Infor LN Application Studio Administration Guide

Publication Information

Release: 10.3.1

Publication Date: June 4, 2014

Document number: U9420JUS

Contents

About this Guide

Document Summary

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

Contacting Infor

If you have questions about Infor products, go to the Infor Xtreme Support portal at http://www.infor.com/inforxtreme.

If we update this product or document after the product release, we will post the new version on this Web site. We recommend that you check this Web site periodically for updates.

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

Upgrade notice

1

Important note for users upgrading from Infor LN Application Studio 8.4.2 to 10.3.1

If you have implemented Application Studio 8.4.2 already, the following additional steps must be performed before you can start to use version 10.3.1:

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 is changed in 10.3.1: you can now connect to multiple development servers using the same workspace.

To define those development servers, complete the following steps:

- **a** Install Application Studio 10.3.1 (note that the development server and project server must have been upgraded already to Enterprise Server 8.5 or higher). See "Installing Application Studio" on page 27.
- **b** Start Application Studio, using a new workspace.
- **c** Configure the Project Server connection. See "Defining connectivity settings" on page 28.
- **d** Define the development environment(s) in the Application Explorer view. See "Defining a development environment" on page 31.

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 28.
- **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".

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

Infor LN Application Studio introduction 2

Application 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. Eclipse is used as a framework for the Infor LN Application Studio, and various other Infor solutions such as Business Studio, an integration environment.

Application Studio enables you to:

- 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.

For more information see the "Application Studio overview" on page 13 section.

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

Application Studio overview

This overview describes the following:

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

Functions and features

This section describes some key characteristics of Application Studio.

Activity based development

The software development process in Application 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 Application 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 22.

Configuration Management (CM)

Currently the configuration management in Application 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 platform, 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 Application Studio workbench is based on Eclipse, and offers these features:

Wizards	Application 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". You can edit the new components using component-specific Multipage Editors.
Multipage Editors	Application Studio contains various multipage editors that you can use to edit software components, such as sessions, menus, labels, libraries and messages. See "Multipage Editors".
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".
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 an software project and is assigned to a user (software engineer). Activities and software projects are stored on the *Project Server*. The Activity Explorer view displays the following components: • Additional File • 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". **Notes** • 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: • Use the **Get** command in the "Component Explorer view". • Use the **Select a Software Component** command in the Application Studio toolbar. • Use the "Open Declaration" command in the "Script Editor". .

	• Project Managers can create new activities through the "Software Project Explorer view" on page 73.
Problems view	Shows errors and warnings from a project build (compilation). See "Problems view".
Tasks view	Shows *Eclipse tasks*. Each Eclipse task represents an action that needs to be done, e.g. 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 Application Studio script editor. For more information, see "Tasks view", "Source Tab", and "ToDo Comments".
Verify Compo-nents 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" • "Verify Software Components (VSC)" in the Web Help on the LN back-end.
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 op-tions	The **Run** command enables you to run sessions from the Application 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" and "Debugging LN sessions".

For details on the these features, see the "Application Studio workbench" on page 17 topic.

Note: To use Application Studio for customizations, such as changing reports and forms, you do not need an additional license. However, if you want to use Application 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 Application Studio functionality depends on the Enterprise Server version of the LN back-end. For an example, see "Application Search".

User roles

The following user roles can be distinguished in Application 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" on page 32.

Role	Description
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 65.
Software Engineer	Software engineers develop software components through the Application 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".

Current scope and limitations

Currently the configuration management in Application 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 Application Studio, see "Application Studio Limitations" on page 51.

Architecture

The Application Studio is implemented in the Eclipse 3.5 framework.

The Application 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.

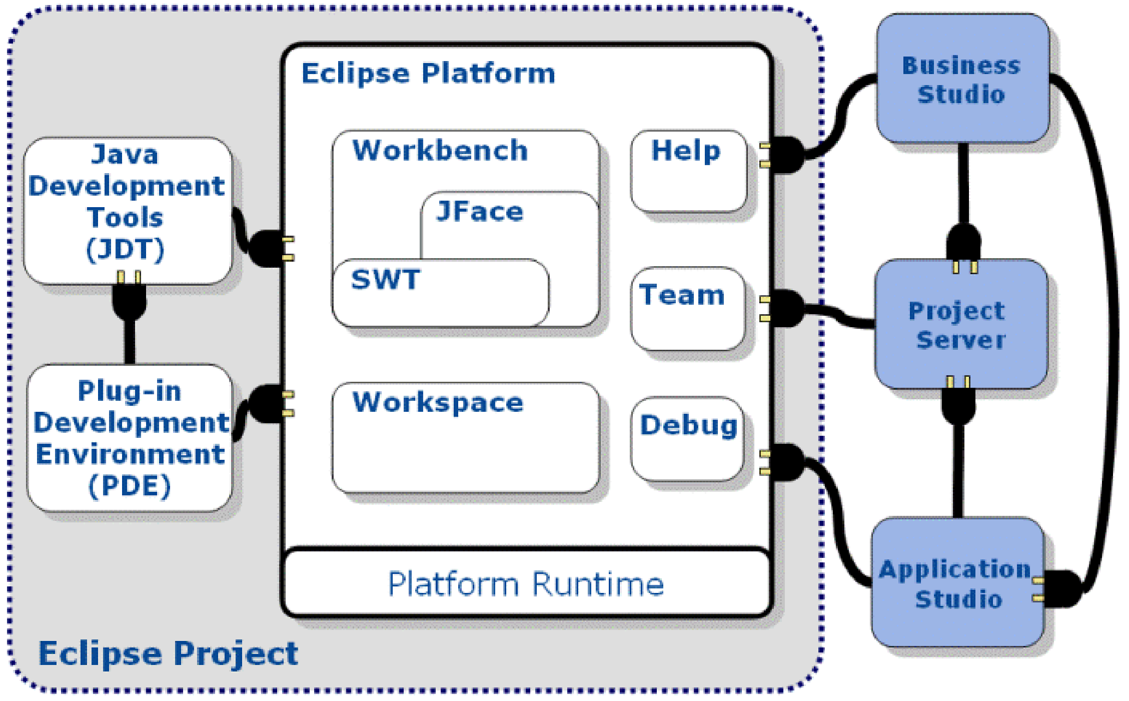
The connection between the client PC and the server(s) is based on JCA: the Application Studio connects through a JCA connectivity plug-in on the client PC to the server.

The following Infor features are available for the Eclipse framework:

- Application Studio Plug-in
- Project Server Plug-in
- Business Studio Plug-in: Business Studio is an Infor platform to build integration technology.
- Reporting Studio Plug-in: Reporting Studio is an Infor platform to create query based reports.
- Infor Connectivity Plug-in

- Infor Solution License Manager Plug-in

The following figure shows an overview of the Application Studio architecture:



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

Troubleshooting

If you encounter any error messages, e.g. on connectivity, please read the following log files for detailed information:

- the Eclipse log files in your local workspace. To find the location of your workspace, use the **Switch Workspace** command on the **file** menu in the Eclipse workbench.
- the log file of the Connectivity plug-in. To find the location of this log file:

  1 On the **Windows** menu, select **Preferences**. The Preferences dialog is displayed.

  2 In the tree that is displayed in the left pane of the dialog, expand the **Infor LN JCA Connectivity** node. Then, click **Log configuration**. The log file location is displayed.

Application Studio workbench

This section describes the structure of the Infor LN Application Studio Workbench

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

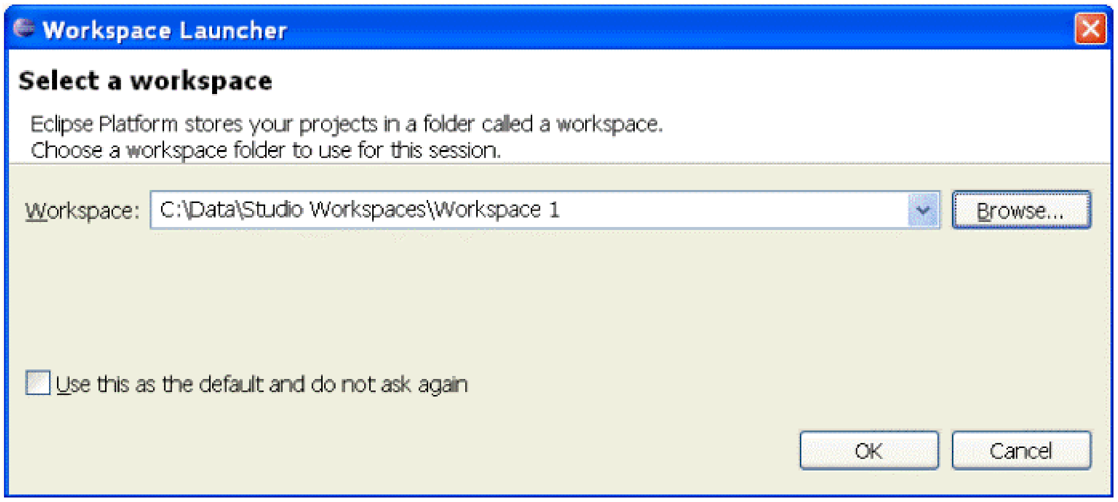
You can use the workbench to:

- 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.

