning Help

This dialog shows help for the selected VSC warning.

You can start the dialog from the "Verify Warning Details" on page 124 dialog.

This table shows the information that is usually displayed:

Type of information	Description
Short Description	A short description of the error. For example: `Presence of goto statements`
Type	The warning type, such as Error or Warning.
Severity	The severity of the warning, such as High or Medium.

Description	A more detailed description of the error.
	For example:
	`The use of 'goto' statement is forbidden.`
	`Exception: within #defines it is allowed. Sometimes it is required (for example exception handling)`

Motivation	The reason why the situation indicated in the error message is not correct. For example: `'Goto' statements reduce the readability of script code dramatically.`
Possible Solution	A hint for a possible solution. For example: `Implement structured code (early return or if-then-else statements)`

Type of information	Description
Examples	Examples to understand and solve the error.

This dialog is important in the procedure to handle warnings.

See "Verifying software components" on page 58.

Miscellaneous

Check In comment

Use this dialog to enter check in comments for software components.

The dialog starts automatically when you check in one or more components.

Click **OK** to save the comment and to check in the component.

Click **Cancel** to cancel the check in. The comment is not saved.

Note: The revision text is stored in the configuration management system on the server. A single text is used for all revisions of a component. This text is displayed each time you check in a new revision of the component, so you can modify or extend it.

Use this comment for all selected components
This check box is only displayed if you check in multiple components in one go.

If this check box is selected, the revision text entered is used for all components you selected in the Activity Explorer.

If this check box is cleared, you must enter a separate revision text for each component.

Configuration Management

Use this dialog to view the configuration management properties of an LN Studio resource.

Configuration Management Information

Type
The type of configuration management system used for the resource.

There is only one type of configuration management: LN Configuration Management (BaanSCM), which is based on the *Software Configuration Management (SCM)* module in Enterprise Server. A more extensive CM solution will be offered in a future release. See "Activity based development" on page 39.

General Information

Checked Out
Indicates whether the resource is checked out.

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

A resource can get dirty, for example, if you do not create a new build after you have modified the resource.

To avoid dirty resources, select the **Build Automatically** option on the **Project** menu.

General Information

Type
The type of resource, such as session or library.

Component Name
The name of the resource.

Version
The VRC (*Version - Release - Customer*) to which the component belongs.

Expired
Indicates whether the resource is expired.

Configurations

Use the Run Configurations and Debug Configurations dialog boxes to create, run, or debug configurations.

To add a configuration:

1 Complete one of these steps:

- To add a run configuration, select **Run > Run Configurations** .
- To add a debug configuration, select **Run > Debug Configurations** .

2 In the left pane of the Run Configurations or Debug Configurations dialog box, right-click **Infor LN Session** and select **New**. These tabs are displayed:

- **Sessions**
- **Profiling**
- **Environment**
- **Common**

3 Specify the configuration settings in these tabs and click **Apply**.

For details on the tabs mentioned, see the field help.

Sessions

Use this tab to specify the session you want to start, and the company in which the session must run.

Activity
Select an activity from the list.

Session
Select the session you want to run. The name of the configuration is automatically filled with the name of the selected session.

To select a session, perform one of these actions:

- Select a session linked to your activity from the list.
- Click **Browse**. The Browse session dialog starts. In this dialog you can select a session on the LN server. The session does not have to be linked to your activity.
- Enter a session code. The session does not have to be linked to your activity.

Note: You can only select sessions that belong to the packages displayed in the Component Explorer

Launch Mode
Specify how the session will be started.

Allowed values

Launch in BW	The session is launched by BW (*BW Configuration*).
Attach to running Web UI on port number	Specify a port number where currently a Web UI instance is running. The session is started in this already running Web UI. See the Web UI About screen for the currently used port number.

Note: If you run a session in Web UI, any new or changed labels are only displayed if the session is compiled and linked to your activity.

Company
Select a company in which the session must be started.

Perform one of these actions:

- Select **Start session in company of connection point** and enter the company linked to your project's Runtime Address connection point. To find out the corresponding company number, complete the following steps:
  1. Select **Windows > Preferences**. The Preferences dialog is displayed.
  2. In the tree that is displayed in the left pane of the dialog, select **Infor LN JCA Connectivity**. The Infor LN JCA Connectivity dialog appears.
  3. Select the Runtime Address connection point and click **Edit**. The company number is displayed.
- Select **Start session in company** and specify another company.

Profiling

Optionally, use this tab to specify profiling parameters.

Profiling is used to display performance statistics for a session. For details, see "The Call Graph Profiler" in the *Infor ES Programmer's Guide*.

Enable profile mode
If this check box is selected, you can specify profiling parameters.

When you run the session, a Call Graph Profile is generated.

Sorted by runtime
Select this check box to add information that is sorted by runtime to the Call Graph Profile.

Filter
Use this field to specify a filter. If this check box is selected, you can, for example, enter the name of an object. The generated profile contains only statistics for the specified object.

This field corresponds with the value of the PROFILE_ALL variable. See "The Call Graph Profiler" in the *Infor ES Programmer's Guide*.

Alternative directory on Infor LN Server
If this check box is selected, you can specify a directory on the LN server to store the Call Graph Profile files. The default value is $BSE_TMP.

This field corresponds with the PROF_DIR variable. For details, refer to "The Call Graph Profiler" in the *Infor ES Programmer's Guide*.

Save profile results on local client
If this check box is selected, you can specify a directory on your client PC to store the Call Graph Profile files.

See the description of the PROF_CLIENT variable in the *Infor ES Programmer's Guide*.

Start default browser
If this check box is selected, the generated profile is automatically displayed in your default Web browser.

See the description of the PROF_CLIENT variable in the *Infor ES Programmer's Guide*.

Environment

Use this tab to specify environment variables.

Environment

Append environment to native environment
Select this option to append the environment variables to the native environment variables.

Replace native environment with specified environment
Select this option to replace the native environment variables with the specified environment variables.

Common

This tab is not used by LN Studio. It contains only standard Eclipse options. See the *Workbench User Guide*.

Copy from other layouts

Use this window to select report layouts you want to copy from an existing report to a new report.

You can start this dialog from the "Report Editor" on page 190.

Click the desired layouts and click **OK** to close the dialog.

Note: The dialog shows the layouts of a single report. To select another report, click **Browse Reports**.

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 will be 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" on page 26.

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" on page 39.

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 will be changed or modified in LN Studio.

Default value

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 will be 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.

Default value

The check box is cleared.

Documentation

Documentation
Use this field to specify additional information.

Create a New Form Command

Use this dialog window to create a new form command.

Session
The session to which you link the form command.

Activation Type
Specifies what your form command activates. You can select the following activation types from the drop-down list. The used type names are self explanatory.

Allowed values

- Function
- Session
- Business Method
- Menu

Function/Session/Business Method/Menu
The name of the function, session, business method, or menu the form command activates.

Note: Any session you specify must exist in the development repository of the connected LN server. The function or business method must be part of the session's program script, or DAL script respectively.

Allowed values

If the activation object is a session, you must specify a session that already exists in the LN application, because you cannot create new sessions in the LN Studio.

The **Browse** button only applies to sessions and not to other activation objects. Use the browse functionality to select a session.

For activation objects other than Session, you do not require to select an pre-existing object. Note, however, that any specified function or business method must be part of the session's program script, or DAL script respectively.

Command Type
Specifies where the form command appears on the session's form. You can select the following command types from the drop-down list:

Allowed values

Form	For all of the session's form tabs, the form command appears on the form's *appropriate menu*, and/or as a button on the form.
Group	The form command only appears on the session's *appropriate menu* when a user selects the tab that includes the group you define in the **Group/Field Number** field.
Field	The form command appears as a button on the form, behind the field whose **Sequence Number** you enter in the **Group/Field Number** field.
Print	The form command appears as an option in the session's print menu.

Group / Field Number
This field specifies the group to which the form command belongs or the sequence number of the field that is linked to the form command. Whether this field specifies a group or a sequence number depends on the value you specified as the **Command Type**: **Group** or **Field**.

This field only applies to form commands of type 'Group' or 'Field'. The field is disabled for the other form command types ('Form' and 'Print').

For form commands of **Command Type Group**, select the group to which the form command belongs. The form command only appears in the session's *appropriate menu* when the user selects the group's tab.

For form commands of **Command Type Field**, select the field behind which the form command appears as a field button.

The pull down list shows the numbers and label codes of the available groups or fields.

Allowed values

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

Duplicate an Infor LN Software Component

Use this wizard to make a copy of an LN Studio component in your local workspace. The copy is an exact duplicate, with the exception of the name of the component. In the wizard, you must specify this name. Make sure you enter a new, unique name.

The wizard saves the copied component on the file system and, if applicable, adds it to source control.

You can only duplicate a component if you are authorized to do so:

- You cannot duplicate components that were automatically generated by LN Studio, such as a label with the `Session, Table or Report Description` context when you create a new session, table or report.
- You cannot duplicate components from an LN Studio Activity that has been re-assigned to another user. After an LN Studio Activity has been re-assigned, you loose the ownership of the activity's components.

To duplicate an LN Studio software component

1 Start the wizard for an existing component

In the Activity Explorer view, right-click the LN Studio component you want to duplicate and select **Duplicate Component**. This starts the Duplicate an Infor LN Software Component wizard.

The component that is to be copied must be present in the current LN Studio Activity project.

The Duplicate an Infor LN Software Component wizard displays the following information about the component you selected for duplication:

Field	Description
Project	The current LN Studio Activity in which the component will be duplicated. The LN Studio Software Project to which this activity belongs is indicated behind the activity name, between square brackets.

Field	Description
Component Type	The type of component, such as Library or Session.
Source Name	The name of the original component that will be duplicated. This name is the identifying attribute of the component.
Source Description	The description of the original component that will be duplicated.

All these properties of the existing component are displayed in read-only mode.

2 Specify properties for the new component

Specify the following properties for the new component:

Field	Description	
Target Name	Enter a unique component name. The format of the name is component specific. Most component names must start with the code of the package in which the component is used. Some component names must additionally include the code of the involved module. For more information on the allowed values, refer to the wizard's messages and the description of the editor of the concerned component. After you copied a component, you cannot change the component's name anymore. For existing components, the name is read-only, because the name forms the identifying attribute of the component. Changing a component's name would make it another component.	
Target Description	Specifying a **Target Description** is optional. If you do not fill in a description, the component inherits the description of the component from which it will be copied. The component description can be any ASCII text with a length of less than 60 characters. **Note:** You cannot specify a **Target Description** for labels, because the label description is one of the label variants and all label variants are duplicated.	
Program Script	This field is only displayed when you duplicate a session. Select one of the following options:	
	Duplicate Script	The UI script of the original session is copied. The new UI script is linked to the target session. Select this option if the new session must have its own UI script.
	Create Reference to Source Session	The new session is linked, through a session reference, to the UI script of the original session, so that both sessions share the same UI script. See the description of the **Session reference** field in the online help of the Session Editor.
DAL options	These options are only displayed when you duplicate a DAL script. The duplicate wizard copies the selected (source) DAL script to a new (target) DAL script. In addition, the wizard also copies the table, to which the source DAL script belongs, to a new table. The new DAL script contains references to the original table. Use the following options to automatically update these references, so the new DAL script refers to the new table.	

Field	Description	
	Replace code '<name of the original DAL script>'	If this check box is selected, the wizard automatically replaces all references to the original table name by references to the new table name. If this check box is cleared, the references are not updated, so the new DAL script still contains references to the original table name.
	Replace description '<description of the original DAL script>'	If this check box is selected, the wizard automatically replaces all references to the original table description by references to the new table description. If this check box is cleared, the references are not updated, so the new DAL script still contains references to the original table description.

3 Duplicate the component

To duplicate the selected component in your local workspace and to exit the wizard, click **Finish**. The duplicated component appears in the Activity Explorer view of the Application perspective.

You can now edit the duplicated component through the corresponding editor. For example, you can edit a duplicated library through the Library Editor.

Some components have dependent child components, such as forms, scripts, libraries or labels. When you duplicate a component, these child components are copied as well (see remarks in table below).

Component	Remarks
Session	When you duplicate a session, the session's form is also copied. If you select the **Duplicate Script** option, the UI script is also copied.
Domain	When you duplicate an enumerated domain, the labels used for the enumeration are also copied.
Label	When you duplicate a label, all label variants will be duplicated (including the label description).

Application Search

Use this dialog to find out in which LN software components an element, selected in the script editor, is used.

The search results are displayed in the **Search** view. Double-click a search result to open the corresponding script or library at the line where the result occurs.

Note:

- This dialog is only available for LN servers with Enterprise Server 8.7 or higher.

- The component you are searching for must exist. If the specified component does not exist, a message is displayed.

You can start this dialog from the shortcut menu in the script editor.

Activity
The activity in which you started the **Application Search** command.

Expression / Component Name
The expression or component name you are searching for. The element that you selected in the script editor is automatically filled in.

Search in Scripts
If this option is selected, the expression or component name is searched in the scripts/sources of the component types specified in the **Component Types** field.

You can perform the search in these component types:

- Sessions
- Functions
- Tables
- Reports
- Libraries

Search as
The component type you want to search for.

The expression or component name is searched in all parts of the component types specified in the **Component Types** field.

Allowed values

Value	Description
Message	The search process searches only for messages. You can perform the search in these component types: • Sessions • Functions • Tables • Reports • Libraries • Domains • Report Designs • BFlow components • Workbenches
Question	The search process searches only for questions. You can perform the search in these component types: • Sessions • Functions

Value	Description
	• Libraries • BFlow components
Session	The search process searches only for sessions. You can perform the search in these component types: • Sessions • Functions • Libraries • BFlow components • Menus
BFlow Component	The search process searches only for BFlow components. You can perform the search in BFlow components.
Table/Field	The search process searches for tables or table fields. The type of search depends on the length of the component name you specify in the **Expression / Component Name** field: • If you specify 8 characters or less, the search process searches for tables. • If you specify more than 8 characters, the search process searches for table fields. You can perform the search in these component types: • Sessions • Functions • Tables • Reports • Libraries • BFlow components
Library	The search process searches only for libraries. You can perform the search in these component types: • Sessions • Reports • Functions • Libraries • Report Designs • Menus This option is only available if it is supported by the LN server.
Label	The search process searches only for labels. You can perform the search in these component types: • Sessions • Reports

Value	Description
	• Functions • Libraries • Report Designs • Workbenches • Menus • Domains This option is only available if it is supported by the LN server.

The search process performs the same search actions as the "Print Where Used" sessions on the LN server. The search results are displayed in the **Search** view.

Note: This option is only available for LN servers with Enterprise Server 10.3 or higher.

Component Types
The component types in which the search will be performed.

Package Selection
The packages in whose components the search must be performed.

Options

Case sensitive
If this check box is selected, the search will distinguish uppercase from lowercase letters.

Include expired components
If this check box is selected, the search process will also search in expired components.

Exclude comment lines
If this check box is selected, the search process will not search in comment lines.

Range

From/To
The range of components in which the search must be performed.

The package code must be omitted. The specified code must start with a module code. For example, you want to search in components in the com module in the tc package. Select tc in the **Package Selection** field, and specify this range:

From	To
com	comzzzzzzzzzzzzzzzzz

Generate Library from WSDL file

Use this dialog to generate LN 4GL code for a general library. To generate the library, you must select a Web Services Description Language (WSDL) file or a URL.

Note: This dialog is only available for LN servers with Enterprise Server 8.7 or higher.

You can start this dialog from the shortcut menu in the script editor, when you edit the source code of a general library.

Note: For information on how to use the generated LN code, see "Web Services from LN " in the *Infor ES Programmer's Guide*.

WSDL File
The absolute path name of the WSDL file to be imported.

Note:

Click **Browse** to start a dialog where you can locate and select the WSDL file.

WSDL URL
The URL that will be used to generate the library code.

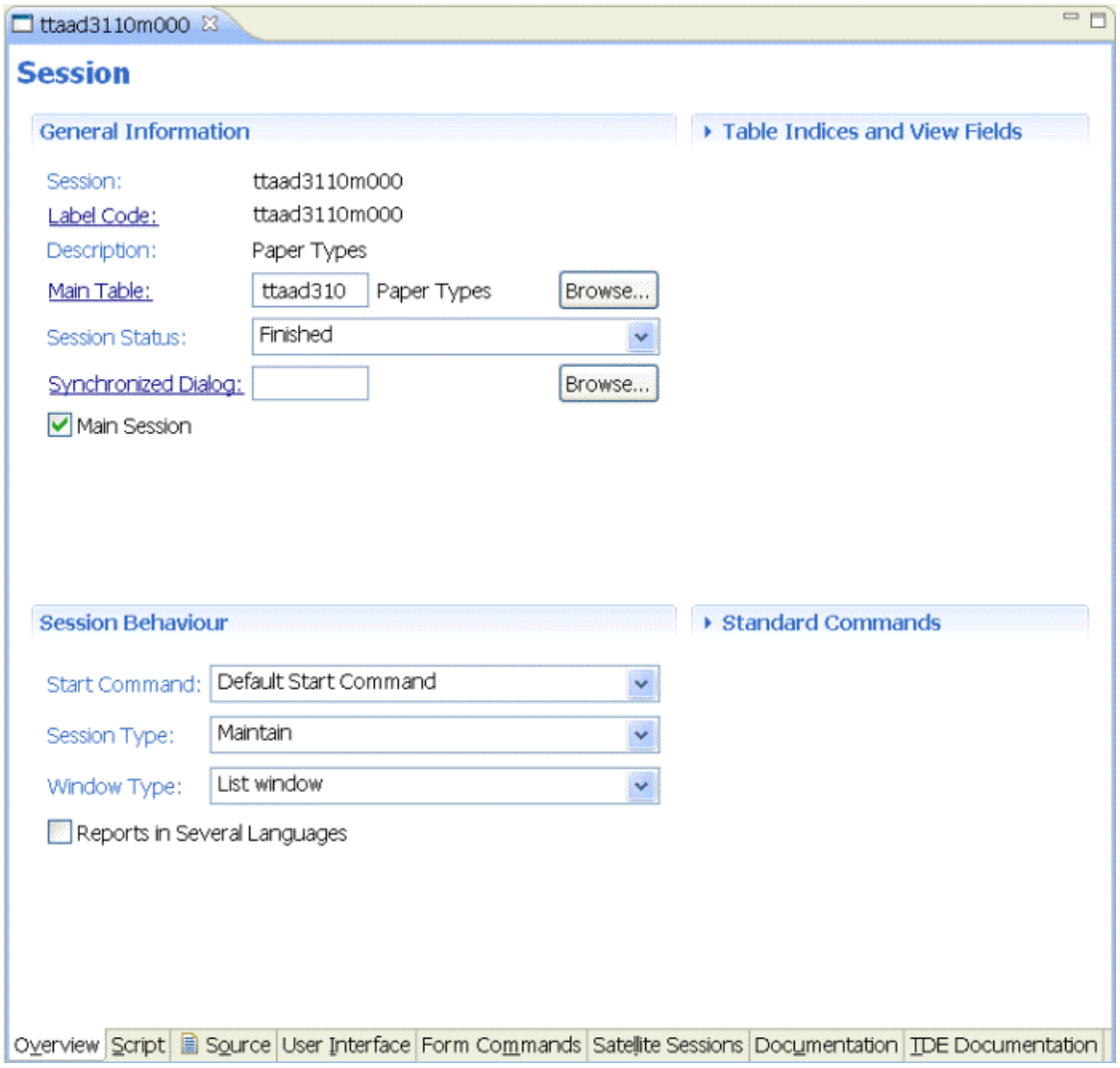
Add library code prefix to function names
If this check box is selected, the code of the library is added as a prefix to the names of all functions that are imported from the file or URL.

Multipage Editors

10

LN Studio contains various multipage editors that you can use to edit software components, such as sessions, menus, labels, libraries and messages.

See the following figure for an example:



These multipage editors are available:

- "Additional File Editor" on page 148
- "BFlow Editor" on page 150
- "Business Object Editor" on page 152
- "Domain Editor" on page 154
- "Function Editor" on page 166
- "Label Editor" on page 168
- "Library Editor" on page 174
- "Menu Editor" on page 178
- "Message Editor" on page 183
- "Question Editor" on page 187
- "Report Editor" on page 190
- "Session Editor" on page 205
- "Session Model Editor" on page 236
- "Table Editor" on page 238
- "Table InContext Model Editor" on page 257

For a detailed explanation of an editor, see the editor's online help.

Editor tabs

Each multipage editor consists of various tabs.

For example, the Session Editor consists of these tabs:

Session Editor tabs	
Tab	**Description**
Overview	Use this tab to specify the general properties for the session.
Script	Use this tab to specify the properties of the session's UI script. You can also link libraries to the session.
Source	Use this tab to view and edit the source of the session's script (if available). The tab consists of a session-specific Script Editor, which forms an integral part of the Session Editor.
User Interface	Use this tab to specify the form properties. You can also link reports and charts to the session, and specify text fields.
Form Commands	Use this tab to maintain the form commands for the session.
Satellite Sessions	If the session is a multi-main table (MMT) controller session, use this tab to maintain the MMT satellites for the session.

Session Editor tabs	
Tab	**Description**
Documentation	Use this tab to enter the session's release notes and technical documentation.
TDE Documentation	Use this tab to view an XML tree with the technical information on the session's Technical Data Entity.

The number of tabs can vary for the different editors. However, the **Overview** and **Documentation** tabs exist in all multipage editors. The **TDE Documentation** tab is displayed depending on the multipage editor preferences.

For details on the editor tabs, see the online help of the multipage editors.

Note: The **Source** tab is only displayed if the source code of the component is present on the LN server. If you edit a component, for which no source code is present, the **Source** tab is hidden.

By default, when you start a multipage editor, the editor opens the **Overview** tab. To change this default setting, click in the Activity Explorer view. As a result, the **Source** tab (if the source code is available) is directly opened when you start an editor. Click again to restore the default setting. See "Activity Explorer view" on page 297.

The editor displays error markers when you enter incorrect or insufficient information on an editor tab. When you save a component which contains errors, the component is only stored in your local workspace but is not saved to the server. In the Activity Explorer, the component has a dirty state ( ) decorator in the bottom right corner of the software component image. When you solve the error and save the component, the component is saved to the server again and the decorator disappears.

Preferences

To specify preference settings for the multipage editors:

1. Select **Window > Preferences**.
2. In the left pane of the Preferences window, select **Infor LN Studio Application > Multipage**.

   The Preferences - Multipage dialog box is displayed. In this dialog box, you can perform these actions:

   - Enable or disable the **TDE Documentation** tab for all multipage editors. This tab is primarily intended for support and troubleshooting purposes. Therefore it is recommended to display the tab only in case of problems.
   - Specify the directory where LN Studio stores the additional files that you edit in the Additional File Editor.

See "Setting user preferences" on page 36.

Starting a multipage editor

To start a multipage editor, double-click a software component in the Activity Explorer view, or right-click the component and select **Open**.

You can also start the editor from various other views, such as the Problems view and the Tasks view. For example:

- When you double-click a problem or task that is associated with a specific library, LN Studio starts the Library Editor and opens the library in the editor's **Source** tab.
- When you double-click a problem or task that is associated with a UI script, LN Studio opens the session to which the script belongs in the Session Editor, and opens the script in the editor's **Source** tab.
- When you double-click a problem or task that is associated with a specific line in a source, the editor opens the source on that line.

Additional File Editor

Introduction

Use this editor to define and maintain *additional files*.

Overview

Use this tab to define and maintain additional files.

General Information

Additional File
The name of the additional file, which includes the package code, the module code, the file identification code, and the filename extension.

For example: tctaxdetermine_tax.gif, or tcccpbl001blsrp0000.xsd.

Description
The description of the additional file. For example, the description of the tcccpbl001blsrp0000.xsd file is "tcccp001-List-Response".

Editable
If this check box is selected, you can view and edit the file. Click the Additional File hyper link to open the file with its default external editor, for example Notepad or Picture Manager.

If this check box is cleared, you cannot edit the file, and you cannot view the file in its default external editor. However, you can view the file in the **Preview** section of this tab.

Download File
Downloads the file to your client computer. Starts a dialog box where you must specify the file name and destination folder. After the download, the full pathname of the file is displayed behind the Open File link.

Upload File
Uploads an additional file to the LN server. Starts a dialog box where you must select the file to upload. After the upload, the full pathname of the file is displayed behind the Open File link.

You can only perform this action if the file is checked out.

Open File
Opens the file, depending on the file extension, in the corresponding default editor.

If the file is not checked out, you are prompted to check out the file. If you do not check out the file, you can open the file but any changes you make in the editor will be lost.

If no file location is displayed behind the Open File link, the file is downloaded before the editor starts.

Preview
This field shows a preview of the file.

For example:

- For a .xsd file, the XML content of the file is displayed.
- For an image file, such as a .gif file or a .jpg file, the corresponding image is displayed.
  - Click to enlarge the image.
  - Click to reduce the image.
  - Click to view the image at its actual size. For large images, only the upper left part of the image is displayed.
  - Click to resize the preview image so that it fits exactly in the preview area.

  These actions do not change the size of the image file, but only affect the preview image.

Note: To optimize performance, for text files, only the first couple of lines is displayed.

TDE Documentation

Use this tab to view an XML tree with the technical information on the additional file's Technical Data Entity.

This optional tab is displayed depending on the multipage editor preferences. See "Multipage Editors" on page 145.

BFlow Editor

Introduction

Use this editor to edit BFlow test scripts and test cases.

Before you can edit a BFlow test, you must create or import the test in LN Studio. See the *Infor LN BFlow Development Guide* and the LN Studio online help.

General

This tab shows general information and contains a comment section.

General Information

Name
The name of the test.

The name has this format:

`[package][module][filename].bflow` in case of test script

`[package][module][filename].bfcase` in case of test case

Description
The description of the test.

Domain
The domain of the test. Only applicable for test cases.

Owner
The owner of the test. Only applicable for test cases.

Company
The company to start the test in. Only applicable for test cases.

Logistic Company
The logistic company to start the test in. Only applicable for test cases.

Financial Company
The financial company to start the test in. Only applicable for test cases.

Comment

Comment
Shows the comment section of the test.

Variables

Use this tab to add, modify, or delete variables.

Variables

Name
The name of the variable.

Type
The variable type such as Long or String.

Domain
The LN domain name such as tcyesno.

Scope
The scope of the variable. For example, In or Internal.

Initialized
Indicates whether the variable is initialized.

Initial Value
The initial value of the variable.

Description
The description of the variable.

Buttons

Add
Adds a variable in the grid.

Up
Moves the selected variable one row up in the grid.

Down
Moves the selected variable one row down in the grid.

Remove
Removes the selected variable from the grid.

Actions

Use this tab to add, modify, or delete test actions.

The tab consists of these sections:

- **Overview**

  This section shows the test actions. If you select an action in the tree, the corresponding action details are displayed in the **Details** section.

  The tree supports cut/copy/paste and drag/drop functionality.

  You can expand or collapse all nodes of the tree by clicking the icons above the tree.

Click the show palette link above the tree to open the BFlow Palette view. This view shows all available actions. To add an action to the test, drag and drop an action from the palette, or use the shortcut menu of the tree.

- **Details**

  This section shows the details of the action selected in the **Overview** section.

  For details about the different action types, see the *Infor LN BFlow Development Guide* and the LN Studio online help.

XML

Use this tab to edit the XML source code of the test directly. The XML must be valid before you can save the test or switch to other tabs.

TDE Documentation

Use this tab to view an XML tree with the technical information on the test's Technical Data Entity.

This optional tab is displayed depending on the multipage editor preferences. See the *Infor LN Studio Application Development Guide* and the LN Studio online help.

Business Object Editor

Introduction

Use this editor to maintain *business objects*.

You can change the **Business Object Name**. The other fields are read-only.

Overview

Use this tab to maintain business objects.

General Information

Business Object

The code that identifies the business object.

Label Code

The name of the label that contains the business object's description.

To edit or view the label in the Label Editor, click the hyperlink.

Description

The business object's description.

Allowed values:

This field is read-only, because the description originates from a label. To edit this label, click **Label Code**.

Business Object Name

The name of the business object. By default this name is identical to the **Description**.

Business Object Document

The business object's BOD file.

To view this file in the Additional File Editor, click the hyperlink.

Interface Project

The business object's interface implementation.

To switch to the Integration perspective and open the implementation in the Interface Implementation editor, click the hyperlink.

Linked Libraries

This list contains the records of all libraries linked to the business object.

To sort the libraries by position, name, or description, click the corresponding column header. When you sort by position, the sort mode is always ascending. When you sort by name or description, click the involved column header to toggle between ascending and descending sort mode.

TDE Documentation

Use this tab to view an XML tree with the technical information on the business object's Technical Data Entity.

This optional tab is displayed depending on the multipage editor preferences.

See "Multipage Editors" on page 145.

Domain Editor

Introduction

Use this editor to maintain the properties of a *domain*.

A domain specifies various properties of a field or variable, such as:

- data type
- length
- value range
- (default) display format

Domains can be linked to table fields, form fields, and program variables.

Overview

Use this tab to specify the general properties of a domain.

General Information

Domain
The unique name of the *domain*.

A domain is identified by the unique combination of *package* code and domain code.

Allowed values:

This field is read-only.

You can only specify a domain name when you create a new domain. The domain name is an ASCII text and must start with the code of the package for which the domain is used. After you created a new domain, you cannot change its name.

A good practice is to always include the module code in the domain name. When you fill in a domain name, we recommend the following format:

`<package code><module code>.<domain code>`

For example, `tdsls.hdst` is the name of the Header Status domain of the Sales Control module (sls) of the Distribution package (td).

Description
The description of the *domain*.

Allowed values:

An ASCII text with a maximum length of 60 characters.

Data Type
The data type of the domain. This is the internal representation of data.

For example:

- string
- long
- double
- date
- UTC Date Time

Allowed values:

Byte	Whole number for which physically 1 byte is reserved.
Integer	Whole number for which physically 2 bytes are reserved.
Long	Whole number for which physically 4 bytes are reserved.
Float	A data type for floating-point numbers, that is, any number containing a decimal point. Physically, 8 bytes are reserved for each float data type.
Double	Whole number for which physically 8 bytes are reserved.
String	A data structure that contains a number of ASCII characters that represent readable text. Each character consists of a single byte.
Enumerated	A data type that consists of a single value from of a list of predefined text values (up to 255). You can define enumerate values with name and language-dependent descriptions on the **Enum/Set Constants** tab of the Domain editor. Example: A field can contain values such as 'yes', 'no' or 'maybe', or color values such as 'red', 'white', 'blue', and so on.
Set	A data type that consists of a set of predefined text values (up to 32). You can maintain the set name and its language-dependent descriptions on the **Enum/Set Constants** tab of the Domain editor. Example: The display feature set, which can contain the values 'bold', 'blinking', 'reverse', but also combinations of these, such as 'bold' and 'blinking'.
Date	A data type that represents a date as string in the form YYYYMMDD, where Y stands for the year, M for the month and D for the day of the month. The date can optionally be appended by the time, which has the following format: THHMMSS, where T stands for time, H for hours, M for the minutes and S for seconds. Example: Date '19990816T040000' denotes eight o' clock in the morning, August 16th, 1999.
Text	Data type used for fields that refer to a text number in the text database. Example: To make it possible to link a text to an item in an item table, you must add a field with a domain of the text data type in the item table definition.

Multibyte String	A data structure that contains a number of multibyte characters that represent readable text. Each character consists of one or more bytes.
UTC Date Time	A time/date format. LN stores UTC Date Time as the number of seconds (a Long value) after the beginning of the year 1970 (00:00:00 GMT, January 1th).

Note: The data type you selected for the domain will largely determine which fields you can fill in for this domain and how you can specify these fields.

Runtime

Convert Domain to Runtime

Click the Convert hyperlink to start the Convert to Runtime Data Dictionary (ttadv5215m000) session. Use this session to convert domains and tables to the runtime data dictionary.

This option is only enabled for checked in components.

Note: In the Convert to Runtime Data Dictionary (ttadv5215m000) session, select the package combination that contains the *Project VRC* of your current *Software Project*. The session converts domains and tables in all *Activities* linked to the software project. So, the session does not only convert domains in your own *Activity VRC*, but converts changed domains in activity VRCs of other developers as well.

Read the session's online help before you start the conversion.

Formatting

Length

The length in characters, if the data type of the domain is a (multi-byte) string, or the default display length for data types UTC Date Time, Date, Enumerated and Set. For the latter two data types, the length defines the maximum number of characters of the descriptions of the Enumerated or Set data type. Any description with more characters will simply be cut off at the specified length.

Allowed values:

A whole number in the range 1 – 999.

Default value:

Data type	Default value
Date	8
UTC Date/Time	15
Other	0
Not Applicable	Not relevant for this data type.

Note: You can use this field for the following data types:

- String

- Multibyte String
- Enumerated
- Set
- Date
- UTC Date/Time

The field does not apply to other data types.

Alignment
The alignment method for various components:

- (multi-byte) strings
- enumerated values
- sets
- dates

Allowed values:

Data type	Default value
Not	No text alignment.
Left	Aligns text to the left.
Right	Aligns text to the right.
Centered	Centers text.
Not Applicable	Not relevant for this data type.

Default value:
Not

Note: You can use this field for the following data types:

- String
- Multibyte String
- Date
- UTC Date/Time

For other data types, the field is automatically set to value ‘Not Applicable’.

Conversion
Conversion method for *strings*. You can convert text strings into upper or lower case.
Allowed values:

Data type	Default value
Not	No text conversion.
Lower Case	Converts entire text into lower case.

Data type	Default value
Upper Case	Converts entire text into upper case.

Default value:

Not

Note: Only use this field for String and Multibyte String data types. For other data types, the field is automatically set to value ‘Not Applicable’.

Internal Format
This field indicates how a number, entered in a form input field with the string data type, must be internally formatted.

For example, if number 37 must be stored as 00000037, use the format 99999999.

Note: Only use this field for String data types. The field does not apply to other data types.

Allowed values:

A formatting text string with a maximum length of 30 characters. Allowed characters are restricted to the following set:

- -
- +
- *
- Z
- 9
- V
- ,
- :

Display Format
The display format used in forms and reports to display specific data.

Allowed values:

The value for the display format must be less than 30 characters. To specify a pre-defined display format from the server, use the browse button and select a value from the select dialog box.

Data type	Format description
Byte, Integer, Long	A string of 'Z' characters followed by '9' characters. The number of characters in the string indicates the number of positions used to display the number. The '9' sign reserves a position for a digit. The sign 0 will be displayed if there is no significant digit on this position. The 'Z' sign also reserves a position for a digit, but displays a blank instead of a zero for an empty position.
Float, Double	In the format of fractional numbers, the decimal sign is denoted by the letters 'VD'. The '9' sign reserves a position for a digit. In runtime, the sign 0 will be displayed if there is no significant digit on this position. To the right of the decimal point, the 'Z'

Data type	Format description
	sign also reserves a position for a digit, but displays a blank instead of a zero if the involved digit is not significant. To the right of the decimal sign ('VD'), .'9' signs are used to indicate the number of decimal places. When the data type is a double, you can use a generic format. To use a generic format, the display format value must contain '%A'. The string that follows %A must be an existing generic format on the back end, for example '%A001'. For example, if you want to display the number '123.45678' as '....000123.45' (points denote blanks), you must specify display format 'ZZZZ999999VD99'.
Date	The format for Date data types must contain '%D' and the string that follows '%D' must exist as a date format on the server, for example '%D001'.
UTC Date/Time	The format for UTC Date/Time data types must contain '%u' and '%U'. The string that follows '%u' (until blank character) must exist as a date format on the server. The string that follows '%U' must exist as a time format on the server, for example '%u001 %U002'.
Text	A string of 'Z' characters followed by '9' characters. The number of characters in the string indicates the number of positions used to display the text.

Note: You cannot use this field for data types Enumerated and Set.

Numeric Data

Digits Before Decimal
For numeric data types, the maximum number of digits before the decimal sign. This implies the maximum number of digits for long, integer and byte data types.

Allowed values:

The value for this field is a whole positive number less than 100. For byte, integer and long data types, the following maximum values apply.

Data Type	Maximum value
Byte	3
Integer	5
Long	10

Default value:

Data type	Default value
Byte	2
Integer	4
Long	6
Float	1

Data type	Default value
Double	6

If the **Display Format** of this domain contains '%A', the default value of a double is 15.

Note: Only use this field for numeric data types:

- Byte
- Integer
- Long
- Float
- Double

For non-numeric data types, the field is automatically set to value 0.

Digits After Decimal
The maximum number of digits after the decimal sign.

Allowed values:

The value for this field is a whole positive number less than 100.

Default value:

If the **Display Format** of this domain contains '%A', the default value for a double is 4. For a float data type, the default value is 0.

Note: Only use this field for Float and Double data types. For whole numbers and non-numeric data types, the field is automatically set to value 0.

Divide Factor
The factor by which the entry in a field is divided.

Divide Factor can be useful for fields containing amounts.

Example:

The **Divide factor** amounts to 100. When a user enters 25009 in the field for which this factor is set, LN divides 25009 by 100 and displays 250.09.

Allowed values:

A whole number (max. 9 digits).

Default value:

1

Note: Only use this field for Float and Double data types. The field does not apply to other data types.

Rounding Method
The way in which a field with the float or double data type can be rounded.

Rounding takes place at the last digit as specified by the number of digits after the decimal point.

Example:

In the following examples, the number of digits after the decimal sign is one.

Entered by user	Round off	Round off upwards	Round off downwards
1.51	1.5	1.6	1.5
2.78	2.8	2.8	2.7
1.55	1.6	1.6	1.5

Allowed values:

- **Round off**
- **Round off upwards**
- **Round off downwards**
- **Not Applicable**

For an explanation of these rounding methods, refer to the previous example.

Default value:

Round off

Note: Only use this field for Float and Double data types. The field does not apply to other data types.

Validation

Illegal Characters

The illegal characters that cannot be entered in a form input field.

Allowed values:

All characters are allowed, with the exception that the characters ' " ^ \ must be preceded by a backslash [\], for example \\ and \". Maximum length is 30.

Note: You cannot use this field for the following data types:

- Enumerated
- Set
- Date
- UTC Date/Time
- Text

The field can be used for all other data types.

If you define some characters as illegal, all other characters are legal, and vice-versa. Therefore, you cannot use this field if a value for **Legal Characters** is filled in.

Legal Characters

The legal characters that can be entered in a form input field. If you do not specify any legal characters, all characters are legal.

Example:

If only numbers can be entered in a field with string data type, the legal characters are specified as: `1234567890.`

Allowed values:

All characters are allowed, with the exception that the characters ' " ^ \ must be preceded by a backslash [\], for example \\ and \". Maximum length is 30.

Note: You cannot use this field for the following data types:

- Enumerated
- Set
- Date
- UTC Date/Time
- Text

The field can be used for all other data types.

If you define some characters as legal, all other characters are illegal, and vice-versa. Therefore, you cannot use this field if a value for **Illegal Characters** is filled in.

Range
The range used to check the validity of the input. If you enter a value in a form input field outside the specified range, LN produces a range message and will ask for input again.

For simple validation checks, you can use the following:

- The data type
- Legal and illegal characters
- The length of the field
- The number of digits before and after the decimal point

For more complicated validation checks, you need the range attribute.

Examples of restrictive ranges:

A string starting with a capital	`$$ in ^[A-Z].*$ (regular expression)`

Allowed values:

The range can be an expression with evaluated value true (valid value) or false (invalid value) (true <> 0, false = 0). Other variables can also be used in the expression. To test the current value in the expression, use 2 dollar signs: $$

The following operators can be used (listed in order of decreasing precedence):

Operator	Description	Example	Operator	Description
( )	expression in brackets	2*(3+5)	=	equal to
IN	range test	$$ IN [10,20]	<>	not equal to
-	unary minus	-10	<=	equal to or less than

Operator	Description	Example	Operator	Description
^	raise to power	2^3 = 8	>=	greater than or equal to
*	multiply	2*3 = 6	<	less than
/	divide	6/2 = 3	>	greater than
\	remainder	3\2 = 1		
&	string concate-nation	"A"&"B" = "AB"	not	logical negation
+	addition	2+3 = 5	and	logical and
-	subtracting	2-3 = -1	or	logical or

Note: You cannot use this field for data type Text. The field can be used for all other data types.

Range Message
The name of the message, including package code, which is issued if an entered value falls outside the specified range.

Allowed values:

You can use the browse button and select a message name from the select dialog box. This dialog box contains all messages available in the package of the domain. You can also enter an existing message name.

Range Message Description
The actual message as shown when activated.

Note: Range Message Description is an information field. You cannot change the message, because the field is read-only. You can create messages in the Message Editor.

Enum - Set Data

This tab only applies to domains with data type Enumerated or Set.

Use this tab to define the constants that belong to a domain of data type Enumerate, or Set, with language-dependent descriptions.

To define a constant:

1. Enter the constant number for internal usage.
2. Enter the name of the constant used in program scripts and defaults in form fields.
3. Enter the language-dependent description that a form field shows at runtime.

This tab contains a table with all the constants used in an Enumerated or Set domain. The constants are displayed as records with the following arguments:

- **Constant**
- **Constant Name**
- **Description**
- **Label**

In the **Enum or Set Values** table, you can add and remove constant records.

- To add a constant record to a domain, click **Add** and fill in the required fields. Specify a code and a name. Alternatively, right-click in the table and, on the shortcut menu, select **Add**.
- To remove a constant record, right-click the record and, on the shortcut menu, select **Remove**.

Enum or Set Values

Constant
The constant number that represents the internal enumerated field value. This number determines the sequence in which the enum descriptions will be displayed at runtime.

If a table index includes/consists of an enumerate field, the constant determines the sequence of records according to the index.

