Infor LN Development Tools Development Guide

Copyright © 2013 Infor

Important Notices

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

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

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

Although Infor has taken due care to ensure that the material included in this publication is accurate and complete, Infor cannot warrant that the information contained in this publication is complete, does not contain typographical or other errors, or will meet your specific requirements. As such, Infor does not assume and hereby disclaims all liability, consequential or otherwise, for any loss or damage to any person or entity which is caused by or relates to errors or omissions in this publication (including any supplementary information), whether such errors or omissions result from negligence, accident or any other cause.

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

Trademark Acknowledgements

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

Publication Information

Document code	U8883M US
Release	10.3
Publication date	December 23, 2013

Table of Contents

About this document

Chapter 1 Development Tools Introduction....13

Development Tools introduction....13

Chapter 2 Overview of Software Components....15

Overview of software components....15

4GL Engine....15

Charts....16

DAL....17

DAL 1....18

DAL 2....19

Field dependencies....20

Business methods....20

Dashboards....21

To create dashboards....23

Domains....23

To create domains....25

Forms....25

Functions....28

To create functions....29

Labels....29

To create labels....30

Libraries....31

To create libraries....31

User Exit DLLs....31

Menus....32

To create menus....33

Messages....33

To create messages....34

MMT Sessions....34
To create MMT sessions....36
Questions....37
To create questions....38
Reports....38
Report structure....39
To create reports....40
Report scripts....41
Report script structure....41
To create report scripts....42
Sessions....42
SQL queries....46
4GL program queries....47
Stand-alone queries....47
Table definitions....48
Fields, domains and indices....49
Related Tables and References....49
To create table definitions....49
UI scripts....50
Event sections....50
To create UI scripts....51

Chapter 3 Data Dictionaries....53
Data dictionaries....53
The Runtime Data Dictionary....53
The Application Data Dictionary....54
Convert to Runtime Data Dictionary....54

Chapter 4 Development Parameters and Authorizations....57
Development parameters and authorizations....57
Development Parameters....57
Developer Authorizations....57

Chapter 5 Software Development Procedures....61
Development process....61
Design Data Model....62
Implement Data Model....63
Create User Applications....63
Create Business Logic....64
Create Menus....64
Document....65
Deploy Application....65
Identify Changes....65
To develop software components....65
To create a data model....66
Procedure aim....67
Procedure result and prerequisites....67
Procedure summary....67
Procedure details....68
To reconfigure tables....71
Procedure aim....71
Procedure result and prerequisites....71
Procedure summary....72
Procedure details....72
To create sessions....73
Procedure aim....73
Procedure result and prerequisites....73
Procedure summary....74
Procedure details....74
To create forms....75
Procedure aim....75
Procedure result and prerequisites....76
Procedure summary....76
Procedure details....76

To edit forms....78
To edit the form content....78
To edit the form structure....81
To edit the form sequence....84
To create commands and buttons for a session....85
To specify indices for a session....87
To create a form for a print session....88
Review Program Script....89
Edit the Form....90
To create reports....96
Procedure aim....96
Procedure result and prerequisites....96
Procedure summary....97
Procedure details....97
To edit reports....99
To edit report input fields....99
To edit report layouts....100
To edit a report script....104
To create dashboards....104
To create the session....105
To create the program script....111
To create the details sessions....115
To create MMT Sessions....117
To create a main entity detail session....118
To create a multi-main table controller session....119
To create a satellite session....120
To assign satellites to the controller....121
To compile and run the controller session....121
To integrate the controller session....122
To translate software components....122
Multi-language....122

To translate labels, questions and messages....123

Chapter 6 Session Personalizations....125

Session Personalizations....125

Chapter 7 Version and Release Management....127

Version and release management introduction....127

An overview of version and release management....127

Package combinations....127

Package VRCs....128

Package VRC code....129

LN software environment....130

VRC derivation....133

Contents of a software environment....136

Copy software components concept....136

Export/import software components concept....136

To use the Software Configuration Management system (SCM)....138

SCM Groups....139

To set up the SCM environment....140

Sessions....140

To use the SCM check-in and check-out procedure....140

Procedure....141

Standard software environment procedure....141

Procedure aim....141

Procedure result and prerequisites....142

Procedure summary....142

One-step software environment procedure....143

Procedure aim....143

Procedure result and prerequisites....143

Procedure summary....144

Copy software components procedure....144

Procedure aim....144

Procedure result and prerequisites....144
Procedure summary....144
Export and import procedure....145
Procedure aim....145
Procedure result and prerequisites....145
Procedure summary....145
Purge a package VRC derivation structure procedure....146
Procedure aim....146
Procedure result and prerequisites....147
Procedure summary....147

Chapter 8 LN Software Maintenance....149
PMC introduction....149
PMC Benefits....149
PMC overview....150
PMC Architecture....150
Where to find the PMC module....151
Dependencies....151
Individual solutions....152
Collections....153
Feature Packs and patches....153
PMC Distributor functionality....159
PMC Recipient functionality....160
PMC distributor procedure....161
Setup....161
To create updates....163
Feature Pack development....170
Customizations....175
PMC distributor session summary....181
PMC distributor sessions....181

Chapter 9 LN Homepages....185

Homepages introduction....185
Introduction....185
Homepage structure....186
To create LN homepages....188
Prerequisites....188
To create homepages....188
To create functions for LN homepages....192
Introduction....192
Homepage function types....192
To translate resource files....199
To translate resource files....200

Chapter 10 LN Reporting Overview....203
LN Reporting Overview....203
4GL reports....203
Report Viewer enabled reports....204
To create a report project....207
To use the Report Viewer....208
Toolbar buttons....208

Chapter 11 4GL Reports....211
4GL Reports Overview....211

Chapter 12 Session-based Reporting....213
Session-based Reporting....213
To create session-based reports....213
To create a Report Viewer device for session-based reports....217
To run session-based reports....218

Chapter 13 Query-based Reporting....219
Query-based Reporting....219
To create query-based reports....222
To create resource files....228

To link a resource file to a report design....230
To create report libraries....230
To link a report library to a report design....231
To create a print session for query-based reports....232
To run query-based reports....234
To translate resource files....235
To translate resource files....235

Appendix A Glossary....237

Index

About this document

Document summary

Infor LN is a business software solution that consists of applications, tools, and an Enterprise Modeler, all of which work together as a fully integrated system to support all aspects of a business. This document is a Developer's Guide that describes how developers must use Infor LN Development Tools to develop new LN applications, or to customize existing software components.

Development Tools is used, among other things, to create package VRCs, sessions, forms, reports and labels, and to edit scripts and libraries. The Development Tools functionality is located in the Tools (tt) package.

This document is divided into the following chapters:

Chapter **Development Tools Introduction** outlines the functionality of the Infor LN Development Tools.

Chapter **Overview of Software Components** briefly describes the various types of LN software components.

Chapter **Data Dictionaries** describes the purpose of the Runtime Data Dictionary, the purpose of the Application Data Dictionary, and how you convert data from the Application Data Dictionary to the Runtime Data Dictionary.

Chapter **Development Parameters and Authorizations** describes the default settings and parameters, and the developer authorizations, that are required to develop software components.

Chapter **Software Development Procedures** describes the procedures to create various types of software components, such as tables, domains, sessions, forms, and reports.

Chapter **Session Personalizations** describes how end users can customize the user interface of LN sessions via the Worktop and Web UI clients, so that the sessions meet their personal demands.

Chapter **Version and Release Management** describes the version and release management concept and the corresponding procedures, for example: the procedure to set up a software environment, and procedures to transfer software components from a development environment to a test environment, or to an operational (live) environment.

Chapter **LN Software Maintenance** describes the Product Maintenance and Control (PMC) concept and the procedures for the PMC distributor.

Chapter **LN Homepages** describes how you can develop Infor LN homepages. Homepages are used as a start point for navigation for particular roles, such as Warehouse Manager or Warehouse Administrator.

Chapter **LN Reporting Overview** briefly describes the report types supported in LN.

Chapter **4GL Reports** describes the structure of the LN 4GL reports and report scripts.

Chapter **Session-based Reporting** describes how to generate Report Viewer enabled reports from existing 4GL reports and how to print these reports.

Chapter **Query-based Reporting** describes how to create and run query-based reports that read data from the LN database.

The **Glossary** defines the terms and acronyms used in this document.

Note:

- A number of figures of Infor LN sessions in this document are from previous Infor LN releases and can differ slightly in appearance to your Infor LN sessions. However, the described functionality is identical.
- This guide contains references to the *LN Programmer's Guide*. The Programmer's Guide is available through solution 22924522.

Comments?

We continually review and improve our documentation. Any remarks/requests for information concerning this document or topic are appreciated. Please e-mail your comments to documentation@infor.com .

In your e-mail, refer to the document number and title. More specific information will enable us to process feedback efficiently.

Contacting Infor

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

If we update this 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 updated documentation.

Chapter 1
Development Tools Introduction

1

Development Tools introduction

Infor LN Development Tools provides tools for developers.

You can use Development Tools to:

- Create Package VRCs in which you can develop software components.
- Set up a data model, consisting of domains and tables, for an application.
- Create various types of software components, such as:
  - sessions
  - forms
  - reports
  - multi-language data field labels
  - questions and messages.
- Create, edit and compile scripts and libraries, e.g. UI scripts, DLLs and DALs.
- And so on.

The Development Tools functionality is located in the Tools (tt) package on the LN server.

For information on the software components that you can develop, and the corresponding procedures, refer to the following sections:

- *Overview of software components (p. 15)*
- *Development process (p. 61)*

Chapter 2
Overview of Software Components

2

Overview of software components

The following sections briefly describe various types of Infor LN software components:

- *4GL Engine (p. 15)*
- *Charts (p. 16)*
- *DAL (p. 17)*
- *Dashboards (p. 21)*
- *Domains (p. 23)*
- *Forms (p. 25)*
- *Functions (p. 28)*
- *Labels (p. 29)*
- *Libraries (p. 31)*
- *Menus (p. 32)*
- *Messages (p. 33)*
- *MMT Sessions (p. 34)*
- *Questions (p. 37)*
- *Report scripts (p. 41)*
- *Reports (p. 38)*
- *Sessions (p. 42)*
- *SQL queries (p. 46)*
- *Table definitions (p. 48)*
- *UI scripts (p. 50)*

4GL Engine

The 4GL engine handles the default behavior of a session. If additional functionality is needed or default functionality needs to be by-passed, this should be programmed in a UI script (Program script).

When a session is started, the session's object will run. The object and the form work together to perform queries and present data on the form.

The sequence of actions that take place are:

1. The object allocates memory space for a record buffer based on the table definition.
2. The object builds a SQL statement to select rows from the main table and related tables. The record buffer is populated within a selectdo loop.
3. The record buffer variables are updated after the query and are available for use on forms.
4. The form fields are mapped to the record buffer and display on the form.

This sequence of actions is an overview of the actions that take place within a session. The detailed actions are coded. The 4GL Engine (also called the standard program) provides this standardized coding and enforces consistency across sessions in the system.

The 4GL Engine controls the processing of a session:

- The session startup including checking for authorizations, loading icons and pull-down menus.
- Loading the forms, reports, and charts.
- Performing queries on the main table for the user's company, which includes selects, inserts, updates, and deletes.
- Scrolling through data selections.
- Checking that a reference to another table exists, such as checking that a Department exists for an Employee.
- Handling the right forms, labels, and menus for a user's language.
- Choose the device for output.

Note

The exceptions to the normal processing are contained in the Session Object. The session object uses events to hook into the 4GL Engine, and describe the actions that should happen as a result of an event. These additional actions are defined in the UI script (program script). For more information, refer to the *UI scripts (p. 50)* section.

Charts

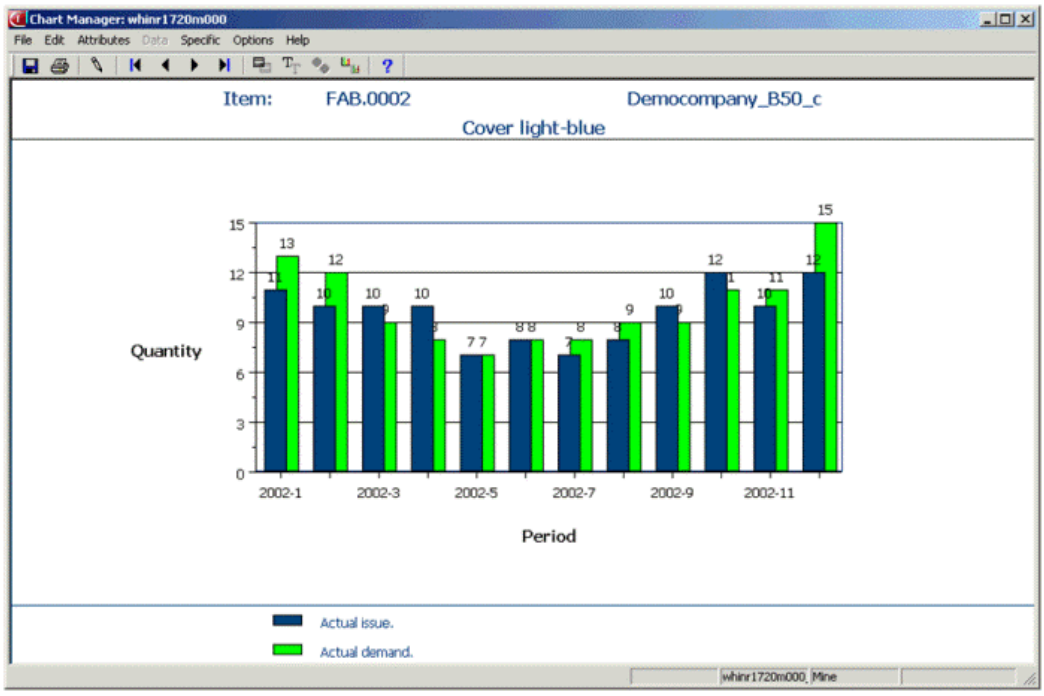
Charts present data in a graphical format to the user. Charts are used in sessions and SQL queries. A session can contain multiple charts. An SQL query can have only one chart.

The following default chart types are present in Infor LN:

- Bar
- Layer
- Line
- Pie
- Scatter

- Stacked bar

See the following figure for an example:



Sample chart

For more information on charts, refer to the Infor Web Help.

DAL

The Data Access Layer (DAL) allows developers to describe rules about data. A DAL is linked to a table, not a session. Therefore, when the table is accessed, the DAL is used. In this way, different sessions can update a table by using the same rules. In addition, integration capabilities use the DAL to ensure updates are also processed with the same rules.

Data Access Layer (DAL) scripts implement the business rules of the application to ensure the logical integrity of the database. DAL scripts are a Library software component that is aligned with a specific table. These scripts are compiled into objects. When a session performs an action on a table, the DAL for the table is loaded.

In Infor LN, two types of Data Access Layer scripts exist: DAL 1 and DAL 2.

Business Rules

A business rule is a statement that defines or constrains some aspect or operation of the business. It is intended to control or influence the behavior of the business operations. Business rules are logical integrity rules, and differ from database oriented referential integrity rules.

Business Rules - example

Business rules in a purchasing system may be:

- An item can be ordered only from its supplier
- When the gross weight and net weight for an item are the same, there is no packaging
- The supplier provides the arrival date of a purchase order after the order is entered into the system.
- Items cannot be removed if there is stock on-hand
- The item on-order quantity is increased when an order for the item is entered
- The item on-hand quantity is increased and the on-order quantity decreased when an order arrives

Referential Integrity Rules - example

In contrast, integrity rules in a purchasing system may be:

- An item must exist for it to be ordered
- If an item has a supplier, the supplier must be known to the system
- Items can not be deleted if the item record is used elsewhere in the system

Business rules are implemented as hooks. A hook is a pre-defined function name that you create in a Data Access Layer script. The 4GL engine will call this function as it executes user interface and database related actions on the table that the DAL is written for.

Business rules fall into categories that determine how the business rule is implemented in the DAL.

The categories, depending on the type of DAL script, are:

- Property or Field hooks
- Object Hooks
- Field Dependencies
- Business Methods

DAL 1

DAL 1 scripts were introduced in Infor Baan 5.0. A DAL 1 script contains database logic (e.g. logical integrity rules), but it usually does not contain *all* business logic for the table it belongs to. Very often business logic is spread across the DAL 1 script and the UI scripts.

Most UI scripts still contain business logic. They can contain, for example, logic to:

- make fields read-only.

- determine which enum constants can be selected in an enumerated field.
- update a field after a change to a related field.

Therefore it is very hard to use the DAL 1 scripts for integrations via Infor LN Business Adapter: the business logic in the UI scripts is skipped if another application connects to a Infor LN table. Therefore this logic must be rebuilt in the BOIs that provide the connection. If you change the business logic in the UI scripts, you must adapt the corresponding BOIs as well.

A DAL 1 script can contain the following types of hooks:

- Object hooks. These hooks operate on the row or record as a whole unit. Object hooks determine if operations on a row are allowed, and specify additional operations that should be performed based on an operation of the row. Examples of object hooks are:
  - Delete the additional employee data in the People package when an employee is deleted.
  - When a sales order line is saved, create a history record of the sales order line.
  - Update the item inventory when a planned stock transaction is created.
- Property hooks. These hooks operate at a field level, and are, among other things, used to verify valid values and calculate default values for table fields. Examples of property hooks are:
  - Validity check that the Quotation expiration date should be after the Quotation entry date.
  - When adding an employee, the language of the company is the default for the employee's language.

For details on DAL 1 scripts, refer to "Data Access Layer" in the *LN Programmer's Guide*.

To create DAL 1 scripts

You can create and edit DAL 1 scripts in the **Program Scripts / Libraries (ttadv2530m000)** session. For DAL 1 scripts, the script type is "DAL".

DAL 2

DAL 2 scripts are introduced in Infor LN. A DAL 2 script contains *all* business logic (database logic + other logic) related to a particular table. The UI scripts of the sessions operating on the table, only contain UI logic.

DAL 2 scripts are therefore perfectly suited for integrations via Infor LN Business Adapter. Other applications can connect, through Business Objects, to Infor LN tables. The business logic in the table's DAL 2 script is executed automatically. So, to build an integration, you don't need to rebuild this business logic in the BOIs that provide the interface.

Various predefined Business Objects, that can be used for integration purposes, are delivered with Infor LN. DAL 2 scripts are only available for tables that are used in these business objects. For the remaining tables, DAL 1 scripts are available. Refer to the Infor LN Business Adapter documentation for details on business objects and integration technology.

In DAL 2 various new hooks are introduced. A DAL 2 script can contain:

- Object hooks: these hooks are identical to the object hooks in DAL 1 scripts.
- Field hooks. Field hooks contain the business logic for table fields. They can contain, for example, logic to:
  - verify valid values on a table field (logical integrity rules).
  - make a field read-only. For example: the contact field of the Business Partner is not a required entry, and is a display field in the UI when the Business Partner is created.
  - make a field invisible.
  - automatically update a field, after a related field was changed. See "Field Dependencies".
  - make input on a field mandatory.
  - hide a particular enumerated constant from an enum list.
- Business method hooks. These hooks are used to disable/enable and hide specific form commands of type "business method".

For details on DAL 2 scripts, refer to "DAL 2 Overview" in the *LN Programmer's Guide*.

To create DAL 2 scripts

You can create and edit DAL 2 scripts in the **Program Scripts / Libraries (ttadv2530m000)** session. For DAL 2 scripts, the script type is "DAL (version 2)".

Field dependencies

In DAL 2 scripts, you can define field dependencies that are used to determine default values. Once these field dependencies are defined, dependent fields update themselves when the fields that they depend on are updated. The field updates are defined in the DAL as a field hook.

Dependencies are also used to update the User Interface. The 4GL Engine uses these field dependencies to disable and enable fields, and to determine enum values.

Examples of field dependencies are:

- The packing field of the Item is yes if the gross weight and net weight of the item are different. A field dependency is defined so that the packing field is updated if the gross or net weight fields change.
- An item without a gross weight cannot have a net weight and packing. A field dependency is defined so that the net weight and packing fields are disabled if the gross weight is zero.

Business methods

Business Methods are operations on a row or set or rows within a table. A business method is a function that performs a task that involves manipulating or checking one or more tables in the database.

Sessions, object hooks or property hooks from another table’s DAL can call a business method. For more information, refer to the *LN Programmer's Guide*.

Examples of business methods are:

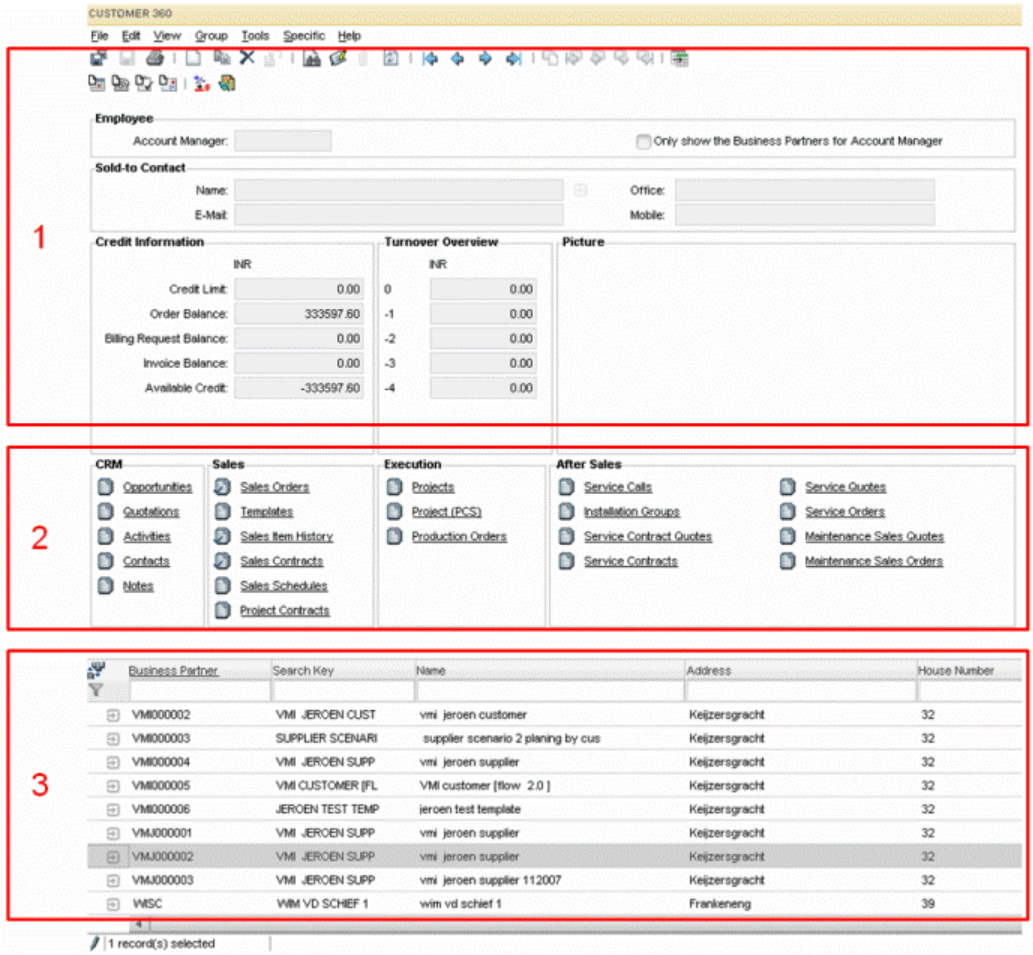
- The sales order line session can remove an order line. The object hook for deleting the sales order line calls a business method to also remove the unpaid commissions.
- In the Purchase Order session, check if it is allowed to change a purchase order header based on the purchase order status and the business partner status.
- If a purchase order issues components, the object hook for the purchase order line calls a business method to issue the components for the purchase order. The business method inserts items of the bill of material into a purchase order components table.

Dashboards

Dashboard sessions are used to quickly access multiple sessions that are related to a specific object. Examples of objects are item, work center, or project. Through a dashboard, users such as engineers or shop floor planners can easily carry out all tasks related to their jobs. Relevant details of the object are presented on the dashboard, as well as the possibility to open the sessions in which additional information for the object is stored.

To summarize, the dashboard gives the user a quick overview of the status of a specific object without having to open all related sessions immediately. With a dashboard, a user can easily perform a specific task.

This diagram shows the sections in a dashboard:



Customer 360 (tdsmi1500m000) dashboard

A dashboard session consists of these sections:

1. Details

   The Details section shows information about the object.

   If you select a record in the list in the lower part of the session (3), the record's details are displayed in the Details section (1).

2. Links

   A series of links (2) is displayed to start sessions that are related to the session's object. A check mark indicates whether the link already contains information related to a selected record of the object.

   Click the links to display, maintain, or create information related to an object's record.

3. Records

For the dashboard's object (3), a number of records is listed.

If you select a record, the record's details are displayed in the Details section (1), and you can click the links (2) to display, maintain, or create information related to a record.

For more information on dashboards, see the Infor Web Help.

To create dashboards

For details on how to create dashboards, see *To create dashboards (p. 104)* .

Domains

Domains define common information about data such as data type, length, alignment, valid ranges, display format, and capitalization rules.

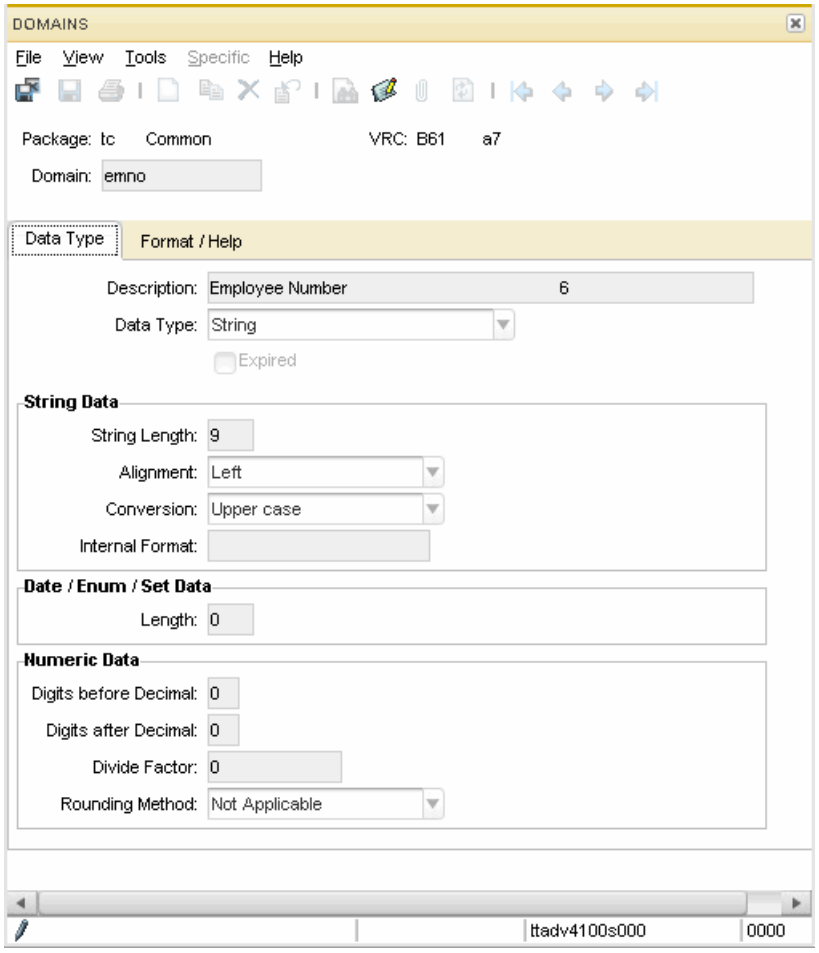
Domains insure consistent data types for fields and variables. Domains can be linked to table fields, form fields, and program variables.

The following table shows some domains of the tc (Common) package.

Domains for tc package

Domain	Description	Type
date	Date	UTC Date/Time
emno	Employee Number	String
relc	Reconciliation Element	String
reop	Reorder Point	Long
rntp	Run Type	Enumerated
rutm	Run Time in Minutes	Double

The following figure shows the details of the tc emno (Employee Number) domain.



Domain properties Example

Domains of type Enumerated or Set have constants with language-dependent descriptions. Each constant has three characteristics:

- Numeric value that is stored in the database
- Constant Name that the programmer can use
- Constant Description that provides the description of the option to the user, in the language of the user

For example: the tc styp (Schedule type) domain has the following constants:

Constant	Constant Name	Description
1	not.applicable	Not Applicable
2	pull	Pull Schedule
3	push	Push Schedule

To create domains

You can create domains in the **Domains (ttadv4500m000)** session.

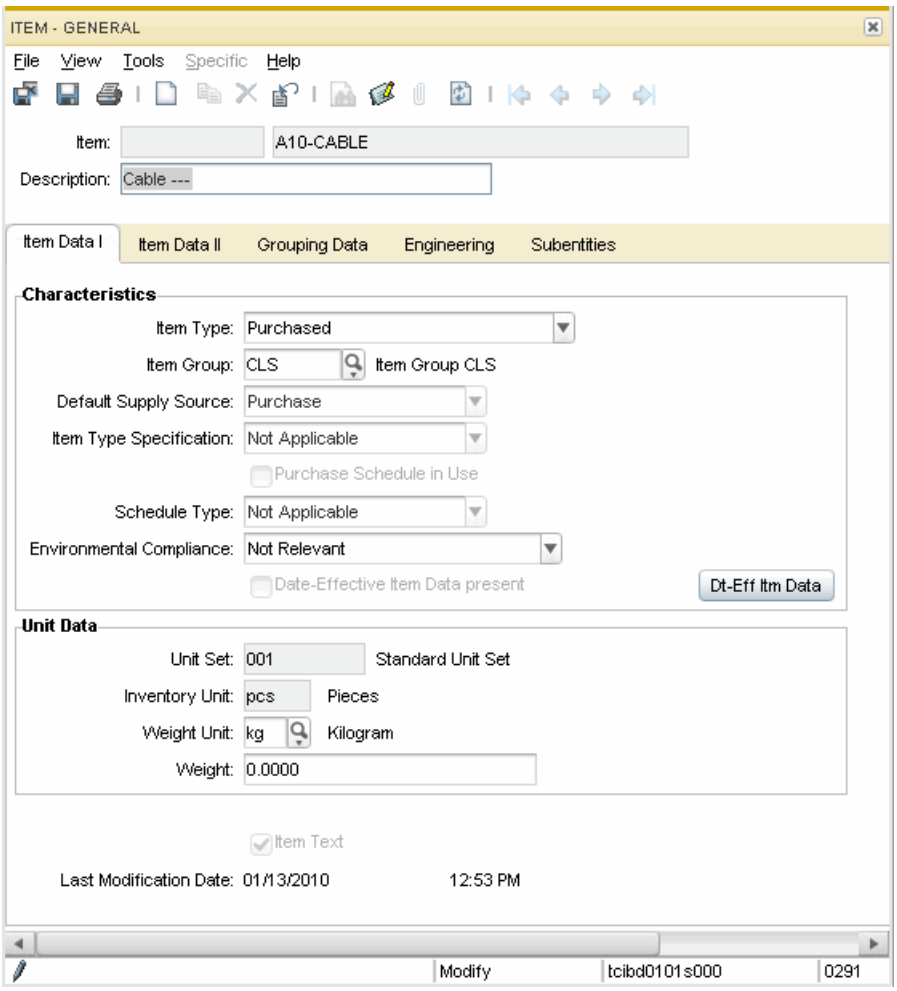
See *To create a data model (p. 66)* .

Forms

The form is the user interface part of the session. Forms, which are presented to users, include data and actions that users can perform on that data. The session and form are integrated; one form per session is defined. The form definition in the session identifies the fields, labels, and options that are available in the session's overview display window and details window.

Example

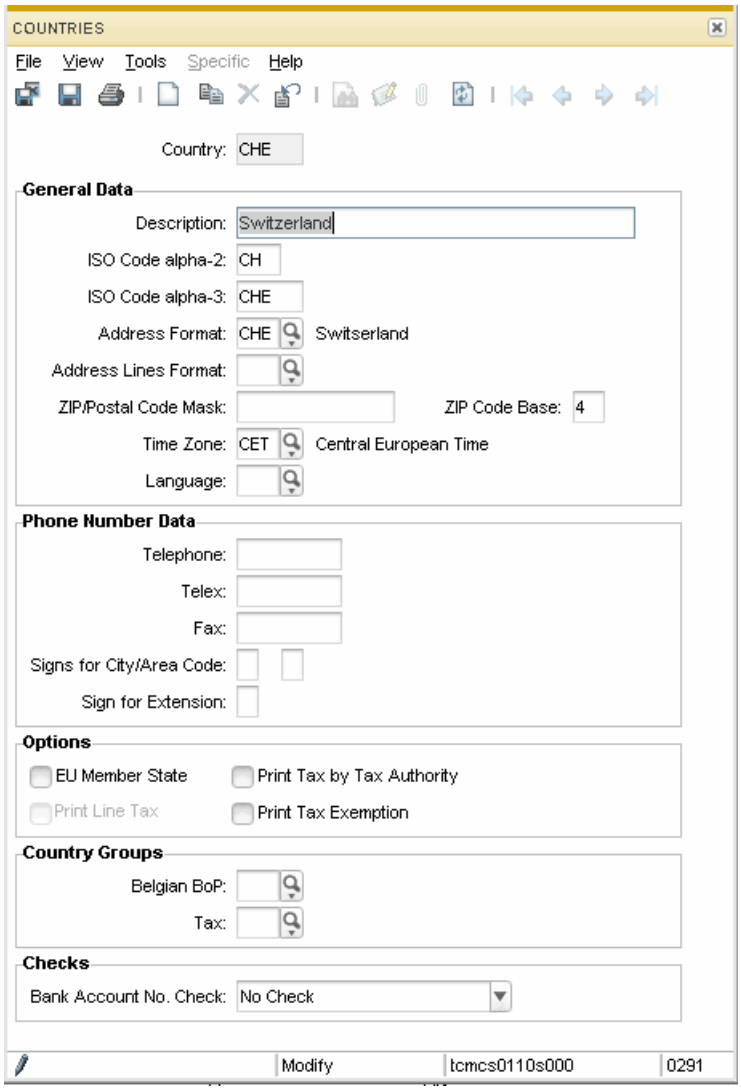
Field groups allow certain fields to be displayed together within a box, or within a tab. The **Item – General** session contains more than one tab.



Item - General

Example

Some sessions can fit all their fields on one tab, like the **Countries** detail session.



Countries

A Form provides the rules for the user dialog, or panel, that is displayed. You create and edit sessions and forms using the Sessions (ttadv2500m000) session. This session acts as a Developer's workbench as it provides access to all the major components of a session.

To create forms

To create and edit forms, you must use the Infor LN Dynamic Form Editor that is part of Development Tools. This is a PC based application that should be installed for each developer. The Infor LN Dynamic Form Editor also provides access to the major components of the session.

The form editor allows you to specify the available session types for the form (overview and detail), the field groups, labels and fields that are used to create the form.

The form that is presented to the user is created dynamically:

- The form layout is used to determine the labels and fields, the grouping of the fields, and the sequence of the fields.
- The table field authorizations of the user insure that only fields that the user is allowed to view are shown on the form.
- The mode the session is started in determines if the session is in overview mode or details mode.

For details, refer to

- *To create sessions (p. 73)*
- *To create forms (p. 75)*
- *To edit forms (p. 78)*
- *To create commands and buttons for a session (p. 85)*

For more information, refer to the following topics in the Infor Web Help:

- Dynamic sessions and dynamic forms

Functions

Functions allow you to perform a programming task multiple times with different values. A function is declared in the functions section of a script, in a library, or in a separate function script (include).

The possibilities for calling a function depend on how the function is declared:

- If a function is declared in the declaration section of a script, you can only call the function within that script.
- If a function is declared in a library, or in a separate function script, you can call the function in multiple scripts and libraries.

For more information, refer to "Functions" in the "3GL Language Features" section in the *LN Programmer's Guide*.

Example

The supplier session calculates the cumulative order volume within the last 30, 60 and 90 days. You declare a function that can perform this calculation for any range of days. The function has a type so that it can be used in an expression.

```
function domain cxquan supplier_volume(long day1,long day2)
{
    ... commands here ...
return( ... )
}
```

If the function is declared as a type, it must provide a return value. You call this function using the syntax:

```
vol_30 = supplier_volume(0,30)
vol_60 = supplier_volume(30,60)
```

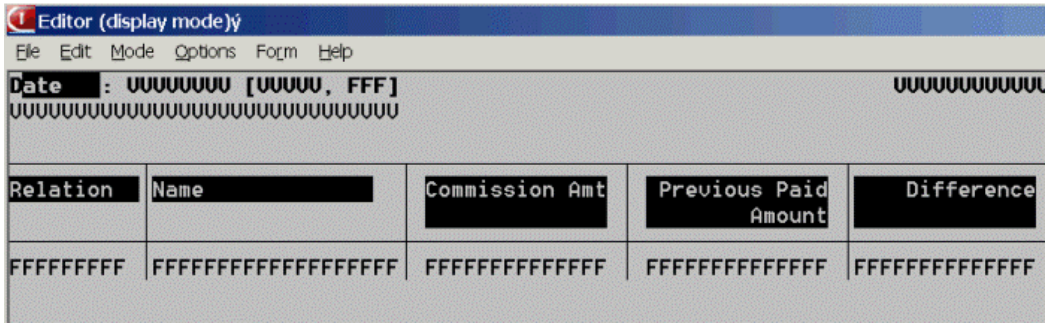
To create functions

To create a function in a program script or in a library, you must edit the script/library via the **Program Scripts / Libraries (ttadv2530m000)** session. If you define a function in a library, you must link that library to the scripts and libraries where you want to call the function.

To create a function in a separate function script:(include), you must create the function script through the **Functions (ttadv2560m000)** session. You must include this function script (through an "#include" statement) in the scripts and libraries where you want to call the function.

Labels

A label is 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.



Labels in a report-layout

For each label code, you can specify:

- A label description. This is the label text which you can edit and translate.
- The length. This is the number of characters of the description.
- The height. This is the number of lines of the description.

Multiple descriptions, with different lengths and heights, can exist for the same label code. See the following table for an example:

Label code	Language	Height	Length	Description
ttttaad501.se-qu	1 (Dutch)	1	6	Volgnr
ttttaad501.se-qu	1 (Dutch)	1	10	Volgnummer
ttttaad501.se-qu	1 (Dutch)	2	6	Volg-%num-mer
ttttaad501.se-qu	2 (English)	1	6	SeqNo.
ttttaad501.se-qu	2 (English)	1	12	Sequence No.
ttttaad501.se-qu	2 (English)	2	8	Se-quence%Num-ber

Note

A % sign in the description indicates a line break. For example: when you place the label with the "Sequence%Number" description in a report layout, the label is displayed as:

```
Sequence
Number
```

When you place a label on e.g. a form or a report layout, you can edit the label length requirements within the form field / report field. In this way you can make sure that the appropriate version of the label (e.g. the shorter version) is used in the form / report.

To create labels

You can create labels in the **Labels (ttadv1140m000)** session.

If you create new labels, you must compile these in the **Compile Labels (ttadv1243m000)** session so that Infor Enterprise Server can display the new labels at run time.

For more information, refer to the Infor Web Help.

Libraries

A library, also called Dynamic Link Library (DLL), provides application-specific functions that can be used throughout the system, by many sessions. A library is a script that is stored in a separate component. The library is compiled independently of the program scripts that use it. Libraries are loaded at runtime by sessions that use them. When a session needs to access a library, the library is loaded, and the relevant routine is executed.

To create libraries

You can create and edit libraries in the **Program Scripts / Libraries (ttadv2530m000)** session.

Libraries are a type of program script. The program script code should start with “dll” and the script type should be “General Library”. This helps categorize the scripts into their general use.

Note

You edit and compile a library the same way that you edit and compile a program script. When you create a function in a library, you should consider these differences:

- When you name the function, precede the function name with the library name.
- Functions are declared as EXTERN.
- The DllUsage / EndDllUsage block is used to communicate how the library function can be used. This text will be displayed when you generate the library documentation or perform a bic_info on the resulting object file.

For more information, refer to "Dynamic-link libraries" in the "3GL Language Features" section in the *LN Programmer's Guide*.

User Exit DLLs

LN supports the use of User Exit DLLs. Customers can implement User Exit DLLs to create extra business logic, without having to change the standard LN software. The business logic in a User Exit DLL is automatically executed before and after the standard 'before' and 'after' handling of save and delete actions. In this way, you can conditionally publish data changes to the outside world.

A User Exit DLL meets these conditions:

- The name of the DLL consists of the table code with "ue" as suffix. For example, `tccom001ue`
- The DLL includes bic_dal, as follows: `#include <bic_dal>`

A User Exit DLL (UEDLL) contains specific hooks, such as the following:

- `ue.after.after.destroy.object()`
- `ue.after.before.save.object()`

In these hooks, you define the extra business logic to be executed. For 4GL sessions, the hooks are executed just before and after the corresponding defined standard sections/hooks of the UI script and/or

DAL script. For 3GL scripts and DLLs, the hooks are executed just before and after the corresponding database operation, such as db.insert, db.update, or db.delete.

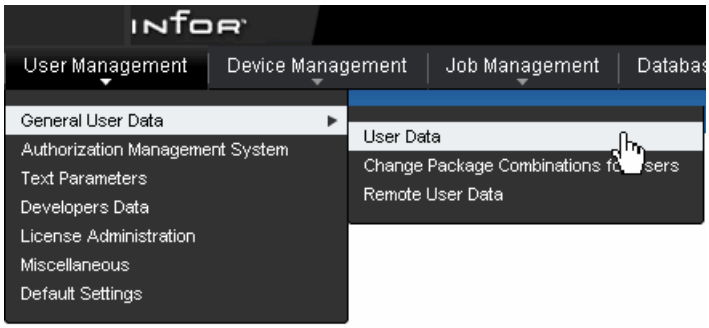
Note

The business logic in a User Exit DLL should not change the standard functionality of the LN software; for example, these DLLs are only intended for integration purposes, or to perform extra steps.

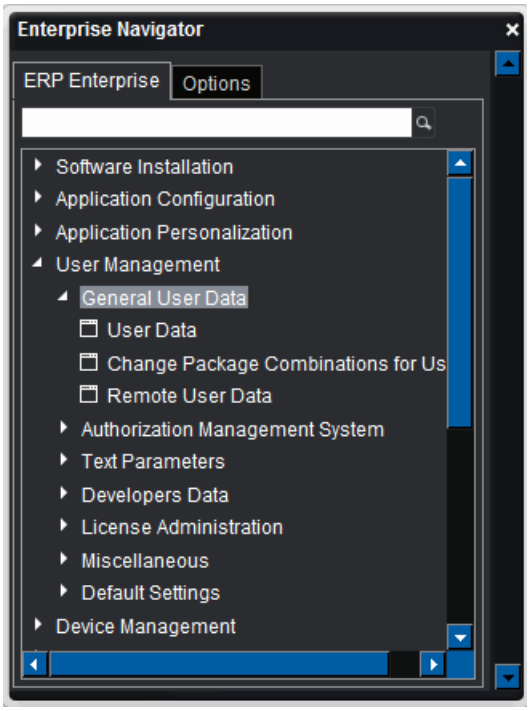
Menus

Menus are used to organize the Infor LN sessions in a logical folder / subfolder structure. The folders and subfolders usually represent Infor LN packages and modules. The Menu browser is used to open the folders and subfolders in order to find the sessions. Sessions can be launched directly from the Menu browser. The Menu browser is used in Infor Ming.le, Infor ES Web UI, and Infor LN Worktop.

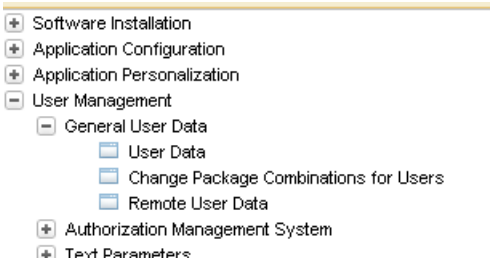
Infor LN users can have their own customized menus. The start menu for a user must be defined in the User Data (ttaad2500m000) session.



Menu navigation in Infor Ming.le- pull-down menus



Menu navigation in Infor Ming.le- Enterprise Navigator Web part



Menu navigation in Web UI

To create menus

To create menus you must use the Menus (ttadv3560m000) and Menu Fields (ttadv3561m000) sessions.

See the "Creating menus" topic in the Infor Web Help.

Messages

Messages are language-independent software components that allow you to customize dialog messages.

For each message, you can specify:

- The message code. This is the unique identifier of the message across all languages.
- The message type, e.g. "Warning".
- The message. This is the message text which you can edit and translate. This message can contain codes that are substituted when the message is displayed. For details on substitution symbols, refer to the description of the sprintf$() function in the *LN Programmer's Guide*.

See the following table for a sample message:

Message code	Message Type	Language	Message
tipcss0292	Warning	1 (Dutch)	Artikel reeds aanwezig als component in productiestuklijst
		2 (English)	Item already found as component in production BOM

Messages are used in a variety of places within the application. You can use messages:

- To replace the default reference error message on table definitions
- Within program scripts and libraries
- To set error messages from a DAL
- As the contents of a form field using the form.text$ function

If a message is produced while a session is executing, a dialog box is displayed, and you must click **OK** to continue. If the session is running in a job, the message is logged in the job history messages.

To create messages

You can create messages in the **Messages (ttadv4551m000)** session.

For more information, refer to:

- the Infor Web Help
- the Message Handling subtopic in the Functions topic in the *LN Programmer's Guide*

MMT Sessions

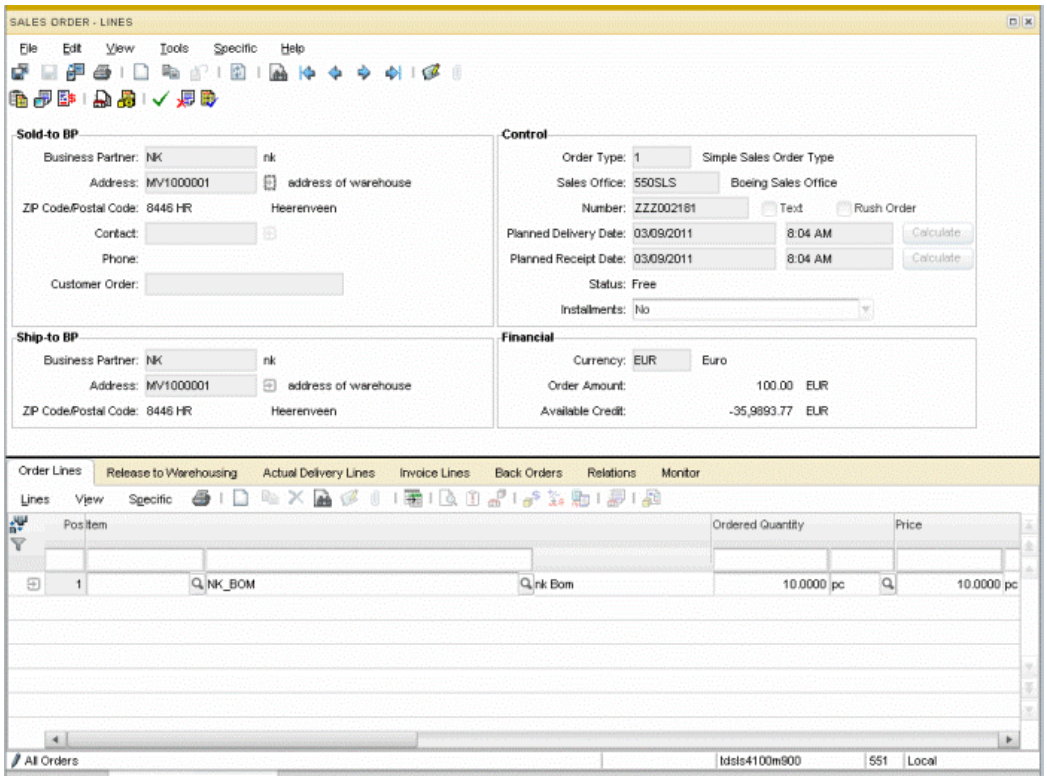
A multi-main table session shows multiple tables and allows you to perform maintenance commands on the main entity and the child tables. Hence, the term “multi-main table”.

A multi-main table session shows detailed information for a main entity. Additional tabs are available that show related tables to the entity.

Example

Customer service representatives use the **Sales Order - Lines (tdsls4100m900)** session to enter, display and process sales orders. When the representative opens an order:

- The order header is displayed
- The order lines are displayed
- Other information that is related to the sales order number is displayed on additional tabs.



Sales Order Multi-Main Table Session

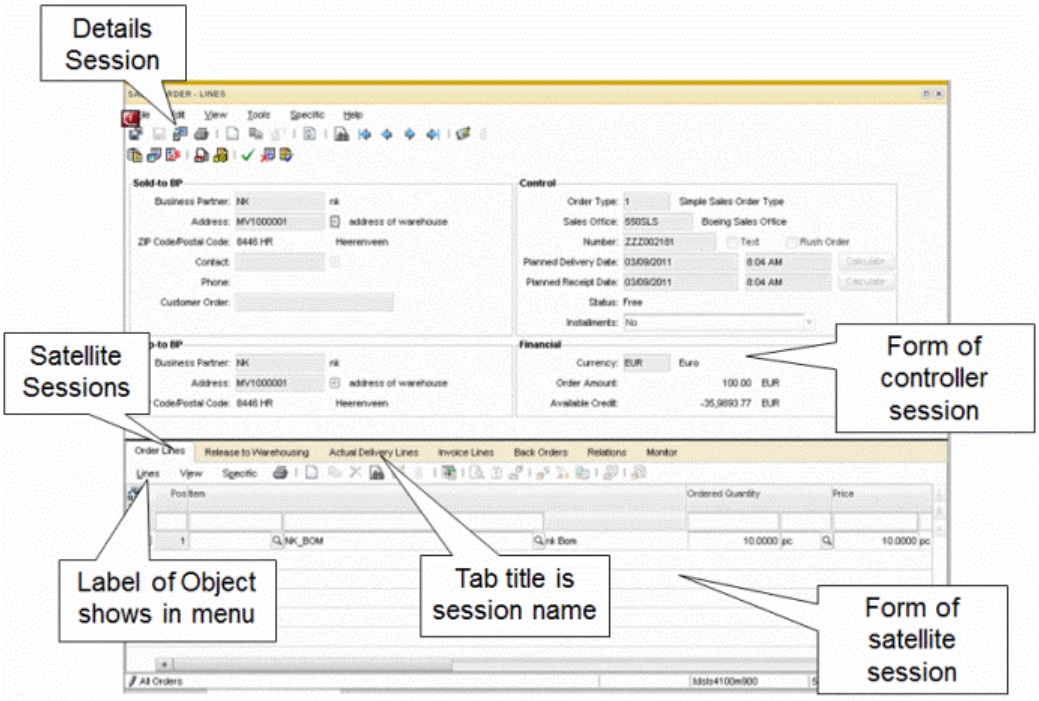
You can identify a multi-main table session from the two tool bars: the first toolbar is associated with the sales order header, and the lower toolbar is associated with sales order lines, deliveries and so on. You can create or edit an order or an order line.

