



Infor Epiphany Sales and Service Mobile Wireless Implementation Guide

Copyright © 2015 Infor

Important Notices

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

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

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

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

Trademark Acknowledgements

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

Publication information

Release: 10.0.1

Publication Date: July 15, 2015

Contents

Chapter 1: Introduction.....	7
About Infor.....	7
Purpose of this Guide.....	7
Product Documentation.....	7
Installation and Configuration Guide.....	7
Implementation Guide.....	8
Reference Guide.....	8
Administrator's Guide.....	8
Computer Telephony Integration Guide.....	8
Integration Guide.....	9
Online Help.....	9
Viewing Release Notes and Manuals.....	9
Printing This Document.....	9
Contacting Customer Support.....	10
Location of the Platform Support Matrix.....	10
Chapter 2: Prerequisites.....	11
Supported Application Server Platforms.....	11
Chapter 3: Working with Mobile Wireless Widget Types.....	13
tr_commandButton.....	14
tr_commandLink.....	15
tr_goLink.....	15
tr_image.....	16
tr_inputDate.....	16
tr_inputText.....	17
tr_panelHorizontalLayout.....	18
tr_selectBooleanCheckbox.....	18
tr_selectOneChoice.....	19
tr_selectOneList.....	19
tr_outputText.....	20
Widgets – How To: Tips and Tricks.....	20
How to configure dependent tr_selectOneChoice.....	20
How to configure tr_goLink to utilize Smartphone features.....	21
How to configure tr_inputText to display numbers and currency.....	22
How to configure tr_inputDate to display date and time.....	22
How to configure tr_selectOneList to assign related BIO and back-fill selected value.....	22

How to configure tr_panelHorizontalLayout to layout widgets.....	26
How to configure Partial Page Rendering (PPR).....	28
Chapter 4: Working with Mobile Wireless Form Types.....	31
NORMAL_TRINIDAD.....	31
LIST_TRINIDAD.....	34
TOOLBAR_TRINIDAD.....	41
LOGIN_TRINIDAD.....	43
FORM SLOT.....	43
Chapter 5: Mobile Wireless Special Form Types.....	45
Mobile Wireless Login Form.....	45
Mobile Wireless Home Page Form.....	47
Mobile Wireless Header Form.....	49
Chapter 6: Events and Extensions.....	53
Mobile Wireless Extensions vs. Desktop Extensions.....	53
Form Events.....	54
preRenderFormOnInit.....	54
preRenderFormOnRedisplay.....	54
preRenderListOnInit.....	55
preRenderListOnRedisplay.....	55
Form Widget Events.....	55
changeEvent.....	55
clickEvent.....	56
preRenderWidget.....	56
returnFromBack.....	57
List Form Events.....	58
listFilterEvent.....	58
listSelectEvent.....	58
listSingleSelectEvent.....	59
listSortEvent.....	60
List Multiple Selection (Not an Event).....	60
listRefreshEvent.....	61
Form Core Extensions.....	61
JSFPrerenderFormSetValue.....	61
Form Widget Core Extensions.....	61
JSFDisableWidgetInNewMode.....	62
JSFHideWidgetInNewMode.....	62
Action Based Core Extensions.....	63
JSFBackFillActionExtension.....	63
JSFContextualCaptionExtension.....	64

JSFCreateRelBetweenBiosAction.....	65
JSFLoginExtension.....	65
JSFMarkDeleteAction.....	66
JSFRefreshBiosAction.....	67
JSFSetupRelBetweenBiosAction.....	67
Defining your own Custom Extensions.....	68
Extension Base Class.....	68
JSF State Interfaces.....	68
Chapter 7: Configuring Navigations.....	69
Overview.....	69
Navigation Configuration.....	69
Event Type.....	70
Navigation Type.....	70
Form Placement.....	71
Navigation Name.....	71
List Order.....	71
Screen (Property Value).....	72
State Handling (Property Value).....	72
Navigation Definition Configuration.....	72
Container.....	72
BIO Path.....	73
Content.....	73
Perspective.....	73
Special-Form type.....	73
Tab List Order.....	73
Overriding Property Values.....	74
State Handling.....	75
Confirm All.....	75
Confirm This.....	75
Confirm Parent.....	75
Discard.....	75
Save All.....	75
Save This.....	76
Save Parent.....	76
Send This.....	76
Chapter 8: Customizing Look and Feel using Infor Stylesheet Identifiers.....	77
Overview.....	77
Customizing Look and Feel.....	77
Stylesheet Identifiers.....	77

About Infor

Infor delivers business-specific software to enterprising organizations. With experience built-in, Infor's solutions enable businesses of all sizes to be more enterprising and adapt to the rapid changes of a global marketplace. With more than 70,000 customers, Infor is changing what businesses expect from an enterprise software provider. For additional information, visit www.infor.com.

Purpose of this Guide

The *Infor Epiphany Sales and Service Mobile Wireless Implementation Guide* provides the procedural information relevant to individuals involved in implementing and customizing the Infor Epiphany Sales and Service Mobile Wireless application.

Product Documentation

The Infor Epiphany Sales and Service product documentation includes the manuals and online help systems described in this section. For a summary of features that are new for this release, late-breaking information about installation and upgrade, and information on fixed or outstanding product issues, see the Infor Epiphany Sales and Service Release Notes. For information on supported platforms, see the Documentation section of the Infor Support Portal, <http://www.inforxtreme.com>.

Installation and Configuration Guide

The *Infor Epiphany Sales and Service Installation and Configuration Guide* includes all of the Infor-specific information required to get the Infor Epiphany Sales and Service applications running. When special configuration is required or recommended for other platform-support software (such as WebLogic, WebSphere, JBoss, SQL Server, Oracle and so on), is also included. The audience for this

book should be familiar with installing and configuring sophisticated enterprise software. They are expected to be experts in their own corporate network configurations and knowledgeable about security topics such as proxy servers and firewalls. General familiarity with database management and maintenance is also assumed.

Implementation Guide

The *Infor Epiphany Sales and Service Implementation Guide* provides procedural information relevant to individuals involved in implementing and customizing the Infor Open Architecture and the Infor Epiphany Sales and Service applications built on it. Implementers typically possess expertise in lightweight Java development, HTML, DHTML, JavaScript, and SQL. They primarily work with the Infor Studio configuration tool (and to some extent with the Designer tools). The *Infor Epiphany Sales and Service Mobile Wireless Implementation Guide* provides additional procedural information relevant to individuals involved in implementing and customizing the Infor Epiphany Sales and Service Mobile Wireless application.

Reference Guide

The *Infor Epiphany Sales and Service Architecture Reference Guide* and the *Infor Epiphany Sales and Service Application Reference Guide* contain technical reference information relevant to implementors involved in implementing and customizing Infor Epiphany Sales and Service at customer sites. These books provide the reference context for the procedural information available in the *Infor Epiphany Sales and Service Implementation Guide*. The *Infor Epiphany Sales and Service Mobile Wireless Architecture Reference Guide* provides additional technical information relevant to individuals involved in implementing the Infor Epiphany Sales and Service Mobile Wireless application.

Administrator's Guide

The *Infor Epiphany Sales and Service Administrator's Guide* provides supervisors, managers, and executives with the information to use the Infor Epiphany Sales and Service and Admin Console functionality to manage the work of their agents and salespeople. Instructions for day-to-day maintenance of the system are included in this book.

Computer Telephony Integration Guide

The *Infor Epiphany Sales and Service Computer Telephony Integration Guide* (CTI) provides the overview and configuration information needed to implement and customize an Infor CTI-enabled product. Its target audience is the implementors who deploy Infor Epiphany Sales and Service at the enterprise.

Integration Guide

The *Infor Epiphany Sales and Service Integration Guide* provides overview and configuration information for the set of tools used to exchange data with a variety of back-end data sources, including generic SQL sources, Java and EJB-based sources, Web services, and other database types (using Infor Enterprise Application Integration, or EAI).

Online Help

Online help documentation for Infor Epiphany Sales and Service is:

- Admin Console
- Sales
- Self-service
- Service
- IBRDesigner
- WorkflowDesigner
- DialogDesigner
- Studio
- Logviewer

These are available for users of the Infor Contact Center, Infor Epiphany Sales, Infor Epiphany Service, Infor Workflow Designer, and Infor Dialog Designer.

Viewing Release Notes and Manuals

Product release notes and manuals are part of the Infor Epiphany Sales and Service package. To view product documentation open the document of interest in Acrobat Reader.

Printing This Document

To print this document at the highest quality resolution, print to a Post-Script driver. Other drivers may not reproduce the screen shots as accurately. This document is designed to be printed on two sides of the page. If your printer is not configured for duplex printing, you may find a blank page at the end of some chapters. This is normal.

Contacting Customer Support

You may contact the Infor Customer Support center by submitting your incident via the web 24x7 at <http://www.inforxtreme.com>, or by placing a call during our scheduled business hours. For a complete listing of our support centers with web addresses and phone numbers, access our support site at <http://www.inforxtreme.com>.

Location of the Platform Support Matrix

For more information on platform support, including hardware and software requirements, see the <http://www.inforxtreme.com> web site.

Supported Application Server Platforms

The Mobile Wireless Application is designed to function on any of the supported platform combinations listed in the latest Sales Service 10.0.1 Platform Support Matrix document.

Working with Mobile Wireless Widget Types

3

A set of new Studio widget types are available for use on forms for Mobile Wireless. Only Mobile Wireless widget types are allowed on forms using Mobile Wireless form types.

Each defined widget will generate one or more Trinidad UI components in the JSPX file for the form.

Every widget type has its own set of properties that affects the behavior of the widget. However, there is a subset of properties that apply to more than one widget-type.

The following properties are common to most widget types:

Property Name	Description
disabled	TRUE indicates that this widget is disabled and cannot be clicked on by the user, although it will still be visible.
immediate	Whether the value is converted and validated immediately in the Apply Request Values phase, or is handled in the Process Validators phase (the default).
inlineStyle	The inline CSS style for this widget.
readOnly	TRUE if the widget is read only and cannot be edited by the user.
rendered	TRUE indicates that the widget should be rendered. FALSE indicates it will not be rendered (hidden).
styleClass	A CSS style reference to style this widget. Leaving this blank will use the default Infor CSS style for that widget.
partialTriggers	The IDs of the widgets that should trigger a partial update.

The properties listed in the table below are common to input related widget types:

- tr_inputText

- tr_inputDate
- tr_selectBooleanCheckBox
- tr_selectOneChoice
- tr_selectOneList

Property Name	Description
label inlineStyle	The inline CSS style for the label of this widget.
label styleClass	The CSS style class to use for the label of this widget.
autoSubmit	If set to TRUE on a form element, the widget will automatically submit the enclosing form when an appropriate action takes place (a click, text change, etc.).
contentStyle	The CSS styles to attach to the content of the widget. For example, you can set the width of that portion to 100 pixels by setting this attribute to "width: 100px".
required	Whether a non-null, non-empty value must be entered. If false, validators will not be executed when the value is null or empty.

tr_commandButton

This widget will generate the Trinidad `<tr:commandButton>` component that renders a button that performs an action when clicked by a user.

Example:



Property Name	Description
partialSubmit	Whether the action should be done through a partial page submit or not. Default is false: no partial page submit.
icon	An URL to an image to use for an icon displayed in the button.

tr_commandLink

This widget will generate the Trinidad `<tr:commandLink>` component that renders a hyperlink that performs an action, when clicked by a user.

Example:

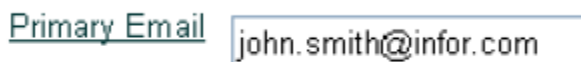


Property Name	Description
partialSubmit	Whether the action should be done through a partial page submit or not. Default is false: no partial page submit.
icon	A URL to an image to use for an icon displayed besides the link.
image	A URL to an image to use for an icon displayed beside the link.
image styleClass	A CSS style class to use for the icon associated with the link.

tr_goLink

This widget will generate the Trinidad `<tr:goLink>` component that renders a special link that integrates with smartphones feature such as E-mail and Telephony.

Example:



Property Name	Description
type	The destination type of this widget. Valid values are "url", "dialit", "mapit" and "mailto". Default is "url"
provider	The map provider to use when the destination type is mapit.

More details are available in the section, "How to configure tr_goLink to utilize Smartphone features" on page 21.

tr_image

This widget will generate the Trinidad <tr:image> component that renders an image.

Example:



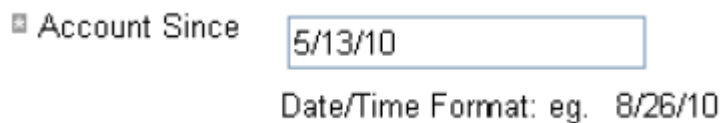
Property Name	Description
source	The URI specifying the location of the image resource

tr_inputDate

This widget will generate the Trinidad <tr:inputText> component widget that can accept a string input from the user in the form of a text box to enter date/time information.

The tr_inputDate has no date picker because it is not supported on most smartphones. Instead, we provide a tooltip that displays the expected date/time format.

Example:



Property Name	Description
type	Specifies what contents the string value will be formatted to include, or parsed. Valid values are "date", "time", and "both".
timeZone	The timezone in which the date and time is displayed.

Property Name	Description
format	The Java date-format string that specifies the format of the date and times displayed in this widget.

tr_inputText

This widget will generate the Trinidad `<tr:inputText>` component that can accept a string input from the user in the form of a text box

Example:

User Name

Property Name	Description
maxLength	The maximum number of characters that can be entered into the text control. Note that this value is independent of the "cols" displayed. If set to 0 or less, the maxLength is ignored.
rows	The number of rows to display in the text control. The default is one. Setting to more than one row precludes the use of some attributes, such as "secret".
secret	A boolean value that only applies to single line text controls. When set to "true", it hides the actual value of the text from the user, and will prevent the actual "value" from being shown to the user.
wrap	Valid Values: soft, hard, off. The type of text wrapping to be used in a multi-row text control. This attribute is ignored for single row inputText. By default (or "soft"), multirow text wraps visually, but does not include carriage returns in the submitted value. Setting this to "off" will disable wrapping; the multirow text will scroll horizontally. Setting it to "hard" specifies that the value of the text should include any carriage returns needed to wrap the lines.
format	An optional Java formatting string that specifies how dates, times, and numerical values are formatted in this widget.

Property Name	Description
precision	The number of digits that should be placed after the decimal point for numerical values.
Default Value	Initial value for a widget associated with a BIO attribute when the focus BIO is a new or query BIO.

tr_panelHorizontalLayout

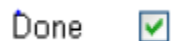
This widget will generate the Trinidad <tr:panelHorizontalLayout> component which is a layout element that arranges its children horizontally.

Property Name	Description
spacerWidth	The width of the spacer item.
spacerHeight	The height of the spacer item.
widgets	The widgets to be grouped in the panel

tr_selectBooleanCheckbox

This widget will generate the Trinidad <tr:selectBooleanCheckbox> component that maps to a standard browser input checkbox, which toggles between selected and unselected states.

Example:

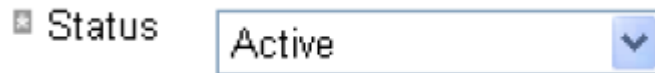


Property Name	Description
disabled if checked	Whether the widget will be disabled if checked.
checked	The value of the BIO attribute if the user selects it.
unchecked	The value of the BIO attribute if the user unselects it.

tr_selectOneChoice

This widget will generate the Trinidad `<tr:selectOneChoice>` component that renders a menu-style component, which allows the user to select a single value from a list of items.

Example:



Property Name	Description
attribute domain	The attribute domain for possible values in the drop-down.
attribute domain dependency	The widget on which the value of the attribute domain depends. The value of the attribute-domain-dependency-form widget determines the attribute domain of the drop-down.
Default Value	Initial value for a widget associated with a BIO attribute when the focus BIO is a new or query BIO.

tr_selectOneList

This widget is a composite widget that generates both the Trinidad `<tr:inputText>` component and the `<tr:commandButton>` component.