Selecting the workspace

When you launch the Application Studio Workbench, a dialog appears that allows you to select where the workspace should be located. 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 Application Studio software or the Eclipse software is installed. It is better to select a different directory, e.g. 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 Application Studio and Eclipse software. This figure shows an example:


Workspace Launcher

Select a workspace

Eclipse Platform stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\Data\Studio Workspaces\Workspace 1

You can select the **Use this as the default and do not ask again** check box to prevent that this dialog is displayed again.

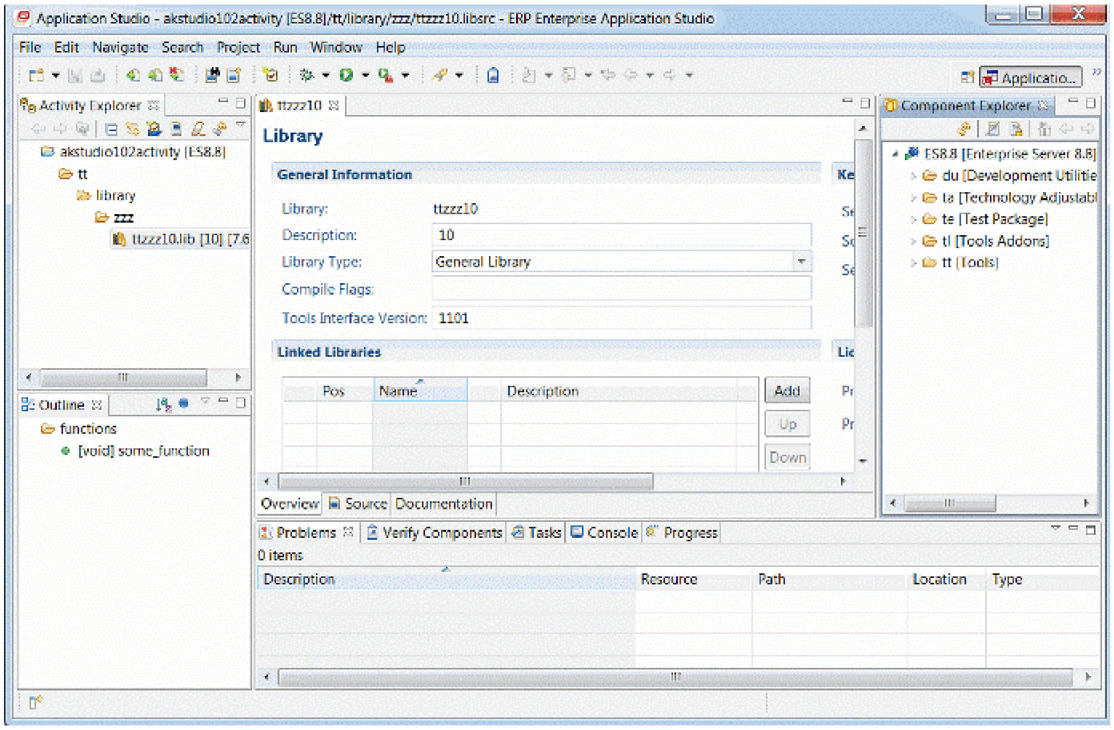
After the workspace location is chosen, a single Workbench window is displayed.

Note: If you want to select another workspace after the Workbench is started, you must use the **Switch Workspace** command on the **file** menu.

Perspectives, editors and views

You can open multiple Workbench windows simultaneously.

This figure shows a sample Workbench window:



Library

General Information

Library:	ttzzz10
Description:	10
Library Type:	General Library
Compile Flags:	
Tools Interface Version:	1101

Linked Libraries

Pos	Name	Description

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 Application Studio workbench that is typically used to edit or browse a resource, e.g. 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 Application Studio workbench window.

A view is a visual component within the Application Studio workbench. It is typically used to navigate a hierarchy of information (e.g. to browse the resources in the development repository), open an editor, or display 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.

Examples of typical Application Studio views are:

- Activity Explorer
- Software Project Explorer
- Component Explorer
- Verify Components

In Application Studio, these perspectives are used:

- Application Studio
- Debug
- Project Server

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 Studio perspective combines views that you would commonly use while editing scripts and libraries, while the Debug perspective contains the views that you would use while debugging programs. As you work in the workbench, you will probably switch perspectives frequently.

For details on these perspectives, see the following sections:

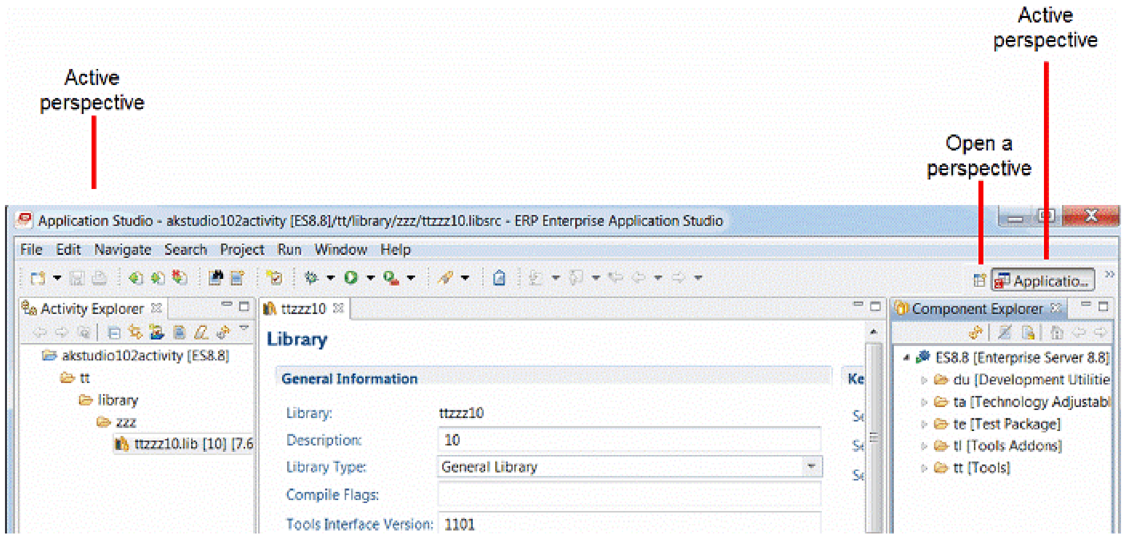
- "Application Studio perspective"
- "Debug perspective"
- "Project Server perspective" on page 73

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 Application Studio perspective is in use.



Library:	ttzzz10
Description:	10
Library Type:	General Library
Compile Flags:	
Tools Interface Version:	1101

Running sessions

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

See "Running LN sessions".

Preferences

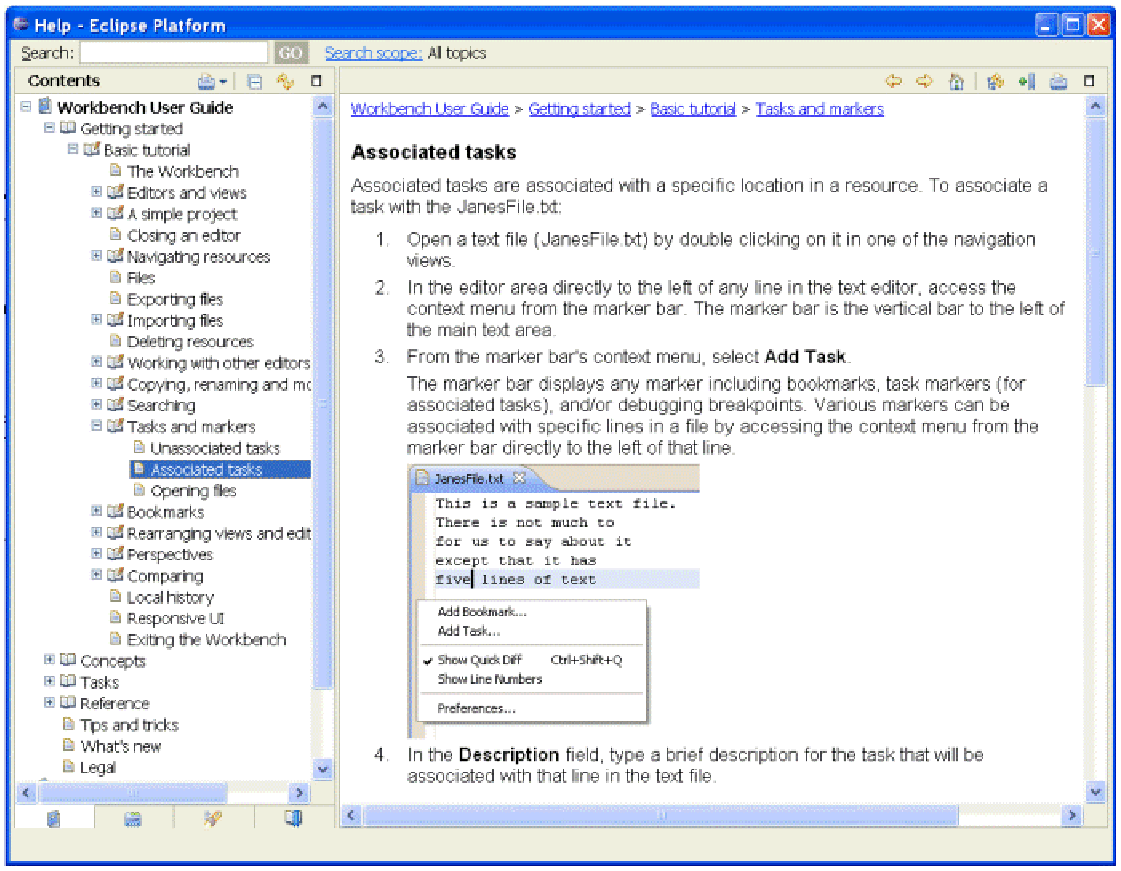
You can set various user preferences for the Application Studio workbench. For example: connectivity settings, preferences for the editor, and label decorations.