Structure

The appearance of the multi-main table session is determined by the following:

- The multi-main table controller session is a maintain session for the main entity. The form of this session determines the top portion of the session.

- In addition to the standard tool bars for maintenance of the main entity, an addition icon is provided. The details icon will open a details session for the main entity. This session is the synchronized dialog for the controller session.
- Satellite sessions are displayed as tabs. The form of the satellite session determines the fields that are displayed in the session. A satellite session can have multiple tabs within the session.
- The title of the tab is the session name of the satellite session.
- The “label of object” field shows in the Lines menu and other menus.



Multi-Main Table Session Structure

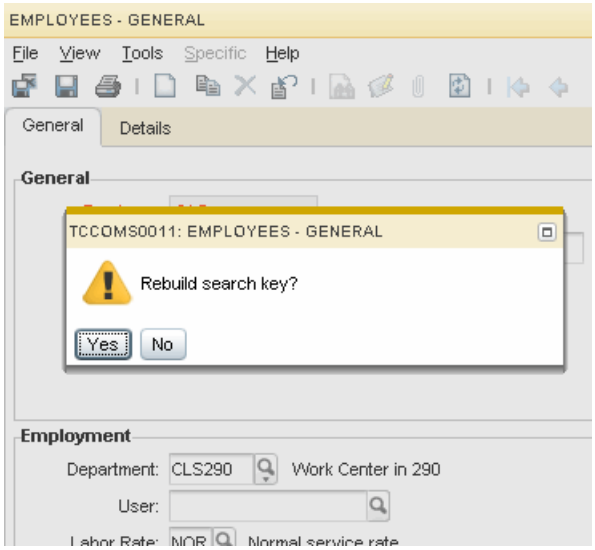
For more information on MMT sessions, refer to the Infor Web Help.

To create MMT sessions

You can create MMT sessions using the **Sessions (ttadv2500m000)** session. This session enables you to create and link all the major components of an MMT session. For details, refer to *To create MMT Sessions (p. 117)* .

Questions

Questions are language-independent software components that are used to ask situation-dependent questions to which the user must respond. At runtime, the questions are displayed in the language that is specified for the current user.



Sample question

For each question, you can specify:

- The question code. This is the unique identifier of the question across all languages.
- The question type, e.g. "Warning".
- The question. This is the question text as it is shown when activated. The question text can contain codes that are substituted when the question is displayed. For details on substitution symbols, refer to the description of the sprintf$() function in the *LN Programmer's Guide*.
- The domain belonging to the question, and the default answer. The domain determines the possible answers to the question. The domain must have the Enumerated data type.

This table shows a sample question:

Question code	Question Type	Domain	Language	Question
ttaad31022	Warning	ttyeno	1 (Dutch)	Device geblok-keerd, opnieuw proberen?
			2 (English)	Device is locked, try again?

To create questions

You can create questions in the **Questions (ttadv4561m000)** session.

You can use questions within program scripts and libraries using the ask.enum() function.

For more information, refer to:

- the Infor Web Help
- the description of the ask.enum() function in the *LN Programmer's Guide*

Reports

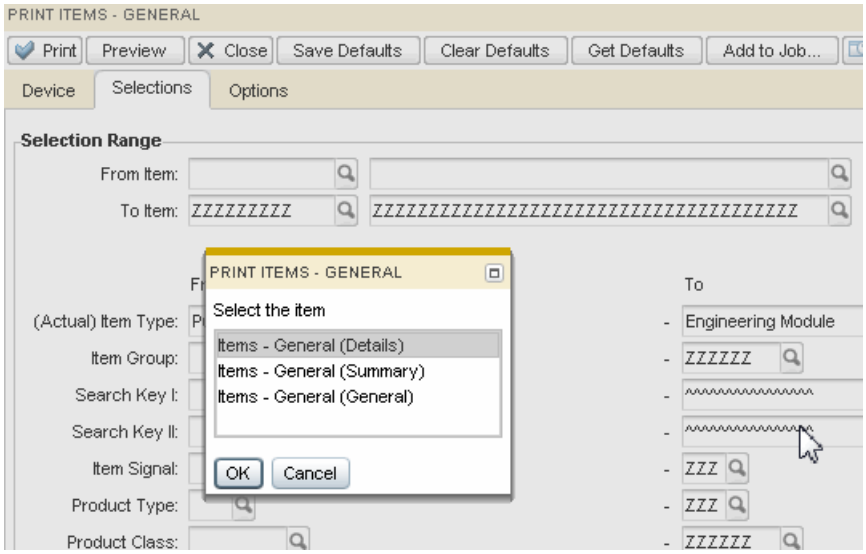
Reports are used to show data on the screen, or print it on a printer or other output device. Reports are used in (print) sessions and SQL queries. A session can have multiple reports. When you run a session that has multiple reports, a list of available reports is displayed. An SQL query can have only one report.

Example

The Print Items - General (tcibd0401m000) session has multiple reports:

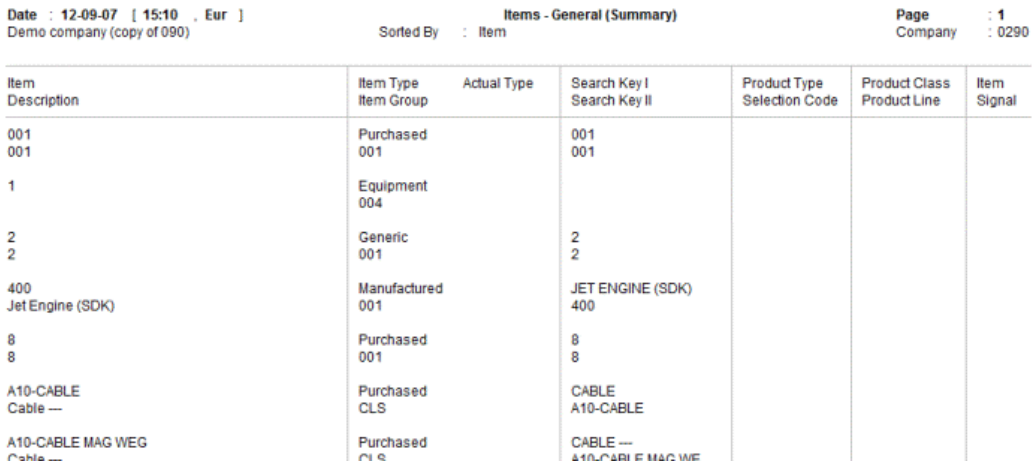
- Items - General (Details)
- Items - General (Summary)
- Items - General (General)

When you click **Print** in this session, you must select one of these reports.



Print Items - General - report selection

When you select the Items - General (Summary) report, this report is printed:



Date : 12-09-07 [15:10 , Eur]
Demo company (copy of 090)

Items - General (Summary)
Sorted By : Item

Page : 1
Company : 0290

Item Description	Item Type Item Group	Actual Type	Search Key I Search Key II	Product Type Selection Code	Product Class Product Line	Item Signal
001 001	Purchased 001		001 001			
1	Equipment 004					
2 2	Generic 001		2 2			
400 Jet Engine (SDK)	Manufactured 001		JET ENGINE (SDK) 400			
8 8	Purchased 001		8 8			
A10-CABLE Cable ---	Purchased CLS		CABLE A10-CABLE			
A10-CABLE MAG WEG	Purchased		CABLE ---			

Items - General (Summary) report

Report structure

LN reports contain one or more of the following layouts:

Layout	Description

Before.report	Layouts of this type are printed once at the beginning of the report. This layout can be used to present information about the report. For example, the title page of the report.
Header	Header layouts are printed at the top of every page.
Before.field	You can define this layout for each input field for which a Sort Mode is defined in the **Report Input Fields (ttadv3532m000)** session. Each time the value of the sorted input field changes, this layout is printed before the Detail lines. In this way the records are grouped. If a group of records spans more than one page, the before.field layout is repeated at the top of the new page (following the header).
Detail	Detail layouts are used for the individual records included in the body of the report. They are printed every time that data is sent to the report.
After.field	You can define this layout for each input field for which a Sort Mode is defined in the **Report Input Fields (ttadv3532m000)** session. Each time the value of the sorted input field changes, the layout is printed after the detail lines. In this layout, the presorted input field still contains the old value. This layout, for example, can be used to print totals for a particular group of detail lines.
Footer	Footer layouts are printed at the bottom of each page. This layout is used to print: ■ Titles ■ Page numbers ■ Page totals
After.report	This layout is only printed at the end of the report, and after all other layouts. If this layout does not fill a complete page, the footer layout will be printed at the bottom of the page, where the After.report layout(s) is printed.

The contents and layouts of reports are defined in the data dictionary. In addition, you can link a report script to a report. In a report script, you can program actions that you want to be performed at particular stages of the report execution. For example, you can create a script to perform calculations on the report data or to read records from related tables that are not automatically available to the report.

You program report scripts in the same way as you program 4GL program scripts, except that report scripts use different event sections and some special functions.

To create reports

You can create reports in the **Reports (ttadv3530m000)** session and in the **Report Generator (ttadv3130s100)** session. You can start these sessions from the **Sessions (ttadv2500m000)** session and from the **Sessions (ttadv2100s000)** session respectively.

You can change the report layout in the Report Editor that is part of Development Tools.

For details, refer to *To create reports (p. 96)* .

For more information, refer to the following topics in the Web Help:

- Reports Overview

Report scripts

A report script allows you to customize the processing of the report. A report script consists of event sections in which you program actions to be performed at particular states of execution of the report. The statements programmed in a report script section consist of a combination of 3GL language statements and report script functions.

You program report scripts in the same way as you program 4GL program scripts, except that report scripts use different event sections and some special functions.

Report scripts allow you to:

- Perform additional queries
- Calculate complex print expressions
- Aggregate complex print expressions
- Update rows in the database
- Add fields from tables that are not queried by the program script, when you are not able to manipulate the program script
- Manipulate pre-defined variables in the report, such as if the layout should be printed or not.

Report script structure

A report script consists of one or more event sections in which you program actions to be performed at particular states of execution of the report. The statements programmed in a report script section consist of a combination of 3GL language statements and report script functions.

Report scripts support the following event sections:

- program sections, e.g. declaration, before.program, and after.program.
- report sections that correspond to a layout, e.g. before.report, header, and detail. Report sections can be processed before a layout, or after a layout.
- text field sections

The report script is compiled automatically with the report when you compile the report.

Note

For details on Report scripts, refer to "Report Script Features" in the *LN Programmer's Guide*. Predefined variables used in a report are also listed.

To create report scripts

You can create and edit report scripts in the **Reports (ttadv3530m000)** session.

For more information, refer to the following topics in the Web Help:

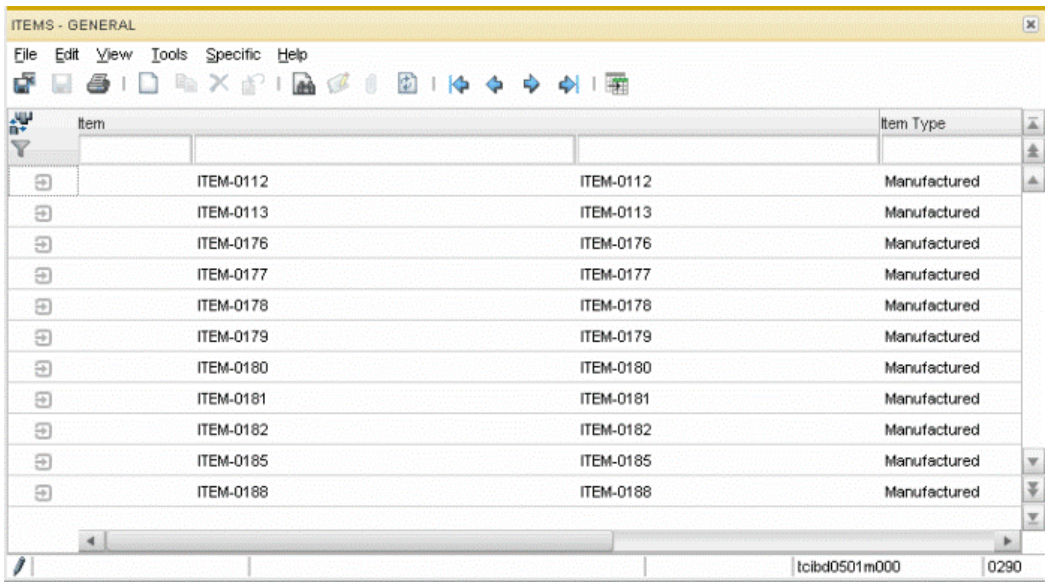
- Reports Overview

Sessions

A session performs an activity. Sessions are used to present data, edit data, and process data. Each session has a code. The session code is displayed in the status bar of the session window. A session consists of multiple components that work together, such as a form and an object. A session object is a compiled UI script.

Overview session and details session

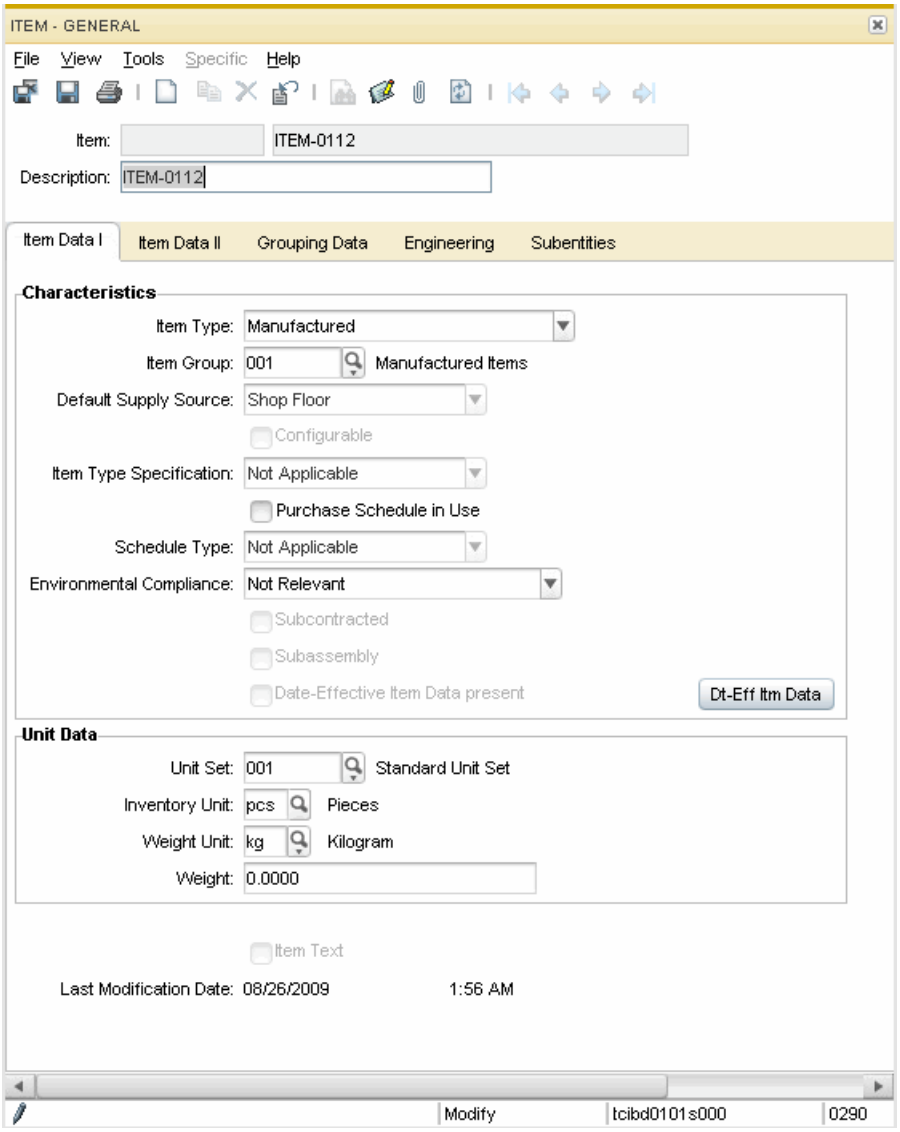
An *overview* session shows multiple rows from a table in a grid. Overview sessions are also called Multi-Occurrence sessions, because they display multiple records, or occurrences, from the same table. Scroll bars allow you to view rows above or below the current rows that are displayed in the grid. Based on security authorizations, you are able to insert, edit, copy, and delete entries in the grid.



Overview session

A *details* session shows one row and is also called a single-occurrence session. The details session is opened when you insert, edit or copy from the overview session. The details session allows you to edit

individual field values and save your changes. The session may also have additional options that can be performed using buttons, icons or specific menu options.



Details session

Synchronization

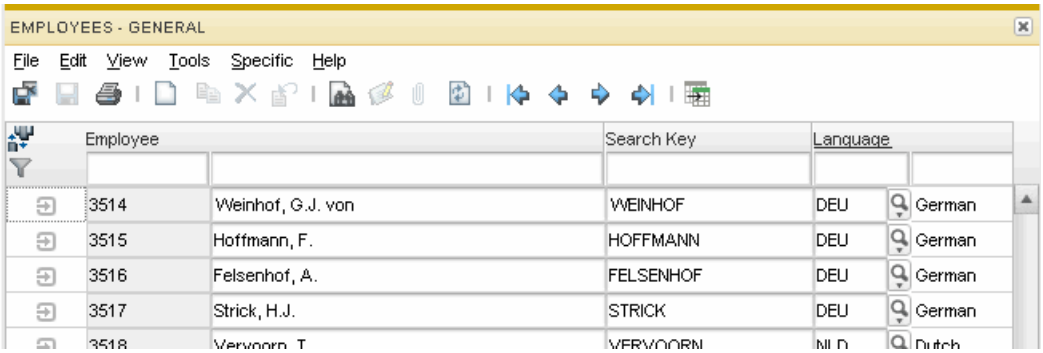
The details session is a synchronized dialog of the overview session. Synchronized sessions work together.

- When the details record changes, the overview session will show the changes if the affected fields are on the overview session form.

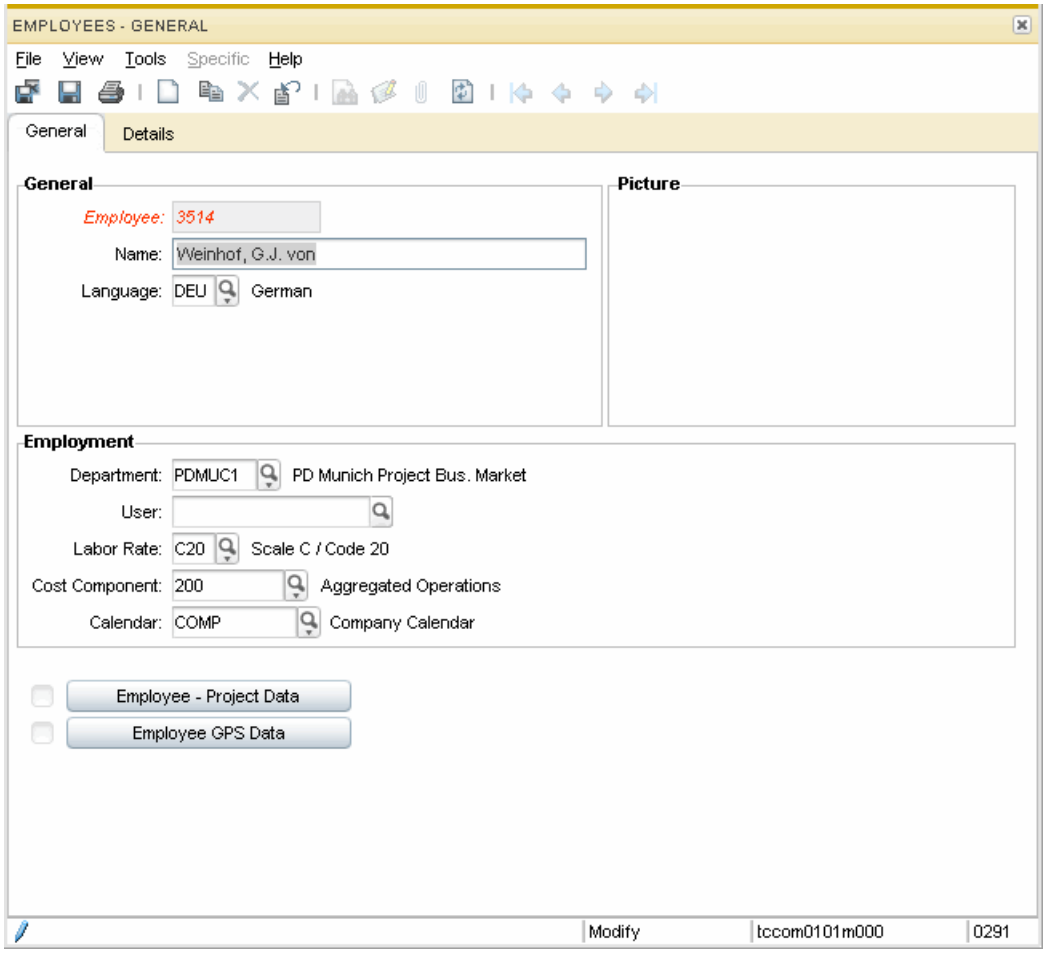
- When a record is selected in the overview session, the details session will show the selected record.

Usually the overview session and the details session are two separate sessions, for example: the Items - General (tcibd0501m000) overview session and the Item - General (tcibd0101s000) details session. See the previous figures.

However, the overview session and the details session can be the same session. For example the Employees - General (tccom0101m000) session can run as an overview session and as a details session. You use the form editor to indicate the fields that appear in the overview session, and which fields appear in the details session.



Employees - General (tccom0101m000) - overview session

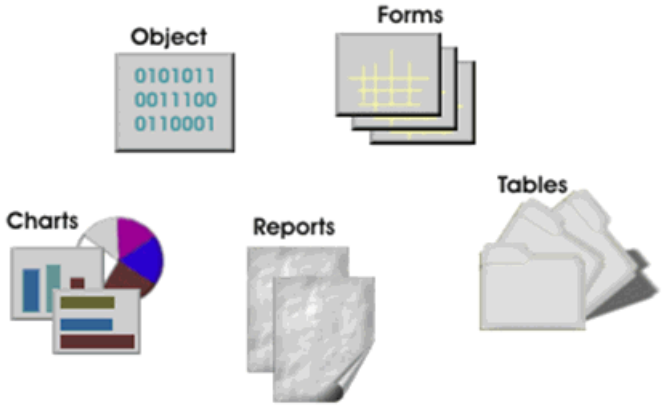


Employees - General (tccom0101m000) - details session

Structure of a Session

A session is a collection of system components that work together to perform a task. Each component will have a unique identifier, or code, to distinguish it from other components. The system components include:

- Form
- Table definitions
- Reports
- Charts
- An object that is a compiled UI script



Components of a session

To create sessions

You can create and edit sessions and forms using the **Sessions (ttadv2500m000)** session. This session acts as a Developer's workbench as it provides access to all the major components of a session. For details, refer to *To create sessions (p. 73)* .

For more information on sessions and forms, refer to the following topics in the Infor Web Help:

- Dynamic sessions and dynamic forms
- Session Synchronization
- To implement dynamic index switching
- Multi Main Table sessions

SQL queries

Queries are the backbone of a database-oriented application. You use Structured Query Language (SQL) to retrieve information from the database.

Infor LN supports multiple Relational Database Management Systems. Each database system is supported in the architecture by using a driver: a database specific program that translates SQL syntax from the SQL used in the object to the SQL used by the database system.

In Infor LN the following types of SQL queries are used:

- 4GL program queries
- Stand-alone queries

4GL program queries

You can create queries in 4GL scripts or libraries, e.g. in a UI script, a report script, a DLL or a DAL. These queries can read and update data in the database. At runtime, the queries are triggered during session execution, for example by an event or by a form command.

Note

There are two ways to use LN SQL in an LN 4GL program. You can embed it in the language (embedded SQL), or you can create queries using 4GL functions (dynamic SQL). For more information, refer to the "SQL programming" subtopic in the " LN SQL" topic in the *LN Programmer's Guide*.

An embedded query starts with a SELECT statement that allows you retrieve a selection of data from a number of tables based on conditions defined in the WHERE clause.

The SELECT statement has the general syntax:

```
SELECT columns ...
FROM table(s) ...
WHERE each row fulfills the condition(s) ...
```

The columns you select are typically table fields that are selected from tables. The where clause in an embedded query allows you to specify conditions using either columns that are selected or based on variables declared in your program script. The rows you select can be processed individually within a selectdo loop. The selectdo loop provides an iteration mechanism for the records selected by the SELECT statement.

For details, refer to " LN SQL" in the *LN Programmer's Guide*.

To create or modify queries in a 4GL script or library, you must use the **Program Scripts / Libraries (ttadv2530m000)** session.

Stand-alone queries

A stand-alone query is a query that is not embedded in any other software component, and that is executed independently.

Stand-alone queries can only *read* data from the database. A stand-alone query cannot add, delete, or update records in the database.

The general structure of a stand-alone query is:

```
SELECT                  For the examples the statements are
 <select list>          written in capitals. In the Text
                        Editor both uppercase and lowercase
                        letters can be used.
FROM
    <from list>
 [WHERE
     <where condition>
 [GROUP BY
```

```
    <group condition>
 [HAVING
    <having condition>
      ]
    ]
  ]
[ORDER BY
   <order list>
]
```

The output of a stand-alone query is sent to a report or to a chart.

You can create stand-alone queries in the **Query Data (ttadv3580m000)** session. This session enables you to create queries with Easy SQL or with the Text Manager:

- Easy SQL helps users with insufficient knowledge of the structure of the data model to define queries in a simple, menu-driven way.
- The Text Manager enables you to create more complex queries.

You can execute stand-alone queries in the following ways:

- via the **Query Data (ttadv3580m000)** session
- via the **Execute Query (ttadv3280m000)** session
- from a menu. You must first add the query, as a menu field of type "Query", to a menu.

For details, refer to the Infor Web Help.

Table definitions

A table definition defines the structure of a table. A table definition contains fields and indices. Table fields are linked to domains that define the data type and several characteristics of the fields.

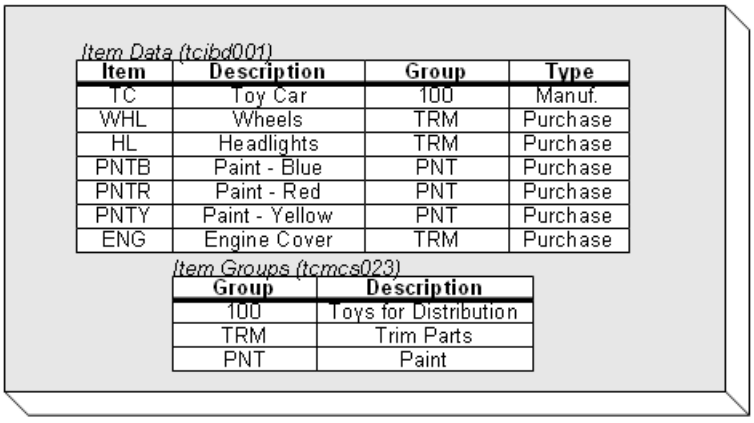


Table Data Example

Fields, domains and indices

A table has fields. Table fields store individual pieces of data such as a customer name, the quantity of an item ordered, or the date that a journal entry was made.

Table fields are linked to domains. Domains are components that define common information about data such as:

- the data type, e.g. a character type for customer name, a number type for quantity, and a date type for journal date,
- valid ranges, or
- special characteristics such as capitalization rules.

Domains insure consistent data types for similar table fields.

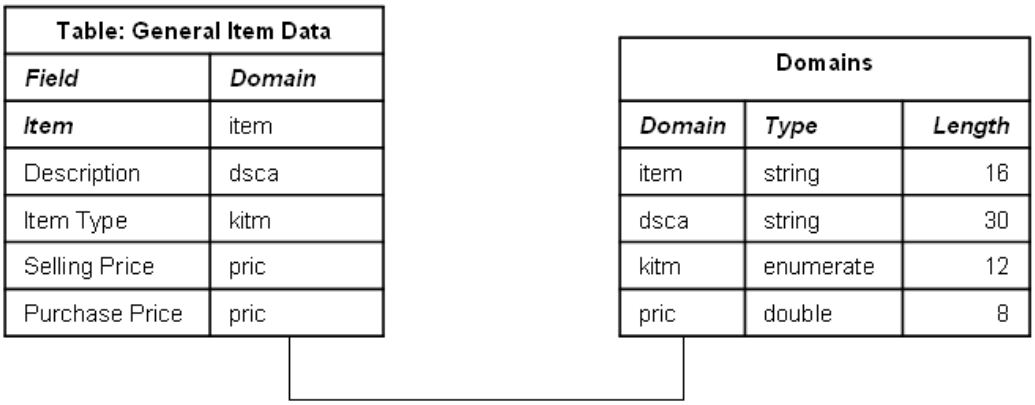


Table: General Item Data	
Field	*Domain*
Item	item
Description	dsca
Item Type	kitm
Selling Price	pric
Purchase Price	pric

Domains		
Domain	*Type*	*Length*
item	string	16
dsca	string	30
kitm	enumerate	12
pric	double	8

Table Fields and Domains Example

A table must have at least one index. An index consists of one or more table fields that are used to sort and search records in the table. The first index is always the Primary key, which is the unique identification for a record in a table.

Related Tables and References

The table may have a related table. A related table means that a field in the table will refer to the key field of another table. In this way, data can have relationships: customers can have orders; inventory stores items in warehouses; and employees work in a department. You can use a tool in the Enterprise Modeler to create diagrams from these defined relationships.

To create table definitions

You can create table definitions in the **Table Definitions (ttadv4520m000)** session.

For details, refer to *To create a data model (p. 66)* .

UI scripts

The default behavior of a session is handled by the 4GL engine. If you require additional functionality or want to bypass the default functionality, you program your changes in the session's UI script (Program script). The UI script is compiled in the session object. The object contains only the exceptions to the normal operating procedures of the system. The 4GL Engine executes the normal operating procedures of the system, and you write the exceptions.

The UI script contains events to hook into the 4GL Engine, and describe the actions that should happen as a result of an event. Events trigger code. When the session runs and an event triggers, the 4GL Engine uses the session object to perform the appropriate actions.

Actions that could take place as the result of an event are:

- Calculate a field value before a field displays.
- Synchronize the current (parent) session with a child session when you run a form command. For example: a parent session that uses the Orders table is synchronized with a child session that uses the Order lines table.
- Send a file to the user’s PC when you save a record.

The syntax of the UI script contains complete programming syntax capabilities such as variable declarations, expressions, operators, transfer of control, iterations (loops), functions, embedded SQL, and a full featured functions library.

Note

As a developer, you must be diligent to provide only user interface elements in the UI script. Logical integrity and business logic elements will be added in the Data Access Layer (DAL).

Event sections

UI scripts consist of one or more event sections in which you program actions to be performed at particular states of execution of the 4GL Engine. It is not necessary to program all sections; only those for which you need to provide the required functionality. The statements programmed in a section can be a combination of 3GL/4GL functions and 3GL statements.

The event sections are categorized. A section type consists of a main section and, for most section types, subsections. A main section indicates the object for which the programmed actions must be executed. A subsection specifies when the actions must be executed. The section types are:

Section Type	Main section	Subsections
Program	Yes	No
Zoom From	Yes	Yes
Form	Yes	Yes
Group	Yes	Yes
Choice	Yes	Yes
Field	Yes	Yes
Main Table	Yes	Yes

Field sections are only used for non-table fields, table fields are handled by the DAL script.

Main table sections must not be used anymore, because the logic for the Main table is done via the DAL.

For more information, refer to the "Event sections" subtopic in the "4GL Language Features" topic in the *LN Programmer's Guide*. The related help topic for the Flow of the 4GL engine is useful for advanced developers to understand event loops and the sequence of events in the 4GL engine.

To create UI scripts

You can create and edit UI scripts in the **Program Scripts / Libraries (ttadv2530m000)** session. When you create a new session in the **Sessions (ttadv2500m000)** session, a corresponding UI script is generated automatically.

Chapter 3
Data Dictionaries

3

Data dictionaries

A data dictionary is a collection of descriptions about a data model or system. LN uses two unique data dictionaries: the Runtime Data Dictionary and the Application Data Dictionary.

This section describes the purpose of the Runtime Data Dictionary, the purpose of the Application Data Dictionary, and how you convert data from the Application Data Dictionary to the Runtime Data Dictionary.

The Runtime Data Dictionary

The Runtime Data Dictionary (RTDD) refers to the files and data that are used to run the system, hence the term “runtime”. The purpose of the RTDD is two-fold:

- Provides configuration information to the technical architecture. This is also called the runtime environment.
- Provides optimized application information when the application is running.

Configuration Information

The RTDD is necessary because the system is configurable in its technical architecture. If this configuration information were stored in a database, the system would not know how to get this information, or from what database to get it. The system would not know the language to present any messages, or the programs it should start to support the system. To address these issues, information is available from the application server’s operating system during the bshell startup procedure.

Optimized Application Information

The application repeatedly requires a lot of information about the application components that are running. While this information could come from a database, because the information may be required frequently, and changes are infrequent, it is faster and more efficient to pre-process this information and store it in a compressed and optimized way. Examples of this type of information are date formats, menus, toolbar texts, menus and forms. Application components like forms, form fields, and labels are stored in a database. To gather all the information from a database when it is needed by a running

system would slow down the system. Therefore, the system uses a convert function to compile this information into an optimized format.

The Application Data Dictionary

While the runtime data dictionary is stored in the application server's operating system and some Enterprise Server tables, the Application Data Dictionary (APDD) is data stored in tables. The APDD data is accessed by sessions.

By having the APDD stored as data and accessible in the sessions, this allows administrators to easily administer the system using sessions, and allows developers to easily develop applications in the system. In this way, the application administration environment and application development environment are fundamentally the same as the application execution environment.

The APDD data is stored in company number 000.

Convert to Runtime Data Dictionary

After the application data dictionary data is maintained in the sessions, the information intended for the RTDD must be extracted from the APDD, reformatted or optimized and made available in the runtime data dictionary. This process is called "Convert to Runtime".

The optimized information is in one of these formats:

- File on the operating system.
- Compressed data in a table.

The categories of making the APDD available at runtime include:

- Directly Available: A conversion step is not necessary because the system will access the data directly from the APDD.
- Convert (to Runtime): This type of conversion takes information out of the APDD, and creates files in the application server's operating system.
- Compile or Dump: Usually performed by programmers, this type of conversion takes information out of tables and script files, and stores them in an optimized and encrypted format, called an object or component, at the operating system level.

How do you know when a convert to runtime is required? This can be a difficult question. The most effective way is to see if the session has a convert to runtime button or a pull down menu item under the **Specific** menu. A convert to runtime is required, for example, for Package Combinations, Tables and Domains.

Other entities require a compile step to make the APDD information available for use. Usually, compile steps do not require signing out of the system (a change to the users start menu is a notable exception). Compilation is required, for example, for:

- Menus
- Labels
- Sessions and Forms

- Reports
- Scripts and libraries

Convert to Runtime Sessions

There is a number of Convert to Runtime sessions that is used during software development. Usually, you can initiate a convert to runtime for an entity from the **Specific** menu in the session that maintains that entity.

When more than one entity requires conversion, or when more than one element in an entity changes, you can use a conversion session that is accessible from the menu browser. For example:

Session	Description
Convert to Runtime Data Dictionary (ttadv5215m000)	The convert to runtime data dictionary in Application Development performs specific processing on the Table Definition and Domain files associated with the Package Combination. The table and domain definition files contain information about the table's fields and data types. If this format changes, a field is added to a table and the table definition changes. However, the physical table that has data needs to adhere to this definition. Because of the requirement that the definition and physical table be in sync with each other, a reconfiguration process performs the changes on the physical table. The system knows what table definitions have changed because the table definitions session adds an indicator entry in a table called Conversion Indicators (ttadv500). When a new table definition replaces an existing table definition, the change is actually pending until a table reconfiguration process is done. A table reconfiguration process is known to the system by an entry in the Reconfiguration Indicators (ttadv501) table.

	The convert to runtime process will look at the indicators table to determine what tables to convert. This session will not convert session data.
Create Runtime Data Dictionary (ttadv5210m000)	This session performs the same task as the **Convert to Runtime Data Dictionary (ttadv5215m000)** session, with these notable differences: ■ The session creates table/domain definition files even if no changes have been made. Therefore, the indicator table is not checked as in the **Convert to Runtime Data Dictionary (ttadv5215m000)** session. ■ The session creates session entries in the runtime data dictionary table (ttadv999). This table is accessed when a session starts up. This session is used after changes affecting the system such as importing data dictionary. The import data dictionary function is able to import application data dictionary components, and is a way to move components from one system, such as a development server, to another server.

Chapter 4
Development Parameters and Authorizations

4

Development parameters and authorizations

To maintain or create software components, a developer requires default development settings and parameters and authorization to at least one package VRC.

Development Parameters

The settings and parameters a developer needs, are defined in a Development Parameters template. Parameters are available for the following:

- Automatic compilation to the run-time data dictionary after changes to forms or menus
- Actions to be performed automatically after the **Copy to Current Package VRC** option
- The parameters that the editor can use to develop software

To create the Development Parameters template, you must use the Development Parameters Template (ttams1150m000) session. On the session's appropriate menu, click **Convert Changes To Runtime DD** to convert the changes to the runtime data dictionary.

The template contains data a group of users share. However, the template is linked to individual users.

To link the template to a user, you must use the User Data (ttaad2500m000) session. Do not forget to convert the changes to the runtime data dictionary. You must log off and log on again to make all changes effective.

For more information, refer to the Infor Web Help.

Note

For every LN user, the data in this template is dumped to LN 's run-time data dictionary in the User Application Parameters (ttadv010) table.

Developer Authorizations

A developer needs authorization for at least one package VRC.

To define developer authorizations for a user, you can do one of the following:

- Link the user to a Developer Authorization template.
- Give authorization for all package VRCs.

Developer Authorization Template

The VRCs a developer is authorized for are defined in a Developer Authorization template.

You can use this template to define some of the authorizations for developers who must customize LN software components.

In this template, you can specify the following:

- The package VRC(s) for which the developer must be authorized to customize and develop software components. This specific package VRC overrides the **Auth for all Package VRCs** check box in the User Data (ttams1100s000) session.
- The languages and modules of the specified package VRC for which the developers must be authorized to customize and develop software components.

To create the developer's authorization template, you must use the Developer Authorization Template (ttams1151m000) session. This session is password-protected. You can only use this session if you have system administrator's rights. Points of attention:

- If you select the **All Modules** and **All Languages** check boxes in this template, the users who are linked to the template are authorized to maintain and develop software components in all modules in LN and in all languages. If the check boxes are cleared, you must specify the specific modules and languages for which the user must be authorized.
- If you select the **Authorization for Components of other Developer** check box in this template, the user is authorized to maintain the software components that are created by other users. This option is helpful, for example, for a senior application developer.
- On the session's appropriate menu, click **Convert Changes To Runtime DD** to convert the changes to the runtime data dictionary.

The template contains data a group of users share. However, the template is linked to individual users.

To link the template to a user, you must use the User Data (ttaad2500m000) session. Do not forget to convert the changes to the runtime data dictionary. You must log off and log on again to make all changes effective.

For more information, refer to the Infor Web Help.

Note

The data in this template is dumped to LN 's run-time data dictionary for each individual LN user to the Version Authorizations by User (ttadv041) table.

Authorization for all package VRCs

Some users, e.g. an application administrator, need authorization for all package VRCs.

To give authorization for all VRCs, start the User Data (ttaad2500m000) session and modify the user data as follows:

- Select the **Auth for all Package VRCs** check box.
- Make sure the **Developer Authorizations** field is empty.

Convert the changes to the runtime data dictionary, and log off and log on again to make the changes effective.

Note

If you link the user to a Developer Authorization Template, the user will no longer be authorized for all package VRCs, but only for the VRCs defined in the template.

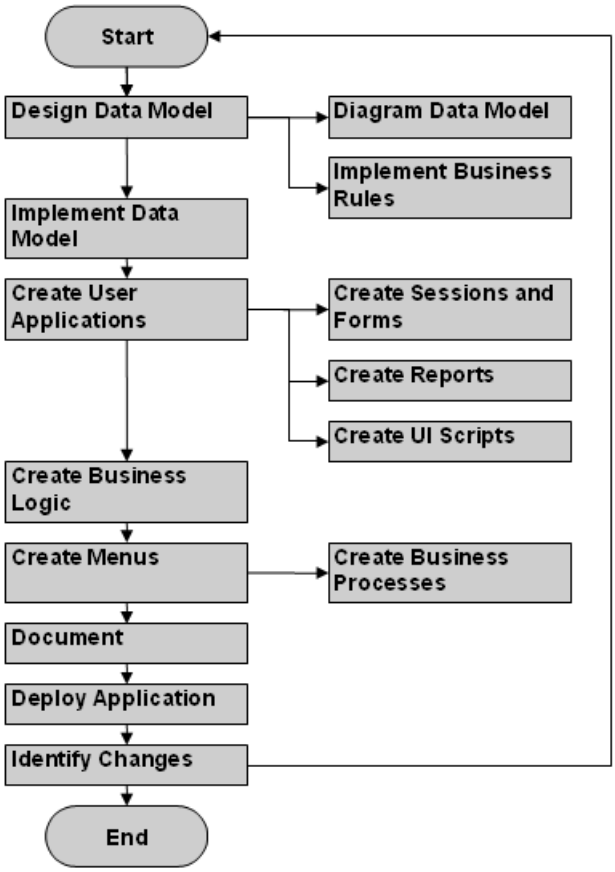
Chapter 5
Software Development Procedures

5

Development process

Developers use the Infor LN Development Tools to develop the software components that make up the application.

There is a logical progression in developing the components to create an application. The development process explains this progression.



The Flow: Development Process

The development process steps are:

Design Data Model

The data model design begins with identifying the entities within the data model, the attributes of the entities and the primary keys of the entities. The elements of the data model are created in Development Tools.

The data model is normalized so that each entity has a unique primary key, all necessary relations are defined, and data redundancy is reduced.

The data model design process will result in a data model consisting of:

- Table Definitions that define the entities in the data model.
- Fields of tables that define the attributes of the tables. Each field will also have a domain, which is a definition of the data type, format and field validation rules. Labels are initially created based on the field description.
- Foreign key relationships to other tables.

- Indices that can be primary keys and alternate keys.

For details, refer to *To create a data model (p. 66)* .

Diagram Data Model

The diagram of the data model can be created at this point. The foreign key relationships are used to show the relationships between tables.

For details, refer to *To create a data model (p. 66)* .

Implement Business Rules

Business rules are implemented in the Data Access Layer. The rules can be based on the fields, or the record. These rules form the core of the business logic that affects the user interface of the application.

Property hooks or field hooks operate at a field level and can be used to determine if the field should be displayed, verify valid values, calculate default values and update field values.

Object hooks operate on the row or record as a whole unit. Object hooks determine if an operation on a row is allowed, and if additional operations should be performed on the row or an operation should be performed on another table.

Business Methods can be implemented in the Data Access Layer. Business Methods are operations that process a row or set or rows within a table. In order to expose business logic to external applications, the preferred method of implementing this type of processing logic is within a library. The interface to the library is implemented through Infor LN Business Studio.

For details, refer to *DAL (p. 17)* .

Implement Data Model

The data model is implemented by creating tables for a company. The tables for a company are created based on the table definitions that are active for the company's package combination.

For details, refer to *To create a data model (p. 66)* .

Create User Applications

The user applications are created. The main focus of this phase of component development is to create user-oriented panels that list, create, modify and delete data based on the table definitions.

Create Sessions and Forms

Sessions are the tasks that execute. Forms are the user interface part of the sessions. Forms specify the fields and labels that allow users to view and manipulate data.

For details, refer to *To create sessions (p. 73)* .

Create Reports

Reports format and output data from a session. Reports can contain logic such as subtotals, counts and totals based on the data that is processed.

For details, refer to *To create reports (p. 96)* .

Create UI Scripts

User Interface (UI) Scripts assist the developer in providing user interface logic for a specific session. The object and property hooks/field hooks in the DAL of the table will be active when the session executes. The UI script provides the additional opportunity to the developer to specify logic. UI Scripts should be session specific.

Examples of UI Script actions are:

1. Import values from another session.
2. Perform a query for a report.
3. Specify what happens when a button is clicked.

For details, refer to *UI scripts (p. 50)* .

Create Business Logic

Business Logic is operations that process a row or set or rows within a table. Business Logic is not session specific, and is implemented within a library. A session or hook can initiate the business logic. An interface to the business logic is implemented in the Infor LN Business Studio. This interface allows external applications to execute the business logic.

Business logic can be implemented as Association Logic or Aggregation Logic.

- Association logic is implementing a related table. For example, the order session requires a business partner, and the logic depends on the business partner data.
- Aggregation logic is implementing a parent-child table relationship. For example, an Order Header process can update all the Order Lines for the order.

The Association and Aggregation Logic are programmed in a Dynamic Link Library (DLL). The interface to the library is implemented via Infor LN Business Studio.

For details, refer to *Libraries (p. 31)* .

Create Menus

After the sessions, forms and business logic is complete, the user needs a way to start the sessions. A menu is created with options for starting sessions or other menus.

For details, refer to *Menus (p. 32)* .

Create Business Processes

Sessions form an ordered procedure for completing a business task. This procedure is implemented as a Business Process in the Enterprise Modeler application.

For details, refer to the Enterprise Modeler documentation.

Document

Developers have a number of tools within the application development environment to document their work. Developers can create technical documentation at the component level based on the component, the release, or implement customization projects to identify components created for a specific project. User context driven help documentation can be created for deployment in the Web Help application.

For details, refer to the Infor Web Help.

Deploy Application

Exporting the software components from the development environment and moving the components to a production server deploy the application. Components are imported to a production server.

For details, refer to *Export and import procedure (p. 145)* .

Identify Changes

As businesses change, and workflow processes adapt and improve, application requirements change over time. As changes are identified, application components can be added, changed or expired to suit the business requirements.

To develop software components

You develop software components using a variety of tools within Development Tools. To develop these components you use a common procedure to identify, establish version control for, check-out, change, compile and execute components in your development activities.

Components are VRC Controlled. Since you are not able to change standard components, you can either

- Create a new component.
- Copy components to a VRC and make changes to the copy, not to the original component.
- Copy a Component to a new Component.

Note

When SCM is active, you must check out a component before you can change it. You must check in the component when it is finished. For details, refer to *To use the Software Configuration Management system (SCM) (p. 138)* and to *To use the SCM check-in and check-out procedure (p. 140)* .

Create new component

To create a new component, insert a record in the component development session. For example: to create a new session, insert a new record in the **Sessions (ttadv2500m000)** session.

Copy component to make changes

To copy a component to make changes, you copy the component to a Current Package VRC. You keep the component code in this process.

To copy a component to a Current Package VRC:

1. Start a component editing session.
2. On the appropriate menu, select **Change Current Package VRC of User**. Ensure your current Package VRC is set to the correct Package and VRC.
3. Select the component you wish to copy from.
4. On the appropriate menu, select **Copy to Current P-VRC**.

   The component is in your current Package VRC and can be edited.

Alternatively, you can copy components in bulk from one Package VRC to another. You can use the **Copy Software Components to New Package VRC (ttadv6265m000)** session to copy components from one Package VRC to another.

Copy component to new component

There are instances in the development process when you wish to re-use a component in creating a new component.

To copy an existing component to a new component:

1. Start a component development session.
2. Use the **Duplicate** command to duplicate the component. Specify a new component code.

   The new component is in your current Package VRC.

To create a data model

This topic describes the procedure to create a data model.

Important!

The sections below describe the sessions that you must use to complete the procedure. However, they do not describe all details of these sessions. For detailed session information, refer to the Infor Web Help.

Procedure aim

LN uses a database system to store the data required by the application. Data is stored in tables in a relational database management system (RDBMS). The data dictionary describes the data model of the data required in the application.

You use Development Tools to define the data dictionary and create:

- Domains
- Table Definitions, including fields, labels, foreign key relationships and indices
- Tables to store data, for one or more companies, based on the table definitions

You can use the **Entity Relationship Diagrams (tgerm1500m000)** session in the Enterprise Modeler to create a graphic design of the relational data model structure. The diagram shows a (multilevel) structure that consists of entity types and entity type relationships. For details, refer to the Infor Web Help.

Note

The LN data dictionary is an active data dictionary. This means that sessions that use a table will discover the fields, domains, relationships, indices and other information about a table when the table is used at runtime, not at compile time. This feature allows you to change data dictionary items without recompiling code.

Procedure result and prerequisites

Result

A data model for an LN application.

Prerequisites

To create a data model you need:

- A development package VRC in which you can create the domains and table definitions.
- Developer authorization for this VRC.

