



Configuration Console Reference Guide: Landmark Pattern Language (LPL)

Release 2022.x

Important Notices

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

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

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

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

Trademark Acknowledgements

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

Publication Information

Release: Infor Landmark Technology 2022.x

Publication Date: September 30, 2022

Document code: lmrk_2022.x_lmrklpltg__en-us

Contents

About this guide.....	4
Contacting Infor.....	4
Available learning resources.....	5
Other notes.....	6
Chapter 2: Examples by use case.....	7
Working with fields.....	7
Working with dates.....	10
Action logic and entrance and exit rules.....	11
Form validation and processing.....	12
Changing form UI.....	12
Changing list UI.....	13
Workflow and integration.....	13
Chapter 3: Examples by component.....	19
Buttons and links.....	19
Dates.....	19
Fields.....	20
Forms.....	21
Lists.....	23
Processes.....	24

About this guide

This guide is a reference for Landmark Pattern Language (LPL) syntax, which can be used to personalize or configure Landmark applications. The content is provided in Backus–Naur form (BNF) notation.

Intended audience

This guide is for developers, administrators, and power users who use Configuration Console for application and security configurations, build and configure reports, or build web service interfaces.

Related documents

Configuration Console User Guide

View document

[Configuration Console Reference Guide: Landmark Pattern Language \(LPL\)](#)

Contacting Infor

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

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

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

Available learning resources

There are a number of resources available to you when using the Configuration Console and Landmark Pattern Language.

Prerequisite knowledge

To better understand the information presented in this guide, you should take a course on Landmark Pattern Language syntax and Configuration Console. There are classes available on Infor Campus. It is also helpful to be familiar with how to read BNF notation.

In particular, Infor recommends taking Lawson Technology: v11 Using Landmark Pattern Language (LPL). The focus of this course is on using the LPL Editor and the Configuration Console to configure business classes, fields, and actions in Landmark applications and to modify user interface elements such as menus, pages, forms, and lists.

Another recommended course is Lawson Technology: Designing and Administering Configuration Console.

Other notes

Non-configurable features

The following objects are not configurable using Configuration Console:

- matrix forms
- field definitions
- navigations

You must create new user-defined objects rather than modifying delivered code.

Chapter 2: Examples by use case

Use the following use case examples to modify your LPL.

Working with fields

Changing lowercase to uppercase

You can convert lowercase to uppercase using this expression:

```
XYZ=Name.GivenName using "%1S"
```

Return first three characters of a field

Create a derived user field to return the first three characters of a field:

```
Derived Fields
ZZZThreeChar is a DerivedField
  type is Alpha size 3
  default label is untranslatable:"Display first three characters"
  return Reference1[0:3]
```

Return substring of a field

Create a derived user field to return a substring of a field. This example returns the first four characters of a field..

```
Derived Fields
ZZZThreeChar is a DerivedField
  type is Alpha size 50
  return Reference1[1:4]
```

Or, this example returns the third through sixth characters.

```
Derived Fields
ZZZThreeChar is a DerivedField
  type is Alpha size 50
  return Reference1[3:6]
```

If statement to derive fiscal quarter

First, create a derived date field as the current date.

```
Derived Fields
  ZZZMyDate is a DerivedField
    type is Date
    return current date
```

Then create another derived date field to extract the quarter number.

```
Derived Fields

  ZZZQuarter is a DerivedField
    type is Numeric size 1
    if (ZZZMyDate month >= 1
      and   ZZZMyDate month <= 3)
      ZZZQuarter = 1
    else
      if (ZZZMyDate month >= 4
        and   ZZZMyDate month <= 6)
        ZZZQuarter = 2
      else
        if (ZZZMyDate month >= 7
          and   ZZZMyDate month <= 9)
          ZZZQuarter = 3
        else
          if (ZZZMyDate month >= 10
            and   ZZZMyDate month <= 12)
            ZZZQuarter = 4
```

Named Types

Named Type is an alternative to using Field Type. Use this if you want to mimic the behavior of a field with the type and size already defined.

The screenshot shows the 'Field' configuration interface. The 'Business Class' is set to 'Test1'. The 'Field Name', 'Default Label', 'Field Type', and 'Initial Value' fields are empty. The 'Default Value' field is also empty. The 'Named Type' dropdown is highlighted in yellow. The 'Optional' radio button is selected. The 'Required When' field is empty. The 'List for Defined Field' modal is open, showing a list of defined fields: ACHPrenotification, ADBAccountMaster, and ADBAggregateBalance.