See the "Setting user preferences" section for details.

More information

The Application 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. See this figure:



Associated tasks

Associated tasks are associated with a specific location in a resource. To associate a task with the JanesFile.txt:

1. Open a text file (JanesFile.txt) by double clicking on it in one of the navigation views.
2. In the editor area directly to the left of any line in the text editor, access the context menu from the marker bar. The marker bar is the vertical bar to the left of the main text area.
3. From the marker bar's context menu, select **Add Task**.

   The marker bar displays any marker including bookmarks, task markers (for associated tasks), and/or debugging breakpoints. Various markers can be associated with specific lines in a file by accessing the context menu from the marker bar directly to the left of that line.

```
This is a sample text file.
There is not much to
for us to say about it
except that it has
five lines of text
```

4. In the **Description** field, type a brief description for the task that will be associated with that line in the text file.

For information on the extra functionality developed by Infor, refer to "Application Studio overview" on page 13.

To access the *Workbench User Guide* and the Application Studio online help, click **Help Contents** on the **Help** menu.

Applications

This topic describes the concept of *applications*.

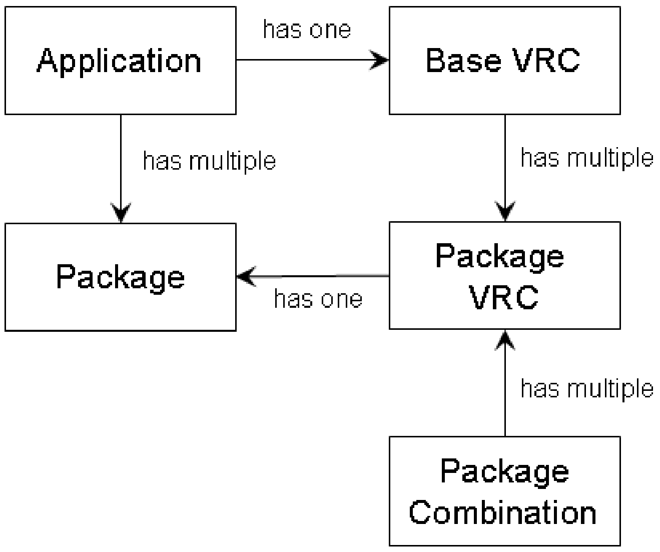
Application Studio uses *applications* to group software components created or modified during a software development project. This is done by bundling one or more LN packages.

An Application 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 Application 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 ES Software Component wizards only allow you to create new components in these packages. This restriction also applies to the Duplicate an Infor ES Software Component wizard.

Installation and configuration

4

First installation

Prerequisites

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

Licenses

To use Application 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 only required if you want to create or modify tables, domains, UI scripts, functions, or libraries.

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)
- MS Windows 32-bit platform (XP or Vista) or MS Windows 64-bit platform (Windows 7)
- Java Runtime Environment (JRE) 1.6 (Java 6)
  You can download the JRE software from the http://java.sun.com website.

Installing Application Studio

For details on the installation of Application Studio, see "Installing Application Studio" on page 27.

Preparing the installation

Note: This section is intended for system administrators.

Preparing the first installation

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

1. Download or copy the zip files with the Application Studio software.
   These zip files are available:
   - `InforApplicationStudio_10.3.1.nnn.zip` (32-bits)
     You can download this file from solution 1009745 on the http://www.infor.com/inforxtreme site.
   - `InforApplicationStudio_10.3.1.nnn-x86_64.zip` (64-bits)
     You can copy this file from the installation medium.

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

   Important: 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. Ensure users can access the zip files. For example, put the files on a network share.

Preparing the installation of updates

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

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

Installing Application Studio

This section describes how you can install Application Studio, Project Server, and the corresponding plug-ins.

Note:

- For information on the prerequisites for the installation of Application Studio, see "Prerequisites" on page 25.
- Before you can install Application Studio, the system administrator must give you access to the zip file with the Application Studio software. See "Preparing the installation" on page 26.

Installing Application Studio

To install Application Studio:

1. Create an installation folder

   Create a new folder on your machine.

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

2. Extract the Application Studio zip file

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

   For Application Studio 10.3.1 these zip files can be available:

   - `InforApplicationStudio_10.3.1.nnn.zip` (32-bits)
   - `InforApplicationStudio_10.3.1.nnn-x86_64.zip` (64-bits)

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

   Important: 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.

   The extraction creates an `eclipse` folder with various sub folders. To start Application Studio, run the `eclipse.exe` executable in the `eclipse` folder.

Note: If you upgrade from Application Studio 8.4.2 to Application Studio 10.3.1, you must perform the actions described in the "Upgrade notice" on page 9.

Post installation steps

1. Configure static connection points

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

- Project Server
- Debug

See "Defining connectivity settings" on page 28.

2 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

See "Defining dynamic connection points".

3 Define Application Studio preferences

a On the **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 Application Studio**. For details about the preferences, see the "Preferences Pages" section in this guide or the dialog's online help.

Defining connectivity settings

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

Infor LN Application 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.

To define the connectivity settings, you must configure the following static connection points:

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 Application 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 Application Studio, the debugger on the LN server uses this connection point to send messages to the Application Studio on the client PC.

Configuring static connection points

To configure the connection points, complete the following steps:

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 Application 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.

Connection point	Points of attention
ProjectServer	Select Activation Type **BW**, **Rexec**, or **BaanLogin**, and specify the corresponding settings. The company number for these connection points must be **000**. The application and project data accessed through these connection points can reside in the same LN environment. If so, you can share these connection points. As a result, only one bshell is used to connect to the back-end, instead of two, which will improve the performance.
Debug	Select Activation Type **Socket-In**. Select a free local port number, for example 7900.

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

Note: You can use the Preferences dialog to set various other user preferences for the Application Studio workbench. See "Setting user preferences". If you want to run or debug sessions after creating or modifying the Debug connection point, you must restart Application Studio.

Installing Application Studio updates

This section describes how you can install updates for Application Studio, Project Server, and the corresponding plug-ins.

Note: Before you can install updates, the system administrator must provide a new Application Studio zip file. See "Preparing the installation" on page 26.

To install updates:

1 Open the Application Studio zip file

Open the zip file that contains the new software.

For Application Studio 10.3.1 these zip files can be available:

- InforApplicationStudio_10.3.1.nnn.zip (32-bits)
- InforApplicationStudio_10.3.1.nnn-x86_64.zip (64-bits)

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

Important: 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 Application Studio zip file

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

Note: If you upgrade from Application Studio 8.4.2 to Application Studio 10.3.1, you must perform the actions described in the "Upgrade notice" on page 9.

Administrative tasks

5

Defining a development environment

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

To define a development environment:

1 Start Application Studio

2 Open the Application Explorer view

On the **Windows** menu, select **Show View**, and subsequently click **Application Explorer**.

The Application Explorer view is displayed.

3 Enter the development environment properties

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

Save the environment and close the dialog. The new environment is displayed in the Application Explorer view.

4 Configure the Administrator connection point for the new environment

Complete these steps:

  1 Double-click the new environment. You are prompted to configure an Administrator *Connection Point*.

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

  3 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 one bshell is used to connect to the back-end, instead of two. This improves the performance.

  For details, see the wizard's online help.

Defining an application

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

Note: Before you can define an application, you must define one or more development environments and the corresponding Administrator Connection Points. See "Defining a development environment" on page 31.

Defining an LN application

To define an LN application:

1 Start Application Studio

2 Open the Application Explorer view

On the **Windows** menu, select **Show View**, and subsequently click **Application Explorer**.

The Application Explorer is displayed.

3 Enter the application properties

Complete these steps:

  1 In the view, right-click the development environment for which you want to define an application.

  2 On the shortcut menu, select **New Application**. The New Application dialog is displayed.

  3 Specify the application properties, such as application name, description, and release.

  For details, see the dialog's online help.

Note: You can now define one or more projects for the new application. See "Defining a software project" on page 65.

Code freeze procedure

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

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

Two types of code freeze are distinguished:

- Intermediate code freeze.

  An existing software project is linked to a new VRC, which is used as the export VRC for the application. See "Intermediate code freeze" on page 33.

- Final code freeze.

  A new application and project and a corresponding VRC are created. All activities are moved to the new project. See "Final code freeze" on page 35.