Procedure summary

The following list shows the procedure steps and the corresponding sessions:

1. Create domains - **Domains (ttadv4500m000)**

2. Create table definitions, including fields, labels, foreign key relationships and indices - **Table Definitions (ttadv4520m000)**
3. Convert the domains and table definitions to the runtime data dictionary - **Convert to Runtime Data Dictionary (ttadv5215m000)**
4. Create tables to store data - **Create Tables (ttaad4230m000)**

Procedure details

Step 1: Create domains

To create a domain:

1. Start the **Domains (ttadv4500m000)** session.
2. Add a row. Enter the data type, length, format and field validation rules for the domain in the **Domains (ttadv4100m000)** details session.
3. Perform this step only if the data type is Enumerated or Set:
   - In the **Domains (ttadv4500m000)** overview session, select the domain and on the appropriate menu, click **Enum/Set Data**. The **Enum/Set Constants (ttadv4501m000)** session starts. Enter the values for the constants, names and descriptions.
   - Note: The constant description should be no longer than the defined length of the domain. If a description is longer than the domain length, you can increase the domain length.
4. Save the new data and close the sessions.

Step 2: Create table definitions

To create a table definition:

1. Start the **Table Definitions (ttadv4520m000) session**.
2. Click **New** to start the **Table Definitions (ttadv4120s000)** details session. Enter the basic table information, such as the table code and description. Save the changes and return to the overview session..
3. Define the table fields:
   - Open the new table. The **Table Definitions (ttadv4520m100)** session will open. This session is a multi-main table session showing the Table Fields, Indices and Program Scripts / Libraries for the table.
   - In the **Table Fields** tab, click **New Table Field**. The **Table Fields (ttadv4122s000)** details session starts. Enter the table field information. Specify, among other things, a field code, a domain, a label, and (optionally) a reference to another table. For details, refer to the session's online help.
   - When finished, save the changes and return to the **Table Definitions (ttadv4520m100)** multi-main table session.
4. Define one or more indices:
   - In the **Indices** tab, click **New Index**. The **Table Indices (ttadv4121s000)** session starts.

- Add a row. Enter the index information, such as the index number, description and the index parts (table fields).

5. Save the new data and close the sessions.

Points of attention:

Labels	As you enter table fields, labels will be created for the table field descriptions. You can create a new labels, or use existing labels. Using existing labels provides an easy method to adapt your companies terminology and aid translation activities. After labels are created, you must compile the labels so that users can use the labels as they execute the system. You should compile labels after you create enumerate domains, tables, sessions and forms, reports, and other components. To compile labels, use the **Compile Labels (ttadv1243m000)** session.
Reference tables	When you specify the fields for a table, you can define references to other tables. For this reason, it is helpful to create tables, that will be referred to by other tables, first.
Combined fields	Table fields can refer to a primary key of another table. In this way, the references are built into the data dictionary. To define a reference to a table whose primary key contains more than one field, you must use combined fields. Combined fields require that you define the combined field in the reference table and in the referring table. Combined fields are only used in defining tables; actual fields are not created within the database. To create a combined field: **1.** Create a new table field and, in the field properties, select the **Combined Field** attribute. **2.** Click **Child Fields** to start the **Child Fields by Combined Field (ttadv4523m000)** session. **3.** Enter the table fields that make up the combined field.

4. Use the combined field as the first index of the reference table.

In the field properties of the combined field in the referring table, you can now define a reference to the reference table.

Example:

A purchase order line for an item can have more than one arrival at the warehouse. For this reason, the table design requires that an arrival table is a child of the purchase order line table. The primary key of the purchase order line table can be viewed as order number and position number. The arrivals table must have a field that can refer to order number and position number. Therefore, a combined field is used. The combined field “cmba” is created in both tables, with the order number and position number as the child fields. In the field properties of the "cmba" field in the arrivals table, a reference to the purchase order line table is defined.

Step 3: Convert the domains and table definitions to the runtime data dictionary

The table definitions are maintained in the application data dictionary. You must make the new domains and table definitions available to the runtime environment. This process is called Convert to Runtime.

A convert to runtime is performed for tables, domains and sessions based on the package combination. The process will search the Package VRCs of the packages in the package combination and create the table and domain runtime files in the runtime data dictionary directory on the application server. This directory is established when you create the package combination.

To convert to runtime:

1. Start the **Convert to Runtime Data Dictionary (ttadv5215m000)** session.
2. Select the **Domains** and **Tables** check boxes. Specify the appropriate ranges of package combinations, packages and tables.
3. Click **Convert**.

Step 4: Create tables to store data

Tables store data. In order to store data, the table definition is used to create a table in the underlying database for a specific company number. In most companies, a person designated as the system administrator or database administrator will create tables.

Before you create tables, the following pre-requisites must be in place:

- The data dictionary must be converted to runtime for the package combination.

- Your LN user id must use the package combination of the company.
- A company must be created with your package combination.
- The database administration and storage requirements must be setup to create tables in the appropriate database and storage area.

To create tables:

1. Start the **Create Tables (ttaad4230m000)** session. Optionally, you can start this session from the appropriate menu in the **Table Definitions** session.
2. Specify the appropriate ranges of companies, packages and tables.
3. Click **Create**.

To reconfigure tables

This topic describes the procedure to reconfigure tables.

Important!

The sections below describe the sessions that you must use to complete the procedure. However, they do not describe all details of these sessions. For detailed session information, refer to the Infor Web Help.

Procedure aim

When you change the table definitions, or the corresponding domains, for tables that already exist in the database, you must reconfigure the tables according to the new table and domain definitions.

A table configuration process will perform these actions:

- Unload data that is in the existing table, for all companies in the package combination
- Drop the existing table, for all companies in the package combination
- Create the table according to the new table and domain definitions
- Re-load the data into the table

This is a process that you will want to control when and who performs the reconfiguration activity. You can separate the convert process and the reconfigure process.

Procedure result and prerequisites

Result

The structure of the tables in the database matches the new table and domain definitions.

Prerequisites

A table reconfiguration requires exclusive access to the table, because the system will drop the table.

Procedure summary

To execute the convert and reconfigure process as 2 separate steps:

1. Convert the data dictionary information to runtime - **Convert to Runtime Data Dictionary (ttadv5215m000)**
2. Reconfigure the tables in conjunction with the DBA or system administrator - **Convert to Runtime Data Dictionary (ttadv5215m000)**

Note

You must log off and log on again to use the new tables.

Procedure details

Step 1: Convert the data dictionary information to runtime

Run the **Convert to Runtime Data Dictionary (ttadv5215m000)** session with the following selection settings:

Check Box	Selected
Domains	Yes
Tables	Yes
Reconfigure Tables	No

Step 2: Reconfigure the tables

To reconfigure the tables in conjunction with the DBA or system administrator, run the **Convert to Runtime Data Dictionary (ttadv5215m000)** session with the following selection settings:

Check Box	Selected
Domains	No
Tables	No
Reconfigure Tables	Yes

To create sessions

This topic describes the procedure to create sessions.

Important!

The sections below describe the sessions that you must use to complete the procedure. However, they do not describe all details of these sessions. For detailed session information, refer to the Infor Web Help.

Procedure aim

The aim of this procedure is to create sessions for your LN application. Sessions are the main applications that a user will use. Sessions are used to present data, edit data and process data.

Session development involves defining the attributes of a session, editing the form layout, and specifying the specific commands and indices.

Important!

You will add the user interface logic and processing logic in later steps.

Procedure result and prerequisites

Result

Sessions that perform activities in your LN application.

Prerequisites

To create sessions you need:

- A development package VRC in which you can create the sessions.
- Developer authorization for the development VRC.
- A data model with tables on which the sessions will operate.

- The Dynamic Form Editor (DFE) must be installed on your PC.

Procedure summary

The following list shows the procedure steps and the corresponding sessions or programs:

1. Add a session - Sessions (ttadv2500m000)
2. Create and edit the form - **Dynamic Form Editor (DFE)**
3. Specify session specific commands and buttons - Form Commands (ttadv3518m000)/ Standard Commands (ttadv3110s000)/ **Dynamic Form Editor (DFE)**
4. Specify indices - Available Indices by Session (ttadv2101s000)
5. Add the user interface logic and processing logic. You can define this logic in the session's UI script, in functions, DLLs and DALs.

Procedure details

Step 1: Add a session

To create a session you must add a session code in your development package VRC:

1. Start the Sessions (ttadv2500m000) session.
2. Click **New**. The Sessions (ttadv2100s000) details session starts.
3. In the **Attributes** tab, specify the session attributes, for example: the session code, the session description and the session type. For details, refer to the session's online help.
4. Complete the session on the **Form** tab. Here you must specify, among other things, the number of occurrences and the default button.
5. Click **Save**.

Step 2: Create and edit the form

To create and edit forms, you must use the Infor LN Dynamic Form Editor. The form editor allows you to specify the available session types for the form (overview and detail), the field groups, labels and fields that are used to create the form.

To start the Dynamic Form Editor:

1. In the Sessions (ttadv2500m000) session, select the session for which you want to create or edit the form.
2. Click **Edit Form ...**. The Dynamic Form Editor starts.

You can now perform basic form editing tasks to quickly produce a working session prototype. For details about creating a new basic form for a session, refer to *To create forms (p. 75)* .

After you have a basic form setup and operational, you can change the form layout to achieve the final result. For details, refer to *To edit forms (p. 78)* .

Step 3: Specify session specific commands and buttons

In addition to labels, fields and field groups, you can enhance the user interface by creating additional elements, such as zoom buttons, standard commands, and form commands, on the form.

For details, refer to *To create commands and buttons for a session (p. 85)* .

Step 4: Specify indices

Specify the indices that must be available for the session.

For details, refer to *To specify indices for a session (p. 87)* .

Step 5: Add the user interface logic and processing logic

You can define this logic in the session's UI script, in functions, DLLs and DALs. For details, refer to the following sections:

- *UI scripts (p. 50)*
- *Functions (p. 28)*
- *Libraries (p. 31)*
- *DAL (p. 17)*

To create forms

This topic describes the basic form editing tasks that allow you to quickly produce a working session prototype. From the basic setup, you will be able to change the form layout to achieve the final result.

Important!

The sections below describe the sessions that you must use to complete the procedure. However, they do not describe all details of these sessions. For detailed session information, refer to the Infor Web Help.

Procedure aim

The aim of this procedure is to create forms for your LN sessions. Forms are the user interface part of the sessions. Forms are used to interact with the user by presenting data and actions that can be performed on that data.

To create and edit forms, you must use the Infor LN Dynamic Form Editor (DFE). The form editor allows you to specify the available session types for the form (overview and detail), the field groups, labels and fields that are used to create the form.

Procedure result and prerequisites

Result

Basic forms for your (new) LN session prototypes.

Prerequisites

To create forms you need:

- Sessions for which you can create forms
- Developer authorization for the development VRC where the sessions are stored.
- The Dynamic Form Editor (DFE) must be installed on your PC.

Procedure summary

The following list shows the procedure steps and the corresponding sessions or programs:

1. Start the Editor - Sessions (ttadv2500m000)
2. Specify Available Session Types - **Dynamic Form Editor**
3. Add Fields - **Dynamic Form Editor**
4. Save the Form - **Dynamic Form Editor**
5. Show Session Prototype - **Dynamic Form Editor**
6. Exit the Editor - **Dynamic Form Editor**
7. Compile the Form - Sessions (ttadv2500m000)
8. Start the Session - Sessions (ttadv2500m000)

Procedure details

Step 1: Start the Editor

To start the Dynamic Form Editor:

1. Start the Sessions (ttadv2500m000) session.
2. Select the session for which you want to create the form.
3. Click **Edit Form ...**. The Dynamic Form Editor starts.

Step 2: Specify Available Session Types

When you create a form, you must specify if the session is available as an overview session, a details session or both.

To specify the available session types:

1. Open the properties for the session. Click **Properties**. The **Session Properties** dialog box is started.

2. Select the available session types: select **Details Session**, **Overview Session**, or both. Additional attributes control the placement of field groups. For details about the fields in the dialog, refer to the Dynamic Form Editor's online help.

Step 3: Add Fields

Add table fields: on the **Insert** menu, click **All Table Fields**. The fields of the session's main table are added to the form.

Alternatively, to select individual fields, you can click **TableFields** on the **Insert** menu.

The fields and their associated label will display within field groups.

Step 4: Save the Form

To save the form, click **Save**.

Step 5: Show Session Prototype

You can view the effect of your changes from within the form editor: On the **File** menu, click **Show Overview Session**. The overview session starts. If you wish to view the details session, click **Show Details Session**.

Alternatively you can exit the editor, and then compile the form and start the session from the Sessions (ttadv2500m000) session.

Step 6: Exit the Editor

To exit the editor, click **Save and Exit**.

Step 7: Compile the Form

You must compile the form to create a form dump within the runtime data dictionary.

To compile the form:

1. In the Sessions (ttadv2500m000) session, select the session to which the form belongs.
2. Click **Compile**.

Step 8: Start the Session

You can start the session from within the developer’s workbench.

To start the session:

1. In the Sessions (ttadv2500m000) session, select the session.
2. Click **Compile**.

To edit forms

After you have a basic form setup and operational, you can change the form layout to achieve the final result. Form edit tasks are performed in the context of the basic form edit procedure.

Many of the typical form editing operations are outlined here, grouped by general area:

- *To edit the form content (p. 78)* describes the labels and fields that are displayed
- *To edit the form structure (p. 81)* describes the grouping of fields and the tabs that are displayed
- *To edit the form sequence (p. 84)* describes the order and relative placement of labels and fields

Important!

For detailed information on the Dynamic Form Editor, refer to the Dynamic Form Editor's online help.

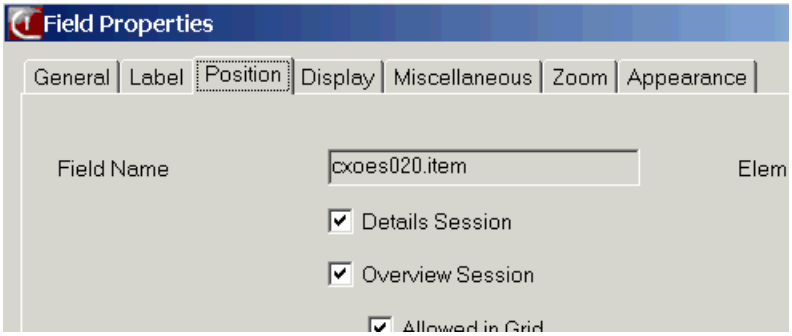
To edit the form content

Form layout changes affect the labels and fields that are presented on the form.

To specify fields for overview and detail sessions

You can specify, per individual field, whether the field must be displayed in the overview session, the details session, or in both sessions:

1. Right-click a form field, and on the shortcut menu click **Properties**. The **Field Properties** dialog starts.
2. On the **Position** tab, select whether the field is displayed in the Details session or in the Overview session or in both sessions.



Select Details and Overview Session

3. To move to the other fields, click the navigation buttons in the dialog.

You can view the effect of your changes from within the form editor: On the **File** menu, click **Show Overview Session**. The overview session starts. If you wish to view the details session, click **Show Details Session**.

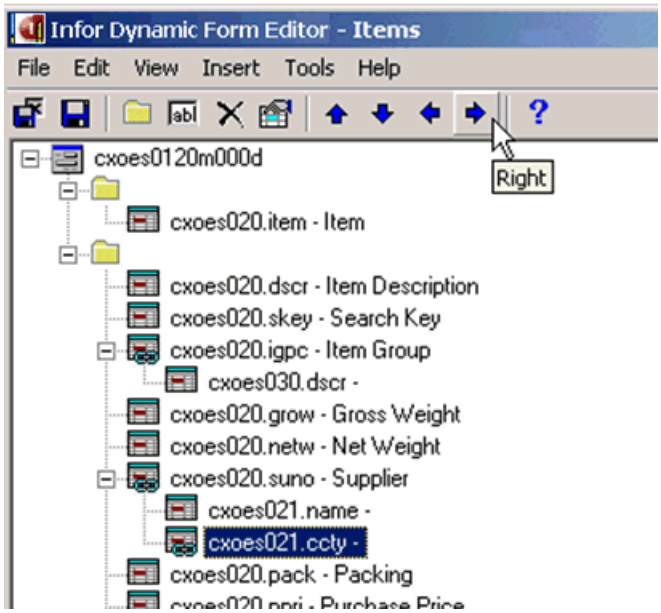
To add a field

You can add a field to a form based on these conditions:

- The field is defined in the main table of the session.
- The field is defined in a reference table of the main table.
- The field is defined in the program (UI) script.

To add a field to the form:

1. Select the field that is before the field you wish to add. On the **Insert** menu, click **Field**. A new field, with a label icon, is added.
2. Edit the field properties. Define the field name and attributes in the **General** tab. If the new field is a table field, you can select it from a list:
   - Click the zoom button next to the **Field Name** field. The Table Fields (ttadv4529m000) session starts.
   - Select a table field. If the field is not in the main table, use the **Find** button to select a different table.
3. Define additional properties for the field. If a field is a reference only field, you can clear the **Label Code** field in the **Label** tab to remove the label.
4. Depending on the importance of the field, you can select if you want the field to be displayed in the overview session, the details session or both.
5. The new field is displayed in the Dynamic Form Editor. An indented field indicates that the field is a “child” field. Child fields display next and to the right of parent fields, at the same indentation level as fields that have the same parent. To change the indentation of a field, use the **Right** and **Left** buttons in the editor's toolbar.
6. Save the form.



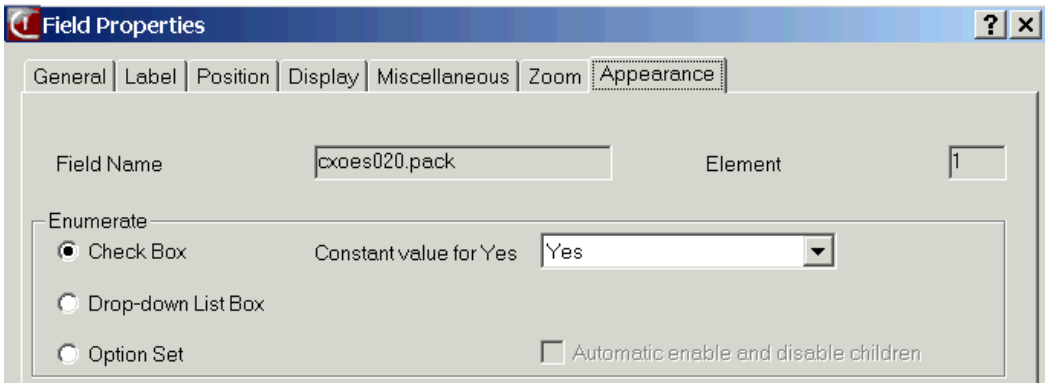
Indent Field

Checkbox Appearance

An enumerated domain that has two values can be viewed as a checkbox. The default appearance is a drop-down list box.

To change the appearance of an enumerated domain:

1. In the field properties for the field with the enumerated domain, select the **Appearance** tab.
2. Select **Check Box**.



Enumerate Check Box

When you show the details session, the result shows the field with the enumerated domain as a checkbox.

Alternatively, you can change the appearance of the domain for any form for an enumerated domain. To do so, insert the domain in the Domains to be Displayed as Checkbox (ttgfd4520m000) session.

To edit the form structure

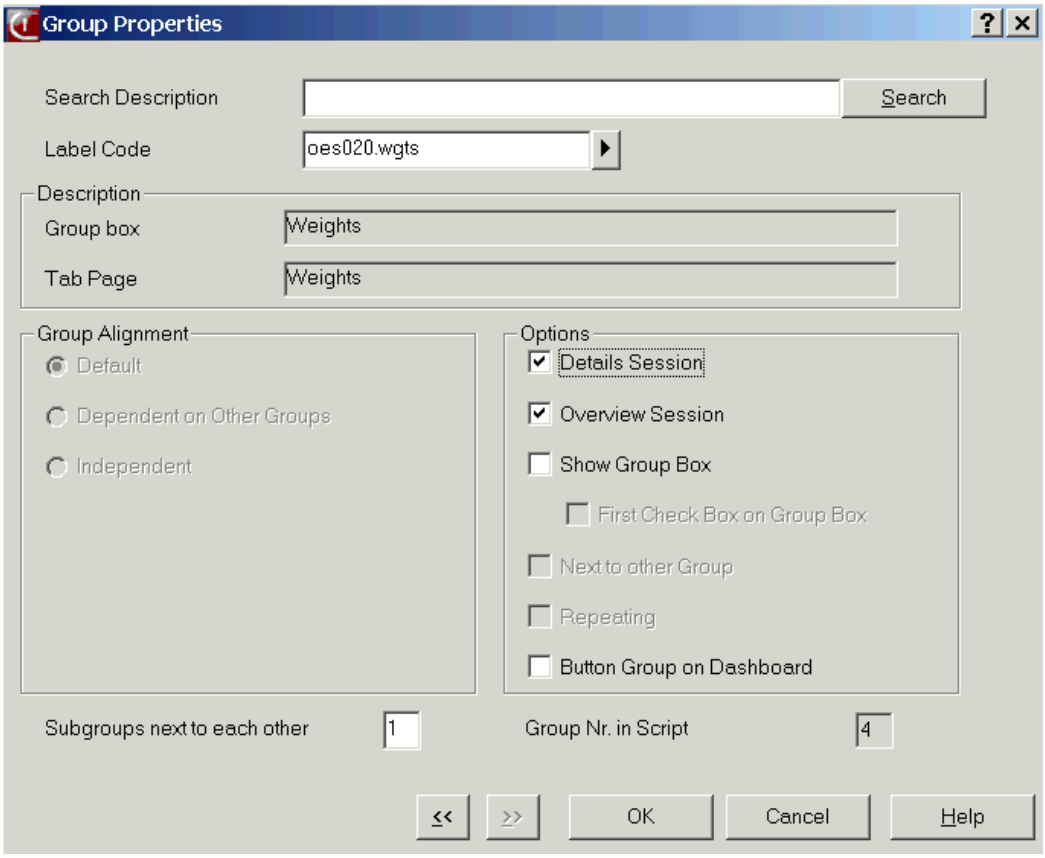
The structure of the form affects the grouping of fields and tab labels.

To add a field Group

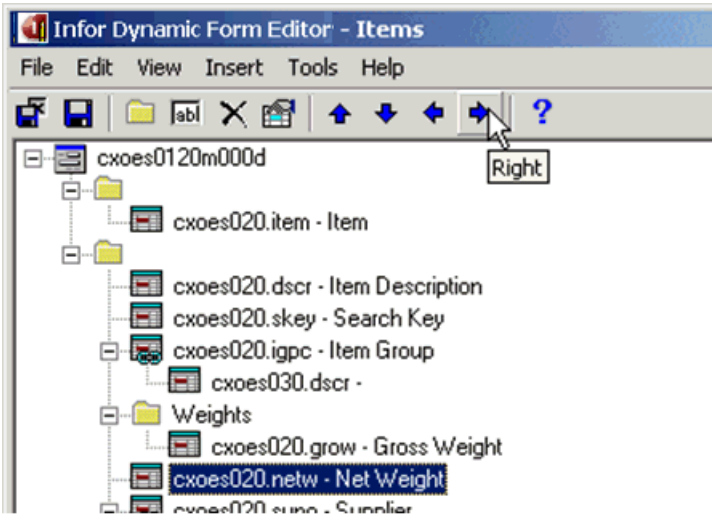
A field group has a title. The fields in the field group are shown within an indented list.

To create a field group:

1. On the **Insert** menu, click **Group**.
2. A new group, with a folder icon, is added. The group is added immediately below the selected field. If you need to, use the **Up** and **Down** arrow buttons to position the field group.
3. Click **Properties** to start the **Group Properties** dialog. Enter a label for the new group, and specify the other group properties. For details about the fields in the dialog, refer to the Dynamic Form Editor's online help.
4. Click **OK**. The field group label will change in the form editor.
5. Use the arrow buttons to move the fields into the group.



Group properties



Move fields into group

To align groups and fields

Field group alignment and field alignment are important in determining the layout of the form. Because the form generation is dynamic, this alignment is the main way to control the placement of labels and fields on the form.

Groups and field placement depends on the indentation level of the field groups, subgroups and fields. The indentation level of a group determines if the group is a tab, or a field group (sub-group) within a tab. Sub-groups can be aligned next to each other.

The indentation level of the field determines if the field is placed on a new line (on the left side of the form), or as a field to the right of a previous field. The alignment of fields in columns depends on the group alignment selection of the field group that the fields belong to.

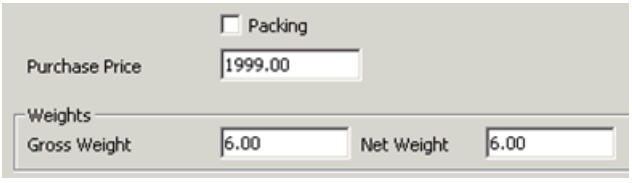
To adjust the indentation level of field groups and fields, use the left and right arrow buttons.

Example- Field Alignment for Items

The Items form has many fields. In the Weights field group, you want the net weight to be placed to the right of the gross weight field. You change the indentation level for the net weight field to be one level to the right.

Field Alignment for Items

This diagram shows the result:



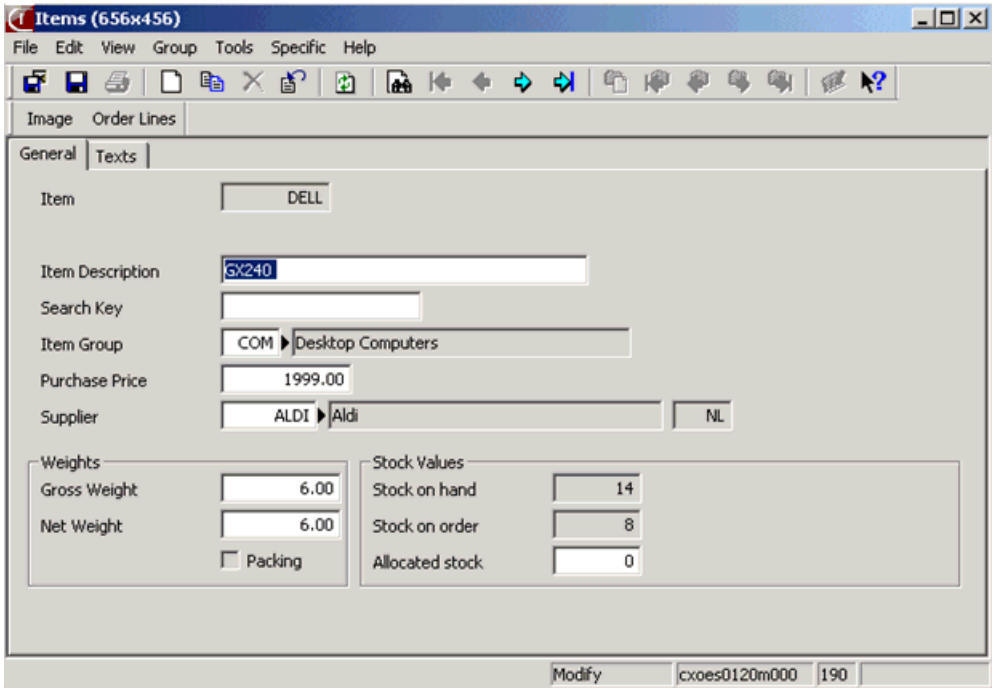
Field Alignment Result

Example- Group Alignment

The General group/tab in the Items form contains two sub-groups: the Weights group and the Stock Values group. To place these sub-groups next to each other:

1. Edit the properties of the General group: specify 2 in the **Subgroups next to each other** field.
2. In the properties of the Weights group, select **Next to other Group**.
3. In the properties of the Stock Values group, select **Next to other Group**.

This diagram shows the result:



Group Alignment Result

The tab sequence for fields in subgroups allows you to tab through the fields in the first subgroup before tabbing to fields in the second sub-group.

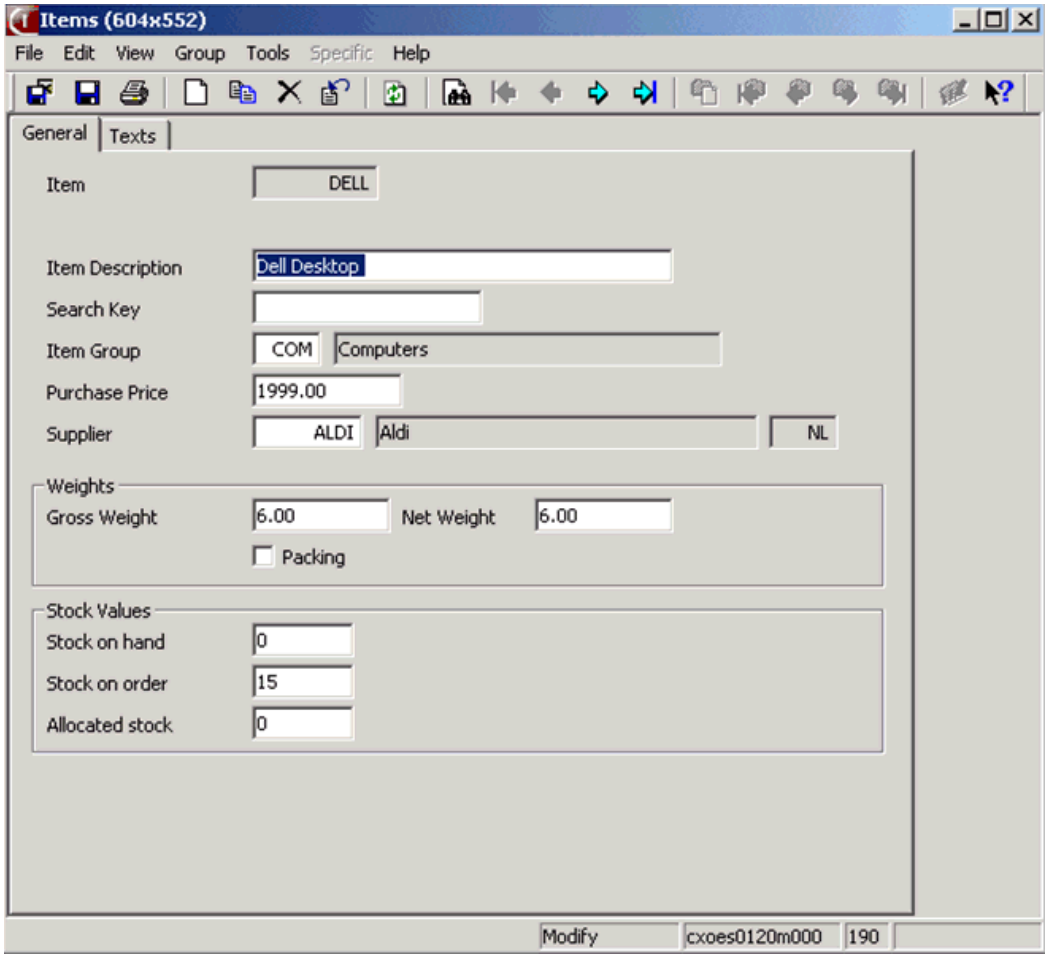
To edit the form sequence

The sequence of the form allows you to present the data relative to its importance to the organization. You can sequence field groups or fields. To change the sequence of a field group or field, select the item and use the **Up** and **Down** buttons in the editor's toolbar to move the item. Alternatively, you can drag and drop the item.

Example- Field Sequence for Items

You move the purchase price field above the supplier information, and the packing field to the Weights field group.

The result shows the desired sequence of the fields.



Field Sequence Result

For more information on group and field alignment, refer to "Alignment" in the Dynamic Form Editor's online help.

To create commands and buttons for a session

You can enhance the user interface of a session by creating additional elements on the form. Some of these elements are:

Element	Description

Zoom buttons	allow you to view and select a record from a reference table.
Standard Commands	control the active toolbar icons and pull-down menu options.
Form Commands	allow you to specify options in the appropriate menu.
Form Buttons	allow a user to invoke a command by pressing a button.
Field Buttons	allow a user to perform a task related to a field.
Group Buttons	allow a user to perform a task related to a field group.
Secondary Toolbars	allow form commands to start by clicking icons in a second toolbar. You use the create.extra.toolbar function within the program script to create the toolbar. For details, refer to the *LN Programmer's Guide*.

To add a zoom button

A zoom button appears next to a field. It is usually used to start an overview session for a reference table, where the user is able to select a value and return to the session, and the selected value is shown as the field value.

Example- Zoom on Item Group

In the Items session, the item group field can have a zoom button to the Item Groups session.

To add a zoom button to the form:

1. In the Dynamic Form Editor, edit the field properties for the form field.
2. Use the **Zoom** tab to specify action when the zoom button is pressed.

For details, refer to the Dynamic Form Editor's online help.

To edit standard commands

Standard commands control the active toolbar icons and pull-down menu options for a session.

To edit the standard commands:

1. In the Dynamic Form Editor, click **Standard Commands** on the **Tools** menu. Alternatively, click **Standard Commands** on the appropriate menu in the Sessions (ttadv2500m000) session. The Standard Commands (ttadv3110s000) session starts.
2. Select the available standard commands from the defined lists of commands. If an option is selected, the option will appear on the session’s toolbar and menus.

3. Click **OK**.

Add Form Command

A form command is a session specific command. The user can invoke a form command using a command in the appropriate menu, a form button, a field button or a group button. You create all these elements in the same way, but use different command types to determine how the command is presented to the user.

To create a form command:

1. In the Dynamic Form Editor, click **Form Commands** on the **Tools** menu. Alternatively, click **Form Commands** on the appropriate menu in the Sessions (ttadv2500m000) session. The Form Commands (ttadv3518m000) session starts.
2. Click **New**. The Form Commands (ttadv3118s000) details session starts.
3. Enter the values for the command. For details, refer to the session's online help.
4. Click **Save and Close**.

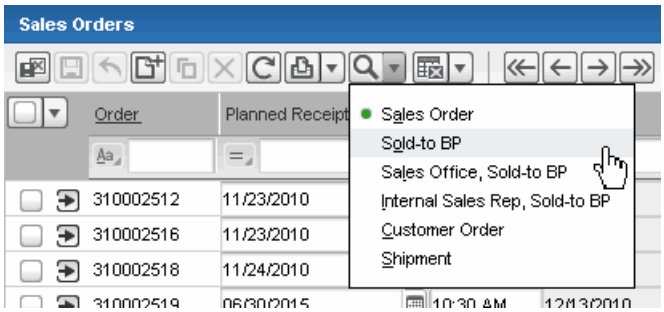
Form commands are typically added in conjunction with other sessions or with program script elements, such as functions.

To specify indices for a session

Session indices allow users to use the defined indices for a table within the session. In the user interface, indices help users sort and categorize groups of records within a table. Web UI users can change the index using the **Sort by** option in a session's **View** menu. Infor Ming.le users can change the index using the down arrow next to the Search button.

Example- Indices for the Sales Orders table

The Sales Orders table has multiple indices. When these indices are active for the session, the user can sort by the indices.

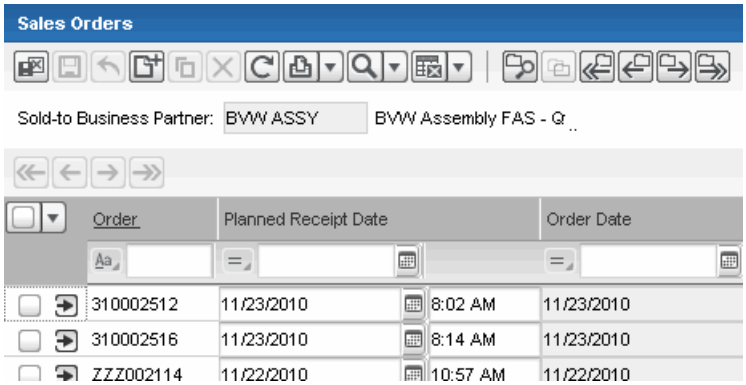


Index Change in Infor Ming.le

The indices of the table can be active for a session. An index that has more than one field can be defined to show the first fields as a view field. A view field allows users to select only those rows that contain the view field value within their index value.

Example- View field

The Sold-to BP index for the Sales Orders table has two fields: Sold-to Business Partner and Order. When the user changes the index to the Sold-to BP index, the form layout changes: the Sold-to Business Partner field is displayed as a view field above the grid. This automatic change of the form layout is called Dynamic Index Switching.



Index Change Result

For more information, see *To implement dynamic index switching* in the Infor Web Help.

To specify the indices for a session

To specify the indices for a session:

1. In the Dynamic Form Editor, click the **Indices** command on the **Tools** menu. Alternatively, click **Indices...** on the appropriate menu in the Sessions (ttadv2500m000) session. The Available Indices by Session (ttadv2101s000) session starts.
2. Select the indices that you wish to have available. For each index, specify the order that the index will show in the **Sort by** window, and the number of fields in the view field.
3. Click **OK**.

To create a form for a print session

When you create a print session, you must also create the form for the session.

The form is used to enter selection ranges and processing options.

- Selection ranges specify a lower and upper range. Ranges are typically used for key fields or alternate key fields. The fields that are inserted on the form need to correspond to the program script variables that are used in the main SQL statement that selects rows from the database. You can extend the SQL statement to add additional range variables.
- Processing options are used to specify additional conditions for selecting rows, or for specifying the sort order. Some processing options will change the report group number so that a different report is selected automatically for the user.

Important!

For detailed information on the Dynamic Form Editor, refer to the Dynamic Form Editor's online help.

Review Program Script

Before you create the form, it is helpful to review the program script. This is particularly true if you created the program script when you created the print session. The program script will contain the default selection ranges.

You may have to extend the generated program script to query references to parent tables.

See the following program script for an example:

```
|*******************************************************
|* cxoes0421  0  VRC B61U a  dv00
|*
|* Development User
|* 2005-02-07
|*******************************************************
|* Main table cxoes021 Suppliers, Form Type 4
|*******************************************************

|***************************** declaration section *******
declaration:

  table   tcxoes021 | Suppliers
  table   tcxoes031 | Countries

   extern  domain  cxsuno           suno.f    fixed
   extern  domain  cxsuno           suno.t    fixed

|***************************** program section ************

|***************************** group section **************
group.1:
init.group:
   get.screen.defaults()

|***************************** choice section *************
choice.cont.process:
on.choice:
   execute(print.data)

choice.print.data:
on.choice:
   if rprt_open() then
```

```
       read.main.table()
       rprt_close()
   else
       choice.again()
   endif

|***************************** field section ***************
field.suno.f:
when.field.changes:
   suno.t = suno.f

|***************************** function section ************
functions:

function read.main.table()
{
    select cxoes021.*, cxoes031.*
    from cxoes021, cxoes031
    where   cxoes021._index1 inrange {:suno.f}
                                     and {:suno.t}
    and     cxoes021.ccty refers to cxoes031
    order by cxoes021._index1
   selectdo
       rprt_send()
   endselect
}
```

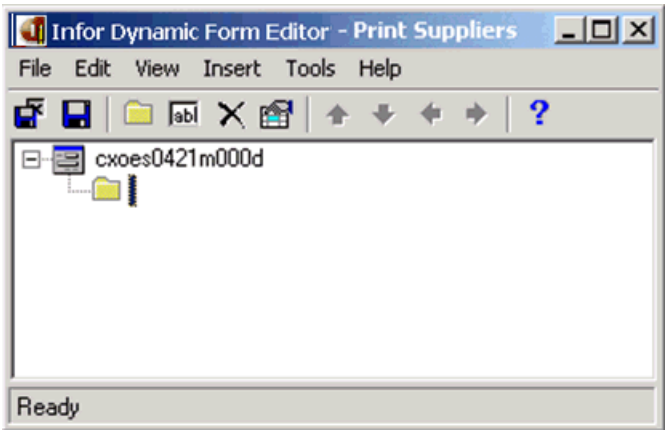
Edit the Form

Create the form for the print session. While you can create a form in a variety of ways, this procedure shows you how a typical form is setup for a print session.

Step 1: Insert a Group

To insert a group in the form:

1. In the Sessions (ttadv2500m000) session, select the session for which you want to create the form.
2. Click **Edit Form ...**. The Dynamic Form Editor starts.
3. Insert a group for selection ranges.



Insert Group

Step 2: Insert Labels

Insert a label only field, with the label corresponding to “From”. Note that the position indicator is “above”.

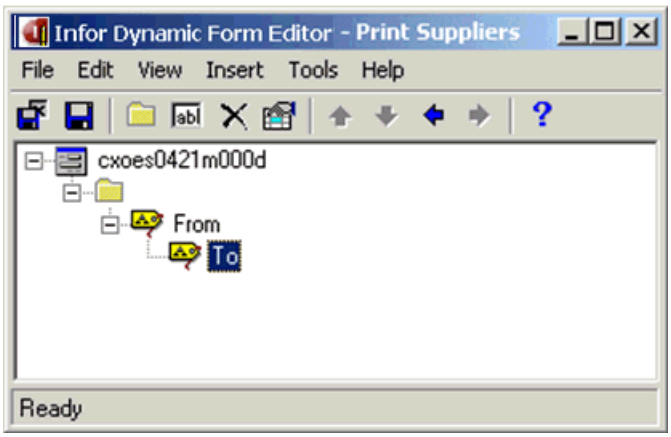
See the following table for an example of the Field Properties:

Field	Input Value
Field Name	label.only
Use Field as Label Only	Yes
Label Code	ttgen.from
Position	Above

Insert a second label only field with a label code corresponding to “To”. The “To” label field is a child of the from label field.

See the following table for an example of the Field Properties:

Field	Input Value
Field Name	label.only
Use Field as Label Only	Yes
Label Code	ttgen.to
Position	Above



Insert Label Fields

Step 3: Insert Selection Fields

The selection fields you insert on the form must correspond to the variables that are used in the main SQL statement in the program script.

Insert the "From" field corresponding to the first selection field. This field is denoted as a table field name appended with a “.f”.

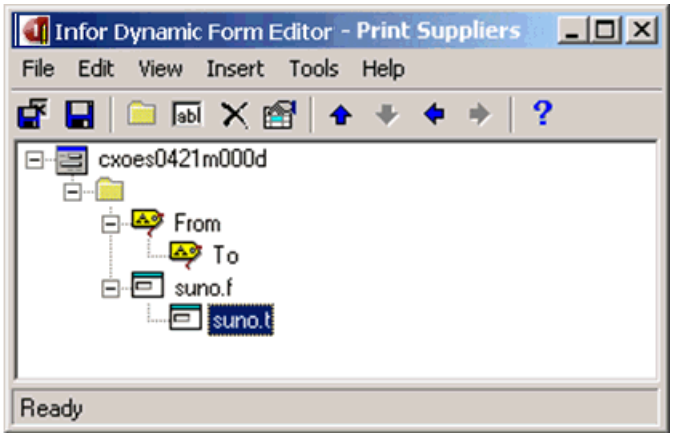
See the following table for an example of the Field Properties:

Field	Input Value
Field Name	suno.f
Use Field as Label Only	
Domain	cx suno
Form Field	Yes, fmin($#)
Label Code	oes021.suno
Position	Before
Zoom	Session
Zoom To	cxoes0121m000
Return Field	cxoes021.suno

Insert the "To" field corresponding to the first selection field. This field is denoted as a table field name appended with a “.t”.

See the following table for an example of the Field Properties:

Field	Input Value
Field Name	suno.t
Use Field as Label Only	
Domain	cx suno
Form Field	Yes, fmax($#)
Label Code	
Position	
Zoom	Session
Zoom To	cxoes0121m000
Return Field	cxoes021.suno



Insert Selection Range Fields

Step 4: Edit Standard Commands

Click **Standard Commands** on the **Tools** menu. The Standard Commands (ttadv3110s000) session starts. Edit the Standard Commands for the form so that the following commands are available:

Commands	Available
Modify Record(s)	Yes
Print Report	Yes
Cancel	Yes
Make Job	Yes (Optional)
Save Defaults	Yes (Optional)
Get Defaults	Yes (Optional)

Step 5: Save the form

Save the form.

You can view the effect of your changes from within the form editor: On the **File** menu, click **Show Details Session**. The session starts.

Form Result

Save and Exit the form.

To create reports

Important!

The sections below describe the sessions that you must use to complete the procedure. However, they do not describe all details of these sessions. For detailed session information, refer to the Infor Web Help.

Procedure aim

The aim of this procedure is to create reports for your LN application. Reports are used to print data to the screen, to a printer or to another output device.

Procedure result and prerequisites

Result

Reports to print data from your LN application.

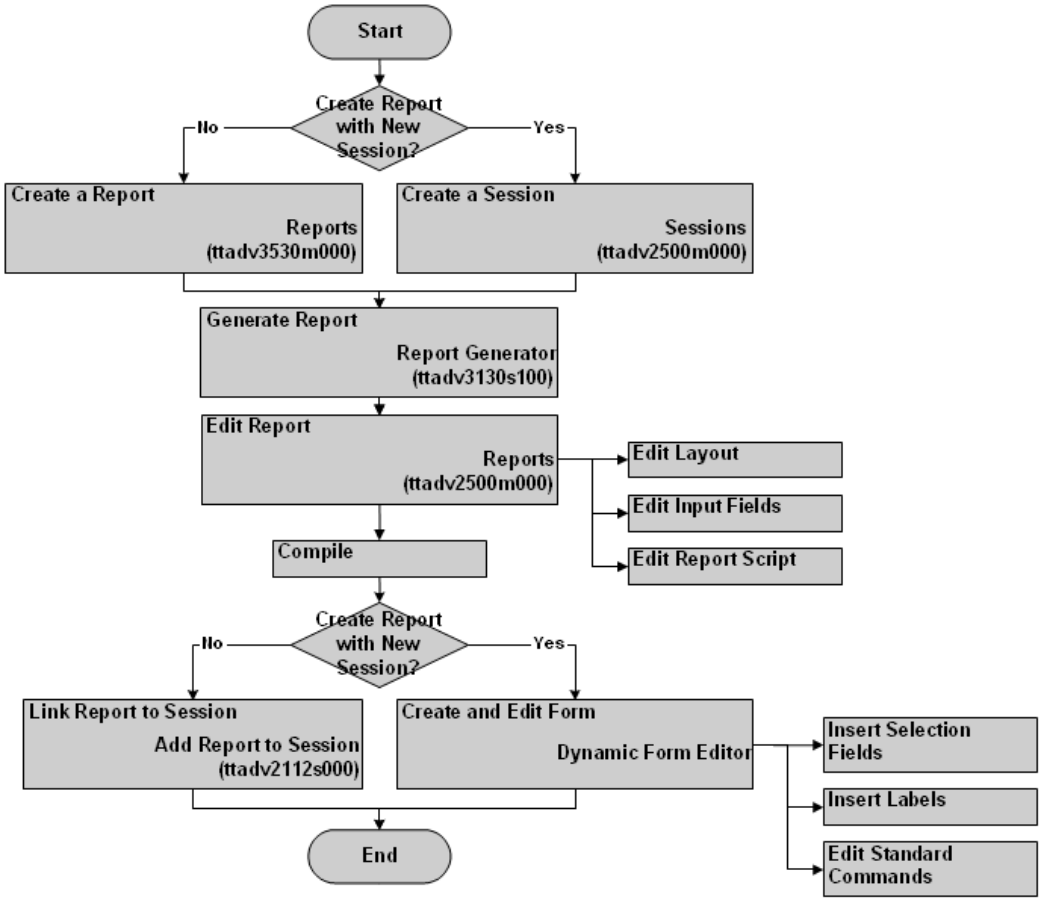
Prerequisites

To create reports you need:

- A development package VRC in which you can create the reports.
- Developer authorization for the development VRC.

Procedure summary

The following figure shows the procedure steps and the corresponding sessions or programs:



To create reports - procedure flow

While not part of the flow, you may also have to edit the program script to complete your report.

Procedure details

Step 1: Create a report or create a session

You can create a new report for an existing print session, or create a new print session with a corresponding report.

To create a report:

1. Start the Reports (ttadv3530m000) session.
2. Click **New**.

3. You will be asked if you want to start the report generator. Click **Yes** to start the generator.
   - Note: If you click **No**, you must create the report manually. In that case, skip the "Generate report" step in this procedure and proceed with the "Edit report" step.

Alternatively, you can also create a report by duplicating an existing report.

To create a new print session:

1. Start the Sessions (ttadv2500m000) session.
2. Click **New**. The Sessions (ttadv2100s000) details session starts.
3. Enter the new session. A print session should have at least these attributes:
   - Session Type: Print
   - Default Button Command Type: Standard Command
   - Default Button: Print.data
4. Save the new session. Do not close the Sessions (ttadv2100s000) details session.
5. Click **Report Generator** to start the generator.

Step 2: Generate report

You can generate a (basic) report. Subsequently, you can edit the generated report until the report matches your requirements.

To generate a report:

1. Start the report generator. You can start the report generator from the Reports (ttadv3530m000) session and from the Sessions (ttadv2100s000) session. See the previous step in this procedure.
2. Edit the report generator options: You must specify the main table and the fields for the report, and various other attributes, such as the report code, the report size, and the report type. For details refer to the online help of the Report Generator (ttadv3130s100) session.
   - Note: to select fields, you must click **Select Fields**. The Select Report Fields (ttadv3130s200) session starts. Select the fields you wish to output in the report, and click **Save and Close** to return to the report generator.
3. Click **Generate**. The report is compiled automatically.

Step 3: Edit report

You can now edit the report until it matches your requirements. You can, among other things, add new layouts to the report, edit the layouts in the report editor, change the sorting and/or grouping of records, and create a report script.

For details, refer to *To edit reports (p. 99)* .

Step 4: Compile

Click **Compile** in the Reports (ttadv3530m000) session. A Compilation result window opens with the result of the compilation.

Step 5: Link report to session or create and edit form

When you create a new report for an existing print session, you must link the report to that session.

To link a report to an existing session:

1. In the Sessions (ttadv2500m000) session, select the session you wish to link the report to.
2. On the appropriate menu, click **Reports...**. The Reports (ttadv3530m000) session starts.
3. On the appropriate menu, click **Link Report to Session**. The Add Report to Session (ttadv2112s000) session starts.
4. Specify the **Report Group**, the **Serial Number**, and the report code you wish to link to the session. For details, refer to the Infor Web Help.
5. Click **OK**. The report is now displayed in the reports list for the session.
6. Exit the Reports session.
7. In the Sessions (ttadv2500m000) session, click **Compile**. This action converts the new session structure to the runtime dictionary.