You can also use the constant number for greater than and less than expressions in scripts. For example, domain `tryesno` has value `yes`, with constant number 1, and value `no`, with constant number 2. For this domain, the expression `tryesno.no > tryesno.yes` is true.

Allowed values:

Whole number in the range 1 -255.

Default value:

The highest present constant number raised by 1.

Constant Name
The constant name combined with the package code and domain code. This is used to refer to an enumerated constant in scripts or expressions on forms and reports.

Example:

Consider package code `ti` (Manufacturing), domain code `yesno` and constant name `yes`. To refer to this constant in a form field with domain code `tiyesno`, use the expression `tiyesno.yes`.

Allowed values:

A text string that starts with an alphabetic character. Spaces are only allowed at the end of the string.

Description
The content description of the label. This description is used as a constant value of enumerated or set data types.

Allowed values:

This field is read-only for existing constant records, because the description is a property of the **Label**. You can modify the description text in the label editor.

You can specify a constant description in the domain editor when you add a new constant record. This description is an ASCII text with a maximum label length of 70 characters.

Label
The label name for the **Description** of the enumerated or set constant.

When hovering above the label name, the domain editor shows a hyperlink. Use this hyperlink to edit the label with the enumerate description in the label editor.

Allowed values:

This field is read-only and is automatically generated when you save the domain.

The generated label name is a concatenation of package code, module code, domain code and constant number. For example:

```
ttadv.cmpt0001
```

Note: The label is generated when you save the domain and can be modified in the label editor.

Step Size
A number used to automatically generate subsequent constant values.

For example, if the first constant value is 10, and the step size is 10, the second and third constant number will be 20 and 30. This step size enables you to insert in-between constant values later.

Default value:

10

Documentation

Use this tab to enter the domain's release notes and technical documentation.

Release Notes
Use this field to specify the release notes of the domain.

Technical Documentation
Use this field to specify the technical documentation of the domain.

TDE Documentation

Use this tab to view an XML tree with the technical information on the domain's Technical Data Entity.

This optional tab is displayed depending on the multipage editor preferences. See "Multipage Editors" on page 145.

Function Editor

Introduction

Use this editor to maintain the properties of a *function*.

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. With the #include statement, you can include a function in a *program script*. It can be useful to define a function and use it in more than one program script.

Overview

Use this tab to specify the properties of a function.

General Information

Function
The unique *function* name, which identifies the function.

A function name consists of a combination of these codes:

- package
- module
- function

The function name forms the identifying attribute of the function and is used in a program script as an include statement.

Allowed values:

This field is read-only.

You can only specify a function name when you create a new function. Afterwards, you cannot change this name.

When you create a new function, enter the full name of the function, which is a combination of the package code, the module code and the function code, for example: `ttaad0001`.

Illegal characters are: $ \ / ~ &

Example:

The Application Administration module of the LN Tools package includes the function Convert User to Runtime DD.

Component	Code	Description
Package	tt	LN Tools
Module	aad	Application Administration

Component	Code	Description
Function	0001	Convert User to Runtime DD

The name to refer to this function is: `ttaad0001`

If you want to include a function in a program script, use the following statement:

```
#include "i<entire code>"
```

Therefore, in this example use the following:

```
#include "ittaad0001"
```

Description
The description of the function.

Allowed values:

An ASCII text with a maximum length of 60 characters.

Source

Use this tab to edit the function's source code. The tab consists of a script editor, which forms an integral part of the function editor.

For more information, refer to "Script Editor" on page 263.

Documentation

Use this tab to enter the function's release notes and technical documentation.

Release Notes
Use this field to specify the release notes of the function.

Technical Documentation
Use this field to specify the technical documentation of the function.

TDE Documentation

Use this tab to view an XML tree with the technical information on the function's Technical Data Entity.

This optional tab is displayed depending on the multipage editor preferences. See "Multipage Editors" on page 145.

Label Editor

Label Editor

Introduction

Use this editor to maintain the properties of a *label*.

Labels are used instead of language-dependent text in these LN components:

- Forms
- Reports
- Domains
- Menus

Labels can also be used in LN Workbenches.

A label consists of a name and a content description. Use the label name as a reference to the label description. The description of a label can differ by language, but the label name remains the same for all languages.

You can specify a label description for each label code. This is the label text you can edit and translate. You can define different variants for a label. Therefore, a label can have a different description in a different context or with a different length or height.

The label editor automatically calculates the length and height of a label description. The length is the number of characters of the description. The height is the number of lines of the description.

Note: The **Languages** property of the application, assigned to the current software development project, determines the primary language of a label. It determines whether you can specify secondary languages and which languages are used as secondary languages. For more information on the application properties, refer to the "Create a new Application" on page 132 dialog box.

Overview

Use this tab to view the label name and to maintain all label variants in the primary language, such as English.

General Information

Label
The name of the label used as identification code.

Allowed values:

This field is read-only.

You can only specify the label name when you create a new label. The name must start with the code of the package for which the label is defined.

After a label is created, you cannot change its name.

Description
The description of the label, which corresponds with the longest variant description in the primary language.

Allowed values:

This field is read-only.

The value of this field corresponds with the value of the description field of the primary language variant with the longest description. This field is a read-only copy of this variant description.

All Variants in (primary language)

All Variants in
The **All Variants in** list contains all label variants in the primary language. Only the variant descriptions are displayed.

The **Variant Properties** field group displays all properties of the label variant you selected in this list.

You can add and remove label variants in the **All Variants in** list:

- To add a label variant, click **Add** and specify a **Context**. Then, enter the **Description** for the new variant in the **Variant Properties** field group.
- To remove a label variant, select a variant and click **Remove** on the variant shortcut menu.

Note: The name of this field includes the primary language of the label. For example, if the primary language is English, the name of the field will read the following: **All Variants in English**. The primary language is determined by the **Primary Language** property of the application for the current software development project.

Allowed values:

A label must have at least one variant. Labels without a variant cannot be saved.

When you create a new variant, specify the **Context** and **Description** of the variant. When you specify the variant **Description**, you cannot enter a text longer than 70 characters or a text that ends with a blank space.

When you enter the variant **Context**, select a value from a drop-down list. You can choose one of the following context values:

- General use
- Form Commands
- Cascading Item on Button

For more information on the **Context** and **Description** fields of a variant, refer to the **Variant Properties** field group.

Note: When you create a new label in the Create New Label wizard, you can specify an `External Use` context. These `External Use` labels cannot possess multiple variants. Therefore, the **Add** button is disabled for these labels.

Variant Properties

Context
The context of the label variant selected in the **All Variants in** field. This indicates how the variant is used in forms, reports, menus, and so on.

Allowed values:

Note: When you add a new variant to a label, you can choose between three contexts:

- `General use`
- `Form Commands`
- `Cascading Item on Button`

However, the label variants not created with the label editor, such as session description labels, can show many other context values.

These other context values are used for labels that are generated automatically.

For example:

- When you create a new enumeration for an Enumerated domain, LN Studio generates a label with the `Enumerates` context.
- When you create a new session, table or report, LN Studio generates a label with the `Session, Table or Report Description` context.

Example:

When you create a session with session code `tezzzrx` and session description `MyNewSession`, LN Studio generates a label called `tezzzrx001` with label description `MyNewSession`.

Description
The description of the label variant selected in the **All Variants in** field.

This is the text that appears where this label variant is used in the following:

- forms
- reports
- menus

Allowed values:

An ASCII text with a maximum length of 70 characters.

Use a % sign to indicate the line break. If the description includes a line break, the height of the label variant is automatically adjusted to a value greater than one. To include the % sign in the description, type %%.

The maximum number of lines (height) is 3.

Example:

Description	Result	Height	Length
Item Code	Item Code	1	9
Item%Code	Item	2	4

Description	Result	Height	Length
	Code		
Percentage [%%]	Percentage [%]	1	14
Perc.%[%%]	Perc.	2	5
	[%]		

Note: The description of a label variant automatically sets the length, height and search argument of this label variant:

- The variant **Length** corresponds with the number of characters in the description's longest line of text.
- The variant **Height** corresponds with the number of text lines in the description.
- The variant **Search Argument** corresponds with the first 10 characters of the description.

Length
The length of the label variant selected in the **All Variants in** field. This is the maximum number of characters on a text line of the variant description.

Allowed values:

This field is read-only.

The length value is generated by the label editor. The length is automatically adjusted to a changed label variant description. For examples of how the length of a label variant is derived from the description, refer to the **Description** field.

Height
The height of the label variant selected in the **All Variants in** field. This is the maximum number of text lines in the variant description.

Allowed values:

This field is read-only.

The height value is generated by the label editor and corresponds with the number of text lines in the variant's **Description**. The height is the number of % signs in the description text raised by one, with an exception for the "%%" string, which you use to include the % sign in the label text displayed to the user.

The height is automatically adjusted to a changed variant description. For examples of how the height of a label variant is derived from the description, refer to the **Description** field.

Search Argument
The search argument of the label variant selected in the **All Variants in** field.

Allowed values:

This field is read-only.

The search argument is generated by the label editor and corresponds with first 10 characters of the variant's **Description**, including the % signs. The search argument is automatically adjusted to a changed variant description.

Note: The label editor changes lowercase characters to uppercase characters.

Expired
If this check box is selected, the label variant selected in the **All Variants in** field has expired.

Expired label variants cannot be used in the release (VRC) of the application of the current development project or any releases derived from it.

When all label variants are expired, the label becomes expired.

Note: If only one variant exists, this check box is disabled. In that case, you can expire the label in the Activity Explorer.

Translation

Use this tab to view and maintain all label variants in the secondary languages, which represent translations of the primary language label variants on the **Overview** tab.

The **Translation** tab is only available if secondary languages are defined for the application. The properties of the application of the current software development project determine whether a label has secondary languages and which secondary languages are used. For more information on the application properties, refer to the "Create a new Application" on page 132 dialog box.

All Variants in (primary language)

Context / Description
List of all label variants in the primary language. The list is identical to the **All Variants in (primary language)** list on the **Overview** tab.

To copy a label variant from the primary language to a new label variant in a secondary language:

1. In the **Software Language** field, select a secondary language.
2. In the **All Variants in (primary language)** list, select a label.
3. Click **Copy**. A new label appears in the **All Variants in Software Language** list.

You can now change the properties of the new variant in the **Variant Properties** field group.

Allowed values:

This field is read-only.

All Variants in Software Language

Software Language
The secondary language of the label variants displayed in the **All Variants in Software Language** list.

Allowed values:

Select a secondary language from the drop-down list. Note that the available secondary languages are determined by the settings of the application of the current software development project.

Context / Description
A list of all label variants in the selected secondary language.

The properties of the label variant you select in this field are displayed in the **Variant Properties** field group on this tab. Some properties are read-only, while some are editable.

In the list, you can add and remove label variants, as described for the **All Variants in** field on the **Overview** tab. You can copy a label variant in the primary language as a new label variant in the secondary language. To do this, use the **Copy** command of the **All Variants in** field on this **Translation** tab.

Variant Properties

Context / Description / Length / etc.
See **Variant Properties** on the **Overview** tab.

Documentation

Use this tab to enter documentation on the label.

Technical Documentation
Use this field to specify the technical documentation of the label.

TDE Documentation

Use this tab to view an XML tree with the technical information on the label's Technical Data Entity.

This optional tab is displayed depending on the multipage editor preferences. See "Multipage Editors" on page 145.

Details

Using labels

Using *labels* instead of plain text has these advantages:

- Improved standardization: a single label is used by all software components
- Easier software maintenance: the layout of forms, reports, and menus becomes language-independent
- Easy and fast translation: the label description is only translated once
- Enhanced customization facilities: changes in the central label file have an effect on all the packages
- The alignment of a label (left or right) on multi-occurrence forms and reports is linked to the alignment of the associated table field

For example, using labels makes it possible to display and print the text in the correct language on these components:

- Forms
- Reports
- Menus

For each *label*, specify this information:

- The label code, which identifies the label.
- The label variant descriptions which can be translated into different languages.

You can specify multiple descriptions with different lengths or heights for a label.

Library Editor

Introduction

Use this editor to maintain the properties of a *library*.

A library is a collection of files, programs or subroutines that is used to carry out specific tasks for other libraries, program scripts or sessions of LN applications. A *dynamic link library* (DLL) can be linked as a function call to a program or session at runtime.

LN distinguishes three types of libraries, all of which are DLL libraries:

- General Library DLL
- Integration DLL
- Business Object Layer DLL

Overview

Use this tab to specify the properties of a library.

General Information

Library
The unique name of the library.
It consists of a combination of the following:

- package code
- module code

- library code

Allowed values:

An ASCII text with a maximum length of 14 characters that conforms to the following format:

```
<package code><module code><library code>
```

Example:

Package Code:	tf	LN Financials
Module Code :	acf	Accounts Payable
Script Code :	dll1250	Receive Purchase Invoices

The total identification of the library is: tfacfdll1250.

Description
The description of the library, such as "Validate Financial Calendar" or "Calculate Tax."

Allowed values:

An ASCII text with a maximum length of 60 characters.

Library Type
The library type. To enter a value, select a library type from the drop-down list.

Allowed values:

The following library types are used:

General Library	General DLL - Dynamic Link Library
Integration DLL	Dynamic Link Library that integrates between packages.

Compile Flags
The compile flag.

In (3GL and 4GL) sources you can use compile flags. This is in order to compile part of the source depending on certain conditions and is called conditional compiling. For more information on conditional compiling and other compile options, refer to sections "Preprocessor" and "Compiler" of the *Infor ES Programmer's Guide*.

Allowed values:

Infor 3GL or Infor 4GL source code with a maximum length of 200 characters.

Example:

```
 | Compile Flag: -DMYTEST | Source ... #if MYTEST message(Some debug
information) #endif ...
```

You can specify several compile flags separated by a space " ". For example:

```
Compile Flags: -DMYTEST -DVERSION_1 -DVERSION3.0
```

Note: The preprocessor only works during compilation of a 3GL source. This is because the standard generator std_gen6.2 does not have a preprocessor pass. Therefore, you cannot use 4GL events in a #if, #ifdef or #ifndef.

Tools Interface Version
The Tools Interface Version (TIV) number of the library.

This is used to guarantee backward compatibility and is related to a particular version of the porting set, the Enterprise Server and the 4GL Tools. For example, the TIV number for Enterprise Server 7, delivered together with LN 6.1 FP1, is 1010. When you insert a new library, the TIV is automatically set to the highest supported TIV number.

Setting a TIV number for each object can result in the behavior of these objects being different. This can occur for incompatible features executed with different versions of the porting set or the LN 4GL Engine.

For example, the "set.synchronized.dialog" function was changed in TIV 1000. Therefore objects in which this function is used, show different behavior for different TIVs:

- If the object's TIV is less than 1000, a synchronized dialog is started when a record is inserted or duplicated in an editable grid.
- If the object's TIV is 1000, no synchronized dialog is started when a record is inserted or duplicated: the new record is entered directly in the grid.

For details about the TIV, refer to the *Infor ES Programmer's Guide*.

Allowed values:

Numeric value (max. 9999).

Keywords

Search Key 1
The first search key.

Allowed values:

An ASCII text with a maximum length of 20 characters.

Search Key 2
The second search key.

Allowed values:

An ASCII text with a maximum length of 20 characters.

Search Key 3
The third search key.

Allowed values:

An ASCII text with a maximum length of 20 characters.

Linked Libraries
This list contains the records of all libraries linked to the current library.

Sorting

- To sort the libraries by position, name, or description, click the corresponding column header.
- When you sort by position, the sort mode is always ascending. When you sort by name or description, click the involved column header to toggle between ascending and descending sort mode.
- When you sort by name or description, the **Up** and **Down** commands are disabled.

Add / remove libraries, change linking order
You can only perform the following actions if you sort the libraries by position.

- To add a library, click **Add**. Then, press CTRL+SPACE in the **Name** field and select a library from the select dialog box.
- To remove a library, right-click a library and, on the shortcut menu, select **Remove**.
- To change the linking order of the libraries, move them up or down the list.

Note: To edit a library, click the icon behind the library name.

Pos
The library sequence number (Position). This field indicates the linking order of the libraries.

This field is read-only. It is filled automatically when you link a library.

Name
The name of the library.

Allowed values:

To specify this attribute, press CTRL+SPACE and select a library from the select dialog box.

Description
The description of the library.

Allowed values:

This field is read-only. LN Studio automatically fills in the library description after you specified the **Name** of the library.

License

Product ID Source
The product ID for which a license is needed to compile the object. If the field is empty, no license is needed to compile the object.

To enter a product ID, click **Browse** and select an ID from the select dialog box.

Product ID Object
The product ID for which a license is needed to run the object. If the field is empty, no license is needed to run the object.

To enter a product ID, click **Browse** and select an ID from the select dialog box.

Source

Use this tab to edit the library's source code. The tab consists of a script editor, which forms an integral part of the library editor.

For more information, refer to "Script Editor" on page 263.

Documentation

Use this tab to enter the library's release notes and technical documentation.

Release Notes
Use this field to specify the release notes of the library.

Technical Documentation
Use this field to specify the technical documentation of the library.

TDE Documentation

Use this tab to view an XML tree with the technical information on the library's Technical Data Entity.

This optional tab is displayed depending on the multipage editor preferences. See "Multipage Editors" on page 145.

Menu Editor

Introduction

Use the Menu Editor to define and maintain *menus*.

An LN menu is a list of options from which a user can perform a desired action, such as starting a session, another menu, and a query.

Overview

Use this tab to specify the following:

General Information	The general properties of a menu, such as the name and description.
Menu entries	These are the options a user can start from the menu. The **All Menu Entries** list contains all options the user can start from the menu, such as sessions, other menus, and queries. In the **All Menu Entries** list, you can add and remove menu entries: • To add a menu entry, click **Add** and specify the properties for the new entry in the grid. • To remove an entry, right-click the entry and, on the shortcut menu, select **Remove**. The order of the entry records in the table corresponds with the order of the entries on the menu. To determine the menu entry sequence, click **Up** or **Down** to move the entry records up or down the table.

General Information

Menu
The name of the menu.

This includes the following:

- package code
- module code
- menu identification code

Allowed values:

This field is read-only.

You can only specify the name of a menu when you create a new menu. After a menu is created, you cannot modify its name.

Label Name
The code of the label that contains the menu description.

To modify the description, click the hyperlink to open the displayed label in the Label Editor.

Allowed values:

This field is read-only. You can only set the label name when you create a new menu. You cannot change this name afterwards.

Description
The menu description, as displayed in the user interface.

Allowed values:

This field is read-only, because the description originates from a label. To edit this label, click **Label Name**.

All Menu Entries

Type
Type of menu entry. This indicates the type of action activated by the menu entry.

Allowed values:

Select one of these entry types:

- Session
- Menu
- Shell Program
- Query

Code
Code of the menu entry. This is the code of the session, menu, shell program, or query activated when you select this menu option. The type of code used depends on the value specified for the **Type** field.

Note:

- If the menu entry type is Session, Menu, or Query, press CTRL+SPACE and select a component from the select dialog box. Click the icon behind the code to open the component in the corresponding editor.
- If the menu entry type is Shell Program, manually enter the program name.

Allowed values:

An ASCII text with a maximum length determined by the value of the **Entry Type** field.

Entry type	Text length
Session	13 characters.
Menu	13 characters.
Shell Program	15 characters.
Query	9 characters.

Description Linked
If you select a session, menu, or query, the corresponding description is automatically placed in the **Description** field.

If this check box is selected, the description is read-only.

If this check box is cleared, you can modify the description.

Note: You cannot select this check box if the menu entry type is Shell Program. You must manually enter the program description in the **Description** field.

Description
The description of the menu entry, as displayed in the menu.

This **Description** field only applies if the **Description Linked** check box is cleared and the **Label Code** field is not filled in. The field is disabled if these conditions are not satisfied.

Note: If you enter the description manually, LN Studio automatically stores the description in a label. The code of this label is displayed in the **Label** field.

Allowed values:

ASCII text of less than 60 characters.

Label
The code of the label that contains the menu entry description. Click the icon behind the label code to open the label in the Label Editor.

To select a label, press CTRL+SPACE and select a label from the select dialog box.

The **Label** field only applies if the **Description Linked** check box is cleared.

Allowed values:

A code of a label that exists on the LN server.

Process Info
This field contains information that will be passed to the program activated by the menu.

If the program is a session, this information can be retrieved in the session with the procesinfo variable. If the program is a shell program, the information is used as the argument of the program.

This field only applies to menu entries of type `Session` or `Shell Program`. This field is disabled for all other menu entry types.

Allowed values:

ASCII text of less than 50 characters.

Active
If this check box is selected, the menu entry is active. Therefore, the menu entry is visible on the menu.

If this check box is cleared, the menu entry is disabled. Therefore, the menu entry is not visible on the menu.

Filter Library
The library/dal to determine whether the menu item, session, or complete menu tree, must be hidden based on the parameter settings of the company of the user. Zooming starts a session that only shows libraries/dals. To open the filter library in the Library Editor, click the icon behind the library name.

Note: This setting has only effect for users that use LN UI. This setting is only available if it is supported by the LN server.

Filter Function
The name of the function in the Filter Library that must be called, without the () at the end of the function. This function has no parameters and must return true or false.

If the function returns false when called without parameters, the menu entry is not displayed in LN UI . In this way you can filter sessions that depend on parameter settings.

Note: This setting has only effect for users that use LN UI. This setting is only available if it is supported by the LN server.

Hidden
If this check box is selected, the menu item is not displayed in the menu at runtime. However, in LN UI users can make the menu visible through the **Personalize Menu** command. Administrators can centrally maintain these personalizations through the Overview Menu Personalizations (ttadv9505m000) session.

Note: This setting has only effect for users that use LN UI. This setting is only available if it is supported by the LN server.

Preview

This tab displays a preview of the menu.

Documentation

Use this tab to enter the menu's release notes and technical documentation.

Release Notes
Use this field to specify the release notes of the menu.

Technical Documentation
Use this field to specify the technical documentation of the menu.

TDE Documentation

Use this tab to view an XML tree with the technical information on the menu's Technical Data Entity.

This optional tab is displayed depending on the multipage editor preferences. See "Multipage Editors" on page 145.

Message Editor

Message Editor

Introduction

Use this editor to maintain the *messages* displayed to the user.

LN messages are language-dependent software components used to show situation-dependent information to the user. They allow you to customize dialog messages.

Message codes can be included in program scripts and can be linked to these components:

- Domains
- Tables
- Table fields
- Form fields
- And so on

For each message, you can specify this information:

- The unique message code. This identifies the message across all languages.
- The message type, such as "Warning", "Information", or "Critical".
- The message. This is the message text which you can edit and translate. It can contain codes that are substituted when the message is displayed. For details on substitution symbols, refer to the description of the `sprintf$()` function in the *Infor ES Programmer's Guide*.

Note: The properties of the application assigned to the current software development project determine the following:

- the primary language of a message
- whether you can specify secondary languages
- which languages are used as secondary languages

For more information on the application properties, refer to the "Create a new Application" on page 132 dialog box.

If an application does not have secondary languages, the message editor will not display the translation fields.

Overview

Use this tab to view and maintain the message properties.

General Information

Message
The unique message name which identifies the message.

Allowed values:

This field is read-only.

You can only specify the message name when you create a new message. The message is identified by the unique combination of the package code and the message code. So, the entire message name you enter in the **Message** field must start with the code of the package for which the message is defined followed by a code for the specific message.

After a message is created, you cannot change its code.

Description in (primary language)
The message expressed in the primary language. This is the text that appears in the message box.

Note: The name of this field includes the primary language of the label. For example, if the primary language is English, the name of the description field will read the following: **Description in English**. The primary language is determined by the Primary Language property of the application for the current software development project.

Allowed values:

A multi-byte text with a maximum length of 132 characters.

The message can contain codes that are substituted when the message is displayed. This table shows some examples:

%1$s	Substitution of a string
%1$d	Substitution of a digit
%1$e	Substitution of an exponent

For details on substitution symbols, refer to the description of the `sprintf$` function in the *Infor ES Programmer's Guide*.

Message Type
The message type indicates a categorization of the messages.

Allowed values:

To specify a value for this field, select one of the following message types:

Information	All goes fine but there is something you may want to know.
Warning	Something went wrong, but the process is still running.
Critical	A fatal error occurred and the process is aborted.

Help Text
If this check box is selected, a help text is linked to this message.

Translation

Target Software Language
Select a language for input of translation. You can only select a secondary language that is not already used for a translation.

To add a new message translation to the **Translations** field, select a target software language. Click **Copy** and translate the message into the selected language.

Note: Selecting a target software language enables the **Copy** button.

Allowed values:

To specify a value for this field, select a language from the drop-down list. This list contains all the secondary languages of the application that are not already used for a translation in the **Translations** field.

Software Language
The software language in which the message text is translated.

Allowed values:

This field is read-only.

The language used for the translation is one of the secondary software languages of the application. You can only specify this software language when you create a new message translation, see **Target Software Language**.

Description
The translated message text.

Allowed values:

A multi-byte text with a maximum length of 132 characters.

The message can contain codes that are substituted when the message is displayed. This table shows some examples:

%1$s	Substitution of a string
%1$d	Substitution of a digit
%1$e	Substitution of an exponent

For details on substitution symbols, refer to the description of the `sprintf$` function in the *Infor ES Programmer's Guide*.

TDE Documentation

Use this tab to view an XML tree with the technical information on the message's Technical Data Entity.

This optional tab is displayed depending on the multipage editor preferences. See "Multipage Editors" on page 145.

Details

Messages

Messages are language-dependent software components that allow you to customize dialog messages.

For each message, you can specify this information:

- The unique message name that identifies the message across all languages.
- The message type, such as "Warning", "Information", or "Critical".
- The message. This is the message text you can edit and translate. It can contain codes that are substituted when the message is displayed. For details on substitution symbols, refer to the description of the sprintf$() function in the *Infor ES Programmer's Guide*.

This table shows a sample message:

Message name	Message Type	Language	Message
tipcss0292	Warning	1 (Dutch)	Artikel reeds aanwezig als component in productiestuklijst
tipcss0292	Warning	2 (English)	Item already found as component in production BOM

Messages are used in a variety of places within the application. You can use messages in these scenarios:

- To replace the default reference error message on table definitions
- Within program scripts and libraries
- To set error messages from a DAL
- As the contents of a form field using the form.text$ function

If a message is produced while a session is executing, a dialog box is displayed. Click **OK** to continue. If the session is running in a job, the message is logged in the job history messages.

Creating messages

To create messages, use the Create a New Infor LN Software Component wizard. To start this wizard, select **File > New > Infor LN Component**.

For more information, see:

- The LN Web Help
- The Message Handling subtopic. This is located in the Functions topic in the *Infor ES Programmer's Guide*

Question Editor

Question Editor

Introduction

Use this editor to maintain the *questions* to which the user must respond.

LN questions are language independent software components. They are used to ask questions to which the user must respond. At runtime, the questions are displayed in the language specified for the current user.

For each question, you can specify the following:

- The unique question code that identifies the question across all languages.
- The question type, such as a "Warning".
- The question. This is the text of the question you can edit and translate.
- The answers to the question to which the system must respond.

Note: The properties of the application assigned to the current software development project determine the primary language of a question and whether you can specify secondary languages. The properties also specify which languages are used as secondary languages. For more information on the application properties, see the Create a new Application dialog box.

If an application does not have secondary languages, the question editor will not display the translation fields.

Overview

Use this tab to view and maintain the question properties.

General Information

Question
The name of the question, which uniquely identifies the question.

Allowed values:

This field is read-only.

You can only specify the question name when you create a new question. A question is uniquely identified by the combination of package code and question code. So, the entire question name you enter in the **Question** field must start with the code of the package for which the question is defined followed by a code for the specific question.

After a question is created, you cannot change its name anymore.

Description in (primary language)
The question put in the primary language of the application.

Note: The actual name of this field includes the primary language of the question. For example, if the primary language is English, the name of the description field will be: **Description in English**. The primary language is determined by the **Primary Language** property of the application for the current software development project.

Allowed values:

A multi-byte text with a maximum length of 132 characters.

The question can contain codes that are substituted when the question is displayed. This table shows some examples:

%1$s	Substitution of a string
%1$d	Substitution of a digit
%1$e	Substitution of an exponent

For details on substitution symbols, refer to the description of the `sprintf$` function in the *Infor ES Programmer's Guide*.

Question Type
The question type indicates a categorization of the questions.

Allowed values:

To specify a value for this field, select one of the following question types:

Information	All goes fine but there is something you may want to know.
Warning	Something went wrong, but the process is still running.
Critical	A fatal error occurred and the process is aborted.

Domain
The domain that determines the possible answers to the question.

Note: Click the Domain hyperlink to open the domain in the Domain Editor.

Allowed values:

Click **Browse** to select a domain. The domain must have the data type **Enumerated**.

Example:

A question is linked to the tcyesno domain. The domain consists of the enumerate constants Yes and No. The question is displayed with a **Yes** button and a **No** button.

Default Answer
The default answer to the question.

Allowed values:

Select one of the enumerate constants belonging to the specified domain.

Help Text
If this check box is selected, a help text is linked to this question.

Translation

Target Software Language
Select the language for input of a translation. You can only select a secondary language of the application not already used for a translation.

To add a new translation of a question to the **Translations** field, select a target software language. Click **Copy** and translate the question into the selected language.

Allowed values:

To specify a value for this field, select a language from the drop-down list. This list contains all the secondary languages of the application not already used for a translation in the **Translations** field.

Software Language
The language into which the question text is translated.

Allowed values:

This field is read-only.

The language used for the translation is one of the secondary software languages of the application. You can only specify this software language when you create a new translation for the question, see **Target Software Language**.

Description
The translated question.

Allowed values:

A multi-byte text with a maximum length of 132 characters.

The question can contain codes that are substituted when the question is displayed. This table shows some examples:

%1$s	Substitution of a string
%1$d	Substitution of a digit
%1$e	Substitution of an exponent

For details on substitution symbols, refer to the description of the `sprintf$` function in the *Infor ES Programmer's Guide*.

TDE Documentation

Use this tab to view an XML tree with the technical information on the question's Technical Data Entity.

This optional tab is displayed depending on the multipage editor preferences. See "Multipage Editors" on page 145.

Details

Questions

Questions are language independent software components used to ask questions to which the user must respond. At runtime, the questions are displayed in the language specified for the current user.

For example, if you change the name of an employee in the Employees - General (tccom0101m000) session, this question is displayed: `Rebuild Search key?` You must respond by clicking **Yes** or **No**.

For each question, you can specify this information:

- The unique question name that identifies the question across all languages.
- The question type, such as "Warning".
- The question. This is the question text as it is shown when activated. The question text can contain codes that are substituted when the question is displayed. For details on substitution symbols, refer to the description of the sprintf$() function in the *Infor ES Programmer's Guide*.
- The domain belonging to the question, and the default answer. The domain determines the possible answers to the question. The domain must have the Enumerated data type.

This table shows a sample question:

Question name	Question Type	Domain	Language	Question
ttaad31022	Warning	ttyeno	1 (Dutch)	Device %1$ geblokkeerd, opnieuw proberen?
ttaad31022	Warning	ttyeno	2 (English)	Device %1$ is locked, try again?

Creating questions

To create questions, use the Create a New Infor LN Software Component wizard. To start this wizard, select **File > New > Infor LN Component**.

You can use questions within program scripts and libraries using the ask.enum() function.

For details, see:

- "Creating a question" on page 106
- The description of the ask.enum() function in the *Infor ES Programmer's Guide*

Report Editor

Introduction

Use this editor to maintain the properties of a *report*.

LN reports are used to output data from the database to a variety of devices, such as printers, displays, and files. One or more LN reports must be linked to an LN print session or to an LN SQL Query. The print session, or SQL Query, reads the data from the Database tables and the data is printed by the Report objects.

The contents and layouts of reports are defined in the data dictionary. You can link a report script to a report. In a report script, you can program actions you want to be performed at particular stages of the report execution. For example, you can create a script to perform calculations on the report data or to read records from related tables not automatically available to the report.

You program report scripts in the same way as you program 4GL program scripts, except report scripts use different event sections and special functions. A report script consists of one or more event sections. Here, you can program actions to be performed at particular states of execution of the report to which the report script is linked. The statements programmed in a report script section consist of a combination of 3GL language statements and report script functions.

The report editor consists of various tabs. Each tab is designed to view particular properties or to carry out specific tasks.

Overview

Use this tab to perform the following tasks:

- Maintain general report information.
- Set various options to customize the page design of a report.
- Manage the libraries (DLLs) used in the report script. You can add or remove DLLs.
- View the debug script. In this script you can set breakpoints for the debug process.

General Information

Report
The identification code of the report.

A report is identified by the unique combination of package code, module code, and report code.

Example:

Package Code:	tt	Tools.
Module Code :	aad	Application Administration
Report Code :	046011000	Time Zones

The total identification of the report is: ttaad046011000.

Allowed values:

This field is read-only and can only be set when you create a new report.

Label Name
The code of the report description label.

Allowed values:

This field is read-only and can only be set when you create a new report.

Note:
To open the label in the Label Editor, click the Label Name hyperlink.

Description
The description of the report. For example, "Help Topics Overview by Search Argument."
Allowed values:

This field is read-only, because the description originates from a label. To edit this label, click **Label Name**.

Tools Interface Version
The Tools Interface Version (TIV) number of the script or library. It is used to guarantee backward compatibility and related to a particular version of the porting set, the Enterprise Server, and the 4GL Tools.

For details about the TIV, see the *Infor ES Programmer's Guide*.

Allowed values:

Whole number in the range 1 - 9999. Initially, the server returns a default version number.

Report Size
The default size of the report.
Allowed values:

To specify a value, click **Browse** and select a report size from the select dialog box. This dialog box provides all predefined report sizes in the development repository of the connected LN server.
This table shows examples of available report sizes:

Report Size	Margin	Top	Bottom	Foot	Length	Width	Font
A3-SMALL	0	1	2	-1	96	132	Double Wide
A3-SMALL-MANUAL	5	1	0	-1	-1	215	User Font 2
A4-LANDSCAPE	5	2	2	-1	-1	-1	Small (17.1 cpi)
A4-LARGE	5	2	2	-1	-1	80	Large (10 cpi)
A4-LARGE-MANUAL	5	1	0	-1	-1	80	Large (10 cpi)
A4-MIDDLE	0	2	2	-1	-1	94	Middle (12 cpi)
A4-MIDDLE-MANUAL	5	1	0	-1	-1	94	Middle (12 cpi)
A4-SMALL	0	2	2	-1	-1	132	Small (17.1 cpi)
A4-SMALL-MANUAL	5	1	0	-1	-1	132	Small (17.1 cpi)

For more information on the previous report size parameters, see the **Size**, **Margin**, and **Font** field groups.

Note: The value you enter in this field will have influence on the report's **Size**, **Margin**, and **Font** fields. For these fields, the selected report size determines the default values and enforces certain constraints.

External Design Present
If this check box is selected, the report has one of these external designs:

- Microsoft SQL Server Reporting Services (SSRS) report design
- Infor Reporting design

A hyperlink to the corresponding additional file is displayed. Click this hyperlink to view the additional file.

For SSRS, you can edit the report design in Microsoft Business Intelligence Development Studio (BIDS). For more information, see these guides:

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

For Infor Reporting, you can edit the report design in Report Studio. For more information, see these guides:

- *Infor Enterprise Server Connector for Infor Reporting Development Guide*
- *Infor Enterprise Server Connector for Infor Reporting Administration Guide*

Font

Report
The font of the report, which is a default for all report layouts. Fonts can be defined in the printer information files of the used printer.

Allowed values:

To specify a value, select a font from the drop-down list. This list provides all predefined fonts in the development repository of the connected LN server.

For example, you can choose one of the following fonts:

- Large (10 cpi)
- Small (17.1 cpi)
- Middle (12 cpi)
- Double Wide
- Near Letter Quality
- Italic
- Superscript

Fixed Device Font
If this check box is selected, the specified font cannot be changed when selecting the device at runtime. It is possible to specify different fonts for each layout in the **Different Font** fields on the **Layouts** tab of the report editor.

If this check box is cleared, it is possible to select another font for the report when you select a device in the Select Device (ttstpsplopen) session. Each layout is printed with the same font. By default, the report font will be used.

Size

Length
The total page length, including top and bottom margin. If this value is -1, the report can be printed on paper types with different page length.

Allowed values:

Whole number in the range from 1 to 999.

Value 0 is not allowed.

Value -1 indicates the page length is dependent on the used paper type.

Width
The page width of the report excluding the left margin. The page width will be used to set the default width for new report layouts. It is used for calculating the number of columns of a multi-column layout.

Allowed values:

Whole number in the range from 1 to 255 (value 0 is not allowed).

Nr. of Multi Column
The number of adjacent columns.

This is useful for printing labels, such as addresses of customers.

If the number of columns is 0, the report generator computes how many columns fit on one line.

Allowed values:

Whole number in the range from 0 to 99. Value 0 indicates a calculated number of columns.

Repeat Expression
An expression which is evaluated at runtime. The result is a numeric value, indicating the number of times each layout on the report will be printed.

Example:

The result of the expression is 2. The report layout will be printed 2 times.

Allowed values:

The expression is an ASCII text with a maximum length of 30 characters.

If you use variables in the repeat expression, these variables must be input fields of the report. See the **Input Fields** tab of the report editor.

Margin

Top
The number of blank lines at the top of each report page.

Allowed values:

Whole number in the range from 0 to 99.

Bottom
The number of blank lines printed at the bottom of each report page. These lines are printed just after the footer section.

Allowed values:

Whole number in the range from 0 to 99.

Left
The number of blank spaces at the beginning of each report line. This left margin is added to the left margin specified in your report script.

Allowed values:

Whole number in the range from 0 to 99.

Foot
The number of lines allocated for the footer. If the value is -1, the footer margin is calculated by the report writer.

Note: When a number of different footer layouts of different sizes are executed on conditions at runtime, it is difficult for the report writer to calculate the footer margin. In this situation, you have to specify a number of lines for this margin.

Allowed values:

Whole number in the range from 1 to 99.

Value 0 is not allowed.

Value -1 indicates a calculated footer margin.

Linked Libraries
This list contains the records of all libraries linked to the report script of the current report.

Sorting

- To sort the libraries by position, name, or description, click the corresponding column header.
- When you sort by position, the sort mode is always ascending. When you sort by name or description, click the involved column header to toggle between ascending and descending sort mode.
- When you sort by name or description, the **Up** and **Down** commands are disabled.

Add / remove libraries, change linking order
You can only perform the following actions if you sort the libraries by position.

- To add a library, click **Add**. Then, press CTRL+SPACE in the **Name** field and select a library from the select dialog box.
- To remove a library, right-click a library and, on the shortcut menu, select **Remove**.
- To change the linking order of the libraries, move them up or down the list.

Note: To edit a library, click the icon behind the library name.

Pos
The library sequence number (Position). This field indicates the linking order of the libraries.

This field is read-only. It is filled automatically when you link a library.

Name
The name of the library.

Allowed values:

To specify this attribute, press CTRL+SPACE and select a library from the select dialog box.

Description
The description of the library.

Allowed values:

This field is read-only. LN Studio automatically fills in the library description after you specified the **Name** of the library.

Script

Create Script
Click this button to create an initial report script for the report.

To view or edit the new report script, close the Report Editor, start the Report Editor again, and go to the **Source** tab.

Remove Script
Click this button to remove a report script from the report.

View Debug Script
Click this link to open the debug script in the debug text editor. In this script, you can set breakpoints for the debug process. See "Using breakpoints" on page 70.

The debug script is generated automatically based on the report layout and the report script, if present.

Input Fields

Use this tab to maintain the input fields of the report.

They are used to specify the table fields or program scripts whose values must appear in the report. All variables used in the report must be defined as input fields.

Note: The variables in the program script must be declared as 'extern' on the program script from which the report is called.

The tab displays a list of all the input fields of the current report.

In the **Input Fields** list, you can add or remove input fields and change their sort mode. You can insert all fields of a table definition using the **Copy Fields from DD** command.

To determine the sort sequence, change the order of the input fields by moving them up or down the list.

When changing the field order, the following constraints prevail:

- You can only change the sort sequence of the sorted input fields (with the sort mode equal to Presorted, Ascending or Descending).
- The options to change the field order are disabled for non-sorted fields (with the sort mode equal to No sorting).
- Sorted fields always precede non-sorted fields.

For more information, see "Input Field shortcut menu" on page 199.

Add and remove input fields

To add an input field:

1. Click **Add**. Alternatively, right-click in the **Input Fields** list and, on the shortcut menu, select **Add**.
2. Enter the **Field Name** of the new input field. Specify either the name of an external variable of the report script, or the name of a table field. To select a table field, press CTRL+SPACE. In the select dialog box, choose a table and a field.
3. Specify the **Depth** (only non-table fields).
4. Specify the field **Domain** (only non-table fields). Press CTRL+SPACE and select a domain from the development repository.
5. When you finished adding input fields, refresh the **Input Fields** list, so the new fields are displayed at the appropriate positions. Right-click in the **Input Fields** list and, on the shortcut menu, select **Refresh**.

For details, see the field help.

To remove an input field, right-click an input field and, on the shortcut menu, select **Remove**. Then, right-click in the **Input Fields** list and, on the shortcut menu, select **Refresh**.

Sort mode

The icon in front of the field name indicates the sort mode of the input field. The following icons are used:

	No sorting
	Ascending
	Descending
	Presorted

If at least one input field of a report has sort mode Ascending or Descending, the report will sort the sent data before printing it. If you are sure the sent data is presorted, use sort mode Presorted, because the sort by the report is time-consuming.