Intermediate code freeze

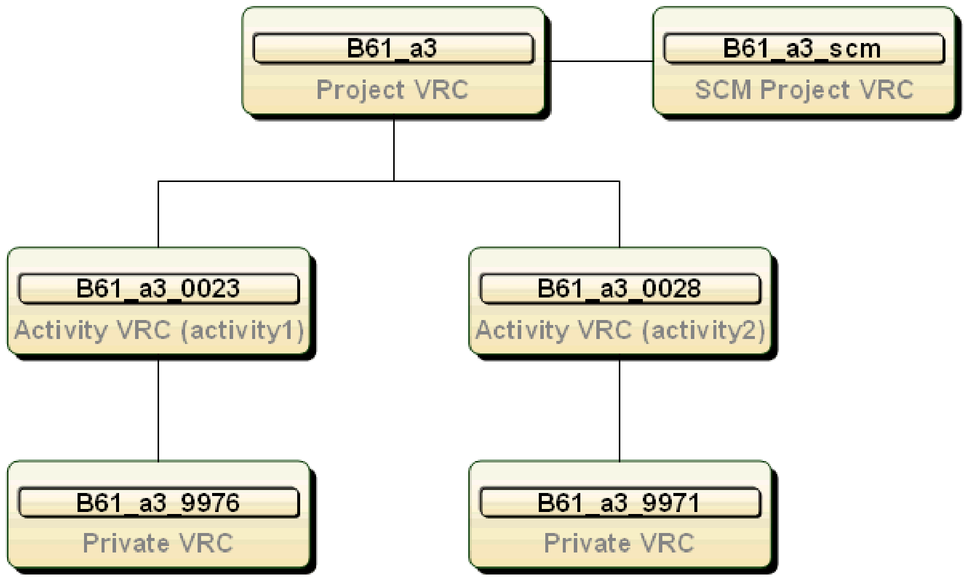
When you perform an intermediate code freeze, the current project is linked to a new project VRC, which is used as the export VRC for the application.

Note:

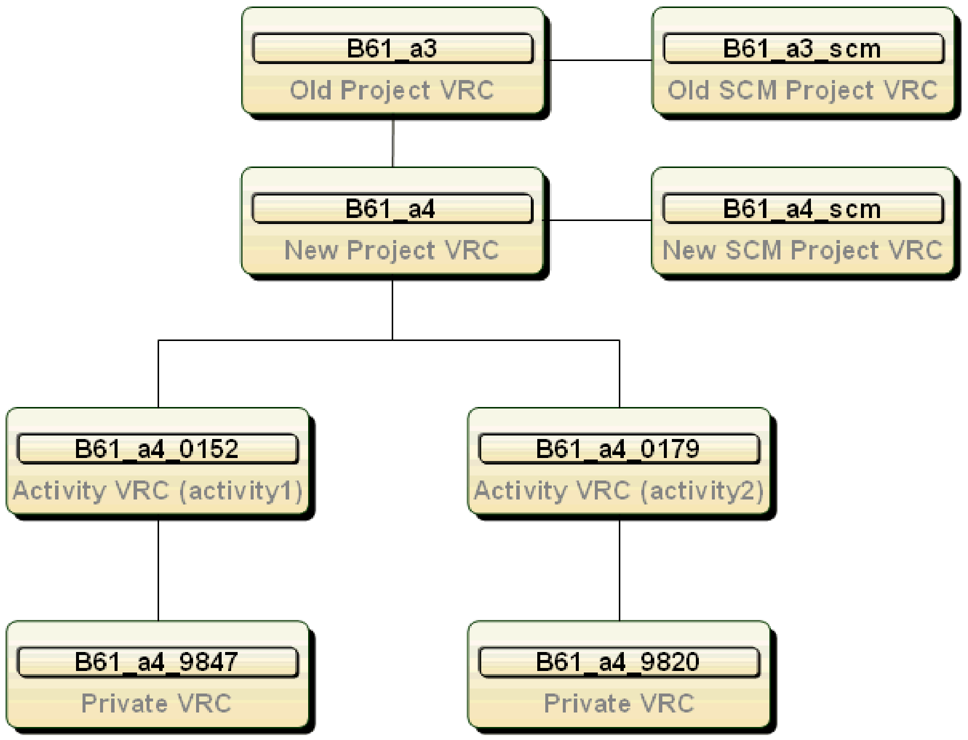
- The code freeze creates new activity VRCs derived from the new project VRC and, if SCM is active, new private VRCs derived from the new activity VRCs.
- All involved activities stay linked to the current project.
- The application and its *Base VRC* do not change.

Example

The following figure shows a VRC structure before an intermediate code freeze. SCM is active.



The following figure shows the VRC structure after an intermediate code freeze.



The new VRC structure contains:

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

Intermediate code freeze procedure

To perform an intermediate code freeze, complete the following steps:

1 Create new project VRC

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

2 Activate SCM for the new project VRC

If SCM is active for the old project VRC, activate SCM for the new project VRC. See "#unique_90".

3 Generate new VRCs for the involved activities

Complete the following steps:

1 Start the Code Freeze (ttadv1222m000) session.
2 Clear the **Final Code Freeze** check box.
3 In the **Source VRC** field, select the old project VRC.
4 In the **Target VRC** field, select the new project VRC.
5 Select the full range of activities.

6. Select the **Update Export VRC in Base VRC's** check box, so the new project VRC becomes the export VRC of the application's base VRC.

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

   Important

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

7. Click **Move**. The session performs the following actions:

   - Creates new activity VRCs, derived from the new project VRC.
   - Creates, if SCM is active, new private VRCs derived from the new activity VRCs.
   - Moves components from the old activity and private VRCs to these new VRCs.
   - Removes the old activity and private VRCs.

4. Inform Application Studio users

   Inform the Application Studio users the code freeze is finished.

   The users must recover their workspace.

   - **Important**

     This recovery must be done after the new project VRC is assigned as the export VRC for the application's base VRC. See step 3 in this procedure.
   - The users must select the **Recover existing activity files** check box in the Recover activity dialog. See "Recovering an activity".

Final code freeze

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

Note:

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

Final code freeze procedure

To perform a final code freeze, complete the following steps:

1. Create new project VRC

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

2. Activate SCM for the new project VRC

   If SCM is active for the old project VRC, activate SCM for the new project VRC. See "#unique_90".

3 Define new base VRC

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

4 Define new application

Start Application Studio and define a new application.

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

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

See "Defining an application" on page 32.

5 Create new project

Start Application Studio and define a new software project.

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

See **To define a project** in the Project Server help.

6 Move activities to the new project

Complete the following steps:

1. Log on to the project server and start the Final Code Freeze (tlprj1220m000) session.
2. In the **Source Project** field, select the old project.
3. In the **Target Project** field, select the new project.
4. Select the full range of activities.
5. Click **Move**. The session moves the activities to the new project.

7 Generate new VRCs for the involved activities

Complete the following steps:

1. Start the Code Freeze (ttadv1222m000) session.
2. Select the **Final Code Freeze** check box.
3. In the **Source VRC** field, select the old project VRC.
4. In the **Target VRC** field, select the new project VRC.
5. Select the full range of activities.
6. Click **Move**. The session performs the following actions:
   - Creates new activity VRCs derived from the new project VRC.
   - Creates, if SCM is active, new private VRCs derived from the new activity VRCs.
   - Moves components from the old activity and private VRCs to these new VRCs.
   - Removes the old activity and private VRCs.

8 Inform Application Studio users

Inform the Application Studio users the code freeze is finished.

For each activity involved, users must complete the following steps:

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

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

3 Recover the activity.

Important

- The users must select the **Recover existing activity files** check box in the Recover activity dialog. See "Recovering an activity".

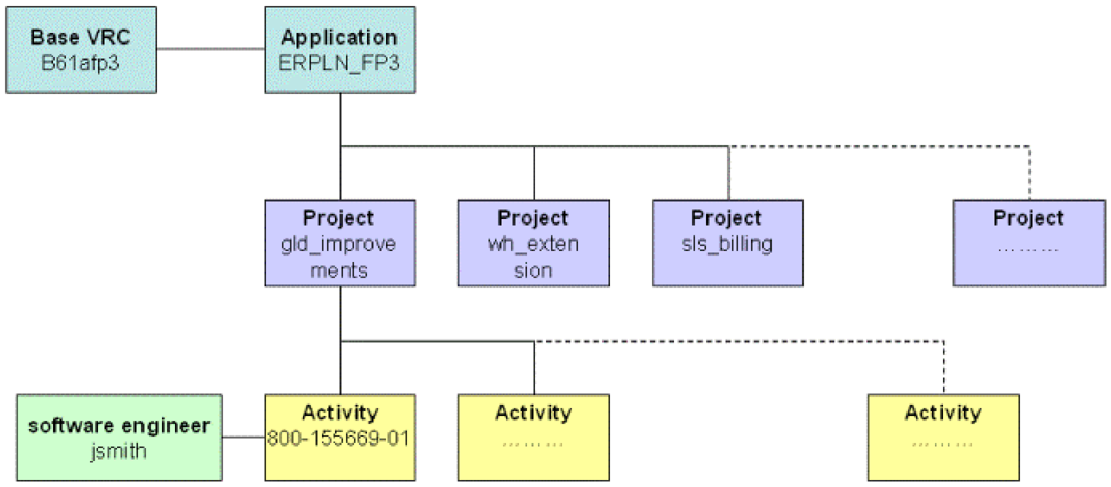
Activity based development

6

Overview

The software development process in Application 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" on page 61. See "Defining an application" on page 32.