When you create a new report as part of a new print session, the report is automatically linked to the new session. Before you can print the new report, you must create a form for the new print session. For details, refer to *To create a form for a print session (p. 88)* .

To edit reports

This topic describes the following report editing operations:

- *To edit report input fields (p. 99)*
- *To edit report layouts (p. 100)*
- *To edit a report script (p. 104)*

Important!

The sections below describe the sessions that you must use to edit reports. However, they do not describe all details of these sessions. For detailed session information, refer to the Infor Web Help.

To edit report input fields

In a report, you can create, edit and delete report input fields.

A report input field defines the data that will appear on the report. You can use any value from the program script as a report input field.

You can add a field as a report input field when:

- The field is a table field and it is selected by the program script
- The field is a variable and it is defined in the program script (or library)

Note

You can define variables in a report script. Report script variables can be used in a print expression to output the variable value. However, you should not define a report script variable as a report input field.

To create a report input field:

1. In the Reports (ttadv3530m000) session, select the report you wish to edit.
2. On the appropriate menu, click **Input Fields...**. The Report Input Fields (ttadv3532m000) overview session starts.
3. If the report field does not exist for the report, click **New** to add a report input field. The details session starts.
4. Enter a report input field, or click the Zoom icon to display table fields. You may wish to enter the table id in the **Field Name** field to quickly find the correct table field.
5. Complete the dialog. Save the changes and close the dialogs.

To edit report layouts

An LN report usually contains multiple report layouts, such as Header, Before.field and Detail.

Report layouts specify the report fields and the position of those fields on a report line. The report layouts execute depending on the data that is processed from the report input fields.

For information on the types of layouts, refer to *Reports (p. 38)* .

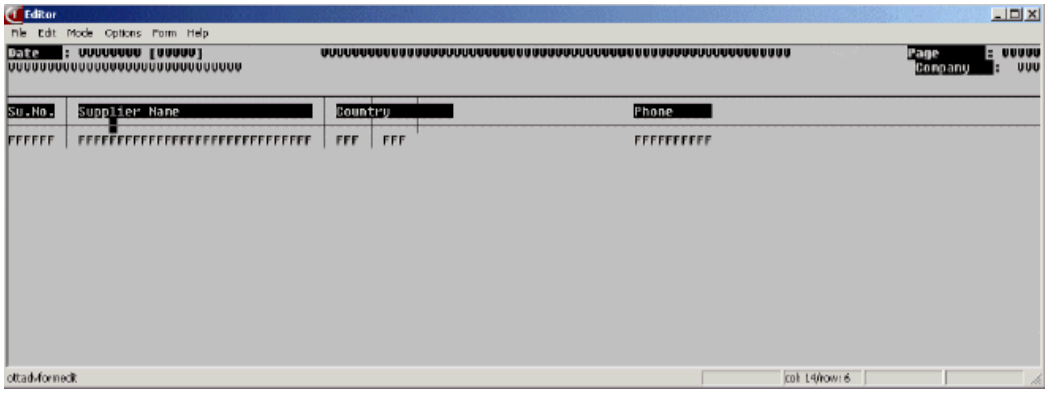
To view report layouts

A report can have multiple layouts. To view the report layouts, in the Reports (ttadv3530m000) session click **Layouts...**, or click the **Layouts...** command on the appropriate menu. The Report Layouts (ttadv3531m000) session starts. The layouts for the report are displayed.

To edit all layouts

You edit the layouts to change the report contents and positions.

To edit all the layouts, in the Report Layouts (ttadv3531m000) session, click the **Edit All Layouts...** command on the appropriate menu. The report layout editor starts.



Edit All Layouts

The layout editor contains many options. Not all the options are shown here. To view the help options, click **Help** on the **Help** menu.

The major editing tasks are moving fields and labels, drawing boxes, and creating report fields.

To move fields and labels

You need to move fields and labels in order to align columns or make room for other data fields. To move a field or label:

1. Select the field or label (do not double-click).
2. Holding your mouse button, drag the mouse or use the arrow keys to place the field in another location in the layout.
3. Drop the field in the desired location. When using the arrow keys, you must press Esc.

In addition to moving labels, you may need to change the label length, or create new labels with shorter lengths in order for the label to fit in the column width.

To create or edit a report field

A report field is denoted in the editor with the capital F's ("FFF").

To create a report field:

1. Place your cursor in the top left position where you want the new field.
2. On the **Edit** menu, click **Form Field**. Alternatively, you can press CTRL + F simultaneously. The Report Fields (ttadv3134s000) session starts.
3. Enter a print expression. A print expression can be:
   - A report input field
   - A report script variable
   - A calculated expression from 1 or more other input fields (or report script variables)
4. Specify the remaining report field options in the **Display**, **Conditions** and **Font** tabs.

To edit a report field: double-click on the field to start the Report Fields (ttadv3134s000) session, where you can modify the field's properties.

To draw boxes and graphical characters

After adding report fields, the content is complete, and you may wish to align the data in columns. You align data into columns using the boxes, or graphical characters.

To draw a box:

1. Position the cursor in the top left position of the box.
2. On the **Edit** menu, click **Draw Box**. The cursor changes to a plus (+) sign.
3. Click the cursor on the bottom right corner of the box. The box is now complete.

Draw Box - cursor changes to plus (+) sign

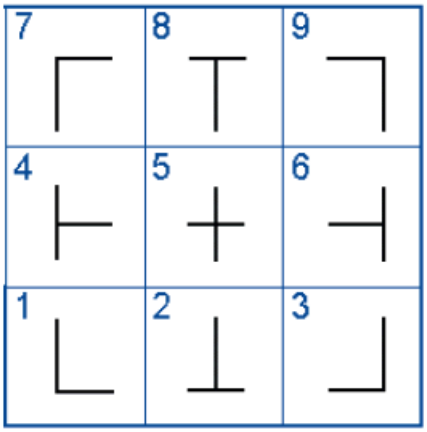
Draw Box Complete

A box will draw irrespective of any adjoining lines.

You can join lines or boxes using graphical characters.

To insert a graphical character:

1. On the **Mode** menu, click **Toggle Graphics** to enter graphical mode.
2. Use your numeric keypad to select the appropriate character. See the figure below. Use the pipe (|) key for vertical lines and the hyphen (-) for horizontal lines.
3. On the **Mode** menu, click **Toggle Graphics** to exit graphical mode.



Numeric Keypad and Graphical Characters

To create a layout

Typical reasons for creating a layout are:

- Before field layouts to create sub-headings
- After Field Layouts to calculate sub-totals using the aggregate functions
- After Report layout to create a grand total using the aggregate functions
- An additional Detail layout to print optional detail information on a new line

To create a layout:

1. In the Report Layouts (ttadv3531m000) session, click **New**. The Report Layouts (ttadv3131s000) details session starts.
2. Enter the layout information.
3. Edit the layout in the report editor: place the desired fields and/or labels in the new layout.

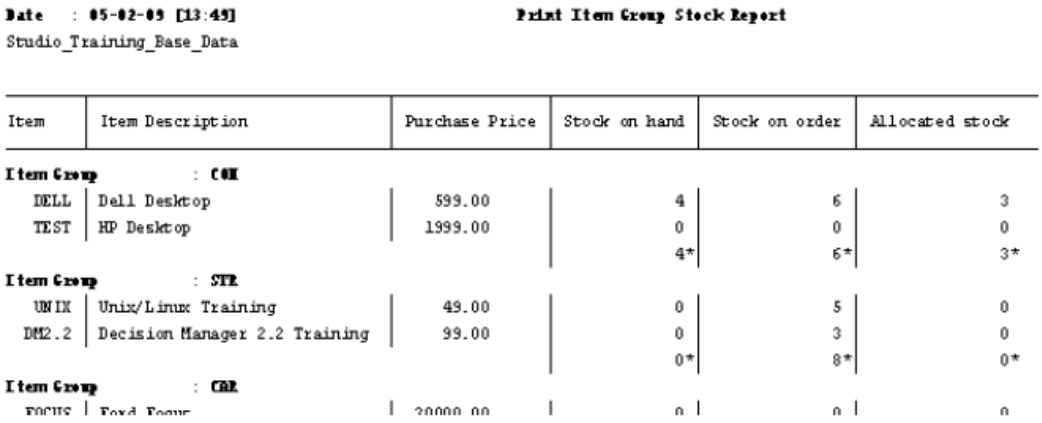
Example

The items are printed by item group. At the end of each item group a sub-total is displayed.

To achieve this, you must:

- Make sure that the records are sorted by Item Group: in the report input field properties, define a sort mode for the Item Group field.
- Add a before.field layout. In the layout properties, you must select the Item Group field.
- Start the report editor and place the Item Group field and the corresponding label in the new before.field layout.
- Add an after.field layout. In the layout properties, you must select the Item Group field.
- Start the report editor and place the desired sub-totals in the after.field layout.

- For example: to place a sub-total for the Stock on Hand field (cxoes020.stoh) you must place the cxoes020.stoh field in the after.field layout. In the field properties you must select the "Total" aggregate function.



Date : 05-02-09 [13:45] Print Item Group Stock Report
Studio_Training_Base_Data

Item	Item Description	Purchase Price	Stock on hand	Stock on order	Allocated stock
Item Group : COM					
DELL	Dell Desktop	599.00	4	6	3
TEST	HP Desktop	1999.00	0	0	0
			4*	6*	3*
Item Group : STR					
UNIX	Unix/Linux Training	49.00	0	5	0
DM2.2	Decision Manager 2.2 Training	99.00	0	3	0
			0*	8*	0*
Item Group : CAR					
FOCUS	Ford Focus	20000.00	0	0	0

Report with sub-totals per Item Group

To edit a report script

A report script allows you to customize the processing of the report.

Report scripts have sections that correspond to a layout. A section can be processed before a layout, or after a layout. The report script is compiled with the report when the report is compiled.

For details on report scripts, refer to:

- *Report scripts (p. 41)*
- "Report Script Features" in the *LN Programmer's Guide*. Predefined variables used in a report are also listed.

To edit a report script, you must use the Reports (ttadv3530m000) session.

To create dashboards

To create a dashboard you must:

- create the dashboard session and its form using the Sessions (ttadv2500m000) session. The dashboard session uses the main table of the entity that you want to select.
- Create a UI script for the dashboard session: The UI script controls most of the actions, including the selection, calculation and starting the details sessions.
- Create details sessions that allow the dashboard user to drill-down into the details of the overview presented in the dashboard.

Important!

The sections below describe the sessions that you must use to create a dashboard. However, they do not describe all details of these sessions. For detailed session information, refer to the Infor Web Help.

To create the session

The dashboard session uses the main table of the entity that you want to select. The session is similar to an overview session. However, an overview session usually has a synchronized dialog to its details session. A dashboard session controls the synchronization to the various details sessions within the program script.

Example- Items Dashboard

You need to create a dashboard session that uses the Items as the main entity. When you select an item, the dashboard will show the item, description, purchase price, stock on-hand and stock on-order for the item. Next to the purchase price, it will calculate the last purchase price. Next to the stock on-hand, it will display the inventory value (valued at purchase price). Next to the stock on-order, it will show the latest order date for orders that have not been delivered. A document button is available to view purchase orders for the item.

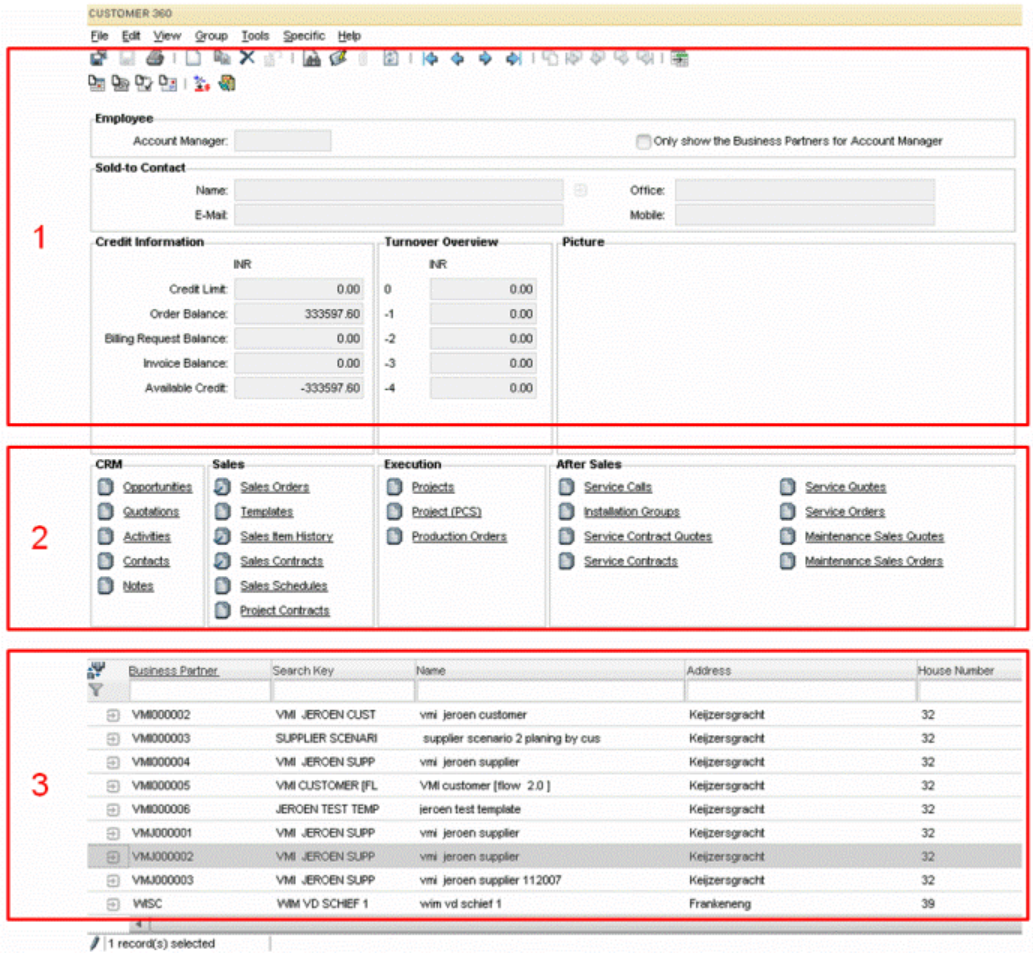
You create an items dashboard session, using the Items table. The session is a display session type.

This table shows the session properties:

Package	cx
Module	oes
Session	0520m000
Description	Item Dashboard
Standard Script	No
Program Script	cx oes 0520
Main Table	cx oes 020
Start Command Type	No Start Command
Start Command	0
Session Status	Developing
Dynamic Session	Yes
Integrated	Yes
Session Type	Display
Window Type	List Window
Main Session	Yes
Synchronized Dialog	[Empty]

To create the form

This diagram shows a dashboard session:



Dashboard session - form sections

The form of a dashboard session consists of these sections:

1. Details

   This section contains fields that are declared as variables, as opposed to table fields. While not required, these fields are typically defined in one or more groups.

2. Links

   This section contains check boxes and buttons. The check boxes are variables that are defined in the session's program script. The check boxes do not have labels. The buttons are field level form commands, that are linked to the check boxes. The form commands use labels to provide the names for the buttons.

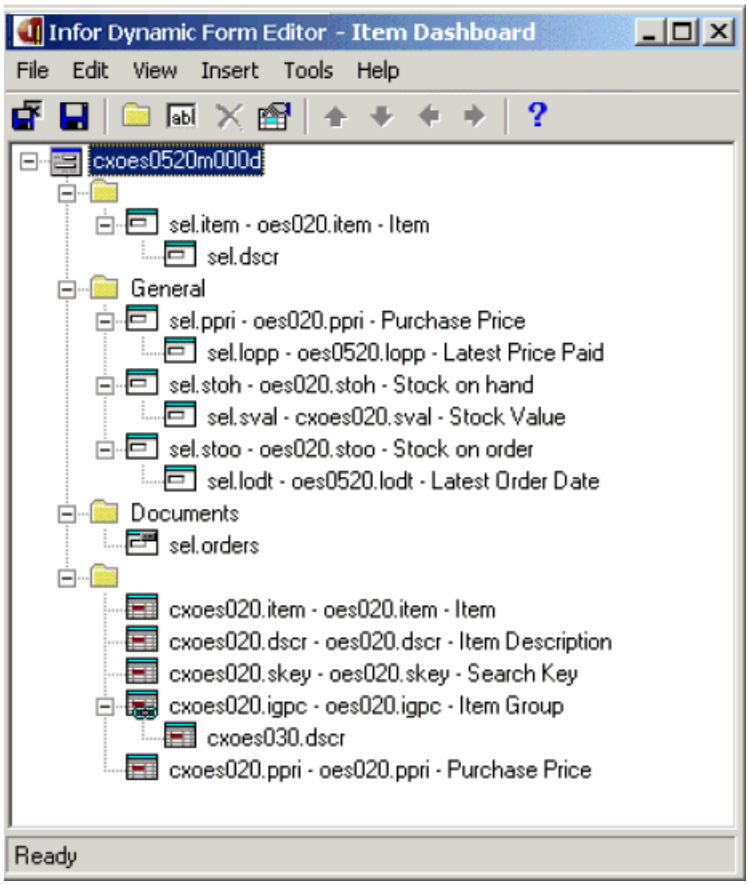
3. Records

The records are displayed in a grid. The grid is typically defined in another group. The grid contains table fields and the number of fields in the grid may be limited to a few key fields. A new group is not required, but helps you in separating the Links and Records sections.

The form contains group boxes and field buttons. Therefore **Rich View** should be selected in the Session Properties of the form.

Example- Items Dashboard

You create an items dashboard session using the Items table. The session is a display session type.



Dashboard Form Editor

In the Dynamic Form Editor you must specify the session properties, the field properties and form commands.

Specify these properties in the **Session Properties** dialog box:

Field	Value
Overview Session	Yes
Rich View	Yes
Remaining fields	No

Fields in the view area are variables that are defined in the session's program script, such as "sel.item". See the sample script in this section for more examples. Specify the following properties in the **Position** tab in the **Field Properties** dialog box:

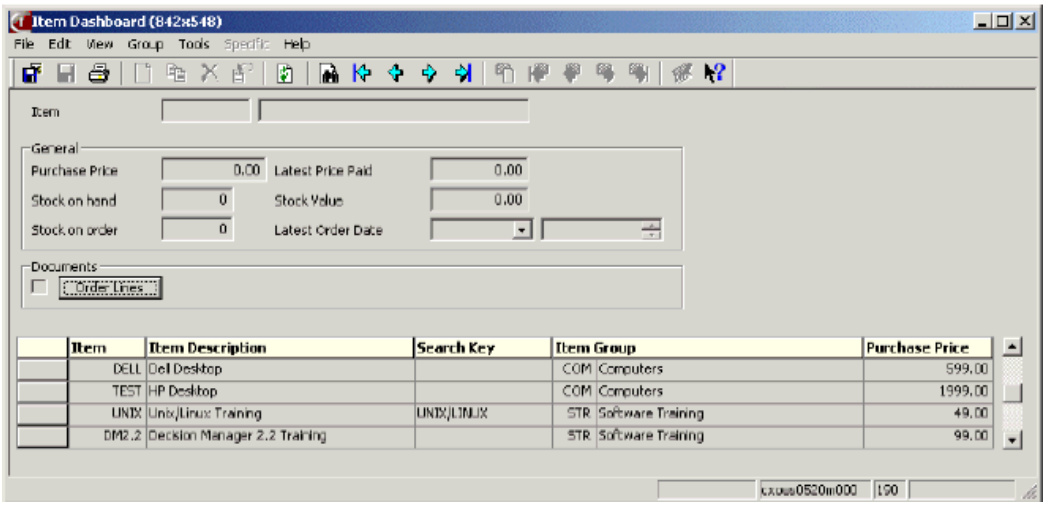
Field	Value
Overview Session	Yes
View Field	Yes
Remaining fields	No / empty

The field properties for the **Documents** check box fields do not have a label code. The button next to each document check box is defined as a field level form command, which is linked to the corresponding check box field. The form command executes a function that will be defined in the program script.

This table shows the properties of the field level form command that is linked to the "sel.orders" check box:

ID	1
Activate a	Function
Session/Function/Business Method	exec.orders
Parent (Menu)	0
Sort Sequence	0
Command Type	Field
Group/Field Number	90
Label	oes022.orli
Long description	Order Lines
Short Description	Order Lines
Shortcut Key	[Empty]
Separator	No

You can use the **Show Overview Session** command in the Dynamic Form Editor to start the session.



Show Overview Session Result

To create the program script

The program script controls most of the actions, including the selection, calculation and starting the details sessions.

The actions that occur when the user selects a row are programmed in a mark.occur event. This event will gather the information for the selected value, and enable the form commands. It will also re-synchronize the details sessions.

The detail sessions are started via a form command. Since the script will control the synchronization, it is preferred that the form commands call a function in the script. This function is defined as an extern function.

Example- Items Dashboard

The Items dashboard needs to:

- Declare the form variables.
- Declare the tables used in the calculation of the form variables (unless a library calculates these values)
- Contain actions for the mark.occur event
- Contain actions for the functions called by the field level form commands
- Start a synchronized child process in the functions called by the field level form command. Uses the start.synchronized.child function.

See the following code for an example:

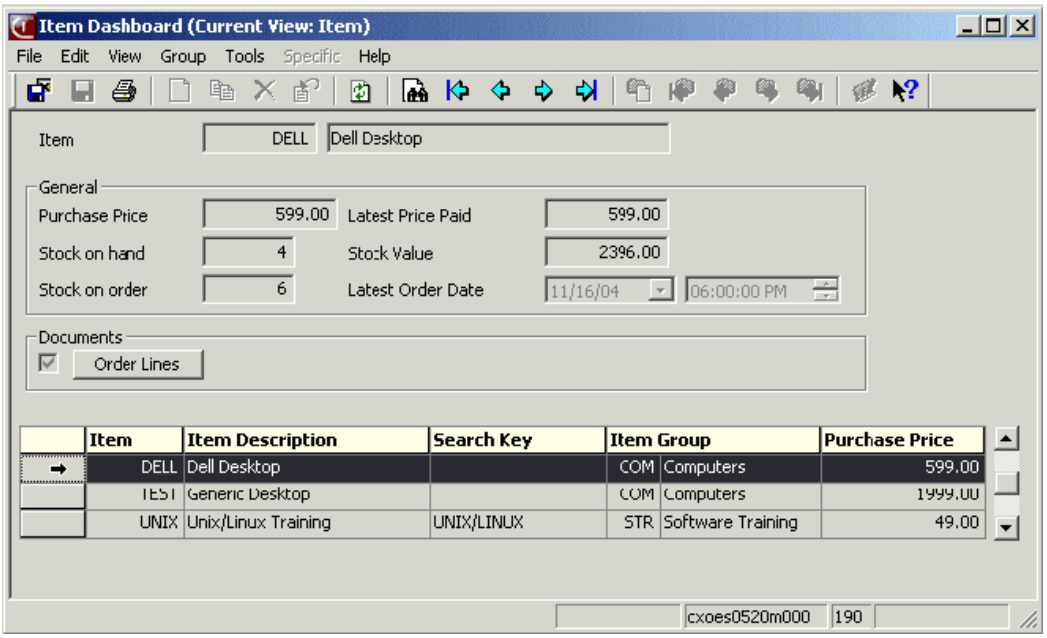
```
|*****************************************************
|* cxoes0520  0  VRC B61U a  dv00
|* Items Dashboard
|* Development User
|* 2005-02-10
```

```
|*******************************************************
|* Main table cxoes020 Items, Form Type 1
|*******************************************************

|***************************** declaration section ******
declaration:

  table   tcxoes020 | Items
  table   tcxoes022 | PO Headers
  table   tcxoes023 | PO Lines

  extern domain cxitem sel.item
  extern domain cxdscr sel.dscr
  extern domain cxprice sel.ppri, sel.lopp, sel.sval
  extern domain cxquan  sel.stoh, sel.stoo
  extern domain cxdate  sel.lodt
  extern domain cxyeno  sel.orders

   long orders.session.child.id, dummy.ret

|***************************** program section **********
|***************************** group section ************
|***************************** choice section ***********
choice.mark.occur:
after.choice:
 if number.of.marks = 1 then
  |use a pre-defined var to test that only
  |one item is selected from the list
  fill.sel.values()
  sync.open.session()
 else
  clear.sel.values()
 endif
 | need to display values because they are not
 | displayed automatically
 display.all()

|***************************** function section *********

functions:
function fill.sel.values()
{
 sel.item = cxoes020.item
 sel.dscr = cxoes020.dscr
 sel.stoh = cxoes020.stoh
 sel.stoo = cxoes020.stoo
 sel.ppri = cxoes020.ppri
 sel.sval = sel.stoh * sel.ppri
 get.last.order.info.for.item()
}

function clear.sel.values()
{
 sel.item = " "
 sel.dscr = " "
 sel.ppri = 0.0
 sel.lopp = 0.0
 sel.sval = 0.0
 sel.stoh = 0
 sel.stoo = 0
 sel.lodt = 0
 sel.orders = cxyeno.no
}
```

```
function get.last.order.info.for.item()
{
 |* last purchase price is latest order, no matter of status
 select cxoes022.*, cxoes023.*
 from   cxoes022, cxoes023
 where  cxoes023.orno refers to cxoes022
 and    cxoes023.item = :sel.item
 order by cxoes022.odat desc
 as set with 1 rows
 selectdo
  sel.lopp = cxoes023.ppri
  sel.orders = cxyeno.yes
 selectempty
  sel.lopp = 0
  sel.orders = cxyeno.no
 endselect

 |* last purchase order date is latest open order
 |* stat <> delivered

 select cxoes022.*, cxoes023.*
 from   cxoes022, cxoes023
 where  cxoes023.orno refers to cxoes022
 and    cxoes023.item = :sel.item
 and    cxoes022.stat <> cxstat.delivered
 order by cxoes022.odat desc
 as set with 1 rows
 selectdo
  sel.lodt = cxoes022.odat
 selectempty
  sel.lodt = 0
 endselect
}

function extern exec.orders()
{
 if sel.orders = cxyeno.yes then
   | document is checked, start the child process
   if orders.session.child.id then
  | already started the session, so re-use it
  | but refresh with new item values
  | if closed, will set child.id to 0
  dummy.ret = synchronize.with.child(orders.session.child.id)
   endif
   if not orders.session.child.id then
  | need to start the session
  session.zoomindex = 1
  | if there is a alt key by item, we could use it
  orders.session.child.id = start.synchronized.child(
     "cxoes0523m100",
     "cxoes020.item",
     "cxoes020.item")
  |* start orders by item. Send parent var of cxoes023.item,
  |* and child var of cxoes020 for query extension to
  |* work in child session.
   endif
 endif
}

function sync.open.session()
{
 | have the dashboard sync the open sessions when a new
```

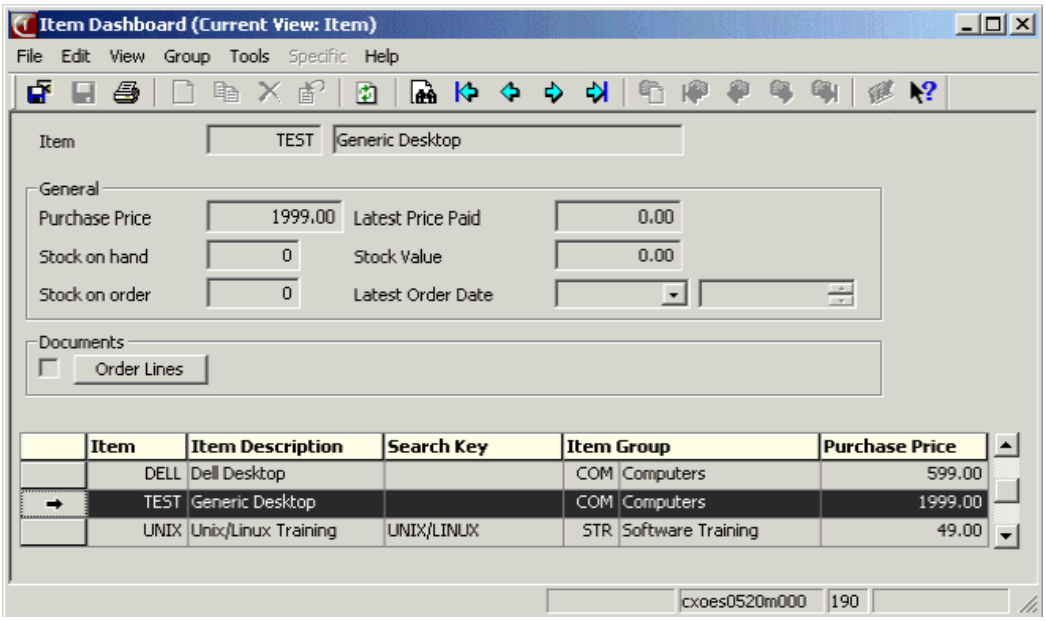
```
 | item value is selected
 if orders.session.child.id then
  | already started the session, so re-use it
  | but refresh with new item values
  dummy.ret = synchronize.with.child(orders.session.child.id)
 endif
}
```

When you compile this script, the session can compute values.



Item Dashboard with Computed Values

Select another item to show the values for that item.



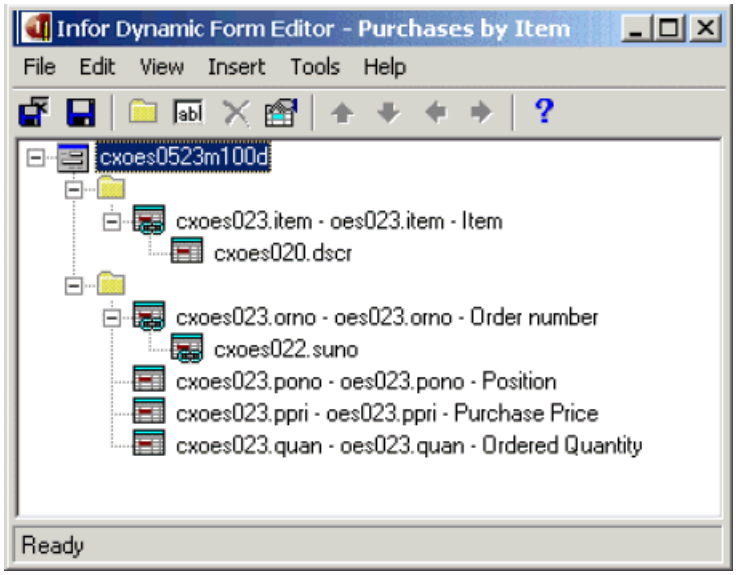
Item Dashboard

To create the details sessions

The details sessions allow the dashboard user to drill-down into the details of the overview presented in the dashboard. A details session will typically import the values from the dashboard session, and start or limit the display to those records. The program scripts of the details sessions can be modified to import these values and in some cases, extend the query so that only the desired rows are displayed.

Example- Items Dashboard

You have created an Items dashboard session using the Items table. The order lines for the Item must be shown in a details session. The Purchase Order Lines (cxoes0523m100) session uses the Purchase Order Lines (cxoes023) table to show purchase order lines. You edit the form of this session to insure that the **Item** field is a view field.

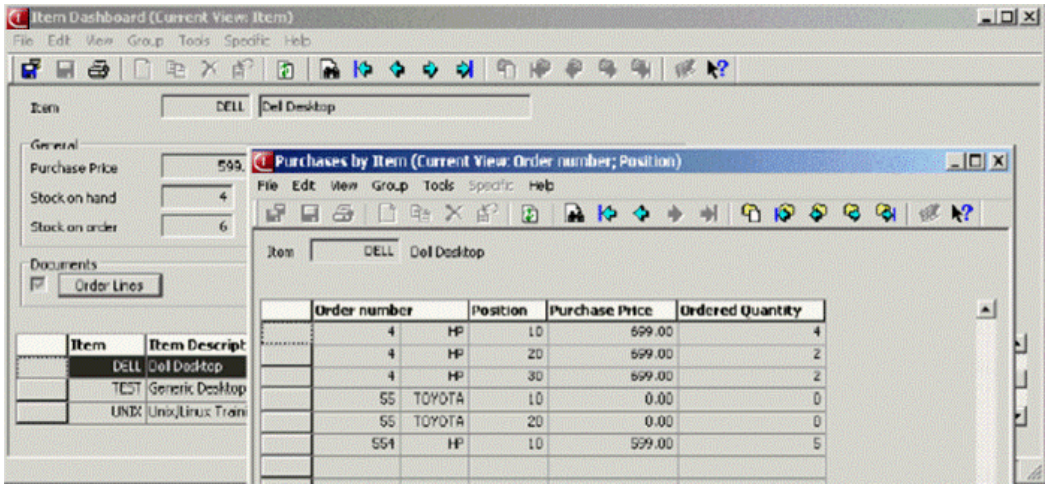


Purchases by Item Form

The program script extends the query in the before.program section. The query extension is required because there is not an index on the field that contains the filter.

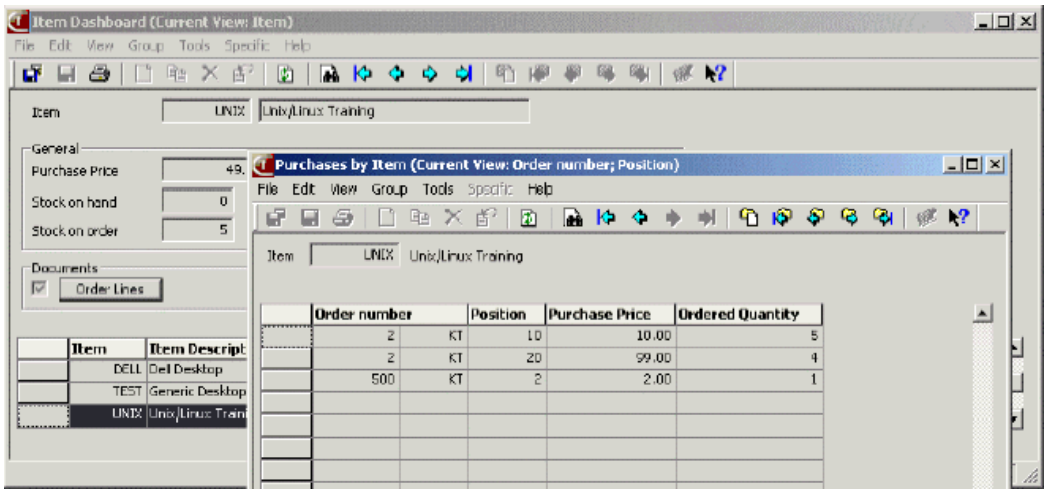
```
|*************************************************************
|* cxoes0523  0  VRC B61U a  dv00
|* Purchases by Item
|* Development User
|* 2005-01-07
|*************************************************************
|* Main table cxoes023 Purchase Order Lines, Form Type 1
|*************************************************************

|***************************** declaration section **********
declaration:

  table   tcxoes020 | Items
  table   tcxoes023 | Purchase Order Lines

|***************************** program section **************
|* import item from calling session
|* query extension to create filter

before.program:
 query.extend.where(" cxoes023.item = :cxoes020.item")

|***************************** group section ****************
```

When this session is started from the dashboard, the session will show only the orders for the item that is selected in the dashboard.



Dashboard with Detail Session

When you select another item, and click the **Documents** button, the details session will synchronize. The item from the dashboard session will be sent to the details session, and the purchase orders for that item will display.



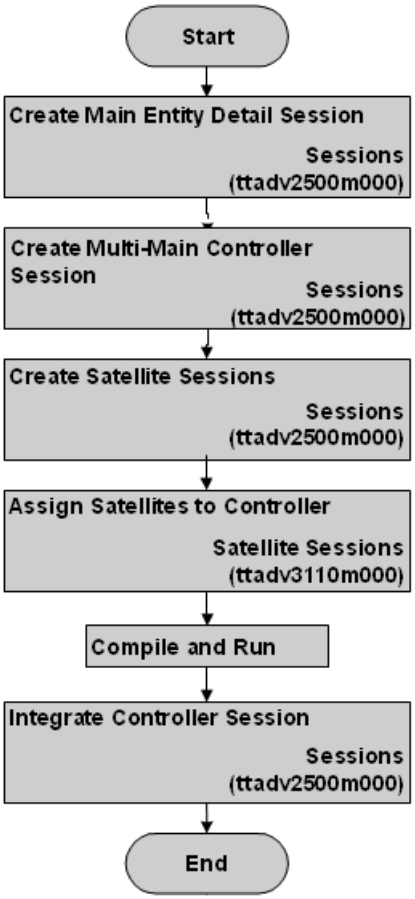
Select Item on Dashboard with Detail Session

To create MMT Sessions

Important!

The sections below describe the sessions that you must use to create MMT sessions. However, they do not describe all details of these sessions. For detailed session information, refer to the Infor Web Help.

This diagram shows the procedure steps and the corresponding sessions or programs:



The flow: Create a multi-main table session

To create a main entity detail session

If you want the multi-main table controller session to use a details session, you must determine if the details session already exists, or if you need to create it.

When you create the details session, you must enter a label code in the **Label of Object** field in the session properties. This allows the label to show on the multi-main table controller session for opening the details session.

Example- details session

The Purchase Order Headers (cxoes0122m000) session already exists, and you can use it as the details session. You enter a label code in the **Label of Object** field.

Field	Input Value
Label of Object	oes022 (Purchase Order Headers)

To create a multi-main table controller session

To create the multi-main table controller session:

1. Start the Sessions (ttadv2500m000) session and specify the properties for the controller session. Points of attention:
   - Session Type: Maintain
   - Window Type: Multi Main Table
   - Main Session: Yes
   - Synchronized Dialog: the code of the main entity detail session, e.g. cxoes0122m000
2. Edit the form and insert the fields you wish to have in the session.

Example- multi-main table controller session

The Purchase Orders multi-main table controller session is created with the following properties:

Field	Input Value
Package	cx
Module	oes
Session	0122s100
Description	Purchase Orders
Standard Script	No
Program Script	cxoes0122s100
Main Table	cxoes022
Start Command Type	No Start Command
Start Command	0
Session Status	Developing
Dynamic Session	Yes
Integrated	Yes
Session Type	Maintain
Window Type	Multi Main Table
Main Session	Yes
Synchronized Dialog	cxoes0122m000

To create a satellite session

You can use an existing satellite session, or create a new one. The key field that links the satellite entity to the main entity must be a view field in the form. You must specify a label of object for the satellite session.

Example- satellite session

The Purchase Order Lines (cxoes0123m000) session is an existing satellite session. You enter a label code in the **Label of Object** field in the session's properties.

Field	Input Value
Label of Object	oes023 (Purchase Order Lines)

To assign satellites to the controller

You must link the satellite sessions to the multi-main table controller session.

To do so:

1. Start the Sessions (ttadv2500m000) session.
2. Select the multi-main table controller session.
3. On the appropriate menu, click **MMT Satellite Sessions**. The Satellite Sessions (ttadv3110m000) session starts.
4. Click **New**.
5. Enter the next index number.
6. Enter the satellite session code.
7. Either enter the field mapping information, or click **Generate Field Mapping**.

Example- assign satellite sessions

You link the Purchase Order Lines (cxoes0123m000) session as a satellite session to the Purchase Orders (cxoes0122s100) multi-main table controller session. You enter the following field mapping:

Field name in MMT session	Field name in satellite
cxoes022.orno	cxoes023.orno

To compile and run the controller session

You must compile the multi-main table controller session after you assign a satellite session.

To compile the session:

1. Start the Sessions (ttadv2500m000) session.
2. Select the session.
3. Click **Compile**.

You can now run the multi-main table controller session.

To integrate the controller session

The multi-main table controller session is the equivalent of a details session for the main entity. Therefore, you should integrate the session, either as a synchronized dialog of the main overview session for the table, or as a child session of an overview display session.

To integrate the controller session, using the session as a synchronized dialog:

1. Start the Sessions (ttadv2500m000) session.
2. Select the main overview session for the table. Modify the properties of this session: Fill the **Synchronized Dialog** field with the multi-main table controller session code.
3. Save the session.

Example- Multi-main table controller session integrated with overview session

You modify the Purchase Orders (cxoes0122m000) overview session so that it uses the multi-main table controller session (cxoes0122s100) as a synchronized dialog. When you open a purchase order in the overview session, the multi-main table controller session opens with the purchase order that you selected.

To translate software components

In LN you can export the labels, questions, and messages from a development or translation system and import them into other LN environments. Export and import of labels is performed by using XML-formatted files. Every XML file contains a selection of translatable components based on the user's settings. The translated language files can be imported back into the Infor environment. The import process includes a conversion to runtime. All descriptions are stored as labels which can be handled by the same import and export processes.

Multi-language

Most LN implementations use several system languages simultaneously. In addition, various languages for user reports such as customer facing documents sometimes also require languages other than the local languages used.

LN uses the Language Translation System (LTS) to both reduce the language-dependency of the LN applications and to lower the costs of media creation.

Translatable software components, such as labels, questions, and messages, are stored separate from language-independent software components, such as form and report designs. The descriptions of the translatable software components are stored per language.

Language-independent components contain only references to translatable components. E.g. a form layout or a report layout contains label codes, and a UI script can contain message codes or question

codes. The descriptions of the labels, messages and questions are read at runtime, depending on the language of the user.

Example

You use 2 system languages: Dutch and English.

Your LN system contains, among other things:

- Form layouts and report layouts (one set for all languages)
- 2 sets of label descriptions, message descriptions and question descriptions. (Dutch and English)

The form of the Add Session to Job (ttaad5102s000) session contains a label ttaad320.seqn. The description of this label is stored in 2 languages:

Language	Label description
Dutch	Volgnummer
English	Sequence No.

When a user runs the session, the label description is displayed depending on the user's language.

High Ascii Tolerance

If you plan to deliver software components or translations from your LN environment and your environment does not run in Unicode mode, you must set the *high_ascii_tolerance* resource to 0.

If *high_ascii_tolerance* is not set to 0, problems will occur when the delivered components are imported in Unicode environments.

To set *high_ascii_tolerance* to 0, add the following line in the *$BSE/lib/defaults/all* file:

```
high_ascii_tolerance:0
```

Important!

The sections below describe the sessions that you must use to translate software components. However, they do not describe all details of these sessions. For detailed session information, refer to the Infor Web Help.

To translate labels, questions and messages

LN enables you to translate labels, questions and messages.

Basic procedure

1. Use the following sessions to translate the labels, questions and messages.:
   - Labels (ttadv1140m000)
   - Question Descriptions (ttadv4166m000)
   - Message Descriptions (ttadv4150m000)
2. Run the Compile Labels (ttadv1243m000) session to store the new label descriptions in the runtime data dictionary.

Additional sessions

The following sessions are useful if you want to distribute translated components from a translation/development system to other systems, or if you want to perform the translation in an external application, e.g. in a professional translation application such as Transit.

Session	Description
Export Language Dependent Data to XML - Wizard (ttadv8910m000)	Use this session to export the components to XML files.
Merge Tool for XML-files with Language Dependent Components (ttadv8930m000)	Use this session to merge multiple XML files containing language-dependent components into one file.
Import Language-Dependent Data from XML - Wizard (ttadv8920m000)	Use this session to import the content of XML files containing language-dependent components into an LN environment. Note: After you import the components, you must compile the labels.

Chapter 6
Session Personalizations

6

Session Personalizations

In the Worktop and Web UI clients, end users can customize the user interface of LN sessions, so that the sessions meet their personal demands.

In the Worktop client, end users can:

- Customize the columns of an overview session.
- Move the toolbar.

In the Web UI client, end users can:

- Hide/unhide selected fields on sessions, including fields of details sessions, and change the texts and fonts of labels.
- Personalize the columns in the grid and the view fields above the grid, in overview sessions.
- Hide/unhide buttons in secondary toolbars.
- Customize menus: for example, hide/unhide commands in a session's appropriate menu.

For more information, refer to *Personalize ERP sessions* in the Infor Web Help.

Customization maintenance

The end user customizations are stored centrally on the LN server. You can maintain these customizations in the Session Personalizations (ttadv9100m000) session. In this session you can, for example, modify existing customizations or remove unwanted customizations.

From this session, you can start the Copy to other Customization Level (ttadv9200s000) session. You can use this session, for example, to copy customizations from a particular user to the company level. For example: copy customizations that are specific for user "jsmith" to company 100, so that these customizations apply to all users in company 100.

For more information, refer to the online help of the sessions mentioned.

Chapter 7
Version and Release Management

7

Version and release management introduction

This section describes LN 's version and release management, which manages the various versions of the LN packages, their corresponding releases, and customizations.

An overview of version and release management

Software is constantly changing. Therefore, you must manage various versions of packages, corresponding releases, and all the various customizations on the standard software. Infor Enterprise Server offers a comprehensive solution with a version and release management concept.

Advantages of the version and release concept are the following:

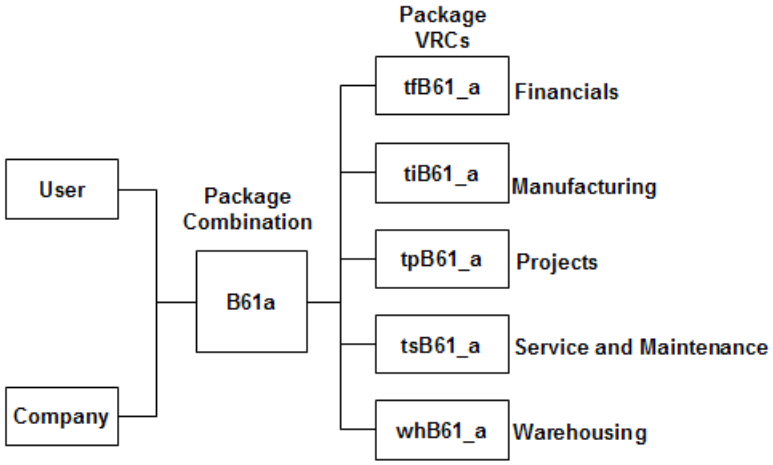
- A flexible development environment
- Flexible management of the following:
  - New versions of the software
  - Patches on the software
  - Customizations on the software
  - Localizations for the software
- Developers can create and test the customizations on the standard software in a separate development environment.
- Operational users will not see new or customized software components until the components are fully tested and released in their own run-time environment.

Package combinations

A package combination is a collection of various LN packages. Every package combination has a unique VRC. The package combination links users to a specific version of the LN software. A package

combination can be linked to one or more users and to one or more companies. However, a package combination can contain only one version of a package, which is identified by a package VRC.

The following figure shows an example of a standard package combination:



Example of a software environment with a standard package combination

This figure shows an example of a standard package combination, which contains a number of standard package VRCs, such as the standard version of LN Financials.

Package VRCs

Each LN package is distributed as a package VRC, which represents a specific version of that package. VRC stands for version release customer code, and it identifies the development stage of the LN software.

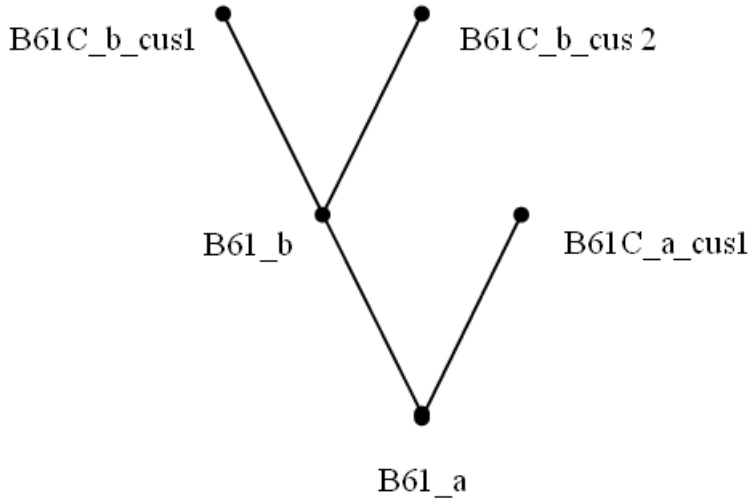
A VRC consists of the following:

- A version code that identifies a major LN software modification
- A release code that identifies a minor LN software modification
- A customer extension code that identifies LN software for a specific customer

A package, for example, LN Financials, can have more than one version. Each version is identified by a specific package VRC. For example, LN Financials can have a package VRC that contains the standard software of LN Financials and a package VRC that contains the customizations on the standard software of LN Financials.

As a rule, a new package VRC is derived from a previous package VRC. To avoid duplication of software components, a new VRC only contains those software components that have changed compared to previous VRCs.

The following figure shows an example of the VRC derivation structure:



An example of a VRC derivation structure

In this example, B61_a is the VRC that contains the standard software. The following two new VRCs are derived from this VRC:

- The VRC B61C_a_cus1 contains a customized version of the standard software
- The VRC B61_b contains a new version of the standard software. From this new standard VRC, two new customized versions are derived:
  - B61C_b_cus1
  - B61C_b_cus2.

  The VRCs B61C_b_cus1 and B61C_b_cus2 contain several customizations on the B61_b standard version.

Package VRC code

A unique identifier identifies package VRCs in LN. LN uses the following format to identify package VRCs, the codes of which are described in the following table:

pp vvvt_rr_gggn

Code	Description
pp	Package code
vvv	Version code

t	Type code. Denotes customizations on standard software. This field is not filled for the standard software. Possible type codes include the following: ■ L Localizations: Customizations for a specific country. ■ B Branch: Customizations for a specific line of business. ■ C Customer: Customizations ordered by the customer, developed by a dealer or partner. ■ O Own: Customizations developed by the customer.
rr	Release code
ggg	Customer code: This field is not filled for the standard software.
n	Sequence code

For example, for the Strangely Brown Chicken company (SBC), the package VRC tc B61O_b_sbc1 represents the first customization on the standard software in the B61_b VRC of LN Common.

Note

Customers can only create new VRCs for their own customizations from the standard software. Infor or a partner delivers all other VRC types.

LN software environment

To create specific software environments, you can use package combinations.

A software environment consists of the following:

- A package combination
- At least one company

You can create a special package combination to customize software components or to create new software components.