Return two digit month

Use this example to extract the single digit month elements from today's date and then build a string datatype to display a two digit month the different date elements.

First, create a new derived user field set to the current date:

```
Derived Fields

ZZZCurrentDate is a DerivedField
  type is Date
  return current date
```

Next, create a derived user field to extract the month. This will return a single digit.

```
Derived Fields

ZZZCurrentMonth is a DerivedField
  type is Date
  return (ZZZCurrentDate month)
```

Then create another derived user field to determine if the month is double digit or single digit. If it is a single digit, add a zero (0) to the front.

```
ZZZMonthTwoChar is a DerivedField
  type is Alpha size 2
  if (ZZZMonth size < 2)
    return ("0" + ZZZMonth)
  else
    return (ZZZMonth)
```

Working with dates

Current date variable

You can view details about where to define current date variables in the "General Definitions" section of the *Configuration Console Reference Guide: Landmark Pattern Language (LPL)*.

Dates and Time Stamps

```

CreateDate ::= create date // returns the most recent create date from the current 'as of' date

CreateEffectiveDate ::= create effective date // returns the most recent create effective date from the current 'as of' date

CreateStamp ::= create stamp[.actor] // references the CreateStamp in the business class
// by itself it is the actual TimeStamp when the instance was created
// the actor keyword references the actor who created this instance

CurrentDateTime ::= ( current time
                     | [system] current date
                     | [system] current year
                     | [system] current period
                     | [system] current timestamp
                     | [system] current anniversary
                     )

DurationVars ::= ( duration begin date
                  | duration end date
                  )

UpdateStamp ::= update stamp[.actor] // references the UpdateStamp in the business class
// by itself it is the actual TimeStamp when the instance was last updated
// the actor keyword references the actor who last updated this instance

```

Add a date or a timestamp to a field

You may want to append a timestamp to a value in a field. To do this, create an alpha derived field and return the field and the data value. This example adds a new derived field to a form that adds the year and day of an entered date field to content entered in another field.

Create a derived field.

```

ZZZTimeStamp is a DerivedField
  type is AlphaUpper size 30
  default label is untranslatable:"Timestamp LPL Example"
  return (RequisitionDescription + "_" + RequestedDeliveryDate year + "_" + Requested
DeliveryDate day)

```

Add the derived field to a form that automatically populates the field value using the entered timestamp and description.

```

RequisitionDescription
  when value changed

```

refresh ZZZTimeStamp
ZZZTimeStamp

From Company
 Q=

From Location
 Q=

Location Rule
 Q=

Single Document To A Purchase Order

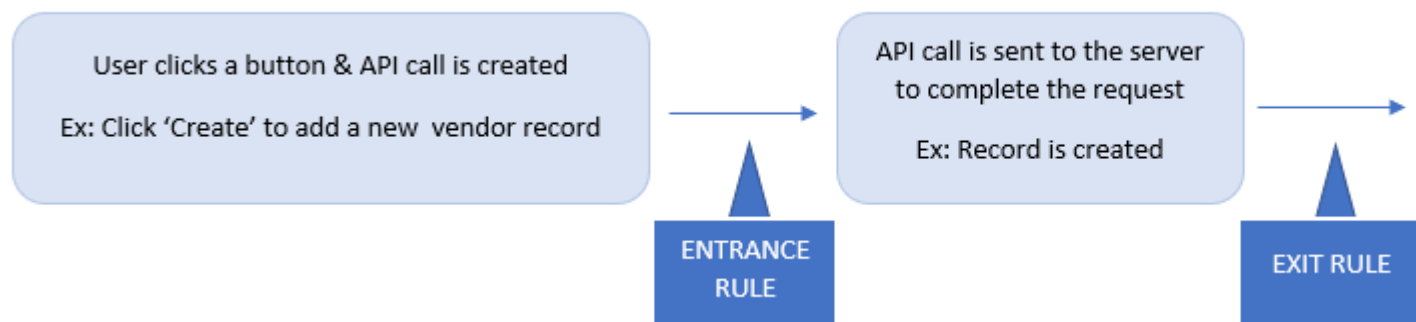
Requisition Description

Timestamp LPL Example

Action logic and entrance and exit rules

Entrance and Exit rules

This diagram shows when an entrance rule or exit rule is triggered:



As an example, these invoke options are available when using exit rules for Update Employee:

- `invoked.StartDate=StartDate`
- `invoked.RelationshipToOrganization=RelationshipToOrganization`
- `invoked.RelationshipStatus=RelationshipStatus`
- `invoked.WorkType=WorkType`

Form validation and processing

Confirmation versus constraint

A confirmation will allow the user to continue with the update. A constraint will prevent the action.