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 view" on page 73. 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 **To define a project** in the Project Server documentation.

Software engineers can open their activities through the Activity Explorer view. See "Developing software components".

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.

Configuration Management

The configuration management in Application 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 platform. 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, refer to *To use the Software Configuration Management system (SCM)* in the LN Web Help.

Note:

Because of SCM limitations, the Application 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 Application Studio application(s).

The following situations can be distinguished:

Scenario	Description
SCM for project VRCs and applica-tions	SCM is activated for the project VRC(s) on the LN server and for the applications in Application Studio. This is the preferred scenario if your LN back-end supports SCM. Each time you check in a component, Application Studio generates a new revi-sion. Other software engineers can use (not modify) your checked in components. See "Development with SCM for project VRCs and applications" on page 42.
No SCM	SCM is disabled for the project VRC(s) on the LN server and applications in Application Studio. Application 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*, take the following steps:

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 displays 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 Application Studio. Open the Application Studio 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 Application Studio is open, you can link software components from the LN dictionary to the project. You can also create new components via the Application Studio.

You can link or create the following 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.

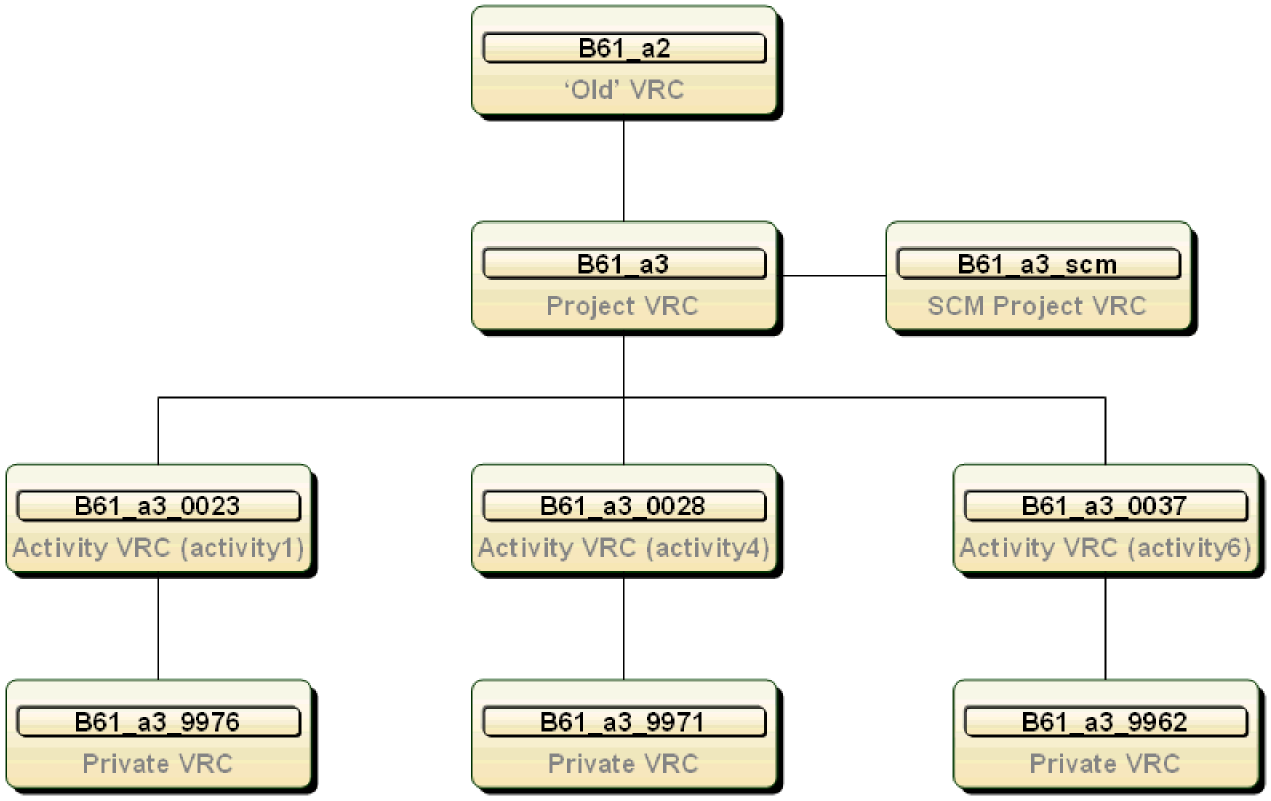
The Application Studio shows a tree of software components on the LN server and a tree of components in your current activity. For details on the layout of the Application Studio, see the "Application Studio workbench" on page 17 section.

Development with SCM for project VRCs and applications

VRC structure

When SCM is enabled for project VRCs and applications, Application 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.

See the following figure for an example:



The following 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. For details, refer to "Activity context".
- 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

Application Studio generates the following 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, Application 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. Application Studio checks whether the component is already present in your activity VRC (for example because of an earlier check out and check in action).

- Application Studio does not support parallel development for applications. Therefore, if the component is not present in your activity VRC, Application 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*, Application 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 Application Studio:

1. Right-click the component in the Activity Explorer view.
2. On the shortcut menu, select **Team**, and then click **Checkout**.

Check in

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

Application 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 all software engineers, end your activity, or cancel the activity or the component.
- 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. For details, refer to "Activity context".

To check in a software component in Application Studio:

1. Right-click the component in the Activity Explorer view.
2. On the shortcut menu, select **Team**, then click **Checkin**. The Check In comment dialog 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 back-end. 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 in one go, you can use the same revision text for all components. See "Check In comment".

Undo checkout

You can undo the check out of a component. Application 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 all software engineers. To unlock the component in the project VRC and to make it accessible for all software engineers, end your activity, or cancel the activity or the component.

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

1. Right-click the component in the Activity Explorer view.
2. On the shortcut menu, select **Team**, then click **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, Application Studio performs the following 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 all 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".

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

To end an activity in Application Studio:

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

2 On the shortcut menu, select **Team**, then click **End activity**.

Cancel Activity

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

Application Studio performs the following 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 Application Studio:

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

2 On the shortcut menu, select **Team**, then click **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 back-end.

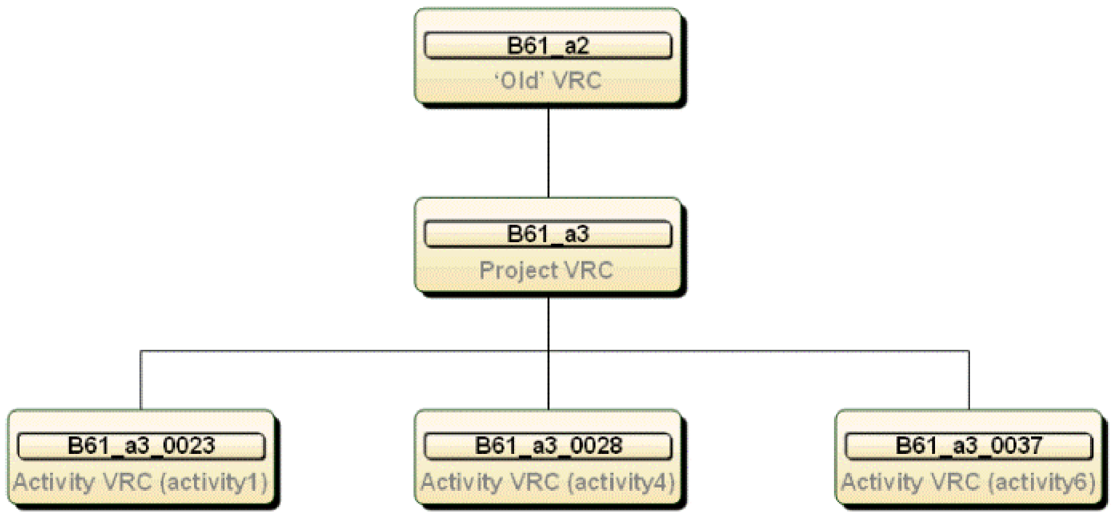
Note:
This session does not update the status of the canceled activities in the Project Server. See the session help.

Development without SCM

VRC structure

When SCM is disabled for an application, Application 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.

See the following figure for an example:



The following 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".

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

Application Studio generates the following 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. Application 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, Application 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 Application Studio:

1. Right-click the component in the Activity Explorer view.
2. On the shortcut menu, select **Team**, then click **Checkout**.

Check in

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

Application 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 all software engineers, end your activity, or cancel the activity or the component.

To check in a software component in Application Studio:

1. Right-click the component in the Activity Explorer view.
2. On the shortcut menu, select **Team**, then click **Checkin**.

Undo checkout

You can undo the check out of a component. Application 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 all 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 Application Studio:

1. Right-click the component in the Activity Explorer view.
2. On the shortcut menu, select **Team**, then click **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, Application Studio performs the following actions:

- Moves all software components in the activity VRC to the project VRC. Therefore, components become available to all 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".