Sorted fields must have a **Depth** of 0 or 1, and non-sorted fields can have greater depth values. If you change the **Depth** of a sorted field to a value greater than 1, the sort mode of the field automatically changes to No sorting. This means that the sorted field becomes a non-sorted field.

Note: Only input variables, with sort mode unequal to No sorting, can be used as sorted fields in a report layout of type Before.field or After.field.

Sort sequence

The position of a field in the **Input Fields** list determines the sort sequence. The field at the top of the list has sort sequence 1, the second field has sort sequence 2, and so on.

When you print a report, the sent data is sorted first on the input field with sort sequence 1, then on the input field with sort sequence 2, and so on.

For example, if you want to sort addresses first by city and then by street, place city at the top of the list (sort sequence 1) and street on the second position in the list (sort sequence 2).

To move fields up and down the list, use **Up** and **Down**.

Note:

- You can only move fields with sort mode Ascending, Descending, or Presorted. Non-sorted fields have sort sequence 99 and are always displayed at the bottom of the list.
- The sort sequence number is not displayed in the **Input Fields** list. To view this number, go to the editor's **TDE Documentation** tab. If this tab is not displayed, enable the corresponding preference setting in the Preferences - Multipage dialog and restart the Report Editor.

Input Fields

Field Name
The name of the variable sent by the program script to the report. This variable can be a table field (like maitm001.item). It can be a variable defined in your program script. This variable must be a global defined variable, type 'extern'.

Allowed values:

In the report editor, this field is read-only. You can only set the field name when you add a new input field. For details, see the **Add** command in the Input Field shortcut menu.

Depth
When the report input variable is an array, this field contains the depth of the array. If the report input variable is a table field, you cannot change its depth.

If the depth of a field is greater than 1, the field cannot be a sorted field (with sort mode equal to Ascending, Descending or Presorted).

Allowed values:

If the input field is a table field, the depth of the field is determined by the table definition. Therefore, for table fields, you cannot change the depth of the field and the depth value is read-only.

If the input field is not a table field and defined as a variable in the report script, you can change its depth to any whole number in the range of 0 - 99.

Note:

- You cannot change the depth of a table field.
- Field depths greater than 1 only apply to non-sorted input fields.

  Therefore, for input fields with a depth greater than 1, the **Sort Mode** command is not available. When you set the depth of a sorted field to a value greater than 1, the sort mode of this input field automatically changes to No sorting, and the field becomes a non-sorted field.

Domain
The domain of the report input field.

Only applicable for non-table fields.

Allowed values:

For non-table fields, specify a **Domain** for the new input field. Select one of the domains available on the current LN server. Press CTRL+SPACE and select a domain from the development repository.

For table fields, the **Domain** of the field is determined by the table definition. You cannot change this field domain here.

Note: To edit the domain, click the icon behind the domain name.

Input Field shortcut menu

Add
Adds a new input field record to the **Input Fields** list. See "Add and remove input fields" on page 197.

Remove
Removes the selected input field record from the **Input Fields** list.

Up
Moves the selected input field up one position in the sort sequence (and in the **Input Fields** list). This command only applies to input fields with a sort mode equal to Presorted, Descending, or Ascending.

Down
Moves the selected input field down one position in the sort sequence (and in the **Input Fields** list). This command only applies to input fields with a sort mode equal to Presorted, Descending, or Ascending.

Refresh
Refreshes the **Input Fields** list, so new fields are displayed at the appropriate positions and have the correct sort sequence assigned. Use this command after you add or remove fields.

Copy Fields from DD
Inserts all table fields of the selected table definition. This table definition must be available on the current LN server.

This command opens a dialog box that shows all table definitions of the current package. To navigate through the table definitions available on the server, use the browse and search facilities of the dialog box. To insert all fields of a table in the **Input Fields** field, select a table definition and click **OK**.

No / Ascending / Descending / Presorted
Sets the sort mode of the selected input field.

This command does not apply to input fields with a **Depth** greater than 1. Such depth fields are always non-sorted (with sort mode equal to No sorting).

Layouts

Use this tab to display the layouts of a report. Through the layout shortcut menu commands, you can perform these actions:

- Add or remove layouts.
- Edit the selected layout.
- Edit all layouts simultaneously.
- Copy a layout from another report.
- View the selected layout.

For more information on the Layout commands, see "Layout shortcut menu" on page 204.

The tab displays a list of all layouts, by their name, used by the report.

Here, you can remove, add/edit layouts, and copy a layout from another report. If you add a layout type that already exists, ensure the sequence number of the new layout is different from sequence number of the existing layouts.

For more information about previous layout operations, see "Add and remove layouts" on page 200.

The **Layouts** list displays the following layout attributes:

- **Layout Type**
- **Field Name**
- **No**
- **Output Expression**

Add and remove layouts

To add a layout:

1. Click **Add**. Alternatively, press Alt+A or right-click in the **Layouts** list and select **Add**.
2. Enter the properties of the new layout. See the field help.
3. When you finished adding layouts, refresh the **Layouts** list, so the new layouts are displayed at the appropriate positions. Right-click in the **Layouts** list and, on the shortcut menu, select **Refresh**.

To remove a layout:

1. Right-click a layout and select **Remove**. Alternatively, select a layout and press Alt+DEL.
2. Right-click in the **Layouts** list and select **Refresh**.

Layouts

Layout Type
The layout type.

Note: You can select a layout type from the list. When you select a layout type that is already used, specify a sequence number different from the numbers of the used layouts.

Allowed values:

A report can contain the following layout types:

Before.report	Layouts of this type are printed once at the beginning of the report. Use this layout to present information about the report. For example, the title page of the report.
Header	Header layouts are printed at the top of every page.
Before.field	You can define this layout for each input field for which a sort mode is defined. Each time the value of the sorted input field changes, this layout is printed before the Detail lines. In this way the records are grouped. If a group of records spans more than one page, the Before.field layout is repeated at the top of the new page (following the header). To suppress automatic printing at the top of every page, use `lattr.break` as an output expression. `lattr.break` is a predefined variable that contains the value 1 (true) when the sort field changes. When you use this output expression, the layout is only printed when the input variable is changed. To define multiple grouping levels in a report: **1** On the Report Input Fields tab, define a Sort Mode and a Sort Seq. for each input field by which the records must be grouped. **2** Define a Before.field for each of these sort fields. An example of multiple grouping levels is to group Purchase Order records first by buyer, and then by buy-from business partner. The order of sorting and grouping depends on the sort sequence that was specified. This is the sequence of the fields in the **Input Fields** list.
Detail	Detail layouts are used for the individual records included in the body of the report. They are printed every time data is sent to the report.
After.field	You can define this layout for each input field for which a Sort Mode is defined in the Input Fields tab. Each time the value of the sorted input field changes, the layout is printed after the detail lines. In this layout, the presorted input field still contains the old value. Among other things, this layout can be used to print totals for a particular group of detail lines.
Footer	Footer layouts are printed at the bottom of each page. This layout is used to print this information: • Titles • Page numbers • Page totals
After.report	This layout is only printed at the end of the report, and after all other layouts. If this layout does not fill a complete page, the footer layout is printed at the bottom of the page, where the After.report layout(s) is printed.

Note: The **Field Name** field is applicable only if you select Before.field or After.field layouts.

Field Name
The name of an input field used to activate Before.field and After.field layouts. When the field value changes, the layout will be printed.

Note: Select a field from the list.

Allowed values:

You can only specify fields, which are defined as a sorted input field in the **Input Fields** tab of the report editor.

The **Field Name** attribute only applies to Before.field and After.field layouts.

No.
The sequence number of the layout.

You can define more than one layout per Layout/Fieldname combination. The layouts of the same Layout/Fieldname will be printed in the order of the sequence number of the layouts.

There are two reasons to define more than 1 layout per Layout/Fieldname combination:

- The maximum size of a layout is 22 lines. If your layout needs more lines, make a second layout of the same type with a higher sequence number.
- If a particular layout must be printed under certain conditions and another layout under other conditions, define more than one layout with different output conditions.
- If the report already uses the previously specified layout type with the same sequence number, LN Studio issues the following warning:

  ```
  Record not unique.
  ```

Note: You can only add a layout type that already exists in the current report if the sequence number of the new layout differs from that of the existing layouts.

Output Expression
An expression which is evaluated at runtime. The result is always true or false. The expression syntax is equal to the 3GL syntax. When a layout must be printed, the system evaluates the output condition of the layout. If the result is true, the layout is printed.

Example:

Output Expression	Print result
`1`	Layout will always be printed.
`0`	Layout will never be printed.
`print.detail = 1 or print.detail = 2`	Layout will be printed if variable print.detail is 1 or 2.
`strip$(tdsls041.cprj) <> ""`	Layout will be printed if field tdsls041.cprj is not empty.

Allowed values:

The variables used in the output expression must be defined as input fields for the report.

Default value:

The default value is 1. Therefore, the expression is always true and LN will print the layout (if not prevented by other controls).

Format

Different Font
The font in which the layout will be printed.

Note: This field only applies to a report that can have different fonts. You can only specify another font if you previously selected **Fixed Device Font** on the **Overview** tab of the Report Editor.

Allowed values:

These fonts can be defined in the printer information files of the printer being used.

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

For example, you can choose one of these fonts:

- Large (10 cpi)
- Small (17.1 cpi)
- Middle (12 cpi)
- Double Wide
- Near Letter Quality
- Italic
- Superscript

Extra Need
This number indicates how many extra lines are required for the layout on the same page.

Before printing a layout, the report writer calculates the free space at the current page. If the layout does not fit on the same page, the layout will be printed on a new page. If the value of this field is 0, only the number of lines of the layout itself is taken into account. With this field you can increase the number of extra free lines for the layout. This can be useful if you want to print at least 2 detail layouts on the same page together with a Before.field layout.

Note: This field does not apply to the Header and the Footer layout types.

Allowed values:

A whole number in the range 1-99.

Multi columns
This field indicates whether the report layout must be printed in more than one column or not.

The number of columns depends on the size of the layout and the paper width. Change the layout size with the command <Esc>[S] in the report layout editor.

Allowed values:

Select one of the following values from the drop-down list:

- None
- Vertical Order
- Horizontal Order

New Page
If this check box is selected, the report layout is printed on a new page. Skipping to a new page always causes printing of present Header, Footer and Before.field layout types. If the current line is at the top of the page, the page command will be ignored.

Note: This field does not apply to Header and Footer layouts.

Number of Columns
The number of characters or positions per line.

Allowed values:

This display field is read-only. This number is calculated automatically.

Number of Rows
The number of lines of the report layout.

Allowed values:

This display field is read-only. This number is calculated automatically.

Layout shortcut menu

Add (Alt+A)
Adds a new layout record to the **Layouts** grid. See "Add and remove layouts" on page 200.

Remove (Alt+DEL)
Removes the selected layout record from the **Layouts** grid. See "Add and remove layouts" on page 200.

Refresh
Refreshes the list of layouts. Use this command after you added or removed a layout.

Copy
Copies the selected layouts. You can paste the copies in the list of layouts in the current report, or in another report.

Paste
Pastes copies of report layouts in the list of layouts.

Edit Layout
Opens the selected layout in an ASCII oriented report layout editor (ottadvformedit) of LN. See the online Help of LN.

Edit All Layouts (Alt+Y)
Opens all layouts in an ASCII oriented report layout editor (ottadvformedit) of LN. See the online Help of LN.

Copy from Other Layout
Copies a layout from another report available in the development repository of the current LN server.

The command starts the Copy from other layouts dialog. In this dialog, choose a report and select one of the layouts of this report.

This is helpful if you repeatedly use the same elaborate layouts, such as with complex expressions or formats.

Source

Use this tab to view and edit the source code of the report's report script. The tab consists of a script editor, which forms an integral part of the report editor.

See "Script Editor" on page 263.

This tab is only displayed, if a report script is linked to the report.

Documentation

Use this tab to enter the report's release notes and technical documentation.

Release Notes
Use this field to specify the release notes of the report.

Technical Documentation
Use this field to specify the technical documentation of the report.

TDE Documentation

Use this tab to view an XML tree with the technical information on the report's Technical Data Entity.

This optional tab is displayed depending on the multipage editor preferences. See "Multipage Editors" on page 145.

Session Editor

Session Editor

Introduction

Use this editor to maintain the properties of a session.

Sessions are the basic functional elements of LN applications and used to perform actions or tasks.

Overview

Use this tab to specify the general properties for the session.

General Information

Session
The session name that uniquely identifies the session. The name must start with the package code and module code to which the session belongs.

Label Code
The name of the label that contains the session's description.

To edit or view the label in the Label Editor, click the hyperlink.

Description
The session's description, displayed in the title bar of the session's window at runtime.

Allowed values:

This field is read-only, because the description originates from a label. To edit this label, click **Label Code**.

Main Table
The session's main table.

Allowed values:

To select a main table, click **Browse**.

To edit or view the table in the Table Editor, click the hyperlink.

Session Status
The development status of the session.

Allowed values:

Developing	The session is being programmed.
Testing	The session is going through software testing.
Finished	The session is fully programmed and tested.

Synchronized Dialog
The name of an overview session's details session. The details session starts when a user double clicks a record in the overview session.

Allowed values:

To select a details session, click **Browse**.

To edit or view the details session in the Session Editor, click the hyperlink.

Main Session
If this check box is selected, the session is a main session and can be placed in a browser menu.

If this check box is cleared, the session is a subsession and cannot be placed in a browser menu. Users can only start the session through an *overview session*, or by a specific form command.

InContext Model
The name of the InContext model for this session.

To open the InContext model in the Session Model Editor, click the hyperlink.

If no InContext model exists for the session, click the hyperlink to generate a model. The Session Model Editor is started.

Related topics:

- "InContext Modeling" on page 75
- *Infor Enterprise Server InContext Modeling Development Guide*
- "Session Model Editor" on page 236

Table Indices and View Fields

Table Indices and View Fields
Use this field to indicate which *indices* of the main table are used in the session.

The field displays a tree structure in which the nodes are represented by check boxes. The tree structure lists all indices of the session and, for each index, the session fields used for this index. By selecting or clearing check boxes, you indicate which index applies and which index fields are displayed as View fields.

The top level **Index** check boxes denote the available indices. Select an index check box to indicate this index is applicable for the session. If this check box is selected, you can activate the index at run time through the **Sort by** command in the session's **View** menu.

The subordinate **View** check boxes indicate which fields of a selected index are displayed on the form as view fields. When you select a particular index field as view field, the preceding index fields will be displayed as view fields.

Click **Up** and **Down** to move indices up and down the list. The sequence of the indices in the list determines the order in which indices are displayed when you activate the **Sort by** command at runtime. The index at the top of the list is the session's start index.

Session Behavior

Start Command
The session's Start Command.

Specify the default start command or one of the standard commands available to the session (see **Standard Commands** field). Alternatively, indicate that the session will not have a start command.

Allowed values:

To specify this field, select a start command from the drop-down list. You can select on of the following values:

Default Start Command	The session uses the default start command that is defined for the session type concerned.
No Start Command	No command is carried out when the user starts the session.

<Standard Command>	One of the session's standard commands, as displayed in the **Standard Commands** field.

Session Type
Specifies the session's type.
Allowed values:

Maintain	For overview and details sessions (when not started in read-only mode), in which users can edit fields.
Update	Type 4 session.
Print	Type 4 session.
Display	Maintain session type, except that users cannot edit fields in an overview session. Users can edit fields in a details session.
Update + Print	Update session that produces a report. This session type behaves in the same way as a Print session, but at runtime the Preview button is not generated.
Graph	Type 4 session.
Conversion	Type 4 session.

Window Type
Specifies the session's window type.
Allowed values:

Modeless window with menu	Deprecated
Modal window with menu	Deprecated
Dialog	A modal secondary window, to display or to edit data.
No Window	The window is not movable and does not contain a title.
Synchronized dialog	Deprecated
List Window (default value)	Starts a details window for a details session, and an overview window for an overview session. If a user zooms to the session, a browse list starts for the session.
Wizard	A wizard window starts for the session. A wizard window usually consists of a sequence of dialog boxes in which you must enter the required details, and through which you can move forwards and backwards. LN contains various sessions with a wizard window, for example: • Export Language Dependent Data to XML - Wizard (ttadv8910m000) and • Import Language-Dependent Data from XML - Wizard (ttadv8920m000)

Parameter	A parameter window starts for the session. LN contains various sessions with a parameter window, for example • Infor Data Navigator Server - Parameters (ttgfd4124m000) and • ART Parameters (ttaad6102s000)
Multi Main Table	A Multi Main Table window starts for the session. Use this window type for Multi Main Table (MMT) sessions. An MMT session is a combination of multiple sessions, each having their own main table, and is used to present data with a typical parent-child structure. For example: a Sales Order and the related Sales Order Lines. The window of an MMT session consists of 2 parts: • The upper part shows parent information. For example, a Sales Order. • The lower part shows the child information. For example, the related Sales Order Lines. **Note:** The child satellite sessions of a MMT session are maintained on the **Satellite Sessions** tab of the session editor.

Reports in several languages
If this check box is selected, you can print the session's reports in languages other than the runtime language.

Note:

- Dependent on software delivery (see the Import- / Export functions module). You can only print reports in the languages you select in the Languages to Deliver by Language (ttiex0101m000) session.
- You can manage multilingual report printing in the session's UI script.

Standard Commands
Use this field to enable / disable the standard commands of a session.

Note: Click **St. Command set** to start a dialog where you can select a standard command set for the session. The commands of the selected command set are displayed in this field.

The field displays a tree structure in which the nodes are represented by check boxes. The tree structure lists all standard menus for sessions and the standard commands for each menu. Select or clear check boxes to indicate which standard commands and associated menus are used.

The top level **Menu** check boxes denote the standard menus. Select a menu check box to select all standard commands of this menu. Clear a menu check box to clear all commands of the menu. The menu check box shows a greyed out check mark if some commands of the menu apply and some do not.

The subordinate **Command** check boxes indicate which standard commands are used for the session. For every standard command a check box is available. Select a check box to enable the command at runtime. Clear a check box to disable the command at runtime. The command preferences set in this field can be overruled by the program script, DAL script, or the user authorization settings.

Script

Use this tab to specify properties of the session's UI script and to link libraries to the session.

To edit or view the UI script, go to the **Source** tab.

General Information

Standard Script
The session's standard script type.

Allowed values:

No	No standard script is used. You must specify a UI script in the **UI Script** field. If you change the Standard Script from 'Type 1/2/3' or 'Type 4' to value 'No', the editor will automatically generate a script name.
Type 1/2/3 (With M.Table)	The session is based on a main table and uses a standard script.
Type 4 (Without M.Table)	The session is not based on a main table, but uses a standard script

Note:

- If you specify a standard script, you cannot specify a UI script.
- When you specify a non-standard script (value 'No'), the editor creates a session reference that can be used as an alias for the session's script.

Default value:

No

UI Script
The name of the *program script*.

A script is identified by its unique name, which is the combination of the package code, the module code, and a script code.

The name and description of the script generally corresponds with the name and description of the session, with the exception of existing sessions with an anomalous name . When you add a script to a script-less session, the session editor will automatically generate a script (skeleton) with a name and description derived from the associated session.

Example:

Package Code:	ti	LN Manufacturing
Module Code :	pcf	Product Configuration
Script Code :	0110	Maintain Features

The total identification of the program script is: tipcf0110

Session reference
The session that owns the UI script of the current session. Click **Browse** and select the session from the list.

UI script ownership

Each session, which does not use a standard script, is linked to a UI script. Each UI script is owned by a session.

A UI script can be used by multiple sessions, but has only one owner. For example, sessions tetst0101m000 and tetst0101s000 both use the UI script tetst0101. Session tetst0101m000 is the owner of the script. For both sessions, the following properties apply:

Session	UI Script	Session reference
tetst0101m000	tetst0101	tetst0101m000
tetst0101s000	tetst0101	tetst0101m000

Note:

- You can only edit a UI script via the session that owns the script.
- When you create a new session with a UI script, LN Studio automatically assigns the new session as the owner of the new script.
- If a script is not owned by a session, use the Check and Convert UI-Script Ownership (ttadv2930m000) session to assign a session as the owner of the script.

  If a script is already owned by a session, you can use this session to assign a different session as the owner of the script.

To change the session reference

When you change the session reference, the **UI Script** field is automatically updated and filled with the script code of the referenced session. This referenced session must be a script owner.

In the **Session reference** field, you can specify the code of the current session itself. If you do this, a new, unique, script is assigned to the session. The new script is a copy of the script that belongs to the previously referenced session. The code of the new script is identical to the code of the current session.

If the **Session reference** field is filled with the code of the current session, you cannot change the session reference, because this would leave the script without an owner.

Expire / Unexpire sessions

A session can only be expired if there is no reference to it from another not-expired session.

A session cannot be unexpired if it has a session reference to an expired session.

Script Type
Specifies the use or nature of the session's program script.

- You can only edit a UI script via the session that owns the script.
- When you create a new session with a UI script, LN Studio automatically assigns the new session as the owner of the new script.

- If a script is not owned by a session, use the Check and Convert UI-Script Ownership (ttadv2930m000) session to assign a session as the owner of the script.

  If a script is already owned by a session, you can use this session to assign a different session as the owner of the script.

This field only applies if the session does not use a standard script, which implies that the **Standard Script** field is set to 'No'.

For sessions without standard scripts, you can use the following script types:

UI Script with Database handling	A 4GL user interface script is used for display and maintain sessions. The script is called by the 4GL engine (standard program) on certain events.
UI Script for Print/Processing Sessions	A 4GL user interface script is used for print and processing sessions. The script is called by the 4GL engine (standard program) on certain events.
3GL (Without 4GL Engine)	A 3GL script has no interaction with the 4GL engine (standard program). Therefore, all logic of the program must be included in this script. Examples of 3GL scripts are the well-known General Table (ttaad4100/ttaad4500) sessions.

Compile Flags
The compile flag.

In (3GL and 4GL) sources, use compile flags to compile a part of the source depending on certain conditions. This is called conditional compiling. For more information on conditional compiling and other compile options, refer to sections "Preprocessor" and "Compiler" of the *Infor ES Programmer's Guide*.

Example:

```
 | Compile Flag: -DMYTEST | Source ... #if MYTEST message(Some debug
information) #endif ...
```

You can specify several compile flags separated by a space " ". For example:

```
Compile Flags: -DMYTEST -DVERSION_1 -DVERSION3.0
```

Note: The preprocessor only works during compilation of a 3GL source. This is because the standard generator std_gen6.2 does not have a preprocessor pass. Therefore, you cannot use 4GL events in a #if, #ifdef or #ifndef.

Tools Interface Version
The Tools Interface Version (TIV) number of the script/library.

This is used to guarantee backward compatibility and is related to a particular version of the porting set, the Enterprise Server and the 4GL Tools. For example, the TIV number for Enterprise Server 7, delivered together with LN 6.1 FP1, is 1010. When you insert a new library, the TIV is automatically set to the highest supported TIV number.

Setting a TIV number for each object can result in the behavior of these objects being different. This can occur for incompatible features executed with different versions of the porting set or the LN 4GL Engine.

For example, the "set.synchronized.dialog" function was changed in TIV 1000. Therefore objects in which this function is used, show different behavior for different TIVs:

- If the object's TIV is less than 1000, a synchronized dialog is started when a record is inserted or duplicated in an editable grid.
- If the object's TIV is 1000, no synchronized dialog is started when a record is inserted or duplicated: the new record is entered directly in the grid.

For details about the TIV, see the *Infor ES Programmer's Guide*.

License

Product ID Source
The product ID for which a license is needed to compile the object. If the field is empty, no license is needed to compile the object.

Note: This field only applies to internal Infor development systems. For other development environments, this field is not available.

Allowed values:

To select a product ID, click **Browse**.

Product ID Object
The product ID for which a license is needed to run the object. If the field is empty, no license is needed to run the object.

Note: This field only applies to internal Infor development systems. For other development environments, this field is not available.

Allowed values:

To select a product ID, click **Browse**.

Linked Libraries
This list contains the records of all libraries linked to the UI script of the current session.

Sorting

- To sort the libraries by position, name, or description, click the corresponding column header.
- When you sort by position, the sort mode is always ascending. When you sort by name or description, click the involved column header to toggle between ascending and descending sort mode.
- When you sort by name or description, the **Up** and **Down** commands are disabled.

Add / remove libraries, change linking order
You can only perform the following actions if you sort the libraries by position.

- To add a library, click **Add**. Then, press CTRL+SPACE in the **Name** field and select a library from the select dialog box.
- To remove a library, right-click a library and, on the shortcut menu, select **Remove**.
- To change the linking order of the libraries, move them up or down the list.

Note: To edit a library, click the icon behind the library name.

Pos
The library sequence number (Position). This field indicates the linking order of the libraries.

This field is read-only. It is filled automatically when you link a library.

Name
The name of the library.

Allowed values:

To specify this attribute, press CTRL+SPACE and select a library from the select dialog box.

Description
The description of the library.

Allowed values:

This field is read-only. LN Studio automatically fills in the library description after you specified the **Name** of the library.

Documentation

Release Notes
Use this field to specify the release notes of the UI script.

Technical Documentation
Use this field to specify the technical documentation of the UI script.

Source

Use this tab to view and edit the source code of the session's UI script. The tab consists of a script editor, which forms an integral part of the session editor.

See "Script Editor" on page 263.

This tab is only displayed, if the source code of the UI script is available.

User Interface

Use this tab to specify the form properties, link reports and charts to the session, and specify text fields.

Form

Edit/View Form
To edit or view the form in the Dynamic Form Editor (DFE), click the **Edit/View Form** hyperlink.

Label of Object
You can enter a label name in this field. At runtime, the label is visible in the pull-down menu of the session if the session belongs to a multi-main table session.

Allowed values:

To specify this field, click **Browse** and select a label name from select dialog box.

Customization prohibited
If this check box is selected,, the customize form functionality in a details session and the customize visibility, ordering and width of columns functionality in an overview session is disabled if the session is started from Web UI.

Default Button

Command Type
Select the command type of the session's default button.

Allowed values:

Not Applicable	The default button is disabled.
Standard Command	The default button starts a standard command.
Form Command	The default button starts a form command.

Default Button
Select a command for the session's default command button. The default command button is the command LN carries out when a user presses the Enter key whilst using the session. The default command button is always the first button in the list of command buttons included on a session's form.

Allowed values:

Not Applicable	Session's default command button is disabled.
If Command Type is Standard Command	One of the session's Standard Commands.
If Command Type is Form Command	One of the session's Form Commands.

Dynamic index switching

Select the session's dynamic index switching options.

You can only enable dynamic index switching if the current session has a Main Table. For sessions without a main table, disable dynamic index switching. To do this, select value 'No'.

Allowed values:

No	When a user chooses a new index, the view fields and session layout do not change. Select this value if the session does not have a main table.
Restart when view-fields differ	The session restarts when a user chooses a new index, which alters the view fields displayed above the session's grid.
Always restart	The session restarts when a user chooses any new index.

Related topics:

- "Implementing dynamic index switching" on page 228
- *Dynamic session*
- *Integrated session*

Show name of index
If this check box is selected, the session's title bar displays the current index for overview sessions with dynamic index switching implemented.

The check box is disabled if **Dynamic Index Switching** has value ‘No’.

Workflow / Web UI

Workflow Enabled
Use this field to specify whether the session can run in a Workflow environment.

Allowed values:

Select one of the following values from the drop-down list. Only three values of the list have a meaning, as indicated in the table below.

Value	Description
Not checked	The session has not been checked in a Workflow environment
Single / Redoable	The session has been checked in a Workflow environment and executed successfully
Single / Not Redoable	Not used
Container / Redoable	Not used
Container / Not Redoable	Not used
Single / Container / Redoable	Not used
Single / Container/ Not Redoable	Not used
Not Workflow Enabled	The session has been checked in a Workflow environment but failed

Web UI Enabled
If this check box is selected, you can successfully run this session in a Web UI environment.

Reports

List of all reports linked to the current session. The reports are listed by report group.

You can add or remove reports:

- To add a report to a session, click **Add**. The Select a Report dialog box starts. Click **Browse** and choose a report. Then, specify a report group number and click **OK**.

  See "Select a Report" on page 96.

- To remove a report from a session, right-click the report and select **Remove**.

You can drag and drop reports from one report group to another.

To change the sequence of the reports in a report group, move them up or down the group list.

Charts

List of all charts linked to the current session. The charts are listed by chart group.

You can add or remove charts:

- To add a chart to a session, click **Add**. The Select a Chart dialog box starts. Click **Browse** and choose a chart. Then, specify a chart group number and click **OK**.

  For details, see "Select a Chart" on page 93.

- To remove a chart from a session, right-click the chart and, on the shortcut menu, select **Remove**.

You can drag and drop charts from one chart group to another.

To change the sequence of the charts in a chart group, move them up or down the group list.

Text Fields

List of all text fields of the current session (text fields are table fields of data type text).

You can add or remove text fields.

To add a text field to a session:

1. Click **Add**.
2. In the **Name** field in the grid, press CTRL+SPACE to start a dialog where you can select text fields.
3. Select the desired text field and click **OK**. The text field is displayed in the grid.
4. Complete one of these steps:
   - Clear the **Writable** check box, so users can only read the text.
   - Select the **Writable** check box, so users can also edit the text.

   The text authorizations by text group, defined in the Text manager module, always have priority over the authorizations you define in this grid. For example, a user is not authorized to change texts in text group ABC. Therefore, even if this check box is selected, he cannot edit a text that belongs to text group ABC.

To remove a text field from a session, right-click the text field and, on the shortcut menu, select **Remove**.

To change the sequence of the text fields, move them up or down the list. The sequence in the Session editor corresponds with the order in which the text fields are displayed in the session at runtime.

Form Commands

Use this tab to maintain the *form commands* for the session.

All Form Commands

List of all form commands of the current session. The tree structure of the list mirrors the menu hierarchy of the commands.

A form command can have activation types Function, Session, Business Method, or Menu. Therefore, the command can invoke these types of activation objects. LN Studio uses the following icons to denote these activation types:

	Function
	Session
	Business Method
	Menu

In the **All Form Commands** field, you can add and remove form commands:

- To add a form command to a session, click **Add**. The Create a New Form Command wizard starts. Enter the properties for the new form command and click **Finish**. See "Create a New Form Command" on page 136.
- To remove a form command from a session, right-click the form command and, on the shortcut menu, select **Remove**.

The tree structure of the displayed form commands mirrors the menu hierarchy of the form commands in the runtime session. Form commands of type **menu** can have child nodes, which must be commands of another activation type. Menu commands cannot contain child items of type menu command. A command that does not activate a menu cannot have child nodes. The Session editor only supports one level of submenus. Therefore, a menu command can only have child nodes and cannot have grandchild nodes.

To change the menu structure, drag and drop a form command from one location to a menu command or to the root of the tree structure. In the latter case, the command appears, at runtime, directly under the session's *appropriate menu*.

To change the sequence of the form commands in a menu, move the commands up or down the command list of a specific menu.

Form Command Properties

Activation

Activation Type
Activation Type specifies what your form command activates. You can select the activation types from the drop-down list. The used type names are self explanatory.

Allowed values:

- Function
- Session
- Business Method
- Menu

Function/Session/Business Method/Menu
The name of the function, session, business method, or menu that the form command activates.

Allowed values:

- A session or menu available in the development repository of the connected LN server. To select a session or a menu, click **Browse**.

  To open the selected menu or session in the corresponding editor, click the Menu or Session hyperlink.

- An external function in the session's UI script. You can select a function from the drop-down list.

  To view or edit the selected function in the script editor, click the Function hyperlink.

- A business method in the session's UI script, or DAL script. You must type in the code of the business method.

Command Type
Command Type specifies where the form command appears on the session's form. You can select the following command types from the drop-down list:

Allowed values:

Form	For all of the session's form tabs, the form command appears on the form's *appropriate menu*, and/or as a button on the form.
Group	The form command only appears on the session's *appropriate menu* when a user selects the tab that includes the group you define in the **Group/Field Number** field.
Field	The form command appears as a button on the form, behind the field whose **Sequence Number** you enter in the **Group/Field Number** field.
Print	The form command appears as an option in the session's print menu.

Group Number / Field Number
This field specifies the group to which the form command belongs, or the sequence number of the field linked to the form command. Whether this field specifies a group or a sequence number depends on the value you entered as the **Command Type**: **Group** or **Field**.

This field only applies to form commands of type 'Group' or 'Field'. The field is disabled for the other form command types ('Form' and 'Print').

The field has a dynamic name derived from the command type: **Group Number** or **Field Number**.

For form commands of **Command Type Group**, select the group to which the form command belongs. The form command only appears in the session's *appropriate menu* when the user selects the group's tab.

For form commands of **Command Type Field**, select the field behind which the form command appears as a field button.

The pull down list shows the numbers and label codes of the available groups or fields.

Allowed values:

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

Category
The form command category used by Infor Ming.le. The category determines the pull-down menu in which the command will be displayed.

Allowed values:

Action	The command is displayed in the session's **Action** menu.
Reference	The command is displayed in the session's **Reference** menu.
View	The command is displayed in the session's **Views** menu.
Special 1	The command is displayed in an additional pull-down menu, which is specific to the session.
Special 2	Use the Special 1 and Special 2 categories to group frequently used commands. Per special category, you must select one default command. The description of this default command is displayed on the special category's menu button. Specify a short description in the **Short Description** field. If you leave the **Short Description** field empty, the **Long description** will be displayed on the menu button, so the button becomes too large.

Note: The category is only used by Infor Ming.le. If the session runs in another UI, the category is ignored.

Form commands in a menu must have the same category as the menu.

Default Command
Use this check box to define the form command as the default command for the session, or for a command category in the session.

You can select this check box in these situations:

- If the command has **Command Type Print**.
- If the command belongs to the Special 1 or Special 2 category.

  If this check box is selected, the command is the default command of its category. If the session runs in Infor Ming.le, the description of the default command is displayed on the category's menu button. See the example.

Example - default command for a category

This table shows an example:

Session/Function/Business Method	Long description	Category	Default Command
exec.user.1	Activate Job	Special 1	Yes
exec.user.2	Queue Job	Special 1	No
exec.user.3	Block Job	Special 1	No
exec.user.4	Cancel Job	Special 1	No

The commands of the Special 1 category are displayed in the **Activate Job** pull-down menu.

Note: You can specify one default command per special category.

The default command is only used by Infor Ming.le. If the session runs in another UI, the default command is ignored.

Display

Label Code
The code of the form command's label.

To edit or view the label in the label editor, click the Label Name hyperlink.

Allowed values:

To specify this field, click **Browse** and select a label name from select dialog box.

Description
The description of the form command's label, as specified in **Label Name**.

Allowed values:

This field is read-only. LN Studio automatically fills in the label description after you specified the **Label Name**.

Shortcut Key
The form command's shortcut key. This property does not apply to form commands that invoke a menu.

Select the form command's shortcut key from the drop-down list.

Allowed values:

The value for **Shortcut Key** is one of the following:

- "Not applicable"
- A key combination that conforms to the following syntax: `CTRL+F1`- `CTRL+F12`, or `CTRL+SHIFT+A` - `CTRL+SHIFT+Z`

Add Ellipses to Label
If this check box is selected, an ellipsis (...) follows the form command's label. Use ellipses to indicate the form command opens a dialog box.

Suppression of Dialog Allowed
If this check box is selected, users can suppress the dialog box, which is started by the form command, at runtime. The dialog box is suppressed if user defaults are defined and quick flow is activated. If the dialog box is suppressed, the corresponding default command is executed.

This check box is only available if the **Add Ellipses to Label** check box is selected.

Separator
If this check box is selected, Enterprise Server inserts a line in the menu under the form command. Use these lines to create separate groups of form commands in the menu.

Availability

Authorization Group
The authorization level users need to carry out the form command. All commands with an authorization group higher than the user's authorization for the session are removed from the *appropriate menu*.

Note: For form commands of type **Menu**, you must specify **Authorization Group Display**.

Example:

A user is authorized to view and print records in a session. The authorization group for that session is 'Print/Display'. All form commands with authorization groups 'Full Authorization' (Delete/Insert/Modify/Print/Display), 'Insert/Modify/Print/Display' and 'Modify/Print/Display' will be removed from the *appropriate menu*.

Allowed values:

- Display
- Print/Display
- Modify/Print/Display
- Modify/Print/Display
- Insert/Modify/Print/Display
- Full Authorization

Choose the authorization group of a form command according to the following guidelines:

Command Type	If this form command…	Authorization Group
Form, Field or Group	gives access to functionality, where a delete action should be possible.	Full Authorization
Form, Field or Group	gives access to functionality, where an insert action should be possible, but no delete action is needed.	Insert/Modify/Print/Display
Form, Field or Group	gives access to functionality, where a modify action should be possible, but no insert or delete action is needed.	Modify/Print/Display

Command Type	If this form command…	Authorization Group
Form, Field or Group	gives access to functionality, where data is displayed, but no print, modify, insert, or delete actions are needed.	Display
Print	gives access to report functionality, which also provides an option for the end-user to delete data (or executes a delete action in the script).	Full Authorization
Print	gives access to report functionality, which also provides an option for the end-user to insert data, but not to delete (or executes an insert action in the script).	Insert/Modify/Print/Display
Print	gives access to report functionality, which also provides an option for the end-user to modify data, but not to insert or delete (or executes a modify action in the script).	Modify/Print/Display
Print	gives access to report functionality, which prints data, but does not provide an option to modify, insert or delete data.	Print/Display

Note: Define authorizations in the Authorization Management System module, and link these to a user on the Authorizations tab of the User Data (ttams1100s000) details session.

Command Availability
The availability of the form command.

Note: This field does not apply to form commands of type menu.

Allowed values:

Always	Authorized users can activate the form command. The programming function disable.command overrules this option.
One Record Selected	Authorized users can only activate the form command when they have selected one record in the overview session's grid. The programming function disable.command overrules this option. This option overrules the enable.command programming function.
Records Selected	Authorized users can only activate the form command when they have selected one or more records in the overview session's grid. The programming function disable.command overrules this option. This option overrules the enable.command programming function.
Never	Users cannot execute the form command. The programming function enable.command overrules this option.

Show as Button in Dialog or Browse Session
If this check box is selected, the form command displays as a field button on the session's form.

Show in Menu bar of Details window
If this check box is selected, the form command is shown on the details session's *appropriate menu*.

Note: Programming functions are available to create an application toolbar that includes a session's form commands.

Initial hidden
If this check box is selected, the form command is initially not displayed in the session. At runtime, you can make the form command visible through the **Personalize Session Menus** or **Personalize Toolbar** option. The initially hidden commands are displayed as a menu option in the **Hidden** panel of the Personalize Session Menus dialog box, just like the other hidden menu options.

Initial hidden when added to Toolbar
If this check box is selected, the session does not display the form command if the form command appears as an icon on the toolbar.

Execution

Before Execution

Question
When the user tries to execute the form command, Enterprise Server prompts the user with the question whose name you enter in this field. This field does not apply to form commands of type menu. If the field is left empty, no question is asked.

Allowed values:

To specify this field, click **Browse** and select a question from the select dialog box.

Answer
To carry out form commands, a user must give the answer you enter in this field when prompted by Enterprise Server with the question in the **Question** field. This restriction only applies if the **Question** field has been specified. This field does not apply to form commands that invoke a menu.

Allowed values:

To specify this field, select a answer from the drop-down list. This list contains all answers available for the selected question.

Multi Session execute
For an overview session, when a user executes a form command with multiple records selected in the grid, if in this field you select:

Once, LN only carries out the form command for the first record selected in the grid.

For every occurrence, LN carries out the form command for every record selected in the grid.

Allowed values:

Once	Form command is only carried out for the first record selected in the grid.
For every occurrence	Form command is carried out for every record selected.

Save data
If this check box is selected, when a user activates the form command, LN saves the session's data before carrying out the form command. This option does not apply to form commands of type menu.

If this check box is cleared, when a user activates the form command, LN carries out the form command without saving the session's data.

Execution

Start mode Session
Select the mode of the session the form command starts.

Note: This field only applies to form commands of type Session.

Allowed values:

- Modal
- Modeless
- Synchronized Child
- Modal Overview
- Not applicable

Kind of Session
Select the type of session the form command starts from the options.

Note: This field only applies to form commands of type Session.

Allowed values:

- Overview Window
- Details Window
- Not applicable

After Execution

Refresh Parent
If this check box is selected, when users execute the form command in a session, the data affected by the form command automatically updates in any related sessions.

If this check box is cleared, users must manually save the changes after they execute the form command for the changes to update in any related sessions.

Satellite Sessions

If the session is a multi-main table (MMT) controller session, use this tab to maintain the MMT satellites for the session.

The fields of this tab are only enabled for sessions that have **Window Type Multi Main Table** specified and **Session Type Maintain**.

An MMT session is a combination of multiple sessions, each having their own main table. It is used to present data with a typical parent-child structure. For example, a Sales Order and the related Sales Order Lines.

The window of an MMT session consists of two parts:

- The multi-main table controller session is a maintain session for the main entity, such as a Sales Order. The form of this session determines the upper part of the session.
- The lower part displays satellite sessions with child information, such as Sales Order Lines. At runtime, the satellite sessions are shown as tabs.

See "Multi Main Table sessions" on page 229.

All Satellite Sessions

This field contains the records of all satellite sessions linked to the current multi-main table (MMT) controller session. Only a MMT controller session can have satellite sessions.

The satellite sessions are child sessions of the MMT controller session, such as the Order Lines session is a satellite session of the Order Header session.

The satellite session records show two attributes:

- The **Session** attribute (the session's name)
- The **Description** attribute

To add a satellite session:

1. Click **Add**.
2. In the **Session** field in the grid, press CTRL+SPACE to start a dialog where you can select satellite sessions.
3. Select the desired satellite session and click **OK**. The satellite session is displayed in the grid.