Constraint First set Exists

Use this code syntax to check to see if a record set exists for a business class. Upon evaluation, if the system encounters any first records in this business class, it is interpreted as true. If there are no records, it will be false. If true, enforce the constraint.

This can be used to prevent the user action if a record does not exist.

This example prevents the ability to release JE if a related supporting document has not been attached to the JE record.

```
constraint (first GeneralLedgerJournalDocument set exists)
  "PleaseAttachDocumentationBeforeReleasing"
```

Changing form UI

Label is not translatable

The camel-case phrases for titles, labels and messages generally acts as a key to a translatable string (so that it can be translated into a different language). It is untranslatable because no translations have been provided. You can make them translatable, but you have to supply the values.

In many instances, you can add them with a globe icon, as this example shows:

```
FinanceDimension1 is a BusinessClass
  owned by GeneralLedger


Persistent Fields
  XYZPrimaryMarket          is Numeric size 1
  States
    Commercial value is 1
    Residential value is 2
    Government value is 3
  default label is configuration:"PrimaryMarket"
  translation for <locale> is "<value>"
```

Changing list UI

Configure list to link to a URL

This example configures an existing list field that can link to a URL.

```
ActivePrimaryWorkAssignmentRel.Location.Description
  label is "Location"
  link is external "https://google.com/search?source=hp&q=<Name.PreferredGivenName>+<Name.PreferredF
  sort order is ByPrimaryLocationDesc
```



Relationship	Location	Organization Unit
Employee	St Paul	Outside Sales
Employee	St Paul	7000

Workflow and integration

Trigger an IPA process

Use LPL to trigger an IPA service from a UI object, and send data from the UI to the process as input data. You can insert this code in an entrance or an exit rule. This example code triggers an IPA service called LPLService, which in turns triggers an IPA process called TestLPL.

Business Class

Action Name

Default Label

In State

Action Type


☐ Effective Date Required ☐ Reason

Entrance Rules **Exit Rules**

```
initiate LPLService process
  resume on error
  title is "TriggeredFromLPL"
  Variables
    TransactionDate
    UniqueJournalID
    AccountingEntity
    GeneralLedgerJournalControl
```

ID: Start

Type:  START

 Breakpoint

Name: Start

Process Variables Common

Process Variables:

Type	Variable name	Expression
String	JENumber	GeneralLedgerJournalControl
String	AcctEntity	AccountingEntity
String	Jeld	UniqueJournalID
String	TxDate	TransactionDate

*Work Unit	431	Activity Node	1	Activity Name	Start
Activity Type	START	Activity Status	Completed		
Start Date	02/21/2019 05:00:04 PM	End Date	02/21/2019 05:00:04 PM		

Log

Variable

Log

Executing Start Activity...

JENumber = 1

AcctEntity = 4000

Jeld = ABC_4000_2018_2_21

TxDate = 20180221

Variables:

AcctEntity(Type=String) = 4000

TxDate(Type=String) = 20180221

Jeld(Type=String) = ABC_4000_2018_2_21

JENumber(Type=String) = 1

Activity completed

Trigger an IPA Process from a custom form

This example creates a custom form to trigger an IPA flow. When a user adds or updates data on this form, it triggers an IPA process, sending the data from the form to the process. This will also allow you to add a button to the form to manually trigger the IPA process.

Create the User Business Class

```
process is vty
```

Ontology

```
symbolic key is RemittanceEmail
```

Persistent Fields

```
VendorGroup
PayGroup
Vendor
PaymentFromDate is Date
PaymentToDate   is Date
Status          is Numeric size 1
  States
    Active   value is 1
    Inactive value is 0
```

Actions

```
Create is a Create Action
```

```
Update is an Update Action
```

```
Delete is a Delete Action
```

```
Submit is an Instance Action
```

Action Rules

```
initiate TestingIPA process
  title is "TestingIPA"
```

Variables

```
VendorGroup
PayGroup
Vendor
PaymentFromDate
PaymentToDate
Status
```

Create the custom form:

Remittance Email Primary

 Submit  Save  Save And New

*Remittance Email

Vendor Group

Pay Group

Vendor

Status

Payment From Date Payment To Date

Configure the IPA service:

Service Definition

      |

Service Definition

Service:

Description:

☒ Service Is Enabled

Criteria Name 1:



Criteria Name 2:

Criteria Name 3:

Processes

Variables

   |

Process Definition		Service Process Is Enabled	Criteria Name
(A) <input type="text" value="TestIPAProcess"/>		 <input type="text" value=""/>	(A) <input type="text"/>
TestIPAProcess		Yes	

Records Per Page:

To test:

- 1 Open the new form and enter data.