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

To end an activity in Application Studio:

1. Right-click the activity in the Activity Explorer view.
2. On the shortcut menu, select **Team**, then click **End activity**.

Cancel Activity

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

Application Studio performs the following 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 Application Studio:

1. Right-click the activity in the Activity Explorer view.
2. On the shortcut menu, select **Team**, then click **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 back-end.

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

Application Studio Limitations

7

This section describes:

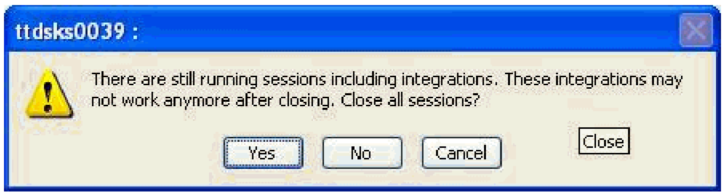
- General limitations
- Script Editor Limitations
- Configuration Management (CM) limitations
- Build limitations
- Debugger limitations
- Web UI limitations

General limitations

Connections

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

When you close the Worktop, the following message is displayed:


ttdsks0039 :

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 Application Studio, running on the LN server. If you click **Yes**, your connections will be closed. As a result, Application 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

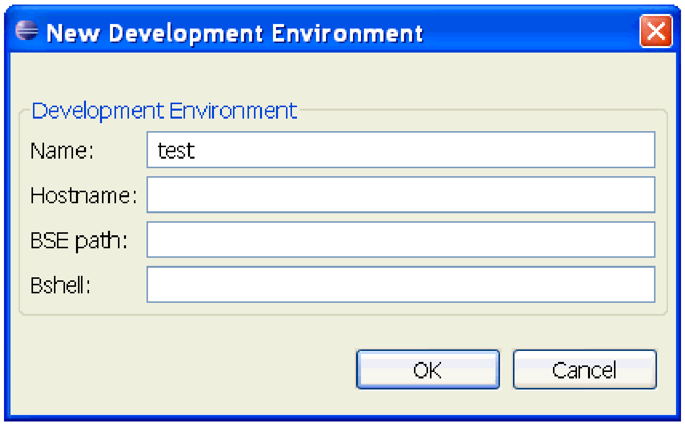
Labels

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.

8 Dialogs

New Development Environment

Use this dialog to define a new development environment.



Name:	test
Hostname:	
BSE path:	
Bshell:	

Name
The name of the development environment.

Hostname
The TCP/IP hostname or IP address of the LN back-end.

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

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

Create a new Application

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

New Application window

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

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

To create a new Application 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 22.
Specifying a release is mandatory.

Note: An Application 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 Application Studio. Therefore, you are recommended to leave this field empty.

InContext Reference Model
The application's InContext Reference Model.

Related topics

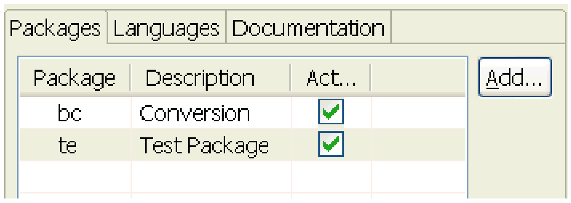
- "InContext Modeling" in the *Infor LN Application Studio User Guide (U8921)*
- *Infor Enterprise Server InContext Modeling Development Guide (U9770)*

Use SCM
If this check box is selected, the software components developed or modified in Application 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 back end. 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	Description	Act...
bc	Conversion	✓
te	Test Package	✓

Package
A package linked to the application.

In Application 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. Application 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 or not.

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

1. Click **Add**. An empty package record is added to the **Packages** grid.
2. 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 entered in the grid.
3. To indicate whether the package is used in the application, select or clear the **Active** check box.

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.

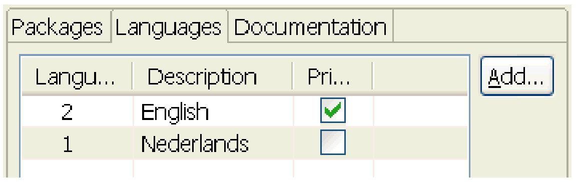
Active
If this check box is selected, the package of the application will be changed or modified in Application Studio.

Default value

The check box is cleared.

Languages

Use this tab to link software languages to the application.



Langu...	Description	Pri...
2	English	✓
1	Nederlands	☐

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 Application 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 Application 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* will be 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 Application 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.

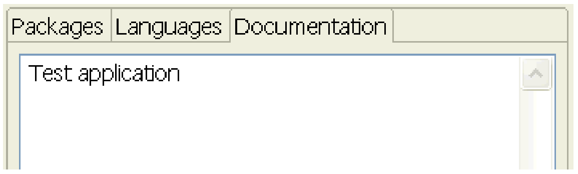
Note:

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



Test application

Documentation
Use this field to enter additional information.

Views

9

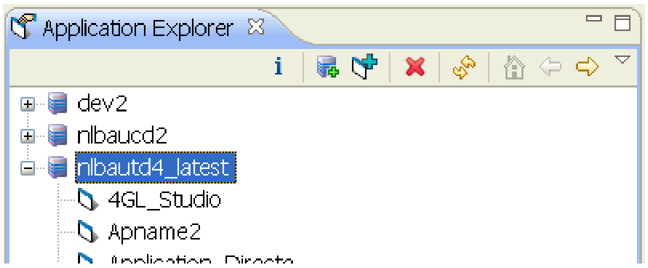
Application Explorer view

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

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

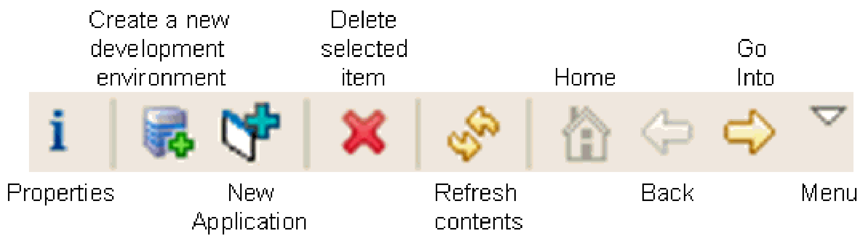
A development environment is a BSE environment on an Infor LN server, in which the software engineers develop software components.

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



Toolbar

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



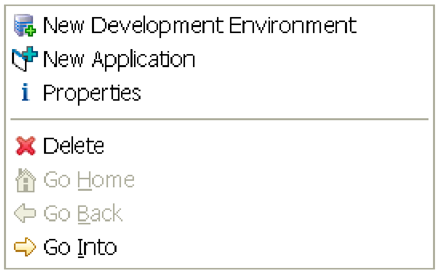
Properties	Starts a dialog where you can view or modify the properties of the selected item.
Create a new development environment	Starts the "New Development Environment" on page 55 dialog. Here, you must enter the properties for a new development environment.
New Application	Starts the "New Application" on page 55 dialog. Here, you must enter the name, release, and other properties for a new application.
Menu	Contains the **New Application** command.

Note:

The remaining buttons are standard Eclipse commands. For details on these buttons, refer to *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 Application Studio-specific commands:

New Development Environment	Starts the "New Development Environment" on page 55 dialog. Here, you must enter the properties for a new development environment.
New Application	Starts the "New Application" on page 55 dialog. Here, you must enter the name, release, and other properties for a new application.
Properties	Starts a dialog where you can view or modify the properties of the selected item.
Delete	Removes the selected item.

For details on the remaining commands, refer to *User interface information* in the *Workbench User Guide*.

Icons

The following icons are displayed in the Application Explorer view:

Icon	Description
🛢	Development Environment
◇	Application

Infor LN Project Server

10

Infor LN Project Server introduction

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

Project Server is implemented in the *Eclipse* framework. Eclipse is a Java based development environment with many plug-ins available. Eclipse is used as a framework for Project Server, and various other Infor solutions such as Application Studio and Business Studio, an integration environment.

For more information refer to "Defining a software project" on page 65.

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 Application 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. Enter the project data

   Complete the following steps:

   1. On the view's toolbar, or on the shortcut menu, select **New Software Project**. The Create a Software Project dialog is displayed.
   2. Specify the project properties. See the online help of the dialog.
   3. Save the project and close the dialog.