It is used like the pop-up widget of the desktop application that maps to an entire BIO instead of a single field.

The `<tr:inputText>` component is always disabled, indicating that a value cannot be entered into it directly. To populate the widget, the user clicks on the `<tr:commandButton>` component which typically navigates the user to a single-select list view to select the related BIO.

Please see the section “How to configure `tr_selectOneList` to assign related BIO and back-fill selected value” for more details.

Example:



Property Name	Description
display attribute	The BIO attribute that is displayed to the end user if the widget is linked to a BIO attribute of type BIO.

tr_outputText

This widget will generate the Trinidad <tr:outputText> component that renders a label with descriptive information.

Example:

Date/Time Format: e.g.
10/13/10

Property Name	Description
escape	An attribute controlling whether output will be escaped for the current markup language or not.
icon	An URL to an image to use for an icon displayed beside the display text.


Widgets – How To: Tips and Tricks

This section provides the “how-to” tips and tricks on the use of the new widget types for some common configuration scenarios.

How to configure dependent tr_selectOneChoice

A dependent tr_selectOneChoice is a dropdown whose domain value depends on the value of another widget.

In the example below, we illustrate this using the **Country** and the **State** widgets on the Address detail form:



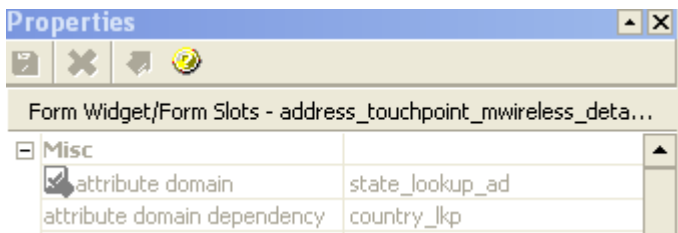
Country

State

The domain value associated with the State tr_selectOneChoice is dependent on the selected value of the Country tr_selectOneChoice.

Here are the steps to configure a dependent tr_selectOneChoice:

- 1 In the Properties window for the State tr_selectOneChoice widget, set the “attribute domain dependency” property to the attribute domain value of the parent tr_selectOneChoice. In this case, it is the attribute domain value of the Country widget we want to set.



Note: state_lookup_ad is a dependent dropdown attribute domain. Please refer to *Studio Online Help* on how to configure a dependent dropdown attribute domain.

2 To refresh the State widget domain value immediately when the Country widget value changes:

- In the Properties window for the Country tr_selectOneChoice, set the “autoSubmit” property to “true”.
- In the Properties window for the State tr_selectOneChoice widget, set “immediate” to **true** and configure “partialTriggers” to associate it with the dependent tr_selectOneChoice widget. In this case, it is the Country widget we want to assign it to.

partialTriggers Vector ...

Vector Values (Custom Filter) [1 Records]				
	Module*	Object Value	Object Val	List Order*
▼				
▶	wireless-ui	address_touchpoint_mwireless_detail_view.country_lkp	form_widget	1
*	customer_...		form_widget	

How to configure tr_goLink to utilize Smartphone features

The tr_goLink widget uses the Trinidad tr:goLink UI component to integrate links to phone numbers, e-mail addresses, maps and HTML links on the smartphones.

The “destination” property of the tr_goLink widget defines the targeted integration link. The supported integration links are:

- mailto – for integrating with Email Client
- dialIt – for integrating with Telephony
- mapIt – for integrating with a Map provider
- url – for HTML hyperlink

The “provider” property defines the map provider to use when the destination property is “mapIt”. Note: we only support Google as the current map provider in this release.

You need to assign a BIO attribute that resolves to the proper destination value to the tr_goLink widget.

Below is an example showing the Dial It! link configuration that is bound to the “formatted_telephone_number” BIO attribute of the telephone BIO.

	Form Widget/Form State (5 Records)			
	Widget*	Widget Type*	Attribute Name*	Widget Name*
0	telephone_number_ui	tr_inputTextForm_widget	area_code (telephone)	area_code
1	telephone_number_ui	tr_inputTextForm_widget	country_code (telephone)	country_code
	telephone_number_ui	tr_inputTextForm_widget	subarea_code (telephone)	subarea_code
2	telephone_number_ui	tr_inputTextForm_widget	format (telephone_number)	format (telephone)
3	telephone_number_ui	tr_inputTextForm_widget	telephone_number (telephone)	telephone_number
4	telephone_number_ui	tr_selectOneForm_widget	telephone_type (telephone)	telephone_type

How to configure tr_inputText to display numbers and currency

The tr_inputText can be used to display number and currency value in addition to normal text value.

The “format” and “precision” are two widget properties that determine the format of the number and currency display value.

Please refer to the *CRM_SS_ImplementationGuide* – Chapter 19 for more details on these properties.

How to configure tr_inputDate to display date and time

The tr_inputDate widget can be used to display date and/or time. The “type” property determines whether the widget is used to display date and/or time.

- date – for displaying date only
- time – for displaying time only
- both – for display date and time

The “format” and “timezone” are two other properties that determine the format of the date/time display value.

Please refer to the *CRM_SS_ImplementationGuide* – Chapter 19 for more details on these properties.

How to configure tr_selectOneList to assign related BIO and back-fill selected value

Most widgets maps to a single field. A tr_selectOneList widget, however, is a widget that can be mapped to an entire BIO. It displays only one attribute of the BIO. Because it is ultimately mapped to an entire BIO, a tr_selectOneList widget typically map to a RELATION attribute.

There are several sets of metadata configuration required for the proper functioning of the tr_selectOneList widget. The configuration differs slightly depending on the usage context of the widget. For example, it can be used as a widget in the detail form or as a search widget in the Advanced Search area in the list form

In the example below, we illustrate the first usage using the Territory tr_selectOneList widget on the Organization detail form.

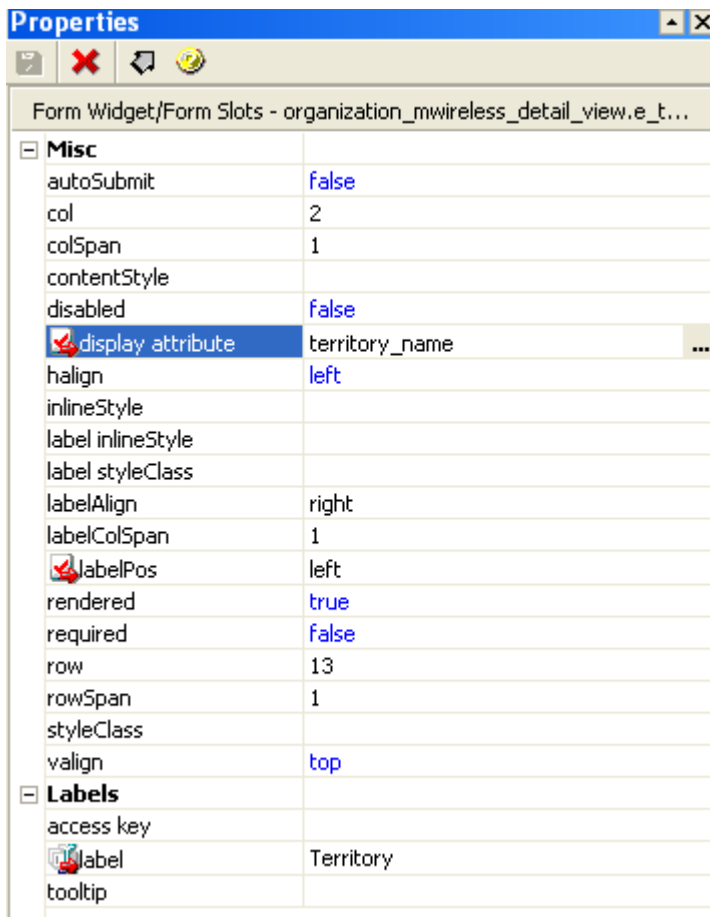
Territory 

Here are the steps to configure a tr_selectOneList widget:

- 1 Assign the RELATION attribute to the widget.

organization_mwireless_ui	tr_selectOneList-form_widget	e_territory (customer)	e_territory
---------------------------	------------------------------	------------------------	-------------

- 2 In the Properties window, set the “display attribute” property to the name of the field in the secondary BIO that should be displayed in the user interface.



Form Widget/Form Slots - organization_mwireless_detail_view.e_t...	
Misc	
autoSubmit	false
col	2
colSpan	1
contentType	
disabled	false
display attribute	territory_name
halign	left
inlineStyle	
label inlineStyle	
label styleClass	
labelAlign	right
labelColSpan	1
labelPos	left
rendered	true
required	false
row	13
rowSpan	1
styleClass	
valign	top
Labels	
access key	
label	Territory
tooltip	

- 3 Configure the Extensions. The tr_selectOneList widget needs two extensions:

- a The QueryAction extension is triggered by the clickEvent. This extension returns an appropriate list of BIOs. It takes two parameters:
 - perspective: typically set to “trinidad list view”
 - type: set to the secondary BIO’s BIO Type.

Event/Extension Mapping [2 Records]

Module*	Event Category	Event	Extension*	EpiExtension	Disabled?	Priority
organization_mwireless_ui		clickEvent (e_territory)	QueryAction		<input type="checkbox"/>	100

Properties

Event/Extension Mapping - QueryAction

Misc	
create new qbe	false
orderByClause	
perspective	trinidad list view
query_string	
show empty list	false
single item navigation name	
single item perspective	
type	e_territory
use_wildcards	
Labels	
title	

- b The JSFBackFillActionExtension is triggered by the returnFromBack event. This extension ensures that the tr_selectOneList widget displays the appropriate value. It takes two parameters:
- return attribute: typically leave this blank
 - target attribute: set this to the RELATION attribute.

Module*	Event Category	Event	Extension*	EpiExtension	Disabled?	Priority
organization_mwireless_ui		returnFromBackEvent (e_territory)	JSFBackFillActionExtension		<input type="checkbox"/>	100

Properties

Event/Extension Mapping - JSFBackFillActionExtension

Misc	
return attribute	
target attribute	e_territory
Labels	
title	

Please refer to "Events and Extensions" on page 53 which describes this extension in detail.

4 Configure the Navigation for the clickEvent.

- Set the Navigation Type to "Screen".
- We want to display a list form showing the secondary BIO records. In this example, this will be the Territory list form where the "Content" configuration of the Navigation Definition refers to it.

Extensions	Navigation	Actions	User Permissions	Application Permissions	Labels	Required Features	Validation Rules	Tab Identifiers
Extension References								
Navigation/Navigation Definitions [1 Records]								
Module*	Event Type*	Navigation Type*	Form Placement	Navigation Name*	List Order			
organization_mwireless_ui	clickEvent	SCREEN		territory	1			
* customer_base								
Navigation Def								
Module*	Container	BIO Source*	BIO Path	Content	Perspective	Spec		
organization_mwireless...	MainSlot	From Extension		e_territory - trinidad list view	trinidad list view			
* organization_mwireless...		From Extension						

- c In the Properties window that appear when you click on the Navigation Definition, set the “rowSelection” property to “single” and save.

Navigation Def - MainSlot - From Extension - e_te...

Navigation Definition Properties

Misc

isTabSelected true

Form Properties

Misc

columns 1

enableFilter true

inlineStyle

readOnly true

rendered true

rowBandingInterval 0

rows 5

rowSelection single

sorting Ascending First

styleClass

view detail false

The tr_selectOneList widget can also be used as a search widget in the Advanced Search area of the list form. For example, the Product Bundle tr_selectOneList widget on the Opportunity list form.

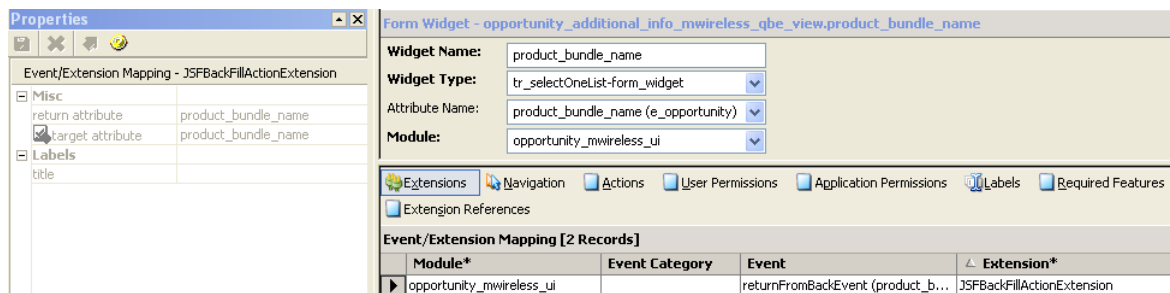
[More Info](#)

Product Bundle 

The steps are almost identical to the step outlined above except:

- 1 Instead of assigning the attribute to a RELATION attribute, we assign a PATH mapped attribute to the secondary BIO.

- 2 The extension parameters for JSFBackFillActionExtension are configured differently.
 - a return attribute: set this to the BIO attribute of the secondary BIO where you want to set as value to the target attribute.
 - b target attribute: set this to the PATH mapped attribute.



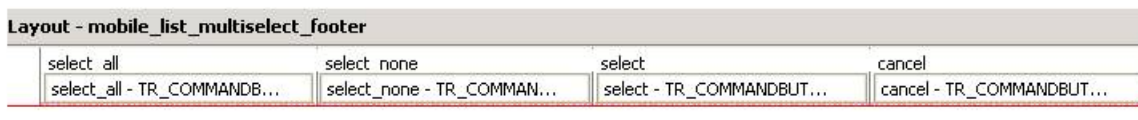
How to configure tr_panelHorizontalLayout to layout widgets

The tr_panelHorizontalLayout widget uses the Trinidad tr:panelHorizontalLayout UI component to layout multiple widgets horizontally.

This widget is typically used to bind a set of related widgets that can be laid out in a single grid column in the Studio layout area.

We can illustrate the usage of this widget by looking at two scenarios where the set of tr_commandButtons in the mwireless_list_multiselect_footer form can be laid out.

- 1 Laying out the widgets without using the tr_panelHorizontalLayout.



When the widgets are laid out individually using a 1 by 4 grid, the generated JSPX results in the four buttons spaced out evenly in the user interface. You can imagine there are 4 columns of equal width and the buttons are aligned to the left of each column.



- 2 Laying out the widgets using the tr_panelHorizontalLayout.



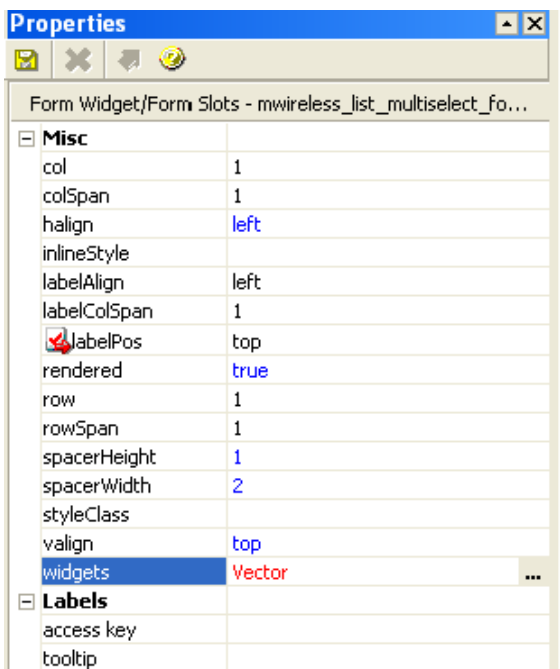
When we use the `tr_panelHorizontalLayout` widget to hold references to the set of the `tr_commandButton` widgets, and laid it out using a 1 by 1 grid, the generated JSPX results in the four buttons docked right next to each other.



Note: You do not have to lay out the widgets contained in the `tr_panelHorizontalLayout` in the layout area. Only the `tr_panelHorizontalLayout` widget should be there.