- 2** Save the data, which enables Submit.
- 3** Click Submit.
- 4** Verify a workunit was created in IPA.

Chapter 3: Examples by component

Use these examples, organized by object, to modify your LPL.

Buttons and links

Adding a button link to a form

You can use `button` and `link is` to add a button link to a form. For example, you may want to add a button that opens another internal navigation, or an external link.

Internal navigations are links from within the business class, or a related business class. New navigations can not be created; only existing navigations are available for use. External navigations can be created using `link is` and specifying a URL.

Example scenario: A button is added to the Request Update Account form that will show General Ledger Transactions.

```
button of "Transactions"  
  link is external <General Ledger Transactions URL>
```

Dates

Add date or timestamp to a field

```
ZZZTimeStamp is a DerivedField  
  type is AlphaUpper size 30  
  default label is untranslatable:"Timestamp LPL Example"
```

```
return (RequisitionDescription + "_" + RequestedDeliveryDate year + "_" + RequestedDeliveryDate day)
```

Fields

Display length of data entered in a field

Use this LPL code to determine the number of characters entered in a field.

Create a transient field:

```
Transient Fields
ZZZGetFieldLength is Numeric size 2
Default label is untranslatable:"Get field length"
```

Then create a form configuration and set the value of the transient field = to <user input field> size. In this example, it's RequisitionDescription. In order to visually display the output, I'm using the when value changed to automatically update the value to the transient field that has been added to a form. Note that if you don't want to display the field on the form, but want to use the when value changed for another purpose in your code, make the field hidden

```
RequisitionDescription
when value changed
ZZZGetFieldLength = RequisitionDescription size
```

Restrict spaces entered in a field

Use this example to restrict spaces from getting entered into a field.

First, create a derived user field to be used in a subscript.

```
Transient Fields
ZZZSubscript is Numeric size 2
```

Then, use an action configuration with an entrance rule to loop through the values entered by the user.

```
Vendor.ZZZSubscript = Vendor.Reference1 size
while (Vendor.ZZZSubscript > blank)
  constraint (Vendor.Reference1[Vendor.ZZZSubscript] entered)
  "EmbeddedSpacesNotAllowed"
  Vendor.ZZZSubscript -= 1
```

Increasing the size of a text field

email from Andrew on 3/16

```
use text area of n lines
E.g.
```

```
Description
  use text area of 10 lines
```

Ensure a field is entered

If you need to ensure a field is entered, you can create an action configuration and use LPL to create an entrance rule. This is typically done with an update or create action. This is an entrance rule that ensures the Reference1 field is populated if a Vendor record is active.

```
if (Vendor.VendorStatus.Active)
  constraint (Vendor.Reference1 entered)
    "Reference1FieldIsRequiredOnActiveVendorRecords"
```

Return first three characters of a field

Example to create a derived user field to return the first three characters of a field.

```
Derived Fields
ZZZThreeChar is a DerivedField
  type is Alpha size 3
  default label is untranslatable:"Display first three characters"
  return Reference1[0:3]
```

Forms

Adding a button link to a form

You can use `button` and `link is` to add a button link to a form. For example, you may want to add a button that opens another internal navigation, or an external link.

Internal navigations are links from within the business class, or a related business class. New navigations cannot be created. Only existing navigations are available for use. External navigations can be created using `link is` and specifying a URL.

Example scenario: A button is added to the Request Update Account form to show General Ledger Transactions.

```
button of "Transactions"
  link is external <General Ledger Transactions URL>
```

Save on next

Use `save on next` as part of a `WizardForm` or `CompositeForm`. The default LPL behavior is to save when a business class boundary is crossed. To save going from one panel to another, use `save on next` on the panel. This can be useful when a multiple step process is lengthy.