   You can only select sessions that have the **List Window** window type.
4. Select the satellite session in the grid and define the field mapping.

To remove a satellite session record, select a record and, on the shortcut menu, select **Remove**.

To change the order of the satellite sessions, move them up or down the list.

Name
The name of the satellite session.

Allowed values:

Press CTRL+SPACE and select a name from the select dialog. This dialog shows the names of all available sessions.

Description
The description of the satellite session.

Allowed values:

This field is read-only. LN Studio automatically fills in the satellite session description after you specified the **Name** of the satellite session.

Field Mapping

This field contains the field mapping records which map the fields of the multi-main table (MMT) controller session with the corresponding fields of its child satellite sessions. The records define a mapping for the satellite session marked in the **All Satellite Sessions** field.

The field mapping records consist of two attributes:

- The **Field in MMT Session** attribute, which specifies the name of the field in the parent MMT session
- The **Field in Satellite Session** attribute, which specifies the name of the corresponding field of the satellite session

For example, the Order Number field of an Order Header session (MMT session) corresponds with Order Number field of its child Order Lines sessions (satellite session).

To add a field mapping record, click **Add** and select the desired field names.

To remove a field mapping record, right-click a record and select **Remove**.

Note: The field mapping records define a mapping between the parent MMT session and one of its satellite sessions. You can compare such a field mapping with a foreign key of a record that points to the primary key of a referenced record. Similarly, the **Field in Satellite Session** field of the satellite session forms a pointer to the **Field in MMT Session** identification field of the main MMT session.

Field in MMT Session
The name of the field of the multi-main table (MMT) controller session.

Usually, the field of the main MMT session you specify here corresponds with the identification of the involved session records (business objects).

Allowed values:

To specify this attribute, select a field name from the drop-down list. This list contains the names of all available fields of the MMT controller session.

Field in Satellite Session
The name of the field of the satellite session.

Usually, the field of the satellite session you specify here corresponds with the identification field of the main MMT session.

Allowed values:

To specify this attribute, select a field name from the drop-down list. This list contains the names of all the fields of the satellite session that is marked in **All Satellite Sessions** field.

Documentation

Use this tab to enter the session's release notes and technical documentation.

Release Notes
Use this field to specify the release notes of the session.

Technical Documentation
Use this field to specify the technical documentation of the session

TDE Documentation

Use this tab to view an XML tree with the technical information on the session's Technical Data Entity.

This optional tab is displayed depending on the multipage editor preferences. See "Multipage Editors" on page 145.

Details

Implementing dynamic index switching

If an overview session's standard command set has the **Sort by** option selected, users can select different indices by which the records in the grid are sorted. The user selects an index from the **View** menu's **Sort by** submenu. When a user selects a new index, the view fields above an overview session's grid and the field order in the grid do not change.

Dynamic index switching allows you to configure session indices, so when a user selects a new index from the **Sort by** submenu, the view fields above the grid and the field order in the grid can change.

You can only implement dynamic index switching for sessions whose standard command set includes the **Sort by** option in the Form Option Sets (ttadv3120s000) session.

To implement dynamic index switching for an overview session:

1. On the **User Interface** tab of the LN Studio Session Editor, select the appropriate session's **Dynamic Index Switching** option.
2. On the **Overview** tab of the Session Editor, specify the session's **Table Indices and View fields** settings.

See the following sections for details.

Selecting dynamic index switching options

To start the Session Editor, double-click a session in the Activity Explorer. On the **User Interface** tab of the Session Editor, select either of the following **Dynamic index switching** check boxes:

- **Restart when view fields differ**. If switching indices results in the view fields above the grid changing, Enterprise Server restarts the session.
- **Always restart**. Users can switch indices in the Overview session, which always restarts the session.

To display the current index in the overview session's title bar, select the **Show name of index** check box.

Note: You can only select the previous **Dynamic index switching** check boxes if your session is integrated. For non-integrated sessions, use the predefined "dynamic.index.switching" variable in the before.program section of the session's program script:

- "dynamic.index.switching = -1" has the same effect as selecting the **Restart when view-fields differ** check box.
- "dynamic.index.switching = -2" has the same effect as selecting the **Always restart** check box .

Specifying index and view field settings

On the **Overview** tab of the Session Editor, select the session's indices for which you want to implement dynamic index switching. For every index check box you select, specify the order the indices display on the session's **Sort by** submenu and indicate the index fields that must be displayed as view fields.

- In the **Table Indices and View fields** field, determine the order the indices display on the session's **Sort by** submenu. To do this, use the **Up** and **Down** buttons to rearrange the sequence of indices. The first entry in the list will be the session's default index.
- In the **Table Indices and View fields** field, indicate which index fields must be displayed as view fields above the overview session's grid. To do this, select the check boxes for the desired index fields. When you select a particular index field as view field, the preceding index fields will be displayed as view fields.

Note: If a field has the **View** check box on the **Properties** tab of the Dynamic Form Editor's Field Properties dialog box selected, the field is always shown as a view field in the overview session, regardless of your Dynamic Index Switching settings. If you want a field to be excluded as a view field for certain indices, clear the **View** check box for that field on the **Properties** tab of the Dynamic Form Editor's Fields Properties dialog box.

Multi Main Table sessions

Introduction

A Multi Main Table (MMT) session is a combination of multiple sessions, each having their own main table.

MMT sessions are designed to present data with a typical parent-child structure, such as the following:

- Sales Order and the related Sales Order Lines
- Financial Business Partner group and the related Ledger accounts and Control accounts
- Work Order and the related Work Order activity lines, Work Order material resources, Work Order other resources, Work Order measurements, Work Order Hours accounting

Although this data is distributed among several main-tables in the system, it can all be presented and edited from within a single parent-child session.

This enables users to enter their data fast and to have a good overview of their work.

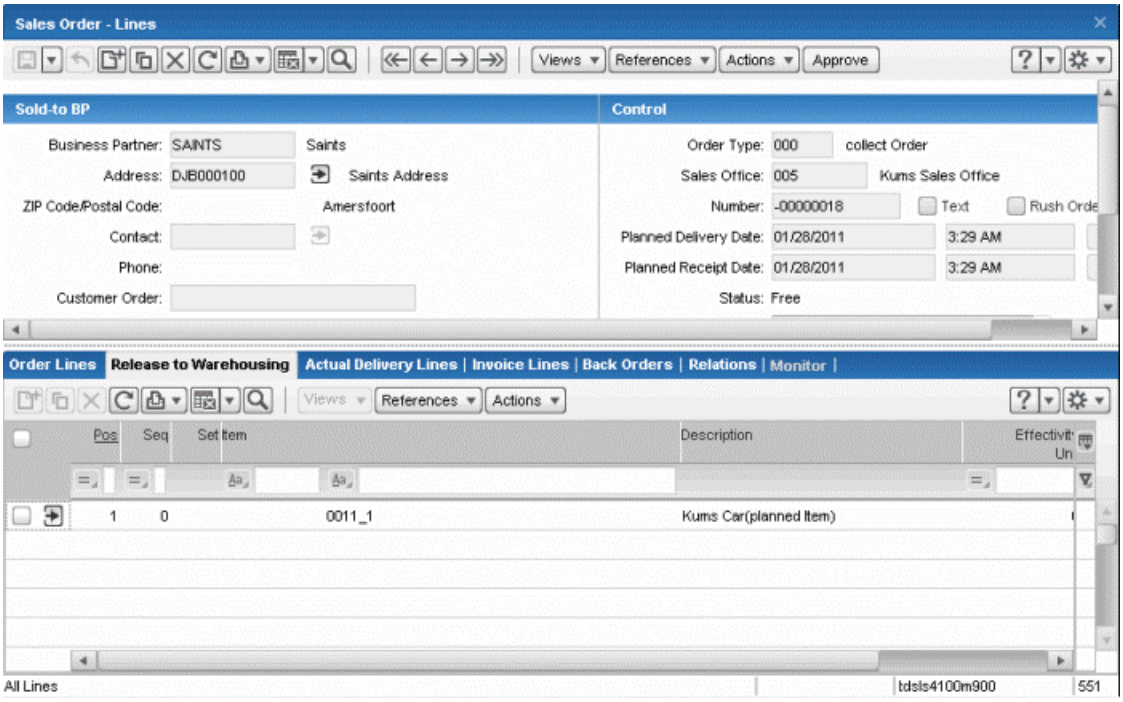
A typical parent-child session has two areas:

- The upper part contains information and functionality about the parent.

- The lower part contains one or more lists that show data about the various types of children.

Parent data and child data can be edited inside the session, which makes the sessions extremely powerful for data entry. A parent-child session gives the user a complete overview of all important data related to the object.

This figure shows an example of a parent-child session:



Sample parent-child sessions

Parent-child MMT sessions are available in various LN packages, such as the following:

Package	Session
Manufacturing	Production Order
Manufacturing	Generic Item – Structure
Order Management	Sales Order – Lines
Order Management	Sales Order Line – Deliveries
Service	Service Order – Lines
Service	Service Order Activity – Lines
Service	Maintenance Sales Order
Service	Work Order – Lines
Finance	Mapping Scheme
Finance	Ledger Mapping

Package	Session
Finance	Dimension Mapping
Finance	Financial Business Partner Group (Accounts Receivable)
Finance	Financial Business Partner Group (Accounts Payable)
Finance	Invoice-Source Relation
Quality Management	Inspection Orders – Easy Entry
Quality Management	Item – Storage Inspection
Object Data Management	Document Revisions – Easy Entry
Object Data Management	Change Proposal – Easy Entry
Tools	Table Definitions
Tools	Business Objects

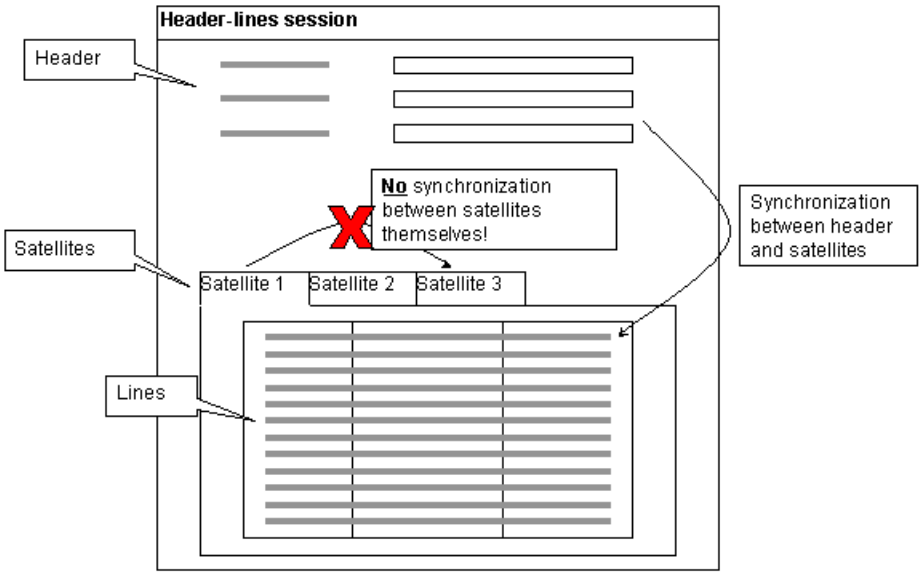
Header and satellites

A parent-child MMT session is an integration of multiple sessions into one single session. It consists of the following:

- A header session: This is the upper part of the MMT session window. It contains information and functionality about the parent, such as the Service Order, see the previous figure. The header is a single occurrence session.
- One or more satellites, displayed as tabs, containing data and functionality for the various types of children, such as for the activities, estimated materials and labor. Each satellite is a separate session linked to the MMT header session. All satellites are multi occurrence sessions.

The header and satellites can also run stand-alone.

This diagram shows the structure of an MMT session:



The header and lines in an MMT session are automatically synchronized. When you select a particular record in the header area, the corresponding lines are automatically displayed in the satellites. There is no synchronization between the satellites themselves.

A satellite tab only contains a grid. Additional fields outside the grid, such as view fields, are only displayed when the satellite runs stand-alone.

Example

When the Order Lines session runs stand-alone, the Order Number is displayed as a view field above the grid. When the session runs as a satellite in the Order MMT session, the satellite tab only displays the grid.

Developer guidelines

This section describes the prerequisites, possible issues, and the procedure for creating an initial MMT header-lines session.

Prerequisites

You can only create an MMT header-lines session if the following conditions are met:

- There is a clear header-lines relation. Enterprise Server only supports synchronization if each satellite is directly related to its header. For example, Sales Order Lines are directly related to the Sales Order Header through the Order Number field.
- No synchronization is required between the satellites.
- The header has a single-occurrence representation. If not, it should be created.
- All satellite sessions you want to define have multi-occurrence representations. If not, they should first be created.

Possible issues

Before you start creating an MMT header-lines session, identify the development effort required to solve the following possible issues:

Deep copy/delete	The header of the header-lines session contains the **Duplicate** and **Delete** standard commands. In an initial MMT session, these actions only execute on the header. Functionality to automatically copy or remove the corresponding lines when a header is copied or removed must be programmed separately. You can re-use existing code to do this.
Implicit saves	An implicit save means that a session and its contents are saved without a user action. Therefore, the user performs a manual "Save" action, such as clicking **Save** on the tool bar or selecting the **Save** menu item. In an MMT header-lines session, all data within each separate session is saved on each focus change from satellite to satellite, from header to satellite, or the other way around. For example, if a sales order is inserted and you switch the focus from the header to one of the satellites, the header is saved. To indicate a save is being executed, a save icon appears in the lower right part of the session window. If you modify data after the last implicit or explicit save and then close the session, you are prompted whether the changes must be saved. If you select **No**, this undoes the changes after the last implicit or explicit save. When programming, be aware of this behavior. It can cause unreferenced data in the database, such as an order header without lines. You might consider programming a way to remove this data directly when it is generated, or to build a session that cleans the database of these unreferenced entries.

Creating an initial MMT session

1 Create MMT header session

To create an MMT header session:

a Start the Create a New Infor LN Software Component wizard. Select **File > New > New LN Component ...**.

b On the first page of the wizard, specify the LN Studio activity in which you create the new component and the **Component Type**. The component type must be Session. To continue, click **Next**.

c On the second page of the wizard, provide the new session with a valid **Name**. Use the code of an existing package and module and a new session number not already in use.

d On the second page of the wizard, fill in the other session properties as follows:

- Select the Multi Main Table **Window Type**.
- For the MMT header session, specify a Details Session as **Synchronized Dialog**. For example, specify the Order Header details session as synchronized dialog for the Order MMT header session. Users can start this session through the **Details** command in the MMT header session.
- In the **Label of Object** field, select the label to be used in the header session's menu items. For details, see "Label of Object".

e To create the session and exit the wizard, click **Finish**.

2 Customize the MMT header session

- Edit the form of the MMT header session. Only include fields which are needed for entering a new record. All other relevant properties will be specified in the Synchronized Dialog in a separate window.
- Ensure the MMT session will fit into Web UI. Web UI supports 1024x768, but because the sessions are hosted within Web UI, they should be smaller. Try your session in Web UI to see if it fits. To maximize a session within Web UI, press ALT+F11. This will allow you to see the full session screen.
- The fields specified in the dynamic form of the MMT header session are displayed in the upper part of the MMT session, above the satellite tabs. By default, these fields should be read-only. They should only be accessible when a new record is inserted in the header session. To do this, use 'add.set' in the program script.

For details on form editing, refer to the Dynamic Form Editor online help. For details on program scripts, see the *Infor ES Programmer's Guide*.

3 Define Label of Object for each satellite

Edit the properties of each satellite session in the Session Editor. In the **Label of Object** field, select the label to be used in the satellite's tab description and menu items. For details, see "Label of Object".

4 Link satellites to MMT Header session

To link the desired satellites to the new MMT header session:

a Open the MMT header session in the Session Editor.

b Go to the **Satellite Sessions** tab and specify the satellites to be linked.

You can only link sessions that have the **List Window** window type.

Per satellite, enter the field relations used to synchronize the MMT header session with the satellite session. To enter a field relation, specify the name of the relevant field in the header's main table, followed by the name of the corresponding field in the main table of the satellite. For example, specify the Order Number field from the Order Headers table and the Order Number field from the Order Lines table.

5 Create/modify program scripts and DAL scripts

The MMT header session and satellite sessions are separate sessions. Therefore you must define a program script (UI script) for each of them, and optionally modify the DAL scripts of the corresponding tables. Other session components, such as Reports, Text fields and Indices must be defined per session.

When a satellite tab is activated for the first time, the corresponding session is started and its startup-sections, before.program, after.form.read, init.group.1, are called.

The program script of the satellite session can run in two modes:

Stand-alone mode	This is the ‘normal’ mode; used when the session is executed as stand-alone session.
Satellite mode	This is the special MMT mode. Use the `is.mmt.satellite()` statement to check whether a session is running as an MMT satellite session. You can program different behavior for the satellite mode if required.

In the satellite mode you can perform the following actions:

Hide fields	Hide certain form fields since they are not relevant in the satellite mode. However, there is only one form for both modes. You must define all fields you want to use in either one of them, in the form.
Enable/disable satellite	Enable or disable a satellite within an MMT session in a particular case. The tab of a disabled satellite is still visible in the MMT session, but cannot be selected or accessed. To disable a satellite, use the `disable.satellite()` function. To enable it, use `enable.satellite()`. Call these functions from the global `after.form.read` section in the header session's program script.
Hide/unhide satellite	Hide a particular satellite using the `satellite.invisible()` function. With this, you can still make the satellite visible at runtime. For example, you can check authorizations and hide/unhide satellites for specific users. You cannot hide/unhide satellites dynamically based on the availability of data in a satellite.

For details on program scripts and DAL, see the *Infor ES Programmer's Guide*.

Label of Object

For both MMT header sessions and satellite sessions, define the **Label of Object**. This label contains a generic description of the object within the session.

To define a **Label of Object** for a session:

1. Create a label.
   See "Creating a label" on page 102.
2. Link the label to the session: in the **Label of Object** field in the Session Editor, specify the label.

Label for Header session

For a Header session, define a single label. Note the following:

- The text should be a singular noun, such as "Purchase Order" for the Purchase Order MMT header session.
- The **Context of Label** must be **Session, Table or Report Description**.

The **Label of Object** is used in the Header session's menu-items to make clear on what object the menu-item-command executes. For example, the menu-item for inserting a new record in the Purchase Order MMT header session will be called **New Purchase Order**, instead of **New**.

Labels for satellite sessions

For a satellite session, you can create 2 labels with the same label code. One label is used in the satellite's menus, the other as the satellite's tab description.

Menu items

For the label used in the satellite's menu-items, note the following:

- The text must be a singular noun, such as "Order Line" for the Purchase Order Lines satellite session.
- The **Context of Label** must be **Session, Table or Report Description**.

This label makes clear on what object the menu-item-command executes. For example, the menu-item for inserting a new record in the Purchase Order Lines satellite session will be called **New Order Line** instead of **New**.

Tab description

By default, the session-description of the satellite is shown on the tab. When this description is not suitable, create a new label with these properties:

- The label code must be identical to the code of the other **Label of Object**.
- The text must be a plural noun, such as "Order Lines" for the Purchase Order Lines satellite session.
- The **Context of Label** must be **General Use**.

Session Model Editor

Introduction

Use this editor to modify an InContext model for a session.

A lot of sections in this editor are similar to sections in the Table InContext Model Editor. See "Table InContext Model Editor" on page 257.

Session

Use this tab to maintain the overall data of the InContext model.

General Information

Session
The session name and description.

To edit or view the session in the Session Editor, click the hyperlink.

Main table
The name and description of the session's main table, if applicable.

To edit or view the table in the Table Editor, click the hyperlink.

(Re)generate
Click this link to (re)generate the InContext model.

Alternatively, you can (re)generate the model through the (Re)generate Implementation Model (tticm1210m000) session.

Hooks

Include Hook/Declaration Hook/Function Hook
Click these links to open the corresponding hooks in the source viewer in the **Hook** section.

Library

Library
The model's library.

To edit or view the library in the Library Editor, click the hyperlink.

(Re)generate
Click this link to (re)generate the library.

Hook

Source viewer panel
This section is similar to the **Hook** section in the Table Editor. See "Table Editor" on page 238.

Context Messages

Use this tab to specify context messages and the corresponding mappings and hooks.

General Information

Session
The session name and description.

To edit or view the session in the Session Editor, click the hyperlink.

Main table
The name and description of the session's main table, if applicable.

To edit or view the table in the Table Editor, click the hyperlink.

Context Messages

Context Messages grid
This section is similar to the **Context Messages** section in the Table Model Editor. See "Table InContext Model Editor" on page 257.

These are the differences compared to the Table Model Editor:

- You can create more than one context message of the same type.
- The table column is empty for manually created context messages.
- The hidden column has only two options: true or false.

For context messages with the 'hidden' element set to true, a strikethrough font is used.

Mapping

Mapping grid
This section is similar to the **Mapping** section in the Table Model Editor. See "Table InContext Model Editor" on page 257.

Hook

Source viewer panel
This section is similar to the **Hook** sections in the Table Model Editor. See "Table InContext Model Editor" on page 257.

Besides table fields, you can also use all form fields in the hooks.

TDE Documentation

Use this tab to view an XML tree with the technical information on the session model's Technical Data Entity.

This optional tab is displayed depending on the multipage editor preferences. See "Multipage Editors" on page 145.

Table Editor

Table Editor

Introduction

Use this editor to define and maintain table definitions and the corresponding DAL scripts.

Tables are data structures used to store data that consists of a list of records, with each entry identified by a unique key and containing a set of related values. A table contains a number of table fields that belong to a specific domain.

Table definitions describe the record fields and the field domains. A table definition shows the following properties:

Table	The name of the table.
Label Name	The name of the label that contains the description of the table.

Description	The description of the table.
Update Ref. Message	The message that appears when you change a record referenced by other records.
Delete Ref. Message	The message that appears when you delete a record referenced by other records.
Relation Type	How table fields are mutually dependent.
Table Fields	The record attributes.
Table Indices	The index fields and their sequence.
Table Script	The Data Access Layer (DAL) script that applies to this table. A DAL script is used to authorize data access and to ensure data integrity.

See "Table definitions" on page 252.

Note: If you are not authorized to change the data model, you can only edit the table's DAL script. See the descriptions of the **Script** and **Source** tabs.

Overview

Use this tab to maintain the overall data of a table definition.

General Information

Table
The name of the table, used as table identification code.

A table is uniquely identified by the combination of package code, module code, and table code.

Allowed values:

The table name is read-only. You can only specify a table name when you create a new table definition. Afterwards, you cannot change this name anymore.

Table names must have the following format: `<package code><module code><table number>`

Package Code	2 letters	tt	(tools)
Module Code	3 letters	adv	(application development)
Table number	3-digit number	001	(sequence number of table)

Label Name
The name of the label that contains the table description.

To open the label in the Label Editor, click the hyperlink.

Allowed values:

The label name is read-only. You can only specify a label name when you create a new table definition. Afterwards, you cannot change this name.

Description
The description of the table.

Allowed values:

The description is read-only, because the description text is stored in a label.

To change the description, click the Label Name hyperlink and edit the label in the "Label Editor" on page 168.

Update Ref. Message
The code of the message which appears if a user tries to change the primary key of a record referred to by another table.

To open the message in the Message Editor, click the hyperlink.

Example:

Table pkitm001, Item table, is linked to Update Reference Message pkitmt0001. This message contains the following text:

```
Item code cannot be changed, item is in use.
```

Table pksls041, Sales order line, refers to the Item table by its item code. If a user tries to change an item code in the item table, which is used in a sales order line, LN displays the defined reference message.

Delete Ref. Message
The code of the message, which appears if a user tries to delete a record referred to by another table.

To open the message in the Message Editor, click the hyperlink.

Example:

Table pkitm001, Item table, is linked to Delete Reference Message pkitmt0001. This message contains the following text:

```
Item code cannot be deleted, item is in use.
```

Table pksls041, Sales order line, refers to the item table by its item code. If a user tries to delete an item code in the item table, which is used in a sales order line, LN displays the defined reference message.

Relation Type
The relation type for the fields in this table.

Note: This indication is important for delayed lock facilities in the bshell. If you use a delayed lock, two users can change one record at the same time. However, they cannot change the same fields or the fields that are related to each other.

Allowed values:

Select one of the following relation types:

No relations	All fields of the record are independent. When a user edits a field, another user can edit other fields of the record.

All Fields	All fields of a record depend on other fields of this record. When a user edits a field, all fields are locked so other users cannot edit the record.
Specified Relations	You can define groups of related fields. Some fields of a record depend on other fields of this record and some fields are independent. When a user edits a field, the related fields are locked. The other fields can be edited by other users.

InContext Model
The name of the InContext model that describes which fields are required for the context messages for this table.

To open the InContext model in the Table InContext Model Editor, click the hyperlink.

If no InContext model exists for the table, click the hyperlink to generate a model. The Table InContext Model Editor is started.

Related topics:

- "InContext Modeling" on page 75
- *Infor Enterprise Server InContext Modeling Development Guide*
- "Table InContext Model Editor" on page 257

Field Relationships

This section contains the Field-Group records. This indicates which table fields belong to which field groups.

Fields can be arranged into groups to indicate their mutual relationships.

To add a Field-Group record, click **Add** and specify a group and a field.

To remove a record, right-click a record and select **Remove**.

Note: This section is only displayed if the **Relation Type** is Specified Relations.

Group
The identification of a *group* of related fields.

Allowed values:

A number in the range 0-9999.

Name
The name of the field, which is related to other fields with the same group number.

Allowed values:

Select a field from the list.

Description
The description of the field.

The description is read-only. It is filled automatically when you select a field.

Runtime

Convert: Convert Tables to Runtime
To start the Convert to Runtime Data Dictionary (ttadv5215m000) session, click the Convert hyperlink. Use this session to convert domains and tables to the runtime data dictionary.

This option is only enabled for checked in components.

Note: In the Convert to Runtime Data Dictionary (ttadv5215m000) session, select the package combination that contains the *Project VRC* of your current *Software Project*. The session converts domains and tables in all *Activities* linked to the software project. Therefore, the session does not only convert tables in your own *Activity VRC*, but converts changed tables in activity VRCs of other developers as well.

Read the session's online help before you start the conversion.

Create: Create Tables
To start the Create Tables (ttaad4230m000) session. Click the Create hyperlink to create this table for a range of companies.

Fields

Use this tab to maintain the table fields of the table definition and do the following:

- Enter the field name and field number of the table field. The field name must be unique in the current table.
- Specify whether the field is combined. For example, is there a combination of two or more fields of the same table?
- You must specify a domain that specifies the data type, legal values, validation check, display format and so on.
- Specify the label data for the field, the initial value, the mandatory input, and whether it is a repeating field.
- Specify a relation to another table, insert check, delete, and update mode and specify whether the field is active.

All Table Fields

List of all table fields.

In the **All Table Fields** list, you can perform the following actions:

- Add or remove table fields.
  - To add a field, click **Add** and enter the basic field properties in the grid. Then, enter the remaining properties in the **Table Field Properties** field group.
  - To remove a field, right-click a field and select **Remove**.
- To change the field sequence of the table, click **Up** and **Down** to move fields up and down the list.

- Link child fields to parent fields. To link a child field to its parent (combined) field, drag the child field on the parent field and release the mouse button. The parent and child fields are displayed as a tree structure.
- Copy fields to an index in the **Indices** tab, or copy fields from the current table to another table. To copy fields to another table, select the fields, right-click, and select **Copy**. Then, right-click a field in the **All Table Fields** list of the other table and select one of these commands:
  - **Paste as Fields**: The fields are added below the selected field.
  - **Paste as Children**: You can only select this command if the selected field is a combined field. The fields are added as child fields of the combined field.

Name
The name of the field.

You can only enter a field name when you add a field. For existing fields, the name is read-only.

Label
The code of the label used as a description of the table field. Press CTRL+SPACE and select a label from the select dialog box.

To open the label in the Label Editor, click the label icon.

For new table fields, the label is generated when you save the table.

Description
The description of the field.

You can only enter a description when you add a field. LN Studio stores the description in a label.

Allowed values:

For existing fields, the **Description** field is read-only, because the description text is stored in a label.

To change the description, click the label icon behind the **Label** field and edit the label in the Label Editor.

Domain
The domain of the table field. Press CTRL+SPACE and select a domain from the select dialog box.

Click the domain icon to open the domain in the Domain Editor.

Hooks
Click the **Hooks** icon to view a list of hooks defined for this field in the table's DAL script.

Table Field Properties

Column

Name
The name of the field that can be used in forms, reports, and scripts in the following way: <table name>.<field name>

Allowed values:

The field name is an ASCII text with a maximum length of 8 characters. The field name must be unique in the table.

Field names are usually abbreviated by leaving out syllables or vowels in the description of a particular property, such as `dscr` for description or `cnty` for country.

Example table : pksls040 (package - module - table number) field name: orno

Label Name
The name of the label used as a description of the table field.

To start a dialog where you can select another label, click **Browse**.

To open the label in the Label Editor, click the hyperlink.

Description
The description of the table field, as defined by a label with the same name as the table field. If a label in a form, report, or menu has the same name as the table field, this description will replace the label.

Note: A percentage symbol % in the label description forces a line break in the label in the form, report, or menu.

Allowed values:

The description is read-only, because the description text is stored in a label.

To change the description, click the Label Name hyperlink and edit the label in the Label Editor.

Combined Field
If this check box is selected, the field is combined. A combined field consists of two or more normal fields, not combined fields.

- A combined field can be used for references and indices.
- To add a child field to a combined field, go to the **All Table Fields** field group and drag the child field on the combined field.

Mandatory Input
If this check box is selected, input in this field by the user is mandatory.

If the field has the byte, integer, long, float or double data type, mandatory input means the value of the field must be unequal to 0. Mandatory input is an addition to the domain definition of the field and has precedence over what is defined as the range of the domain. If the user tries to make a field empty while input is mandatory, Enterprise Server gives range message of the domain.

Example		
Field:	Employee number	(datatype long)
Range	Mandatory input	Allowed range on the field
$$ IN [0, 999999]	no	0 - 999999
$$ IN [0, 999999]	yes	1 - 999999
$$ IN [1, 999999]	no	0 - 999999 (0=empty)

$$ IN [1, 999999]	yes	1 - 999999

Elements
The depth of a repeating field.

If you specify a value greater than 1, the field is repeating. A repeating field occurs a number of times in the table with the same name and characteristics. Identification of an element is done using a sequence number.

Example: The Help Topics table contains the following fields:

Field name	Elements
Help Topic Code	0
Topic Title	0
Topic Keywords	4

"Topic Keywords" is a repeating field of 4 levels. In the Help Topics table, you can define four key words for each Help Topic Code.

Note: You can only specify the depth when you add a field. You cannot change the depth of existing fields.

Allowed values:

A number in the range 0-99.

To indicate the field is not repeating, enter 0.

To indicate the number of elements for a repeating field, specify a value greater than 1.

Domain Name
The name of the domain of the table field.

To open the domain in the Domain Editor, click the hyperlink.

Allowed values:

Select a domain from the field's drop-down list. The list contains all domains available on the current LN server.

Data Type
The data type of the domain.

Allowed values:

This field is read-only. The data type is determined by the specified domain.

Length
The length of the domain.

Allowed values:

This field is read-only. The length is determined by the specified domain.

Generate Unique Default Value
This check box indicates whether a unique value must be generated for the selected table field during reconfiguration of the table.

If you select this check box:

- The **Default Value** field becomes disabled, showing no content.
- In the database, the value "$__unique" is stored as the "Initial Value".
- The value "$__unique" is used to start a bdbreconfig. It generates a unique ID for the selected field.

If this check box is cleared (default value):

- The **Default Value** field becomes enabled.
- The default value for **Default Value** is empty. You can manually specify a default value.
- The value of **Default Value** is stored in the database.

Default Value
If a user adds a record to the table, this is the initial value for the field.

If the **Generate Unique Default Value** check box is cleared, you must specify a default value.

If the **Generate Unique Default Value** check box is selected, this field is read-only. The default value is determined by the specified domain.

Field Dependency Editor
To start the Field Dependency Editor, click this hyperlink. This editor enables you to view and modify the field dependencies defined in the table's DAL 2 script. For details on DAL (2) scripts and field dependencies, see the *Infor ES Programmer's Guide*.

Reference

Reference Table
The related table to which this field refers.

Note:
The reference table is also known as the parent table.

To open the reference table in the Table Editor, click the hyperlink.

Allowed values:

To start a dialog where you can select a reference table, click **Browse**.

Reference Mode
Indicates whether or not the referenced field must exist in the reference table. The reference field is the field referred to by the current field.

Allowed values:

Select one of these reference modes:

Mandatory	The foreign key value (the reference record) must exist in the reference table.

Mandatory unless empty	Same as **Mandatory**, but the foreign key value can be empty.
Not mandatory	The foreign key value will not be checked against the reference table.
Not Applicable	The field does not refer to a field of another table.

Reference Message
The message code which LN gives, if you enter a value in an input field in the form, where no related record in the reference table is present.

Note: The reference message is only relevant if the reference mode is mandatory, or mandatory unless empty.

To open the reference message in the Message Editor, click the hyperlink.

Allowed values:

To start a dialog where you can select a reference message, click **Browse**.

Delete Mode
The action to be taken by LN if a referenced record in the reference table is deleted.

Allowed values:

Select a delete mode from the drop-down list.

Update Mode
The action to be taken by LN if the primary key of a referenced record in the reference table is changed.

Allowed values:

Select an update mode from the drop-down list.

Technical Doc.

Use this field to specify the technical documentation of the table field.

Release Notes

Use this field to specify the release notes of the table field.

Indices

Use this tab to maintain the table indices and index fields of the table definition.

You can perform these actions:

- Add indices and index fields:
  - To add an index, click **Add Index** and enter the index properties in the grid. See the field help.
  - To add an index field, select an index in the grid and click **Add Child**. Then, in the **Name** field, select the desired field from the list.

Alternatively, right-click a field in the **Fields** tab and select **Copy**. Then, go back to the **Indices** tab, right-click an index and select **Paste**.

- Remove indices and index fields. To remove an index or index field, right-click the index or field and select **Remove**.
- To determine the index sequence, use the **Up** and **Down** buttons to move the index records up or down the list. The index numbers change correspondingly. The index with number 1 is the primary key.

All Table Indices

All table indices. Each index is represented by a record.

Name
The number of the index or the name of the index field.

Note: The index with number 1 is the primary key.

Description
The description of the index or index field.

You can only enter a description when you add an index. LN Studio stores the index description in a label.

Label
The label that contains the description of the index.

This field is read-only.

To open the label in the Label Editor, click the label icon.

Note: For new indices, the label is generated when you save the table.

This field does not apply to index fields.

Active
If this check box is selected, the index is active. The primary index (index number 1) is always active.

Note: This information is not used in the data dictionary or the database layer.

This field does not apply to index fields.

Unique
If this check box is selected, the index is unique. Therefore, different records must have different values in the index parts.

If this check box is cleared, different records can have the same values in the index parts. These are called duplicate records.

Note: This field does not apply to index fields.

Script

Use this tab to specify the properties of the table's DAL script. You can also link libraries to the table.

Data Access Layers (DALs) are program scripts to guarantee data integrity. The DAL script considers for all types of dependencies, various business rules, specific constraints and so on. For more information on DALs and the availability of two versions of these scripts, see "DAL" on page 253.

To edit or view the DAL script, go to the **Source** tab. For more information on the LN Studio Script editor, see "Script Editor" on page 263.

For more information about the script properties, see the **Script** tab of the session editor.

General Information

Script Type
The type of program script used for this table. The script type is DAL or DAL2.

Allowed values:

Select one of the following options:

DAL	Data Access Layer that contains the Business Logic related to a table. The DAL is called by the 4GL engine on certain events. Since version 2 of DAL is available, you are advised to use DAL (version 2) scripts.
DAL (version 2)	Data Access Layer that contains the Business Logic related to a table. The DAL is called by the 4GL engine on certain events. DAL (version 2) is more integrated with the Business Data Entity (BDE) interfaces and the integration software.
Not Applicable	No DAL script is used for this table.

DAL
The name and description of the DAL *script*.

Usually, a script is uniquely identified by the package code, module code, and script code. The code of a DAL script corresponds with the name of the table for which the script is used.

Compile Flags
The compile flag.

For details, see the **Script** tab of the Session Editor.

Tools Interface Version
The Tools Interface Version (TIV) number of the script/library.

For details, see the **Script** tab of the Session Editor.

License

Product ID Source
The product ID for which a license is needed to compile the object. If the field is empty, no license is needed to compile the object.

Note: This field only applies to internal Infor development systems. For other development environments, this field is not available.

Allowed values:

To enter a product ID, click **Browse** and select an ID from the select dialog box.

Product ID Object
The product ID for which a license is needed to run the object. If the field is empty, no license is needed to run the object.

Note: This field only applies to internal Infor development systems. For other development environments, this field is not available.

Allowed values:

To enter a product ID, click **Browse** and select an ID from the select dialog box.

Linked Libraries
This list contains the records of all libraries linked to the DAL script of the current table.

Sorting

- To sort the libraries by position, name, or description, click the corresponding column header.
- When you sort by position, the sort mode is always ascending. When you sort by name or description, click the involved column header to toggle between ascending and descending sort mode.
- When you sort by name or description, the **Up** and **Down** commands are disabled.

Add / remove libraries, change linking order
You can only perform the following actions if you sort the libraries by position.

- To add a library, click **Add**. Then, press CTRL+SPACE in the **Name** field and select a library from the select dialog box.
- To remove a library, right-click a library and select **Remove**.
- To change the linking order of the libraries, move them up or down the list.

Note: To edit a library, click the icon behind the library name.

Pos
The library sequence number (Position). This field indicates the linking order of the libraries.

This field is read-only. It is filled automatically when you link a library.

Name
The name of the library.

Allowed values:

To specify this attribute, press CTRL+SPACE and select a library from the select dialog box.

Description
The description of the library.

Allowed values:

This field is read-only. LN Studio automatically fills in the library description after you specified the **Name** of the library.

Documentation

Release Notes
Use this field to specify the release notes of the table's DAL script.

Technical Documentation
Use this field to specify the technical documentation of the table's DAL script.

Source

Use this tab to view and edit the source code of the DAL script of the table. The tab consists of a script editor, which forms an integral part of the table editor.

See "Script Editor" on page 263.

This tab is only displayed, if the source code of the DAL script is available.

Documentation

Use this tab to enter the table definition's release notes and technical documentation.

Release Notes
Use this field to specify the release notes of the table definition.

Technical Documentation
Use this field to specify the technical documentation of the table definition

TDE Documentation

Use this tab to view an XML tree with the technical information on the table's Technical Data Entity.

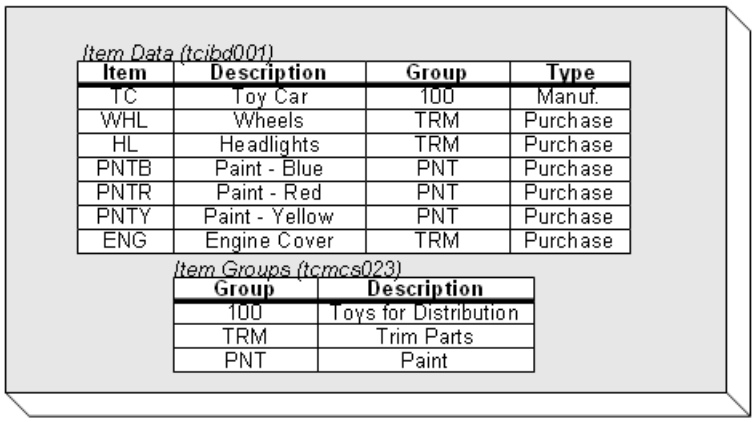
This optional tab is displayed depending on the multipage editor preferences. See "Multipage Editors" on page 145.

Details

Table definitions

Table Definitions define a structure to the data. The table code and name are attributes of a table. A table is a collection of data, arranged in rows (records) and columns (fields).

This diagram shows a table data example:



Fields, domains and indices

A table has fields. Table fields store individual pieces of data such as a customer name, the quantity of an item ordered, or the date a journal entry was made.

Table fields are linked to domains. Domains are components that define common information about data such as the following:

- The data type, such as a character type for customer name, a number type for quantity, and a date type for journal date
- Valid ranges
- Special characteristics such as capitalization rules

Domains ensure consistent data types for similar table fields.

This diagram shows a table fields and domains example:

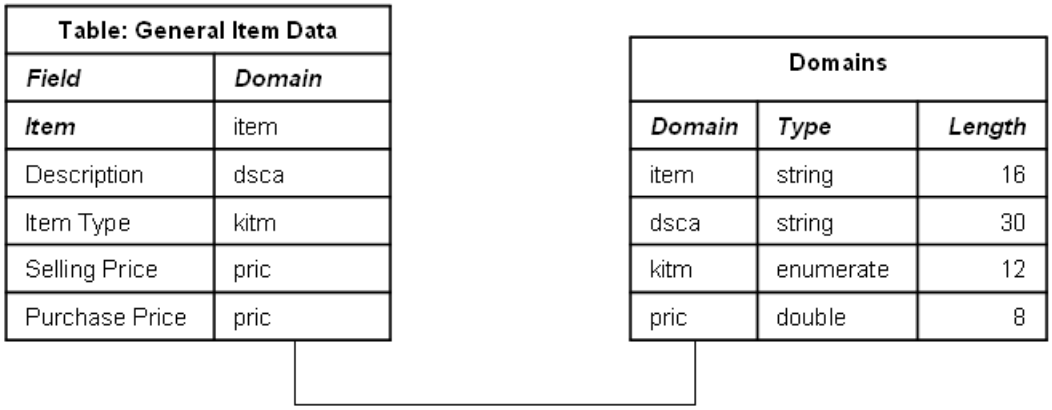


Table: General Item Data	
Field	*Domain*
Item	item
Description	dsca
Item Type	kitm
Selling Price	pric
Purchase Price	pric

Domains		
Domain	*Type*	*Length*
item	string	16
dsca	string	30
kitm	enumerate	12
pric	double	8

A table must have at least one index. An index consists of one or more table fields used to sort and search records in the table. The first index is always the Primary key, the unique identification for a record in a table.