If this is the desired layout you like to see in the user interface, you should use `tr_panelHorizontalLayout` widget.

You add the widgets to the `tr_panelHorizontalLayout` using the “widgets” property.



When you click on the “widgets” property, it pops up a window for you to add the related widgets. The List Order determines the order in which the widget appears horizontally.

Vector Values (Custom Filter) [2 Records]				
	Module*	Object Value	Object Val	List Order*
▼				
▶	cial_forms ▼	mwireless_list_multiselect_footer.select	form_widget	1
	mwireless_...	mwireless_list_multiselect_footer.cancel	form_widget	2
*	customer_...		form_widget	

The `tr_panelHorizontalLayout` widget has two additional properties to customize the layout of the containing widgets:

- `spacerHeight`: The height of the spacer item in pixels.
- `spacerWidth`: The width of the spacer item in pixels. This is also the horizontal spacing between the containing widgets.

How to configure Partial Page Rendering (PPR)

Trinidad UI components have built-in PPR support. PPR refers to a technique to refresh a part of a page without having to entirely reload it.

Since Mobile Wireless widget types use the Trinidad UI components, they are PPR-enabled too. The widget properties needed to support PPR are:

- `partialSubmit` or `autoSubmit`: These properties must be set to “true” for the widget to become a PPR trigger
- `partialTriggers`: This property refers to those widgets with `partialSubmit` or `autoSubmit` set to “true” that are triggers for this widget to be affected by a PPR action.

The dependent drop down configuration explained earlier is an example of using PPR for rendering dependent drop down values.

Another simple example of PPR usage is using the `partialSubmit` property on the `tr_commandLink` and `tr_commandButton` widget.

When set to “true”, the `clickEvent` delivered by these widgets will happen by an AJAX event.

This will send a notification to the server, but will not change any content in the page. To do that, you have to tell Trinidad what needs to be redrawn in response to this. The easiest way to do this is with the `partialTriggers` property, which connects widgets that must be repainted with the widget that is the reason for them to repaint.

Shown below is a sample JSPX snippet of the generated Trinidad tags representing the `tr_commandButton` and the `tr_outputText` widget.

```
<tr:commandButton text="Do Something"
                  id="myButton"
```

```
        partialSubmit="true"
        actionListener="#{BEAN_XXX.processClick}"/>
<-- repaint the outputText any time 'myButton' has an event -->
    <tr:outputText id="computedRevenue" value="#{ BEAN_XXX.bio.computed_
revenue}"
        partialTriggers="myButton"/>
        partialTriggers="myButton"/>
```

In the above snippet, we want to refresh the “computedRevenue” widget value when the “myButton” widget is clicked without doing a full page reload.

The “computedRevenue” widget value is bound to the BIO attribute “computed_revenue” where the value can be updated by an extension that is hooked up to the clickEvent of the “myButton” widget. The “computedRevenue” widget has the partialTriggers configured to reference the “myButton” widget.

It is possible to have a widget that is affected by multiple triggers. To enable this, just configure the “partialTriggers” property to add reference to the trigger widgets.

This chapter describes how to use the different form types to design forms for Mobile Wireless.

When creating a new Form for Mobile Wireless, the first step is to identify the Form type to be used to contain the widgets. The Form type to pick is based on the data binding requirements for your form.

Every form defined in Studio will generate a JSPX file with a filename that matches the form name with a “.jspx” extension. Inside, the file will contain Trinidad tags for the widgets and formatting to render the form. You should not edit the generated jspx files directly. If you need to make a change, then change them in Studio and regenerate using “MakeJSF”.

Only Mobile Wireless widget types are allowed on forms using Mobile Wireless form types.

The sections that follow describe each form type and when it should be used.

NORMAL_TRINIDAD

The NORMAL_TRINIDAD form type is typically used to render a view that display a BIO and related BIOs such as a detail form of a BIO type. It is usually associated with the ‘trinidad detail view’ perspective.

Note: The NORMAL_TRINIDAD form type can also be used when there is no data binding needed. For example, the main screen after logging in is implemented using a NORMAL_TRINIDAD form type.

The NORMAL_TRINIDAD form type includes the following properties that can be specified in the Properties area of the Studio application window.

Properties that are marked with an asterisk (*) apply to all widgets that belong to this type of form:

Property Name	Description
columns	The number of columns in the form. This value is automatically set by the Studio form layout tool.
rendered	When set to false, no output will be delivered for widgets in this form (the component will not in any way be rendered, and cannot be made visible on the client).

Property Name	Description
styleClass	The cascading style sheet (CSS) class assigned to this form.
inlineStyle	The cascading style sheet (CSS) style assigned to this form.
*col StyleClass	The styleClass to be assigned to the column containing the form widget.
*labelCol StyleClass	The styleClass to be assigned to the column containing the label of the form widget.
col*	The column in which this form widget is displayed, where 1 is the leftmost column of the form. This value is automatically set by the Studio form layout tool.
colSpan*	The number of columns in which this form widget is displayed. This property is automatically set by the Studio form layout tool.
row*	The row in which this form widget is displayed, where 1 is the topmost row of the form. This property is automatically set by the Studio form layout tool.
rowSpan*	The number of rows in which this form widget is displayed. This property is automatically set by the Studio form layout tool.
labelPos*	Specifies whether the form widget label is positioned to the left, right, on top, or if it should be hidden. This property is automatically set by the Studio form layout tool.
labelColSpan*	If labelPos is Left, the number of columns used by the label. This property is automatically set by the Studio form layout tool.
labelAlign	If labelPos is Left, the horizontal alignment of the label (Left, Center, or Right). This property is automatically set by Studio.
halign*	The horizontal alignment of the grid row elements. The acceptable values are "center", "left", "right", "start", and "end".
valign*	The vertical alignment of the grid row elements. The acceptable values are "middle", "top", and "bottom".
existing record caption	The caption that appears as part of the contextual caption in the header for an existing record.

Property Name	Description
new record caption	The caption that appears as part of the contextual caption in the header for a new record.

The initial layout of the NORMAL_TRINIDAD form type is similar to the NORMAL form type that uses a tabular layout with rows and columns. There are two primary areas:

The diagram illustrates the layout of the NORMAL_TRINIDAD form type, divided into two main sections:

- Form Area (Top):** A vertical stack of six slots, each with a diagonal hatching pattern. The slots are labeled from top to bottom:
 - name_slot - SLOT
 - basic_info_slot - SLOT
 - organization_info_slot - SLOT
 - telephone_slot - SLOT
 - more_info_slot - SLOT
 - related links
 Below these slots is a horizontal bar labeled "related_links - TR_PANELHORIZONTALAYOUT".
- Gray Area (Bottom):** A large, solid gray rectangular area. At the top of this area, there are three small, light gray rectangular boxes containing the text:
 - related_addresses - T...
 - related_interactions - ...
 - related_tasks - TR_C...

- 1 Form area. Widgets positioned in this area will appear in the user interface and will appear in the position they occupy in the form area.
- 2 Gray area. Widgets positioned in this area will appear in the user interface.

The example below shows the Contact detail form.

Name

Salutation

First Name

Middle Name

Last Name

Generation

Honorific

Gender

▶ Basic Info

▶ Organization Info

▶ Phone

▶ More Info

Addresses Interactions Tasks

LIST_TRINIDAD

The LIST_TRINIDAD form type is used to display a list of BIOs of the same type in a table. It will contain a model binding object “table” that holds a BioCollectionBean set as the focus of the form. It is usually associated with the ‘trinidad list view’ perspective.

The LIST_TRINIDAD form type includes the following properties that can be specified in the Properties area of the Studio application window. Properties that are marked with an asterisk (*) apply to all widgets that belong to this type of form:

Property Name	Description
columns	The number of columns in the form. This value is automatically set by the Studio form layout tool.
rows	The maximum number of rows to display in the list window. To display all rows at once, set this attribute to 0.
rowSelection	Valid Values: none, single, multiple

Property Name	Description
rowBandingInterval	The interval between which the row banding alternates. For example, rowBandingInterval=1 would display alternately banded rows in the Grid.
rendered	When set to false, no output will be delivered for widgets in this form (the component will not in any way be rendered, and cannot be made visible on the client).
styleClass	The cascading style sheet (CSS) class assigned to this form.
inlineStyle	The cascading style sheet (CSS) inline style assigned to this form.
enableFilter	When set to true, the form is enabled for filtering.
view detail	When set to true, a view detail icon will be rendered in the row to allow viewing the detail of the selected row
sorting	By default, the sorting will be done in ascending order first. Change the value to No Sort if you do not want sortable lists. Change the value to Descending first if you want the sorting to be done in descending order the first time.
listCol*	The column in which this form widget is displayed, where 1 is the leftmost column of the form. This value is automatically set by the Studio form layout tool.
width*	The preferred width of this column, e.g., "30%", "100px".
noWrap*	Whether or not the column contents should be allowed to wrap.
sortable*	Whether or not the column is sort-able. A sort-able column has a clickable header that (when clicked) sorts the table by that column's property. Note that in order for a column to be sort-able, this attribute must be set to "true" and the underlying model must support sorting by this column's property.
align*	The alignment for this column. The legal values are "start", "end" and "center", for left-justified, right-justified, and center-justified respectively.
col*	Applies to the widget in More Fields filter area. The column in which this form widget is displayed, where 1 is the leftmost column of the form. This

Property Name	Description
	value is automatically set by the Studio form layout tool.
colSpan*	Applies to the widget in More Fields filter area. The number of columns in which this form widget is displayed. This property is automatically set by the Studio form layout tool.
row*	Applies to the widget in More Fields filter area. The row in which this form widget is displayed, where 1 is the topmost row of the form. This property is automatically set by the Studio form layout tool.
rowSpan*	Applies to the widget in More Fields filter area. The number of rows in which this form is displayed. This property is automatically set by the Studio form layout tool.
labelPos*	Applies to the widget in More Fields filter area. Specifies whether the form widget label is positioned to the left, right, on top, or if it should be hidden. This property is automatically set by the Studio form layout tool.
labelColSpan*	Applies to the widget in More Fields filter area. If labelPos is Left, the number of columns used by the label. This property is automatically set by the Studio form layout tool.
labelAlign*	Applies to the widget in More Fields filter area. If labelPos is Left, the horizontal alignment of the label (Left, Center, or Right). This property is automatically set by Studio.
halign*	Applies to the widget in More Fields filter area. The horizontal alignment of the grid row elements. The acceptable values are "center", "left", "right", "start", and "end".
valign*	Applies to the widget in More Fields filter area. The vertical alignment of the grid row elements. The acceptable values are "middle", "top", and "bottom".
additional attributes*	The additional attributes to display in the list data column
searchFormWidget*	Holds reference to the search form widget if the widget type in the search more field /filter row is changed.
list caption	The caption that appears as part of the contextual caption in the header for this list form.

The initial layout of the LIST_TRINIDAD form type is similar to the LIST form type. There are three areas:

The diagram illustrates the initial layout of the LIST_TRINIDAD form type, divided into three main sections:

- Top Section:** A vertical stack of four slots, each labeled "SLOT":
 - basic_info_qbe_slot - SLOT
 - organization_info_qbe_slot - SLOT
 - phone_info_qbe_slot - SLOT
 - address_info_qbe_slot - SLOT
- Middle Section:** A horizontal row of four input fields:
 - primary_formatted_te...
 - parent_organization_...
 - active_status_lkp - TR...
 - job_title - TR_INPUTT...
- Bottom Section:** A section containing a "Filter Row" and two buttons:
 - Filter Row:** A table with two columns: "full_name" and "TR_INPUTTEX...".
 - Buttons:** "Show Widgets..." and "Hide Widgets..."

The More Fields Filter area (this area is discussed later)

- 1 The Gray area. Widgets in this area do not appear in the user interface.
- 2 The List area. Widgets in this area appear in the list table.

The example below shows the Contact list form:

A screenshot of a mobile application interface. At the top, there is a search bar with a magnifying glass icon. Below it, a list of items is shown. The first item is 'Mac Tab' with a pencil icon on the left and an upward arrow on the right. Below 'Mac Tab' are details: 'Phone: +1-555-1234567', 'Employer: Helping Hand INC.', 'Job Title: Supplies Manager', and 'Status: Active'. The next item is 'Scott Tiger' with a downward arrow on the right. The third item is 'TyCld Jr' with a downward arrow on the right. Below the list are three buttons: '+ New', 'Refresh', and 'Adv. Search'. Below these buttons is a section titled 'Advanced Search' with a downward arrow. Under 'Advanced Search' are two expandable sections: 'Basic Info' and 'Organization Info'. The 'Organization Info' section is expanded, showing two input fields: 'Organization' and 'Relationship'. Below these are two more expandable sections: 'Phone Info' and 'Address Info'.

Configure List Window Size

The Trinidad table has built-in features to handle windowing of a large amount of rows and list navigation controls to scroll through the window. Windowing is done automatically when the rows are greater than the row limit specified in the list form properties. The row limit is set using the “rows” property. By default, it is set to 5. When the number of records to display exceed the list window size, the list navigation controls automatically appear at the top of the list table.

Model List Data Columns showing Primary and Secondary Attributes

The List Data Column can be configured to display multiple information in a single table cell. The information can be categorized as Primary attribute and Secondary attributes.

Primary attribute refers to the first piece of information that appears in the list table cell and usually represents an important piece of information of the list data record. Secondary attributes are the supportive information that stacks right below the Primary attribute.

A list view in mobile wireless application has a fixed layout. The layout is designed for the most optimal display on the smartphones using a single column view. This means that you can only have one widget designated as a Primary attribute in the list area.

You do not need to explicitly assign a widget as Primary or Secondary attributes. The widget that appears in the list area represents the Primary attribute of the list data column. .

If the BIO attribute associated with the Primary attribute widget is marked as 'Queryable', a filter row will appear in the list table to allow filtering the list records base on this bio attribute.

Primary attribute widget can be assigned one or more Secondary attribute widgets by adding the widgets using the "additional attributes" property.

The "additional attributes" property is a Vector type. This allows you to add multiple widgets and assign a list order to the added widgets. The list order determines the order of appearance of the secondary attributes in the list data columns.

Below is an example showing the "additional attributes" configuration of the "full_name" primary attribute widget of the Contact list view.

Vector Values (Custom Filter) [2 Records]				
	Module*	Object Value	Object Val	List Order*
▼				
▶	ireless ui	individual_mwireless_list_view.primary_formatted_telephone_string	form_widget	1
	individual_...	individual_mwireless_list_view.parent_organization_name	form_widget	2
*	customer_...		form_widget	

By default, all secondary attribute widgets display their associated label in the list view. To hide the label associated with the secondary attribute widget, you need to set the labelPos property to 'hidden'.

Enable List Form for Single/Multiple selection

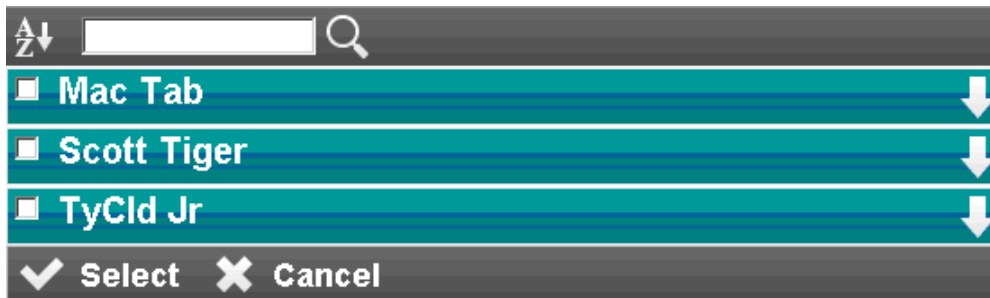
You can re-use the same list form in different usage contexts. To enable a list form to be used for single or multiple selections, you need to set the "rowSelection" property that appears in the Properties area of the Studio application window when you clicked on the **Navigation Definition**. The Properties window display the form properties associated with the Form Type of the Navigation Content.