Example scenario: During the candidate wizard, use `save on next` to save each panel of information.

```
BasicCandidateInformation is a Panel
  valid when (!ProfileExists)
  form is RSSEnterNewCandidate
  save on next
```

```
AdditionalCandidateInformation is a Panel
form is RSSAdditionalCandidateInformation
```

Search form is inline

You can configure or personalize a list and define a search form inline, instead of using an existing form. You can determine which fields are displayed on search forms.

Use `search form is inline` to define a search form inline.

Example scenario: You can define a search form inline and show the product, product name, unit cost, available status, and quantity.

```
ProductListInlineSearch is a list
  title is "ProductsWithInlineSearch"
  search form is inline
  Layout
    Paragraph
      Manufacturer
      ProductName
  DisplayFields
    Product
      sort order is ProductNumberSet
      is default
    ProductName
      sort order is ProductNameSet
    UnitCost
    AvailableFlag
      label is "IsAvailable"
      allow update
    Quantity
```

Specifying a different form to use when printing to PDF

Use `print form is` to define a different form when printing to PDF.

This action replaces the form that is printed.

Example scenario: During the print process, `print form is` can be used to print a form that excludes fields with personally identifying information.

```
FormDefinition ::=
  <FormName> is a Form
  [print form is <FormName>]

CompositeFormDefinition ::=
  <FormName> is a (CompositeForm | WizardForm)
  [print form is <FormName>]
```

Visited action is

Use `visited action is` in a `WizardForm` or `CompositeForm` when a user has accessed a step or panel. Typically `visited action is` is used to set a Boolean condition in the application, such as a step is in progress. `visited action is` calls an action to set the condition.

`Visited action is` is often used in conjunction with `in progress when` or `completed when`.

Example scenario: During the life event update process, `visited` action is can be used to indicate which panels have been seen by the user.

```
LifeEventDetails is a Panel
    form is EmployeeLifeEventDetailsResponsive
    visited action is SetViewedDetailsVisited
    completed when (ViewedPanels.ViewedDetails)
Instructions is a Panel
    form is InstructionTextResponsive
    visited action is SetViewedInstructionsVisited
    completed when (ViewedPanels.ViewedInstructions)
NameChange is a Panel
    form is NameChangeResponsive
    visited action is SetViewedNameChangeVisited
    completed when (ViewedPanels.ViewedNameChange)
```

Lists

Changing the list that opens on double-click

Use this sample code to change what page or list opens on a double-click action.

For example, to change the form that opens when you open a resource on Resources.

```
HasRole1
    when (actor has role "role1")

Then the visible when (!HasRole1) should work.
```

Specifying alternate lists of fields to use when exporting to CSV

Use `csv fields from` to define data to be exported when exporting to CSV.

For example, to exclude fields with personally identifying information from being exported.

```
csv fields from <ListName>
```

Note:

The list cannot be a `TreeView`, `OrgChartView`, `CubeView`, or `ColumnarView`.

The list is used to define display fields and it is not a complete replacement for the list.

Specifying alternate lists of fields to use when printing to PDF

Use `print fields from` to define data to be displayed when printing to PDF.

For example, to exclude fields with personally identifying information from being printed.

```
print fields from <ListName>
```

Note:

The list cannot be a `TreeView`, `OrgChartView`, `CubeView`, or `ColumnarView`.

The list is used to define display fields and it is not a complete replacement for the list.

Processes

Trigger an IPA Process from a custom form

This example provides guidance if you need to create a custom form to trigger an IPA flow. When a user adds or updates data on this form, it will trigger an IPA process, sending the data from the form to the process.

This will also allow you to add a button to the form to manually trigger the IPA process


```
process is vty
```

Ontology

```
symbolic key is RemittanceEmail
```

Persistent Fields

```
VendorGroup
```

```
PayGroup
```

```
Vendor
```

```
PaymentFromDate is Date
```

```
PaymentToDate is Date
```

```
Status is Numeric size 1
```

```
States
```

```
Active value is 1
```

```
Inactive value is 0
```

Actions

```
Create is a Create Action
```

```
Update is an Update Action
```

```
Delete is a Delete Action
```

```
Submit is an Instance Action
```

```
Action Rules
```

```
initiate TestingIPA process
```

```
title is "TestingIPA"
```

```
Variables
```

```
VendorGroup
```

```
PayGroup
```

```
Vendor
```

```
PaymentFromDate
```

```
PaymentToDate
```

```
Status
```