Related Tables and References

The table may have a related table. A related table means that a field in the table will refer to the key field of another table. Therefore, data can have relationships: customers can have orders; inventory stores items in warehouses; and employees work in a department. To create diagrams from these defined relationships, use a tool in the Enterprise Modeler .

Creating table definitions

Create table definitions in the Create a New Infor LN Software Component wizard and the Table Editor. See "Creating a table" on page 111.

DAL

Data Access Layer (DAL)

The Data Access Layer allows developers to describe rules about data. A DAL is linked to a table, not a session. Therefore, anytime the table is accessed, the DAL is used. Therefore, different sessions can update a table and the same rules apply in the same way. In addition, integration capabilities use the DAL to ensure updates occur in the same way.

Data Access Layer scripts implement the business rules of the application to ensure the logical integrity of the database. DAL scripts are a Library software component aligned with a specific table. These scripts are compiled into objects. When a session performs an action on a table, the DAL for the table is loaded.

In LN, two types of Data Access Layer scripts exist: DAL 1 and DAL 2.

Business Rules

A business rule is a statement that defines, or constrains, some aspect or operation of the business. It is intended to control or influence the behavior of the business operations. Business rules are logical integrity rules, and differ from database oriented referential integrity rules.

Business Rules - example

Business rules in a purchasing system could be as follows:

- An item can be ordered only from its supplier
- When the gross weight and net weight for an item are the same, there is no packaging
- The supplier provides the arrival date of a purchase order after the order is entered into the system.
- Items cannot be removed if there is stock on-hand
- The item on-order quantity is increased when an order for the item is entered
- The item on-hand quantity is increased, and the on-order quantity decreased, when an order arrives

Referential Integrity Rules - example

In contrast, integrity rules in a purchasing system could be as follows:

- An item must exist for it to be ordered
- If an item has a supplier, the system must know the supplier
- Items can not be deleted if the item record is used elsewhere in the system

Business rules are implemented as hooks. A hook is a pre-defined function name you create in a Data Access Layer script. The 4GL engine will call this function as it executes user interface and database related actions on the table the DAL is written for.

Business rules fall into categories that determine how the business rule is implemented in the DAL.

The categories, depending on the type of DAL script, are as follows:

- Property or Field hooks
- Object Hooks
- Field Dependencies
- Business Methods

DAL 1

DAL 1 scripts were introduced in Infor Baan 5.0. A DAL 1 script contains database logic such as logical integrity rules, but it usually does not contain all business logic for the table it belongs to. Very often, business logic is spread across the DAL 1 script and the UI scripts.

Most UI scripts still contain business logic to do the following:

- Make fields read-only.
- Determine which enum constants can be selected in an enumerated field.
- Update a field after a change to a related field.

Therefore, it is difficult to use the DAL 1 scripts for integrations via the LN Business Adapter software: the business logic in the UI scripts is skipped if another application connects to an LN table. Therefore,

this logic must be rebuilt in the Business Object Interfaces (BOIs) that provide the connection. If you change the business logic in the UI scripts, you must also adapt the corresponding BOIs.

A DAL 1 script can contain the following types of hooks:

Object hooks.	These hooks operate on the row or record as a whole unit. Object hooks determine if operations on a row are allowed, and specify additional operations that should be performed based on an operation of the row. Examples of object hooks are as follows: • Delete the additional employee data in the People package when an employee is deleted. • When a sales order line is saved, create a history record of the sales order line. • Update the item inventory when a planned stock transaction is created.
Property hooks.	These hooks operate at a field level, and are used to verify valid values and calculate default values for table fields. Examples of property hooks are as follows: • Validity check that the Quotation expiration date should be after the Quotation entry date. • When adding an employee, the language of the company is the default for the employee’s language.

For details on DAL 1 scripts, refer to "Data Access Layer" in the *Infor ES Programmer's Guide*.

To create DAL 1 scripts

Create and edit DAL 1 scripts in the Table Editor. For DAL 1 scripts, the script type is "DAL".

DAL 2

DAL 2 scripts are introduced in LN. A DAL 2 script contains all business logic related to a particular table. The UI scripts of the sessions operating on the table only contain UI logic.

DAL 2 scripts are suited for integrations via the LN Business Adapter software. Other applications can connect to LN tables through Business Objects. The business logic in the table's DAL 2 script is executed automatically. To build an integration, you do not need to rebuild this business logic in the BOIs that provide the interface.

Various predefined Business Objects, that can be used for integration purposes, are delivered with LN. DAL 2 scripts are only available for tables used in these business objects. For the remaining tables, DAL 1 scripts are available. Refer to the LN Business Adapter documentation for details on business objects and integration technology.

In DAL 2 various new hooks are introduced. A DAL 2 script can contain the following:

Object hooks	These hooks are identical to the object hooks in DAL 1 scripts.
Field hooks	These hooks contain the business logic for table fields and can contain logic to do the following:

	• Verify valid values on a table field (logical integrity rules). • Make a field read-only. For example, the contact field of the Business Partner is not a required entry, and is a display field in the UI when the Business Partner is created. • Make a field invisible. • Automatically update a field, after a related field was changed. See "Field Dependencies". • Make input on a field mandatory. • Hide a particular enumerated constant from an enum list.
Business method hooks.	These hooks disable/enable and hide specific *form commands* of type "business method".

For details on DAL 2 scripts, refer to "DAL 2 Overview" in the *Infor ES Programmer's Guide*.

Field dependencies

In DAL 2 scripts, you can define field dependencies used to determine default values. when these field dependencies are defined, dependent fields update themselves when the fields they depend on are updated. The field updates are defined in the DAL as a field hook.

Dependencies are also used to update the User Interface. The 4GL Engine uses these field dependencies to disable and enable fields, and determine enum values.

Examples of field dependencies are as follows:

- The packing field of the Item is yes if the gross weight and net weight of the item are different. A field dependency is defined so the packing field is updated if the gross or net weight fields change.
- An item without a gross weight cannot have a net weight and packing. A field dependency is defined so the net weight and packing fields are disabled if the gross weight is zero.

Business methods

Business Methods are operations on a row or set of rows within a table. A business method is a function that performs a task that involves manipulating or checking one or more tables in the database.

Sessions, object hooks, or property hooks from another table’s DAL can call a business method. For more information, refer to the *Infor ES Programmer's Guide*.

Examples of business methods are as follows:

- The sales order line session can remove an order line. The object hook for deleting the sales order line calls a business method to also remove the unpaid commissions.
- In the Purchase Order session, check if it is allowed to change a purchase order header based on the purchase order status and the business partner status.
- If a purchase order issues components, the object hook for the purchase order line calls a business method to issue the components for the purchase order. The business method inserts items of the bill of material into a purchase order components table.

Table InContext Model Editor

Introduction

Use this editor to modify an InContext model for a table.

Table

Use this tab to maintain the overall data of the InContext model.

General Information

Table
The table name and description.

To edit or view the session in the Session Editor, click the hyperlink.

Dependent InContext Models
This link shows the dependent Table and Session Models in separate view below the editor. You can use that view to refactor dependent models.

See "Dependent InContext Models view" on page 314.

(Re)generate
Click this link to (re)generate the InContext model.

Alternatively, you can (re)generate the model through the (Re)generate Implementation Model (tticm1210m000) session.

Drillback

Drillback session
The session that will be started when the business object is referenced and must be displayed.

To edit or view the session in the Session Editor, click the hyperlink.

Click **Browse** to view all sessions where the main table is the same as the table of the current model.

From Enterprise Server version 10.3 you can define the session code through a hook. To add a hook, click the Hook link. To remove a hook, clear the hook code. The hook must return a String.

Mode
Indicates whether the drillback session will start as single or multi occurrence session, assuming that the session supports the selected mode. Default value is Single Occ.

From Enterprise Server version 10.3 you can define the mode through a hook. This hook must return either SINGLE_OCC or MULTI_OCC.

Index
The index the drillback session will start with.

From Enterprise Server version 10.3 you can define the index through a hook. This hook must return a Long.

If the session code is specified using a hook, the index must also be specified through a hook.

If you leave this field blank, the default index, 1, is used.

Referenced Tables

Referenced Tables grid
The grid shows all referenced tables from the model. If a hidden element is present in the model and has the value 'true', a strikethrough font is used. The descriptions are read from the server. If the same table occurs multiple times as referenced table, the sequence number element is used to make the descriptions unique.

The 'hidden' check box toggles the 'hidden' element of the referenced table.

The grid synchronizes with the **Table Reference** section.

To create a row in the grid and add a referenced table, click **Add**. This field supports code completion. The table is validated and the description is added. These default settings are generated for the table reference:

- Reference type: where
- Mandatory: false
- Field mapping: all primary key fields of the referenced table that are also present as fields in the current model table.

To remove a table from the grid, right-click the table and select **Remove**. This is only possible for referenced tables that are not present in the active Reference Model.

Include Hook/Declaration Hook/Function Hook
Click these links to open the corresponding hooks in the source viewer in the **Hook** section.

Table Reference

Type
The reference type.

The TableRead Hook hyperlink is only active in case of Type 'hook'. If the Type is changed, the Mappings are generated following the rules.

Mandatory
If this check box is selected, the referenced table must be present.

Field Mapping grid
This mapping defines how the current table is related to the referenced table. The **From** fields must exist in the current table. The **To** fields must exist in the referenced table.

To create a row in the grid and add a mapping, click **Add**. Code completion is possible on both, **From** and **To** fields.

To remove a mapping from the grid, right-click the mapping and select **Remove**.

Use the **Up** and **Down** buttons to change the position of the mapping, although this does not have any impact.

Hook

Source viewer panel
Use this viewer to edit the selected hook.

This editor supports all LN Studio Script Editor features, such as code completion, text hover tool-tips, open declaration, and syntax highlighting. See "Script Editor" on page 263.

In the hooks, you can use all table fields from the current table and all referenced tables.

Click the Previous hyperlink to move to the previous hook in the model.

Click **Next** to move to the next hook in the model.

Click **Compile** to generate a source with the three generic hooks (Include, Declaration and Function) and a wrapper around the current hook. The Include and Declaration hooks themselves are compiled solely.

Context Messages

Use this tab to specify context messages and the corresponding mappings and hooks.

General Information

Table
The table name and description.

To edit or view the session in the Session Editor, click the hyperlink.

Context Messages

Context Messages grid
The grid shows the context messages from the model.

The **Hidden** column can have these values:

- **No**: the context message will be sent for this table and is also available for the tables that refer to this table.
- **Current level**: the context message will not be sent for this table, but is available for the tables that refer to this table.
- **Child level**: the context message will be sent for this table but is not available for the tables that refer to this table.
- **All**: the context message will neither be sent for this table nor will be available for the tables that refer to this table.

For context messages with the 'hidden' element set to true, a strikethrough font is used.

Click **Add** to add a context message to the grid. Only messages that are present in `tagencontextmessages.xml` are allowed. Code completion is supported. For context messages that are added manually, you cannot specify the table and sequence number.

To remove a context message from the grid, right-click the message and select **Remove**. This is only possible for context messages that are not present in the active Reference Model.

The **Refresh** button performs these actions:

- Refresh the context messages for all referenced tables.
  - Read the context messages from the referenced model.
  - Compare them to the present ones related to that table.
  - Add/delete if required.
  - Change only the context messages that are not manually changed.
- This refresh is also executed on save if a referenced table has been added. You must confirm this; the save will not be performed if you do not choose to refresh. If this referenced table has a table model, the context messages are copied from this model. If there is no model, the context messages are copied from the reference model. Note that the sequence number makes the context message unique if the same context message is linked to the same table multiple times.

The grid synchronizes with the **Mapping** section.

Mapping

Mapping grid

This editable grid shows the Mapping elements from the model.

This table shows the functionality required for the columns in the grid:

Column	Description
Target	These values come from the model; they are the placeholders in the XML template of the context message.
Type	The type of mapping. This table shows the possible values for this column:

Type	Description
field	The placeholder can be replaced directly with the value of the field.
none	There is no mapping; the element that refers to this mapping in the XML template of the context message will be skipped when the message is constructed.
hook	A hook will calculate the value for the placeholder. Selecting the row automatically loads the hook into the hook editor.
function	A function is called; the return value of the function will be used for the substitution.

<table>
<tr><th>Column</th><th>Description</th></tr>
<tr><td>Source</td><td>The possible values for this column depend on the type of mapping specified in the Type column. This table shows the possible values for the Source column:
<table>
<tr><th>Type of mapping</th><th>Value for Source column</th></tr>
<tr><td>field</td><td>A table field from the current table</td></tr>
<tr><td>none</td><td>Empty</td></tr>
<tr><td>hook</td><td>Empty</td></tr>
<tr><td>function</td><td>A function from the Infor ES Programmer's Guide, Functions hook, or from a library that has been linked to the model by “pragma used dll” in the Include hook.</td></tr>
</table>
</td></tr>
</table>

Condition Hook/Before Mappings Hook/After Mappings Hook
Click these links to open the corresponding hooks in the source viewer in the **Hook** section.

Hook

Source viewer panel
This section is similar to the **Hook** section on the Tables page.

The **Inherit** check box indicates whether the hook type is 'inherit' or 'override':

- If you select this check box, the current hook is deleted and the corresponding hook from the referenced table is displayed. The hook editor is read-only.
- If you clear this check box, the hook is copied and can be edited. The new hook overrides the logic of the inherited hook.

If there is no hook to inherit, the **Inherit** check box is disabled.

TDE Documentation

Use this tab to view an XML tree with the technical information on the table model's Technical Data Entity.

This optional tab is displayed depending on the multipage editor preferences.

See "Multipage Editors" on page 145.

Script Editor

11

The LN Studio script editor is used to edit the source code of UI scripts, libraries, DALs, functions, and report scripts.

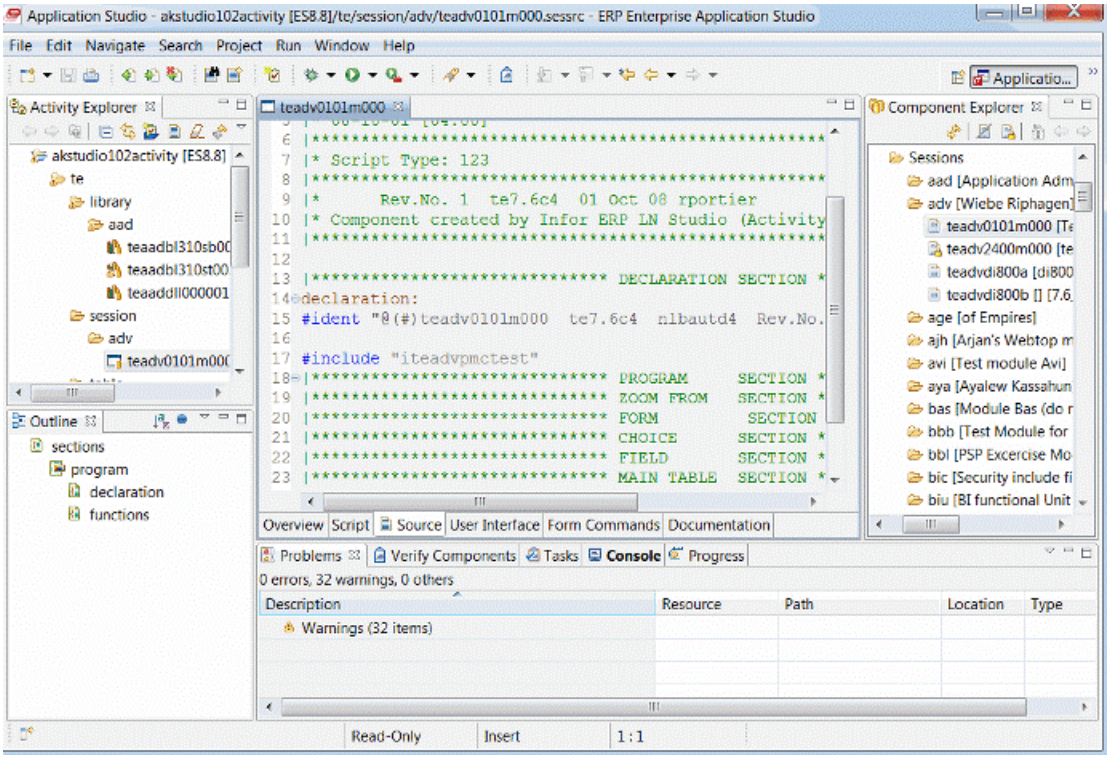
The script editor is embedded in the **Source** tab of these multipage editors:

- "Function Editor" on page 166
- "Library Editor" on page 174
- "Report Editor" on page 190
- "Session Editor" on page 205
- "Table Editor" on page 238

You can edit multiple software components simultaneously. Each component is displayed in a separate tab.

Associated with the script editor is an Outline view, which shows the structure of the active source code. The Outline view is updated automatically when you edit the source code.

See the following figure for an example.



Editing source code

To edit the source code of a component, go to the **Source** tab of the corresponding multipage editor:

- To edit the source code of a function, open the function in the Function Editor and go to the **Source** tab.
- To edit the source code of a library, open the library in the Library Editor and go to the **Source** tab.
- To edit the source code of a report script, open the corresponding report in the Report Editor and go to the **Source** tab.

  Note: You can also start a separate script editor to set breakpoints in the report's debug script. See "Using breakpoints" on page 70.

- To edit the source code of a UI script, open the corresponding session in the Session Editor and go to the **Source** tab.
- To edit the source code of a DAL script, open the corresponding table in the Table Editor and go to the **Source** tab.

Note: The **Source** tab is only displayed if the source code is available.

The editor displays error markers to indicate errors in the source code. When you save a component which contains errors, the component is only stored in your local workspace but is not saved to the server. In the Activity Explorer, the component has a dirty state ( ) decorator in the bottom right corner

of the software component image. When you solve the error and save the component, the component is saved to the server again and the decorator disappears.

Alternative ways to open source code

You can also start the script editor in the following ways:

- Through the **Open Declaration** command. See "Open Declaration" on page 285.
- From various views, such as the Problems view, the Tasks view and the Bookmarks view. For example:
  - When you double-click a problem, task, or bookmark associated with a specific component, the editor opens the corresponding source.
  - When you double-click a problem, task, or bookmark associated with a specific line in a specific component, the editor opens the corresponding source on that line.

  Note: The editor is not started when you double-click a generic problem, task, or bookmark not associated with any specific component.

  Instead of double-clicking, you can use the **Go To** command on the view's shortcut menu.

Script editor features

This topic focuses primarily on the LN Studio specific features. It does not describe the standard Eclipse editor functionality, such as the **Cut**, **Copy**, **Paste**, **Delete**, and **Save** commands.

For details on the standard Eclipse editor features, see the *Workbench User Guide*.

For details on the LN Studio specific editor features, see these sections:

- "Comparing Source Code Versions" on page 275
- "Source Tab" on page 267
- "Comments" on page 279
- "Open Declaration" on page 285
- "Keyword Search" on page 283
- "Code Assist" on page 276
- "Incremental Find" on page 283
- "Generating Source from WSDL" on page 291
- "Folding" on page 281
- "ToDo Comments" on page 290
- "Toggle Breakpoints" on page 290
- "Text Hovering" on page 289
- "Application Search" on page 275
- "Mark Occurrences" on page 284
- "Quick Outline view" on page 288

- "Outline view" on page 321
- "Executing Assignment Expressions" on page 70(only during debugging)

Note: All features listed are available if you start the editor in the Application perspective. When you debug software components, the editor starts automatically in the Debug perspective. During the debugging, only a part of the editor's functionality is available.

For information on preference settings for the editor, see "Script Editor Preferences" on page 266.

For information on the limitations of the editor, see "LN Studio Limitations" on page 77.

Script Editor Preferences

You can define various preference settings for the Script Editor.

LN Studio preferences

Use these pages to define LN Studio-specific preferences:

- Preferences - Editor

  Use this page to define various editor preferences. You can specify the "Marker Identifier" used in maintenance comments, and define background color settings for the editor. You can also enable Mark Occurrences, and indicate whether the editor must highlight matching brackets.

- Preferences - Syntax

  Use this page to define different font colors for the various sections in the 4GL scripts and libraries.

- Preferences - Code Assist

  Use this page to define Code Assist settings, such as background color and foreground colors for completion proposals, and the auto activation trigger character.

- Preferences - Folding

  Use this page to define the preference settings for Folding. For example: you can select the region types for which folding will be enabled, and you can specify whether folding must be enabled when you open a new editor.

- Preferences - Templates

  Use this page to define templates for various 4GL programming statements, such as `case`, `for`, `repeat` and `while`. You can use Code Assist to include these templates in your scripts or libraries.

To access these preference pages:

1 Select **Window > Preferences**.
2 In the left pane of the Preferences window, select **Infor LN Studio Application > Editor**.

Standard Eclipse preferences

There are various pages you can use to define standard Eclipse preferences.

For example:

- Use the Preferences - Editors page to define various preferences. For example: You can specify the size of the recently opened files list, and indicate whether you wish to show multiple editor tabs.
- Use the Preferences - Annotations page to specify where and how the different annotation types (such as error markers, warning markers, and task markers) are displayed in the editor. For example: you can specify per annotation type whether markers must be displayed in the text or in the overview ruler.
- Use the Preferences - Quick Diff page to configure settings for the Quick Diff feature. Quick Diff automatically adds color-coded change indicators in the marker bar while you are typing. In the preference page you can, for example, define colors for the change indicators.
- Use the Preferences - Text Editors page to define appearance options for the editor. For example: You can specify whether the current line must be highlighted, and whether line numbers must be displayed on the left side of the text editor.

To access these preference pages:

1 Select **Window > Preferences.**

2 In the left pane of the Preferences window, select **General > Editors**.

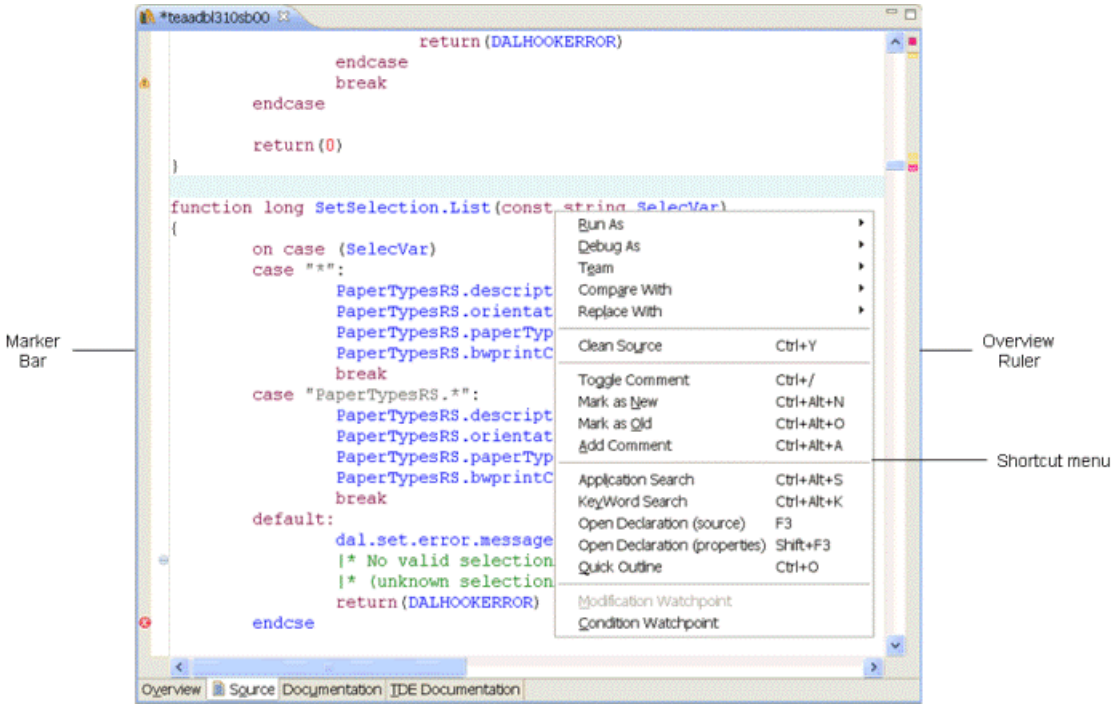
See the *Workbench User Guide* and the online help of the pages mentioned.

Note: Alternatively, you can access the standard Eclipse text editor preference pages through the **Preferences** command in the shortcut menu of the script editor's marker bar.

Source Tab

The LN Studio script editor is embedded in the **Source** tab of various multipage editors. In this tab, you can edit the source code of UI scripts, libraries, DALs, functions, and report scripts.

This figure shows the source tab and its shortcut menu:



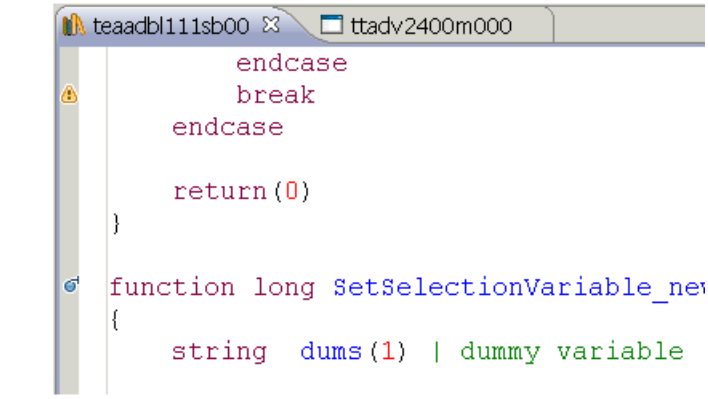
Markers and Marker bar

A marker is like a yellow sticky note stuck to (a line in) a script or library. On a marker you can record information, such as location and severity, about a problem or a task to be done. Or you can simply record a location for a marker as a bookmark.

The Marker bar is the vertical bar located at the left of the editor area.

This bar is used to display various types of markers, for example: error markers, warning markers, breakpoint markers, task markers, and so on.

This figure shows an example:



Each marker is related to the line next to which it is displayed. When you hover over a marker, the description of the corresponding error, warning, breakpoint, etc. is displayed in a ToolTip. For example, when you hover over the warning marker in the previous figure, the following description is displayed: `"Statement not reached"`.

The following markers can appear in the marker bar:

Icon	Description
	Bookmark
	Task
	Search result
	Error
	Warning
	Line breakpoint - enabled
	Line breakpoint - disabled
	Method breakpoint - enabled
	Method breakpoint - disabled
	Modification watchpoint - enabled
	Modification watchpoint - disabled
	VSC (Verify Components) warning

For details on breakpoints and on the corresponding breakpoint markers, see these sections:

- "Using breakpoints" on page 70
- "Breakpoints view" on page 306

For details on VSC warnings, see "Verifying software components" on page 58.

For details on the other marker types, refer to the *Markers* topic in the *Workbench User Guide*.

Note:

- The markers are not only displayed in the marker bar, but they are also shown in various views, such as the Problems view, the Breakpoints view and the Tasks view. From these views, you can quickly jump to the marked location within the corresponding source. For example, when you double-click a problem in the Problems view, the script editor opens the corresponding source on the involved line.
- The appearance of the different markers in the marker bar depends on the settings in the Annotations preferences page. You can access this preference page via the Preferences dialog.

  See "Script Editor Preferences" on page 266.

Marker bar shortcut menu

Right-click the marker bar to display the marker bar's shortcut menu. This menu contains the following commands:

Command	Description
Add /Remove Bookmark Add /Remove Task	Adds or removes a bookmark or task for a line. To define a bookmark or task for a specific line in your source code, right-click on the marker bar next to that line, and click the relevant command on the context menu. A dialog appears where you can enter the bookmark or task properties. When finished, a new bookmark or task marker appears in the marker bar. **Note:** You can also view (and modify or remove) the new bookmark or task in the Bookmarks view or in the Tasks view. To remove a bookmark or task, right-click the corresponding marker in the marker bar, and click the relevant **Remove** command on the shortcut menu.
Show Quick Diff	Enables / disables the QuickDiff feature. QuickDiff automatically adds color-coded change indicators in the marker bar while you are typing. The colors show additions, deletions, and changes to the editor buffer as compared to a reference, for example, the contents of the file on disk. When you hover the mouse cursor over a change indicator in the marker bar, the original content is displayed. You can configure QuickDiff settings, such as colors for the change indicators, in the Quick Diff - Preferences page. You can access this preference page via the Preferences dialog. See "Script Editor Preferences" on page 266 and the *Workbench User Guide*. **Note:** This command is not only used to enable the QuickDiff feature, but is also used to disable QuickDiff.
Show Line Numbers	Toggle to display line numbers.
Show Mark Occurrences	Enables/disables the Mark Occurrences feature. See "Mark Occurrences" on page 284.
Highlight Matching Brackets	Enables/disables the highlighting of matching brackets in the source code. If this option is selected, the corresponding bracket is highlighted automatically when you place the cursor directly behind a bracket. This functionality applies to the following types of brackets:

Command	Description
	• Parentheses: (,). • Braces: {, }.
Show Whitespace Characters	Enables/disables the whitespace characters.
Preferences	Opens the Eclipse Text Editors preferences page. See the *Workbench User Guide*. **Note:** To change the LN Studio script editor preferences, start the Preferences - Editor page. See "Setting user preferences" on page 36.
Folding/Collapse All	Collapses all expanded code elements such as functions. See "Folding" on page 281.
Folding/Expand All	Expands all folded code elements such as functions. See "Folding" on page 281.
Toggle Breakpoint	Adds or removes a line breakpoint for the current line. If the current line is part of a function header, the command adds or removes a method breakpoint. **Note:** • When you define a method breakpoint, the corresponding marker is always displayed next to the line that contains the word "function". • Alternatively, you can double-click in the marker bar to add or remove a line breakpoint or a method breakpoint. • You can also use the Breakpoints view to disable, enable, or remove breakpoints. See "Using breakpoints" on page 70.

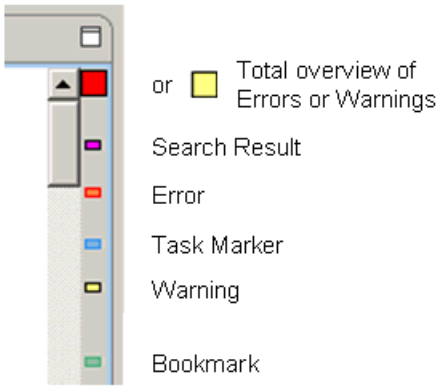
Note: The following marker types cannot be added via the marker bar:

Error and warning markers	These markers are system-generated. You can view the corresponding error and warning details in the Problems view.
Search result markers	These markers are generated by the search command. The corresponding search result details are displayed in the Search view.

Overview ruler

If enabled, the overview ruler is displayed at the right hand side of the editor area, and shows all markers concerning the entire document.

See the following figure for an example:



The markers are shown relative to their position in the document and do not move as you scroll the document.

You can click a marker to navigate to the corresponding location in the source code. When that portion of the document is visible, there usually is a corresponding marker on the marker bar at the left hand side of the editor area.

Note: The appearance of the different markers in the overview ruler depends on the settings in the Annotations preferences page. You can access this preference page via the Preferences dialog.

For more information, see the "Script Editor Preferences" on page 266 section.

Editor shortcut menu

When you right-click on any location in the source code, a shortcut menu is displayed.

This table describes the shortcut menu commands, except basic commands, such as copy and paste:

Command	Description
Compare With	Contains commands to compare different versions of the source code. See "Comparing Source Code Versions" on page 275.
Clean Source	Opens a cleaned version of the source in a separate editor. In the cleaned source, all code marked as old is removed: The script editor removes the following: • Lines that contain the "|#<Current ActivityName>.o" comment. • Blocks where the first line contains the "|#<Current ActivityName>.so" comment and the last line contains the "|#<Current ActivityName>.eo" comment. In code marked as new, the script editor removes the corresponding comments: • "|#<Current ActivityName>.sn" • "|#<Current ActivityName>.en" • "|#<Current ActivityName>.n"

Command	Description
	For details on code marked as old, and code marked as new, see "Comments" on page 279. **Note:** • The cleaned source is read-only. • When you close the cleaned source, the corresponding file is removed from the workspace.
Toggle Comment / Mark as New / Mark as Old / Add Comment	Use these commands to comment out and uncomment lines of code. You can toggle comments, mark modified lines, mark new lines, and add comments. See "Comments" on page 279.
Add Linked Library	Use this command to link a library to the component that you are editing. Use this command as follows: **1** In the source code, select a function call. The selected function must be stored in a library. The function name must meet the coding standards. For example, the source contains this code: `RETIFNOK(tcmcs.dll0095.read.parm("tdsls000"))`. Select `tcmcs.dll0095.read.parm("tdsls000")`. **2** Right-click and run this command. The library in which the selected function is stored, for example tcmcs.dll0095, is added to the list of linked libraries in the **Script** tab.
Generate Source from WSDL	Use this command to generate LN 4GL code, based on a URL or a Web Services Description Language (WSDL) file, for a general library. See "Generating Source from WSDL" on page 291.
Application Search	Use this command to search in which LN software components the expression, that you selected in the script editor, is used. See "Application Search" on page 275.
Keyword Search	Performs a keyword search in the online help. See the "Keyword Search" on page 283 section.
Open Declaration (source)	Use this command to view the source code of the selected software component. LN Studio opens the **Source** tab of the corresponding multipage editor. See "Open Declaration" on page 285.
Open Declaration (properties)	Use this command to view the properties of the selected software component. LN Studio opens the **Overview** tab of the corresponding multipage editor. See "Open Declaration" on page 285.

Command	Description
Quick Outline	Lists the main structural elements of the source code you are editing. You can use the displayed elements to quickly navigate to the corresponding sections in the source code. See "Quick Outline view" on page 288.
Modification Watch-point	Adds a modification watchpoint for the selected variable or table field. A watch expression is added automatically. When debugging, the session execution suspends each time the variable or table field changes. Important: You must be sure that the selected text really is a variable or table field. The script editor does not check this. See "Using breakpoints" on page 70.
Watch	Creates a watch expression for the selected variable. When debugging, the watch expression is evaluated each time a session suspends. See "Evaluating expressions" on page 69.
Condition Watchpoint	Starts a dialog where you can enter a condition watchpoint for a local or global variable. A watch expression is added automatically. When debugging, the session execution suspends when the variable gets a certain value. See "Using breakpoints" on page 70.

Keyboard shortcuts

This table shows keyboard shortcuts you can use in the script editor:

Keyboard shortcut	Description
Ctrl+.	Jump to next error line.
Ctrl+,	Jump to previous error line.
Ctrl+Shift+Up Arrow	Select complete word or string.
F3 after selecting a table field	Select a table field in the source code and press F3 to start the Table Editor. The **Fields** tab is opened automatically and the properties of the selected table field are displayed.

Details

Comparing Source Code Versions

In the script editor, you can compare different versions of the source code that you are editing.

To compare two source code versions, right-click in the source code and select one of these options:

- **Compare With > Project**
  Compares the current version of the source code with the version stored in the *Project VRC*.
- **Compare With > Activity**
  Compares the current version of the source code with the version stored in the *Activity VRC*.

When you run these commands, the Text Compare window is displayed. This window shows both versions of the source code side by side. The differences between the versions are highlighted.

Application Search

You can use the **Application Search** command to find out in which LN software components an element, selected in the script editor, is used.

The application search functionality depends on the Enterprise Server version of the LN server.

Procedure for servers with Enterprise Server 8.6 and lower

To find out in which LN software components an element is used:

1. Select the desired element in your source code. For example, select a domain code, a message code, or a function call.
2. Right-click the selected element and, on the shortcut menu, select **Application Search**. The Search Script items (tlanl0400m000) session starts.
   The first time you use **Application Search**, you must define appropriate defaults for this session. To do this, select all types of components and enter the full range of packages, scripts, functions and reports. Subsequently, click **Save Defaults**, close the dialog, and run the **Application Search** command again.
3. Specify the desired selection ranges and parameters and click **Continue**. The Select Device (ttstpsplopen) session starts.
4. Select an output device and print the search results.

Procedure for servers with Enterprise Server 8.7 and higher

To find out in which LN software components an element is used:

1. Select the desired element in your source code. For example, select a domain code, a message code, or a function call.
2. Right-click the selected element and select **Application Search**. The Search dialog box starts.
3. In the Application Search tab, specify the desired selection ranges and parameters and click **Search**. See "Application Search" on page 139. The search results are displayed in the Search view.
4. Double-click a search result to open the corresponding script or library at the line where the result occurs.

   If the script or library is not present in your work area, it is automatically retrieved from the LN server and stored in your work area.

Code Assist

When you enter text in the editor, Code Assist can provide you with a list of suggested completions for partially entered strings.

To use code assist:

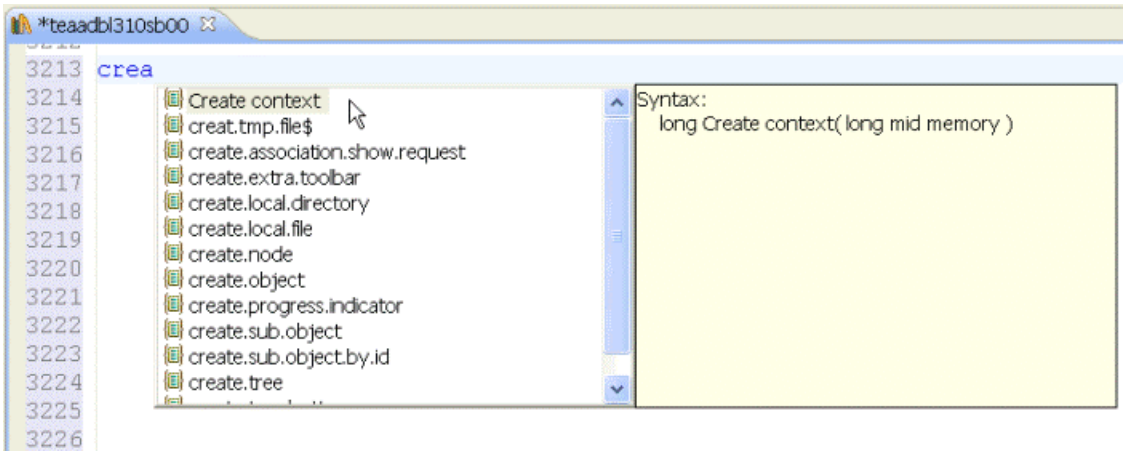
1. Press **Ctrl+Spacebar** directly after the last character of the string you want to complete.

   A list of completion proposals is displayed. This list can contain:

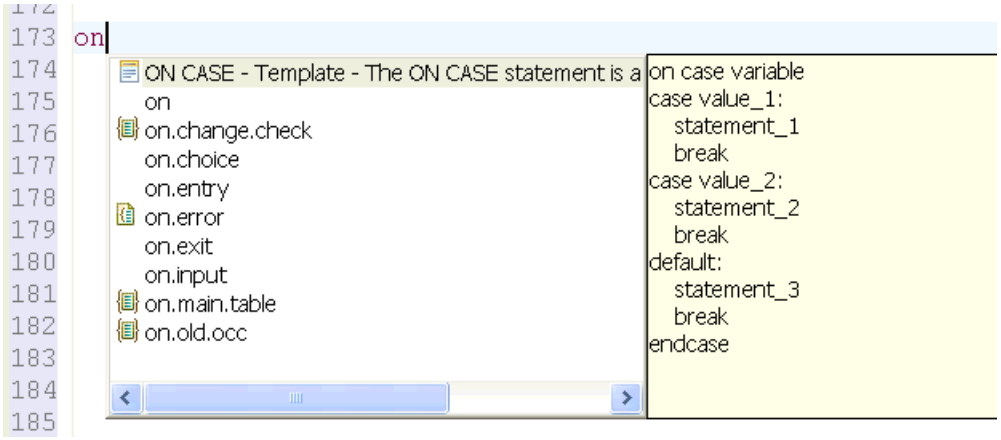
   - Various types of 4GL elements, such as functions, predefined variables, keywords, and program sections. These are read from the *Infor ES Programmer's Guide*.
   - Templates for programming statements, such as the `on case` statement and the `while` statement. These templates are defined in the Preferences - Templates dialog.
   - Functions that are declared in the source you are editing.
   - Functions declared in includes and libraries that are used in the source you are editing.
   - Domains.
   - Enumerated constants.
   - Tables.
   - Table fields.
   - Index fields in the where clause of a select statement.
   - External functions in the DAL that is selected in the dal.start.business.method function. You must press **Ctrl+Spacebar** after the comma behind the first argument of the dal.start.business.method function.
2. Double-click the completion proposal you want to use. The completion proposal, for example a function name, or a template text, is added to your source code.

See the following figures for some examples .

The following is displayed when you start Code Assist after typing "crea" (functions from *Infor ES Programmer's Guide*):



The following is displayed when you start Code Assist after typing "on":



When you double-click the `on case` statement in the previous figure, the template is inserted in the editor:

```
on case variable
case value 1:
        statement 1
        break
case value 2:
        statement 2
        break
default:
        statement 3
        break
endcase
```

Note: After inserting a template in the editor, press Tab to navigate through the template's variables.

Icons

Code Assist can display suggestions for various types of 4GL elements. The following icons are used to distinguish the different element types:

Icon	Description
	Function
	Predefined variable
	Choice section
	Program section
	Template

Note:

- Code Assist can also display suggestions for keywords, SQL statements, some standard macro definitions, and various sub sections. However, these types are displayed without icons.
- Code Assist does not display report main sections and `form.<form number>` sections.