Below is an example of the Studio screenshot showing the Navigation configuration associated with the Associate Toolbar button of the related Contacts of the Organization.

+ New **Associate** **Refresh** **Adv. Search**

The screenshot shows two panels from a development tool. The left panel is the 'Properties' window for a 'Navigation Def - MainSlot - From Extension - Individu...'. It lists various properties under 'Misc' and 'Form Properties'. The 'rowSelection' property is set to 'multiple'. The right panel is the 'Form Widget' configuration for 'organization_indiv_mwireless_list_view Toolbar_Trinidad.add_existing'. It shows fields for 'Widget Name' (add_existing), 'Widget Type' (tr_commandButton-form_widget), 'Attribute Name', and 'Module' (organization_mwireless_ui). Below these are tabs for 'Extensions', 'Navigation', 'Actions', 'User Permissions', 'Application Permissions', 'Labels', 'Required Features', 'Validation Rules', and 'Tab Identifiers'. The 'Navigation' tab is active, showing 'Navigation/Navigation Definitions [1 Records]' and a table with columns: Module*, Event Type*, Navigation Type*, Form Placement, Navigation Name*, and List Order. The table has one record for 'organization_mwireless_ui' with 'clickEvent' as the event type and 'SCREEN' as the navigation type. Below this is a 'Navigation Def' table with columns: Module*, Container, BIO Source*, BIO Path, Content, and Perspective. It has two records: one for 'organization_mwireless...' with 'MainSlot' as the container and 'individual - trinidad list view' as the content, and another for 'organization_mwireless...' with 'From Extension' as the source and 'trinidad list view' as the content.

The “rowSelection” property is set to “multiple”. When the Associate Toolbar button is clicked, it displays the Contacts list view with check boxes in the list table to allow multiple selections.



For multiple selection list view, you will need to bind the extension JSFSetupRelBetweenBiosAction. This is in addition to the QueryAction extension that you typically need to bind to the clickEvent associated with a Navigation that navigates to a list view. More detail on this extension is available in "Events and Extensions" on page 53.

Enable List Form Sorting

The list form can be sorted by the Primary attribute. You can enable sorting on a list form by setting the 'sortable' attribute of the widget assigned as the Primary attribute to 'true'. By default, the 'sortable' property is set as 'false'. The sorting order can be configured using the 'sorting' property of the LIST_TRINIDAD form type. By default, the initial sorting order is 'Ascending First'.

More Fields Filter Area

The More Fields Filter Area allows you to add widgets or form slots to the list view to support filtering the list records based on other attributes in addition to the filter row that appear in the list table.

Laying out widgets in the More Fields Filter is identical to layout widgets in a NORMAL_TRINIDAD form type. You can right-click the More Fields Filter to get popup menus that let you add or remove rows and columns.

Below is an example of the More Fields Filter area seen in the user interface where we have used form slots to contain the search widgets. See the section Form Slot at the end of this chapter for more information on how to use Form Slots.

The screenshot shows a mobile form titled "Advanced Search". It features a "Basic Info" section with several input fields: "Status" (a dropdown menu), "Job Title", "Employer", "E-mail", "Assistant Name", "Manager Name", and "Contact Since". Below these fields is a label "Date/Time Format: e.g. 13-10-10". At the bottom of the form, there are three expandable sections: "Organization Info", "Phone Info", and "Address Info", each indicated by a right-pointing arrow.

TOOLBAR_TRINIDAD

The TOOLBAR_TRINIDAD form type is used to contain a set of widgets that typically perform common tasks related to the focus of the form.

When you create a form where the form type is NORMAL_TRINIDAD or LIST_TRINIDAD, Studio automatically generates a toolbar_trinidad-form slot and a Toolbar form based on the TOOLBAR_TRINIDAD form type. The Toolbar forms are always named <name of parent form> Toolbar_Trinidad.

The generated toolbar_trinidad-form slot always appears in the "toolbar" area at the top of the Layout view in Studio. Double-clicking the toolbar area evokes the layout for the Toolbar.

Layout - organization_mwireless_detail_view	
organization_mwireless_detail_view Toolbar_Trinidad - TOOLBAR	
Name	full_name - TR_INPUTTEXT
Status	active_status_lkp - TR_SELECTONECHOICE

Toolbar Layout - organization_mwireless_detail_view Toolbar_Trinidad		
save	refresh	delete
save - TR_COMMANDBUTTON	refresh - TR_COMMANDBUTTON	delete - TR_COMMANDBUTTON

Form Widget/Form Slots [31 Records]				
Module*	Widget Type*	Attribute Name	Widget Name*	
organization_mwireless_ui	normal-form_slot		telephone_slot	
organization_mwireless_ui	toolbar_trinidad-form_slot		organization_mwireless_detail_view Toolbar...	
organization_mwireless_ui	tr_commandLink-form_widget		related_addresses	
organization_mwireless_ui	tr_commandLink-form_widget		related_contacts	

Slot Defs							
Name*	BIO Path	Form Placemen	Tab Identifier	Content	List Order	Tab Identifier List Or	Perspective
organization_mwireless_detail_view...					1		organization_mwireless_detail_view Toolbar_Trinidad
*							

The form slot definition associated with this form slot points to a perspective that is set to the toolbar's auto-generated perspective. The BIO path value is blank, which means that the BIO of the parent form will be used for the form slot contents as well.

The TOOLBAR_TRINIDAD form type includes the following properties that can be specified in the Properties area of the Studio application window.

Property Name	Description
columns	The number of columns in the form. This value is automatically set by the Studio form layout tool.
spacerWidth	The width of the spacer item in pixels. This is also the horizontal spacing between the widgets in the Toolbar.
spacerHeight	The height of the spacer item in pixels.
halign	This sets the horizontal positioning of the Toolbar relative to the form. The horizontal alignment to one of start, end, left, right, or center. Default is center.

Beside the properties described above that are form-specific, this form type has the same set of widget specific properties as the NORMAL_TRINIDAD form type.

Laying out widgets in the TOOLBAR_TRINIDAD form type is identical to layout widgets in a NORMAL_TRINIDAD form type.

LOGIN_TRINIDAD

The LOGIN_TRINIDAD form type is meant to handle the authentication of the user when no valid SSO token is available to authorize access to the BIO service.

It has the same set of properties similar to the NORMAL_TRINIDAD form type.

More detail on the usage of this form type is provided in Chapter 4 – Mobile Wireless Special Form Types.

FORM SLOT

A form slot is a placeholder used to embed one form inside of another form.

There are two supported form slot types that can be used on Mobile Wireless Form Types: toolbar_trinidad-form_slot and normal-form_slot.

toolbar_trinidad-form_slot is used to embed a Toolbar (which is technically a form base on TOOLBAR_TRINIDAD form type) in a form.

Normal-form_slot is used to display a single form. It can be placed in any location on the parent form.

In most cases, a form slot's form is based on a BIO other than the base BIO. A BIO path is an element of metadata associated to the form slot definition that defines the BIO(s) to put in the subform. It has the form <primary BIO>:<secondary BIO>.

Below is an example showing the Form Slot Definition configuration for the Primary Telephone phone slot in the Organization detail form.

▶ organization_mwireless_ui	normal-form_slot		telephone_slot
organization_mwireless_ui	tr_inputText-form_widget	web_address (organization)	web_address
organization_mwireless_ui	tr_goLink-form_widget	web_address (organization)	web_address_link
* mwireless_special_forms			

Slot Defs							
Module*	Name*	BIO Path	Form Placemen	Tab Identifier	Content	List Order	Ta
▶ organization_mwireless_ui	telephone_slot	organization:primary_telephone			telephone - trinidad detail view	1	
* mwireless_special_forms							

The normal-form_slot is represented as a Trinidad tr:showDetail UI component in the generated JSPX. The tr:showDetail component provides a means of toggling a group of widgets between being disclosed or undisclosed.

Below is an example showing the Primary Telephone phone slot in the Organization detail form.

Home > Organizations > **Organization:** Helping Hand INC.

▶ Basic Info

▼ Phone

Country Code

Area Code

☐ Number

Extension

☐ Type

Dial It!

▶ More Info

The disclosed/undisclosed text beside the expand/collapse icon can be customize by changing the “title” property that is available when you click on the Form Slot Definition.

There are three special forms that exist in the Mobile Wireless Application.

- Login Form
- Home Screen Form
- Header Form

These special forms provide a framework to handle login, access to specific entities in Sales and Service, and global navigation features.

Customizations can be made using Studio. Any form changes should be saved as delta modules to the existing 'mwireless_special_forms' module.

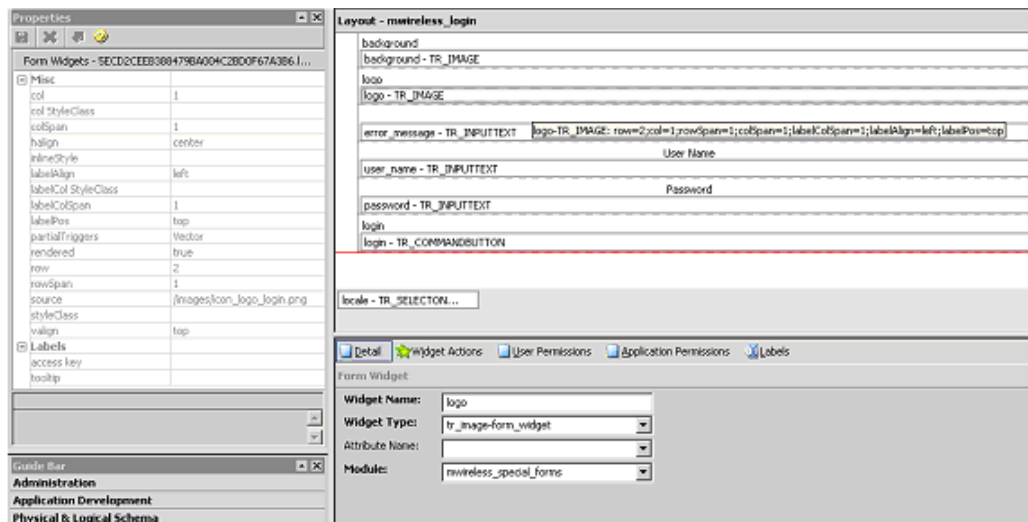
Several CSS styles specific to the special forms are provided to allow further customization of the look and feel, specific to each smartphone. See chapter 8 of this guide for details on CSS customization.

Details for configuring Extensions and creating Navigation Definitions in Mobile Wireless are covered in chapters 6 and 7 of this guide.

Mobile Wireless Login Form

The login form is defined in Studio as 'mwireless_login' in the module 'mwireless_special_forms'. URL requests that match the context-root of the Mobile Wireless application will direct the user to this login page.

The layout of the form can be customized using Studio.



Mobile Wireless login page layout



Mobile Wireless Login Page

Additional logic can be executed during the login process by attaching extensions to the clickEvent of the 'Login' button, or at the form level.

Form Widget - mwireless_login.login

Widget Name: login

Widget Type: tr_commandButton-form_widget

Attribute Name:

Module: mwireless_special_forms

Extensions Navigation Actions User Permissions Application Permissions Labels Required Features

Validation Rules Tab Identifiers Extension References

Event/Extension Mapping [1 Records]

Module*	Event Category	Event	Extension*	EpiExtensi	Disabled?	Priority	Epi
mwireless_special_forms		clickEvent (login)	JSLoginExtension		<input type="checkbox"/>	100	Epi
*					<input type="checkbox"/>	100	Epi

Login Widget Extensions

Upon successful validation of credentials, the user is navigated to the Mobile Wireless Home Page Form. If a logged in user has been idle for too long (30 minutes by default), the next UI action that triggers a navigation will redirect the user to the Login Form. Additional details on the Login process extensions are discussed in chapter 6 of this document.

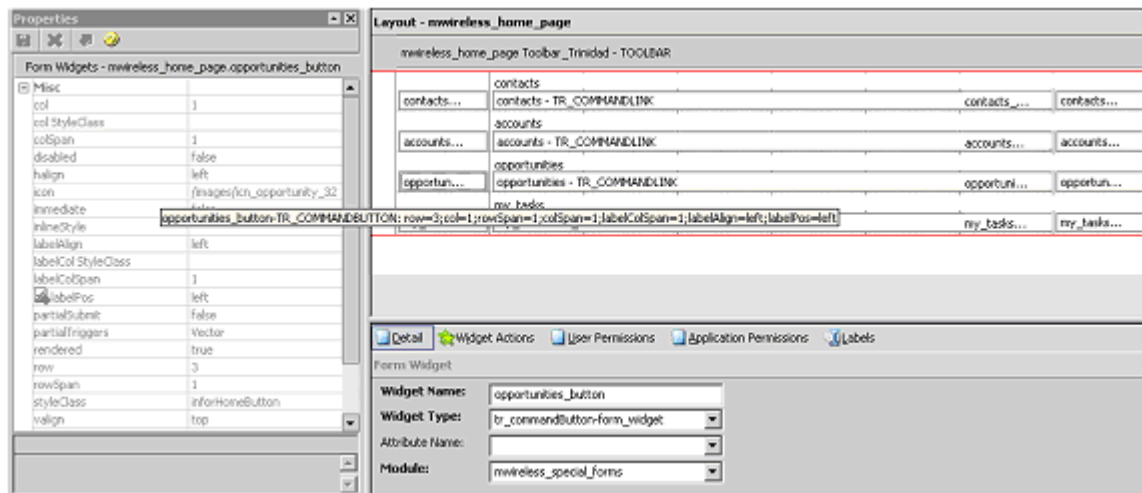
Note: New users must first 'self-register' by logging into the desktop Sales or Service application before they can login to the Mobile Wireless application.

Mobile Wireless Home Page Form

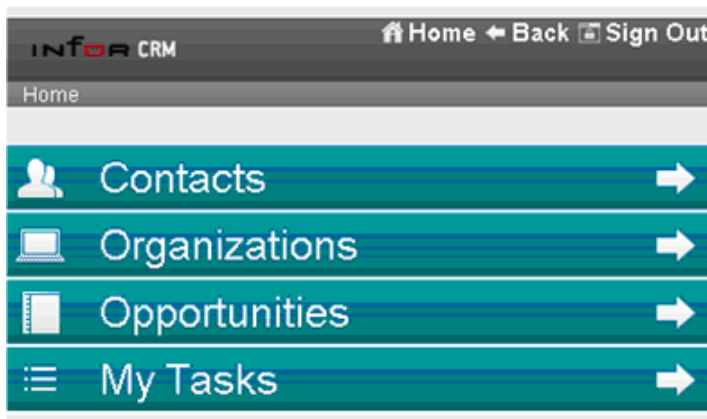
The Home Page form is defined in the 'mwireless_special_forms' module and is named 'mwireless_home_page'. The form type is NORMAL_TRINIDAD.

The Home Page Form is the first screen the user will see after login, and serves as a root location from which links and navigations to other forms in the application are defined.

The layout of the Home Page can be customized using Studio. Out of the Box, the Home page has links for four entities: Contacts, Organizations, Opportunities, and My Tasks. Each is represented by a row with three clickable links: two button widgets and a text hyperlink. CSS styles are applied to customize the look of the button widgets and the cell background.



Mobile Wireless Home Page layout



The Mobile Wireless Home Page

When custom LIST_TRINIDAD and NORMAL_TRINIDAD forms are added to the application, an implementer can add Trinidad button or link widgets, to provide a way to view the new forms. As with the any new form, the appropriate extensions and navigations must be attached to the widget to fully integrate the new forms.