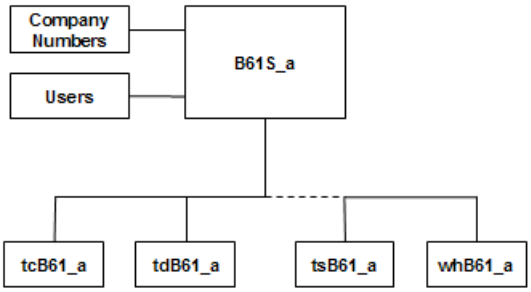
The following table lists the possible software environments in LN:

Software environment	Description
Standard environment	A package combination that contains the standard version of all LN packages.
Runtime environment	A package combination that contains the package VRCs that normal users use at runtime.
Development environment	A package combination that contains package VRCs in which developers can create customizations.

Standard environment

The standard environment contains the standard software of all LN package VRCs of all LN packages. You can use the standard environment to compare customized software with standard software.

The following figure shows an example of the standard environment.



Standard environment

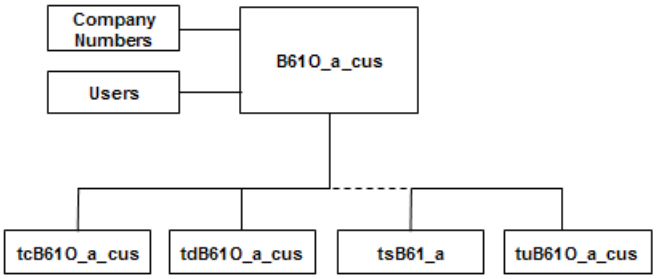
This figure shows an example of a user who is linked to a specific package combination. Use the User Data (ttams1100s000) details session to link the user to a package combination. For more information, refer to “To Create LN users,” in the “User Management” section.

Runtime environment

A runtime package combination can contain some package VRCs that are derived from standard package VRCs. However, not all packages in the package combination have to have the same package VRC. If no customizations exist for a certain package, the standard VRC of this package is used in the run-time package combination. For example, the Service and Maintenance package (tsB61_a) in the following figure is not customized. The other packages are customized and for these the derived package VRC is used in the package combination.

To specify that the package combination is customer defined, make sure you add not only the customer extension (cus) but also the extension O. This letter indicates that the package combination is the customer's own.

The following figure shows an example of a runtime environment.



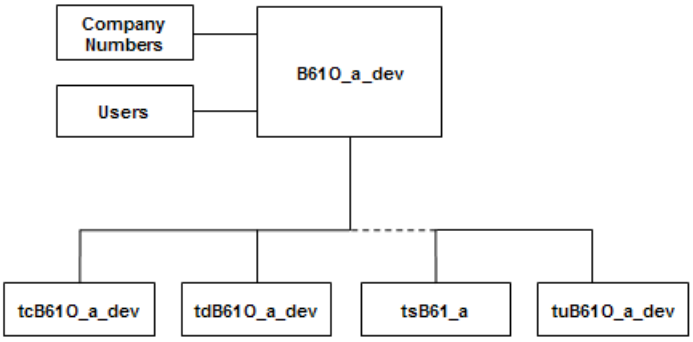
Run-time environment

Development environment

As in the run-time environment, a package combination in a development environment contains package VRCs that are derived from standard package VRCs. Also, not all packages in the package combination must have the same package VRC. If no customizations exist for a certain package, the standard VRC of this package is used in the run-time package combination.

Developers use a development environment to develop new software components or to change existing components.

The following figure shows an example of a development environment:



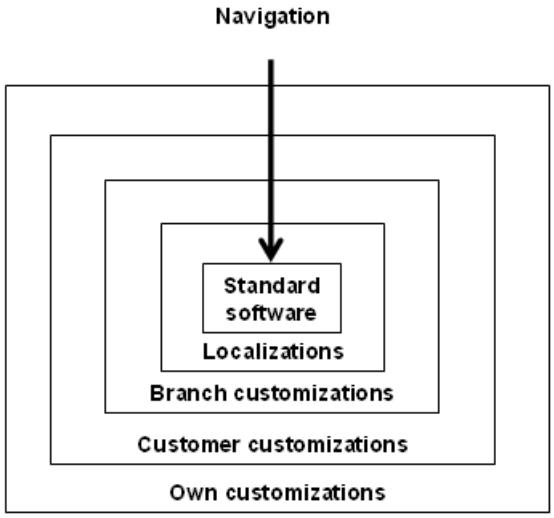
The development environment

The name of this package combination, such as the previous one, contains the character O. In this case, the customer extension is replaced by the extension dev, which stands for development.

VRC derivation

If a user starts a software component, for example a session, the Virtual Machine (VM) searches from the outside to the inside, as shown in the following figure. The Virtual Machine (VM) searches for any customizations on the component made by the customer in the own customization. If no own customizations exist, the Virtual Machine (VM) searches for a customer-specific customization, and so on.

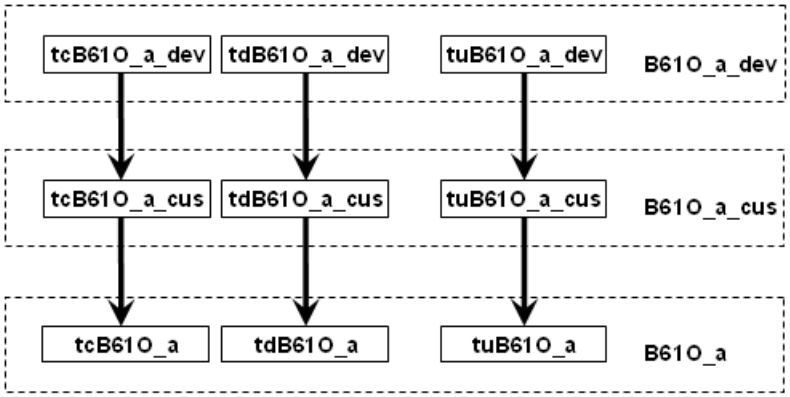
The following figure shows how the Virtual Machine (VM) navigates through the customizations on a component.



Navigation of the Virtual Machine (VM) through the VRC structure

The search path also depends on the package combination and company to which the user is linked. For example, the package VRCs in a development package combination are derived from the VRCs in the run-time package combination, these run-time package combinations were, in turn, derived from standard package VRCs.

The following figure shows the derivation structure of the development package combination.



Package VRC derivation structure

In this previous figure, the VRCs with customer extension dev are derived from the VRCs with the cus extension, which themselves are derived from standard VRCs. If a user works with package combination b61O_a_cus and starts a software component, the Virtual Machine (VM) searches for a customization in the VRC with the cus extension. If such a customization is found, this customized component is started. If no such customization exists, the component in the standard VRC is started.

If a user works with package combination b61O_a_dev and starts a software component, the Virtual Machine (VM) searches for a customization in the VRC with the dev extension. If no such customization is found, the Virtual Machine (VM) continues the search in the cus VRC. If this VRC also does not contain a customized version of the component, the Virtual Machine (VM) starts the component in the standard VRC.

If you create many package VRCs, derived from each other, the derivation structure can get complicated.

The consequences of a cluttered and complicated VRC derivation structure are:

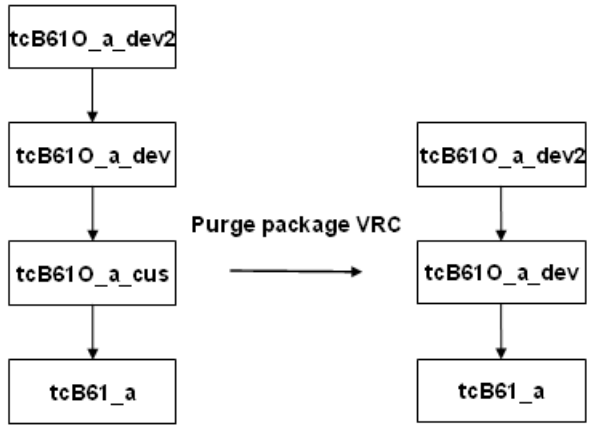
- A decrease in system performance due to the longer derivation structure
- A cluttered directory structure due to a large number of newly-created package VRCs

For this reason, you must regularly purge the VRC derivation structure. Before you can purge a package VRC, you must make sure that the VRC is not connected to a package combination. You can disconnect the package VRC from a package combination with the Packages by Package Combination (ttaad1121m000) session.

Note

If you purge a package VRC, the derived-from structure changes. The user files and fd6.2 files, for the involved users and package combinations, are automatically regenerated so that they contain the new derived-from structure.

The following figure provides an example of a package VRC derivation structure before and after the purge:

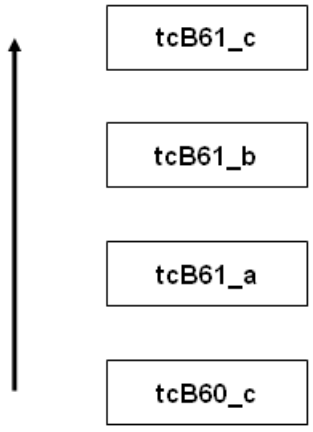


Package VRC derivation structure before and after the purge procedure

Typically, only customer-defined VRCs are purged that contain customized software components that do not belong to a package combination. However, you can also purge the package VRCs that contain the standard software.

For example, the following figure shows the derivation structure of the package VRCs that contain the standard software of the LN Common package. A new standard can contain new functionality, feature packs, bug fixes, and so on.

A new standard is put on top of its predecessor, so software components can be duplicated. Therefore, you must purge the standard package VRCs regularly to remove the duplications.



Derivation structure of the LN Common package

The maximum VRC depth, in other words, the number of VRC that you can stack on top of each other, is 40 VRCs. However, for best results, keep the maximum VRC depth as low as possible.

Contents of a software environment

After you create a software environment, you must fill the environment with software components. In a single-system environment, you can copy the software components from another environment to your new environment. In a multisystem environment, you can use LN 's export/import tool.

Copy software components concept

You can copy software components from one software environment to another. For example, you can customize software components in a dedicated development environment and then copy the components to a new run-time environment or to an environment that is already in use.

If you copy the software components into another environment, you must compile the components in the new environment and then convert the components to the new environment's run-time data dictionary.

The result of this example is that normal users can use the released software components without the need to create new package VRCs or package combinations. Developers can continue to create customizations on the standard software components in their dedicated development VRC.

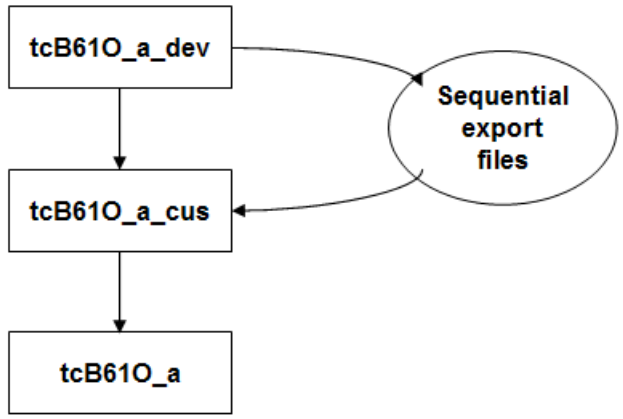
The components that are copied into a new runtime environment are positioned at a higher level. For example, components that are copied from the package VRC tfB60O_b1 into the new run-time environment are placed in the package VRC tfB60O_b2.

For more information, refer to the *Copy software components procedure (p. 144)* .

Export/import software components concept

You can export software components of a package VRC to sequential files. You can import these files into a package VRC in another environment. If you have imported the software components into another environment, you must convert them to the runtime data dictionary of the new environment.

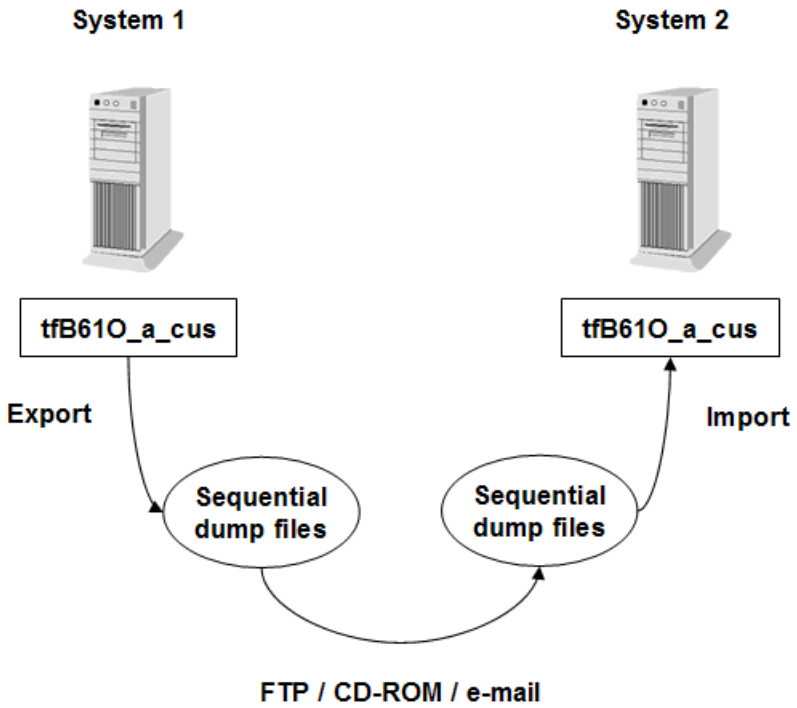
The following figure shows an example of the export/import procedure:



Example of the export/import procedure

Use the export/import method to fill an environment in a multisystem environment. The sequential files that contain the exported software components are sent from one system to another by means of File Transfer Protocol (FTP), a CD-ROM or by e-mail.

The following figure shows the import/ export concept in a multi-system environment:



Import/export procedure in a multi-system environment

FTP is a protocol used to copy files to and from remote computer systems on a network using TCP/IP, such as the Internet. This protocol also enables you to use FTP commands to work with files, such as listing files and directories on the remote system.

For more information, refer to the *Export and import procedure (p. 145)* .

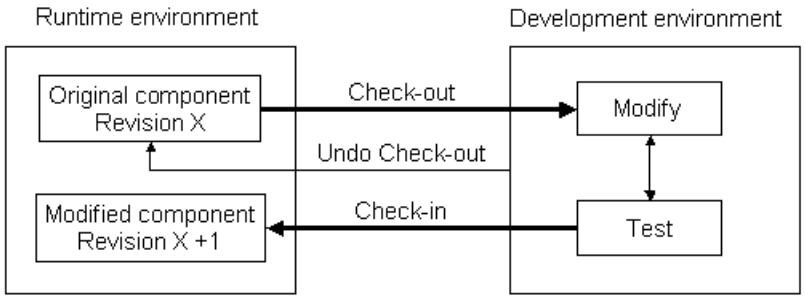
To use the Software Configuration Management system (SCM)

Developers can use the Software Configuration Management System (SCM) to make a copy of the software component and place the component into a dedicated development VRC. The component can then be modified, for example, to fix a bug. If the component is finished and tested, the component can be placed back into the run-time environment. This process is called the check-out and check-in process. The component can still have the same VRC code, but it has a different revision number. Older revisions of the component are still retrievable.

During the check-out period:

- the original version is still available for the regular users in the run-time environment.
- other users cannot modify or check-out the original version.
- other users can test the checked out component. Refer to *SCM Groups (p. 139)* for details.

The following figure shows LN 's Software Control Management (SCM) concept:



LN 's Software Configuration Management (SCM) concept

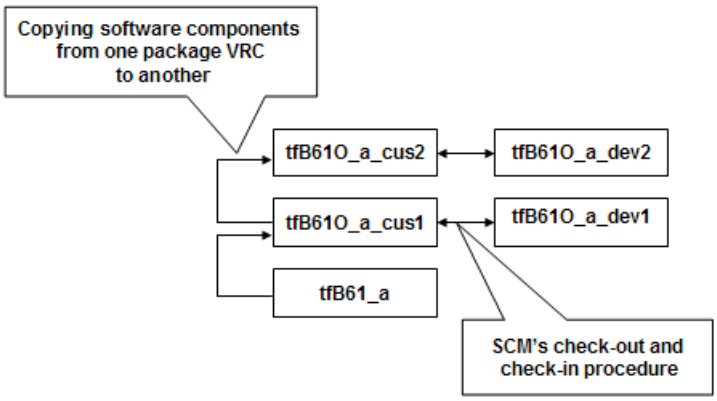
SCM controls only the revisions of components in one VRC. If a component is copied to another VRC, the revision information is no longer available.

Note

There is a fundamental difference between copying software components in a new run-time environment and copying components into a development VRC with LN 's SCM tool. The components, which are copied into a development VRC with the SCM tool, are positioned at the same level as the original and

have the development extension code, for example, tfB60O_b1_dev. This process is known as the check-in and check-out process.

The following figure shows the relation between the SCM procedure and the copy procedure.



Relation between the SCM procedure and the copy procedure

Note

SCM only supports configuration management for the 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.

SCM makes use of the RCS software on the LN Server. Revision Control System (RCS) is freeware software. For more information on SCM, refer to Software Configuration Management in the Infor Enterprise Server Web Help. For details on RCS, refer to http://www.gnu.org/software/rcs/rcs.html .

In earlier versions of Infor Enterprise Server, SCM is only available on UNIX platforms. From version 8.4.2 you can also activate SCM on Windows platforms. For details, refer to solution 230630.

SCM Groups

In SCM groups, developers can share their checked-out runtime components, so they can test each other's components, but they cannot change them.

When you start using SCM, the system automatically generates an SCM group for you. The name of your SCM group is identical to your user name.

Normally, you can only access components that belong to your own SCM group.

To test a component that belongs to another SCM group (a component that was checked-out by another developer), you must modify your login configuration:

1. Open you Windows Client configuration (.bwc file).
2. In the **Command** field of the **BW Configuration Properties** dialog box, fill the **BAAN_SCM_GRP** variable with the name of the desired SCM group. For example, specify

-- -set BAAN_SCM_GRP = peter to test software components that belong to SCM group "peter".

3. Log in using the new configuration.

To return to your own SCM group: log off, remove the *-- -set BAAN_SCM_GRP = ...* setting from your Windows client configuration, and log in again.

To set up the SCM environment

Take the following steps to activate SCM for a VRC:

1. Start the Package VRCs (ttadv1511m000) session, and create a development VRC. The new VRC must be derived from the original VRC. See the previous figure for an example: the tfB60O_b1_dev development VRC is derived from the tfB60O_b1 original VRC.
2. Start the (De)activate SCM and Component Management by Package VRC (ttscm0501m000) session. Select the original VRC (e.g. tfB60O_b1) and click **Activate SCM (and CM)** on the context menu.
3. The Activate SCM and Component Management by Package VRC (ttscm0110s000) session is started automatically. Select the desired development VRC and click **OK**.
4. Log off and log in again to make the changes effective.

Repeat these steps for each VRC for which you want to activate SCM.

Sessions

The following sessions can be started from the Software Configuration Management (SCM) menu:

- (De)activate SCM and Component Management by Package VRC (ttscm0501m000)
- Global Check-In of Components (ttscm1200m000)
- Delete Component Revisions (ttscm1201m000)
- Global Undo Check-Out of Components (ttscm1202m000)
- Copy/Move Checked Out Components (ttscm1203m000)

To use the SCM check-in and check-out procedure

This topic describes how to maintain software components if SCM (Software Configuration Management) is activated. For details on SCM, refer to *To use the Software Configuration Management system (SCM) (p. 138)* .

Procedure

To maintain a software component:

1. check out the software component from the original VRC to a development VRC. To do this, on the appropriate menu click **Check-Out**, or click the corresponding button.
2. If the changes on the software component are tested and completed, check-in the component, so that the changes are made available to other users. To do this, on the appropriate menu click **Check-In**, or click the corresponding button.

Note

This procedure only applies to the main component types such as Program Scripts/Libraries, Forms, Sessions, Table Definitions, Reports and Functions. SCM does not support the 4GL component types such as Labels, Questions, and Messages.

The **Check-Out** and **Check-In** commands are available in the sessions that are used to maintain these main components, such as Reports (ttadv3530m000) and Sessions (ttadv2500m000).

On the appropriate menu of these sessions, you can select **Revisions** to start the Component Revisions (ttscm1501m000) session, to view and maintain a component's revisions.

If necessary the procedure can be reversed by using the undo check-out procedure. To do this, on the appropriate menu click **undo check-out**.

Important!

- You can only check-out components that are stored in the original VRC, that is: the VRC from which the development VRC is derived.
- Make sure that the original VRC is selected as your current VRC, *not* the development VRC. Select **Change Current package VRC of User** in the appropriate menu to change the current VRC.
- If the component is not present in the original VRC, you must first copy it to the original VRC.

Standard software environment procedure

Procedure aim

In this procedure, a software environment is created that you can use to create your own customized software for LN. The environment contains a dedicated package combination and package VRCs for a customization department.

Procedure result and prerequisites

Result

A new software environment with package combinations and package VRCs.

Prerequisites

Make sure that you have at least 10 MB free disk space and 40 MB database space available.

This section describes only the most important steps in the standard software environment procedure. For detailed instructions on how to enter data in the sessions that are mentioned in the procedure steps, refer to the session help.

Note

You can use the procedure on both UNIX and Windows platforms. You can use the ${BSE} notation for both platforms, as well as for the slash (/) and backslash (\) characters.

Points of attention

Points of attention for the package combination:

- The new package combination must contain a VRC for each package, except Tools (tt) and Tools Add-on (tl):
- The Data Director (da), OpenWorld Middleware-enabling (tm), and New Technology (nt) must not be customized. Therefore, you must include *existing* (not your own!) VRCs for these packages. For example, include the highest update or localization VRC in the derivation structure.
- For the remaining packages (except tt and tl) you must include your own VRCs.

Procedure summary

The following list shows the procedure steps and the corresponding sessions.

1. Create new package VRCs - Package VRCs (ttadv1511m000)
2. Create new directories for the software components of the new package VRCs - Directories of Software Components (ttadv1115m000)
3. Create a new package combination - Package Combinations (ttaad1520m000)
4. Link package VRCs to the new package combination - Packages by Package Combination (ttaad1121m000)
5. Link companies to the new package combination - Change Package Combination by Company (ttaad1101m000)
6. Link users to new package combination - Change Package Combinations for Users (ttaad2200m000)
7. Change the authorizations for the developers - Developer Authorization Template (ttams1151m000)

8. Optionally: Initialize Test tool for Business Data Entities (BDEs) - **Initialize Tools (tlcom0200m000)**
9. Authorize normal users for a new or other environment - User Data (ttaad2500m000)

One-step software environment procedure

Procedure aim

In this procedure, a software environment is created that you can use to create your own customized software for LN. The environment contains a dedicated package combination and package VRCs for a customization department.

You cannot use this procedure to maintain existing environments. To maintain existing environments, you must use the sessions discussed in the *Standard software environment procedure (p. 141)* . The prerequisites for and result of this procedure are identical to those of the Standard software environment procedure.

Procedure result and prerequisites

Result

A new software environment with package combinations and package VRCs.

Prerequisites

Make sure that you have at least 10 MB free disk space and 40 MB database space available.

This section describes only the most important steps in the one-step procedure. For detailed instructions on how to enter data in the sessions that are mentioned in the procedure steps, refer to the session help.

Note

You can use the procedure on both UNIX and Windows platforms. You can use the ${BSE} notation for both platforms, as well as for the slash (/) and backslash (\) characters.

Points of attention

Points of attention for the package combination:

- The new package combination must contain a VRC for each package, except Tools (tt) and Tools Add-on (tl):
- The Data Director (da), OpenWorld Middleware-enabling (tm), and New Technology (nt) must not be customized. Therefore, you must include *existing* (not your own!) VRCs for these packages. For example, include the highest update or localization VRC in the derivation structure.

- For the remaining packages (except tt and tl) you must include your own VRCs.

Procedure summary

The following list shows the procedure steps and the corresponding sessions.

1. Create the software environment - Create New Package Combination / VRCs (One Step) (ttaad1222m000)
2. Optionally: Initialize Test tool for Business Data Entities (BDEs) - **Initialize Tools (tlcom0200m000)**

Copy software components procedure

Procedure aim

An environment must be filled with software components. To fill a new environment with software components, you can select components from an already existing environment and copy them into the new one.

Procedure result and prerequisites

Result

Normal users, for example, can use the new environment as a run-time environment. Developers can also use this environment as a development environment to create new components or to customize existing components.

Prerequisites

Make sure that the environment is created successfully. You can create an environment through the *Standard software environment procedure (p. 141)* , or through the *One-step software environment procedure (p. 143)* .

The copy software components procedure is usually used for a single system environment. If you use more systems, use the *Export and import procedure (p. 145)* .

Procedure summary

The following list shows the procedure steps and the corresponding sessions.

1. Copy components in a software environment - Copy Software Components to New Package VRC (ttadv6265m000)
2. Create the run-time data dictionary for labels- Compile Labels (ttadv1243m000)

3. Create the run-time data dictionary of the components in the target package combination- Create Runtime Data Dictionary (ttadv5210m000)
4. Compile the program scripts and library scripts of the copied components - Compile Program Scripts / Libraries (ttadv2230m000)
5. Compile the reports of the copied components - Compile Reports (ttadv3240m000)
6. Create the run-time data dictionary of the forms and menus - Create Runtime Data Dictionary (Forms and Menus) (ttadv5214m000)
7. Install Infor Web Help for the components in the target environment

Export and import procedure

With the import/export procedure, the software components in a package VRC are exported to sequential files and imported into a package VRC in another environment. If you import the software components into another environment, you must convert them to the runtime data dictionary in the new environment.

Procedure aim

An environment must be filled with software components. You can use the export/import procedure to fill an environment with software components.

Procedure result and prerequisites

Result

The process results in a new environment that normal users, for example, can use as their runtime environment. Developers can also use this environment as a development environment to create new components to customize existing components.

Prerequisites

Make sure that the environment is created successfully. You can create an environment through the *Standard software environment procedure (p. 141)* , or through the *One-step software environment procedure (p. 143)* .

Procedure summary

The following list shows the procedure steps and the corresponding sessions.

1. Export components from a software environment to sequential files - Export Data Dictionary (ttiex1280m000)
2. Only if the source and the target environments are located on different systems: transfer the sequential files to the target system

3. Import components from the sequential files into the target environment - Import Data Dictionary (ttiex1285m000)
4. Create the run-time data dictionary for labels- Compile Labels (ttadv1243m000)
5. Create the run-time data dictionary of the components in the target package combination- Create Runtime Data Dictionary (ttadv5210m000)
6. Compile the program scripts and library scripts of the copied components - Compile Program Scripts / Libraries (ttadv2230m000)
7. Compile the reports of the copied components - Compile Reports (ttadv3240m000)
8. Create the run-time data dictionary of the forms and menus - Create Runtime Data Dictionary (Forms and Menus) (ttadv5214m000)
9. Install Infor Web Help for the components in the target environment

Note

The compilation of program scripts/libraries and reports, and the creation of the runtime dictionary for forms and menus is only necessary if you did not export objects/dumps for these components.

Purge a package VRC derivation structure procedure

Procedure aim

If you create many own package VRCs, derived from each other, the derivation structure can get very long.

The consequences of a cluttered VRC structure are:

- A decrease in system performance due to the longer derivation structure
- A cluttered directory structure due to a large number of newly-created package VRCs

The maximum depth of the package VRC tree is 40. Purging a package VRC reduces the depth of the package VRC tree, which makes it possible to add a new package VRC to the tree.

When you purge a package VRC:

- software components that are no longer used, for example, because of a new customization on the same component in a more recent package VRC, are deleted.
- The components that are still in use are copied to the package VRC that is derived from the purged VRC.

Procedure result and prerequisites

Result

As a result of this procedure, a shorter derivation structure is created and, therefore, better performance is achieved. In addition, the directory structure on your operating system, for example, UNIX or Windows, is more transparent.

Prerequisites

The status of the package VRC must be Expired.

Procedure summary

The following list shows the procedure steps and the corresponding sessions.

1. Change the status of the package VRC to Expired - Package VRCs (ttadv1511m000)
2. Purge the package VRC - Purge Entire Package VRC (ttadv6255m000)
3. Recreate the user files - Convert to Runtime DD (ttams2201m000)

Chapter 8
LN Software Maintenance

8

PMC introduction

To help you manage software updates to your LN system, Infor offers the Product Maintenance and Control (PMC) Tool – an efficient, highly effective tool for managing functional software updates (Feature Packs) and other software updates (Individual Solutions).

The PMC module manages the installation of Feature Packs and Individual Solutions. The PMC module is delivered to all customers as part of the master CD-ROM of LN. The PMC module is used to check software updates for completeness and customization interference.

Without a tool such as PMC, software updates could be installed irrespective of other software updates that were already installed. Technically, an update that contains an earlier version of a software component than the version already present on your system, could be installed. In this case, you lose the updates contained in the original software components.

The PMC module contains two sub-modules.

- PMC distributor
- PMC recipient

PMC Benefits

The PMC module is an easy-to-use tool that enables you to install, configure, and run new software with confidence. The PMC module simultaneously addresses a wide range of software update challenges, including:

- Automatic checks for updates for any previous dependencies, to ensure that these exist on the system on which you install the update.
- Analysis of potential conflicts with system customizations at installation time, which enables the user to identify any conflicting customizations.
- Option to maintain copies of all previous software components. This feature enables the user to roll back updates easily to any point in history.
- Overview of all Feature Packs and Individual Solutions, that are installed.

Note

A customization component does not have a ‘maintenance date’ in its object. Therefore, if you deliver customization components to a customer, whose maintenance license has expired, the customer can still deploy the components.

PMC overview

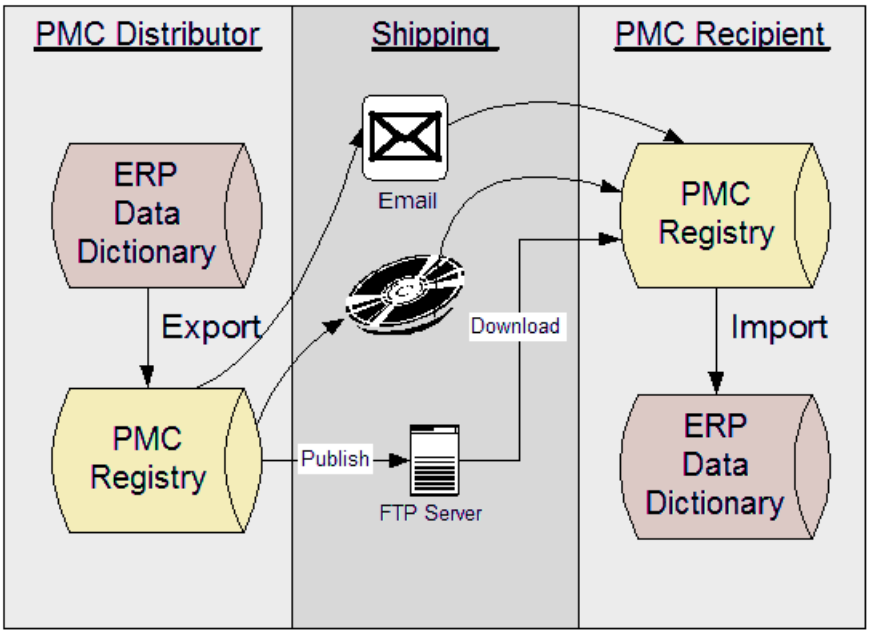
This chapter provides an overview of the Product Maintenance and Control (PMC) module.

This chapter contains the following sections:

- PMC Architecture
- Where to find the PMC module
- Dependencies
- Individual Solutions
- Collections
- Feature Packs and Patches
- PMC Distributor functionality
- PMC Recipient functionality

PMC Architecture

The following figure illustrates the architecture of the PMC module:



The software developer creates or updates software components in the *Data Dictionary* of the developers ERP system. Software is packaged by the PMC distributor and stored in the PMC Registry. The software is published on an FTP server. Software can also be shipped on other media, such as CD-ROM, or can be sent by e-mail.

The customer can download the software from an FTP Server, or can order a software CD-ROM. PMC recipient builds or updates the PMC Registry on the customer's ERP system. Software in the customer's PMC Registry can be imported in the customer's data dictionary.

Where to find the PMC module

The PMC module is part of Infor Enterprise Server (Tools). The software supplier is the distributor of the solutions, and the user or customer of the applications is the recipient. The PMC module contains functionality and sessions for both the distributor and recipient role.

To navigate to the recipient part of the PMC module, complete these steps:

1. Log on to LN.
2. On the ERP Menu Browser, click **Tools --> Software Installation --> Miscellaneous --> Recipient**.
3. Click **Setup, Operational, and Miscellaneous**.

Alternatively, complete these steps to navigate to the recipient part of the PMC module:

1. Log on to LN.
2. On the ERP Menu Browser, click **Tools --> Application Development --> Product Maintenance and Control --> Recipient**.
3. Click **Setup, Operational and Miscellaneous**.

To navigate to the distributor part of the PMC module, complete these steps:

1. Log on to LN.
2. On the ERP Menu Browser, click **Tools --> Application Development --> Product Maintenance and Control --> Distributor**.
3. Click **Setup and Operational**.

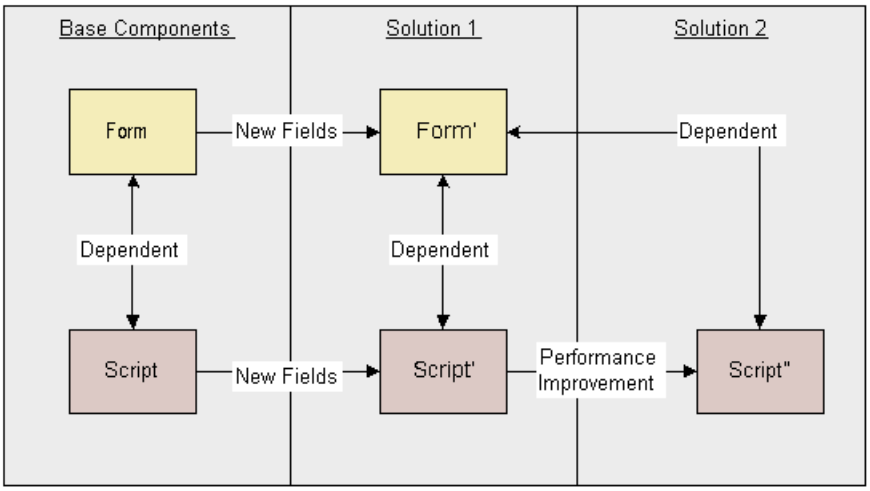
Dependencies

When the PMC recipient installs software updates, the recipient checks the presence of the required depending solutions.

Three dependency types exist:

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

The following figure shows a pre-requisite dependency: solution 1 must be installed before solution 2.



The following section describes a customer scenario.

Example

A dependency requires you to install the previous version of a software component first. Suppose that you installed LN from the master CD-ROM without customizations or solutions. You receive an updated version of the Copy Sales Order (tdsls4201s000) session, which contains the session object, script, and forms that are interdependent of each other.

You decide not to install the dump and, a few months later, you discover that a change is required in the script to correct a problem that is keeping your sales force from selling an item. The change is sent as a dump that contains only the session object and script. You can install this dump only if you installed the previous version of the software component, because the dump requires the proper form of the first solution. PMC automatically installs the previous solution.

Suppose you deliberately decided not to install a solution, for example, because you do not know if the existing version of the Copy Sales Order (tdsls4201s000) session was customized. The PMC module provides protection against this as well. In this case PMC informs you that the current version is customized. You cannot overwrite software, because PMC uses update VRC levels and maintains copies of all previous software components. This setup enables you to roll back updates easily to any point in history.

Individual solutions

A PMC Distributor creates and publishes individual solutions.

Individual solutions created for standard products of Infor are available for download 24 hours a day, seven days a week, from the Infor Support Web site http://www.infor.com/inforxtreme for Infor-owned products.

You can only download these solutions one by one.

After you download and scan a solution, the PMC recipient checks the dependencies with other solutions. In some cases, other dependent solutions might be missing at the installation system and must be downloaded and scanned, as well. The process to install individual solutions can take a long time if many dependent solutions must be downloaded and scanned.

You can use the Multilevel Download functionality to download all the dependent solutions, without user interaction, to simplify and speed up the download process.

Collections

Collection dumps enable customers to update their system with the latest software.

Collections are groups of all individual solutions that are released in a particular period.

These individual solutions are gathered at the PMC distributor side in a single dump file.

Collections for standard products of Infor are published on an FTP server.

You can scan collections in a single action. While you scan the collection with the Scan Solution/Patch File (ttpmc2200s000) session, the dump file is split into all individual solutions that were included in the collection.

Afterwards, you can install the individual solutions.

You can use collection dumps in two ways: for pro-active installation or corrective installation.

- **Pro-active installation**
  The user installs all individual solutions of the collection. The user runs the most up-to-date version of the software, including the latest released fixes.
- **Corrective installation**
  The individual solutions of the collection are not installed. The solutions have status *Available* in the PMC registry. If the user runs into a problem, and a solution is advised for this problem, the user can install that solution immediately. All dependent solutions will be installed automatically.

Feature Packs and patches

Periodically, individual solutions can be bundled into Feature Packs or patches. In general, these types of updates contain solutions created in a larger time period than a collection. In the PMC Tool, the term ‘patch’ is used both for Feature Packs and patches. The patch entity is known at both the PMC distributor and PMC recipient side. Feature Packs and patches are an indivisible set of solutions. You cannot install or uninstall individual solutions that belong to a Feature Pack or a patch by the PMC recipient. You can only install or uninstall entire Feature Packs and patches. Dependencies between Feature Packs and patches can exist.

The subsequent sections describe the justification, characteristics, and differences of Feature Packs and patches.

Justification of Feature Packs and patches

During the life cycle of a product, in general, two types of changes are implemented in the product:

- Corrective fixes for defects
- Functional enhancements to further enrich the product

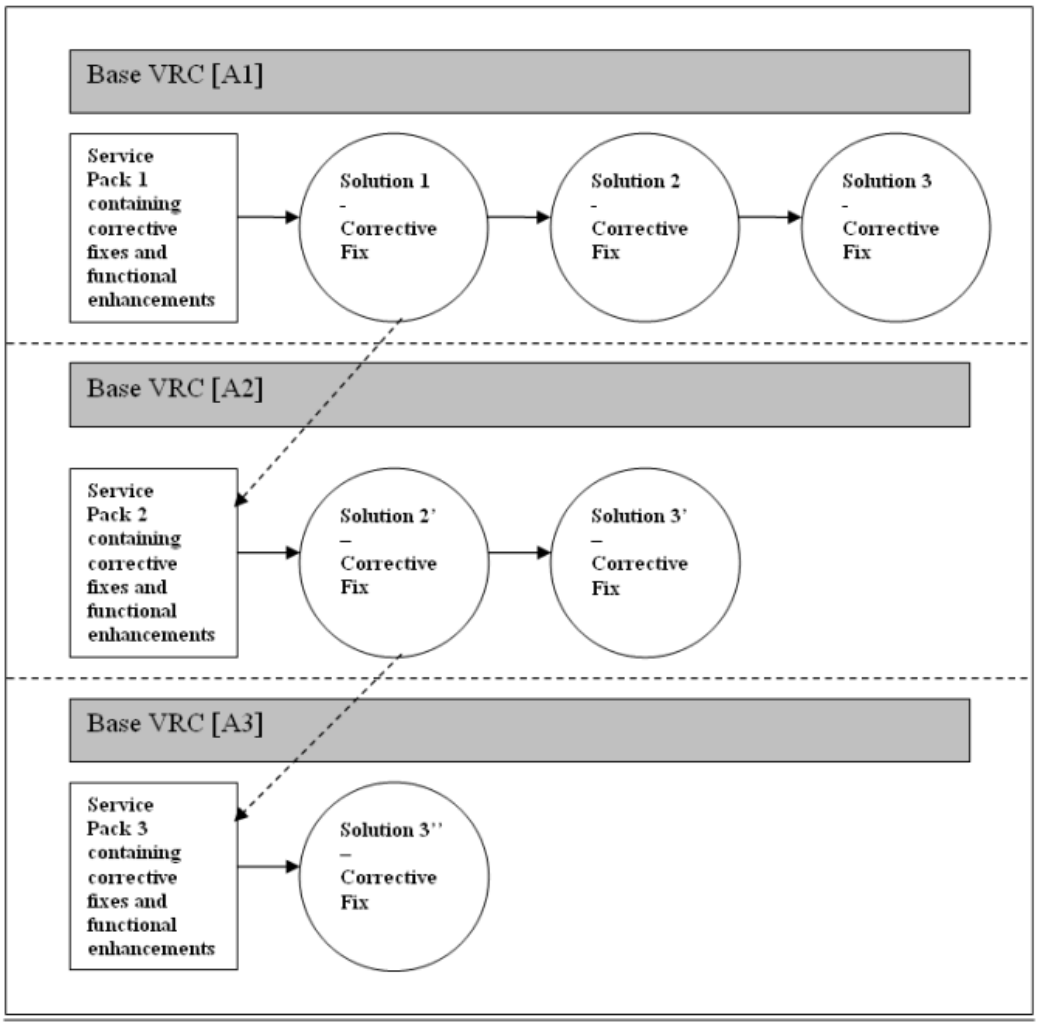
Functional enhancements are often accompanied by changes in the data model, changes in the user interface, and changes in the behavior of the application. In general, functional enhancements require a careful preparation before installation in an operational environment. PMC generates dependencies between solutions, that contain the same components to guarantee that the solutions are installed in the correct order and no necessary solutions are skipped. In some cases, users must first install a number of functional enhancements before the users can install a corrective fix for a relative small defect. Installation of the functional enhancements can be required at unexpected and undesirable moments.

Feature Packs enable you to separate functional enhancements from the flow of corrective fixes. This separation enables users to adopt and implement functional enhancements in a more controlled way. Patches do not offer this possibility.

Feature Packs

Feature Packs enable you to separate corrective fixes from functional enhancements. Multiple parallel maintenance baselines exist for the product. Every new baseline is linked to a unique base VRC. Functional enhancements are, in principle, only implemented at the start of a new baseline.

The Feature Packs mechanism is as follows:



Feature Pack 1 is linked to base VRC A1. The Feature Pack contains a number of corrective fixes and some functional enhancements. Corrective maintenance is performed for this Feature Pack, which results in solutions 1, 2, and 3 in the previous figure.

In parallel, the software distributor develops Feature Pack 2, which is linked to base VRC A2. Whenever possible, the distributor immediately ports and includes corrective solutions created on top of Feature Pack 1 in Feature Pack 2. Refer to solution 1 and the dashed arrow in the previous figure. These ported solutions are registered in Feature Pack 2 as so-called obsolete solutions.

The same process repeats when the distributor develops Feature Pack 3, which is linked to base VRC A3.

After the release of Feature Pack 2 and 3, the distributor also creates corrective solutions on top of these Feature Packs.

When Feature Pack 2 is being installed at a recipient, the PMC Tool checks whether all solutions that were installed on top of Feature Pack 1 are also available for Feature Pack 2.

These solutions can be available in two different shapes for Feature Pack 2:

- **As an obsolete solution**
  Included in Feature Pack 2. Refer to solution 1 and the dashed arrow in the previous figure. These obsolete solutions are integrated in Feature Pack 2 and do not have a separate physical solution dump.
- **As an individual solution**
  Available on top of Feature Pack. Refer to solutions 2’ and 3’ in the previous figure. These so-called missing solutions are not included in Feature Pack 2 and do have a physical solution dump.

  Note: If Feature Pack 1 is installed in multiple update VRCs, the PMC Tool performs the missing solution check in all update VRCs, that contain Feature Pack 1.

By default, the PMC Tools will refuse to install Feature Pack 2 if not all solutions installed on top of Feature Pack 1 are also available for Feature Pack 2. Otherwise, old defects would come back after you install Feature Pack 2. The distributor is responsible for ensuring that all solutions of the preceding Feature Pack are also available for the subsequent Feature Pack before publishing the subsequent Feature Pack.

Note

You can change this default behavior, so that you can install Feature Pack updates without additional individual solutions. For details, refer to *To install a 'clean' Feature Pack (p. 156)* .

Feature Packs are cumulative. Every new Feature Pack also includes all preceding Feature Packs. This enables you to skip the installation of Feature Packs at the PMC recipient. For example, you can immediately upgrade from Feature Pack 1 to Feature Pack 3 without installing Feature Pack 2 in between. The installation process is optimized in such a way that only components that are changed as compared to the current installed Feature Pack will be installed.

You can install Feature Packs in an existing update VRC. Therefore, creating a new update VRC is unnecessary. As a result, the VRC structure at a recipient system can remain unchanged. However, the base VRC linked to the update VRC will change. If desired, you can, of course, also install a new Feature Pack in a new update VRC.

To install a 'clean' Feature Pack

When you update an existing environment with a Feature Pack via PMC or via the Installation Wizard, PMC requires that all solutions, that were installed on top of a previous Feature Pack, are installed for the new Feature Pack as well. PMC requires these additional solutions to prevent that old problems come back after installation of the Feature Pack. This also applies if you install the Feature Pack in another, or even in a new, Update VRC. This is because you can upgrade your companies to the Package Combination of that Update VRC and subsequently you might see old problems again.

In some situations this functionality is too strict, for example when you want to develop customizations for different customers on different Feature Pack levels.

To change this default behavior, you must set the PMC_IGNORE_SOL environment variable to 1. You can now install a 'clean' Feature Pack, without taking care of the content of already installed Feature Packs and additional solutions.

You can specify the environment variable in different ways:

- In the command field of the BW configuration file (-set PMC_IGNORE_SOL=1). In this way the setting only applies to the user that uses this configuration.
- In the $BSE/lib/bse_vars file (add the line PMC_IGNORE_SOL=1). If you use the Installation Wizard you can add this line when you are prompted to change the bse_vars configuration file. In this way the setting applies to all users.

Note

- It is recommended to remove the PMC_IGNORE_SOL setting after each Feature Pack installation.
- If you install the Feature Pack in an already existing Update VRC, the solutions that were installed in that Update VRC for the previous Feature Pack, will also be installed for the new Feature Pack. So, the possibility to ignore additional solutions only applies to solutions of other Update VRCs.

Example

You want to install Feature Pack 2, which includes solutions 133, 146 and 154 as obsolete solutions already.

The following table shows, for some Update VRCs with different Feature Packs and additional solutions installed:

- what happens if PMC_IGNORE_SOL is not set.
- what happens if PMC_IGNORE_SOL is set.

	Update VRC		
	B61U_a_stnd	B61U_a_fp1	B61U_a_fp2
Feature Pack installed	-	FP1 including obsolete solutions 133 and 146	-
Solutions installed	133 146 154 168	154 168 172	-

Action	Install FP2 in an 'FP0' Update VRC with some individual solutions installed.	Install FP2 in an Update VRC with FP1 and some additional solutions installed.	Install FP2 in a new Update VRC.
Result if PMC_IGNORE_SOL is *not* set	Fails unless additional solutions 168 and 172 are present. Reason for failure: Solution 168 was already installed in this Update VRC, and solution 172 was installed on top of FP1 in Update VRC B61U_a_fp1.	Fails unless additional solutions 168 and 172 are present. Reason for failure: Both solutions were already installed in this Update VRC.	Fails unless additional solutions 168 and 172 are present. Reason for failure: Solutions 168 and 172 were installed on top of FP1 in Update VRC B61U_a_fp1.
Result if PMC_IGNORE_SOL is set to 1	Fails unless additional solution 168 is present. Reason for failure: Solution 168 was installed in this Update VRC, so it is needed on top of FP2. Solution 172 can be ignored now.	Fails unless additional solutions 168 and 172 are present. Reason for failure: Both solutions were installed in this Update VRC, so they cannot be ignored.	Succeeds. Reason for success: PMC can ignore all solutions, because it is a new Update VRC.

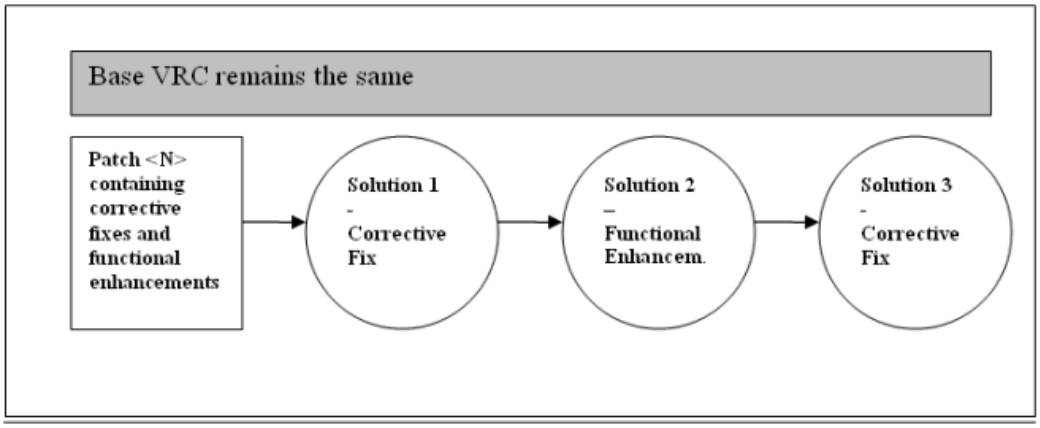
Important!

Do not set PMC_IGNORE_SOL to 1 if you install the Feature Pack in an Update VRC and Package Combination, and you plan to upgrade your companies to this Package Combination. This can result in old problems coming back or, even worse, loss of data if additional solutions on top of the previous Feature Pack introduced new table fields.

Patches

Patches do not enable you to separate corrective fixes from functional enhancements. Only one maintenance baseline of the product exists. Changes are always implemented in the latest version of a software component. The base VRC remains the same. Functional enhancements will be included in the chain of depending corrective solutions.

The Patch mechanism is as follows:



To install solution 3 in the this picture, you must first install solution 2, which contains a functional enhancement.

Patches are not cumulative, which means you cannot skip the installation of Patches at a recipient's location. For example, before you install Patch 3, you always must install Patch 2 in advance. You cannot immediately upgrade from Patch 1 to Patch 3.

Distributor's Policy

The software distributor decides if Feature Packs or Patches are being applied. A mix of both types is also possible:

- When applying Feature Packs, the distributor can decide to release a functional enhancement as an individual PMC solution on top of one of the released Feature Packs.
- When applying Feature Packs, the distributor can decide to periodically also release Patches on top of the Feature Pack.