4. Add one or more activities to the new project

   Complete the following steps:

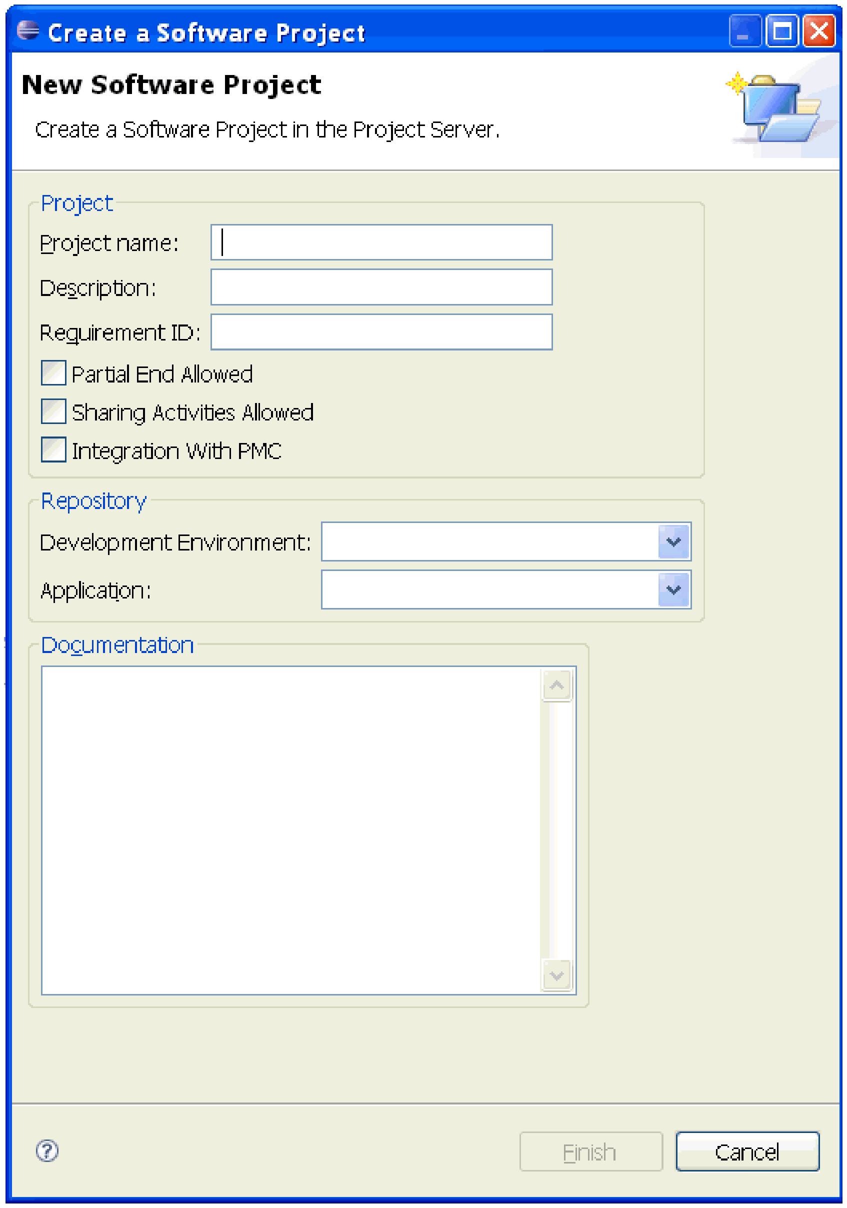
   1. In the Software Project Explorer view, right-click the new project and, on the shortcut menu, select **New Activity**. The Create a new Activity dialog is displayed.
   2. Specify the activity properties. See the online help of the dialog.
   3. Save the activity and close the dialog.

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 Studio perspective in the Application Studio workbench.

Dialogs

Create a Software Project

Use this dialog window to create a new *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".

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

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

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

Note:

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

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

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

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

Repository

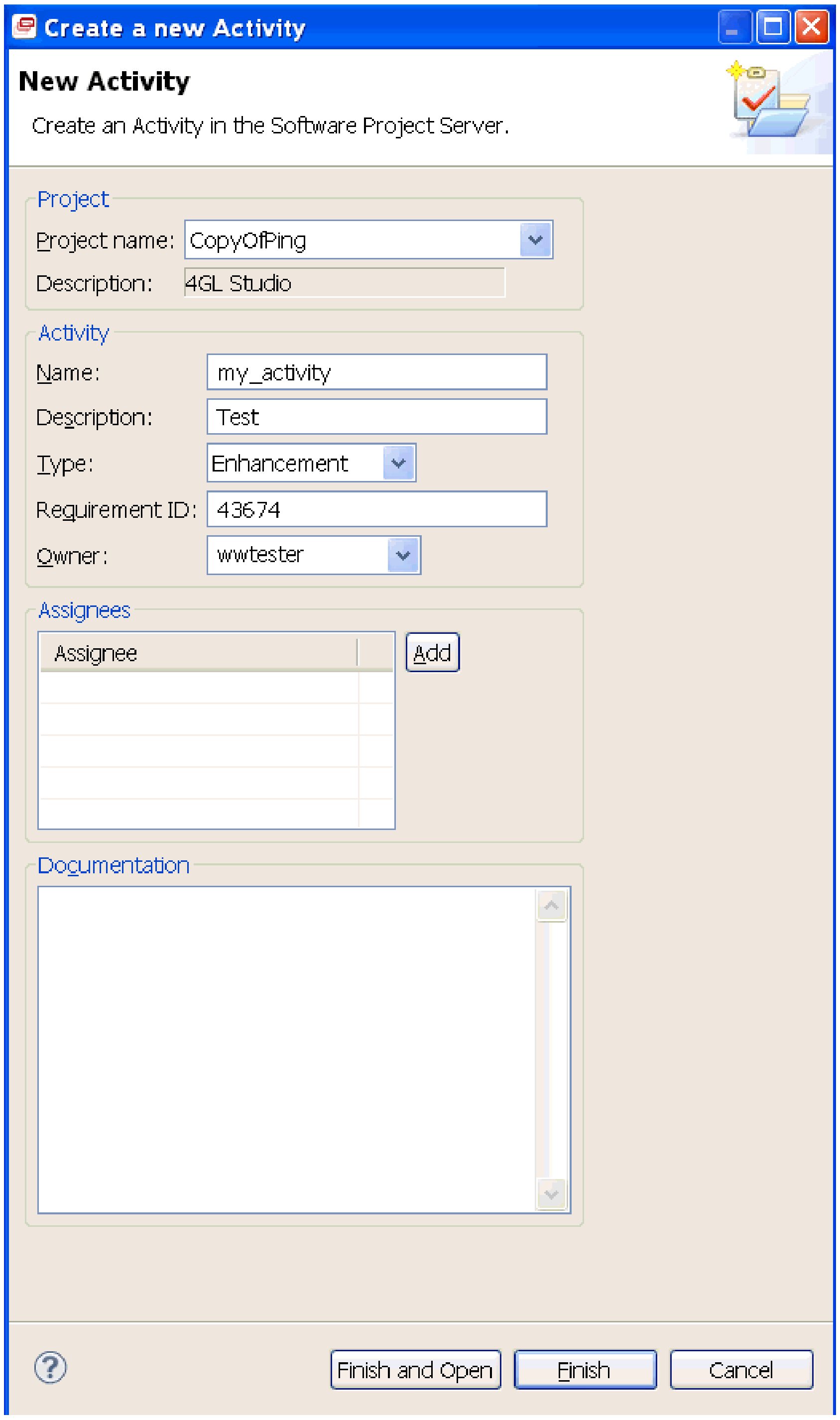
Development Environment
The LN environment in which the software for the project will be developed. Select the environment from a drop-down list.

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 window to create a new *activity*.


Create a new Activity

New Activity

Create an Activity in the Software Project Server.

Project

Project name: CopyOfPing

Description: 4GL Studio

Activity

Name: my_activity

Description: Test

Type: Enhancement

Requirement ID: 43674

Owner: wwtester

Assignees

Assignee

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.

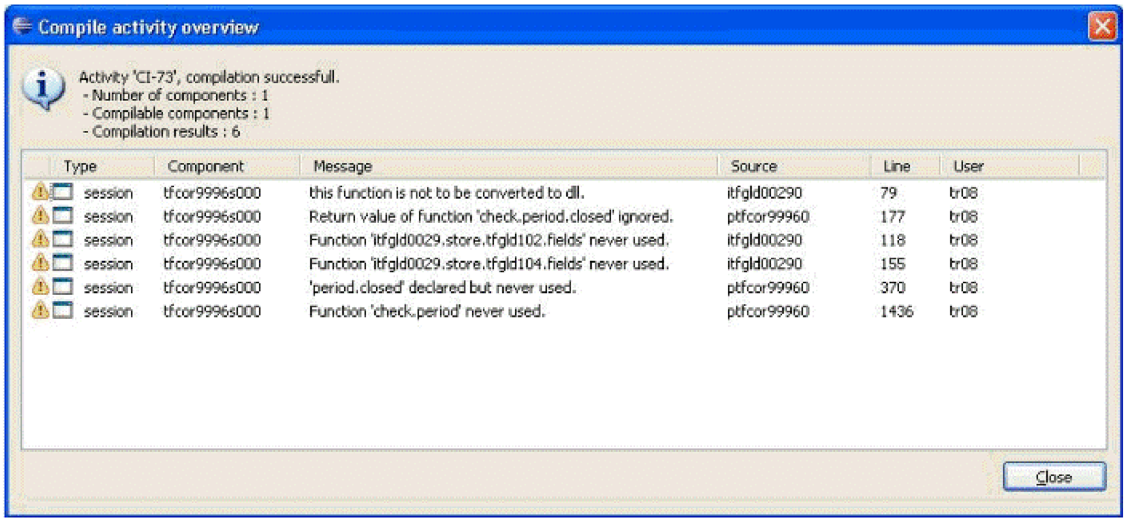
Documentation
Enter documentation about the activity.

Finish
Saves the new activity and closes the dialog.