Example:

If the LIST_TRINIDAD form called 'e_lead_list' was created in Studio, it could be added to the application by following these steps:

- 1 Add a tr_commandLink or tr_commandButton to the Home Page form.
- 2 Double click the new widget to view detail.
- 3 Add an extension with Event 'clickEvent' and Extension 'QueryAction' and save.
- 4 In the extension's property sheet, set the perspective to 'trinidad_list_view'.
- 5 For type, select the BIO this list view represents (in this case, the e_lead bio type) and save.
- 6 Select the Navigation tab and make a navigation with Event Type of 'clickEvent' and Navigation Type of 'SCREEN'. Give it a name and save.

- 7 Under the Navigation Def below, select Container of 'Main Slot' and BIO source 'From Extension'. For content, choose the trinidad list form you wish to display (in this case, e_lead – trinidad list view).

Mobile Wireless Header Form

The Header is a special form that is designed to provide a set of common functionality for all pages within the Mobile Wireless application. It is included within all application forms except 'mwireless_login'. The Header Form is named 'mwireless_form_header' and has a form type of NORMAL_TRINIDAD.

Header customizations can be made using Studio. The Header provides space for a logo, navigation links, error message display, a contextual caption, and a form caption.

The Mobile Wireless Header Page as seen from the Task Detail view Page. The logo, links, caption and error message are part of the header. The Task Detail Page begins with the toolbar.

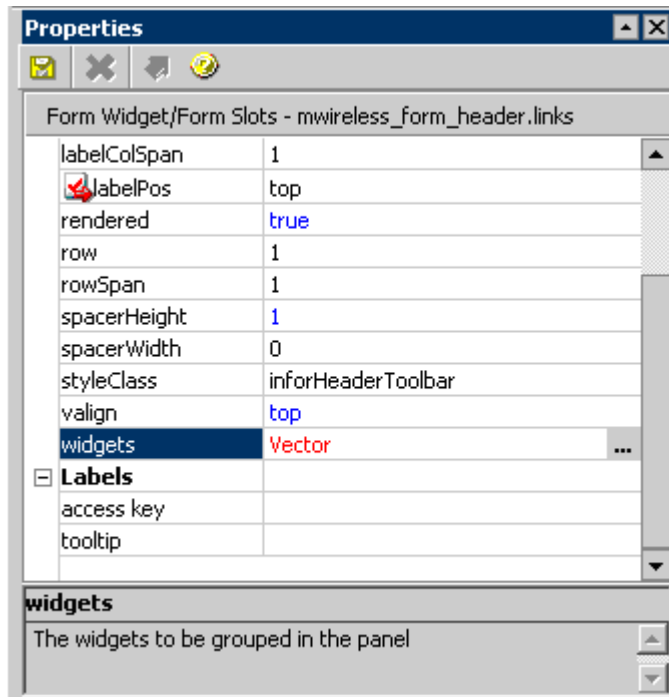
Replace the logo by updating the 'source' property of the 'logo' widget to reference your image file.

The Navigation Links displayed in the upper right corner of the header are defined individually by the 'home', 'back' and 'sign_out' widgets. They are grouped into the 'links' container widget, which is added to the form layout. The Navigation links may be cosmetically altered, or extended to include additional

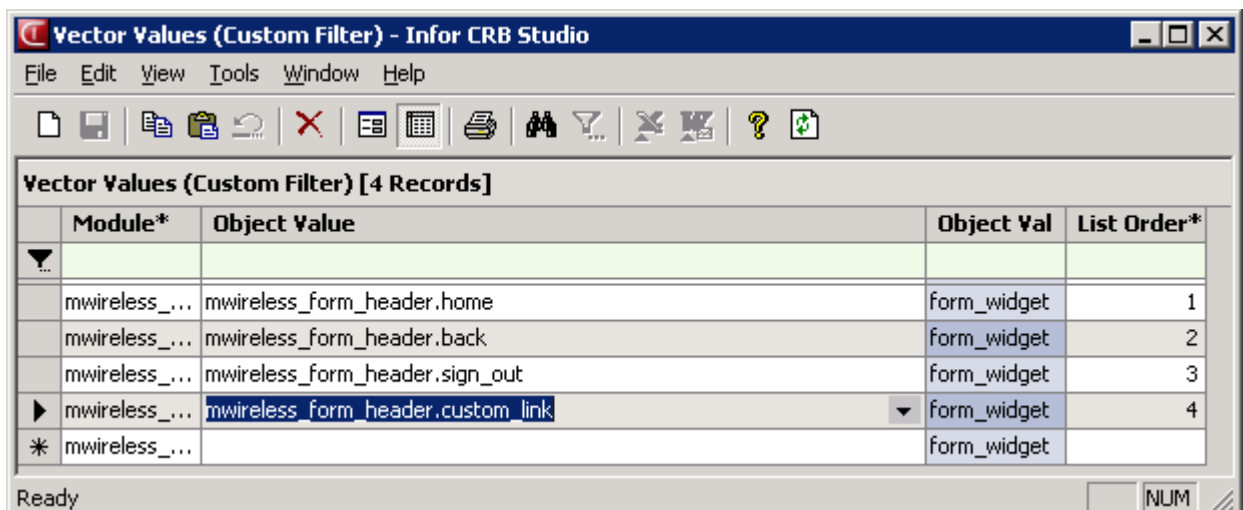
logic. Implementors should not modify the existing extensions and navigations for these links, or remove them from the header.

To add a new item to the Navigation Links, create a 'commandLink' widget with appropriate extension and navigation, then add it to the 'links' property vector called 'widgets'.

Note: Additional links may require modification of the form layout due to the limited horizontal display area on smartphones.



'links' Widget Property Sheet



'links' Widget Property Vector Values

The 'error_message' widget is provided as a place for a form to display general error messages. Do not modify the functionality of this widget.

The 'caption' widget provides a representation of the context of the page you are currently viewing. The content for this widget is generated by the extension JSFContextualCaptionExtension. Information on how to customize the display of the caption widget using parameters for the extension, can be found in chapter 6 of this guide. Implementers may choose to write their own implementation of a contextual caption extension to replace the out of box one, if desired.

Properties

Event/Extension Mapping - JSFContextualCaptionExtension

Misc

delimiter

>

max length

80

max levels

3

trim ellipsis

..

trim num visible chars

2

Labels

title

delimiter

The delimiter character between levels of history

Form Widget - mwireless_form_header.caption

Widget Name:caption

Widget Type:tr_outputText-form_widget

Attribute Name:

Module:mwireless_special_forms

Extensions

Navigation

Actions

User Permissions

Application Permissions

Labels

Required Features

Validation Rules

Tab Identifiers

Extension References

Event/Extension Mapping [1 Records]

Module*	Event Category	Event	Extension*	EpiExtension	Disabled
mwireless_special_forms		preRenderWidget (caption)	JSFContextualCaptionExtension		<input type="checkbox"/>
*mwireless_special_forms					<input type="checkbox"/>

Configurable Properties for the Caption Widget Extension

The 'form caption' widget is provided as a placeholder for displaying caption of the form. The value of the widget is bound to an EL expression "#{state.formCaption}". This means that you can invoke the setFormCaption() API of the state instance to dynamically set the form caption. Out of the Box, the sales_stage_step_mwireless_list_view uses a pre-render form extension 'SetSalesStepListFormCaption' to set the caption of the form to the Stage name associated with the Tasks.

As in the Sales & Service desktop application, events are the triggers that occur when a user interacts with the Mobile Wireless application. Based upon what the user does to interact with the UI, certain events will be fired that can be handled by extensions that are mapped as part of the application.

Events have a source (where did the event come from) and a target (the object to which the event will be delivered). The source of an event will typically be from the JSF framework dealing with the user interaction, but it can also come from other places such as the BIO layer during a `bioBeforeInsertEvent`. The target will be a UI runtime object such as a form or widget that represents the button being displayed.

For example: A user clicking on a button will fire a `clickEvent` targeted at a `JSFRuntimeWidget` instance for that particular button.

Extensions are Java code that act on an event being triggered. While you cannot add Events to the application, you can add behavioral logic by creating your own extensions and mapping them to one of the pre-defined events.

There are two types of extensions: core and application. The core extensions are not customizable, but you are free to map them to any supported event in your own forms or change the mappings in existing forms. Application extensions are provided to you out of the box with the source code. You are free to change them or extend them.

Mobile Wireless Extensions vs. Desktop Extensions

Mobile Wireless uses an alternate framework from the desktop when handling extensions. While it is possible to share extensions between the two applications, this is discouraged and should only be done when the common behavior is only data related (i.e. modifying BIO attributes).

You may notice that some core extensions are shared among Mobile Wireless and the Desktop application. For example: `QueryAction` is used to invoke a query and to populate a list view in both applications.

Note: Core Extensions that can only used for Mobile Wireless and cannot be used by the desktop will be prefixed with the “JSF” in the name.

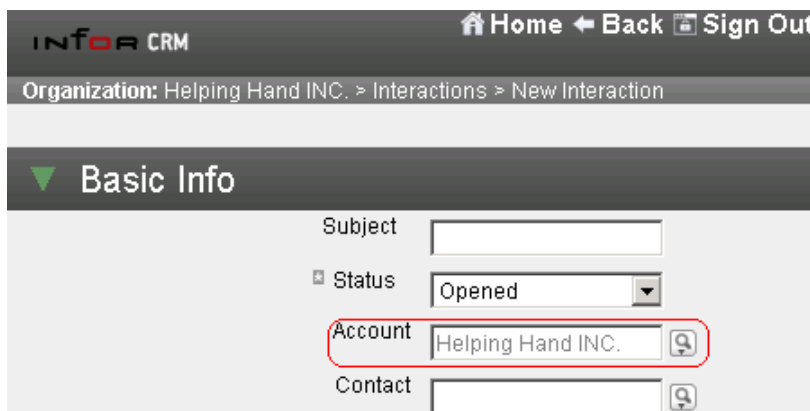
Form Events

preRenderFormOnInit

The preRender Form event applies to following form types: LOGIN_TRINIDAD, NORMAL_TRINIDAD, TOOLBAR_TRINIDAD. It is fired by the framework when the form has just completed rendering for the first time.

A typical use case for hooking an extension to this event is to initialize a widget data on the form. Your extension can look for a specific widget in your form and set an appropriate initialization value. Unlike a default value defined in metadata, the initialization values set using this approach marks the form dirty and will adhere to any validation rules, if you save the form.

Another common use case is to disable or hide a widget based on a business rule implemented inside the extension.

The screenshot shows the Infor CRM interface for a 'New Interaction' form. At the top, there's a navigation bar with 'Home', 'Back', and 'Sign Out' links. Below that, a breadcrumb trail reads 'Organization: Helping Hand INC. > Interactions > New Interaction'. The main section is titled 'Basic Info' with a dropdown arrow. It contains four input fields: 'Subject' (empty), 'Status' (a dropdown menu currently showing 'Opened'), 'Account' (a text field containing 'Helping Hand INC.' with a magnifying glass icon to its right), and 'Contact' (empty with a magnifying glass icon to its right). A red rectangular box highlights the 'Account' field and its associated magnifying glass icon.

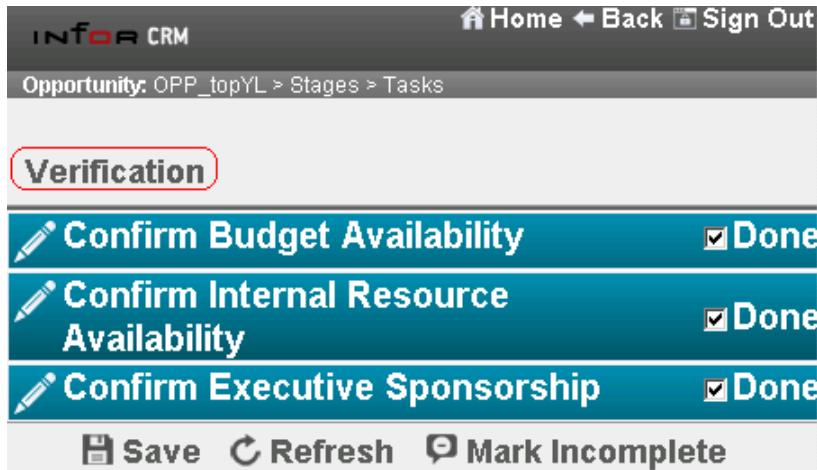
preRenderFormOnInit Sample Screen

preRenderFormOnRedisplay

The preRenderOnRedisplay is similar to preRenderOnInit except that it is fired instead of preRenderOnInit if the form is refreshed or is re-entered on a “back” navigation. You may want a different behavior when the user revisits the same form vs. the first time initialization, and in which case this is the event to map to differentiate that.

preRenderListOnInit

This event is identical to `preRenderOnInit` except that it applies only to `LIST_TRINIDAD` form types. Use this instead on list views.



preRenderListOnInit Sample Screen

preRenderListOnRedisplay

This event is identical to `preRenderOnRedisplay`, except that it applies only to `LIST_TRINIDAD` form types. Use this instead on list views.

Form Widget Events

changeEvent

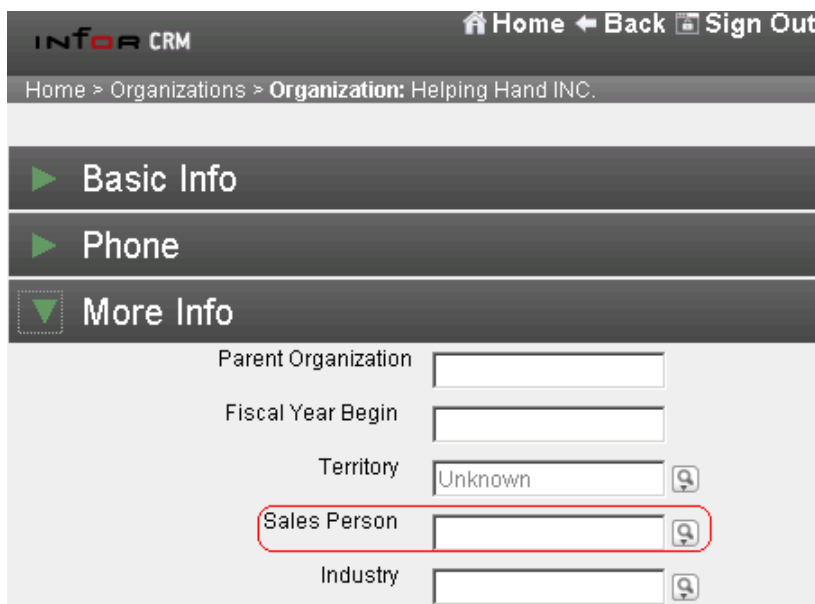
This event is fired whenever a value is submitted on a widget that is not equal to the current value. For most widgets that support the “autoSubmit” property value (such as `tr_selectOneList`), this will control whether this event is fired immediately a new value is selected in the dropdown or whether it will be submitted by other action widgets such as a button click.

Note: If a user enters the same value as the current value in a widget, it does not result in firing of this event.

Note: A ChangeEvent is fired at the ProcessValidations phase which occurs before the UpdateModelValues phase in the JSF lifecycle. That means that the model values from the BioBeans, accessed in the extensions, will not have the changed values. To get the changed values, you will need to get them directly from the component objects themselves.

clickEvent

The click event fires when the end user clicks on the widget. The widget must be of clickable type such as a tr_commandButton, tr_commandLink, or tr_selectOneList type.



The screenshot shows the Infor CRM interface. At the top, there's a navigation bar with 'Home', 'Back', and 'Sign Out' links. Below that, a breadcrumb trail reads 'Home > Organizations > Organization: Helping Hand INC.'. The main content area has three expandable sections: 'Basic Info', 'Phone', and 'More Info'. The 'More Info' section is expanded, revealing several input fields: 'Parent Organization', 'Fiscal Year Begin', 'Territory' (with a dropdown arrow and the value 'Unknown'), 'Sales Person' (highlighted with a red rectangle and a dropdown arrow), and 'Industry' (with a dropdown arrow). The 'Sales Person' field is the focus of the 'clickEvent' discussion.

clickEvent Sample Screen

preRenderWidget

This event is similar to preRenderForm in that it is fired to all widgets that are rendered as the target on the form. The preRenderWidget event will fire after the preRenderForm is fired. If you only need to perform an action during initialization of a widget, it is better to map your extension to this event.