When applying Feature Packs, the distributor's policy determines how many base VRCs are maintained in parallel and how long the base VRCs are maintained.

PMC Distributor functionality

The software developer generates additional software and makes software changes on the existing LN software packages.

The supplier of the LN software uses the PMC Distributor functionality to manage the software updates and prepares those updates for delivery to the customers.

The following is a process overview of the PMC Distributor module.

- **Create individual solutions**
  - **a.** Define a unique identifier for the solution and a brief description.

   b. Link one or more software components to the solution.
   c. Define or generate the dependencies on other solutions.
   d. Export the solution, the software dumps are now created.
   e. Release the solution, the solution is now ready for delivery.
- **Create collections**
   a. Define a unique identifier for the collection and a brief description.
   b. Link one or more individual solutions to the collection.
   c. Export the collection.
   d. Release the collection.
- **Create patches**
   a. Define a unique identifier for the patch and a brief description.
   b. Link one or more individual solutions to the patch.
   c. Define or generate the dependencies on other patches.
   d. Validate the patch.
   e. Export the patch.
   f. Release the patch.
- **Create Feature Packs**
   a. Define a new base VRC for the Feature Pack.
   b. Define a unique identifier for the Feature Pack and a brief description.
   c. Define the dependency on previous Feature Packs.
   d. Generate the Feature Pack, in other words, link all components in a specified VRC to the Feature Pack.
   e. Validate the Feature Pack.
   f. Export the Feature Pack.
   g. Release the Feature Pack.
- **Additional distributor functionality**
   - Mutilevel export.
   - SCM integration.
   - Maintenance History.

PMC Recipient functionality

Customers use the PMC Recipient module to install the LN software updates on their ERP system.

The following is an overview of the PMC Recipient module.

- **Download solutions**
  You can download solutions from the FTP server or copy the solutions from a medium such as a CD-ROM.

- **Scan and connect solutions**
  The downloaded solution files can be scanned. In other words, the user must extract and store the solutions into the PMC registry. The extracted solutions are also connected to an update VRC.
- **Process solutions**
  a. Check to install. Report the following: Pre- and post-installation instructions, customized components, and missing dependent solutions.
  b. Install the solutions: Store the software components in the data dictionary, report additional post-installation instructions.
- **Additional recipient functionality**
  - Uninstall
  - Mutilevel download
  - Compare installed solutions
  - Solution History
  - PMC cleanup
  - Copy PMC registry
  - View installation runs.

PMC distributor procedure

This chapter describes the PMC distributor procedures. This chapter contains the following sections:

- Setup
- To create updates
- Feature Pack development
- Customizations

Setup

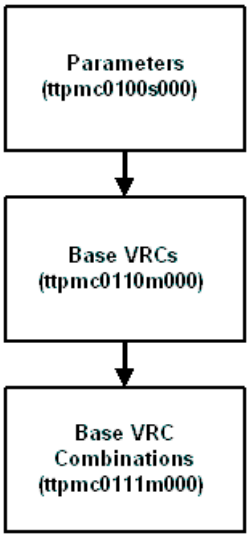
This section describes the procedures for the update distributor to set up the PMC administration.

The setup consists of the following sections:

- Setup procedure
- Parameters
- Base VRCs
- Base VRC combinations

Setup procedure

This section provides a flowchart that describes the steps of the distributor set-up procedure. The subsequent sections provide a detailed explanation of the procedure.



Distributor Set-up Procedure

Parameters

The very first step in the distributor set-up procedure is to define the PMC parameters.

The following two group boxess are available:

- **Recipient**
  Even if you do not use the PMC recipient part of the PMC module, for technical reasons, you must fill in the parameters in this group box.
- **Distributor**
  The PMC distributor part of the PMC module uses these parameters, and therefore you must fill in these parameters.

  Most of the parameters are directories on the operating system on which the solution dumps will be stored.

For details, refer to the Parameters (ttpmc0100s000) session.

Note

Because the definition of the parameters is a once only step, you must be aware of the impact if you change the parameters afterwards.

If you change the directories on the operating system, you must move the directories and files according to the new directory specifications.

base VRCs

You must define a base VRC before you can create and link solutions, collections, patches, and Feature Packs to the base VRC.

A base VRC is an administrative identifier for a software product and is not a physical VRC, but must always be connected to a physical VRC, which contains the software components.

The PMC distributor should connect a base VRC to an export VRC, the PMC recipient should connect a base VRC to an update VRC.

Note

A Feature Pack always starts a new base VRC.

For details, refer to the Base VRC's (ttpmc0110m000) session.

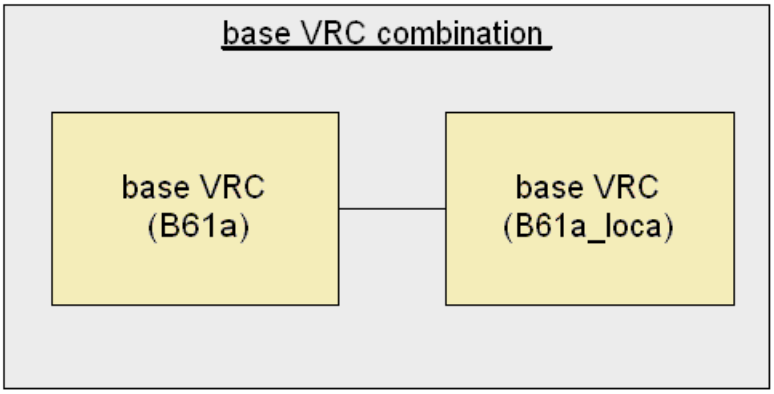
base VRC combinations

Base VRC combinations are required to define relations between base VRCs.

Any base VRC must always be a member of a base VRC combination, even if no relations to other base VRCs exist.

If you have not linked a base VRC to a base VRC combination, you cannot create updates.

The following example shows a base VRC combination with two base VRCs with a relation:



For details refer to the Base VRC Combinations (ttpmc0111m000) session.

To create updates

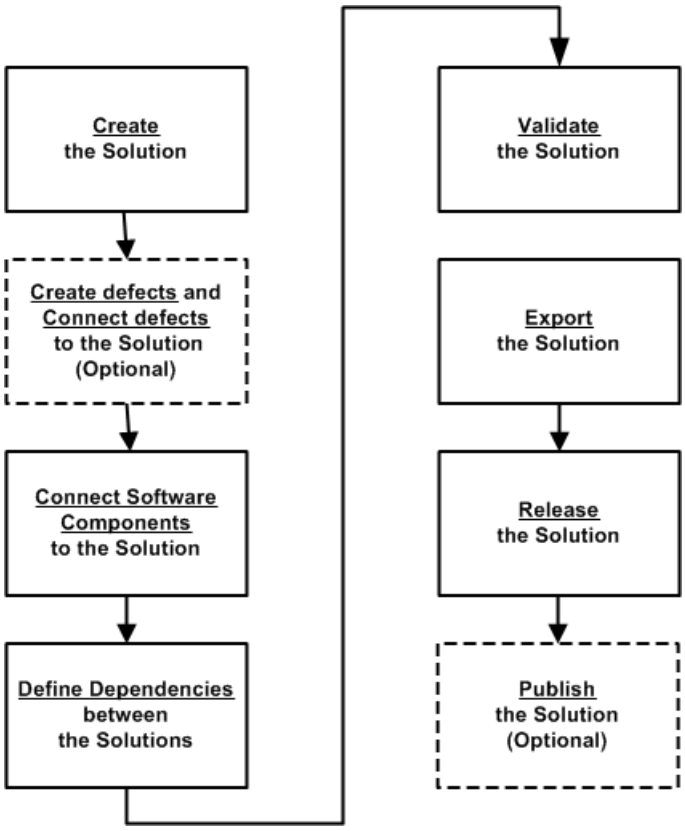
This section describes the following PMC distributor procedures to create software updates:

- To create solutions
- To create collections
- To create patches

- To create Feature Packs

To create solutions

This section describes the procedure steps to create solutions:



You can create or maintain solutions in the Solutions (ttpmc1100m000) session.

This session is the central session from which you can maintain all aspects of solutions.

The initial status of a solution after creation is In Progress.

If you create a solution, you must take the following steps:

- **Create defects and connect the defects to the solution**
  You can create and connect the defects in the Defects by Solution (ttpmc1110m000) session.
  This step is optional.
- **Connect software components to the solution**
  To connect software components to the solution, use the Components by Solution (ttpmc1520m000) session and the Component by Solution (ttpmc1120s000) session.
  You can also connect software components to a solution in the Connect Components to PMC Solution (ttpmc1221m000) session.

- **Define dependencies between solutions**
  To define dependencies between solutions, run the Dependencies (ttpmc1140m000) session.
- **Validate the solution**
  To validate a solution, you must click **Validate Solution** on the appropriate menu in the Solutions (ttpmc1100m000) session. This option starts the Validate Solution (ttpmc1404s000) session. This session checks, for example, if components are available, or if components are not compiled in debug mode, and so on.
- **Export the solution**
  To export a solution, you must use the Export Solution/Patch (ttpmc1200s000) session.
- **Release the solution**
  To release a solution, you must change the **Status** from Exported to Released in the Solutions (ttpmc1100m000) session.

  You can only release a solution if the **Status** of the solution is Exported.
- **Publish the solution**
  To publish a solution, select the **Published** check box in the Solutions (ttpmc1100m000) session.

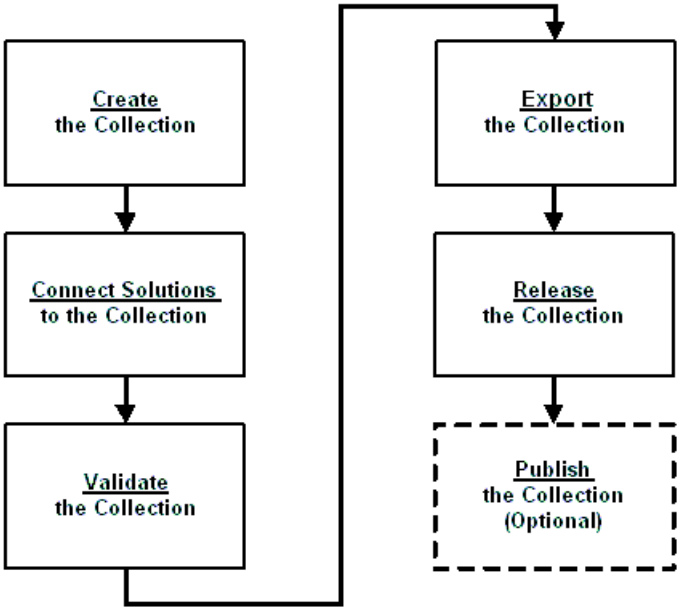
  You can only publish a solution if the **Status** of the solution is Released.

  This step is optional.
- **View or print solutions**
  To view solutions, you must use the Components by solution (ttpmc1521m000) session.

  To print solutions, you must use the Print Components by Solution (ttpmc1420m000) session.
- **View the solution history.**
  To view the solution history, you must use the Maintenance History (ttpmc1560m000) session.

To create collections

This section shows the procedure steps to create collections:



You can create or maintain collections in the Collections (ttpmc1503m000) session.

This session is the central session from which you can maintain all aspects of collections.

The initial status of a collection after creation is In Progress.

After you create a collection, take the following steps:

- **Connect solutions to the collection**
  You can connect solutions to a collection in the Solutions by Collection (ttpmc1151m000) session.
- **Validate the collection**
  You can validate collections in the Validate Collection (ttpmc1402s000) session.
- **Export the collection**
  You can export a collection in the Export Collection (ttpmc1210s000) session.
- **Release the collection**
  To release a collection, you must change the **Collection Status** from Exported to Released in the Collections (ttpmc1101s000) session.

  You can only release a collection if the **Collection Status** is Exported.
- **Publish the collection**
  To publish a collection, select the **Published** check box in the Collections (ttpmc1101s000) session.

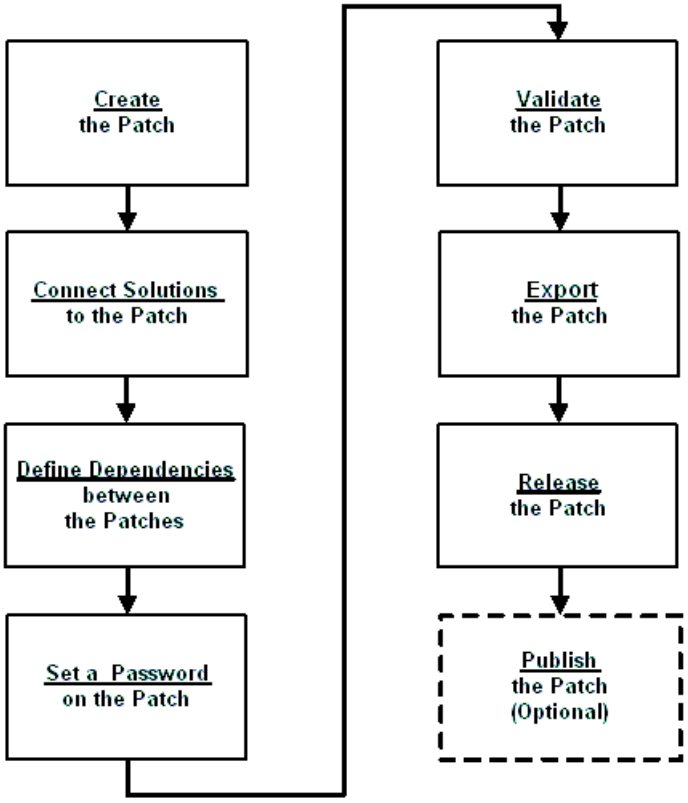
  You can only publish a collection if the **Collection Status** is Released.

This step is optional.

- **View the collection history**
  To view the collection history, use the Maintenance History (ttpmc1560m000) session.

To create patches

This section describes the procedure to create patches:



You can create or maintain patches in the Patches (ttpmc1501m000) session.

This session is the central session from which you can maintain all aspects of patches.

You must clear the **New Base VRC** check box. Otherwise, the patch is a Feature Pack.

The initial status of a patch after creation is In Progress.

After you create a patch, you can take the following steps:

- **Connect solutions to the patch**
  To connect solutions to a patch, you must use either the Solutions (ttpmc1100m000) session or the Generate Patch (ttpmc1250s000) session.

- **Define dependencies between patches**
  To define dependencies between patches, you must use the Dependencies (ttpmc1140m000) session.
- **Set or change the password of the patch**
  To set or change the password of a patch, you must use the Password by Patch (ttpmc1105s000) session.
- **Validate the patch**
  To validate a patch, you must use the Validate Patch (ttpmc1401s000) session.
- **Export the patch**
  To export a patch, you must use the Export Solution/Patch (ttpmc1200s000) session.
- **Release the patch**
  To release a patch, you must change the **Status** from Exported to Released in the Patches (ttpmc1501m000) session.

  You can only release a patch if the **Status** is Exported.
- **Publish the patch**
  To publish a patch, select the **Published** check box in the Patches (ttpmc1501m000) session.

  You can only publish a patch if the **Status** of the patch is Released.

  This step is optional.
- **View the patch history**
  You can view the patch history in the Maintenance History (ttpmc1560m000) session.

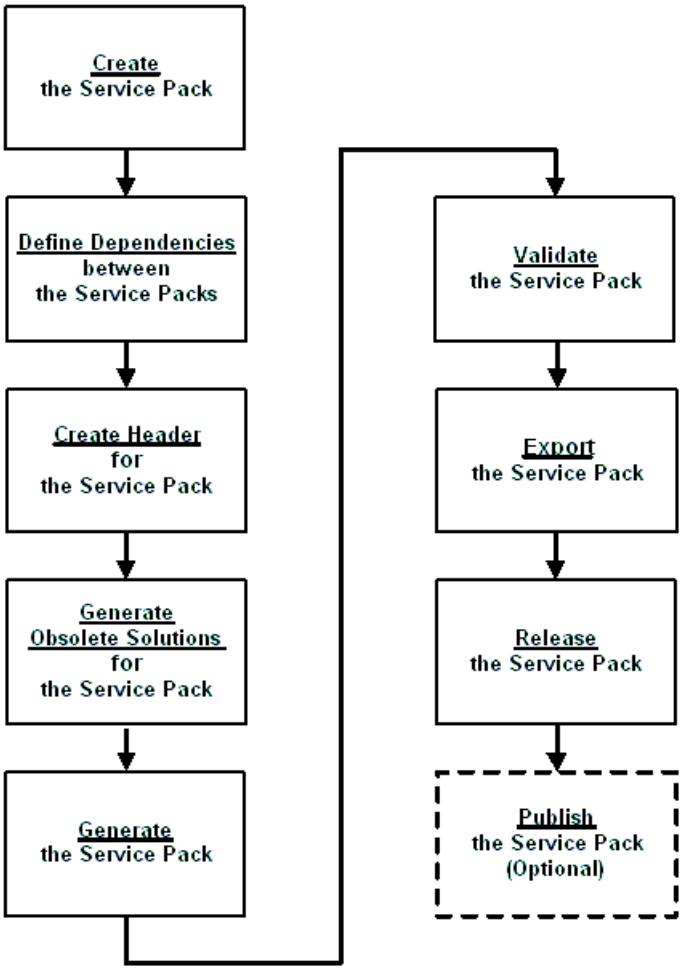
To create Feature Packs

Note

The process to create Feature Packs requires a specific setup of your development environment.

The *Feature Pack development (p. 170)* section describes in detail how the setup must look.

This section describes the procedure steps to create Feature Packs:



You can create or maintain Feature Packs in the Patches (ttpmc1501m000) session.

This session is the central session from which you maintain all aspects of Feature Packs.

You must select the **New Base VRC** check box, otherwise, the patch is not a Feature Pack.

The initial status of a Feature Pack after creation is **In progress**.

If you have created a Feature Pack, you must take the following steps:

- **Define dependencies between Feature Packs**
  Define dependencies between Feature Packs in the Dependencies (ttpmc1140m000) session.
- **Create the Feature Pack header**
  Create the Feature Pack header in the Patch Header (ttpmc1106m000) session.

- **Generate obsolete solutions for the Feature Pack**
  Generate obsolete solutions for the Feature Pack in the Generate Obsolete Solutions (ttpmc1252s000) session.
- **Generate the Feature Pack**
  Generate a Feature Pack in the Generate Patch for New Base VRC (ttpmc1253s000) session.
- **Validate the Feature Pack**
  Validate a Feature Pack in the Validate Patch (ttpmc1401s000) session.
- **Export the Feature Pack**
  Export a Feature Pack in the Export Solution/Patch (ttpmc1200s000) session.
- **Release the Feature Pack**
  To release a Feature Pack, change the **Status** from **Exported** to **Released** in the Patches (ttpmc1501m000) session.

  You can only release a Feature Pack if the **Status** of the Feature Pack is **Exported**.
- **Publish the Feature Pack**
  To publish a Feature Pack, you must select the **Published** check box in the Patches (ttpmc1501m000) session.

  You can only publish a Feature Pack if the **Status** of the Feature Pack is **Release**.

  This step is optional.
- **View the Feature Pack history**
  You can view the Feature Pack history in the Maintenance History (ttpmc1560m000) session.

Feature Pack development

This section describes various aspects applicable to the Feature Pack development and build process.

This section describes the following topics:

- System setup for standard products
- System setup for derived products
- Rules for upgrading derived products
- To build Feature Packs and the Infor Installation Wizard

System setup for standard products

The subsequent setup assumes that the distributor side includes a combined development/maintenance environment. This combined environment supports the maintenance of already released Feature Packs, development of a new Feature Pack, and the physical creation of solutions and Feature Packs.

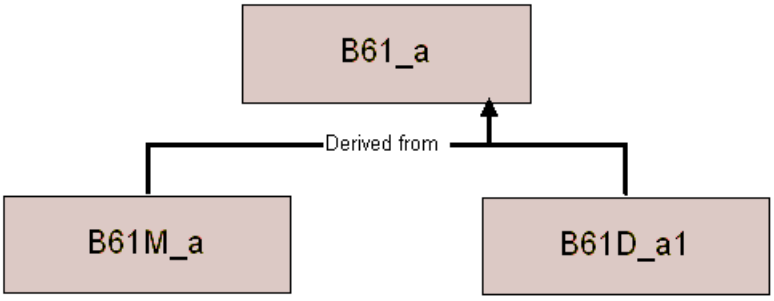
Note

The following examples use the following conventions for VRC naming:

- **Version: B61<X>**
  X = **M** aintenance, **D** evelopment

- **Release: a<Y>**
  Y = Feature Pack number
- **Customer: <zzzz>**
  zzzz = extension/localization code

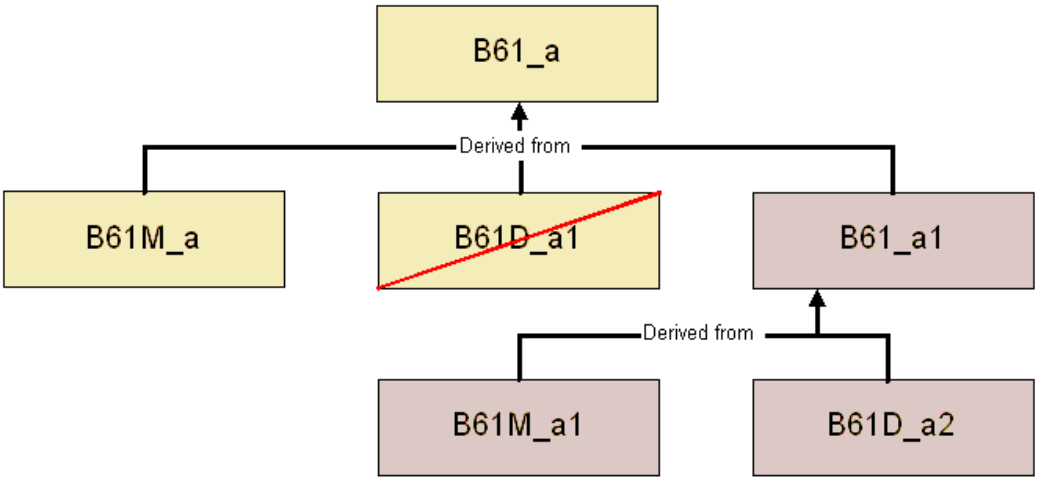
The following figures illustrate the results over time:



As soon as the initial master product, B61_a, which for the sake of simplicity is referred to simply as Feature Pack 0 in the remainder of this document, is released, the maintenance starts for B61_a in the maintenance VRC B61M_a. The development of the new Feature Pack is carried out in B61D_a1. All fixes in B61M_a are ported to B61D_a1. In the PMC registry of B61D_a1, these fixes are registered as obsolete solutions. After B61D_a1 is frozen, a final Feature Pack is created for the Feature Pack and installed in the same environment to enable you to perform maintenance on the Feature Pack.

Note

The SCM revision data is not available in the Feature Pack VRC B61_a1, because the export/import software does not support the transfer of revisions. Because the original development VRC B61D_a1 is still present, you can still view revisions.



Currently, you can maintain B61M_a and B61M_a1, Feature Pack 0 and Feature Pack 1, in parallel. You must port bug fixes that are created for Feature Pack 0 between the freeze moment and the time of installation of Feature Pack 1 first. Feature Pack 2, B61D_a2, is currently in development.

You can repeat this cycle unlimited times for each new Feature Pack.

System setup for derived products

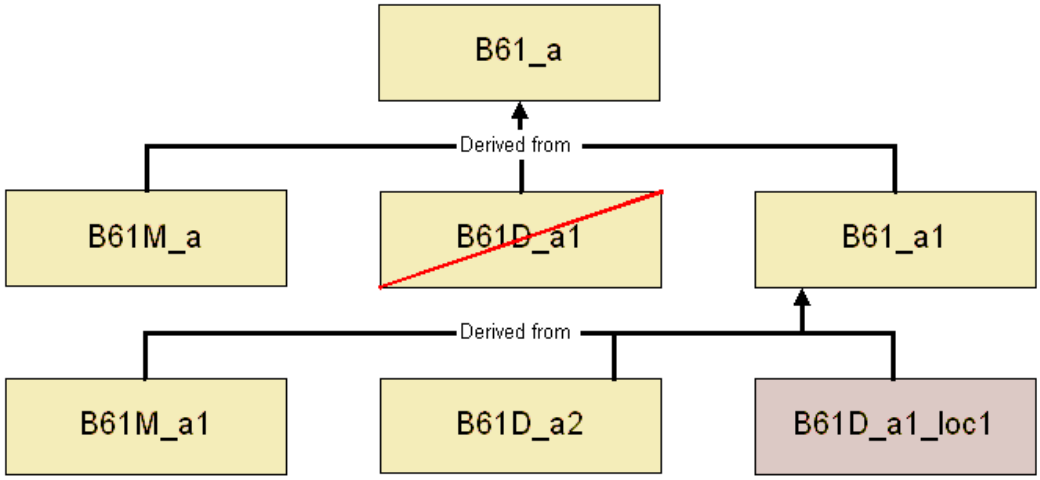
The setup becomes a bit more complex if derived products such as localizations or extensions enter the scope.

Note

Customizations are also a type of derived products. For customizations, the situation differs somwhat. The section *Customizations (p. 175)* describes how to deal with customizations.

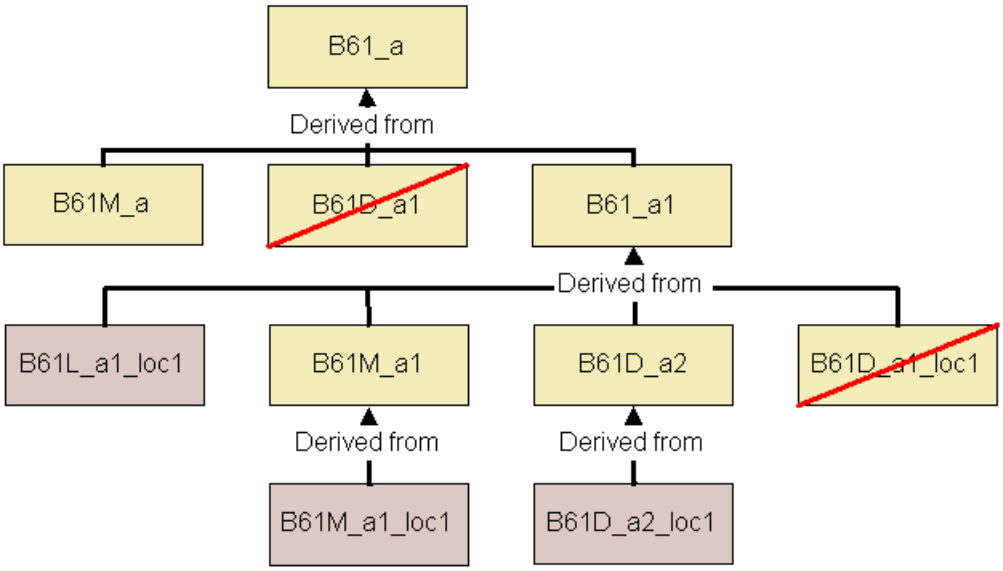
Suppose you must create a localization on top of Feature Pack 1.

A new VRC is created on top of Feature Pack 1, B61D_a1_loc1:



The localization is built on top of the standard Feature Pack and is delivered as a Feature Pack in PMC. Therefore, you must not create master media for derived products. As soon as Feature Pack 0 for the localization is available, the Feature Pack is installed in the maintenance VRC, B61M_a1_loc1.

The localization is also copied to a development VRC B61D_a2_loc1:



The first action that you must perform is to merge all bug-fixes of the standard VRC B61M_a1 into the localization maintenance VRC B61M_a1_loc1 and to create PMC solutions for the localization.

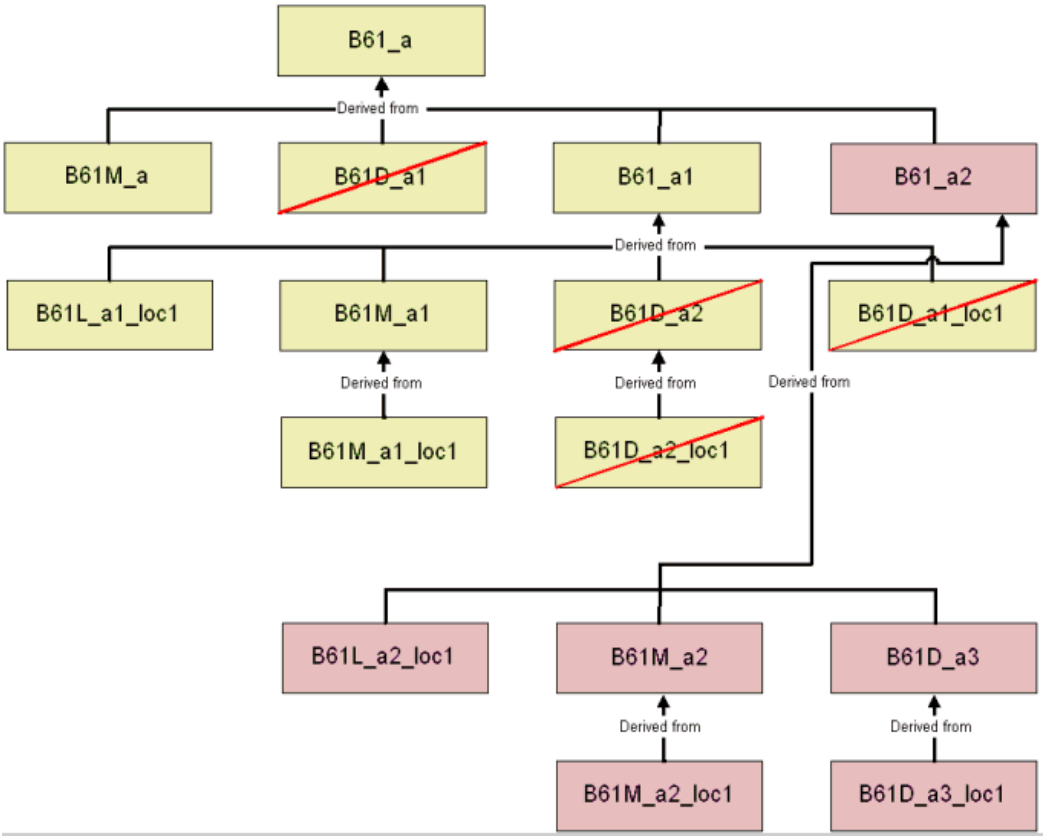
In addition, you must merge the B61D_a2 functional enhancements to B61D_a2_loc1.

From this moment on, the localization follows the same Feature Pack cycle as the standard software.

Repeat the same approach for Feature Pack 2.

The physical Feature Packs for the standard software and the localization are created and installed in a separate VRC tree B61_a2 and B61L_a2_loc1. New maintenance and development VRCs are created.

The maintenance and development VRC for the localization are populated with the components from the installed Feature Pack B61L_a2_loc1.



Rules to upgrade derived products

The following rules apply with respect to the creation of Feature Packs for derived products:

- You must only build Feature Packs for derived products on top of Feature Packs of the parent product. Do not build Feature Packs on top of a Feature Pack plus a number of individual solutions of the parent product. The reason for this is because you can only define PMC dependencies between the Feature Pack of the standard and the derived products. You cannot define PMC dependencies from the Feature Pack of the derived product to individual solutions of the parent product.
- The Feature Pack frequency of derived products can be greater than the Feature Pack frequency of the parent product. Therefore, you can build multiple derived Feature Packs on top of a single Feature Pack of the parent product.
- The Feature Pack frequency of derived products cannot be lower than the Feature Pack frequency of the parent product. If you do not update a derived product in time, you will most likely face various compatibility issues between the parent and the derived product.

Building Feature Packs and the Infor Installation Wizard

PMC Feature Packs are installable units that you can install by means of the Infor Installation Wizard. Metadata that is included in a Feature Pack controls the behavior of the Infor Installation Wizard. The following aspects are relevant if you want to make your Feature Pack suitable for installation with the Infor Installation Wizard:

- Before you build a Feature Pack, be sure to fill in the installable unit key, name, and version are filled at your PMC base VRC. If you do not fill in these values, the Infor Installation Wizard will not recognize the installable unit. For more details, refer to the Base VRC's (ttpmc0110m000) session.
- Before you export a Feature Pack, be sure that the Feature Pack header contains the correct properties. Based on the properties defined in the Feature Pack header, the Infor Installation Wizard can automatically perform particular actions at a recipient system, such as create new package combinations, create new update VRCs, and link new update VRCs to an existing or new package combination. For more details, refer to the Patch Header (ttpmc1106m000) session.

Customizations

This section describes various aspects relevant to the development and maintenance process of customizations.

Customizations are, in most cases, built by organizations other than the producer of the related parent standard product.

The vendor who builds customizations installs the standard parent product in an update VRC.

The customization is built on top of the installed parent standard product.

This situation is specific in the sense that PMC serves both as recipient and distributor in such a development environment.

The following sections describe best practices how to develop and maintain customizations.

- Patches and customizations
- Feature Packs and customizations
- Dependencies for customizations
- Other

Patches and customizations

In some cases, the vendor of the standard parent product applies patches rather than Feature Packs as a method to provide incremental updates to the market. As opposed to Feature Packs, patches are not cumulative. The installation of a patch in an update VRC requires that you also install all preceding update VRCs in the same update VRC. This issue has consequences for the VRC setup at the distributor.

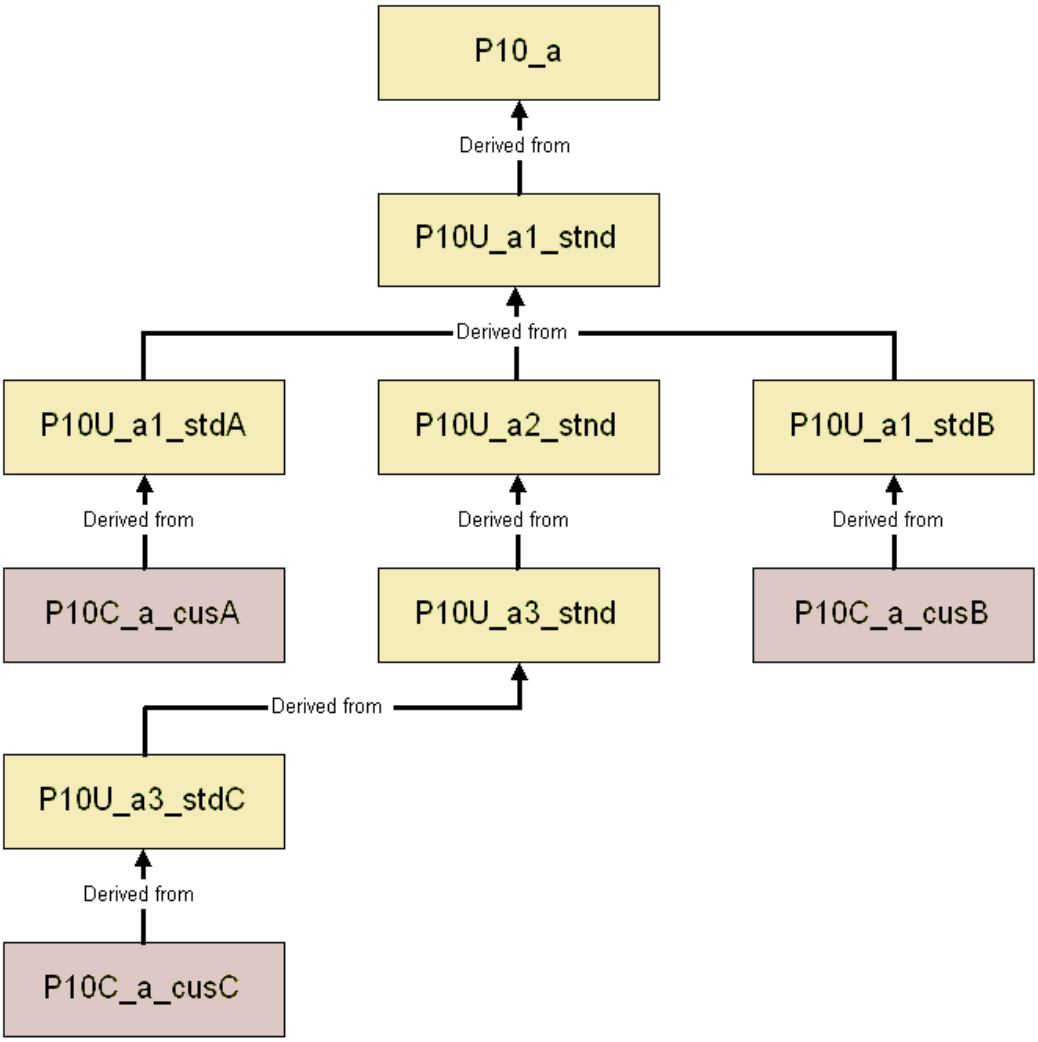
In most cases, you use a single environment to develop and to maintain multiple customizations. You build every customization on a specific patch. In addition, you must periodically upgrade existing

customizations to a newer patch. As a result, you must have multiple Feature Packs of the standard parent product installed in separate VRCs. The following method is the best way to achieve this:

Install patch 1 of the standard parent product in update VRC P10U_a1_stnd. When patch 2 is available, install the patch in a new, separate update VRC P10U_a2_stnd, which is derived from P10U_a1_stnd. Patch 2 will have patch 1 as a prerequisite. You cannot install patch 2 in the new update VRC P10U_a2_stnd because patch 1 is not installed in this new update VRC. Therefore, run the Copy Solution Registry to Derived Update VRC (ttpmc2290m000) session and copy the registry of update VRC P10_a1_stnd into P10_a2_stnd. After you perform this step, you can install patch 2 directly in the P10U_a2_stnd update VRC. You can then repeat this same process for subsequent patches. The P10U_a<n>_stnd VRCs must only contain net patches. No additional solutions must be installed in these VRCs that do not belong to the relevant patch.

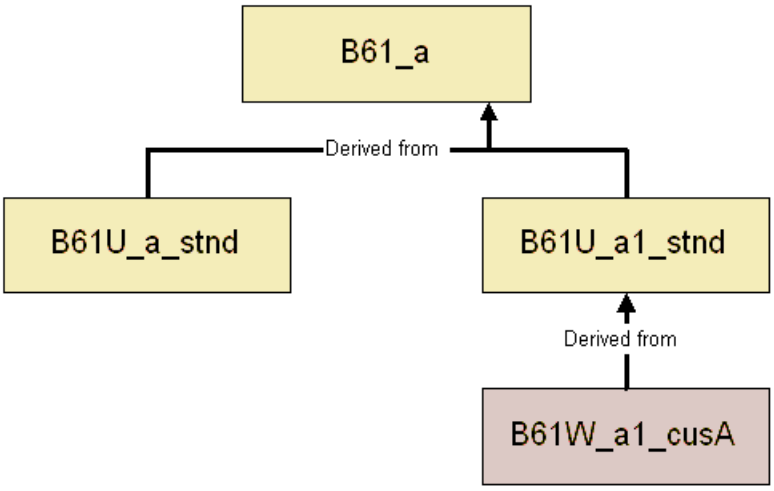
At this point, you have a VRC tree in which the net patches are installed in separate VRCs. In general, customizations are developed based on a net patch. However, in some cases customizations are not derived from a net patch, or a customer might want to install parent standard product solutions that have components which overlap with the customization. As a result, the customer must have the option to derive the customization VRC from any possible maintenance stage of the parent standard product. The P10U_a<n>_std update VRCs are introduced to achieve this. For every customization an intermediate VRC layer is introduced in which standard solutions can be installed that do not belong to the underlying patch. You can again use the Copy Solution Registry to Derived Update VRC (ttpmc2290m000) session to populate the PMC registry for the P10U_a<n>_std update VRCs.

The customization VRCs P10C_a_cusA, P10C_a_cusB, and P10C_a_cusC represent various various customizations that are derived from various patches. Suppose customization P10C_a_cusA is developed based on the net patch 1. During development of the customization, the P10U_a1_stdA VRC will still be empty. After the customization is released and the customer installs the customization, the customer might want to install solutions of the standard parent product that also require an update of the customization. In this case, the customization vendor can install the solutions of the parent standard product in the P10U_a1_stdA VRC, port the solution to the P10C_a_cusA and provide a corresponding solution for the customization VRC. Installation of solutions in the P10U_a1_stdA VRC will only have impact on the P10C_a_cusA customization and will not impact the other customizations in the environment.



Feature Packs and Customizations

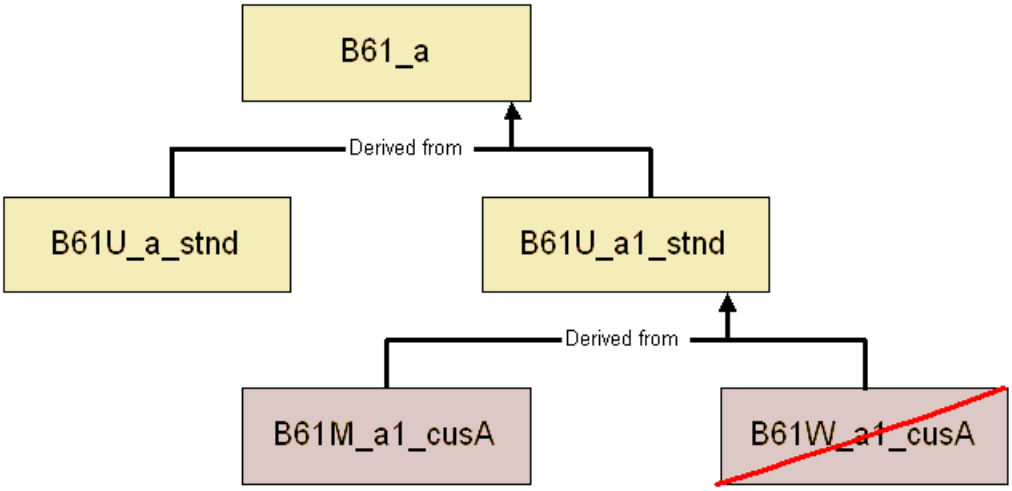
To a certain extent, the system setup is comparable to the setup described for localizations and extensions described in the section *Feature Pack development (p. 170)* . The main difference lies in the fact that Feature Packs for the parent standard product are now installed in an update VRC by means of PMC. Another differentiator is the fact that Feature Packs are cumulative, which means that Feature Packs also contain all preceding Feature Packs. To upgrade a customization to a higher Feature Pack level, having a VRC available that only contains the net changes of a Feature Pack of the standard parent product can be useful. Suppose you must create a customization on top of Feature Pack 1 for a parent standard product. Feature Pack 1 is installed in the B61U_a1_stnd update VRC and a new VRC B61W_a1_cusA is developed on top of the Feature Pack.



The customization is built on top of the standard Feature Pack and will be delivered as a Feature Pack in PMC, so no master media must be created for the customization. As soon as Feature Pack 0 for the customization is available, the Feature Pack is installed in the maintenance VRC B61M_a1_cusA.

Note

In theory, you can also skip the installation of the FP in the B61M_a1_cus1 VRC and start maintenance immediately in the B61D_a1_cus1 VRC. However, best practice is to install and maintain the official build that customers also receive.

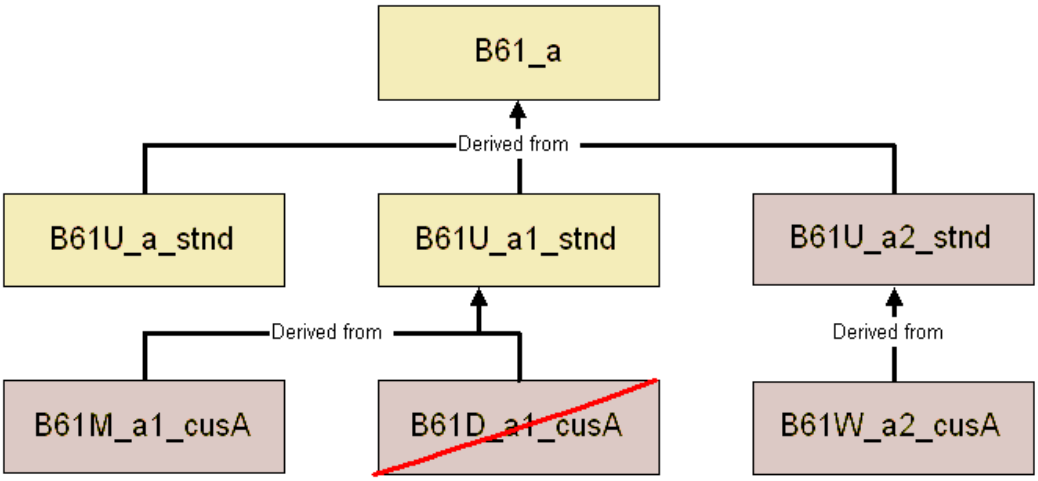


Solutions for Feature Pack 1 of the parent standard product will be installed in the B61U_a1_stnd VRC. When you scan solutions for the parent standard product, you can also print a customization report. The customization report provides an overview of which components must be updated in the B61M_a1_cusA VRC. As soon as you install the standard solution, you can merge the changes to the B61M_a1_cusA VRC and create a corresponding PMC solution for the customization.

The approach for subsequent Feature Packs is as follows: Install Feature Pack 2 for the parent standard product in the B61U_a2_stnd VRC, which is derived from B61_a VRC. Copy the contents of the B61M_a1_cusA VRC to the B61W_a2_cusA VRC. As a result, all maintenance changes you create on top of Feature Pack 0 of the customization until the moment of the copy action immediately become available in the B61W_a2_cusA VRC. After the copy, you must also port all maintenance changes in the B61M_a1_cusA VRC to the B61W_a2_cusA VRC. Feature Pack 2 for the standard product is cumulative, which means that all changes implemented in Feature Pack 1 are also present in Feature Pack 2. Therefore, you must create an overview of all components that were changed in Feature Pack 2 as compared to Feature Pack 1, and these components also must be present in the customization. To achieve this, you can use the Compare Package VRC's (ttadv6450m000) session. In this session, specify values in the following fields as follows:

VRC 1	B61U_a1_stnd
VRC 2	B61U_a2_stnd
Where Software Components present in VRC	B61W_a2_cusA

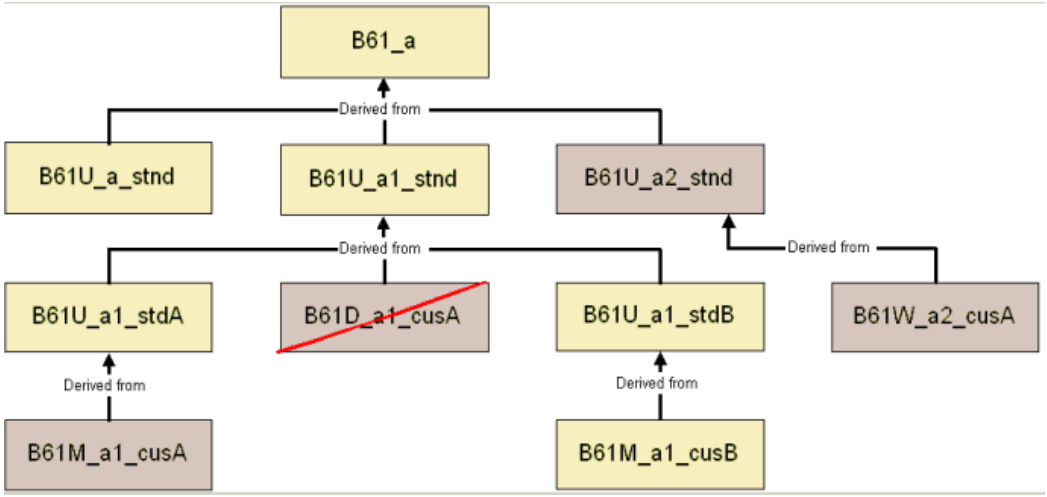
Using these settings enable you to create a net overview of customized components that must be updated to be compatible with the Feature Pack 2 of the parent standard product. You can then merge all changes implemented in Feature Pack 2 of the standard product from the B61U_2_stnd VRC into the B61W_a2_cusA VRC. You can then build Feature Pack 1 for the customization, based on the contents of the B61D_a2_cusA VRC.



You can repeat this procedure for an unlimited number of Feature Packs. In addition, you can develop and maintain multiple customizations in parallel on various levels of the standard parent product.

If multiple customization VRCs are derived from, for example, the B61U_a1_stnd VRC, then for every customization, an additional update VRC layer B61U_a1_std is required between the customization

VRC and the B61U_a2_stnd VRC. You can use the additional update VRC layer to install additional solutions of the parent product, which you must port to the customization. You must use the Copy Solution Registry to Derived Update VRC (ttpmc2290m000) session to populate the PMC registry for the B61U_a1_std update VRCs. Installation of additional solutions in the B61U_a1_stdA update VRC only impacts the B61M_a_cusA VRC and does not influence the B61M_a1_cusB VRC.



Dependencies for customizations

Individual solutions, patches and/or Feature Packs for customizations must have dependencies to corresponding updates of the parent standard product. Not having these types of dependencies can result in runtime compatibility problems if only a solution for the standard parent product is installed, while installation of an update for the customizations was also required.

The best way to establish these dependencies is to define co-requisite dependencies between the updates of the parent product and the corresponding updates for the customized product. The co-requisite dependencies are generated automatically if the customization uses the same solution numbers that were also used in the parent standard product. Of course this method only works if the base VRC of the customization, together with the base VRC of the standard parent product, are part of the same base VRC combination at the distributor side. At the recipient side, the update VRCs for the parent standard product and the customization must also be included in a VRC combination.

The co-requisite relationship will only be in one direction: from the customization to the standard parent product. This one-directional co-requisite makes it impossible to install a solution for the customization without simultaneously installing the corresponding standard solution for the parent product. However, as long as you have not scanned the customized solution, you can still install a solution for the standard parent product without simultaneously installing the corresponding solution of the customization. This can result in run time compatibility problems. The customization report solves this problem. When you check standard solutions in a customized environment, you can optionally print a customization report, which signals customized components and informs the recipient that, together with the standard solution, an update for the customization must also be installed.