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

Compile activity overview

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

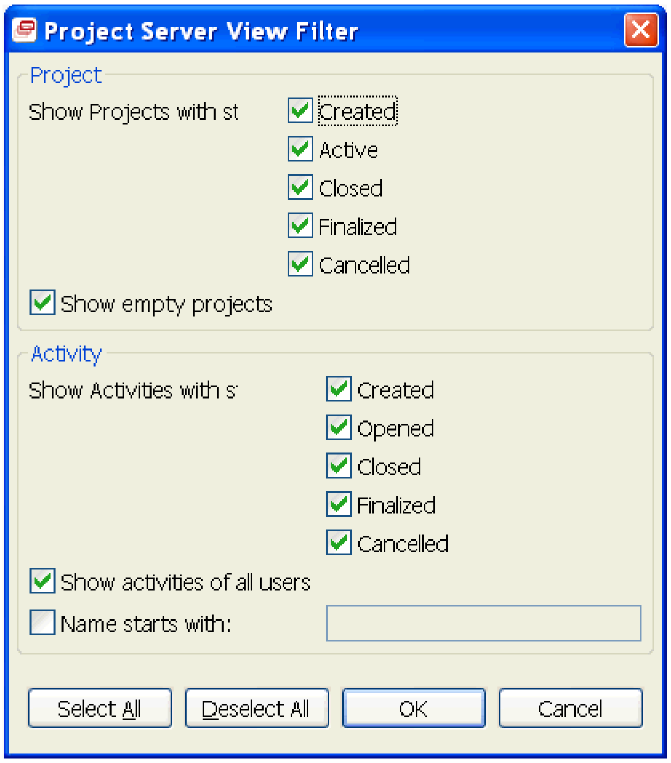


Activity 'CI-73', compilation successfull.
- Number of components : 1
- Compilable components : 1
- Compilation results : 6

Type	Component	Message	Source	Line	User
session	tfcor9996s000	this function is not to be converted to dll.	itfgld00290	79	tr08
session	tfcor9996s000	Return value of function 'check.period.closed' ignored.	ptfcor99960	177	tr08
session	tfcor9996s000	Function 'itfgld0029.store.tfgld102.fields' never used.	itfgld00290	118	tr08
session	tfcor9996s000	Function 'itfgld0029.store.tfgld104.fields' never used.	itfgld00290	155	tr08
session	tfcor9996s000	'period.closed' declared but never used.	ptfcor99960	370	tr08
session	tfcor9996s000	Function 'check.period' never used.	ptfcor99960	1436	tr08

Project Server View Filter

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



Show Projects with st
- [✓] Created
- [✓] Active
- [✓] Closed
- [✓] Finalized
- [✓] Cancelled

[✓] Show empty projects

Activity

Show Activities with s
- [✓] Created
- [✓] Opened
- [✓] Closed
- [✓] Finalized
- [✓] Cancelled

[✓] Show activities of all users

[ ] Name starts with:

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

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

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

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

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

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

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

Note: The filter is case sensitive.

Perspectives

Project Server perspective

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

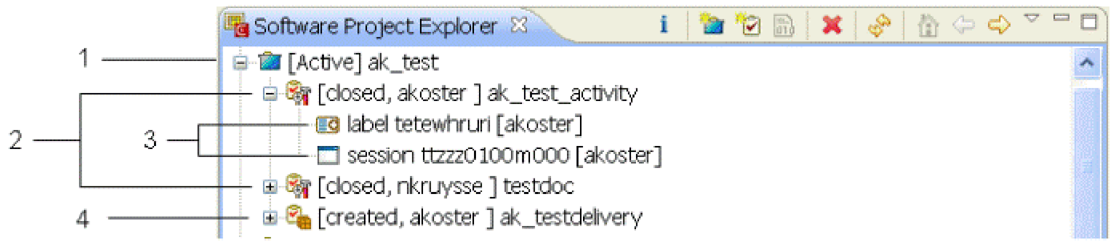
This table shows the views in the perspective:

"Software Project Explorer view" on page 73	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 65.
Application Explorer	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 "To define an application" in the Application Studio documentation.

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.


Software Project Explorer
1
2
3
4
[Active] ak_test
[closed, akoster] ak_test_activity
label tetewhruri [akoster]
session ttzzz0100m000 [akoster]
[closed, nkruysse] testdoc
[created, akoster] ak_testdelivery

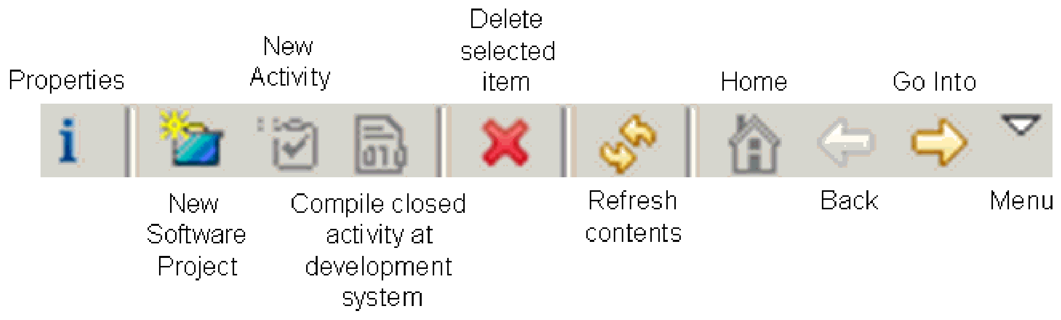
This table describes the different nodes in the previous figure:

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:



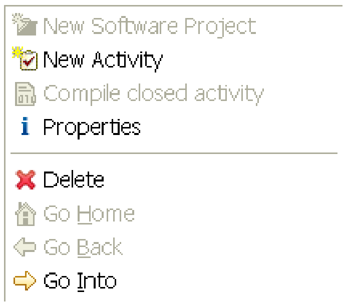
Properties	Displays the properties of the selected project or activity. For details on the project and activity properties, see the description of the Create a Software Project and Create a new Activity dialogs.
New Software Project	Adds a project on the project server.
New Activity	Adds a new activity 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	Provides the following menu items: • New Software Project • New Activity • Compile closed activity • Filters This command starts a dialog where you can specify filter criteria for the Software Project Explorer view. See "Project Server View Filter" on page 71.

Note: The remaining buttons are standard Eclipse commands. For details on these buttons, refer to *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 most of the toolbar commands.

Icons

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

Icon	Description
📁	Project
⚙	Development Activity
📦	Delivery Activity
➕	Additional File
@	Domain
●	Function
🏷	Label
📚	Library
📋	Menu
💬	Message
?	Question
📜	Report
▭	Session
▦	Table

Glossary

3GL script

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

4GL engine

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

Synonym: standard program

4GL script

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

Activity

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

You can define the following types of activities:

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 the following 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 55 for which the *Software Projects* are defined. For more information on the administrator's tasks, refer to "Defining an application" on page 32.

Application

A software application is a consistent set of packages that functionally belong together, and developed in the *Application 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.

Application Studio

A platform for activity based development of LN software. Application Studio is implemented in the Eclipse framework and makes use of the *Software Configuration Management* functionality on the LN server. The Application 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.

Application Studio

A platform for activity-based development of LN software. Application Studio is implemented in the Eclipse framework and uses the Software Configuration Management functionality on the LN server. The Application 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.

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.

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. 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 Application Studio Script Editor.

See "Code Assist".

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 back-end.

Application Studio uses the following connection points:

- ProjectServer
- Administrator
- Debug
- Development Address
- Runtime Address

Connection Point

A logical endpoint in the JCA connectivity architecture. A connection point 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.

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.

For this, a number of 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.

Three 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 defines the following:

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

At runtime, the dynamic form displays 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 framework for various Infor solutions, such as Application Studio, Project Server and Business Studio.

Eclipse

Eclipse is an open platform for tool integration built by an open community of tool providers, which provides tool developers with ultimate flexibility and control over their software technology. The Eclipse platform operates under an open source paradigm, with a common public license that provides royalty-free source code and world wide redistribution rights. Eclipse is used as a framework for various Infor solutions, such as Infor LN Application Studio, Infor LN Project Server, and Infor LN Business 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 Application Studio script editor. For more information, refer to "Tasks view", "Source Tab", and "ToDo Comments".

Editor

An editor is a visual component within the Application 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 Application 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 Application 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.

The language code corresponds to the language number by the following convention:

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.

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 the following:

- Package code, such as 'tc'
- A version (VRC) code, such as *B61O a cus1*, built up of the following:
  - 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 Studio 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.

The following perspectives are delivered with Application Studio:

- Application Studio
- Debug
- Project Server

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. As you work in the workbench, you will probably switch perspectives frequently.

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 *LN Programmer's Guide*. Access this guide through the Application 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 65.

Project Server

The Project Server contains the *Software Projects* for *Application 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.

Some examples of rules are as follows:

- 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 the following:

- 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 the following 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 Application Studio. Before editing a component, engineers must first open an *Activity* the *Project Manager* assigned to them.

Software Project

A software project for *Application 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 the following 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 backend.

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 Mem-ber State	5	Enumerat-ed
...			

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 the following:

- **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 Application 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.

Examples of typical Application Studio views are as follows:

- 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.