The screenshot shows the Infor CRM interface for a 'New Interaction' form. The breadcrumb trail is 'Organization: Helping Hand INC. > Interactions > New Interaction'. The 'Basic Info' section is expanded, revealing several input fields. The 'Account' field, which contains the text 'Helping Hand INC.', is highlighted with a red rectangular border. Other visible fields include 'Subject', 'Status' (set to 'Opened'), and 'Contact'. The top navigation bar includes links for 'Home', 'Back', and 'Sign Out'.

preRenderWidget Sample Screen

returnFromBack

This event is used to invoke an extension after a “back” navigation has occurred. It is launched on the widget that initiated navigation to a child form. Once the child form is dismissed using a navigation name of “back”, the form returns to the parent form and this extension will have a chance to examine the focus of the child form.

This is analogous to processing a return value from a subroutine where the subroutine here is a child form such as a list form, where you select certain entries to be processed.

Note: The current focus for any extension hooked on this event will be from the child form, not the current form.

Note: This event is new to Mobile Wireless. It is not available to the desktop application.

This screenshot is identical to the one above, showing the Infor CRM 'New Interaction' form. The 'Basic Info' section is expanded, and the 'Account' field, containing 'Helping Hand INC.', is highlighted with a red rectangular border. The interface elements, including the breadcrumb trail and navigation links, are the same as in the previous image.

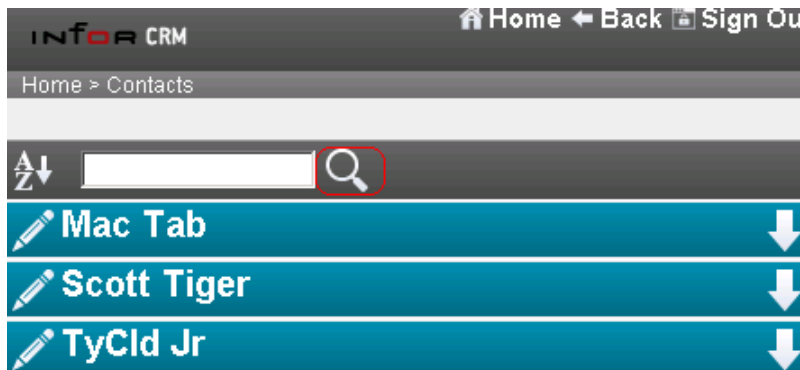
returnFromBack Sample Screen

List Form Events

A LIST_TRINIDAD form has built-in events that control the table of items being displayed on the screen. When you create a new LIST_TRINIDAD form, many of the events are mapped to core extensions automatically to control the table. You may wish to add functionality by writing your own extension and hooking them up to the same events, or you may want to replace the default behavior with your own extensions.

listFilterEvent

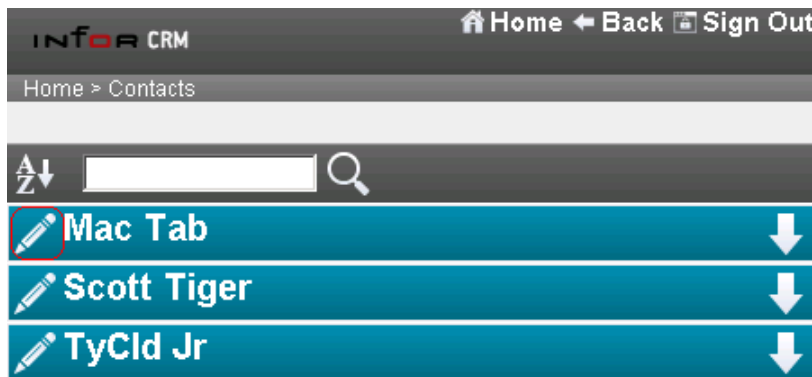
A list form contains a filter button on the header row allowing the user to submit filtering criteria entered on the filter row and on the “Advanced Search”. When the button is clicked, it triggers a listFilterEvent. The default mapping will map this event to a ListFilterPreQueryAction and QueryAction extensions.



listFilterEvent Sample Screen

listSelectEvent

A list form contains a column with an editing icon that allows the user to view the details for that row. When the user clicks on an editing icon, it triggers a listSelectEvent. The default mapping will map this event to a ListSelectAction extension.

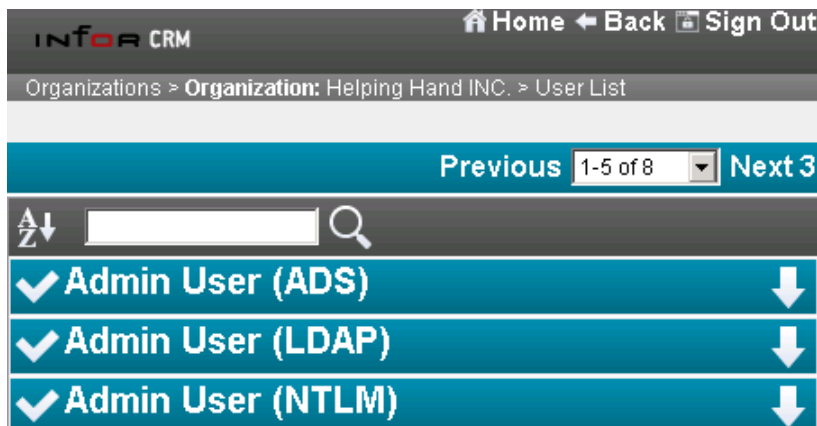


listSelectEvent Sample Screen

listSingleSelectEvent

A list form may be used to fulfill a `tr_selectOneList` widget, where the user needs to select one or more BIOs to relate to another BIO. When used this way, the property value “rowSelection” will determine if the selection is “single” or “multiple”. In the case of a single selection, the user will see that the list form will have a selection button on each row instead of the view details button. A click on the selection button will trigger this event to indicate that the user has chosen this event.

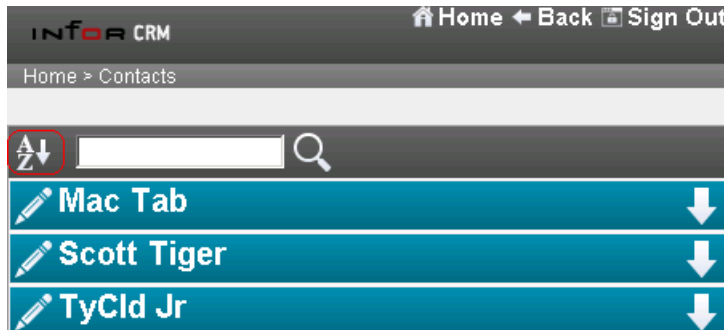
Note: The default mapping will map this event to a `ListSelectAction` extension.



listSingleSelectEvent Sample Screen

listSortEvent

A list form contains a header row with a sort icon. This sort icon is clickable and doing so will trigger a listSortEvent. The default mapping will map this event to a ListSortAction extension.

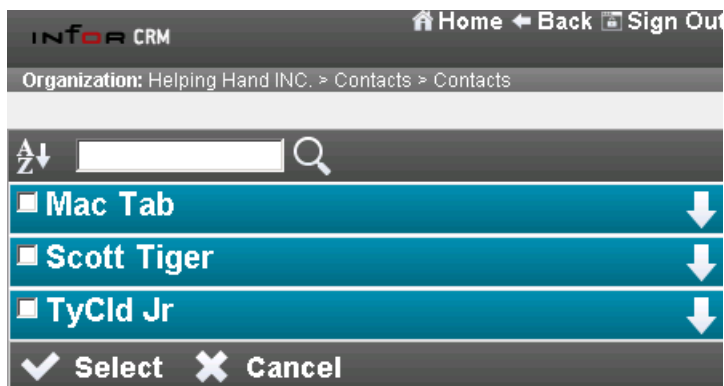


listSortEvent Sample Screen

List Multiple Selection (Not an Event)

In the case of a list file with the rowSelection property_value set to “multiple” selection, no event is raised when the user picks items from the list. Instead, the view detail column will contain a checkbox for every row allowing the user to select more than one entity to relate.

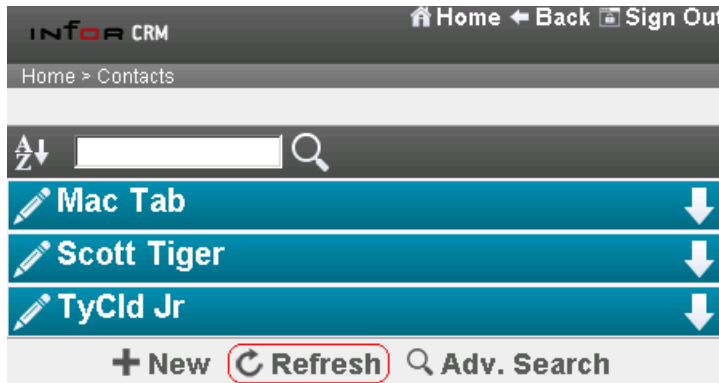
A footer toolbar will appear at the bottom of the list using the form “mwireless_list_multiselect_footer”. This form contains buttons to **Select** and **Cancel** that will respectively accept the items the user chooses in the multi-select, or just return to the previous form. It is in the **Select** button where the extension JSFCreateRelBetweenBiosAction will run and relate the items selected in the list with the relationship configured in the extension JSFSetupRelBetweenBiosAction.



List Multiple Selection Sample Screen

listRefreshEvent

Every list form has a “Refresh” button to refetch the list data from the data source. A click on this button will trigger this event. The default mapping will map this event to a JSFRefreshBiosAction extension.



listRefreshEvent Sample Screen

Form Core Extensions

Package: `com.infor.shr.ui.jsf.view.extensions`

JSFPrerenderFormSetValue

This extension allows you to specify a name/value pair for each instance of the extension mapping to explicitly set a value to a widget. It is useful for setting an initialization value but, unlike the “default value” property_value, it will mark the form as dirty.

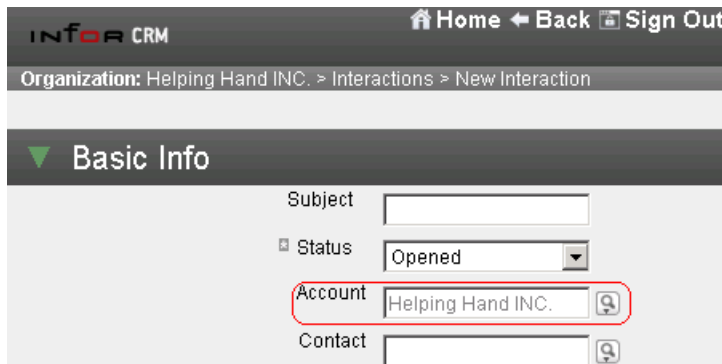
Form Widget Core Extensions

Package: `com.infor.shr.ui.jsf.view.extensions`

JSFDisableWidgetInNewMode

Use this extension if you want to disable a widget only when you are creating a new entity. This is good practice because it prevents the user from navigating away from the main entity to relate to other entities without first saving the main entity.

A disabled widget is visible to the end user, but the end-user cannot change the value.

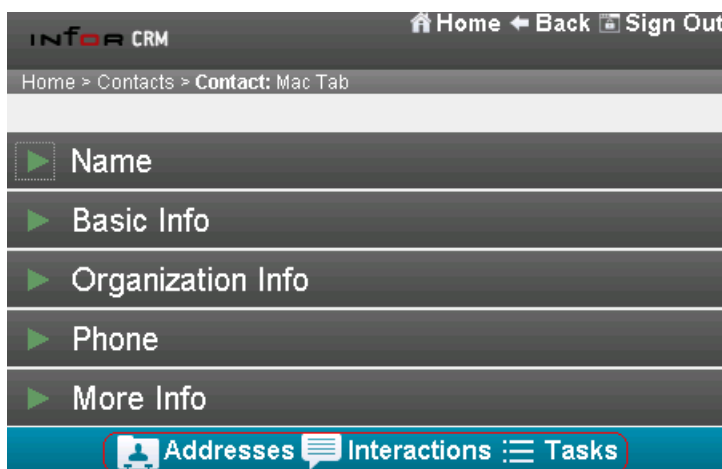


The screenshot shows the Infor CRM interface for creating a new interaction. The breadcrumb trail is 'Organization: Helping Hand INC. > Interactions > New Interaction'. Under the 'Basic Info' section, there are fields for 'Subject', 'Status' (set to 'Opened'), 'Account' (set to 'Helping Hand INC.'), and 'Contact'. The 'Account' field is highlighted with a red rectangle, indicating it is disabled.

JSFDisableWidgetInNewMode Sample Screen

JSFHideWidgetInNewMode

Similar to JSFDisableWidgetInNewMode, this extension will hide a widget when the user is creating a new entity in a detail view and it has not been saved yet.



The screenshot shows the Infor CRM interface for a contact detail view. The breadcrumb trail is 'Home > Contacts > Contact: Mac Tab'. On the left, there is a list of tabs: 'Name', 'Basic Info', 'Organization Info', 'Phone', and 'More Info'. At the bottom, there is a row of tabs: 'Addresses', 'Interactions', and 'Tasks'. The 'Addresses', 'Interactions', and 'Tasks' tabs are highlighted with a red rectangle, indicating they are hidden.

JSFHideWidgetInNewMode Sample Screen

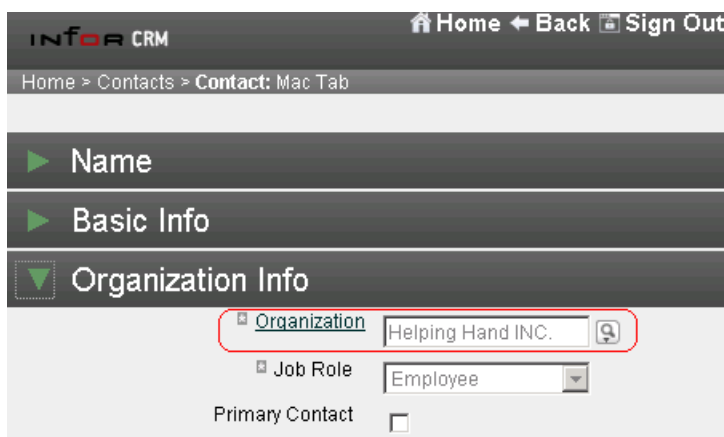
Action Based Core Extensions

Package: `com.infor.shr.ui.jsf.action.extensions`

JSFBackFillActionExtension

Property name	Description
Return attribute	Set this to the attribute name of the BIO selected in the list view as the source of the value to be saved. If you want to use the BIO selected itself (as a BioBean object), then do not set a value here.
Target attribute	Set this to the attribute name of the focus BIO to save the return value. At the end of the execution, you should expect that the result will be the invocation: <code>focus.set(target attribute, bioFromListSelected.get(return attribute))</code>

The backfill action extension is used in conjunction with a navigation using a `tr_selectOneList` widget. It is used to set an attribute value of the focus BIO from an attribute value of the BIO from the list that was selected. This is typically the case when trying to relate a 1-1 BIO. You need to map this extension to the “returnFromBack” event such that it will get executed after the user has selected the item to relate in the list view.



The screenshot shows the Infor CRM user interface. At the top, there's a navigation bar with 'Home', 'Back', and 'Sign Out' links. Below that, a breadcrumb trail reads 'Home > Contacts > Contact: Mac Tab'. The main content area has three expandable sections: 'Name', 'Basic Info', and 'Organization Info'. The 'Organization Info' section is expanded, showing a form with three fields: 'Organization' (text input with 'Helping Hand INC.' and a search icon), 'Job Role' (dropdown menu with 'Employee' selected), and 'Primary Contact' (checkbox, currently unchecked). The 'Organization' field is highlighted with a red rectangular box.