Preferences

You can define various preferences for Code Assist, such as the following:

- Automatic insert of completion proposals
- Background color for the completion proposal
- Foreground color for the completion proposal
- Auto activation trigger character (default is the dot (.) sign). Code Assist is started automatically when you type this character.
  - Important: the auto activation trigger character is used as part of the search string. For example, when you type "aud.", Code Assist shows a list of functions whose names start with "aud." (including the dot sign!); for example "aud.audit.is.on.for.table" and "aud.close.selection". Function names starting with "aud" (without the dot sign), such as "aud_close_audit" and "aud_get_audit_dd", are not displayed.

To define preferences for Code Assist:

1. Select **Window > Preferences**. The Preferences dialog box is displayed.
2. In the tree structure in the left pane of the dialog box, select **Infor LN Studio Application > Editor > Code Assist**. The Preferences - Code Assist dialog box is displayed. See the dialog box's online help for details.

Code Assist Templates

LN Studio contains predefined templates for various programming statements, such as `on case`, `for`, `repeat` and `while`. You can use Code Assist to include these templates in your scripts or libraries. See the previous figures for an example on the `on case` statement.

You can also add you own templates and use them in the same way as the predefined templates.

To add your own templates:

1. Select **Window > Preferences**. The Preferences dialog box is displayed.
2. In the tree structure in the left pane of the dialog box, select **Infor LN Studio Application > Editor > Templates**. The Preferences - Templates dialog box is displayed.
3. Click **New** to add a template. A details dialog box is displayed.
4. In the **Name** field, enter the name for the template. This name does not have to be unique. It is used as a display name only.
5. In the **Description** field, enter a brief description of the template.
6. Enter the template pattern in the **Pattern** field. The pattern may contain pre-defined variables. See "Preferences - Templates" on page 118.
7. Click **OK** to return to the Preferences - Templates dialog.
8. Click **Apply** to save the new template.
9. Click **OK** to close the Preferences dialog.

You can now use Code Assist to insert the template in your scripts and libraries.

Comments

The LN Studio Script Editor enables you to easily comment out and uncomment lines of code. You can toggle comments, mark modified lines, mark new lines, and add comments.

Toggling comments

Use the **Toggle Comment** command to (un)comment one or more lines:

1. Put the cursor in the desired line, or select the desired lines.
2. Right-click and select **Toggle Comment**.

When you comment lines, a comment pipe symbol [|] appears at the beginning of each of the selected lines. This prevents the lines from being compiled or interpreted.

This figure shows an example:

```
|group.1:
|init.group:
|       if background then
|               disable.commands(ADD.SET)
|       endif
```

The comment pipe symbols are removed when you uncomment the lines.

Maintenance comments

You can easily add maintenance comments to existing scripts or libraries. You can mark existing lines that you modified, and new lines that you added.

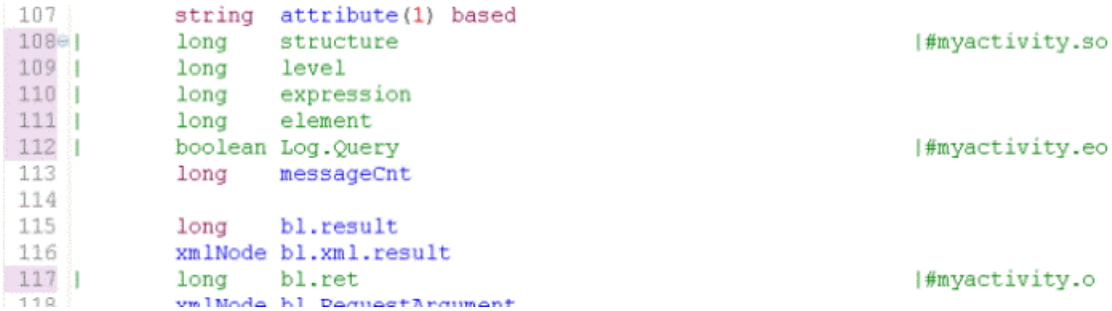
When you add a maintenance comment, the editor generates a comment text that consists, among other things, of a Marker Identifier. The Marker Identifier is determined by the editor preferences, and can be the name or requirement ID of your current activity or project. See "Preferences - Editor" on page 114.

To mark modified lines

You can also easily mark modified script parts:

1. Put the cursor in the desired line, or select the desired lines.
2. Right-click and, on the shortcut menu, select **Mark as Old**.
   - All selected lines are commented out (a pipe symbol [|] appears at the beginning of each line)
   - Behind the first old line a comment is inserted with the following text: " `|#<Marker Identifier>.so` ". (" `so` " means Start - Old)
   - Behind the last old line a comment is inserted with the following text: " `|#<Marker Identifier>.eo` ". (" `eo` " means End - Old)
   - Note: if you selected only one line, the comment text is: " `|#<Marker Identifier>.o` ".

This figure shows an example:

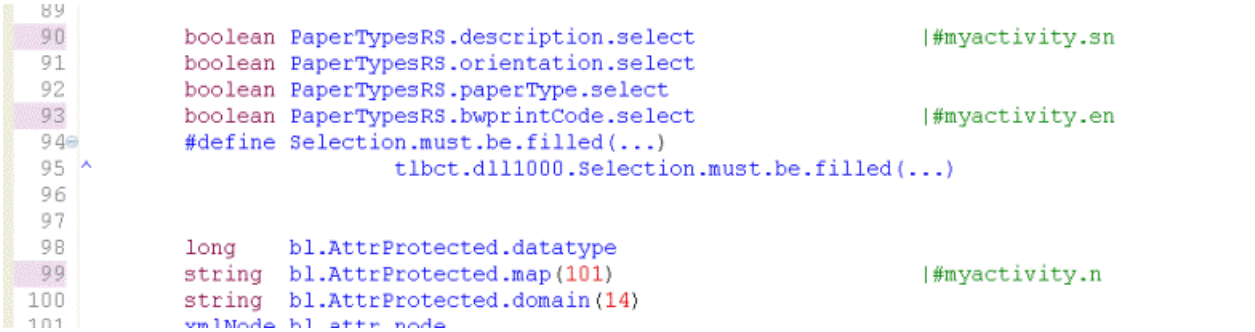


Marking new lines

You can also easily mark new lines that you added to your script:

1. Put the cursor in the desired line, or select the desired lines.
2. Right-click and, on the shortcut menu, select **Mark as New**.
   - Behind the first new line a comment is inserted with the following text: " `|#<Marker Identifier>.sn` ". (" `sn` " means Start - New)
   - Behind the last new line a comment is inserted with the following text: " `|#<Marker Identifier>.en` ". (" `en` " means End - New)
   - **Note:** If you selected only one line, the comment text is: " `|#<Marker Identifier>.n` ".

This figure shows an example:



Note: When you use the **Toggle Comment** command to uncomment a line, only the pipe symbol [|] at the beginning of the line is removed. Comment text at the end of the line (for example maintenance comment text such as " `|#myactivity.so` ") is not removed.

Adding comments

You can use the **Add Comment** command to read descriptions, such as domain and table descriptions, from the data dictionary on the LN server. These descriptions are added as comments in your script.

To add these comments to your script:

1. Right-click and select **Add Comment**.
2. A message is displayed: you are prompted to save changes and to perform the action. Click **OK** to add the comments.

This sample code shows table names that were added through the **Add Comment** command:

```
table tttaad300 |* Device Data
table tttaad305 |* Report Defaults
table tttaad310 |* Paper Types
table tttaad311 |* Fonts by Paper Type
table tttaad312 |* Windows Paper Size
```

Folding

In the script editor you can fold and unfold code elements (regions). When you fold a code element, the element is collapsed so that the corresponding source code is hidden. The folding of elements can provide a better overview of the script structure, especially in large scripts.

The script editor supports folding of the following region types:

- functions
- header comments. Note: only header lines with a double *, for example lines starting with “|**”, are folded.
- macros
- sections

To fold a region, click the minus sign that is displayed at the beginning of the region. A plus sign is displayed at the beginning of the folded region.

This figure shows an example:

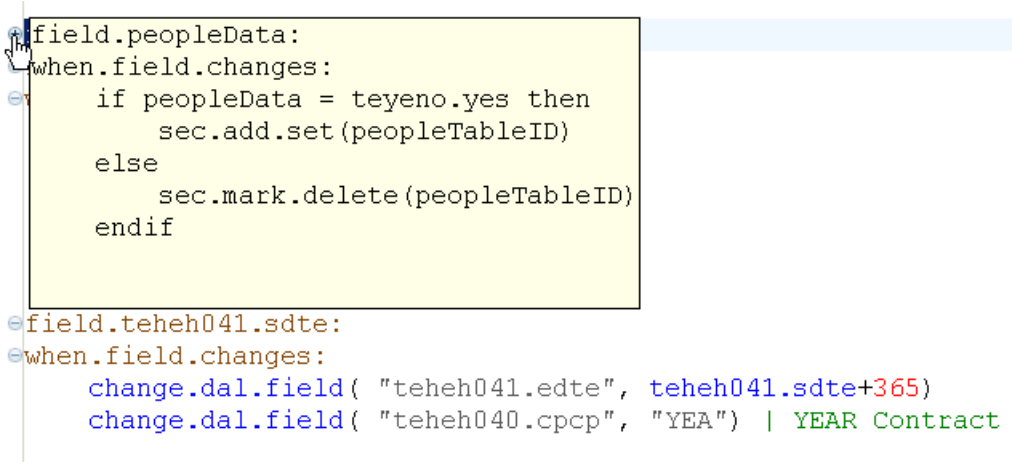
```
field.peopleData:
field.projectData:
when.field.changes:
    if projectData = teyeno.yes then
        sec.add.set(projectTableID)
    else
        sec.mark.delete(projectTableID)
    endif

field.teheh041.sdte:
when.field.changes:
    change.dal.field( "teheh041.edte", teheh041.sdte+365)
    change.dal.field( "teheh040.cpcp", "YEA") | YEAR Contract
```

To unfold the region again, click this plus sign.

When you hover over a folded 4GL element, the hidden code is displayed.

This figure shows an example:



Note: You can unfold all folded code elements in one go: right-click the marker bar (located at the left of the editor area) and select **Collapse All**.

Preferences

You can define various preferences for Folding.

To define these preferences:

1 Select **Window > Preferences**. The Preferences dialog box is displayed.

2 In the tree structure in the left pane of the dialog box, select **Infor LN Studio Application > Editor > Folding**. The Preferences - Folding dialog box is displayed.
3 Define the preference settings, for example:

- Specify whether folding must be enabled when you open a new editor.
- Select the region types for which folding will be enabled.

See the dialog's online help.

Incremental Find

You can use the **Incremental Find Next** and **Incremental Find Previous** commands to search for expressions in the active editor. As you type the search expression, the cursor incrementally jumps to the next or previous exact match in the active editor.

To use the **Incremental Find Next** command:

1 Press CTRL+J in the script editor or select **Edit > Incremental Find Next**.
2 Type the expression you are searching for. The search expression is displayed in the Workbench status line. As you type, the editor finds the next exact match of the text and updates the selection after each character typed.

For example:

- when you type the character "a", the cursor jumps to the next exact match of the character "a".
- When you subsequently type an "s", the cursor jumps to the next exact match of the string "as"
- When you subsequently type a "c", the cursor jumps to the next exact match of the string "asc"

Note: to undo the last action, press Backspace.

3 Use the Up Arrow and Down Arrow keys to navigate between matches.
4 To cancel the search, press one of the following keys: Left Arrow, Right Arrow, Enter or Esc.

The **Incremental Find Previous** command works in a similar way. When you type an expression, the editor finds the previous exact match of the text and updates the selection after each character typed. To start this command: press Ctrl+Shift+J in the script editor or, on the **Edit** menu, select **Incremental Find Previous**.

Keyword Search

In the script editor you can select a piece of text, and perform a keyword search to get more information on it.

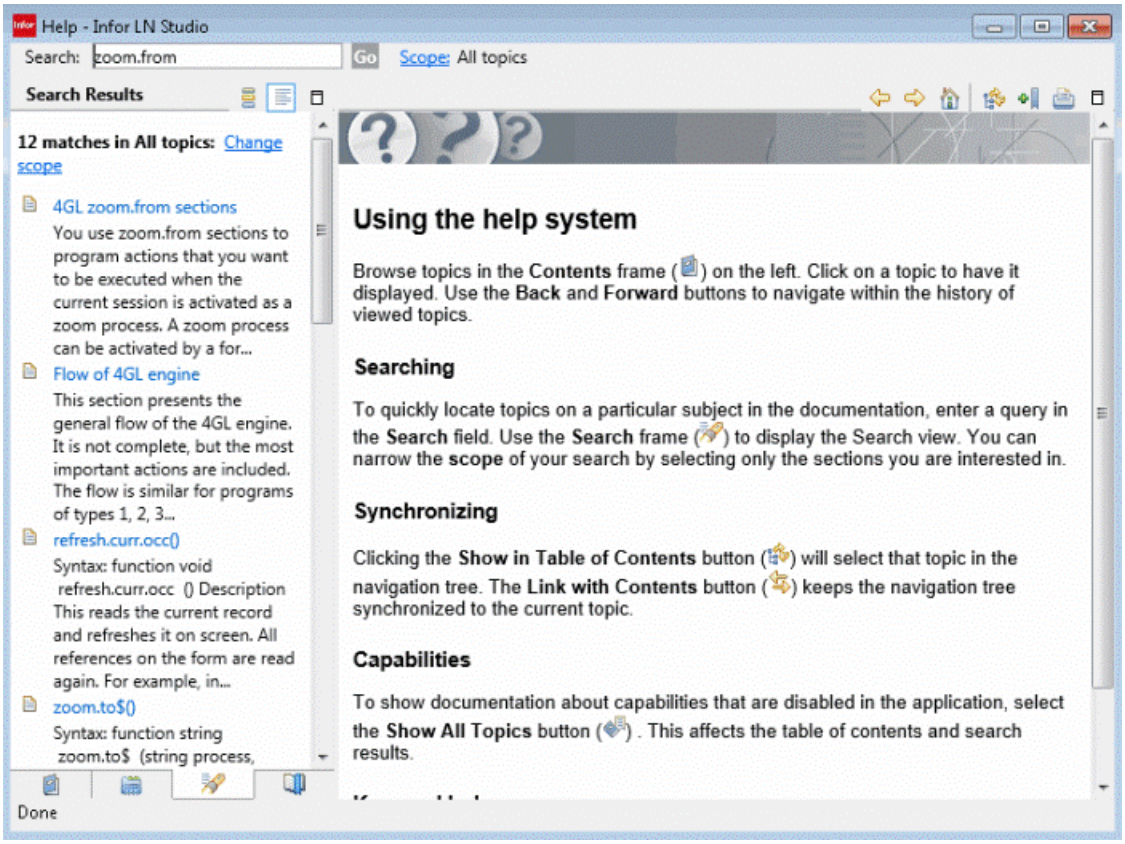
The Keyword Search feature searches for multiple string hits of the selected text in the LN Studio online help. This online help includes the *Infor ES Programmer's Guide*.

To perform a keyword search:

1 Select the text for which you want to perform the search.

2 Right-click and select **KeyWord Search**.

For example, you select the text `zoom.from` in the source and you run the **KeyWord Search** command. This figure shows the information that is displayed:



Note: When you use KeyWord Search for the first time, the help is indexed automatically. This may take a while.

Mark Occurrences

You can use this feature to mark occurrences of the selected element in the current source code.

If Mark Occurrences is enabled, the script editor can display markers to indicate where a variable, method, type, or other element is referenced.

Enabling Mark Occurrences

To enable Mark Occurrences, complete one of the following steps:

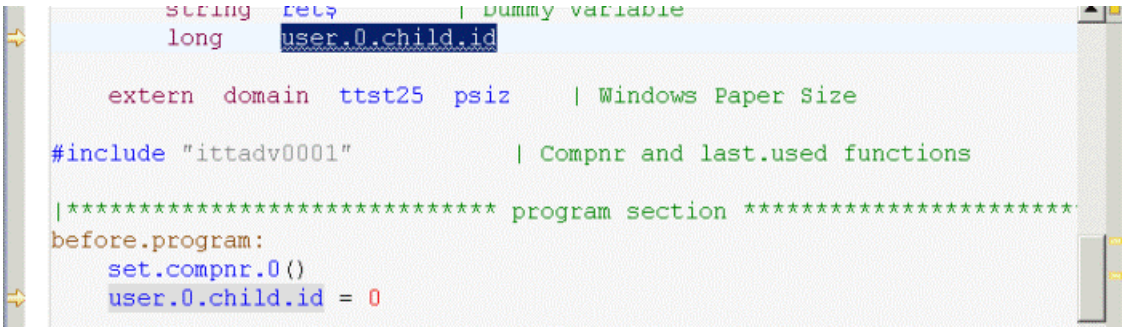
- Right-click the script editor's marker bar and select **Show Mark Occurrences**.
- Select the **Show Mark Occurrences** check box in the Preferences - Editor dialog box. You can access this preference page via the Preferences dialog.

See "Script Editor Preferences" on page 266.

Marking occurrences

To mark occurrences of an element, double-click the element. The script editor displays markers in the marker bar and overview ruler, to indicate references of the selected element.

This figure shows an example:



Note: The appearance of the markers depends on the settings in the Annotations preferences page. You can access this preference page via the Preferences dialog. See "Script Editor Preferences" on page 266.

Open Declaration

Use the **Open Declaration** command to view detailed information on an element in your source code, such as a domain code, a table code, a message code, or a function call.

Open Declaration can display the general properties or the source code of the selected element.

Note:

- **Open Declaration** only works on compiled scripts.
- **Open Declaration** does not work for all types of 4GL elements. See the following section.

Viewing general properties

To view the general properties of an element:

1. In the script editor, select the desired element.
2. Press SHIFT+F3 or, right-click the selected element and select **Open Declaration (properties)**. The **Overview** tab of the corresponding multipage editor starts.

Example

In the script editor, you select the ttccom000 table code. Then you press SHIFT+F3. The **Overview** tab of the Table Editor starts. The properties of tccom000 are displayed.

Viewing source code

To view the source code of an element:

1. In the script editor, select the desired element.
2. Press F3 or, right-click the selected element and select **Open Declaration (source)**. The **Source** tab of the corresponding multipage editor starts.

Example

In the script editor, you select the ttccom000 table code. Then you press F3. The **Source** tab of the Table Editor starts. The source code of the DAL script of tccom000 is displayed.

Functions and arguments

The following table shows a list of functions and, per function, the argument types that support **Open Declaration**.

- If you run **Open Declaration** for one of these functions, the corresponding help is displayed.
- If you run **Open Declaration** for one of the function argument types listed below, the corresponding multipage editor starts.

Function	Argument
abort.io skip.io	message
act.and.sleep activate wait.and.activate	session
ask.enum	question
brp.open brp.open.language spool.open	report
dal.set.error.message dal..warning.message dal.set.info.message	message
dialog.add.button dialog.add.field dialog.add.listbox	label
disable.satellite enable.satellite satellite.invisible	session
enum.descr$	domain

Function	Argument
form.text$	message
get.session.permission	session
load_dll exec_dll_function get_function parse_and_exec_function	library
mess	message
pcm.activate.session	session
set.max set.min set.fmax set.fmin	domain
set.input.error	message
set.synchronized.dialog	session
start.session	session
start.synchronized.child	session
substitute.session	session
tt.align.according.domain	domain
tt.index.desc	table
tt.is.domain.separated	domain
tt.label.desc	label
tt.report.desc	report
tt.session.desc	session
tt.session.present	session
tt.table.desc	table
zoom.to$	session

Example

A script contains the following code: `spooler = spool.open("rtccom040101000","D",0)`.

When you run **Open Declaration** for the `spool.open` function, the function's help text is displayed.

When you run **Open Declaration** for the `rtccom040101000` argument, LN Studio opens the tccom040101000 report in the Report Editor.

Miscellaneous elements

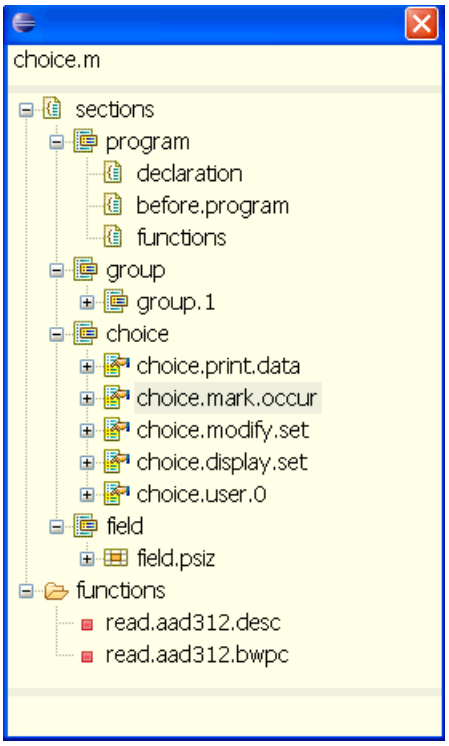
This table shows a number of elements and the actions that are performed when you run **Open Declaration** for those elements.

Element	Action
#pragma used dll	Adds the DLL to the workspace. Opens the DLL in the LN Studio script editor.
Macro call	The cursor jumps to the macro definition.
Local Functions	For example: You use **Open Declaration** on a name of a function that is declared somewhere else in the same script or library. The focus is set on the function declaration.

Quick Outline view

This view shows an outline of the source code that is currently open in the Script Editor, and lists the main structural elements.

This figure shows an example:



You can use the displayed elements to quickly navigate to the corresponding sections in the source code. You can for example:

- Click a folder icon to expand the corresponding tree. Subsequently, click an element to go to the corresponding section in the source code. The view closes automatically.

- Enter a search pattern in the input field in the upper part of the view. As you type the search expression, the element with the first match is highlighted. Click the element to go to the corresponding section in the source code. The view closes automatically.
- Click in the view and use the arrow keys to navigate to an element, and press ENTER to go to the corresponding section in the source code. The view closes automatically.

If you do not want to navigate, click the **Close** button or press ESC to close the view.

Note: This view is similar to the "Outline view" on page 321.

Text Hovering

Text hovering means that informational text is displayed when you hover with the mouse over a piece of text in the editor area.

The 4GL script editor supports text hovering for various elements, such as:

- LN standard functions: When you hover over a function name, the editor automatically shows the syntax of the function. The function syntax is displayed as tool-tip information.

  Press F2 to display the corresponding manual page from the *Infor ES Programmer's Guide*.

- Folded code elements (regions): When you hover over a folded 4GL element, the hidden code is displayed.

  For an example, see "Folding" on page 281.

- The markers in the marker bar (to the left of the editor area), and in the overview ruler (to the right of the editor area). For example: When you hover over a problem marker, or over a warning marker, the corresponding description is displayed in a ToolTip. For more information on the marker bar and the overview ruler, see "Source Tab" on page 267.
- Initiated variables in the Debug perspective: For example, when you debug a session, you can hover the cursor over an initiated variable in the editor area in the Debug perspective. The variable's value is displayed in a ToolTip.
- Domains, enumerated constants, tables, table fields, and table indices.
- Macros: When you hover over the name of a macro, which is defined in the current script or library, the editor displays the syntax of the macro.
- Messages and questions: when you hover over the name of a message or question, the corresponding description is displayed.
- Enums: during debugging, when you hover over the name of an enum constant, the corresponding numeric value and description are displayed.

Note: For all functions, you can use the "Keyword Search" on page 283 command in the editor's shortcut menu, to view the corresponding manual page from the *Infor ES Programmer's Guide*.

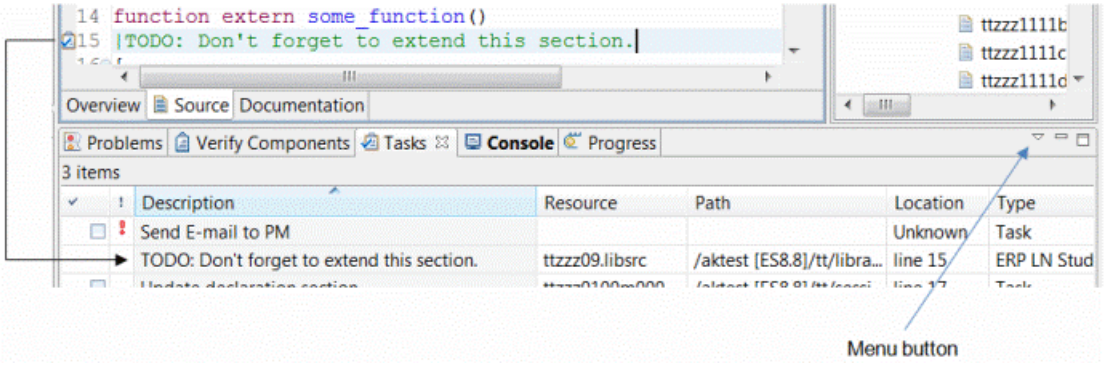
ToDo Comments

In the script editor, you can create to-do reminders in the script or library you are editing. These reminders are displayed in the Tasks view in the LN Studio workbench.

To create a to-do reminder, enter "|TODO:" and type the text for the reminder.

You can type the characters "TODO" in uppercase, in lowercase, or use a combination of both cases. So, "|ToDo:" and "|todo:" are also valid.

This diagram shows an example:



Note: To ensure the ToDo reminders are displayed in the Tasks view, click the **Menu** button and select **Show > Show all**. All tasks, including the ToDo reminders, are displayed.

Alternatively, define and activate a filter to display only ToDo reminders:

1 To define the filter:

   a Click the **Menu** button and select **Configure Contents**. The Configure Contents dialog box is displayed.

   b In the Configurations pane, select **TODOs**.

   c Under **Scope**, select **On any element**.

   d In the **Types** box, select **Infor LN Studio Todo**. Clear the other check boxes.

   e Click **OK**.

2 To activate the filter, in the Tasks view, click the **Menu** button and select **Show > TODOs**.

Toggle Breakpoints

A breakpoint is a point in a program that, when reached, triggers some special behavior useful to the process of debugging.

In the LN Studio, you can define multiple types of breakpoints:

- You can easily set and remove line breakpoints, and define watchpoints, in the script editor.
- You can toggle method breakpoints in the Outline view.

For details on these types of breakpoints, and on how to define them, see "Using breakpoints" on page 70.

Generating Source from WSDL

You can generate LN 4GL code, based on a URL or a Web Services Description Language (WSDL) file, for a general library.

Note: This functionality is only available for LN servers with Enterprise Server 8.7 or higher.

To generate library code from a WSDL file:

1. Create a library of type General Library. See:
   - "Creating a library" on page 104
   - "Create a New Infor LN Software Component" on page 98
2. Check-out the new library.
3. Edit the library. In the Library Editor, click the **Source** tab.
4. Right-click in the source code and select **Generate Source from WSDL**. The Generate Library from WSDL file dialog starts.
5. Specify the required information and click **OK**. See "Generate Library from WSDL file" on page 143.

Note: For information on how to use the generated LN code, see "Web Services from LN " in the *Infor ES Programmer's Guide*.

Perspectives

12

Application perspective

This *perspective* provides functionality for software engineers to develop software components.

The perspective consists of multiple *views* and an editor area.

For detailed information on the views in the perspective, see:

- "Activity Explorer view" on page 297
- "Component Explorer view" on page 309
- "Outline view" on page 321
- "Problems view" on page 324
- "Verify Components view" on page 332
- "Tasks view" on page 327
- "Progress view" on page 326

These editors can run in the editor area:

- Various multipage editors that you can use to edit software components, such as sessions, menus, labels, libraries and messages. See "Multipage Editors" on page 145.
- A script editor that supports the 4GL programming language. A content assistant is available, supporting syntax highlighting and completion of keywords. The editor also supports an outline of the scripts or libraries. See "Script Editor" on page 263.

For details on the procedure to develop software components, see "Developing software components" on page 55.

This table shows the most important buttons in the perspective's toolbar:

Button	Description
Check Out a Component (Alt+C, Alt+O)	Checks out the component you selected in the Activity Explorer.
Check In a Component (Alt+C, Alt+I)	Checks in the component you selected in the Activity Explorer.

Button	Description
Undo Check Out a Component (Alt+C, Alt+U)	Cancels the check-out for the component you selected in the Activity Explorer.
Select a Software Component (Alt+Q)	Starts the Select Component(s) dialog box. See "Select Component(s)" on page 94.
Create a New Software Component	Starts the Create a New Infor LN Software Component wizard. See "Create a New Infor LN Software Component" on page 98.
Create a New Activity (Ctrl+Alt+N)	Starts the New Activity dialog box. See "Create a new Activity" on page 340.
Debug	Use this option to debug LN sessions. See "Debugging LN sessions" on page 64.
Run	Use this option to run LN sessions. See "Running LN sessions" on page 62.
Verify Component	Starts the verification process for the component you selected in the Activity Explorer. See "Verifying software components" on page 58.

Debug perspective

The Debug perspective provides functionality for software engineers to debug software components.

You can use this perspective to perform these actions:

- Suspend and resume the execution of a component.
- Step through the execution of a program.
- Inspect values of variables.
- Evaluate expressions.
- Enable and disable breakpoints.

The perspective consists of multiple *views* and an editor area.

For detailed information on the views in the perspective, see:

- "Debug view" on page 312
- "Variables view" on page 329
- "Breakpoints view" on page 306

- "Expressions view" on page 315
- "Outline view" on page 321
- "Tasks view" on page 327

Note: By default, the Expressions view is not displayed in the Debug perspective. To open this view, select **Window > Show View > Expressions**.

For details on the LN Studio editors, which can run in the editor area, see:

- "Script Editor" on page 263
- "Multipage Editors" on page 145

For details on how to debug LN sessions, see "Debugging LN sessions" on page 64.

Views

13

Activity Explorer view

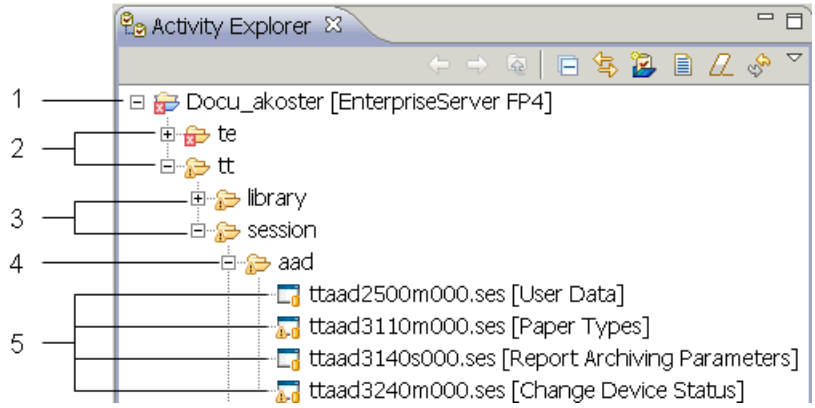
The Activity Explorer view provides a hierarchical view of the *activities* you have opened, and the software components linked to these activities.

The Activity Explorer enables you to perform these tasks:

- Open, close and view an activity.
- Check in, (un)checkout a software component (if Configuration Management is applicable).
- Open a component in the corresponding editor.

The Activity Explorer is part of the Application perspective.

This figure shows the Activity Explorer view:



Double-click a software component to open the component in the corresponding editor.

You can add other software components to your activity through the Component Explorer, and through the **Select a Software Component** command in the LN Studio toolbar.

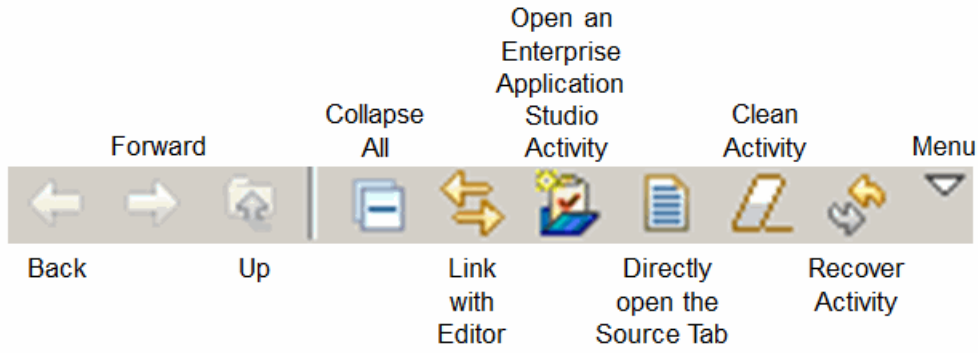
This table describes the different nodes in the previous figure:

Nr. in figure	Description
1	Name of an open activity, with the Project Name between square brackets.

Nr. in figure	Description
2	LN package code, displayed in ascending order (for example: td, tf, tp).
3	Main Component Type, displayed in ascending order. For example: domain, report, session, or table.
4	LN module code, displayed in ascending order. The module code is displayed for all component types, except domains, labels, messages, and questions.
5	LN software component, with the component description between brackets, displayed in ascending order. To open a software component in the corresponding editor, double-click the component.

Toolbar

The toolbar of the Activity Explorer view includes these buttons:



Link with Editor	Toggles whether the Activity Explorer selection is linked to the active editor. When this option is selected, changing the active editor will automatically update the Activity Explorer selection to the resource being edited.
Open an LN Studio Activity	Starts the Open an Activity wizard, where you can select an activity that is assigned to you. See "Developing software components" on page 55.
Directly open the Source Tab	If this option is selected, the multipage editor directly opens the **Source** tab (if the source code is available) when you open a component. If this option is not selected, the editor opens the **Overview** tab. See "Multipage Editors" on page 145.
Clean Activity	Removes unchanged components from your activity. The command only removes files that are not checked out, created, or modified. See "Cleaning an activity" on page 51. This command is useful when your activity contains a lot of unchanged components, such as when you have frequently used the **Open Declaration** command.

Recover Activity	Restores the components in your activity from the Configuration Management System on the server to your local workspace. Optionally, unchanged components are removed from the workspace. See "Recovering an activity" on page 52.
Menu	Provides, among other things, menu items that allow you to do the following: • Open, deselect and change working sets. See "Working sets" in the *Workbench User Guide*. • Sort the items in the view by full name, or by type/extension. • Apply a filter. • Toggle whether the view selection is linked to the active editor. • Show Checked Out/Modified components only. This option is useful if your workspace contains a lot of components not checked out, such as when you have frequently used the **Open Declaration** command. See the *Workbench User Guide*.

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 a resource in the view.

This menu provides, among other things, the following commands:

New	Allows you to open a new *activity* in the root node, or to create a new component in the current activity. To open an activity, on the submenu, select **Infor LN Studio Application Project (Open Activity)**. The Open an Activity wizard starts. When you open an activity, the following occurs: • The activity is automatically placed under the control of configuration management. • The content of the associated development repository is displayed in the Component Explorer. This view is refreshed and a (closed) tree with the LN packages is displayed for the *Application* linked to the *software project* to which the activity belongs. To create a new component, on the submenu, select **Infor LN Component**. The Create a New Infor LN Software Component wizard starts.
Open	Opens the selected software component in the LN Studio in the corresponding multipage editor.
Open With	Allows you to open the selected software component in an editor other than the component's default editor. Select the desired editor from the submenu.
Copy	Copies the selected resource to the clipboard.

Paste	Pastes resources on the clipboard into the selected project or folder. The resource is not added to the *Revision Control System*. If a resource is selected, the resources on the clipboard are pasted as siblings of the selected resource.
Delete	Deletes the selected resource from the workspace. **Note:** This command can only delete components that are not checked out, created, or modified in the current activity. These components have a decorator in the bottom right corner of the software component image.
Move	Moves the selected resource to another location. A dialog will appear, requesting the location to which the resource will be moved.
Rename	Allows you to specify a new name for the selected resource.
Import	Opens the import wizard and allows you to select resources to import into the Workbench.
Export	Opens the export wizard and allows you to export resources to an external location.
Refresh	Refreshes the Workbench's view of the selected resource and its children. For example, when you create a new file for an existing project outside the Workbench and want the file to appear in the Activity Explorer view.
Duplicate Component	Starts the Duplicate an Infor LN Software Component wizard, where you can duplicate the selected component.
Verify Component	Performs quality control on the selected software component(s). The command executes various checks based on the Infor LN design principles. When you verify software components, a list of *warnings* is generated. You can decide to accept each warning, or to solve the problem. See "Verifying software components" on page 58.
Link Run Configurations	Starts a dialog where you can link existing run configurations to the selected activity. See "Linking run configurations to an activity" on page 54. This command is only displayed if you select an activity.
Run As	This menu contains these commands:

Infor LN Session Alt+Shift+X, L	Starts the selected session and automatically generates a run configuration for the session. This option is only displayed if you right-click a session in the view.
Run Configurations	Starts the Configurations dialog. Use this dialog to create or modify run configurations.

For more information, see "Running LN sessions" on page 62.

Debug As	This menu contains these commands:	
	Infor LN Session Alt+Shift+D, L	Starts the selected session in *debug* mode and automatically generates a run configuration for the session. This option is only displayed if you right-click a session in the view.
	Debug Configurations	Starts the Configurations dialog. Use this dialog to create or modify run configurations.
	See "Debugging LN sessions" on page 64.	
Team	This menu provides various Configuration Management functions. See "Activity based development" on page 39. If you select a software component, these commands are available:	
	Recover	Restores the selected component, which was saved on the LN server, to your local workspace. See "Recovering an activity" on page 52.
	Checkout	You must lock (check out) a component before it can be changed. When a check out is performed, the latest version of the software component is retrieved from the development repository. It is then made editable in the workspace. **Note:** This option is not available for components already checked out.
	Checkin	Use this command when you finished editing a component, to unlock a component and store the changes in the development repository. **1** You are prompted to save any outstanding changes in the component. **Note:** If you cancel the “modified source” dialog, the check in is cancelled. **2** Then, you are asked to enter a revision text: The updated version of the component is sent to the development repository and the component is made read-only in the workspace. **Note:** This option is unavailable for components not checked out.
	Uncheck-out	Use this command when you have finished editing the component, and you do not want to send the changes to the development repository. **1** The component is unlocked (and changes are discarded). **2** You are prompted to confirm the command: If the command is confirmed, then the current local representation of the software component is replaced by the latest remote representation. **Note:** • If you check out a component for the first time and then perform an uncheckout, the component is considered as unchanged. The compo-

	nent has a decorator in the bottom right corner of the software component image.
	• You cannot perform an uncheckout in the following situations:
	  • If SCM is disabled.
	  • If the component does not support SCM, for example if the component is a label, message, question, domain, or additional file.
	You must perform a checkin because the changes cannot be rolled back.

Compare	Starts the Compare Package VRC's (ttadv6450m000) session. The session prints the differences between the latest version and the previous version of the component. The differences can be printed in detail, if required. See the session help.
Show Revisions	Starts the History view . This view shows all revisions of the selected component, which are present in the derivation path of your *Activity VRC*. See "History view" on page 318.
Expire	Performs a check out of the component, sets the Expired flag, and performs a check in. A decorator is displayed over the component image. **Note:** A session can only be expired if there is no reference to it from another not-expired session. See "Session Editor" on page 205.
Unexpire	Performs a check out of the component, unsets the Expired flag, and performs a check in. The decorator disappears. **Note:** A session cannot be unexpired if it has a session reference to an expired session. See "Session Editor" on page 205.
Delete	Removes the selected component. **Note:** This command can only delete components that are created or modified in the current activity. These components have a decorator in the bottom right corner of the software component image.

If you select an activity, the following commands are available:

Reassign Activity	Starts the Reassign Activity dialog, where you can assign the activity to another user.
Cancel Activity	Use this command to undo all changes in an activity.

	End Activity	Closes the activity and releases the changed components to the *Project VRC*. In the Preferences - Infor LN Configuration Management dialog, you can set the default behavior for this command.
Compare With	Provides a mechanism to compare the selected software component with other versions of that component in the workspaces' local history.	
Replace With	Provides a mechanism to replace the selected software component with other versions of that component in the workspaces' local history.	
Restore from Local History	Restores deleted Workbench resources with a state from the local history. See "Restoring deleted resources from local history" in the *Workbench User Guide*.	
Properties	Starts the property page of the selected item.	
Refactor	If you select an InContext Table Model, this command is available:	
	Dependent InContext Models	Starts the Dependent InContext Models view. You can use that view to refactor the table model's dependent models. Related topics • "Dependent InContext Models view" on page 314 • "InContext Modeling" on page 75 • *Infor Enterprise Server InContext Modeling Development Guide* • "Table InContext Model Editor" on page 257

Note:

- Some commands in the shortcut menu, such as **Checkout** and **Run as**, are also available as buttons in the toolbar in the Application perspective.

  See "Application perspective" on page 293.

- The previous list does not describe all commands in the shortcut menu. The remaining commands are standard Eclipse commands. For details on these commands, see "User interface information" in the *Workbench User Guide*.

Icons and decorators

Icons

The following icons are displayed in the Activity Explorer view:

Icon	Description
	Activity
	Folder
	Additional File
	Domain
	Function
	Label
	Library
	Menu
	Message
	Question
	Report
	Session
	Table

Decorators

CM decorators

The following decorators are used to show Configuration Management related status information, which is mutually exclusive. The decorators are displayed in the bottom right corner of the software component image.

Decorator	Description
	Checked Out
	Dirty. The component is changed, but not saved to the Configuration Management (ERP LN) yet. Note: The dirty state for sessions, tables and reports can not be determined accurately, because the modification date is not available in ERP LN.
	Edited. The component is created or changed in your current activity.
	Unchanged. The component is not changed in your current activity.