Other

The facts described in the sections *Rules to upgrade derived products (p. 174)* and *Building Feature Packs and the Infor Installation Wizard (p. 175)* are also applicable to the development and maintenance of customizations.

PMC distributor session summary

PMC distributor sessions

Title	Description
Parameters (ttpmc0100s000)	Maintain the parameters for the PMC module.
Base VRC's (ttpmc0110m000)	Maintain the relation between base VRCs and export VRCs.
Base VRC Combinations (ttpmc0111m000)	Display the base VRC combinations.
Base VRC Combinations (ttpmc0111m100)	Maintain the base VRC combinations.
Base VRCs by Base VRC Combination	Link the base VRCs to a base VRC combination.
Solutions (ttpmc1100m000)	Maintain solutions.
Collections (ttpmc1101s000)	Maintain collections.
Password by Patch (ttpmc1105s000)	Enter a password for a patch. A patch is meant to be installed as a whole. If you want to break up a patch to install the solutions it contains separately, a password is required.
Patch Header (ttpmc1106m000)	Maintain patch headers.
Packages by Patch Header (ttpmc1107m000)	Link packages to a patch header.
Defects by Solution (ttpmc1110m000)	Specify which defects are solved by a solution.
Component by Solution (ttpmc1120s000)	Link a component to a solution.

Sessions by Component (ttpmc1125m000)	Display the sessions that use a component that is linked to a solution.
Dependencies (ttpmc1140m000)	Define the dependencies between solutions or patches.
Solutions by Collection (ttpmc1151m000)	Add solutions to a collection.
Export Solution/Patch (ttpmc1200s000)	Create the export dump for a solution or a patch.
Export Solution Multi-level (ttpmc1202s000)	Create a multilevel export dump for a released solution.
Export Collection (ttpmc1210s000)	Export a collection, which means that a dump of the collection is made.
Connect Components to PMC Solution (ttpmc1221m000)	Connect software components in the export VRC to a PMC solution.
Generate Dependencies (ttpmc1240s000)	Generate the dependency relations between solutions which must be installed together at a customer system.
Generate Patch (ttpmc1250s000)	Generate a patch.
Generate Collection (ttpmc1251s000)	Add a range of solutions to a collection.
Generate Obsolete Solutions (ttpmc1252s000)	Generate obsolete solutions.
Generate Patch for New Base VRC (ttpmc1253s000)	Generate patch for New Base VRC.
Select "In Progress" Solution (ttpmc1299m000)	Link a component to an **In progress** solution.
Print Solution/Patch (ttpmc1400m000)	Print solutions and patches.
Validate Patch (ttpmc1401s000)	Validate a patch.
Validate Collection (ttpmc1402s000)	Validate a collection.
Print Collection (ttpmc1403m000)	Print the contents of a collection.
Validate Solution (ttpmc1404s000)	Validate a solution.

Print Components by Solution (ttpmc1420m000)	Print components connected to a solution.
Print Dependencies (Multilevel) (ttpmc1440m000)	Print the dependencies.
Print Maintenance History (ttpmc1460m000)	Print the history of changes for a solution or patch.
Patches (ttpmc1501m000)	Maintain patches.
Patches (ttpmc1501m100)	Maintain patches
Collections (ttpmc1503m000)	Maintain collections.
Components by Solution (ttpmc1520m000)	Connect the modified components to a solution.
Components by solution (ttpmc1521m000)	View the components that are linked to a solution.
Dependencies (ttpmc1541m000)	View the dependencies between solutions or patches.
Solutions by Collection (ttpmc1551m000)	View the solutions that are part of a collection.
Maintenance History (ttpmc1560m000)	Display the status changes of a solution or patch for a base VRC.
General Maintenance History (ttpmc1561m000)	Display the status changes of a all solutions and patches.
Specify Another Solution (ttpmc1821s000)	Specify another solution.

Chapter 9
LN Homepages

9

Homepages introduction

Introduction

Infor delivers various predefined homepages with Infor LN, such as the following:

- Sales Administrator homepage.
- Accounts Payable Administrator homepage.
- Accounting Manager homepage.
- Service Coordinator homepage.

Homepages are used as a start point for navigation for particular roles, such as Warehouse Manager or Warehouse Administrator. A homepage automatically preselects all the work a user must do in LN.

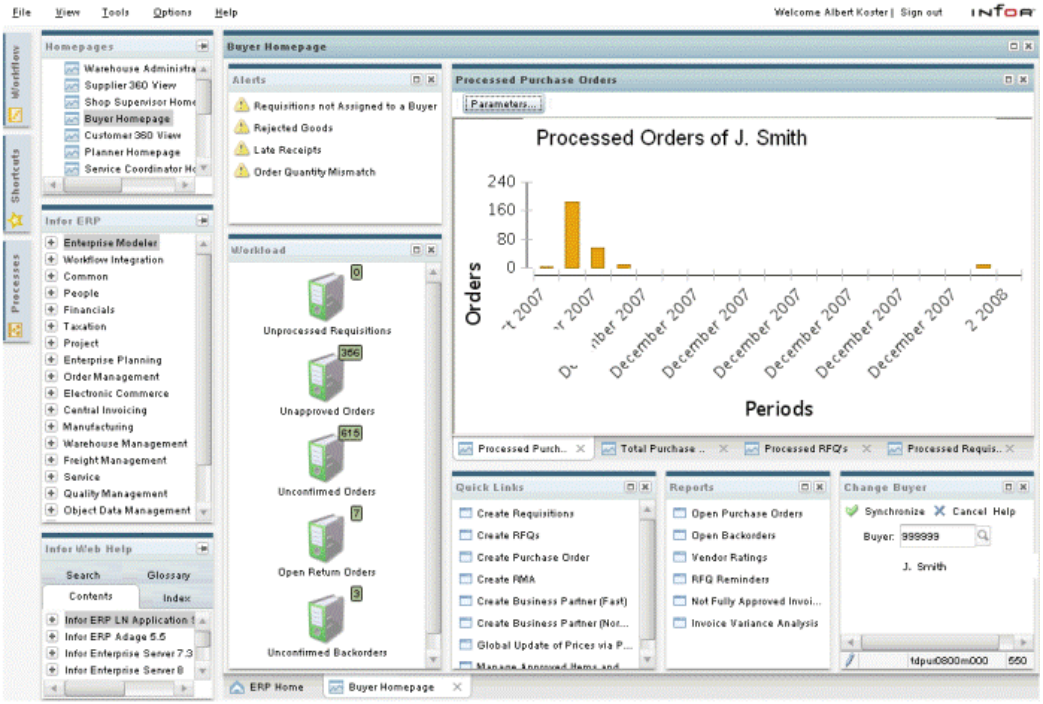
Each homepage contains various dockable panels which can display HTML content. The homepage functions as an up-to-date dashboard on performance and workload.

Important!

Homepages are only available in Web UI.

Homepage structure

A homepage consists of multiple panes such as the following.



A homepage delivered by Infor usually consists of the following panes, but this is not mandatory:

Alert pane	Displays the alerts (if any). Examples of alerts are: ■ Orders that have passed the due date. ■ Late receipts. ■ Orders that are not released yet. ■ A price variance that falls outside the tolerance. Alerts are linked to conditions. An Alert is only displayed if the corresponding condition is met.
Workload pane	Displays the workload assigned to a specific record, for example a specific buyer.

	The Workload pane in the previous figure shows the tasks that you must perform for a specific buyer: ■ There are purchase orders waiting for approval. ■ A number of purchase orders must be confirmed. ■ There are no requisitions that must be processed. ■ And so on. When you double-click a task, the corresponding (overview) session starts. For example, when you double-click the Unapproved Orders task, the Purchase Orders (tdpur4100m000) session starts.
Shortcut pane (Quick Links pane)	Displays useful quick links. A quick link is a shortcut to a details (data entry) session, or to a batch session, for example a global update session.
Report pane	Contains hyperlinks to frequently used reports. The hyperlinks can start LN sessions or Web pages, for example Cognos reports.
Graph pane	Displays reports that show major statistics or key performance indicators. The reports offer many useful features, such as: ■ Many chart types are supported, for example pie charts, line charts, and bar charts. ■ A graph can have a dialog attached, where you can enter selection criteria.

Resource files

The language-dependent content of an LN homepage, such as the homepage's title and the titles of the panes used in the homepage, is stored in resource files.

Each resource file contains data for only one language. The homepages and reports developed by Infor are delivered with English resource files. To run these homepages and reports in other languages, you must translate the corresponding resource files.

For more information, refer to *To translate resource files (p. 199)* .

Personalization by Web UI end users

To change the look and feel of their homepages, end users can do the following:

- Move and dock panels in different locations.
- Resize, minimize, and maximize panels.
- Drag items to another position within the Work Load pane, the Alert pane, and the Shortcut pane.
- Add new shortcuts.
- Switch between large and small icons.
- Personalize the Work Load pane and Alert pane: For example, an end user can define thresholds. A Web UI administrator can also specify refresh rates.

For more information, refer to the Web UI online help.

To create LN homepages

Prerequisites

To create homepages, one of the following must apply:

- You must be a Web UI administrator.
- Your Web UI user profile must have the “homepage creator” role assigned.

To create homepages

To create a new homepage, complete the following steps:

Step 1: Create an empty homepage

1. Start the Web UI, and on the **View** menu, click **Create Home Page**. The Home Page Settings dialog appears.
2. Enter the ID and title for the new homepage, and click **OK**. An empty homepage is displayed in the Work Area.

Step 2: Add a Workload pane

To add a Work Load pane, right-click the homepage's title bar and on the shortcut menu click **Add Work Load Pane**. A dialog appears, where you must enter the following properties for the pane.

Plugin	From the list, select **Infor ERP**.
Use predefined Workload	■ If this check box is selected, you can select one of the predefined workload functions delivered by Infor, from the list. The full name of the selected function is displayed in the **Workload list** field. For example, if you select "Warehouse Manager Tasks" from the list, the "owhinh0800.get.dashboard.workload.items" function is selected and displayed in the **Workload list** field. ■ To select a workload function that is not predefined, such as an own workload function, clear this check box. You must type the name of the function in the **Workload list** field.
Title:	Enter a title for the pane.

Workload list	You can only access this field if the **Use predefined Workload** check box is cleared.
	Enter the name of the workload function; this function must be present on the LN server. For details on the structure of workload functions, refer to *To create functions for LN homepages (p. 192)* .
	You must enter the function name in the following format:
	`o<script or library that contains the function>.<name of the function>`
	For example:
	`otdpurdll8500.get.dashboard.workload.items`

To save the settings and close the dialog, click **OK**.

Step 3: Add an Alert pane

To add an Alert pane, right-click the homepage's title bar and on the shortcut menu click **Add Alert Pane**. A dialog appears, where you must enter the following properties for the pane.

Plugin	From the list, select **Infor ERP**.
Use predefined Alert	■ To select one of the predefined alert functions delivered by Infor, select this check box. The full name of the selected function is displayed in the **Alert list** field. For example, if you select "Planner Alerts" from the list, the "ocpintdllh-page.get.planner.alerts" function is selected and displayed in the **Alert list** field. ■ To select an alert function that is not predefined, such as an own alert function, clear this check box. You must type the name of the function in the **Alert list** field.
Title:	Enter a title for the pane.

Alert list	You can only access this field if the **Use predefined Alert** check box is cleared.
	Enter the name of the alert function. This function must be present on the LN server. For details on the structure of alert functions, refer to *To create functions for LN homepages (p. 192)* .
	You must enter the function name in the following format:
	`o<script or library that contains the function>.<name of the function>`
	For example:
	`otdslsdll0801.get.dashboard.alerts`

To save the settings and close the dialog, click **OK**.

Step 4: Add a Report pane

To add a Report pane, right-click the homepage's title bar and on the shortcut menu click **Add Report Pane**. A dialog appears, where you must enter the following properties for the pane.

Plugin	From the list, select **Infor ERP**.
Use predefined Report	■ To select one of the predefined report functions delivered by Infor, select this check box.

	The full name of the selected function is displayed in the **Report** field. For example, if you select "Processed Sales Orders" from the list, the "otdslsdll0700.processed.orders.chart" function is selected and displayed in the **Report** field. ■ To select a report function that is not predefined, such as an own report function, clear this check box. You must type the name of the function in the **Report** field.
Title:	Enter a title for the pane.

Report	You can only access this field if the **Use predefined Report** check box is cleared.
	Enter the name of the report function; this function must be present on the LN server. For details on the structure of report functions, refer to *To create functions for LN homepages (p. 192)* .
	You must enter the function name in the following format:
	`o<script or library that contains the function>.<name of the function>`
	For example:
	`ocpintdllhpage.utilization.chart`

To save the settings and close the dialog, click **OK**.

Step 5: Add a Shortcut pane

To add a shortcut pane, complete the following steps:

1. Right-click the homepage's title bar and on the shortcut menu click **Add Shortcut Pane**. A dialog appears, where you must enter the ID and title for the pane.
2. To save the settings and close the dialog, click **OK**.
3. Add shortcuts to the new shortcut pane. To do this, complete the following steps:
   - Right-click in the shortcut pane and, on the shortcut menu, click **Add new Shortcut**. A dialog appears, where you can specify a shortcut to a Web Page, an LN Homepage, or an LN session.
   - To save the settings and close the dialog, click **OK**.

Step 6: Position the panes in the homepage

All panes you add are displayed as tabs in the Work Area, so you can only view one pane at a time.

Drag the panes to the homepage's title bar so they are displayed within the homepage. Move the panes to the appropriate locations so you can view all panes in one glance. Resize the panes when necessary.

Step 7: Save the homepage

To save the homepage, right-click the homepage's title bar. On the shortcut menu, click **Save Home Page**. Therefore, a shortcut to the homepage is displayed in the **Home Pages** panel.

To create functions for LN homepages

Introduction

When you create a homepage, you must create functions on the LN server, which contain the logic to display the homepage's workload items, alerts, and so on.

To create a homepage function, complete the following steps:

1. Log on to the LN server.
2. Start the Program Scripts / Libraries (ttadv2530m000) session.
3. Either create a library and start the editor, or edit an existing library.
4. Add the function syntax to the library.
5. Save and compile the library.

You can now link the function to a homepage. For details, refer to *To create LN homepages (p. 188)* .

Homepage function types

For descriptions and examples of the following homepage function types, see the following section:

- A function to display workload items.
- A function to display alerts.
- A count function.
- A function to start a session.
- A report item function.

Workload items function

Interface

```
function extern long <workloadFunction>(long i.node)
```

i.node has the following content:

```
<CompanyInfo>
  <compnr> </compnr>
  <logcompnr> </logcompnr>
  <fincompnr> </fincompnr>
 </CompanyInfo>
```

Always returns an XML node:

```
<WorkloadItems>
 <Item>
  <id> </id>
  <label> </label> Label in personalize dialog
  [<type> </type>] Long (could be extended in the future when there are requirements for it). If not
 specified, not possible to filter
  [<unit> </unit>] Only used for filter
  <countFunction> </countFunction>
  <sessionStartFunction> </sessionStartFunction>
  [<supportCount> </supportCount>] default = false
  [<supportAmount> </supportAmount>] default = false
  [<additionalInfo> </additionalInfo>] xml node which is send back to the backend for the count and
 start function to prevent additional reads.
 </Item>
 . . .
</WorkloadItems>
```

Example

```
function extern long get.dashboard.workload.items(long i.node)
{
 long items
 items = xmlNewNode("WorkloadItems")

 < Add the Items >

 return(items)
}
```

The items node could look like the following:

```
<WorkloadItems>
 <Item>
  <id>wl_1</id>
  <label>Open Orders</label>
  <countFunction>get.open.orders.count</countFunction>
  <sessionStartFunction>start.open.orders</sessionStartFunction>
  <supportCount>true</supportCount>
  <supportAmount>false</supportAmount>
  <additionalInfo>
   <buyer>76300</buyer>
  </additionalInfo>
 </Item>
 <Item>
  <id>wl_2</id>
  <label>Created Orders greater than</label>
  <type>long</type>
  <unit>Eur</unit>
  <countFunction>get.create.orders.count</countFunction>
  <sessionStartFunction>start.create.orders</sessionStartFunction>
  <supportCount>true</supportCount>
  <additionalInfo>
   <buyer>76300</buyer>
```

```
  </additionalInfo>
 </Item>
</WorkloadItems>
```

Alerts function

Interface

```
function extern long <alertFunction>(long i.node)
```

i.node has the following content:

```
<CompanyInfo>
  <compnr> </compnr>
  <logcompnr> </logcompnr>
  <fincompnr> </fincompnr>
 </CompanyInfo>
```

Always returns an XML node:

```
<Alerts>
 <Item>
  <id> </id>
  <label> </label> Label in personalize dialog
  [<type> </type>] Long (could be extended in the future when there are requirements for it). If not
 specified, not possible to filter
  [<unit> </unit>] Only used for filter
  <countFunction> </countFunction>
  <sessionStartFunction> </sessionStartFunction>
  [<supportCount> </supportCount>] default = false
  [<supportAmount> </supportAmount>] default = false
  [<additionalInfo> </additionalInfo>] xml node which is send back to the backend for the count and
 start function to prevent additional reads.
 </Item>
 . . .
</Alerts>
```

Example

```
function extern long get.dashboard.alerts(long i.node)
{
 long items
 items = xmlNewNode("Alerts")

 < Add the Items >

 return(items)
}
```

The items node could look like the following:

```
<Alerts>
 <Item>
  <id>al_1</id>
  <label>Open Orders</label>
  <countFunction>get.open.orders.count</countFunction>
  <sessionStartFunction>start.open.orders</sessionStartFunction>
  <supportCount>true</supportCount>
  <supportAmount>false</supportAmount>
 </Item>
```

```
 <Item>
  <id>al_2</id>
  <label>Late Orders</label>
  <type>long</type>
  <unit>days late</unit>
  <countFunction>get.late.orders.count</countFunction>
  <sessionStartFunction>start.late.orders</sessionStartFunction>
  <supportCount>true</supportCount>
  <imageLarge>td/ipu/bipuoutbox.gif</imageLarge>
  <imageSmall>td/ipu/bipuoutbox_s.gif</imageSmall>
 </Item>
</Alerts>
```

Count function

Interface

```
function extern void <countFunction >(long i.node)
```

i.node has the following content:

```
<Item>
  <id> </id>
  <compnr> </compnr>
  <logcompnr> </logcompnr>
  <fincompnr> </fincompnr>
  [<amount> <amount>]   If a type is defined
  [<additionalInfo> </additionalInfo>]
 </Item>
```

Return always an XML node:

```
<Item>
 <id> </id>
 <description> </description>
 [<count><count>]
 [<amount><amount>]
 [<amountDescription><amountDescription>]
</Item>
```

Example

```
function extern long get.create.orders.count(long i.node)
{
 string id(5)
 long amount
 long infoNode

 id = xmlDataElement$(i.node, "id", 0)
 amount = xmlDataElement$(i.node, "amount", 0)
 infoNode = xmlFindFirst("additionalInfo", i.node)

 select . . .
 . . .
 return(node)
}
```

The node could look like the following:

```
<Item>
  <id>wl_2</id>
  <description>Orders greater than 100.000 Eur</description>
```

```
  <count>5<count>
</Item>
```

Function to start a session

Interface

```
function extern void <sessionStartFunction>(long i.node)
```

i.node has the following content:

```
 <Item>
  <id> </id>
  <compnr> </compnr>
  <logcompnr> </logcompnr>
  <fincompnr> </fincompnr>
  [<amount> <amount>]   If a type is defined
  [<additionalInfo> </additionalInfo>]
 </Item>
```

The function does not return anything, but should start a session.

Example

```
function extern void start.create.orders(long i.node)
{
 string id(5)
 long amount
 id = xmlDataElement$(i.node, "id", 0)
 amount = xmlDataElement$(i.node, "amount", 0)

 . . .
 start.session(., .,  ., .)
}
```

The node could look like the following:

```
 <Item>
  <id>wl_2</id>
  <amount>100000</amount>
 </Item>
```

Report item function

Interface

```
function extern long <reportFunction>(long i.node)
```

i.node has the following content:

```
 <CompanyInfo>
  <compnr> </compnr>
  <logcompnr> </logcompnr>
  <fincompnr> </fincompnr>
 </CompanyInfo>
```

Always returns an XML node:

```
<Report>
 <design> </design> Name of the report design
 [<parameterSession> </parameterSession>] Session to start to change the parameters by the end user

 [<parameter
  [name=’. . .’]  Attribute ‘name’ specifies the name of the parameter.
  > </parameter>]* The value of each parameter should be specified.
</Report>
```

Example

```
function extern long purchased.order.amount.report(long i.node)
{
 long node
 . . .
 return(node)
}
```

The node could look like the following:

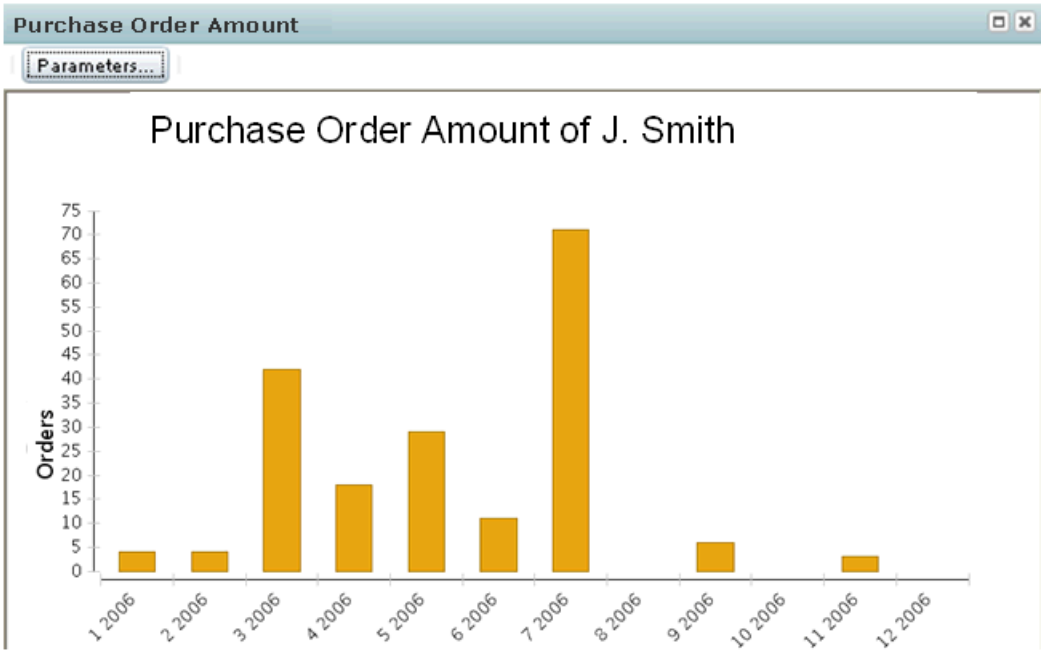
```
<Report>
  <design>Purchased Order Amount</design>
  <parameterSession>gazzzreptest</parameterSession>
  <parameter name=’Period Table’>MONTH</parameter>
  <parameter name=’Start Date’>Jan 01, 2006</parameter>
  <parameter name=’End Date’>Dec 31, 2006</parameter>
  <parameter name=’Buyer’>jsmith</parameter>
  <parameter name=’Default Currency’>EUR</parameter>
 </Report>
```

At runtime this node creates the following link to the report:

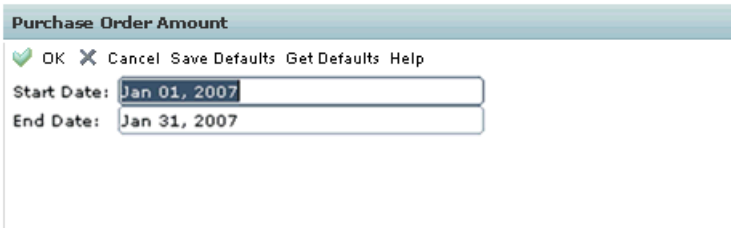
```
... /run?__report=Report\Purchased Order Amount.rptdesign&Period Table=MONTH&Start Date=Jan 01,
```

```
2006&End Date=Dec 31, 2006&Buyer=jsmith&Default Currency=EUR
```

The user can click the link to display a graph. For an example, see the following figure:



When you have defined a parameter session for the report, a **Parameters** button is displayed. When the user presses the button, the corresponding parameter session starts.



With the session you created, the user can change the parameters.

The parameter session is an LN session. Therefore, when developing this session, you have all LN programming possibilities such as browsing and so on. In the choice.end.program section you must call the send.parameters.to.report(…) function. This function requires the parameters specified as pairs, with the name of the parameter followed by the value. See the following example:

```
choice.end.program:
before.choice:
 send.parameters.to.report("Start Date", start.date, "End Date", end.date)
```

At runtime, this code results in a call to the backend. For example:

```
<RESPONSE>
   <ARG NAME="callResult">
       <callResult>
           <SESSION ID="6-2" STATE="CLOSED">
```

```
            <REPORTPARAMETERS>
                <parameter NAME="Start Date">Jan 01, 2007</parameter>
                <parameter NAME="End Date">Jan 31, 2007</parameter>
                <processinfo>report_0</processinfo>
            </REPORTPARAMETERS>
        </SESSION>
    </callResult>
  </ARG>
</RESPONSE>
```

As a result of this call, the homepage does the following:

- Searches for the correct report and replaces the parameters with the new values. The values are only replaced for the parameters sent to the report. For parameters not sent to the report, the values stay the same.
- Refreshes the pane where the graph is displayed.

To translate resource files

Resource files contain language-dependent content of LN homepages and query-based reports. Each resource file belongs to a particular homepage or query-based report. For example:

- A resource file linked to a homepage, contains the homepage's title and the titles of the panes used in the homepage.

- A resource file linked to a query-based report contains the report title, column headings, chart titles, and other static labels used in the report. For more information, refer to *Query-based Reporting (p. 219)* .

Note

- Resource files do not contain language dependent LN content such as label and message descriptions.
- Each resource file contains data for only one language. The homepages and reports developed by Infor are delivered with English resource files. To run these homepages and reports in other languages, you must translate the corresponding resource files.
- When a user opens a homepage or report, Web UI automatically downloads the corresponding resource file from the LN server. Web UI automatically selects the resource file that corresponds to the user's language, as defined in the User Data (ttaad2500m000) session, so that the homepage or report is displayed in the user's own language. If there is no resource file for the user's language, Web UI downloads the original (English) resource file delivered with the homepage.

To translate resource files

To translate a resource file, complete the following steps:

Step 1: Export resource files

1. Start the Export Resource Files (ttadv8913m000) session.
2. Enter the required settings and export the resource files to a directory on the LN server. The session generates an ASCII file for each exported resource file.

Step 2: Rename ASCII files

The names of the generated ASCII files have the following structure:

```
<filename>_<ISO 639 language code>_<ISO 3166 country code>.properties
```

Note

- The country code is optional and used for only a few languages.
- The language code of a report's resource file must be identical to the language code of the resource file of the homepage to which the report is linked. For details, refer to the *Infor Enterprise Server Web UI - Installation and Configuration Guide (U8715 US)*.

Rename the ASCII files so they match the target language code and optionally the target country code. Ensure you use the correct ISO codes, because otherwise the Web UI cannot read the resource file at runtime.

For example, the Export Resource Files (ttadv8913m000) session has generated the following ASCII file: bwhwmdwarehousemanagerhomepage_en.properties.

- To translate the file into German, rename the file to bwhwmdwarehousemanagerhomepage_*de*.properties.
- To translate the file into French, rename the file to bwhwmdwarehousemanagerhomepage_*fr*.properties.
- To translate the file into Dutch, rename the file to bwhwmdwarehousemanagerhomepage_*nl*.properties.

Step 3: Translate content of ASCII files

The ASCII files consist of lines with the following structure: resource ID=resource description. For example, the bwhwmdwarehousemanagerhomepage_en.properties file contains the following lines:

```
warehousemanagerhomepage.whinh2400m000=Warehouse Orders
warehousemanagerhomepage.whinh2100m000=Warehouse Orders
warehousemanagerhomepage.whinh4130m000=Create Shipments
warehousemanagerhomepage.workload=Warehouse Manager Tasks
warehousemanagerhomepage.Warehouse_Orders=Warehouse Orders
warehousemanagerhomepage.whinr1400m000=Inventory Transactions
warehousemanagerhomepage.whina1210m000=Inventory Valuation
```

Translate the resource descriptions in the ASCII files (the text behind the "=" in each line) to the target language.

Step 4: Import resource files

1. Start the Import Resource Files (ttadv8923m000) session.
2. Enter the required settings and import the translated ASCII files into a VRC in the LN environment.

Chapter 10
LN Reporting Overview

10

LN Reporting Overview

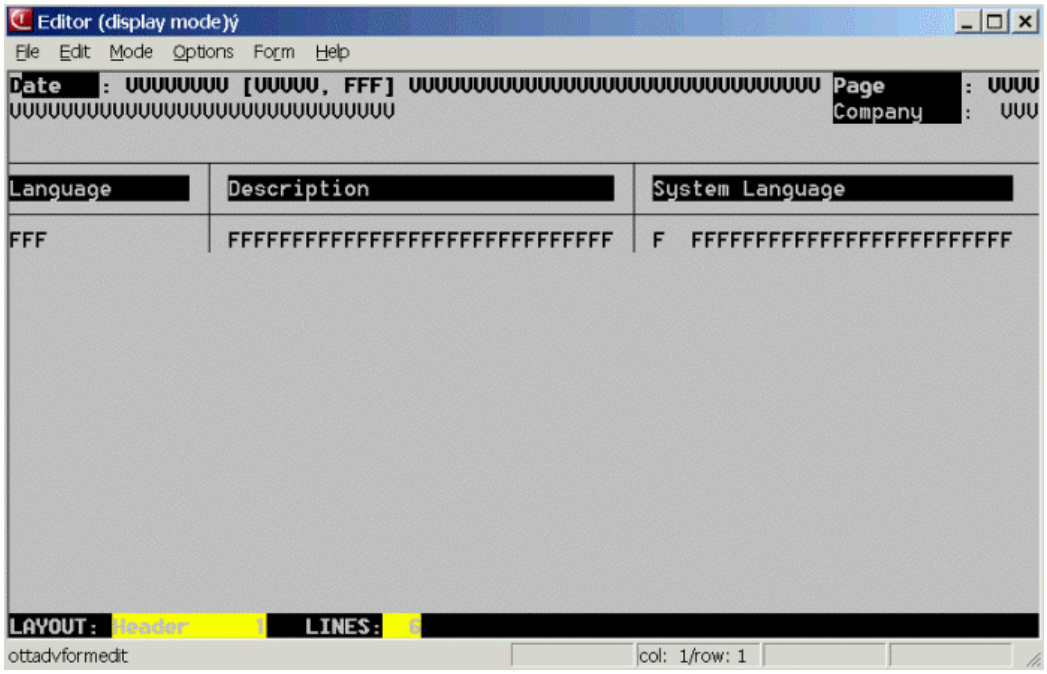
LN supports the following report types:

- 4GL reports
- Report Viewer enabled reports

See the following sections for details.

4GL reports

Most reports in LN are 4GL reports. These reports consist of a layout part and, optionally, a report script. The layout part usually consists of multiple report layouts, such as a header, a before.field and a detail. You can edit these report layouts in the 4GL Report Editor. This editor has an ASCII user interface and does not support advanced features, such as charts and (animated) gif files.



4GL Report in LN 4GL Report Editor

When you finished a 4GL report, you must compile the report. The compiler generates a report object in the runtime data dictionary.

You can print the 4GL reports to different device types, such as a printer, a file, or your own display. The reports generate a fixed font output.

For details on 4GL reports, refer to:

- *Reports (p. 38)*
- "Report Script Features" in the *LN Programmer's Guide*

Note

You can convert 4GL reports to XML report designs. Subsequently, you can enhance the report designs in the Infor LN Reporting Studio. For more information, refer to *Session-based Reporting (p. 213)*

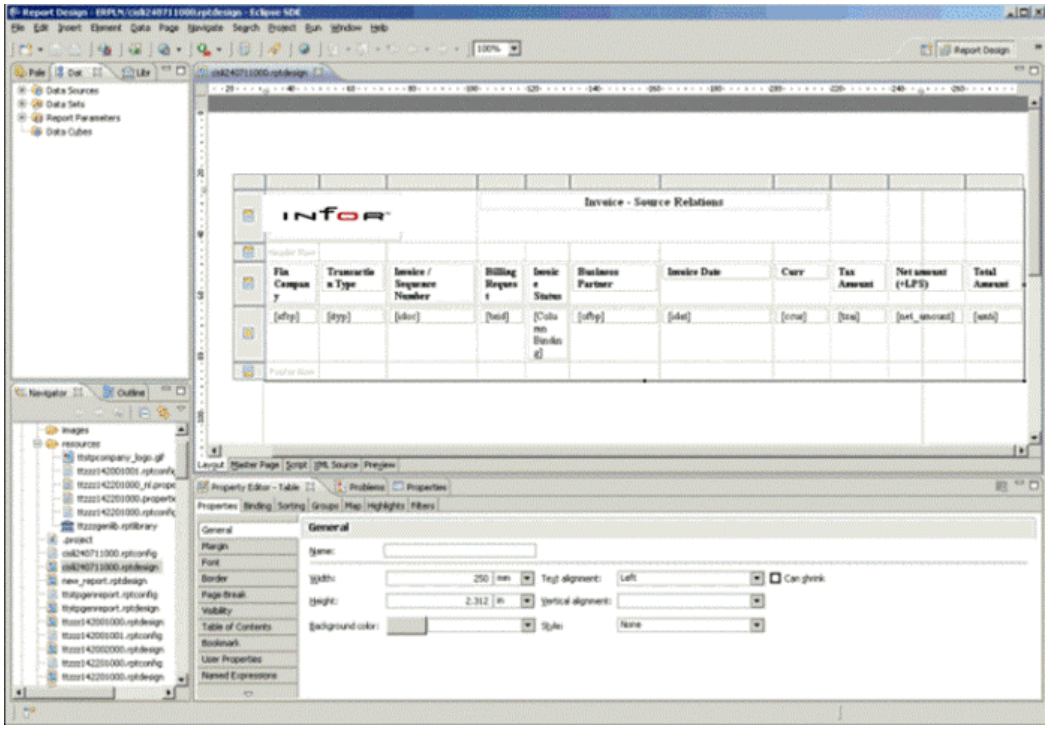
Report Viewer enabled reports

LN also supports Report Viewer enabled reports. You can edit these reports in the Reporting Studio Report Designer. When printed, these reports are displayed in the LN Report Viewer.

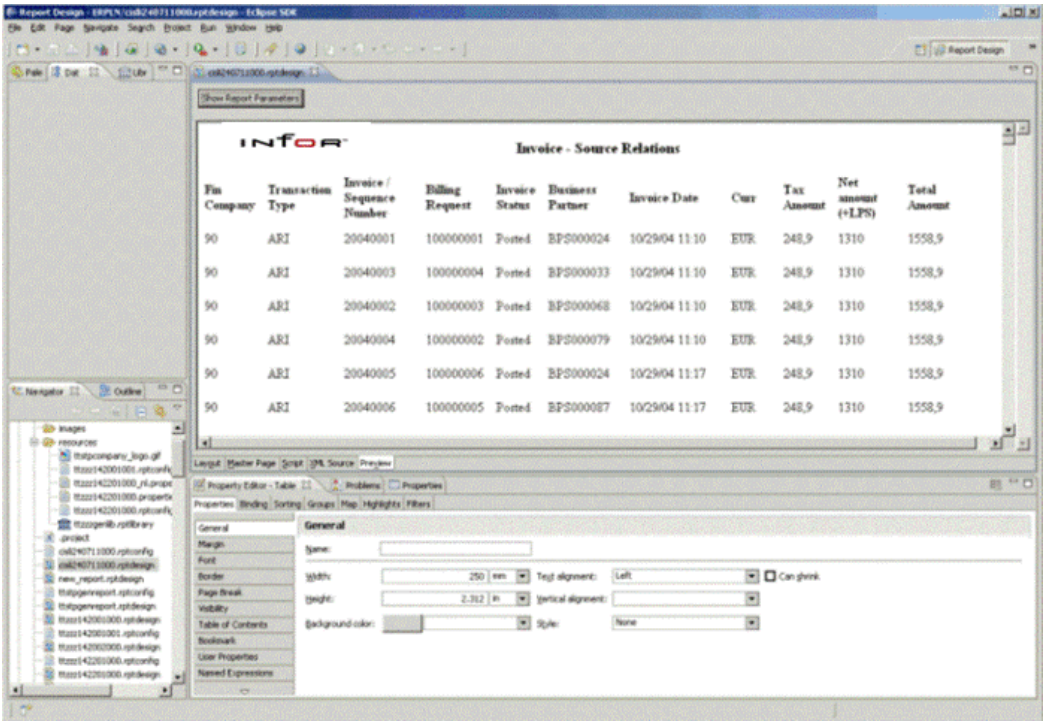
The Reporting Studio Report Designer offers various useful features. For example, you can:

- Add various chart types to your reports, for example pie charts, line charts, and bar charts.
- Attach a dialog to a graph, where users can enter selection criteria.
- Create crosstab reports that show data in two dimensions, for example: sales per quarter.

For details on the Reporting Studio Report Designer, refer to the Reporting Studio online help.



Report Preview in Reporting Studio Report Designer



Report in Reporting Studio Report Designer

In the runtime data dictionary, the report designs are stored in XML files with a ".rptdesign" filename extension.

Note

The LN Report Viewer only works in combination with Web UI. Worktop users cannot print Report Viewer enabled reports.

From the Report Viewer, you can send a report to, among other things, a printer or a PDF file. For details, refer to *To use the Report Viewer (p. 208)* .

Report Viewer enabled reports are created in a report project. For details, refer to *To create a report project (p. 207)* .

Report types

You can create 2 types of Report Viewer enabled reports:

- Session-based reports. For details , refer to *Session-based Reporting (p. 213)* .
- Query-based reports. For details , refer to *Query-based Reporting (p. 219)* .

To create a report project

Before you can create Report Viewer enabled reports, you must first create a report project.

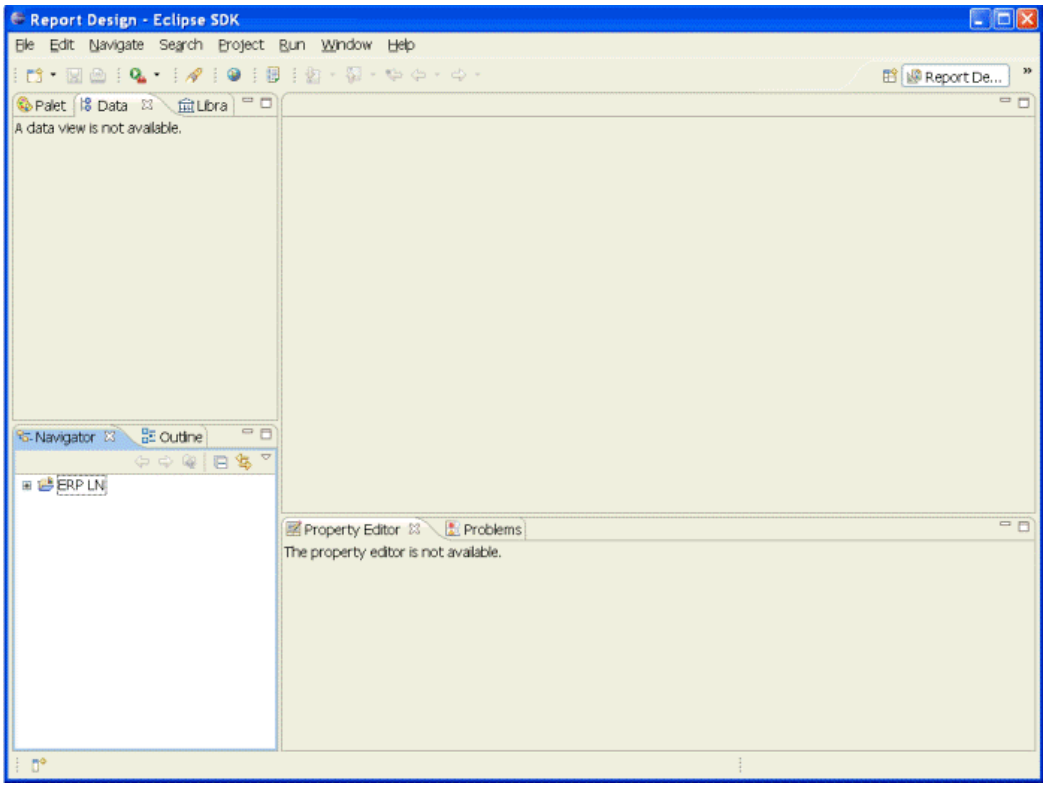
To create a report project, complete the following steps:

1. Start the Reporting Studio.
2. On the **File** menu, click **New**. Then click **Project**. The **New Project** dialog starts.
3. In the **Wizards** pane, expand the **Business Intelligence and Reporting Tools** folder and select **Report Project**.
4. Click **Next**. The **New Report Project** wizard starts.
5. Specify the name of the project, such as LN, and define the location to store the project's files.

   The **Location** property is optional and maps to the directory structure in the file system in which the project and the project's resources are stored.

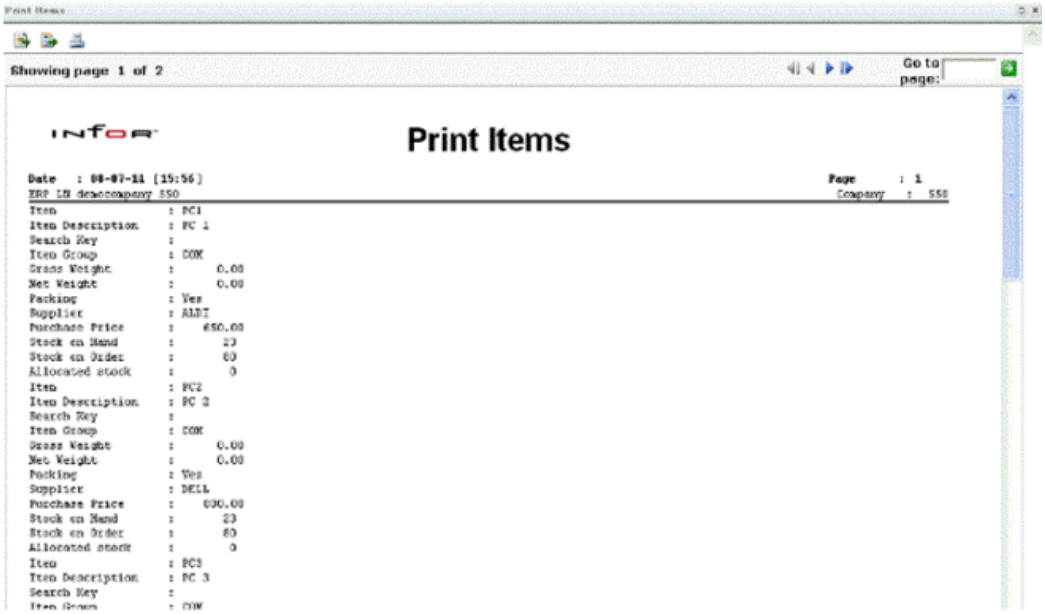
   If you select the **Use default location** check box, the Eclipse workspace is used as the root directory for the project. To create the project and the project's related resources in a different location, clear the **Use default location** check box and specify the new location.
6. Click **Finish**.

The project is displayed in the **Navigator** view. See the following figure for an example.



To use the Report Viewer

The Report Viewer is used to display session-based and query-based reports.



In the Report Viewer you can perform the following actions:

- Navigate through the report data.
- Export report data.
- Print the report.

For details, refer to the descriptions of the toolbar buttons.

Toolbar buttons

The following toolbar buttons are available.

Button	Name	Description
	Export data	Starts the **Export Data** dialog, where you can select report columns and meta data, such as layout names and data types, to be exported. The data is exported in csv (Comma Separated Values) format. You can store the data in

		a file or view it directly, for example in MS Excel.
	Export report	Starts the **Export Report** dialog, where you can select the following: ■ The output format: Excel, PostScript, PDF, Word, or PowerPoint. ■ The pages to export. ■ The report size. You can store the data in a file or view it directly.
	Print report	Starts the **Print Report** dialog, where you can select the following: ■ The output format: HTML or PDF. ■ The report size (only for PDF). ■ The pages to export. The output is displayed in a print preview window. If you selected HTML, you are automatically prompted to select a printer. If you selected PDF, click **Print** to print the output.
	First page	Navigates to the first page.
	Previous page	Navigates to the previous page.
	Next page	Navigates to the next page.
	Last page	Navigates to the last page.
	Go to	Navigates to the page specified in the **Go to page** field.

Chapter 11
4GL Reports

11

4GL Reports Overview

LN 4GL reports are used to output data from the database to a variety of devices (for example, printers, displays, and files).

One or more LN 4GL reports must be linked either to an LN print session or to an LN SQL Query. The print session or SQL Query reads the data from the Database tables and the data will be printed by the Report objects.

4GL reports can automatically generated on two ways:

- By creating a Print session
- By creating a SQL Query

LN 4GL reports contain one or more of the following layouts:

- **Before.report**
- **Header**
- **Before.field**
- **Detail**
- **After.field**
- **Footer**
- **After.report**

The contents and layouts of 4GL reports are defined in the data dictionary. In addition, you can link a report script to a report. In a report script, you can program actions that you want to be performed at particular stages of the report execution. For example, you can create a script to perform calculations on the report data or to read records from related tables that are not automatically available to the report.

You program report scripts in the same way as you program 4GL program scripts, except that report scripts use different event sections and some special functions.

A report script consists of one or more event sections in which you program actions to be performed at particular states of execution of the report to which the report script is linked. The statements programmed in a report script section consist of a combination of 3GL language statements and report script functions.

Report scripts support the following event sections:

- program sections
- report sections
- text field sections

The following program sections can be used:

- declaration
- before.program
- after.program
- after.receive.data
- functions

The following report sections can be used:

- before.report
- after.report
- header
- footer
- before.field
- after.field
- detail

Note

Report sections consist of a main section and a subsection. You use a main section to specify the particular layout for which the programmed actions are to be executed. You must follow a main section with either a **before.layout** or an **after.layout** subsection. These latter specify whether the actions are to be executed before or after the particular layout is printed.

The following Special Report functions can be used in a Report script:

- layout.again()
- need(expr)
- page()
- reset.suppress()
- skip(expr)
- skip.to(expr)
- to.page(expr)

For details on Report scripts, refer to "Report Script Features" in the *LN Programmer's Guide*.

Chapter 12
Session-based Reporting

12

Session-based Reporting

You can convert existing 4GL reports to a report design in XML format, so that the report becomes Report Viewer enabled. You can customize the layout of the converted report in the Reporting Studio.

To print the report, run the corresponding print session on the LN server, and send the output to a Report Viewer device. All business logic defined in the session and the original 4GL report is executed, therefore, a session-based report can be part of the business process. For example, the report can perform transactions in the database, or change the status of a record.

Setup

To set up session-based reporting, you must:

1. Create a report project. For details, refer to *To create a report project (p. 207)* .
2. Create session-based report designs. For details, refer to *To create session-based reports (p. 213)* in this chapter.
3. Create a Report Viewer device. For details, refer to *To create a Report Viewer device for session-based reports (p. 217)* in this chapter.

Execution

For information on how to run session-based reports, refer to *To run session-based reports (p. 218)*

To create session-based reports

To create session-based reports that are Report Viewer enabled, you must convert existing 4GL reports to XML report designs. Subsequently, you can enhance the report designs in the Infor LN Reporting Studio. For details, see the procedure steps.

To create a session-based report, complete the following steps:

Step 1: Convert the report to an XML report design

Start the Enable Reports for RPT Design (ttadv3241m000) session. Specify the required information and click **Convert**.

For details, see the session help.

The generated report designs are stored in the report sources folder and have a ".rptdesign" filename extension.

Note: the following steps are optional. Only perform these steps if you want to customize the report design in Reporting Studio.

Step 2: Transfer the report design to your client PC

Before you can edit a report design in Reporting Studio, you must transfer the report design to your client PC. To do this, complete the following steps:

1. Start the Reports (ttadv3530m000) session.
2. Select the report and, on the appropriate menu, click **Export RPT file...**. The **Export/import file** dialog is displayed.
3. In the **Export to** field, specify the destination directory and file name. The default export destination is the ${BSE_TMP} directory on your client PC.
4. To transfer the report design, click **Continue**. The following message is displayed: `Export file successfully processed`. To close the message window, click **OK**.
5. Close the Reports (ttadv3530m000) session.

Step 3: Start the Reporting Studio

To start the Reporting Studio, run the `eclipse.exe` executable in the `eclipse` subdirectory in the Reporting Studio installation directory.

For example, run `C:\Infor\ERP\ReportingStudio\eclipse\eclipse.exe`.

Step 4: Import the report into a report project

To import the report, complete the following steps:

1. In the **Navigator**, right-click the report project and, on the shortcut menu, click **Import**. Alternatively, on the **File** menu, click **Import**. The **Import** wizard's **Select** dialog starts.

   Note If there is no report project, you must first create one. For details, refer to *To create a report project (p. 207)* .
2. In the **Select an import source** pane, expand the **General** folder and select **File System**. Then click **Next**. The **File system** page is displayed.

3. Specify the location where the report design you want to import is stored. To do this, click the **Browse...** button next to the **From directory** field. Note Select the directory specified during the check out/export of the report, such as the ${BSE_TMP} directory on your client PC.

   In the right-hand pane, the wizard displays all files from the selected directory. Report designs have a ".rptdesign" filename extension. Select the report design you want to import.

4. In the **Into folder** field, specify the report project into which you want to import the report design. To select the appropriate project, click **Browse...**.
5. Select **Create selected folders only**.
6. Click **Finish**.