JSFBackFillActionExtension Sample Screen

JSFContextualCaptionExtension

Property name	Description
Delimiter	The character(s) that divides pages in the contextual caption hierarchy
Max length	A maximum value for the total length of the caption. Calculated values exceeding this value will be abbreviated according to 'trim ellipsis' and 'trim num visible chars' properties. Set this to 0 for no upper limit.
Max levels	The number of levels displayed for the caption. Example: If you navigated from Home to Organization List, Organization Detail, Address List, the caption would show: Organizations > Organization > MyOrg > Addresses
Trim ellipsis	The character(s) that will be substituted into the caption string if it exceeds the 'max length' parameter. Only applicable if the caption string length exceeds the value set for 'max length'.
Trim num visible chars	An integer value defining the number of characters displayed before and after the 'trim ellipsis' string when the caption is abbreviated. Only applicable if the caption string length exceeds the value set for 'max length'.

The contextual caption widget provides the user with information about which page within the Mobile Wireless application they are currently viewing, and in what context.

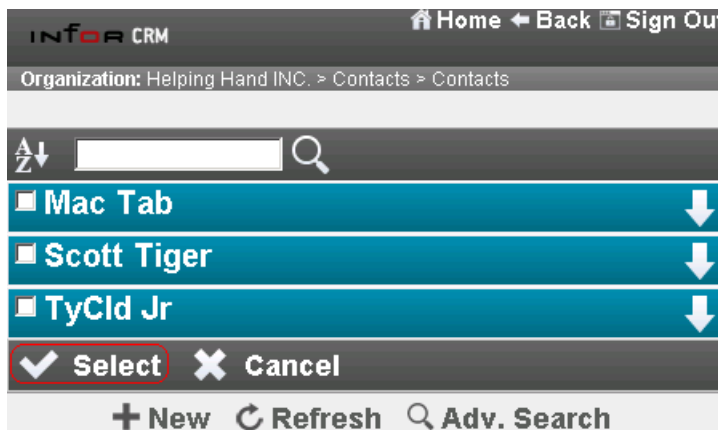
JSFContextualCaptionExtension Sample Screen

JSFCreateRelBetweenBiosAction

Property name	Description
Left side form node level	The number of levels to go up from this current form to get at the relationship attributes that identify the left side of the relationship. This is the same level as the form that invoked JSFSetupRelBetweenBiosAction and saved the relationship identification attributes to the buckets.
Uncommitted uow	Set to true to use save the uncommitted changes to the state. Otherwise, the changes will be saved immediately.

This extension is used in conjunction with JSFSetupRelBetweenBiosAction to relate a M2M relationship. Place this extension on a button that will be clicked on in a multi-select list view form where the items selected will be used to relate the primary entity configured in the left side BIO. You may use the desktop extension “GetSelectedItemsInList” to obtain the list of selected items in the focus before this extension runs.

Note: This extension will save the default UnitOfWork.



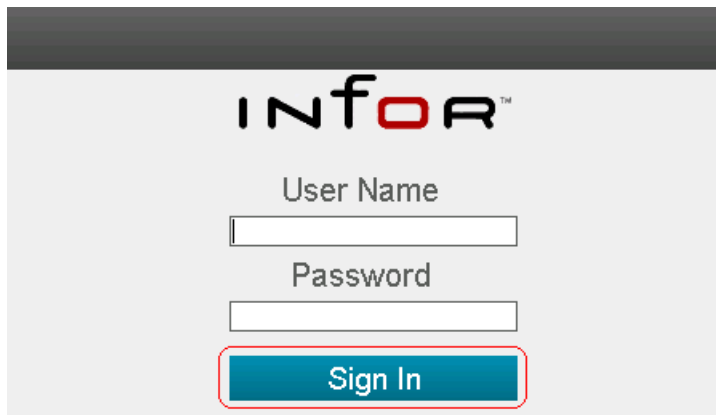
JSFCreateRelBetweenBiosAction Sample Screen

JSFLoginExtension

This extension is associated with a LOGIN_TRINIDAD form type and should be mapped to a Login button clickEvent that will initiate passing credentials to SSO to obtain a token.

The username and password are properties of the LoginControllerBean. These are set as UI widgets using the widget names “user_name” and “password”.

If successful in obtaining a SSO token, a loginSuccessfulEvent is fired. Otherwise, a loginUnsuccessfulEvent is fired.

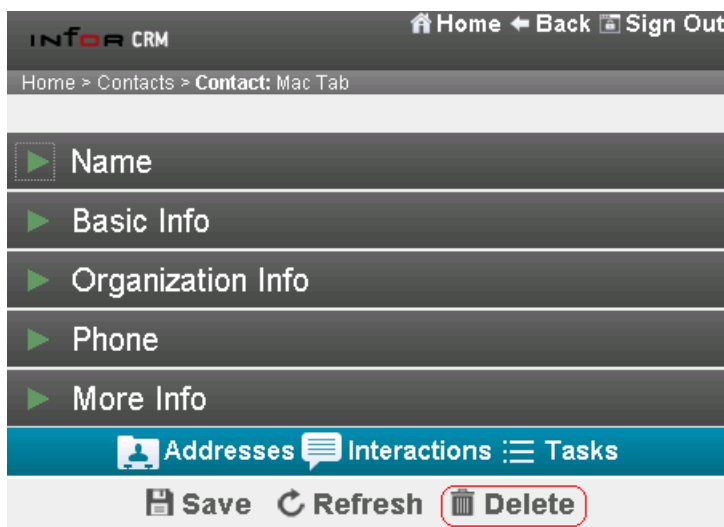


JSFLoginExtension Sample Screen

JSFMarkDeleteAction

This extension is used to implement a delete confirmation prompt. Map this to a clickEvent on a delete button, and the first click will inform the user that the item is about to be deleted. The user can click on the delete button again to confirm or refresh to cancel.

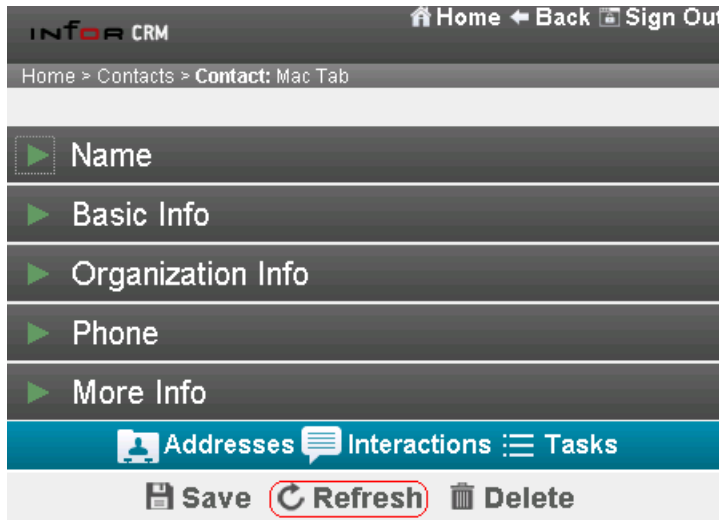
Note: All other widgets are disabled except for delete and refresh when prompting the user for confirmation.



JSFMarkDeleteAction Sample Screen

JSFRefreshBiosAction

This extension is used to refresh the default UnitOfWork, clearing the internal cache, and re-fetching any BIOs being displayed from the datasources. It also removes the delete confirmation prompt.



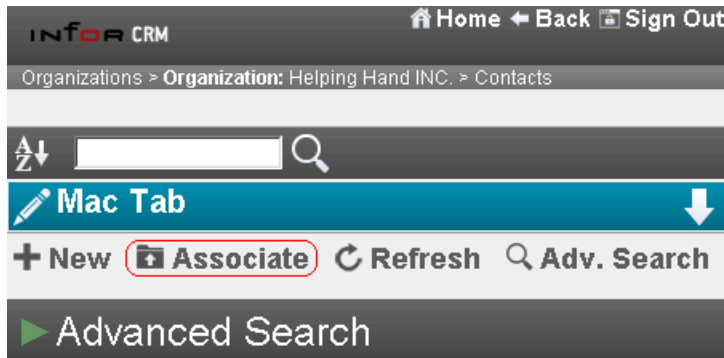
JSFRefreshBiosAction Sample Screen

JSFSetupRelBetweenBiosAction

Property name	Description
Left side bio from chained extension	Set to true if you want to obtain the left side BIO from the focus of a chained relationship instead of using the "left side bio node level".
Left side bio node level	The number of levels to go up the form tree to retrieve the left side BIO in the relationship
Relationship attribute	The name of the attribute on the left side that is used to set the relationship.
Role name	The role of the relationship (if any)

This extension is used to relate M2M relationships. Map this extension to a clickEvent on a button that will launch navigation to a list form where you can choose the entities on the right side of the relationship.

Out-of-the-box, the Associate button in a multiple selection list view uses this extension to relate Contacts to Organization, Contacts to Opportunity, and Products to Opportunity.



JSFSetupRelBetweenBiosAction Sample Screen

Defining your own Custom Extensions

Any extensions you write or extend from existing core or application will be considered a custom form based extension. You will need to determine the type of events you want to map to first and then based on that use the appropriate base class.

Extension Base Class

To start, you must write a Java class that extends one of the following based on the extension type:

Form: `com.infor.shr.ui.jsf.views.extensions.JSFFormExtensionBase`

Widget: `com.infor.shr.ui.jsf.views.extensions.JSFFormWidgetExtensionBase`

Action: `com.infor.shr.ui.jsf.action.extensions.JSFActionExtensionBase`

You will then need to override one (or more) of the public methods in this abstract class depending on which event you are interested in monitoring. The methods and the argument details are described in the source code template available to you.

JSF State Interfaces

The arguments passed into your extension will allow you to obtain contextual information needed to process your event. For backwards compatibility, some interfaces offer both JSF and non-JSF based APIs. You should use the JSF based APIs (such as `EpnysJSFState`) whenever possible when implementing an extension for the Mobile Wireless application. The JSF interfaces will limit the set of methods to only the ones that are supported by the framework for Mobile Wireless.

Overview

Navigation definitions are the metadata settings that allow you to specify when and where the UI will transition from one form to the next. In Mobile Wireless, the navigation always begins with some sort of event based on a user interaction such as a `clickEvent` on a button. The processing of the event will result in an outcome value; either from an extension that fired, or by taking the default navigation from metadata. The outcome is then mapped to one of the navigations defined for that action, which will contain a destination form and perspective to which to navigate.

In the runtime, the UI Framework ultimately delegates the execution of navigations for Mobile Wireless application to the JSF framework. A JSF application uses the outcome result as a key to lookup one of the navigation entries for the target JSPX page that is defined in `faces-config.xml`.

Since `faces-config.xml` is generated at design time, the metadata must indicate all possible outcome values from the navigation configuration for each source action triggering the navigation.

Note: Changes made to navigation metadata requires you to rebuild the application using `MakeJSF` and restart the server. This is true even if you have configured a development environment.

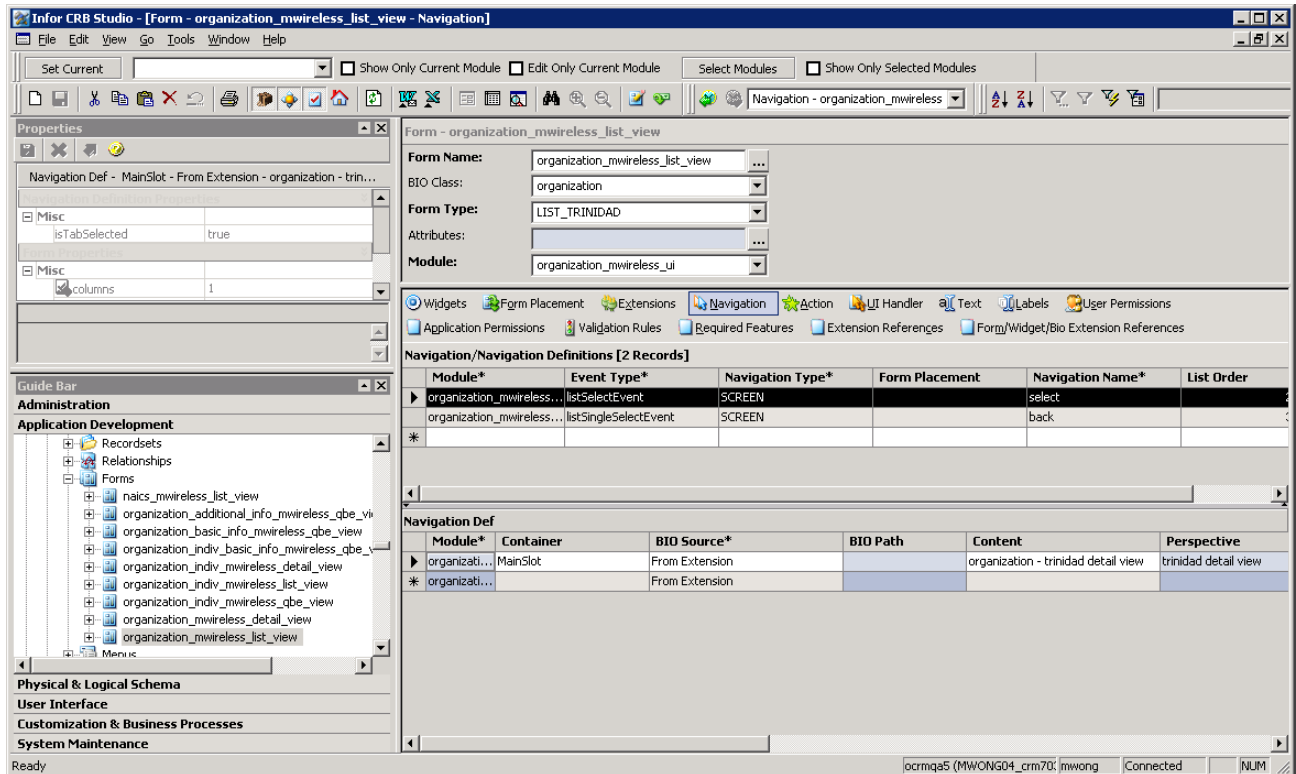
Navigation Configuration

The Studio screens for configuring navigations is identical to that of the desktop application, however, there are a few limitations imposed by the Mobile Wireless that you need to be aware of described in this section.

Every Mobile Wireless form has a **Navigation** tab in Studio that contains the Navigation definitions for the form events fired. Use this if you plan to include navigation as part of processing the form events.

Below is a screenshot of the navigation for the “`organization_mwireless_list_view`” form events.

Configuring Navigations



If your navigation is based on a form widget event, double click on the form widget in Studio and you will see a **Navigation** tab for that widget.

Whether or not you are mapping form or form widget event navigations, the process is similar and you will need to configure the following entries described below.

Note: Regardless of what you set the navigation to be, if validations are checked and have failed (see "State Handling" on page 75) or if an exception is thrown at any time the event is processed, the navigation will not proceed but will preserve the dirty UI values.

Event Type

This is the event you want to map to the navigation. When the event occurs, the default navigation outcome will be mapped to the entity defined.

Navigation Type

While the desktop application supports many navigation types, the Mobile Wireless application only supports: SCREEN, SELF and NO CHANGE.

SCREEN navigation will transition the entire screen currently being rendered to a new JSPX page. The current screen and its state will be saved to the context so that you can return to it using a “back” navigation.

SELF and NO CHANGE navigation are identical and both result in refreshing the current screen with the same form as what is currently being displayed. Use this if you want to simply not navigate to another form.

Note: All action-based widgets including `tr_commandLink`, `tr_commandButton`, `tr_selectOneList` MUST have a navigation defined. Otherwise, you will get a scrutiny error. Simply map it to NO CHANGE if you don’t want to navigate away from the current form when they are clicked on. This includes any label that is clickable using the `labelClickEvent`.

Form Placement

Not supported for Mobile Wireless. Leave this blank.