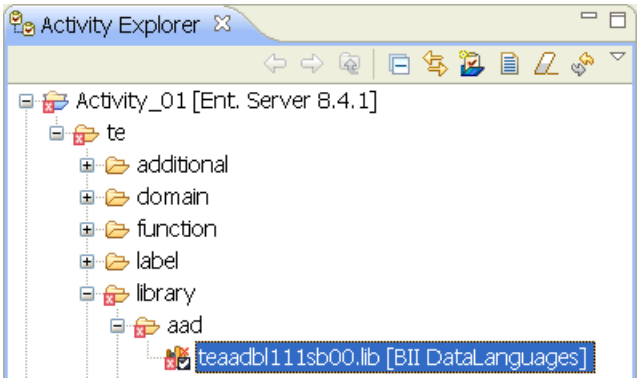
Compiler decorators

The following decorators are used to indicate errors and warnings. The decorators are displayed in the bottom left corner of the images of the involved nodes.

Decorator	Description
	Warning
	Error

The decorators are displayed for the involved software component and for all parent nodes.

For example, if a library contains invalid code, error decorators are displayed for the library itself, and also for the module, the main component type, the package, and the activity to which the library belongs. See the following figure:



VSC decorators

The following decorators are used to indicate the VSC status of a software component. The decorators are displayed in the upper right corner of the software component image.

Decorator	Description
	Component is checked by VSC and OK.
	VSC error or component is not checked yet.

For details on VSC, see "Verifying software components" on page 58.

"Expired" decorator

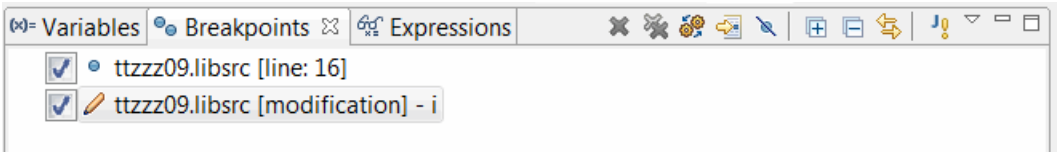
The following decorator is used to indicate a software component is expired. The decorator is displayed over the software component image.

Decorator	Description
	The component is deprecated (expired).

Breakpoints view

The Breakpoints view shows a list of all available breakpoints and their state (enabled, disabled, resolved and invalidated).

This figure shows an example:



In this view, you can perform these actions:

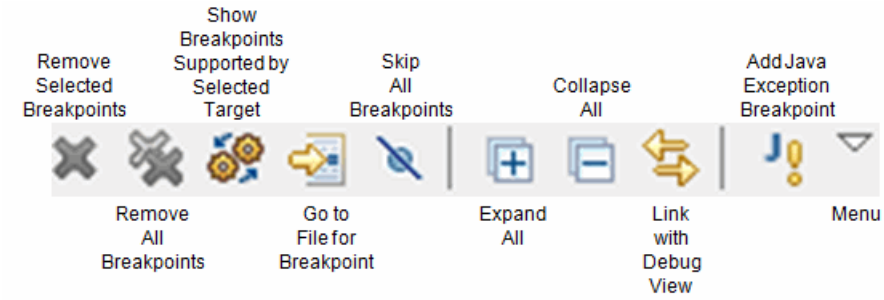
- Double-click a breakpoint, or use the **Go to File** command, to display its location in the Script Editor.
- Enable or disable breakpoints.
- View or modify the properties of a breakpoint.
- Delete breakpoints.

Note: You cannot add new breakpoints in this view. New breakpoints are added via the Script Editor and via the Outline view.

For information on adding breakpoints, see "Using breakpoints" on page 70.

Toolbar

The toolbar of the Breakpoints view contains these buttons:



Show Breakpoints Supported by Selected Target	When you use multiple programming languages within the Eclipse workbench such as Infor 4GL and Java, you can define different types of breakpoints per language. If this option is toggled on, only breakpoints applicable to the selected debug target are displayed, such as the following: • You debug an LN session, only the LN specific breakpoints are displayed. • You debug a Java program, only the Java specific breakpoints are displayed.

Go to File for Breakpoint	Displays the source code and the line associated with the selected breakpoint in the Editor area. Double-clicking a breakpoint has the same effect.
Skip All Breakpoints	Disables all breakpoints.
Link with Debug View	When this option is toggled on, the breakpoints hit by the debugger are automatically highlighted in the Breakpoints view.
Add Java Exception Breakpoint	Starts the Add Java Exception Breakpoint dialog box. See the online help of the dialog box.

Shortcut menu

Right-click on any breakpoint in the view to open a shortcut menu. The menu contains these commands:

- **Go to File**
- **Enable**
- **Disable**
- **Remove**
- **Remove All**
- **Select All** Ctrl + A
- **Copy** Ctrl + Insert
- **Paste** Shift+Insert
- **Export Breakpoints**
- **Import Breakpoints**
- **Properties**

Use the shortcut menu to enable and disable breakpoints. Alternatively, select or clear the check boxes displayed in front of the breakpoints.

When you run the **Properties** command, a dialog box is started where you can view and modify the properties of the selected breakpoint. In this dialog box you can, for example, enable or disable a breakpoint. See:

- "Line Breakpoint properties" on page 112
- "Method Breakpoint properties" on page 113
- "Watchpoint properties" on page 113

Icons and decorators

Icons

These icons are displayed in the Breakpoints view:

Icon	Description
	Line breakpoint - enabled
	Line breakpoint - disabled
	Method breakpoint - enabled
	Method breakpoint - disabled
	Watchpoint (Suspend on Variable Modification) - enabled
	Watchpoint (Suspend on Variable Modification) - disabled
	Watchpoint (Suspend on Variable Value) - enabled
	Watchpoint (Suspend on Variable Value) - disabled

Decorators

The following decorators are used to indicate whether a breakpoint is installed or invalidated. They are displayed in the bottom left corner of the breakpoint icon.

Decorator	Description
	Resolved: The breakpoint has been resolved (is valid and active) by the debug server.
	Invalidated: The breakpoint has been invalidated (is invalid and not active) by the debug server.
	Entry breakpoint.
	Disabled entry breakpoint.

Note: The "Invalidated" decorator is not applicable for method breakpoints.

BFlow Palette view

This view shows the actions you can use in BFlow tests.

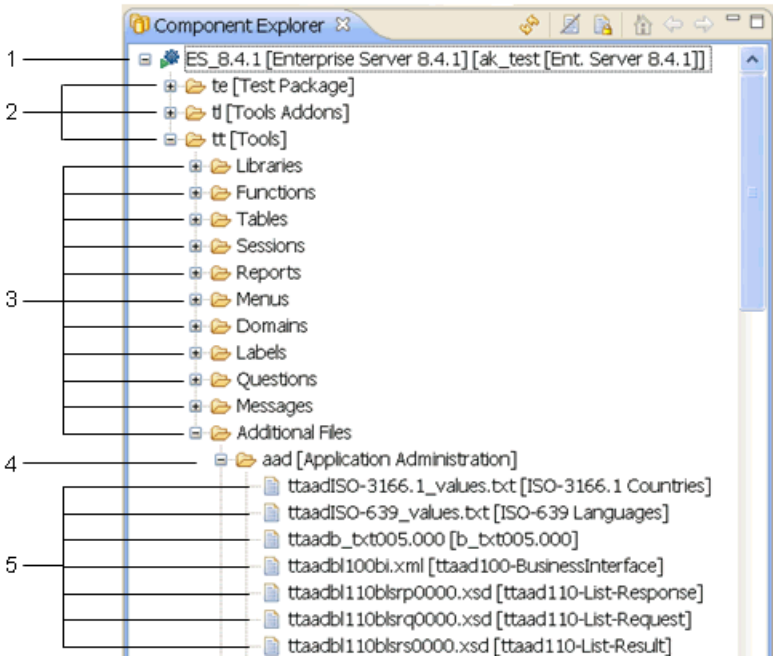
Use the buttons in the title bar to change the way the actions are displayed. The actions can be displayed in these ways:

- As a list, in alphabetical order
- Grouped by category

To add an action to a Bflow test, drag the action to the tree in the **Actions** tab in the BFlow Editor. Alternatively, use the shortcut menu of the tree.

Component Explorer view

This view provides a hierarchical view of the software components on the LN server.



Use this view to add these software components to your personal activity in the LN Studio:

- Additional Files
- Domains
- Functions
- Labels
- Libraries
- Menus
- Messages
- Questions
- Reports
- Sessions
- Tables

The following table describes the different nodes in the previous figure.

Nr. in figure	Description
1	*Application* that determines the development repository to be used.
2	LN package code displayed in ascending order (for example: td, tf, tp).
3	Main Component Type. For example: Domain, report, session, or table.

Nr. in figure	Description
4	LN module code displayed in ascending order. The module code is displayed for all component types, except domains and labels.
5	LN software component displayed in ascending order. Double-click a software component to add the component to your personal activity.

To find a particular software component, perform one of these actions:

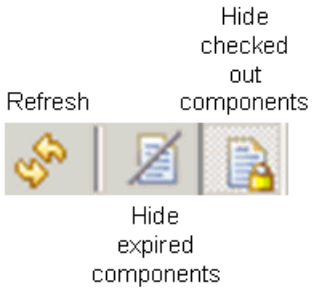
- Expand the relevant folders in the Component Explorer view. For example, to find the Employees - General (tccom0101m000) session, expand the following folders: tc (Common), Sessions, and com (Common Data).
- On the LN Studio toolbar, click **Select a Software Component**, or press ALT+Q. The Select Component(s) dialog starts. Use this dialog to find components and link them to your current activity.

Note: Labels are not sorted by module, but by search key. They are divided into subgroups.

Messages and questions are sorted by module. If a message or question code does not start with a valid module code, the message/question is categorized in the "other" subgroup.

Toolbar

The toolbar of the Component Explorer view includes the following buttons:



Refresh	Refreshes the contents of the view. For example, use this command when you have created a new session on the LN server and want the session to appear in the Component Explorer view.
Hide expired components	Hides components with the Expired status.
Hide checked out components	Hides components already *checked out*.

Shortcut menu

To open the shortcut menu, select one or more software components in the view. Then right-click.

The shortcut menu contains the **Get** command. Use this command to link the selected software components to your current activity.

The components are now displayed in the Activity Explorer view. You can now check out and edit the components. For more information, see "Developing software components" on page 55.

Note: When you use the **Get** command, the editor can automatically start for the selected components. However, this depends on your workbench preference settings. See "Preferences - Workbench" on page 121.

Icons and decorators

Icons

The following icons are displayed in the Activity Explorer view:

Icon	Description
	Application (active)
	Application (inactive)
	Folder. For example: Package, Main Component Type, or Module.
	Software component
	Checked out component. The component is checked out in your current activity.
	Expired component
	Locked component. The component is checked out in another activity.
	Locked and expired component
	Changed component. The component is changed in your current activity.
	Changed and expired component
	Problem
	Loading...

Note: The **Loading...** node is displayed when information is retrieved from the server. For example, this happens when you open a node for the first time.

Such a load process is time-consuming. Therefore the information is gathered in the background.

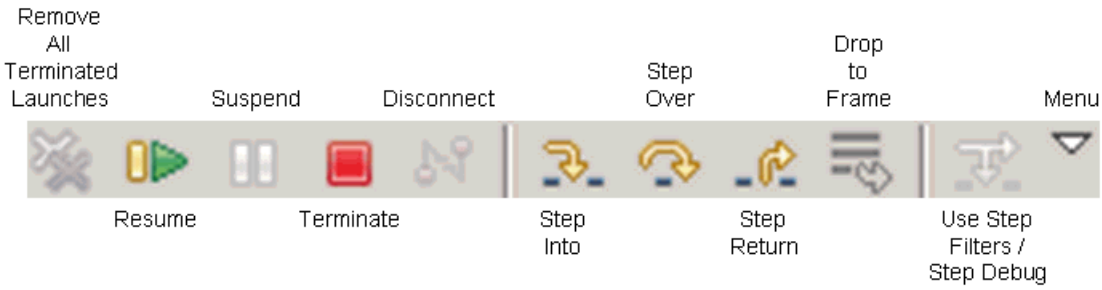
Debug view

The Debug view allows you to manage the debugging or running of Infor LN sessions in the workbench. The view displays a list of sessions, either started normally (in run mode) or started in debug mode. It also shows related information (stack frames) and state information (terminated, running or suspended).

If you suspend a session, the view displays the *stack frames* for the suspended session. A session is displayed as a node in the tree. The corresponding stack frames are displayed as child elements.

Toolbar

The toolbar of the Debug view contains the following buttons:



Remove All Terminated Launches	Removes all terminated sessions from the Debug view.
Resume	Resumes a suspended session. The session resumes its execution and stack frames are no longer displayed for the session. The Variables view is cleared.
Suspend	Requests to suspend a running session. The session will be suspended when the next debugable line is hit. See "Suspending sessions" on page 68.
Terminate	Terminates the session related to the selected item.
Disconnect	This command is not supported by LN Studio.
Step Into	Steps into the current statement. This command invokes the next instruction to be executed. The execution suspends at the next executable line. **Note:** Ensure the **Reuse editor when displaying source code** option in the Run/Debug preferences dialog is disabled. If this option is enabled, and you use the **Step Into** command while the script is not in the workspace, an error page is displayed in some situations.
Step Over	Steps over the current statement. This command executes the next instruction. The execution suspends on the next executable line.

Step Return	Continues the execution until the end of the current function. **Note:** This command is equivalent to the **return** action in the classic 4GL Debugger.
Drop to Frame	This command is not supported by LN Studio.
Use Step Filters / Step Debug	This command is not supported by LN Studio.
Menu	Contains the **View Management** command. This command starts a preference page where you can configure the debugger to open debug-related views, such as the Variables view and the Breakpoints view, in non-debug perspectives.

Note:

- Most of these commands are also available in the **Run** menu in the Eclipse Workbench.
- The **Run** menu contains an additional command: **Run to Line (CTRL+R)**. This command moves the execution pointer, without executing code, to the line you selected in the Script Editor. **Run to Line** is equivalent to the **Go to Line** action in the classic 4GL Debugger.
- For more information, see "Debugging LN sessions" on page 64.

Shortcut menu

Right-click on any resource in the view to open a shortcut menu. This table shows commands in the menu:

Copy Stack	Copies the selected stack of suspended sessions and the state of the running sessions to the clipboard. You can paste the information from the clipboard into various applications, such as MS Word and Notepad.
Find	Starts a dialog where you can enter a search expression to find items in the Debug view.
Terminate and Relaunch	Terminates the session related to the selected item and relaunches the session.
Remove All Terminated	Removes all terminated sessions from the Debug view.
Relaunch	Re-launches the session related to the selected item.
Edit <session name>	This command is not supported by LN Studio.
Terminate and Remove	Terminates the session related to the selected item and removes the session from the Debug view.
Terminate All	Terminates all sessions.
Properties	Starts a property page with process information for the session you selected in the Debug view.

For information on shortcut menu commands that are not mentioned in the previous table, see the descriptions of the Toolbar commands.

Icons

These icons are displayed in the Debug view:

Icon	Description
	Session in run mode - running
	Session in run mode - terminated
	Session in debug mode - running
	Session in debug mode - suspended
	Session in debug mode - terminated
	Thread - running
	Thread - suspended
	Thread - terminated
	Stack frame

Note:

- You can only suspend sessions started in debug mode. If a session was started normally (in run mode), you cannot suspend the session, but only terminate it.
- *Stack frames* are only available when the session is suspended.

Limitations

Exit value not displayed

The exit value of an LN process is not displayed in the Debug view.

“An error has occurred. See error log for more details” error message

While debugging, the message `"An error has occurred. See error log for more details"` may be given by Eclipse. This is not a fatal error. Click the **OK** button and resume debugging.

Dependent InContext Models view

This view shows the dependent InContext Models of a selected InContext Table Model.

After you modified an InContext Table model, you can use this view to refactor all dependent models. To open this view, click the Dependent InContext Models link in the Table InContext Model Editor.

In this view, you can select one or more InContext Models and then use toolbar buttons or shortcut menu options to refactor these models.

To refactor the dependent models:

1. Select the dependent models in the view.
2. Check out the selected models.
3. Refresh context messages for the selected models.
4. Regenerate libraries for the selected session models.
5. Check in the selected models.

Toolbar

This table shows the descriptions of the toolbar buttons:

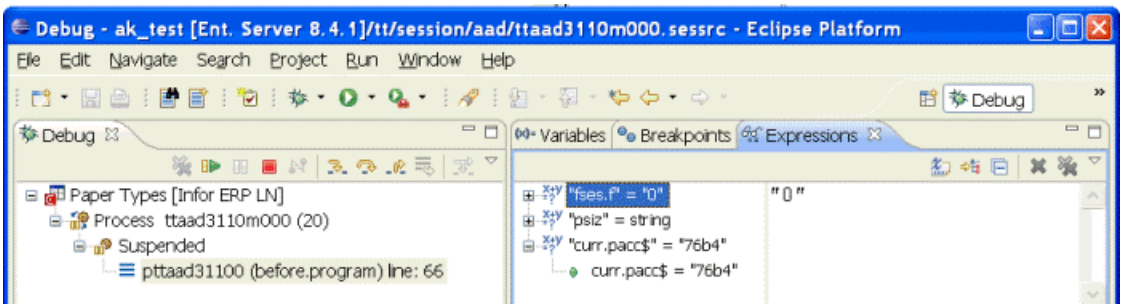
Button	Description
	Refreshes the contents of the view.
	Checks out the selected InContext models.
	Checks in the selected InContext models.
	Refreshes the context messages for the selected InContext models. Only checked-out models are processed.
	(Re)generates the libraries for the selected InContext session models.

Shortcut menu

Right-click in the view to open a shortcut menu. The commands in the menu are also present in the toolbar. See the description of the toolbar buttons.

Expressions view

The Expressions view contains a list of watch expressions. Only variables are supported. These watch expressions are evaluated each time a session suspends. The expressions are evaluated in the context of the currently selected stack frame.



The left part of the screenshot shows the Debug view with the selected stack frame. The right part shows the Expressions view with the watch expressions. In the screenshot, the Expressions view consists of an overview pane (left) and a details pane (right).

The details pane shows this information:

- Value of primitive variables.
- Hexadecimal value of variables of type string.
- XML structure of variables of type XML.

Expressions that cannot be evaluated in the context of the current stack frame will be indicated as “cannot be resolved”.

You can determine the amount of information displayed in the Expressions view. For example, you can show / hide the following:

- Data type information.
- The details pane.

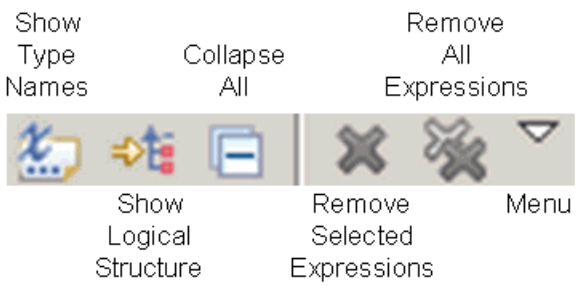
See the description of the toolbar and shortcut menu commands.

Note: Try to keep the list of expressions limited. This is, because all expressions are evaluated each time a session suspends, which can affect the performance adversely.

If an array variable contains more than 10 dimensions, the variable is divided into partitions of 10 dimensions.

Toolbar

The toolbar of the Expressions view contains these buttons:



Show Type Names	You can toggle this option to show or hide data type names. The data type names are displayed in front of the expression names.
Show Logical Structure	Displays certain complex data structures in a more compact and meaningful form. For example, elements of a list are displayed as an ordered collection. Therefore, you can easier navigate complex data structures in terms of their logical structure, than in terms of their implementation.
Collapse All	Collapses all expanded expressions.
Menu	Contains these layout options: • Vertical View Orientation: The details pane is displayed below the overview pane. • Horizontal View Orientation: The details pane is displayed next to the overview pane. • Expressions View Only: The details pane is not displayed.

Shortcut menu

Use the shortcut menu to add, remove, and edit watch expressions. If you right-click a watch expression when the debugger is active, the shortcut menu also contains the **Reevaluate Watch Expression** command. This command evaluates the watch expression again, which is useful after editing a watch expression.

Note: To define an expression that must be executed only once, click **Execute Assignment Expression**. The Execute Assignment Expression dialog box starts. See the dialog box's online help.

To create a watch expression that is executed each time you step through the execution, click **Add Watch Expression**. The Add Watch Expression dialog box starts. In this dialog box, specify the expression in this format: `<variable>:=<value>`. Then, select the **Enable** check box and click **OK**.

Icons and decorators

Icons

These icons are displayed in the Expressions view:

Icon	Description
	Watch expression
	Global variable
	Local variable
	External variable
	Internal variable
	Array element
	Array partition

Decorators

The following decorators are used to indicate whether a variable is fixed and / or based. Decorators are displayed in the upper right corner of the variable icon.

Decorator	Description
B	Based variable
F	Fixed variable

Limitations

Limited to variables

Currently, it is only possible to evaluate expressions that are actually variables. It is not possible to evaluate other expressions, such as I=0.

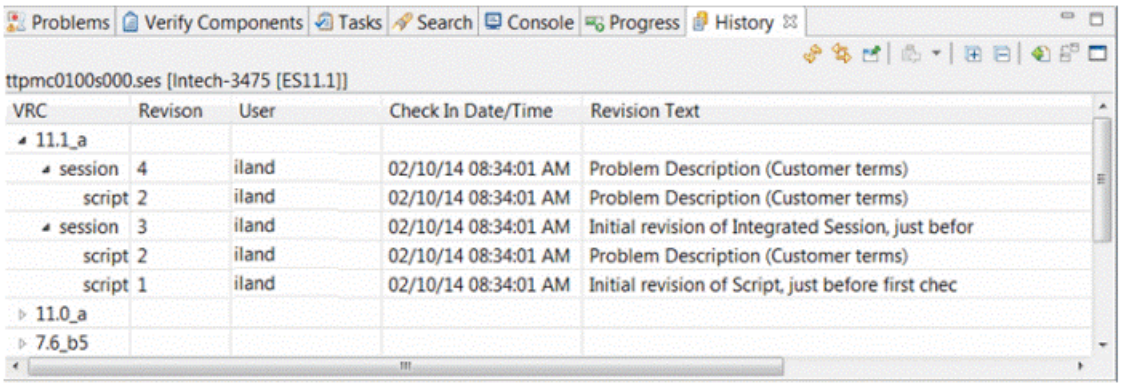
Changing values not reflected in UI

The value of an expression is not updated, when the value of the contained variable is changed. To solve this, use the **Reevaluate Watch Expression** command on the expression.

History view

This view shows all revisions of the selected component, which are present in the derivation path of your *Activity VRC*.

This figure shows the revisions of a session that was selected in the Activity Explorer:



Revisions can exist for these components:

- Domain
- Function
- Label
- Library
- Menu
- Message
- Question
- Report
- Session
- Table

Columns

This table shows the columns in the view:

Column	Description
VRC	A tree structure that shows this information: • The Package VRC to which the revision belongs. • The component type. For sessions and tables, separate revision numbers are stored for the session or table itself and for the corresponding UI script or DAL script. You can expand the component type node to view the corresponding script revisions. See the previous figure. The other components have only one revision number.

Column	Description
	If you expanded a session or table node, the shortcut menu items are only enabled if a script is selected.
Revision	The number of the revision.
User	The user that created the revision.
Check In Date/Time	The date the revision was created.
Revision Text	The revision text that describes the functionality of the changed software component.

If the text in a column does not fit in the column, hover over the text. A tooltip with the complete text is displayed. Click inside the tooltip, or press F2, to view the text in a dialog box.

Toolbar

This table shows the most important toolbar buttons:

Button	Description
	Refresh Refreshes the contents of the view.
	Link with Editor and Selection When enabled, the view shows the history for the resource of the active editor or of the active selection.
	Pin this History View When enabled, pins the view and its contents. Any new requests for history open a new instance of the History View.
	Show Previous Histories When enabled, you can select a resource, for which a history has been viewed in the past, from a drop-down menu. This button is disabled when the History view is opened for the first time.
	Expand All Expands all categories.
	Collapse All Collapses all categories.
	Checkout to current Checks out the component and updates it to the selected revision.

Button	Description
	Compare This button is only enabled if two items are selected. Opens the Compare view where you can view the differences between both revisions.
	Compare To Local This button is only enabled if one item is selected. Opens the Compare view where you can view the differences between the selected revision and the local version.
	Show revision details Opens the selected item. Double-clicking an item has the same effect as this command.

Shortcut menu

Right-click on a selected item to open a shortcut menu. The commands in the menu are also present in the toolbar. See the description of the toolbar buttons.

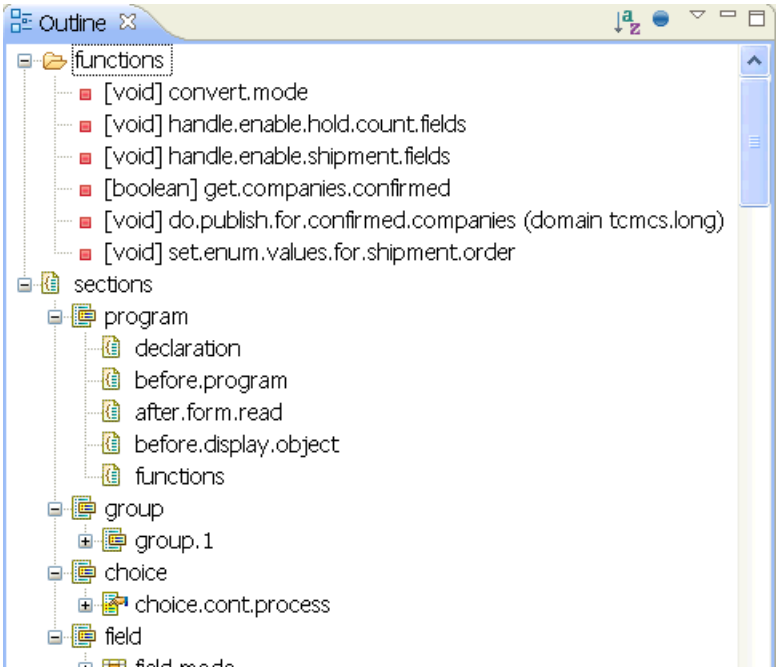
Outline view

This view displays an outline of a script or library currently open in the script editor, and lists the main structural elements.

The contents of the outline depend on the type of library or script. For example:

- For a library, the Outline view displays the corresponding functions.
- For a UI script, the outline consists of functions (if any) and script sections, such as:
  - Program sections.
  - Form sections.
  - Group sections.
  - Choice sections.
  - Field sections.
  - ZoomFrom sections.
  - Main Table sections.

This figure shows the outline of a UI script:

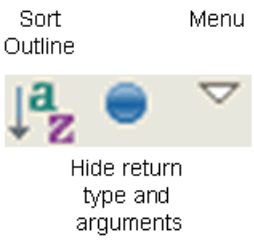


When you select a structural element in the Outline view, the cursor will move to the corresponding line in the script editor.

The Outline view is updated automatically when you edit a script or library.

Toolbar

The toolbar of the Outline view includes the following buttons:



Sort Outline	Sorts the displayed folders and their contents alphabetically.
Hide return type and arguments	Hides return types and arguments of functions.
Menu	Contains these options: • Link With Editor: Toggles whether the contents of the Outline view are linked to the active editor. When this option is selected, changing the active editor will automatically update the contents of the Outline view to the resource being edited.

- Hide return type and arguments: Hides return types and arguments of functions.
- Hide local functions: Hides local functions, so only external functions are displayed.
- Hide sections: Hides sections, so only functions are displayed.
- Show external functions first (in sorted list): if you sort the displayed folders and their contents, external functions are displayed at the top of the list.

Shortcut menu

The Outline view contains a shortcut menu. This enables you to define method breakpoints.

This table shows the commands in the shortcut menu:

Toggle Method Breakpoint	Adds or removes a method breakpoint for the selected function. When started in debug mode, the session suspends when the execution reaches a function for which a method breakpoint was set. This command is only available for functions and not for other structural elements. For more information on breakpoints, see "Using breakpoints" on page 70.

Icons

The following table explains the icons used to identify the structural elements.

Icon	Description
	Program section in the 4GL script.
	Private function. Functions declared without the EXTERN keyword are local functions. They are accessible only within the program in which they are declared.
	External function. Functions declared with the EXTERN keyword are public functions. They are accessible outside the program in which they are declared.
	Form section. Use form sections to program actions you want to execute when forms are activated or ended.
	Group section. Use group sections to program actions you want to execute when a particular group is activated or ended.
	Choice section. Use choice sections to program actions you want to execute when standard commands or form commands are activated or ended.
	Field section. Use field sections to program actions you want to execute for a variety of field events.
	Zoom.from section. Use zoom.from sections to program actions you want to execute when the current session is activated as a zoom process.
	Main Table section. Use main table i/o sections to program actions you want to execute when read or write actions occur on the main table.
	Sub section. For example: the init.group sub section in a group section, or the on.entry sub section in a zoom.from section.
	Report section. These sections only occur in report scripts. Use report sections to program actions you want to execute when the report is activated. Examples of report script sections are: header, footer, detail, before.field and after.field.

For details on the 4GL components mentioned in the previous table, see the *Infor ES Programmer's Guide*.

Problems view

The Problems view displays system-generated errors, warnings, or information associated with a resource. These are typically produced by builders/compilers. For example, if you build a source file that contains syntax errors, the errors are automatically logged in this view.

Note: If a problem is associated to (a line within) a component, double-click the problem to edit the component. The script editor opens the component at the involved line.

The problems are grouped into the following severity categories: Errors, Warnings, and Information.

This table shows the columns in the view:

Description	A description of the problem. Also indicates the severity of the problem: Error, Warning or Info.
Resource	The name of the resource associated with the problem.

Path	The folder in your workspace where the resource is stored.
Location	The line number of the problem within its associated resource.
Type	The type of problem, such as "Script Problem".

Toolbar

This table shows the buttons in the toolbar of the view:

	Provides commands to sort and filter the contents of the view, and to define preferences.
Menu	The **Configure Contents** command in the menu starts a dialog box where you can specify filters. You can filter items according to which resource or group of resources they are associated with. This is done by text string within the **Description** field, or by problem severity. **Note:** To view problems and warnings related to the following, select the appropriate check box in the Configure Contents dialog box: • Configuration management: select the **SCM Problem** check box. • The source code in your scripts and libraries: select the **Script Problem** check box.

Known limits

The filter does not work properly if the **Link with Editor** option in the Activity Explorer is selected. For example, if you filter “On selected element only”, the compilation errors and warnings belonging to the source file are not displayed properly.

Shortcut menu

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

This menu provides various commands, such as a **Properties** command to view the properties of the selected problem.

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

Icons

These icons are used by the Problems view:

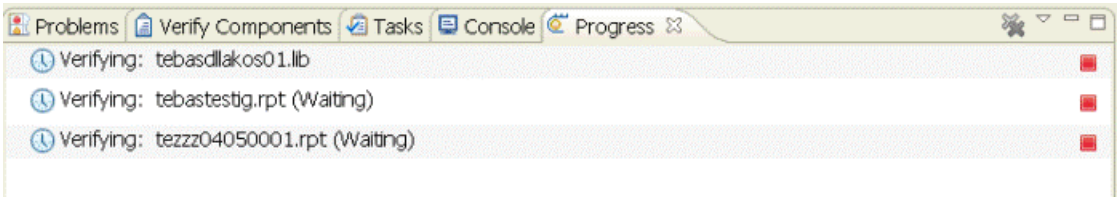
Icon	Description
	Information
	Warning
	Error

Progress view

The Progress view shows the progress of background processes. For example, it is used when the following occurs:

- You open an *activity*.
- CM (Configuration Management) information is loaded from the LN server. This can occur when you start LN Studio, or the first time you open a node in the Component Explorer view. In both situations, the requested data is retrieved by a background process. When the process is finished, the data is displayed in the Component Explorer view.
- You build a software component or a project.
- You verify a software component.

This figure shows an example:



The progress of these processes is displayed in the Progress view, and indicated in the bottom-right corner of the workbench window. See the previous figure.

Toolbar

The toolbar of the Progress view includes these buttons:

Remove All Finished Operations	Removes all finished operations from the Progress view.
Menu	The menu contains these options: • Remove All Finished Operations. • Preferences: Starts a dialog box where you can indicate whether the view must display sleeping and system operations.

Icons

The following icons are used in the Progress view

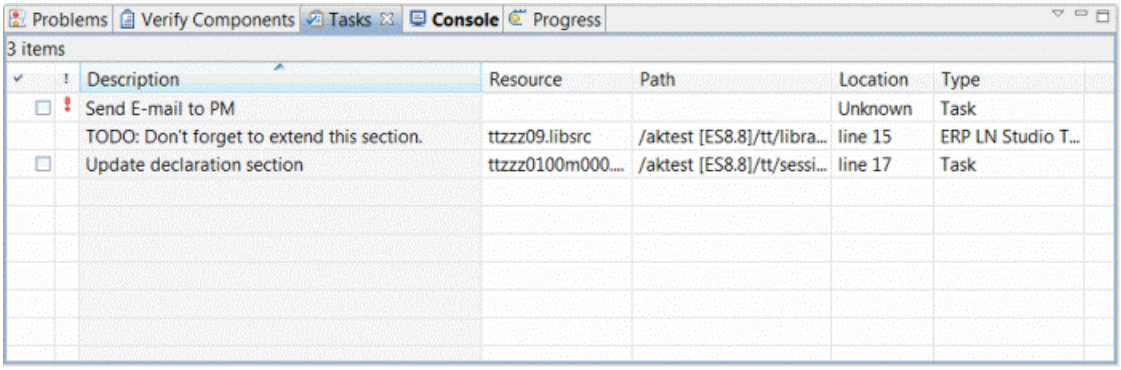
Icon	Description
	Process
	Cancel Operation

Tasks view

The Tasks view displays *Eclipse Tasks* you add manually.

Each Eclipse task represents an action that needs to be carried out, such as to modify a particular line in a script. You can associate a task with a resource in the Workbench, but this is not required.

The Tasks view enables you to add new tasks, and to maintain and remove existing tasks.



The second and third task in the screenshot are associated to a line in a script. The first task is not associated.

Note:

- If a task is associated to (a line in) a script or library, double-click the task to edit the script or library. The editor opens the script or library at the involved line.

- You can add new tasks from the LN Studio script editor.
  For more information, see "Source Tab" on page 267.

This table shows the columns in the view:

The first column	Indicates whether the task is completed. You can manually select or clear the check box.
The second column	Indicates whether the task is high, normal, or low priority.
Description	A description of the line item.
Resource	The name of the resource associated with each line item.
Path	The folder in your workspace where the resource is stored.
Location	The line number of the line item within its associated resource.
Type	The type of task, such as "Task" or "Java Task".

You can edit the first three columns directly in the grid, and via the **Properties** command in the shortcut menu.

Toolbar

This table shows the buttons in the toolbar of the view:

	Provides commands to sort and filter the contents of the view, and to define preferences.
Menu	The **Configure Contents** command in the menu starts a dialog box where you can specify filters. You can filter items according to which resource or group of resources they are associated with. This is done by text string within the **Description** field, by task priority, or by task status. **Note:** You can define a specific filter for TODO reminders. See "ToDo Comments" on page 290.

Shortcut menu

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

This menu provides various commands, for example:

- Commands to add or delete tasks
- A command to change the properties of the selected task
- A command to delete completed tasks

See "User interface information" in the *Workbench User Guide*.

Icons

The following icons are used by the Tasks view:

Icon	Description
	High priority task
	Low priority task

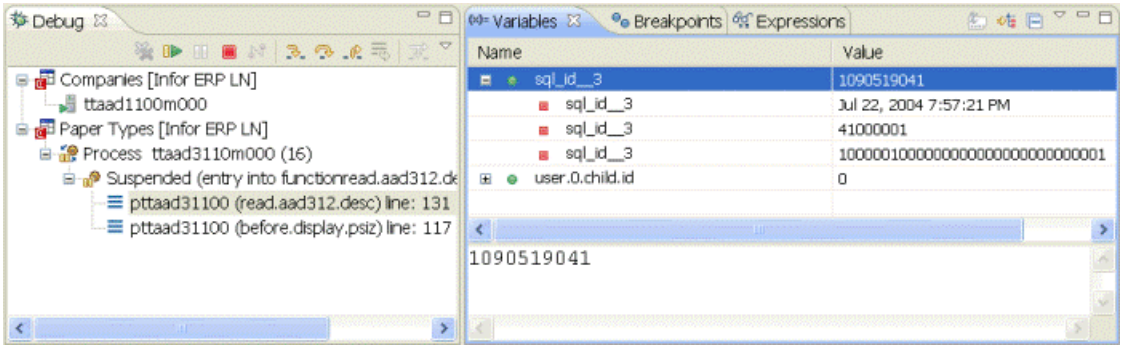
Variables view

The Variables view is part of the Debug perspective.

The Variables view displays a list of variables that exists in the scope of the selected *Stack frame*. The values of primitive variables are displayed. To examine a complex variable, expand the variable to show its members. XML nodes are displayed in an XML tree, which you can collapse and expand.

The view is usually empty. However, it is automatically filled when a session in debug mode is suspended, and a stack frame is selected in the Debug view.

This figure shows an example:



The left part of the screenshot shows the Debug view with the selected stack frame. The right part shows the Variables view with the stack frame's variables. In the screenshot, the Variables view consists of an overview pane (top) and a details pane (bottom).

The details pane shows this information:

- The value of primitive variables.
- The hexadecimal value of variables of type string.
- The XML structure of variables of type XML.

You can determine the amount of information displayed in the Variables view. For example, you can show / hide the following:

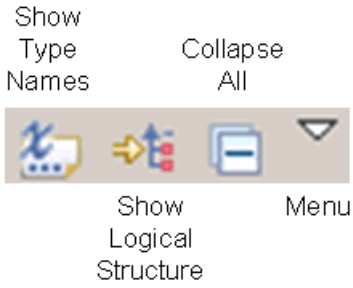
- Data type information.
- The details pane.

See the description of the toolbar and shortcut menu commands.

Note: The Variables view displays all variables that are in scope. If a session contains a lot of variables, the performance can be affected. To boost the performance, close the Variables view and use the Expressions view.

Toolbar

The toolbar of the Variables view contains the following buttons:



Show Type Names	You can toggle this option to show or hide data type names. The data type names are displayed in front of the variable names.
Show Logical Structure	Displays certain complex data structures in a more compact and meaningful form. For example, the elements of a list are displayed as an ordered collection.
Collapse All	Collapses all expanded variables.

Menu	Contains these menu items:
	• Layout: Opens a menu with the following options:
	  • Vertical View Orientation: The details pane is displayed below the overview pane.
	  • Horizontal View Orientation: The details pane is displayed next to the overview pane.
	  • Variables View Only: The details pane is not displayed.
	  • Show Columns: The overview pane displays columns with, for example, the names and values of the variables.
	  • Select Columns: Starts a dialog box where you can select the columns to be displayed.
	• Show Global Variables
	• Show Local Variables

Shortcut menu

To open the shortcut menu, right-click on a variable in the view. The menu contains these commands:

- **Select All** Ctrl+A
- **Copy Variables** Ctrl+Insert
- **Find** Ctrl+F
- **Change Value**
- **Create Watch Expression**

Use the **Create Watch Expression** command to define a watch expression. The selected variable is copied to the Expressions view. Watch expressions are evaluated each time a session suspends.

Icons and decorators

Icons

The following icons are displayed in the Variables view:

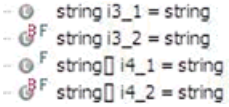
Icon	Description
	Global variable
	Local variable
	External variable
	Internal variable
	Array element
	Array partition

Decorators

The following decorators are used to indicate whether a variable is fixed and / or based. The decorators are displayed in the upper right corner of the variable icon.

Decorator	Description
B	Based variable
F	Fixed variable

This figure shows an example:



Limitations

Filters not correctly initialized

The filters in the Variables view (**Show Global Variables** and **Show Local Variables**) are not correctly initialized. To correct the settings, click the stack frame twice or select a variable in the Variables view.

Verify Components view

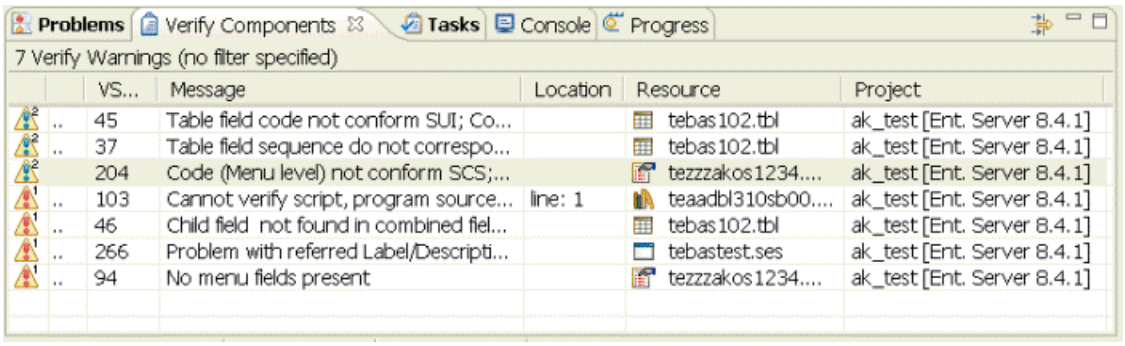
This view shows the *warnings* generated when you verify software components. The verification process performs various checks based on the LN design principles.

For more information on the verification of software components, see these sections:

- "Verifying software components" on page 58
- "Verify Software Components (VSC)" in the Web Help on the LN server.

You can view the generated warnings and the corresponding details and decide to accept each warning individually, or solve the problem.