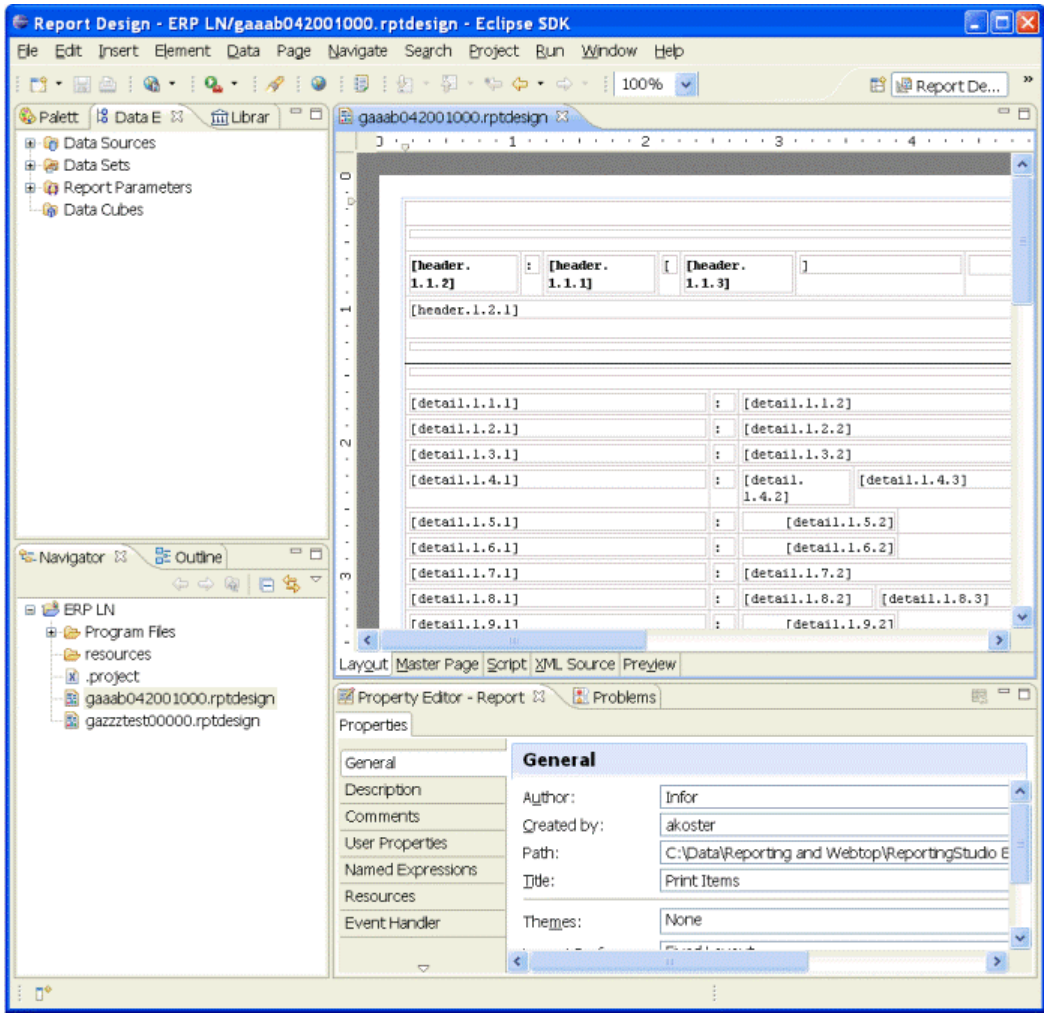
   The report design is now displayed in the **Navigator**.

Step 5: Open the report design

To open the report design, go to the **Navigator** and double-click the report design. Alternatively, select the report design and, on the shortcut menu, click **Open**.

The report layout is displayed in the editor area and report data is displayed in the **Property Editor**.

This diagram shows an example:



Step 6: Customize the report design

Beautify the layout of the report. For example, you can add a graph or company logo.

You can add charts and images through the **Insert** menu, the report editor's shortcut menu, and the **Palette**.

When you select a component in the report layout, the component's properties are displayed in the **Property Editor**. Here you can edit, for example, the font type, size and color.

Important!

- Only use Reporting Studio to beautify the layout of the report. For example, you can add a graph or a company logo.

- To change the structure of the report, for example to add layouts, input fields, or labels, use the classic report editor. For details, refer to *To edit reports (p. 99)* .

Step 7: Save and close the report

Click **Save**, and close the report layout in the editor area.

Step 8: Upload the report design into the LN data dictionary

To upload the report design into the data dictionary on the LN server, complete the following steps:

1. Start the Reports (ttadv3530m000) session.
2. Select the report concerned and, on the appropriate menu, click **Import RPT file...**. The **Export/import file** dialog is displayed.
3. In the **Import from** field, specify the directory and file name of the exported report design on your client PC.
4. To import the report design, click **Continue**. This message is displayed:
   `Import successfully processed.`
5. To remove the message window, click **OK**.
6. Close the Reports (ttadv3530m000) session.

Important!

If you install a solution for a customized report, all changes made in Reporting Studio will be lost.

To create a Report Viewer device for session-based reports

To print session-based reports, you need a device that sends the report output to the LN Report Viewer.

To create a Report Viewer device, complete the following steps:

1. Start the Device Data (ttaad3500m000) overview session.
2. To add a new device, click **New**. The Device Data (ttaad3100s000) details session starts.
3. Enter the properties for the new device. Points of attention are as follows:
   - **Device**: RPT
   - **Description**: Report Viewer
   - **Device Type**: Rewrite File
   - **Locale**: <empty>
   - **Intermediate File in XML Format**: Yes
   - **Driver**: <empty>
   - **Shell Command**: <empty>

- **4GL Program**: <empty>
- **Argument**: XML -RPT
- **Path**: ${HOME}/rpt

4. Save the new device and close the sessions.

To run session-based reports

This section describes how to print a session-based report to the LN Report Viewer.

To print a session-based report to the LN Report Viewer, complete the following steps:

1. Start Web UI and log onto your LN server.
2. Start the print session to which the report is linked.
3. Enter the required selection ranges and print options, and click **Print**. The Select Device (ttstpsplopen) session starts.
4. Select a Report Viewer device and click **Continue**.

Note

The LN Report Viewer only works with Web UI. Worktop users cannot print Report Viewer enabled reports.

From the Report Viewer, you can send a report to, among other things, a printer or a PDF file. For details, refer to *To use the Report Viewer (p. 208)* .

Chapter 13
Query-based Reporting

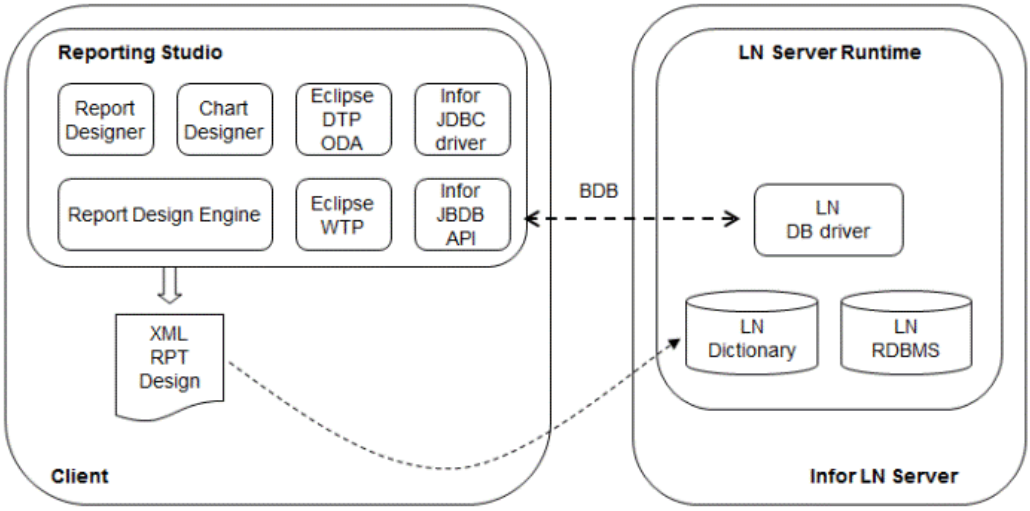
13

Query-based Reporting

In Reporting Studio you can develop query-based reports that read data from the LN database. A query-based report only reads data; it does not perform any transactions or updates, and therefore cannot be part of the business process.

Architecture

The following figure shows an architectural overview of the Reporting Studio and the Enterprise Server runtime environment.



Query-based reporting architecture

Reporting Studio uses the Infor JDBC driver to connect to the LN query-based data source. The Infor JDBC connector must be available on the development client machine. The JDBC connector uses the JBDB API to connect to the LN database driver.

Note

- In LN, the report designs are stored in the Reports (ttadv330) table. The runtime files are stored in the report sources folder and have a ".rptdesign" filename extension.
- When you run a report, Web UI downloads the report design to your machine.

Resource files

The report designs are language independent.

When you want to insert label and text elements in a report, you specify resource keys rather than static text. The report design is linked to one or more resource files that contain the resource keys and their values.

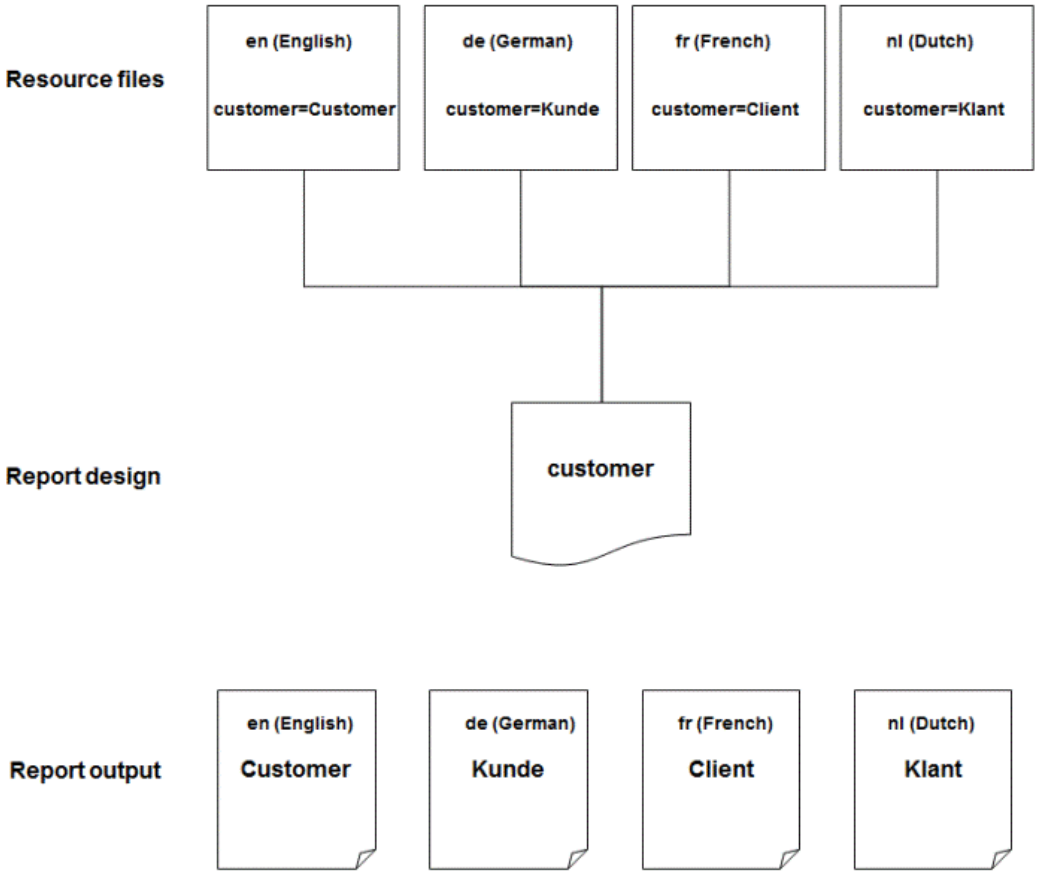
You can create resource files per ISO 639-1 language code and ISO 3166 country code.

In this way you can localize report titles, column headings, chart titles, and other static labels.

When you print a report, Web UI passes the locale to the Report Viewer. The locale of the machine is not used. Therefore, when you create resource files, you must use the same language – country codes as Web UI does.

When you print a report, the Report Viewer uses the locale passed by Web UI, the resource keys, and resource files to find the appropriate text value to display.

See the following figure for an example.



Sample resource files

The figure shows the following elements:

- Resource files for English, German, French, and Dutch. Each file contains the resource key "customer", and the corresponding localized text value.
- Report design. The design is linked to the resource files, and contains the resource key "customer".
- Report output when the report is run in the English, German, French, and Dutch locales.

Note

- In LN, the resource files are stored in the Additional Files table (ttadv270). The runtime files are stored in the additional files folder and have a ".properties" filename extension.
- When you run a report, Web UI downloads the locale-specific resource file to your machine.

Report libraries

Reporting Studio supports the use of shared report elements through libraries: You can create report libraries that contain code you want to share in multiple reports, and link them to your query-based report designs.

Note

- In LN, the report libraries are stored in the Reports (ttadv330) table. The runtime files are stored in the report sources folder and have a ".rptlibrary" filename extension.
- When you run a report, Web UI downloads the report libraries, which are linked to the report design, to your machine.

Setup

To set up query-based reporting, you must:

1. Create a report project. For details, refer to *To create a report project (p. 207)* .
2. Create query-based report designs. For details, refer to *To create query-based reports (p. 222)* . When you design a report, you must also create the required resource files and libraries, and link them to the report design. For details, refer to *To create resource files (p. 228)* and *To create report libraries (p. 230)* .
3. Add the report to a Web UI homepage, or create a 4GL session and link the report to this session. For details, refer to *To create a print session for query-based reports (p. 232)* .

Implementation

You can execute a query-based report from a Web UI homepage, and through a 4GL print session on the LN server.

For details, refer to *To run query-based reports (p. 234)* .

To create query-based reports

To create a query-based report, you must first define a new, empty, report on the LN server. Subsequently, you must define the layout of the report in the Infor LN Reporting Studio.

To create a query-based report, complete the following steps:

Step 1: Create a report definition

To create an empty report definition, complete the following steps:

1. Start the Reports (ttadv3530m000) session. Ensure you select the correct development VRC as your current package VRC.

2. Click **New**. The following question is displayed:
   `Start report generator?`
3. Click **No**. The Reports (ttadv3130s000) details session starts.
4. Specify the module and report code and the description for the new report. In the **Report Type** field, select "Query Based Report".
5. Save the new report and close the details session.

The generated report design is stored in the report sources folder and has a ".rptdesign" filename extension.

Step 2: Transfer the report design to your client PC

Before you can edit a report design in Reporting Studio, you must transfer the report design to your client PC. To do this, complete the following steps:

1. In the Reports (ttadv3530m000) session, complete one of the following steps:
   - If SCM is active, select the report and click **Check-Out**. The report is checked out and the **Export/import file** dialog is displayed.
   - If SCM is not active, for example if LN runs on a Windows platform, select the report. Then, on the appropriate menu, click **Export RPT file...**. The **Export/import file** dialog is displayed.
2. In the **Export to** field, specify the destination directory and file name. The default export destination is the ${BSE_TMP} directory on your client PC.
3. To transfer the report design, click **Continue**. The following message is displayed: `Export file successfully processed`. To close the message window, click **OK**.
4. Close the Reports (ttadv3530m000) session.

Step 3: Start the Reporting Studio

To start the Reporting Studio, run the `eclipse.exe` executable in the `eclipse` subdirectory in the Reporting Studio installation directory.

For example, run `C:\Infor\ERP\ReportingStudio\eclipse\eclipse.exe`.

Step 4: Import the report into a report project

To import the report, complete the following steps:

1. In the **Navigator**, right-click the report project and, on the shortcut menu, click **Import**. Alternatively, on the **File** menu, click **Import**. The **Import** wizard's **Select** dialog starts.

   Note If there is no report project, you must first create one. For details, refer to *To create a report project (p. 207)* .
2. In the **Select an import source** pane, expand the **General** folder and select **File System**. Then click **Next**. The **File system** page is displayed.

3. Specify the location where the report design you want to import is stored. To do this, click the **Browse...** button next to the **From directory** field. Note Select the directory specified during the check out/export of the report, such as the ${BSE_TMP} directory on your client PC.

   In the right-hand pane, the wizard displays all files from the selected directory. Report designs have a ".rptdesign" filename extension. Select the report design you want to import.

4. In the **Into folder** field, specify the report project into which you want to import the report design. To select the appropriate project, click **Browse...**.

5. Select **Create selected folders only**.

6. Click **Finish**.

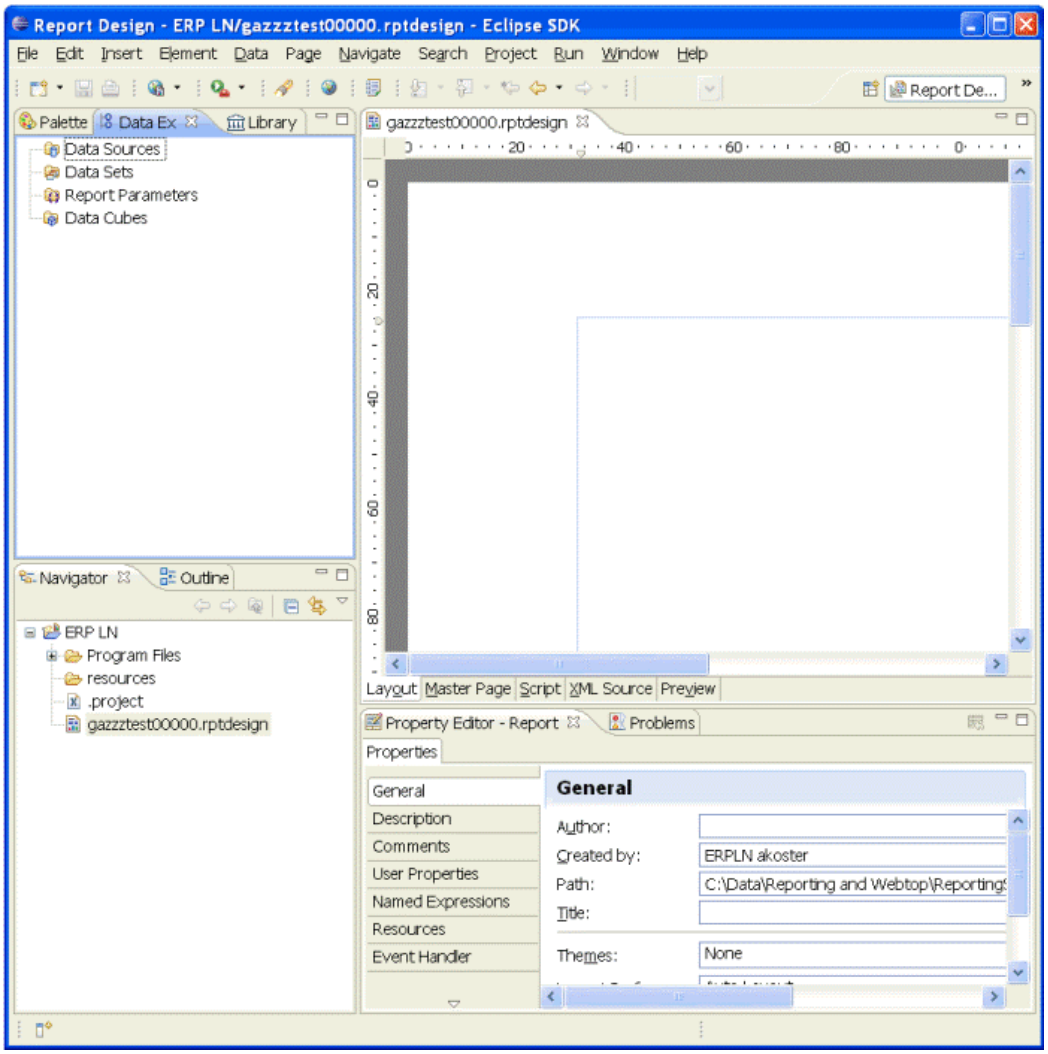
   The report design is now displayed in the **Navigator**.

Step 5: Open the report design

To open the report design, go to the **Navigator** and double-click the report design. Alternatively, select the report design and, on the shortcut menu, click **Open**.

An empty report layout is displayed in the editor area and report data is displayed in the **Property Editor**.

See the following figure for an example.



Step 6: Define a data source for the report

To define a data source, complete the following steps:

1. Go to the **Data Explorer**.
2. Right-click **Data Sources** and, on the shortcut menu, click **New Data Source**. The **New Data Source** dialog is displayed.
3. Select **Create from a data source type in the following list**. Then select **Infor BDB Data Source** and specify a data source name, such as "ERP". Then click **Next**. The **Infor BDB DataSource** dialog is displayed.
4. Click **Test Connection**. If the "DB Connection Client" has been installed properly, the following message is displayed :

```
Connection successful.
```

5. To remove the message, click **OK**. In the **Infor BDB DataSource** dialog, click **Finish**. The Data Source to access the Infor LN database has now been added to the report.

Step 7: Define a Data Set for the report

A data set is a query that specifies the data to retrieve from the data source.

To add a data set, complete the following steps:

1. Go to the **Data Explorer**.
2. Right-click **Data Sets** and, on the shortcut menu, click **New Data Set**. The **New Data Set** dialog is displayed.
3. Specify an appropriate name for the data set, such as “Employees”, and select the new Data Source. In the **Data Set Type** field, select **Infor BDB Data Set**. Then click **Next**. The **BDB DataSet Wizard** starts.
4. In the left pane, use the appropriate package and module to navigate to the table for which you want to define a query. To create a query, drag the table and one or more of its columns from the left pane to the right pane. You can also manually edit the query in the right pane.

   When the query is finished, click **Finish**. The **Edit Data Set** window is displayed.
5. Optionally, make additional changes to the data set. For details, in the window’s lower left corner, click **?**.

   In the left pane, click **Preview Results** to display a preview of the data that will be retrieved by the query.

   When finished, click **OK**. The new data set is now ready to be used in the report.

Step 8: Define the Report Layout

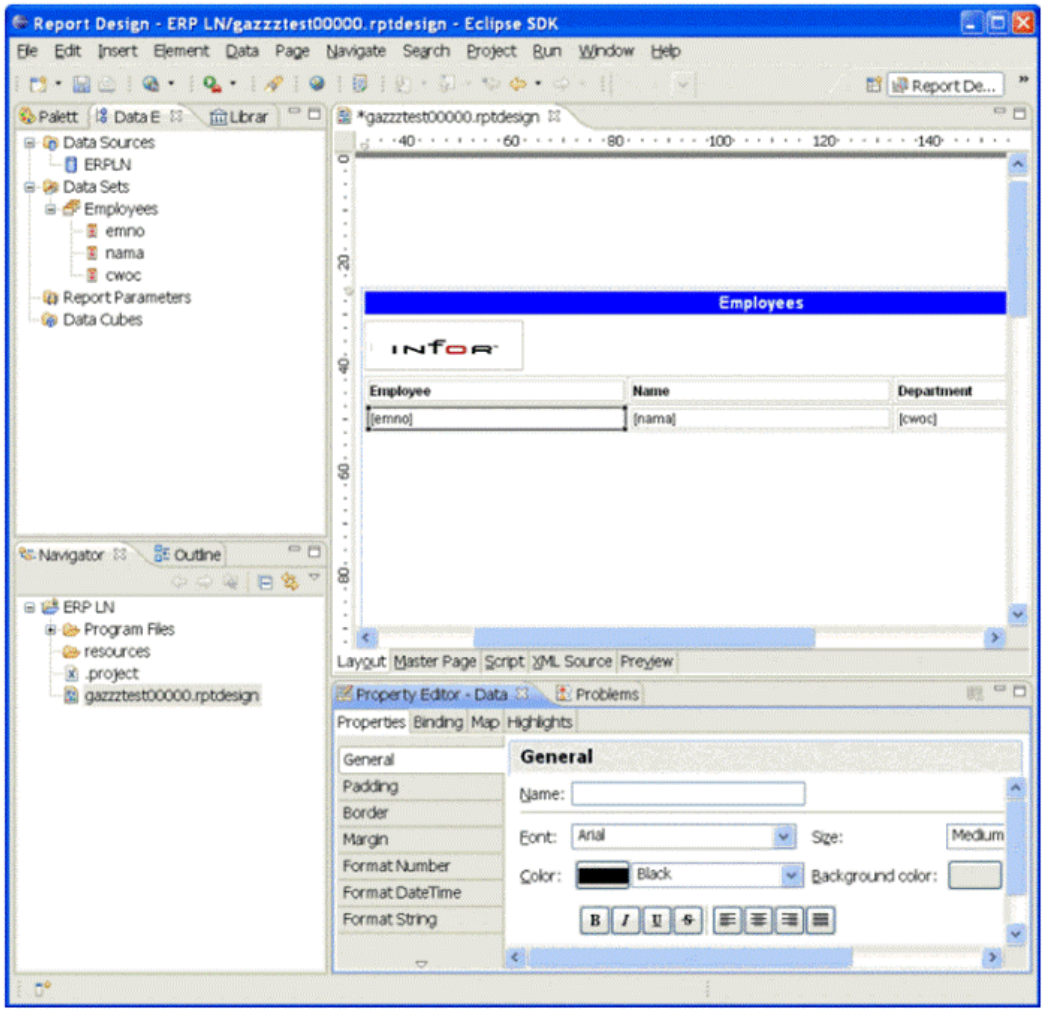
In the report layout, you can add table columns and various other components, such as text, labels, grids, images, charts, and crosstabs.

- To add a table column, in the **Data Explorer**, expand the data set and drag the column to the report layout.
- To add another component type, complete one of the following steps:
  - On the **Insert** menu, click the desired component type.
  - Right-click in the Editor area and, on the shortcut menu, click **Insert**. Then click the desired component type.
  - Go to the **Palette** and double-click the desired component type.

  Depending on the component type, a dialog may appear, where you must specify the properties for the new component. For example, when you add a grid, you must specify the number of columns and the number of rows.

- **Note**
  When you select a component in the report layout, the component's properties are displayed in the **Property Editor**. Here you can edit, for example, the font type, size and color.
- To display a preview of the report, click the Editor's **Preview** tab.
- Optionally, you can create resource files and report libraries, and link them to the report design. For details, refer to *To create resource files (p. 228)* and *To create report libraries (p. 230)* .

For details on report editing, refer to the Reporting Studio documentation.



Sample report layout with grid, labels, table columns, and image

Step 9: Save and close the report

Click **Save**, and close the report layout in the editor area.

Step 10: Upload the report design into the LN data dictionary

To upload the report design into the data dictionary on the LN server, complete the following steps:

1. Start the Reports (ttadv3530m000) session.
2. Complete one of the following steps:
   - If SCM is active, select the report concerned and click **Check-In**. The **Export/import file** dialog is displayed.
   - If SCM is not active, select the report concerned and, on the appropriate menu, click **Import RPT file...**. The **Export/import file** dialog is displayed.
3. In the **Import from** field, specify the directory and file name of the exported/checked out report design on your client PC.
4. To import the report design, click **Continue**. The following message is displayed:
   `Import successfully processed`.
5. To remove the message window, click **OK**. If SCM is active, the following question is displayed:
   `Do you want to check in this Report?`
6. Click **Yes**. A text editor starts. Then, to check-in the report, specify a revision text and close the text editor.
7. Close the Reports (ttadv3530m000) session.

To create resource files

To create a resource file, you must first create an initial, empty, resource file on the LN server. Subsequently, you must complete the resource file in the Infor LN Reporting Studio.

To create a resource file:

Step 1: Create an initial resource file

To create an initial resource file, complete the following steps:

1. Start the Additional Files (ttadv2570m000) session. Ensure you select the correct development VRC as your current package VRC.
2. Click **New**. The Additional File (ttadv2170s000) details session starts.
3. Enter the properties for the resource file:
   - Enter the module code.
   - In the **Additional File** field, enter the name of the new resource file. The name must have the following structure: `<filename>_<ISO 639 language code>_<ISO 3166 country code>.properties`. For example: `MyResource_en_US.properties`. Note: Use the same language – country codes as Web UI does.
   - In the **Additional File Description** field, enter a description.
   - Select the **Editable** check box.

4. Click **Save**. A message indicating the file does not yet exist is displayed. You are prompted to create the file.
5. To create the file, click **Yes**. A text editor starts.
6. Close the text editor and do not enter any text.
7. Close the details session and overview session.

The initial resource file is stored in the additional files folder and has a ".properties" filename extension.

Step 2: Transfer the resource file to your client PC

Before you can edit a resource file in Reporting Studio, you must transfer the file to your client PC.

To transfer a resource file to your client PC, complete the following steps:

1. Start the Additional Files (ttadv2570m000) session.
2. Select the resource file and, on the appropriate menu, click **Export RPT file...**. The **Export/import RPT file** dialog appears.
3. In the **Export to** field, enter the destination directory and file name. The default export destination is the ${BSE_TMP} directory on your client PC.
4. Click **Continue** to transfer the resource file.
5. Import the resource file into the Eclipse workspace.

Step 3: Edit the resource file

1. Start Reporting Studio and open the resource file.
2. In the file, enter the resource keys and their values for the language and country concerned.

For details, refer to the Reporting Studio documentation.

Step 4: Upload the resource file into the LN data dictionary

When you have completed a resource file, you must upload the file into the data dictionary on the LN server.

To upload a resource file, complete the following steps:

1. Start the Additional Files (ttadv2570m000) session.
2. Select the resource file concerned and, on the appropriate menu, click **Import RPT file...**. The **Export/import RPT file** dialog appears.
3. In the **Import from** field, enter the directory and file name of the exported/checked-out resource file on your client PC.
4. To import the resource file, click **Continue**.

To link a resource file to a report design

To use the resources defined in a resource file, you must link the resource file to your reports. For details, refer to the Reporting Studio documentation.

To create report libraries

To create a report library, you must first create an initial report library on the LN server. Subsequently, you must complete the new library in the Infor LN Reporting Studio. For details, see the procedure steps.

To create a report library:

Step 1: Create an initial report library

To create an initial report library:

1. Start the Reports (ttadv3530m000) session. Make sure you select the proper development VRC as your current package VRC.
2. Click **New**. A question "Start report generator?" is displayed.
3. Click **No**. The Reports (ttadv3130s000) details session starts.
4. Enter the module and report code and the description for the new report library. In the **Report Type** field, select "Report Library".
5. Save the new report library and close the details session.

The initial report library is stored in the report sources folder and has a ".rptlibrary" filename extension.

Step 2: Transfer the report library to your client PC

Before you can edit a report library in Reporting Studio, you must transfer the report library to your client PC:

- If SCM is not active, for example if LN runs on a Windows platform, you must export the report library before you can edit it.
- If SCM is active, you must check out and export the report library before you can edit it.

To transfer a report library to your client PC:

1. In the Reports (ttadv3530m000) session, take one of the following steps:
   - If SCM is active, select the report library and click **Check-Out**.
   - If SCM is not active, select the report library and, on the appropriate menu, click **Export RPT file...**.

   The **Export/import RPT file** dialog appears.
2. In the **Export to** field, enter the destination directory and file name. The default export destination is the ${BSE_TMP} directory on your client PC.

3. Click **Continue** to transfer the report library.

Step 3: Edit the report library

Start Reporting Studio and open the report library. Enter the desired code in the library. You can, for example, define functions you want to share in multiple reports.

For details, refer to the Reporting Studio documentation.

Step 4: Upload the report library into the LN data dictionary

Once you have completed a report library, you must upload the report library into the data dictionary on the LN server:

- If SCM is not active, you must import the report library on your LN server.
- If SCM is active, you must import and check in the report library.

To upload a report library:

1. Start the Reports (ttadv3530m000) session.
2. Take one of the following steps:
   - If SCM is active, select the report library concerned and click **Check-In**.
   - If SCM is not active, select the report library concerned and, on the appropriate menu, click **Import RPT file...**.

   The **Export/import RPT file** dialog appears.
3. In the **Import from** field, enter the directory and file name of the exported/checked out report library on your client PC.
4. Click **Continue** to import the report library.
5. If SCM is active, a question "Do you want to check in this Report?" is displayed. Click **Yes** and enter a revision text to check in the report library.

To link a report library to a report design

To share the functionality in a report library, you can link the library to multiple report designs. For details, refer to the Reporting Studio documentation.

Example- code in a report design to link report libraries

The following code shows how the ChartDataFinal and GeneralFinal report libraries are linked to a report design.

```
<list-property name="libraries">
        <structure>
            <property name="fileName">ChartDataFinal.rptlibrary</property>
            <property name="namespace">ChartDataFinal</property>
        </structure>
        <structure>
            <property name="fileName">GeneralFinal.rptlibrary</property>
            <property name="namespace">GeneralFinal</property>
```

```
      </structure>
</list-property>
```

To create a print session for query-based reports

To create a print session that can print a query-based report, take the following steps:

Step 1: Define a print session

1. Start the Sessions (ttadv2500m000) session. Make sure you select the proper development VRC as your current package VRC.
2. Click **New**. The Sessions (ttadv2100s000) details session starts.
3. Enter the module and session code and the description for the new session, and specify the following session properties:
   - **Standard Script**: No
   - **Generate Script**: No
   - **Program Script**: <The program script's code, consisting of package code, module code and the first 4 characters of the session code.>
   - **Main Table**: <Empty>
   - **Session Type**: Print
   - **Window Type**: Dialog
   - **Main Session**: Yes
   - **Synchronized Dialog**: <Empty>

   For the other fields, use the default values.
4. Save the new session and close the Sessions (ttadv2100s000) details session.
5. In the Sessions (ttadv2500m000) overview session, click **Compile**.

Step 2: Link a query-based report to the session

1. In the Sessions (ttadv2500m000) session, select the new session and, on the appropriate menu, click **Reports...**. The Reports (ttadv3530m000) session starts.
2. Click **New**. The Add Report to Session (ttadv2112s000) session starts.
3. Enter the report group and the serial number. Select the desired report and click **OK**.
4. Save the changes and close the Reports (ttadv3530m000) session.

Step 3: Edit the session's UI script

1. In the Sessions (ttadv2500m000) session, select the new session and, on the appropriate menu, click **Program Scripts...**. The Program Scripts / Libraries (ttadv2530m000) session starts.
2. Select the script and click **Edit / View Script ...**. The script editor starts.

3. Edit the script. For example:
   - Declare the variables you want to use as input fields on the session's form. Note: the variable names must be identical to the variable names you defined in the Reporting Studio.
   - Define defaults for the input fields on the form.

   See the following sample UI script.
4. Save the script and close the editor.
5. Compile the script and close the Program Scripts / Libraries (ttadv2530m000) session.

Sample UI script:

```
declaration:
 extern domain ttaad.pacc  pack_from fixed
 extern domain ttaad.pacc  pack_to  fixed
 extern domain ttyeno  incl_pack | print packages y/n

|************************** field sections ***********************************

field.pacc_from:
before.zoom:
 ttaad120.pacc = ""

when.field.changes:
 pack_to = pack_from
 display("pack_to")     |#icc2.6.n

field.pacc_to:
before.zoom:
 ttaad120.pacc = pack_from
```

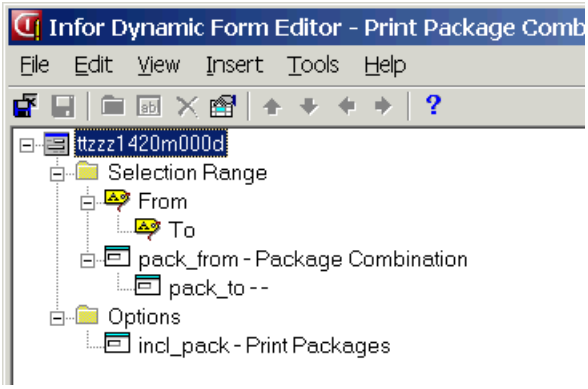
The sample script belongs to a session that prints the package combinations, and optionally, the corresponding package VRCs. Points of attention:

- The pack_from and pack_to variables are used as input fields in the form, to enter the selection range. In the report design in Reporting Studio, pack_from and pack_to are mapped to parameters "1" and "2" that are used in the query's between statement (`between :1 and :2`).
- The incl_pack variable is used as an input field on the form. The value entered by the user (Yes or No) determines whether the packages are printed. In Reporting Studio, the incl_pack variable is used in a print condition. Based on the value of incl_pack, the detail that contains the package VRCs is printed or suppressed.

Step 4: Design the session's form

1. In the Sessions (ttadv2500m000) session, select the new session and click **Edit Form ...**. The Dynamic Form Editor starts.
2. Edit the form: Add form fields for the input variables you defined in the UI script, and customize the structure of the form until it matches the requirements. See the following figure for an example.
3. Save the form and close the editor.

4. In the Sessions (ttadv2500m000) session, click **Compile**.



Sample form

To run query-based reports

You can run a query-based report in 2 ways:

- From a Web UI homepage.
- Through a 4GL print session on the LN server.

To run a query-based report from a Web UI homepage

To run a query-based report from a Web UI homepage:

1. Start Web UI and log on to your LN server.
2. Open the homepage that contains the report you want to run. The report is displayed in the LN Report Viewer.

For details on homepages, refer to the Web UI online help.

To run a query-based report through a 4GL print session

To run a query-based report through a 4GL print session:

1. Start Web UI and log on to your LN server.
2. Start the print session to which the report is linked.
3. Enter any selection ranges and print options, and click **Print**.
   - If only one report is linked to the session, the report is displayed in the LN Report Viewer.
   - If multiple reports are linked to the session, you must first select the desired report from a list.

Note

The LN Report Viewer only works in combination with Web UI. Worktop users cannot print Report Viewer enabled reports.

From the Report Viewer, you can send a report to, among other things, a printer or a PDF file. For details, refer to *To use the Report Viewer (p. 208)* .

To translate resource files

Resource files contain language-dependent content of LN homepages and query-based reports. Each resource file belongs to a particular homepage or query-based report. For example:

- A resource file linked to a homepage, contains the homepage's title and the titles of the panes used in the homepage.
- A resource file linked to a query-based report contains the report title, column headings, chart titles, and other static labels used in the report. For more information, refer to *Query-based Reporting (p. 219)* .

Note

- Resource files do not contain language dependent LN content such as label and message descriptions.
- Each resource file contains data for only one language. The homepages and reports developed by Infor are delivered with English resource files. To run these homepages and reports in other languages, you must translate the corresponding resource files.
- When a user opens a homepage or report, Web UI automatically downloads the corresponding resource file from the LN server. Web UI automatically selects the resource file that corresponds to the user's language, as defined in the User Data (ttaad2500m000) session, so that the homepage or report is displayed in the user's own language. If there is no resource file for the user's language, Web UI downloads the original (English) resource file delivered with the homepage.

To translate resource files

To translate a resource file, complete the following steps:

Step 1: Export resource files

1. Start the Export Resource Files (ttadv8913m000) session.
2. Enter the required settings and export the resource files to a directory on the LN server. The session generates an ASCII file for each exported resource file.

Step 2: Rename ASCII files

The names of the generated ASCII files have the following structure:

`<filename>_<ISO 639 language code>_<ISO 3166 country code>.properties`

Note

- The country code is optional and used for only a few languages.
- The language code of a report's resource file must be identical to the language code of the resource file of the homepage to which the report is linked. For details, refer to the *Infor Enterprise Server Web UI - Installation and Configuration Guide (U8715 US)*.

Rename the ASCII files so they match the target language code and optionally the target country code. Ensure you use the correct ISO codes, because otherwise the Web UI cannot read the resource file at runtime.

For example, the Export Resource Files (ttadv8913m000) session has generated the following ASCII file: `bwhwmdwarehousemanagerhomepage_en.properties`.

- To translate the file into German, rename the file to `bwhwmdwarehousemanagerhomepage_de.properties`.
- To translate the file into French, rename the file to `bwhwmdwarehousemanagerhomepage_fr.properties`.
- To translate the file into Dutch, rename the file to `bwhwmdwarehousemanagerhomepage_nl.properties`.

Step 3: Translate content of ASCII files

The ASCII files consist of lines with the following structure: `resource ID=resource description`. For example, the `bwhwmdwarehousemanagerhomepage_en.properties` file contains the following lines:

```
warehousemanagerhomepage.whinh2400m000=Warehouse Orders
warehousemanagerhomepage.whinh2100m000=Warehouse Orders
warehousemanagerhomepage.whinh4130m000=Create Shipments
warehousemanagerhomepage.workload=Warehouse Manager Tasks
warehousemanagerhomepage.Warehouse_Orders=Warehouse Orders
warehousemanagerhomepage.whinr1400m000=Inventory Transactions
warehousemanagerhomepage.whina1210m000=Inventory Valuation
```

Translate the resource descriptions in the ASCII files (the text behind the "=" in each line) to the target language.

Step 4: Import resource files

1. Start the Import Resource Files (ttadv8923m000) session.
2. Enter the required settings and import the translated ASCII files into a VRC in the LN environment.

Appendix A
Glossary

A

4GL engine

The program that provides default functionality for a session to prevent application programmers from having to develop a session from scratch. The 4GL Engine, formerly called standard program (STP), is used because essentially sessions are alike. In addition to providing default functionality, the 4GL Engine also provides a mechanism to change the 4GL Engine's default behavior, and to program dedicated functionality for a specific session. Each time a session is started, a separate 4GL Engine instance is activated to handle the session.

Synonym: standard program

Acronym: STP

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.

Base VRC

A means 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, but can also be a code not related 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.

BOI

See: *Business Object Interface (p. 238)*

Business Object

A Business Object is an object understandable 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, that 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 these tables that are 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 and third-party applications and the LN software, as well as connecting 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.

Acronym: BOI

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.

Check in

A process that releases the checked-out software component and copies the software component from the *Development VRC (p. 239)* to the *Original VRC (p. 240)* . This process also stores a historical version.

A revision text for the changed software component is required for a check-in process.

Check out

A process that locks the software component for other developers. During the check-out phase, other related components are locked as well while the component can be updated and tested. A copy of the component is made from the original VRC to the Development VRC.

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 etc. 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.

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.

Customization

A derived product, meant to fulfill the requirements of a specific customer or group of customers.

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.

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.

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.

Form command

A command:

- that starts a session, function or (sub)menu by means of which a user can carry out a particular task.
- that, as opposed to standard menu commands such as the **Exit** command, must be especially defined for a session tab.

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.

LN 4GL

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

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. These software components will be changed in the *Development VRC (p. 239)* .

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, that determines which version of the software the user can use. In the Companies (ttaad1100m000) session, each company is linked to a package combination, to indicate which version of LN is appropriate to handle the data in that company.

package VRC

See: *Version - Release - Customer (p. 245)*

Package VRC

A version of a package, for example, **tc B61O a cus1**. In general, 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 modify software components only in a particular package VRC.

The code of a package VRC consists of:

- Package code, for example, 'tc'
- A version (VRC) code, for example 'B61O a cus1', built up of:
  - 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.

PMC

See: *Product Maintenance and Control (p. 242)*

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.

Post-requisite

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

Product Maintenance and Control

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

With the PMC tool, you can check all patches against the customer's 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. In addition, using the PMC tool enhances the quality of the support being available to customers.

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

Acronym: PMC

Program script

A sequence of instructions that is used to program a number of actions that must take place in addition to the standard program. Two different program scripts are available, namely 3GL scripts and 4GL scripts.

Synonym: UI script

PVRC

See: *Version - Release - Customer (p. 245)*

SCM

See: *Software Configuration Management (p. 242)*

Software Configuration Management

With software configuration management, a developer can modify and test an own revision of a software component. Using a check-out and check-in functionality, a software component is locked for others developers. This method guarantees that no more than one developer can modify the same software component at the same time.

Acronym: SCM

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 following statuses describe the progress of the maintenance of solutions, Collections, and patches. To keep the descriptions readable, only solutions are described. However, in each case, collections and patches are intended as well, unless explicitly excepted.

The status of the solution is only used at the distributor side. A history of status changes is recorded in the **Maintenance History** table.

To distinguish maintained solutions at the distributor side from imported solutions at the recipient side, the status is cleared when the solution is imported. This also applies to patches, but not to collections. The reason is that a collection is not known as such at the recipient side. Only the solutions contained in the collection are known at the recipient side.

- **In progress**
  The initial status of the solution upon creation.
- **Dependencies defined**
  The dependencies between the various solutions are defined, and the solution is ready to be exported. You must select this status manually, because you might want to create dependencies that cannot be created automatically.
- **Exported**
  The solution is exported. The system handles this status. You cannot set the status manually. However, you can set a solution with the status Exported back to In Progress, Solved, Completed, or Dependencies Defined. If you change anything in an exported solution, the status is always set back to Dependencies Defined, and you must export the solution again. Before you can release a solution, the status must be Exported.
- **Released**
  The solution is released, which means that the solution is frozen. You can no longer change anything in the solution. To release a solution, you must change the status from Exported to Released, and you must save the solution with this status.

standard program

See: *4GL engine (p. 237)*

STP

See: *4GL engine (p. 237)*

Table

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

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

Example

Table: tc mcs010 **Countries**

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

Table fields

UI script

See: *Program script (p. 242)*

Undo check-out

A command of *Software Configuration Management (p. 242)* that deletes the checked-out software component from the *Development VRC (p. 239)* .

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.

Version - Release - Customer

The version - release - customer (VRC) code is an identification of a stage in the development of the LN software. An example of a VRC is *B61_a_ams*.

A VRC code consists of:

- **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 that is contained in the preceding VRC and not explicitly modified or set to expired in the current VRC will also be available in the current VRC.

Synonym: package VRC

Acronym: VRC, PVRC

VRC

See: *Version - Release - Customer (p. 245)*

Index

4GL engine, 237
4GL Reports, 211
appropriate menu, 237
architecture, 150
authorizations, 57
base VRC, 163
Base VRC, 237
base VRC combination, 163
Base VRC combination, 237
benefits, 149
BOI, 238
Business Object, 238
Business Object Interface, 238
Chart, 238
Check in, 238
check-in and check-out procedure, 140
Check out, 238
collection, 153
    create, 166
Collection, 238
Copy software components, 144
Co-requisite, 239
Customization, 239
dependency, 151
Dependency, 239
Development VRC, 239
Export and import procedure, 145
Export VRC, 239
feature pack, 154
    create, 168
Feature Pack, 239
feature packs and patches, 153
Form command, 240
Homepages
    development, 188, 192
    functions, 192
homepages
    introduction, 185
    structure, 186
    Web UI, 185, 186, 188, 192
Infor LN Development Tools
    4GL Engine, 15
    application data dictionary, 53
    chart, 16
    component, 65
    Customization, 125
    DAL, 17
    DAL 1, 17
    DAL 2, 17
    dashboard, 21, 104
    Data Access Layer, 17
    data dictionary, 53
    data model, 66, 71
    development process, 61
    DFE, 75, 78
    DLL, 31
    domain, 23, 66, 71
    Dynamic Form Editor, 75, 78
    form, 25, 75, 78, 88
    form command, 85
    function, 28
    index, 87
    introduction, 13
    label, 29
    Language Translation Support, 122
    library, 31
    LTS, 122
    menu, 32
    message, 33
    MMT session, 34, 117
    Multi Main Table session, 34, 117
    print session, 88
    Program script, 50
    query, 46
    question, 37
    reconfigure, 71

report, 38, 96, 99
Report Editor, 96, 99
Report Layout, 99
report script, 41
Report Script, 99
runtime data dictionary, 53, 71
session, 42, 73
software component, 65
software components, 15
SQL query, 46
standard command, 85
table, 66, 71
table definition, 48, 66, 71
Translation, 122
UI script, 50
zoom button, 85
introduction, 149
Label, 240
LN 4GL, 240
Obsolete solution, 240
Original VRC, 240
overview, 150
Package combination, 127**,** 240
one-step software environment procedure, 143
standard software environment procedure, 141
package VRC, 245
Package VRC, 128**,** 241
one-step software environment procedure, 143
standard software environment procedure, 141
patch, 158
create, 167
Patch, 241
PMC distributor, 241
functionality, 159
parameters, 162
policy, 159
session summary, 181
setup procedure, 162
PMC, 242
architecture, 150
base VRC, 163
base VRC combination, 163
benefits, 149
collection, 153
create collections, 166
create feature packs, 168
create patches, 167
create solutions, 164
dependency, 151
distributor functionality, 159
distributor parameters, 162
distributor policy, 159
distributor session summary, 181
distributor setup procedure, 162
feature pack, 154
feature packs and patches, 153
individual solution, 152
introduction, 149
overview, 150
patch, 158
recipient functionality, 160
where to find the PMC module, 151
PMC recipient, 241
functionality, 160
Post-requisite, 241
Pre-requisite, 242
Product Maintenance and Control, 242
Program script, 242
Purge a package VRC derivation structure procedure, 146
PVRC, 245
Reporting
device, 217
homepages, 199, 235
overview, 203
print session, 232
project, 207
query-based reporting, 219
query-based reports, 208, 222, 232, 234
Reporting Studio, 199, 203, 207, 208, 213, 213, 217, 218, 219, 222, 228, 230, 232, 234, 235
report libraries, 230
report project, 207
reports, 199, 208, 213, 213, 217, 218, 219, 222, 228, 230, 234, 235
report viewer, 208, 217
report viewer device, 217
resource files, 199, 228, 235
session-based reporting, 213
session-based reports, 208, 213, 217, 218
translation, 199, 235
Reports, 211

Report scripts, 211
SCM, 138**,** 242
check-in and check-out procedure, 140
Software components
copy software components procedure, 144
export and import procedure, 145
purge a package VRC derivation structure procedure, 146
Software Configuration Management, 242
Software configuration management (SCM), 138
check-in and check-out procedure, 140
software environment, 130**,** 136
Software environment
one-step software environment procedure, 143
standard software environment procedure, 141
Solution, 243
solution
create, 164
individual solution, 152
Solution status distributor, 243
standard program, 237
STP, 237
Table, 244
UI script, 242
Undo check-out, 244
Update, 244
Update VRC, 245
Version and release management
copy software components procedure, 144
export and import procedure, 145
introduction, 127
one-step software environment procedure, 143
overview, 127
package combinations, 127
package VRCs, 128
purge a package VRC derivation structure procedure, 146
SCM check-in and check-out procedure, 140
software configuration management system (SCM), 138
software environment, 130, 136
standard software environment procedure, 141
VRC derivation, 133
Version - Release - Customer, 245
VRC, 245
VRC derivation, 133
Web UI
homepages, 185, 186, 188, 192