This figure shows an example:



Note: When you double-click a warning, the appropriate editor for the software component starts automatically at the location where the error occurs.

Severity	The view displays a warning icon with a number that indicates the priority of the *warning*: • 1 - High • 2 - Normal • 3 - Suspicious • 4 - Low These priorities are automatically assigned by the verification process: • The priorities for warnings that are generated based on *source analyze codes* are defined in the Source Analyze Codes (tlvsc3111m000) session.

	• The priorities for warnings that are generated based on all other types of checks are hard coded in the *VSC* software on the LN server. **Note:** If a warning blocks the check in of a component, an error icon is displayed instead of a warning icon.
Accepted	Shows a check mark if the warning is accepted.
VSC ID	The unique code that identifies the message assigned to the warning. For example: the message ID for a warning is 5053. The corresponding message text is `Function 'sleep' is not supported by Web UI .` **Note:** This column corresponds with the **Warning Id** field in the Verify Warning Details dialog box.
Message	The text of the message that is assigned to the warning. For example: `Function 'sleep' is not supported by Web UI`
Location	The location in the software component where the error occurs. For example: • A line number in a script or library. • A form or report in a session.
Resource	The LN Studio resource that represents the software component for which the warning is generated. For example: • tfgld1101m000.ses (session) • tfgld1101m000.sessrc (program script)
Project	The *software project* to which the software components belong.

Toolbar

The toolbar of the Tasks view includes the following buttons:

Verify Warning Filter

Verify Warning Filter	Starts the Verify Components View Filter dialog box. You can, for example, define a filter that displays only warnings with a particular warning level. See "Verify Components View Filter" on page 123.

Shortcut menu

Right-click on any resource in the view to open a menu that contains the **View Details** command. This command starts the Verify Warning Details dialog box. In this dialog box, you can view the warning details, and change the warning status to "Accepted". See the dialog box's help.

Icons

These icons are used in the view:

Icon	Description
	Warning level 1 (High)
	Warning level 2 (Warning)
	Warning level 3 (Suspicious)
	Warning level 4 (Low)
	Accepted
	Blocking error

Infor LN Project Server

14

Infor LN Project Server introduction

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.

Delivering software components

This topic describes how to deliver the components after you end a development activity.

After you end a development activity, you can distribute the corresponding components through PMC.

If the **Integration With PMC** check box is selected in the properties of the project to which the activity belongs, use one of these procedures to deliver the components.

- Generate and deliver a solution in one go. This is the fastest way to deliver components.
- Manually create and deliver a solution. This procedure contains more manual actions, and therefore offers more flexibility.

If the **Integration With PMC** check box is not selected, manually create and release a solution though the PMC sessions on the LN server.

Note: The following procedures describe how to release a PMC solution from Infor LN Project Server. Only the steps to perform the procedure are described. No background information on PMC is included. For details on the PMC terminology and PMC-specific actions, such as Generate Dependencies, Export Solution, and Release Solution, see the *Infor LN - Development Tools Development Guide*.

Manually creating and delivering a solution

1 Choose to perform a manual delivery.

If you end an activity that belongs to a project for which the integration with PMC is enabled, this question is displayed.

```
Do you want to deliver this Activity and create a Solution?
```

To perform a manual delivery, click **No**.

2 Create a delivery activity.

Complete these steps:

a In the Software Project Explorer view, click **New Activity**. The Create a new Activity dialog is displayed.

b Create an activity of type "Delivery". See the online help of the dialog.

c Save the activity and close the dialog.

3 Place the delivery activity in the delivery explorer.

In the Software Project Explorer view, select the delivery activity and drag it to the Delivery Explorer.

Alternatively, use the **Open a Delivery Activity** command in the Delivery Explorer.

4 Add software development activities to the delivery activity.

Add the software development activities, whose components you want to deliver, to the delivery activity.

To do this, drag these activities to the Delivery Explorer and drop them on the delivery activity.

Alternatively, use the **Add Development Activity** command in the shortcut menu in the Delivery Explorer.

5 Add a solution.

In the Delivery Explorer, right-click the delivery activity and, on the shortcut menu, select **Create Solution**. The Create a Solution wizard starts. Enter a code and description for the solution and click **Finish**. The Solution Editor starts.

6 Edit the solution.

Use the Solution Editor to modify the solution.

For example, you can perform these actions:

- Add components to the solution.
- Add installation instructions.
- Define additional dependencies.

See"Solution Editor" on page 353.

When finished, save the changes and close the editor.

7 Generate dependencies.

To regenerate dependencies after you added or deleted components to/from the solution:

a In the Delivery Explorer, right-click the solution.

b On the shortcut menu, select **Generate Dependencies**.

The solution status changes to "Dependencies defined".

8 Export the solution.

In the Delivery Explorer, right-click the solution and, on the shortcut menu, select **Export Solution**.

The solution status changes to "Exported".

9 Release the solution.

In the Delivery Explorer, right-click the solution and, on the shortcut menu, select **Release Solution**.

The solution status changes to "Released".

You are prompted to finalize the delivery activity and remove the activity from the Delivery Explorer.

10 Finalize the delivery.

In the question window, click **Yes**. The delivery activity disappears from the Delivery Explorer and its status is set to "Finalized".

This step does not change anything to the solution.

Note: If you click **No**, you must complete these steps to manually finalize the delivery activity:

a In the Delivery Explorer, right-click the delivery activity.

b Select **Finalize Delivery**.

Generating and delivering a solution in one go

1 Choose to generate and deliver a solution in one go.

If you end an activity that belongs to a project for which the integration with PMC is enabled, the following question is displayed.

```
Do you want to deliver this Activity and create a Solution?
```

To generate and deliver a solution in one go, click **Yes**. The Create a Solution wizard starts.

2 Create and release a solution.

In the Create a Solution wizard:

a Specify the name and description of the delivery activity to be generated, or accept the defaults.

b Specify the code and description of the solution to be generated, or accept the defaults.

c Select **Export and release solution without editing**.

d Click **Finish**.

The wizard performs these actions:

- Generates a delivery activity.
- Generates a solution.
- Exports the solution.
- Releases the solution.
- Finalizes the delivery activity.

Note: If you want to edit the solution before the release, select **Edit solution before export and release** instead of **Export and release solution without editing**. When you click **Finish**, the Solution Editor starts. In this editor you can, among other things, add components to the solution and define dependencies. See the editor's online help. After you have edited the solution, you must manually perform these actions:

- Generate Dependencies.
- Export the solution.
- Release the solution.
- Finalize the delivery activity.

Dialogs

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 will be ended. These components will be checked in to the project VRC, while the other components stay available in the activity VRC.

See "Developing software components" on page 55.

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" on page 336.

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 will be 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
Enter an activity name.

Description
Enter 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
Enter 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.

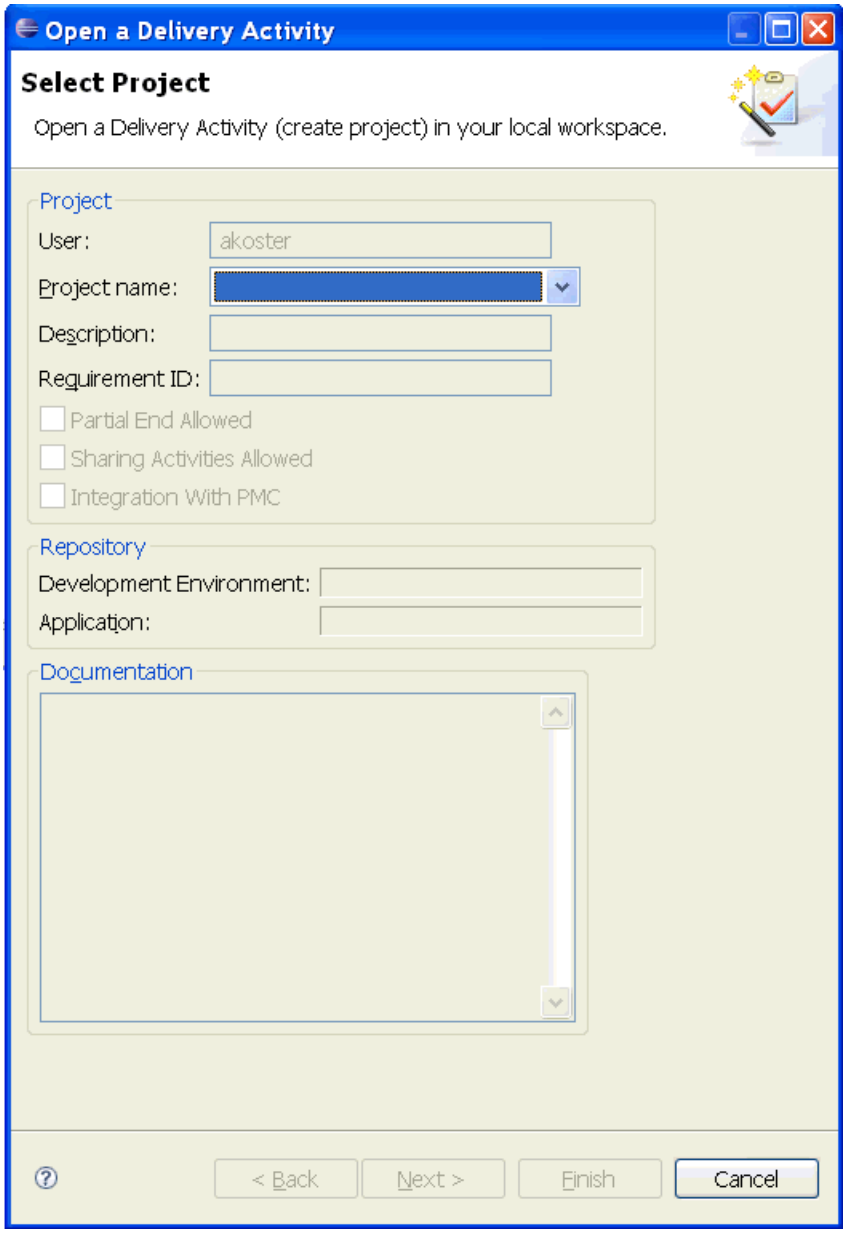
Open a Delivery Activity

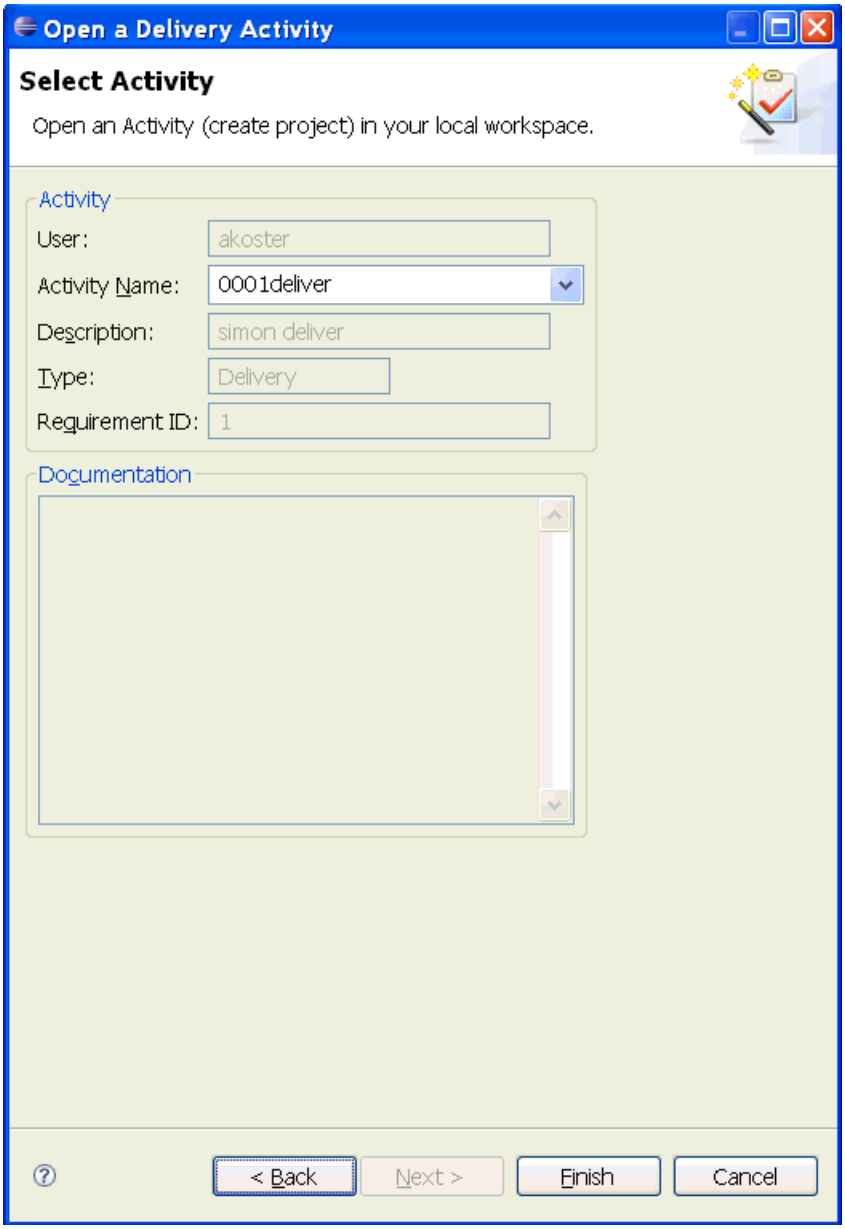
Use this wizard to open a delivery *activity*.

To open an activity:

1. In the Select Project page, select your *software project* from the list and click **Next**. The Select Activity page is displayed.
2. Select your activity from the list and click **Finish**.

These figures show an example:


Open a Delivery Activity
Select Project
Open a Delivery Activity (create project) in your local workspace.
Project
User:
akoster
Project name:
Description:
Requirement ID:
Partial End Allowed
Sharing Activities Allowed
Integration With PMC
Repository
Development Environment:
Application:
Documentation
< Back
Next >
Finish
Cancel



Select Project

Use this page to select the project to which the activity belongs.

All fields, except **Project name**, are read-only.

Project name
The project to which the activity belongs.

You can select the project name from a drop-down list.

All other fields
For details on the other fields, see "Create a Software Project" on page 339.

Select Activity

Use this page to select the activity you want to open.

All fields, except **Activity name**, are read-only.

Activity name
The activity you want to open.

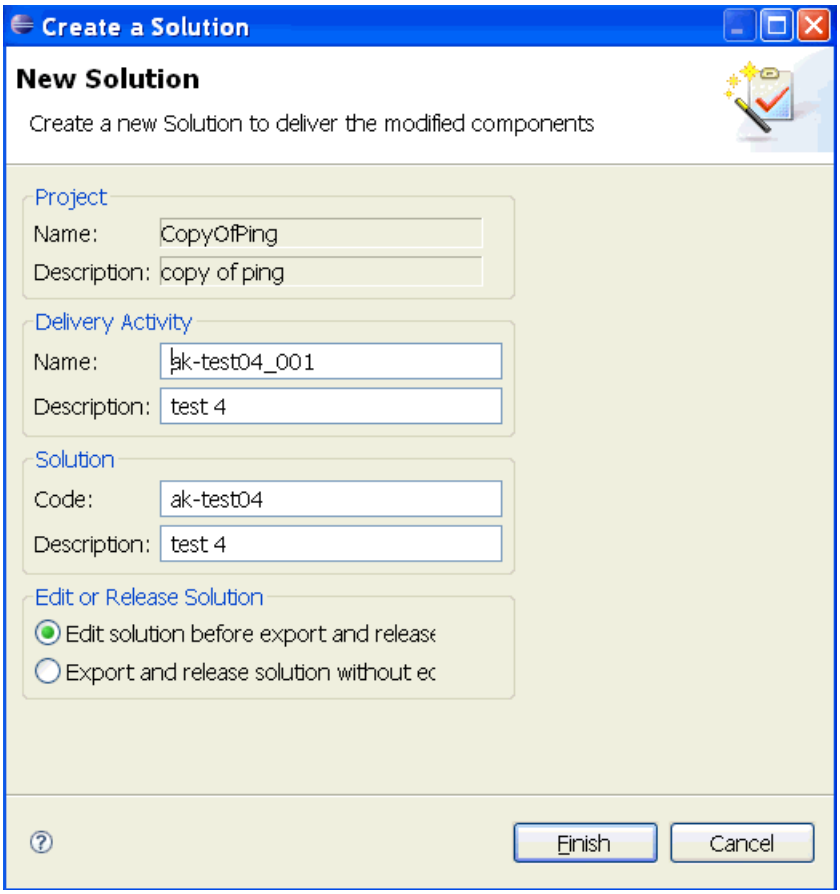
You can select the activity name from a drop-down list.

All other fields
For details on the other fields, see "Create a new Activity" on page 340.

Create a Solution

Use this wizard to create and deliver a new PMC solution.

This figure shows an example:



Project

Name
The *software project* to which the solution belongs.

Description
The project description.

Delivery Activity

Name
The name of the delivery *Activity* to which the solution belongs.

The wizard generates this activity and links it to your development activity.

Description
The activity description.

Solution

Name
The name of the solution to be generated.

Description
The description of the solution.

Edit or Release Solution

Edit solution before export and release
If this option is selected, you can edit the solution before it is exported and released. When you click **Finish**, the Solution Editor starts.

After you have edited the solution, you must manually perform these actions:

- Generate Dependencies (if you defined dependencies).
- Export the solution.
- Release the Solution.
- Finalize the delivery activity.

See "Delivering software components" on page 336.

Export and release solution without editing
If this option is selected, you cannot edit the solution before it is exported and released.

When you click **Finish**, the wizard automatically performs these actions:

- Exports the solution.
- Releases the Solution.
- Finalizes the delivery activity.

Add development activity

Use this dialog to add an existing development *activity* to a delivery activity.

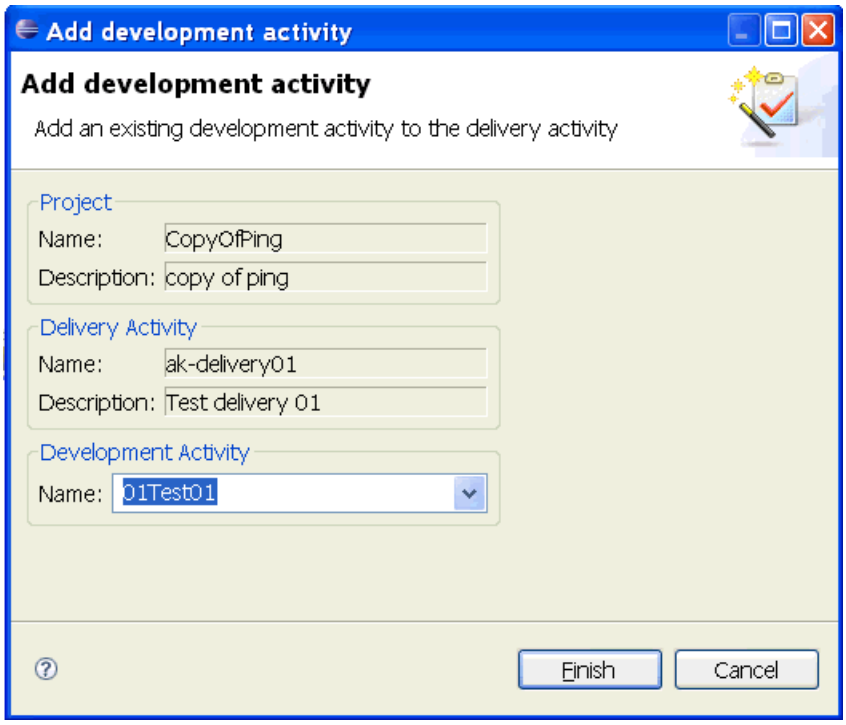
You can start this dialog from the Delivery Explorer.

To add an existing development activity to a delivery activity:

1 In the **Name** field, select a development activity from the list.

2 Press **Finish**.

The development activity is displayed, under the delivery activity, in the Delivery Explorer.



Project

Project name
The *software project* to which the delivery activity belongs.

Description
The description of the selected project.

Delivery Activity

Name
The delivery activity to which the development activity is added.

Description
The description of the delivery activity.

Development Activity

Name
The name of the development activity. Select a development activity from the drop-down list.

Browse Solutions

You can start this dialog box from the Solution Editor.

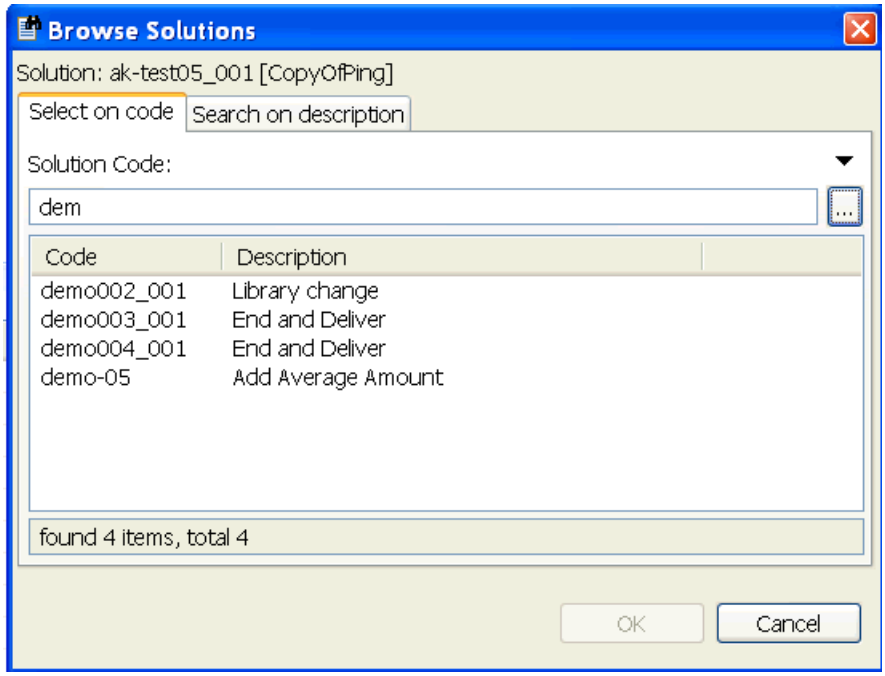
Use the dialog box to link another solution to the solution you are editing.

The dialog box consists of two tabs, that you can use to search in two different ways:

- Select on code
- Search on description

Select on code

Use this tab to search on the first part of the solution code.



To link a solution to the solution you are editing

Complete these steps:

1 Optionally: In the upper right part of the tab, click ▼. The Browse Solution Types dialog box starts. In this dialog box you can specify additional settings for the search process.

2 In the **Solution Code** field, enter the first part of the solution code.

3 Click ..., or press CTRL+SPACE. A list of solutions, which match the pattern, is displayed.

4 Select the desired solution and click **OK**.

To sort search results

- To sort the search results by code or description, click the corresponding column header.
- To toggle between ascending and descending sort mode, click the involved column header.

Fields

Solution Code
Enter the (first part of the) solution code you are searching for.

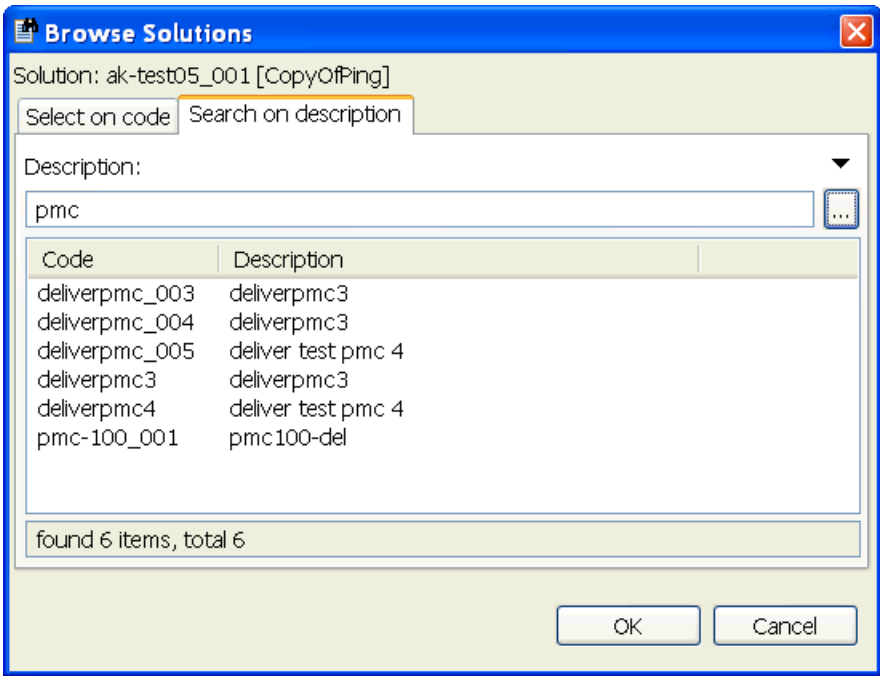
To display a list of solutions, which match the pattern, click ..., or press CTRL+SPACE.

Code
The solution code.

Description
The description of the solution.

Search on description

Use this tab to search on a part of the solution's description.



To link a solution to the solution you are editing:

1. Optionally: In the upper right part of the tab, click ▼. The Browse Solution Types dialog box starts. In this dialog box you can specify additional settings for the search process.
2. In the **Description** field, enter the search pattern.
3. Click ···. A list of solutions that match the search pattern is displayed.
4. Select the desired solution and click **OK**.

Fields

Description
Enter the search pattern.

To display a list of solutions, which match the pattern, click ···.

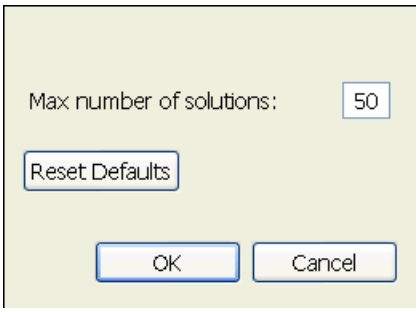
Name
The solution code

Description
The description of the solution.

Browse Solution Types

Use this dialog to specify additional settings for the search process that you want to start in the Browse Solutions dialog.

This figure shows the Browse Solution Types dialog box:



Max number of solutions
Specify a maximum number of solutions to be displayed per search.

Reset Defaults
Restores the default settings for this dialog.

Select Component(s) for delivery

Use this dialog box to link components to a solution.

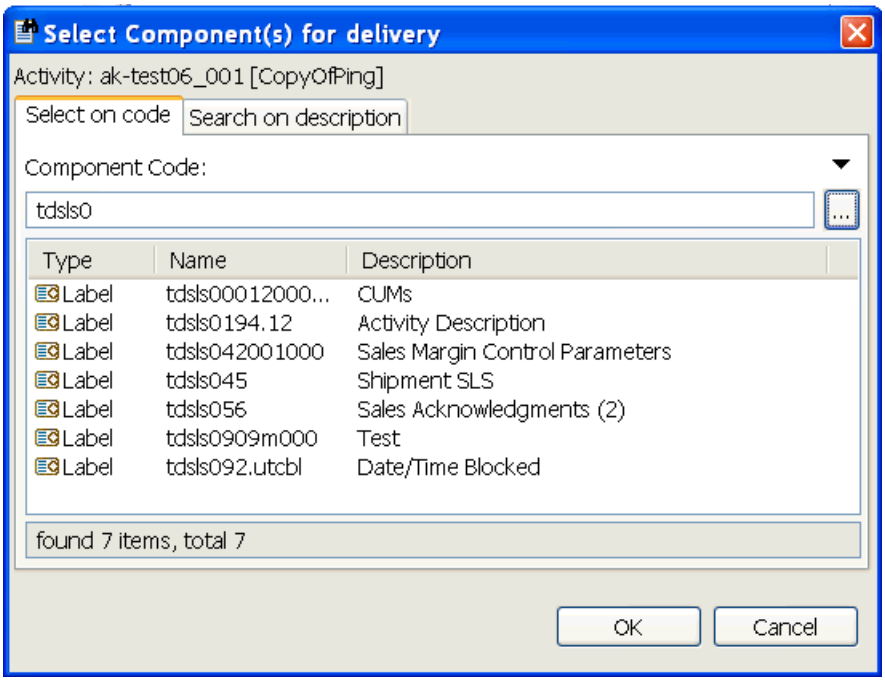
You can start the dialog box from the Solution Editor.

The dialog box consists of two tabs, that you can use to search in two different ways:

- Select on code
- Search on description

Select on code

Use this tab to search on the first part of the component code.



To add a component to the solution you are editing:

1. Optionally: In the upper right part of the tab, click ▼. A dialog box, where you can specify additional settings for the search process, starts.
2. In the **Component Code** field, enter the first part of the component code. For example: `tdsls0`.
3. Click ..., or press CTRL+SPACE. A list of components, which match the pattern, is displayed. If you type additional characters, The dialog box automatically updates this list.
4. Select the desired component and click **OK**.

To sort search results:

- To sort the search results by type, name, or description, click the corresponding column header.
- To toggle between ascending and descending sort mode, click the involved column header.

Note: As you type the package code and the module code (the first 5 characters of the component code), The dialog box automatically shows a list of matching packages and modules, and updates this list after each character typed. For example:

Component Code	Search results displayed
t	All packages starting with t.
td	Package td and all its modules.
tdr	All modules of package td that start with r. For example, tdrec and tdrpl.

Fields

Component Code
Enter the (first part of the) component code you are searching for.

To display a list of components, which match the pattern, click ..., or press CTRL+SPACE.

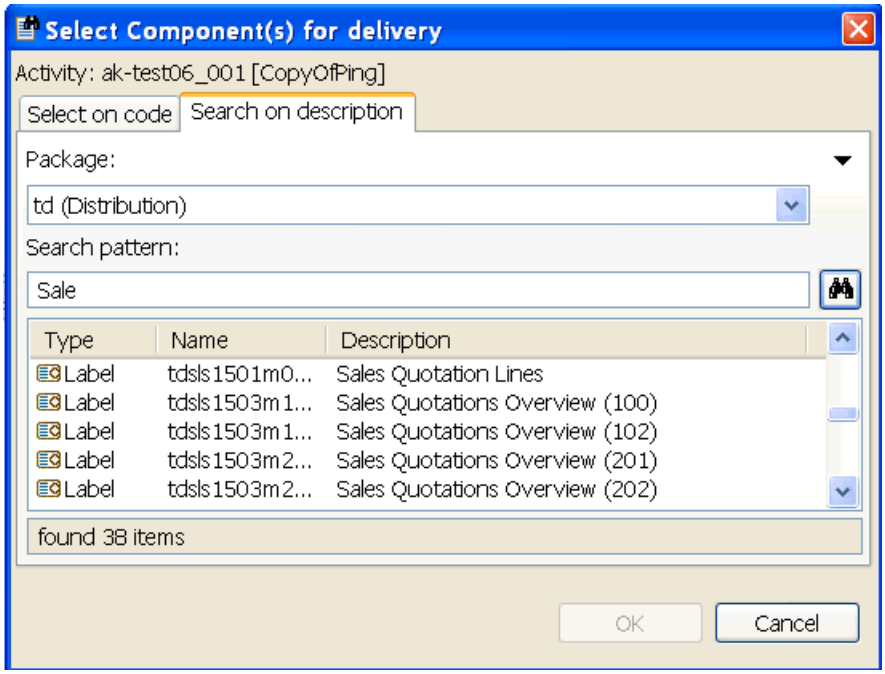
Type
The component type, such as session or label.

Name
The component name.

Description
The description of the component.

Search on description

Use this tab to search on a part of the component's description.



To add a component to the selected activity:

1. Optionally: In the upper right part of the tab, click ▼. A dialog box, where you can specify additional settings for the search process, starts.
2. In the **Package** field, select a package from the list.
3. Enter the search pattern, and click 🔭. A list of components that match the search pattern is displayed.
4. Select the desired component and click **OK**.

Fields

Package
Select a package from the list.

Search Pattern
Enter the search pattern.

To display a list of components, which match the pattern, click 🔭.

Type
The component type, such as session or label.

Name
The component name.

Description
The description of the component.

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.

Multipage Editors

Solution Editor

Introduction

Use this editor to define and edit PMC solutions.

In PMC, a solution is 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.

Overview

Use this page to perform these actions:

- Maintain the basic data of a solution.
- Specify pre- and post-installation instructions.
- Link additional components to the solution.
- Define dependencies between the current solution and other solutions.

General Information

Name
A unique identification of the solution in a *Base VRC*.

The solution code has these characteristics:

- Datatype: alpha-numerical string.
- Maximum length: 14 characters.
- Forbidden characters: All characters that have a special meaning in UNIX, such as $, *, and so on. If you enter a forbidden character, the code will not be accepted.

Description
The description of the solution.

Base VRC
Base VRC

Status
Solution status distributor

Published
If this check box is selected,, the solution is published immediately upon release. If a solution is published, the solution is copied to the directory specified in the **Path for Published Solutions** field in the Parameters (ttpmc0100s000) session.

Note: If the **Copy Published Solutions** check box in the Parameters (ttpmc0100s000) session is cleared, the solution cannot be published, regardless of the selection of the current check box.

Installation Instructions

Pre
The pre-installation instructions. The instructions are added to the solution text.

For example: `Before you install this solution, ensure there are no open integration transactions.`

Post
The post-installation instructions. The instructions are added to the solution text.

For example: `After you install this solution, generate MRP orders again.`

Components

Components
Use the grid to link components to the solution.

Add / remove components

- To add a component, click **Add** and fill out the fields in the grid.
- To remove a component, right-click a component and, on the shortcut menu, select **Remove**.

Type
The type of software component, such as session, message, or label.

Name
The component identification, such as a session code.

To add a component:

1 Press CTRL+SPACE. The Select Component(s) for delivery dialog box is displayed. The dialog box only shows components of the selected type.

2 Select one or more components and close the dialog box. See the dialog box's online help.

Dependencies

Dependencies
Use the grids to define dependencies between the current solution and other solutions.

If not all dependencies that would be generated are present, you receive a warning during the export of the solution or patch.

After you define all the dependencies, you can change the solution status to **Dependencies defined** and the solution is ready to be exported. You can change the status in the Delivery Explorer.

If the status of the solution was **Exported**, and you change the dependencies, then the solution status is reset to **Dependencies defined**.

Add / remove dependencies

- Use the upper grid, to enter dependencies of type *Co-requisite* and *Pre-requisite*.
- Use the lower grid to indicate the solution is a *Post-requisite* of another solution.
- To add a dependency, click the relevant **Add** button and fill out the fields in the grid.
- To remove a dependency, right-click a dependency and, on the shortcut menu, select **Remove**.

Type
The type of dependency: Co-requisite or Pre-requisite.

Related VRC
The *Base VRC* of the solution or *Patch* to which the current solution or patch is related.

The related VRC can be a different base VRC if the dependency type is Co-requisite or a Pre-requisite from a *Feature Pack* to the previous Feature Pack.

Solution
The solution or *Patch* with which a *Dependency* relation exists.

To add a solution:

1 Press CTRL+SPACE. The Browse Solutions dialog box is displayed.

2 Select a solution and close the dialog box. See the dialog box's online help.

Description
The solution description.

Manual
If this check box is selected, the dependency was not generated, but defined manually.

This solution is post-requisite of:

Related VRC
The *Base VRC* of the solution or *Patch* to which the current solution or patch is related.

Solution
The solution or *Patch* with which a *Dependency* relation exists.

To add a solution:

1 Press CTRL+SPACE. The Browse Solutions dialog box is displayed.
2 Select a solution and close the dialog box. See the dialog box's online help.

Description
The solution description.

History

Use this page to display the status changes to a solution for a *Base VRC*.

Every status change of a solution or patch status is recorded. This page shows all these recorded changes. If you delete a solution, this fact is recorded with the new status as blank.

The page shows the maintenance history for the relevant solution or patch up until the specified time.

Maintenance History

Date
The date on which the status of the solution was changed.

Time
The time when the status of the solution was changed.

Serial
Number to make the key unique.

User
The name of the user who changed the solution.

Status
The new status of the solution.

Perspectives

Project Server perspective

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

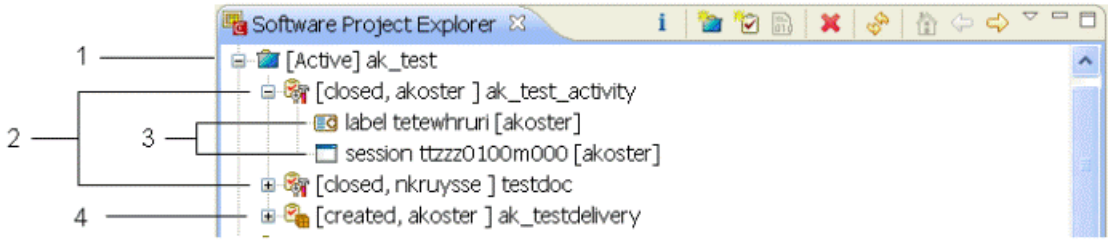
This table shows the views in the perspective:

"Software Project Explorer view" on page 357	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 335.
"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 "Defining an application" and "Create a new Application" on page 132.

Views

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.



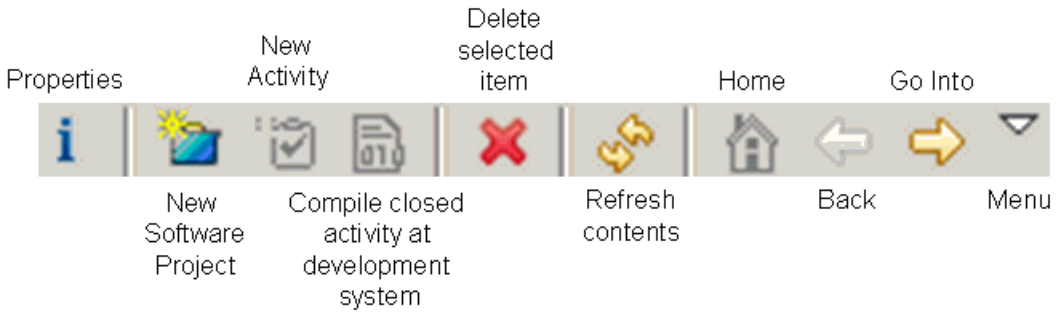
This table describes the different nodes in the previous figure:

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

Nr. in figure	Description
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 353.

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

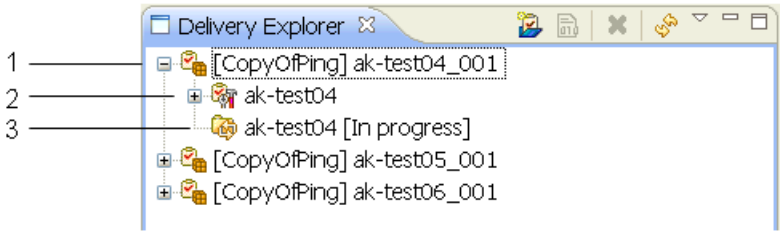
The following icons are displayed in the Software Project Explorer view:

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

Delivery Explorer view

The Delivery Explorer view provides a hierarchical view of the delivery activities, and the corresponding development activities and solutions.

This figure shows an example:

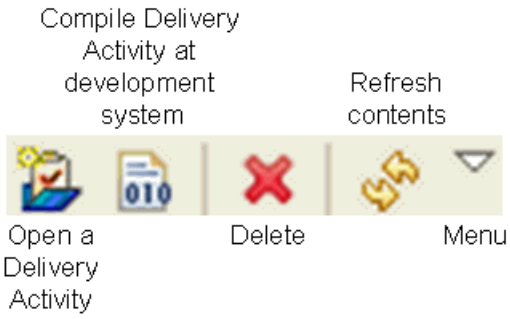


This table describes the different nodes in the previous figure:

Nr. in figure	Description
1	Delivery Activity
2	Development Activity
3	Solution

Toolbar

The toolbar of the Delivery Explorer view contains these buttons:



Open a Delivery Activity	Starts the Open a Delivery Activity wizard, where you can select an activity that is assigned to you.
Compile Delivery Activity at development system	Compiles the software components in the selected delivery activity.
Delete	Deletes the selected item.
Menu	Contains the **Open a Delivery Activity** 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 following pop-up menu, right-click on a resource in the view:

This menu provides the following Project Server-specific commands:

Add Development Activity	Starts the Add development activity dialog, where you can link a development activity to the selected delivery activity.
Compile Delivery Activity	Compiles the software components in the selected delivery activity.
Create Solution	Starts the Create a Solution dialog, where you can create a PMC solution.

Generate Dependencies	Generates the dependencies defined in the selected solution. For details on the PMC terminology and PMC specific actions, such as Generate Dependencies, Export Solution, and Release Solution, see the *Infor LN - Development Tools Development Guide*.
Export Solution	Exports the solution.
Release Solution	Releases the solution.
Finalize Delivery	Sets the status of the selected delivery activity to **Finalized**. The delivery activity disappears from the Delivery Explorer.

Icons

These icons are displayed in the Delivery Explorer view:

Icon	Description
	Delivery Activity
	Development Activity
	Solution

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. See "Activity based development" on page 39.

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" on page 132 for which the *Software Projects* are defined. For more information on the administrator's tasks, refer to "Defining an application".

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*.

See "Activity based development" on page 39.

Check out

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

See "Activity based development" on page 39.

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.

See "Code Assist" on page 276.

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 needs to be done, 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. For more information, refer to "Tasks view" on page 327, "Source Tab" on page 267, and "ToDo Comments" on page 290.

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 will be changed in the *Private VRC*. If SCM is not active, these components will be 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 will remain and will be 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*.

For more information on the project manager's tasks, see "Defining a software project" on page 335.

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 done 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 will be moved to history and a new registry will be 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.

See "Activity based development" on page 39.

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, will be 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.