Navigation Name

The navigation name is the name of the outcome that will be passed to JSF to perform the navigation. It must be unique for each navigation entry you create per form or form widget. You can have multiple navigation entries for the same event but the first one in the list will be considered the default and will be used, unless it is overridden by an extension that sets the navigation using the API inside the extension as follows:

```
jsfActionContext.setNavigation(navigationName)
```

Two special navigation names are reserved for the following purpose described below:

back: Indicates navigation where you want the navigation to return to the previous form that is in the form stack. All non-reserved navigation pushes the current form into this stack and a “back” navigation brings the user back by popping the form at the top of the stack.

home: Indicates a navigation where you want the navigation to return to the home form. The home form is the form that is immediately after the login form in the form stack and has the form name “mwireless_home_page”.

Note: While you could define a navigation for the **Logoff** button, it is generally not needed because the call to `EpnyJSFState.logout()` will invalidate the session and force the framework to automatically redirect the user to the login screen.

List Order

This is an integer value that is used to order the navigation entries in ascending order. Having multiple navigation entries with the same event type may require that you order your entries in such a way that

the top most list order be the one you want to designate as the default navigation. The default navigation is the one that will be used when the event occurs and no other extension has invoked the `setNavigation()` API with another outcome.

Screen (Property Value)

The only screen property value you should use is **Root Screen**.

State Handling (Property Value)

Choose one of the state handling values described in the section "State Handling" on page 75.

By default, the **State Handling** is set to **Confirm All**.

Navigation Definition Configuration

The navigation entry has a sub-form in Studio where you can define one or more navigation definitions for that entry. A navigation definition is necessary because it has configuration about the how to locate the destination form to navigate to when the event occurs.

Unlike the desktop application, the Mobile Wireless application only allows at most 1 navigation definition per navigation. Although Studio allows you to create more than one, it will only use the first one. Since the behavior is unpredictable, do not attempt to create more than one Navigation definition entry.

Container

This is used to indicate which part of the screen to which the navigation should be applied. Mobile Wireless only supports SCREEN navigation, you should always use "MainSlot" container.

Note: You will not see this configuration for a NO_CHANGE navigation.

BIO source

A very common use case is when you navigate to a new form, you want that form to use a focus BIO based on some action in the current form. For example: If you are going from a list form to a detail form, you want to take the selected BIO in the list form and set the focus to the detail form so that the user can see the details of what he/she selected.

Here are the possible options you can choose for the BIO Source:

None: Use this if you don't wish to set the focus to the target form.

From Extension: The source BIO is set by the extension that is assigned to the specified event type for this navigation definition. This is the most common use case.

From Source Form: The source BIO is set by the focus of the form from which the navigation originates.

From Target Form: Not supported.

From Target Form's Parent: Not supported.

BIO Path

This is an optional setting where you can specify a path relative to the BIO source to set the focus of the target form. For example, a navigation from a detail individual to a list view of related addresses, you should set the focus of the target form to the BIO path “individual:addresses”.

Content

The content identifies the target BIO and perspective of the destination form to which you wish to navigate.

Note: You do not need to set the content for the reserved navigations for “home” and “back”.

Perspective

This is a read-only field that informs you the name of the perspective of the target Content you have selected. It must be one of the Trinidad supported perspectives. Do not navigate to a form that does not have a perspective that does not start with “trinidad” in the name.

Special-Form type

Another read-only field that informs you what the form type is in the case of a special form type based on the selected target Content.

Tab List Order

Not used by Mobile Wireless.

Overriding Property Values

The property value window when picking a Navigation Definition will contain a sub-tree called “Form Properties” that can be used as overriding property values to the form that you are navigating to. Any value that you specify here that is not default (non-blue in color such as rowSelection in the screenshot below) will override the property value definition with the same name to the target form to which you are navigating. Overriding a form property is useful when you are sharing the same form metadata from multiple navigations where the use cases differ by one or more property value settings.

For example, the screenshot below illustrates a tr_selectOneList widget that will navigate to the “organization_mwireless_list_view” but with the rowSelection set to “single”. When the list view is rendered, the overriding rowSelection will take effect and instead of displaying a list view of organizations with a **view detail** button, a **selection** button will appear allowing the user to choose one organization in the list.

The screenshot shows the Infor CRB Studio interface. The main window is titled "Infor CRB Studio - [Form Widget - individual_organization_infor_mwireless_detail_view.organization - Navigation]". The Properties window is open, showing the "Form Properties" section for the "organization" widget. The "rowSelection" property is set to "single" (highlighted in blue). The "view detail" property is set to "true" (highlighted in blue). The "Navigation/Navigation Definitions" table shows two records for the "individual_mwireless_ui" module.

Module*	Event Type*	Navigation Type*	Form Placement
individual_mwireless_ui	clickEvent	SCREEN	
individual_mwireless_ui	labelClickEvent	SCREEN	

The "Navigation Def" table shows the following records:

Module*	Container	BIO Source*	BIO Path	Cont
individual...	MainSlot	From Extension		organ
* individual...		From Extension		

State Handling

State handling is a property value of a navigation entry. It is used to control what happens to the uncommitted data model values that have been transferred from the UI by JSF.

Confirm All

If the UI has values that have been changed by the user, the form will be marked dirty and the user will be prompted with a confirmation that they will either need to Save the values (via a **Save** button) or discard them (via a **Refresh** button). The navigation will not proceed unless the user does one or the other. If the UI is not marked dirty, then the navigation will proceed without prompting.

This option is helpful to warn the user that he/she has unsaved values and should either save them or discard them before navigating away from the current screen. This is typically used when displaying a detail view of a root entity.

Confirm This

Same as “Confirm All”. Use **Confirm All** instead.

Confirm Parent

Same as “Confirm All”. Use **Confirm All** instead.

Discard

Any UI values that have not been committed will be discarded from the state. Since this resets the values to whatever is currently committed, there is no validation checking with this setting.

You should use **Discard** for a **Refresh** or a **Cancel** button or any type of operation where you do not want to save any of the values changed in the UI.

Save All

If the UI has dirty values, the dirty values will be preserved to the state for that form. When the user returns to the form at a later time, the dirty values will be restored back in the UI.

Note: This is NOT the same as committing to a database. No data is being saved to the op-db. If you want to save to the database, you will need to invoke the `SaveAction` extension.

This is typically used when you want to implement a link to a related entity in a detail view. The contents of the unsaved values of the entity will be preserved in the state while the user is navigated to the related entity screens. When the user returns back to the root entity, all unsaved values will be restored to the UI.

Save This

Same as “Save All”. Use **Save All** instead.

Save Parent

Same as “Save All”. Use **Save All** instead.

Send This

The dirty values from the UI will be saved to the state much like “Save All” except that validations will not be run. This is ideal in the case where you are trying to create a new entity in a detail view, where you need to navigate to a child form to associate a BIO such as in a `tr_selectOneList` widget. In this case, the user may not have all of the required fields filled out using “Send This” will preserve the dirty values but yet still allow the user to navigate to the child form.

Customizing Look and Feel using Infor Stylesheet Identifiers

8

Overview

This chapter provides information on how to change the look and feel of the Mobile Wireless application, using Infor style sheet identifiers for different smartphones.

Customizing Look and Feel

The Mobile Wireless application gives users the ability to customize the look and feel of each smartphone on an individual basis. To change the style of the application for a smartphone, you need to change the specific Cascading Style Sheet (CSS) file located at this location.

```
<<S&S Install Folder>>\shared\webapps\mwireless\skins\<<smartphone>>\<
<smartphone>>_custom.css
```

Note: There is also a <<smartphone>>.css file located in the same folder which contains the application standard styles for that specific smartphone. This file should not be modified.

For instance, if you want to change the look and feel of the application for a Black Berry smartphone, then `blackberry_custom.css` must be modified. Note that the `blackberry.css` file (which is also located in the same folder) should not be modified, as this file contains the application standard styles for the BlackBerry device.

Stylesheet Identifiers

In the sections that follow, we will discuss the different style sheet identifiers present in the Mobile Wireless application that can be customized to change the look and feel of the application.

Each entry below contains a brief description of each style and a sample output to illustrate where the style is applied on the form. The following outputs are generated by setting a background-color of light

brown and a dark brown border to each style, in order to identify the area where the style will be applied. For these examples, the `desktop_custom.css` file was used.

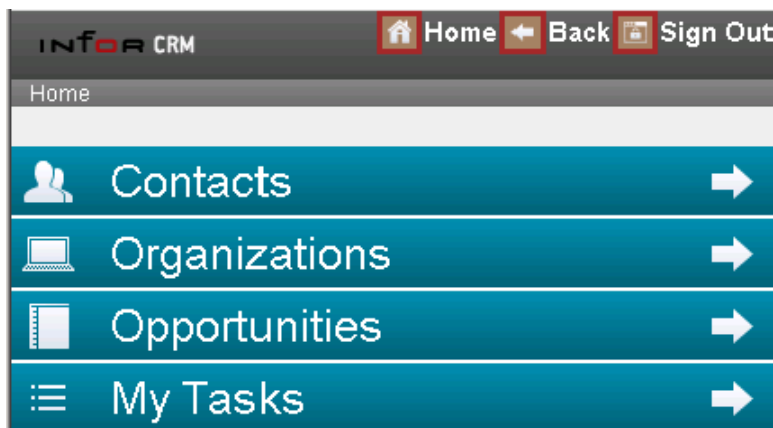
In those instances where multiple stylesheet changes need to be applied to achieve a specific layout effect (see the `inforDetailDataLabel` example), these will be highlighted in an associated bolded note.

The following CSS settings were used to set the border and background for all of the examples.

```
"border:1px solid brown; background: #A68064;"
```

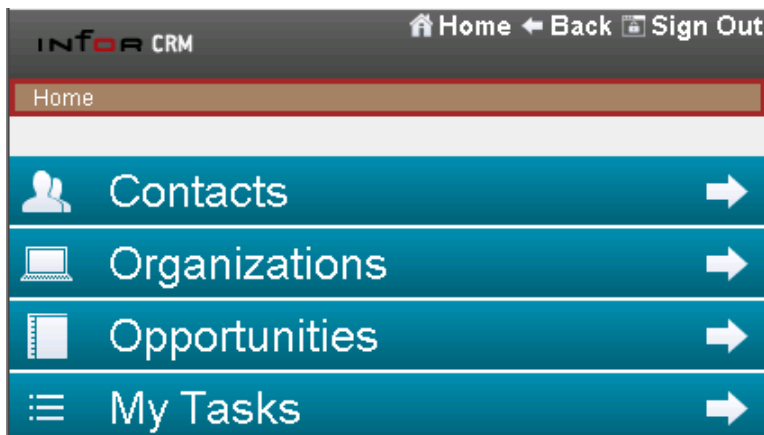
inforCommandLinkIcon

The generic style applied to the icon widget of all commandlinks including the header commandlinks.



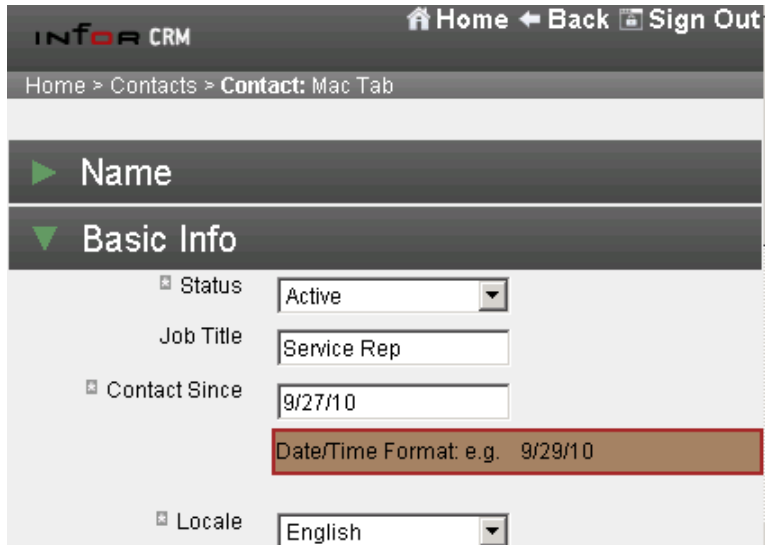
inforContextualCaption

The style for the contextual caption row.



inforDateTimeFormat

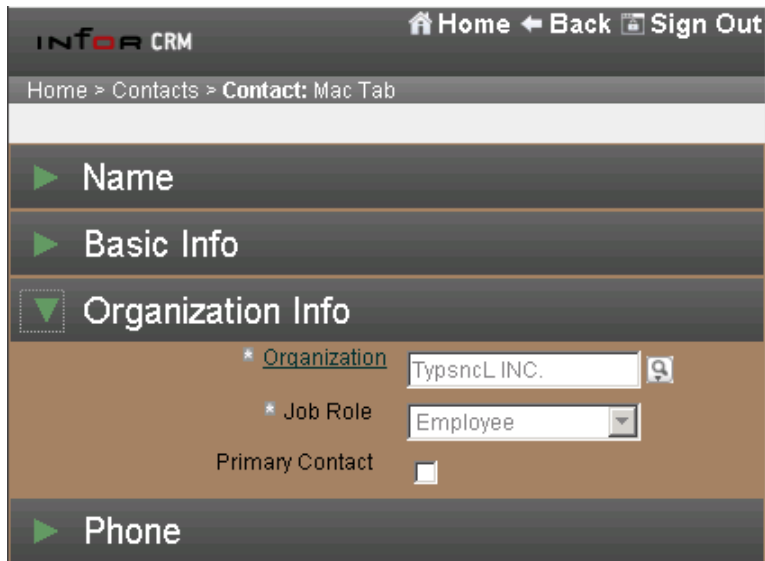
The style applied the DateTime Format.



The screenshot shows the Infor CRM interface for editing a contact. The breadcrumb trail is 'Home > Contacts > Contact: Mac Tab'. The 'Basic Info' section is expanded, showing fields for Status (Active), Job Title (Service Rep), Contact Since (9/27/10), and Locale (English). A red box highlights the 'Contact Since' field, with a tooltip indicating the 'Date/Time Format: e.g. 9/29/10'.

inforDetailData

The style in the detail view for each row of data.



The screenshot shows the Infor CRM interface for editing a contact. The breadcrumb trail is 'Home > Contacts > Contact: Mac Tab'. The 'Organization Info' section is expanded, showing fields for Organization (TypesncL INC.), Job Role (Employee), and Primary Contact (checkbox). The 'Phone' section is also visible. The 'Organization Info' section is highlighted with a brown background, indicating the 'inforDetailData' style.

inforDetailDataLabel

The style for the field labels in a detail view.

Note: A combination of `inforDetailDataLabel` and `inforDetailDataValue` should be used to get the final desired result.

infor CRM Home Back Sign Out

Home > Contacts > Contact: Mac Tab

▶ Name

▶ Basic Info

▼ Organization Info

* Organization TypsncL INC. 🔍

* Job Role Employee ▼

Primary Contact ☐

▶ Phone

`inforDetailDataValue`

The style for the input fields of the data row in a detail view.

infor CRM Home Back Sign Out

Home > Contacts > Contact: Mac Tab

▶ Name

▶ Basic Info

▼ Organization Info

Organization TypsncL INC. 🔍

Job Role Employee ▼

Primary Contact ☐

▶ Phone

`inforDetailShowDetail`

The style associated with the collapsible slots in a detail view.

The screenshot shows the Infor CRM interface for editing a contact named 'Mac Tab'. The breadcrumb trail is 'Home > Contacts > Contact: Mac Tab'. The page has a dark header with 'infor CRM', 'Home', 'Back', and 'Sign Out' links. The main content area has a light gray background. There are four expandable sections: 'Name', 'Basic Info', 'Organization Info', and 'Phone'. The 'Organization Info' section is expanded, showing fields for 'Organization' (TypsncL INC.), 'Job Role' (Employee), and 'Primary Contact' (checkbox). The 'Organization' field has a search icon. The 'Job Role' field is a dropdown menu.

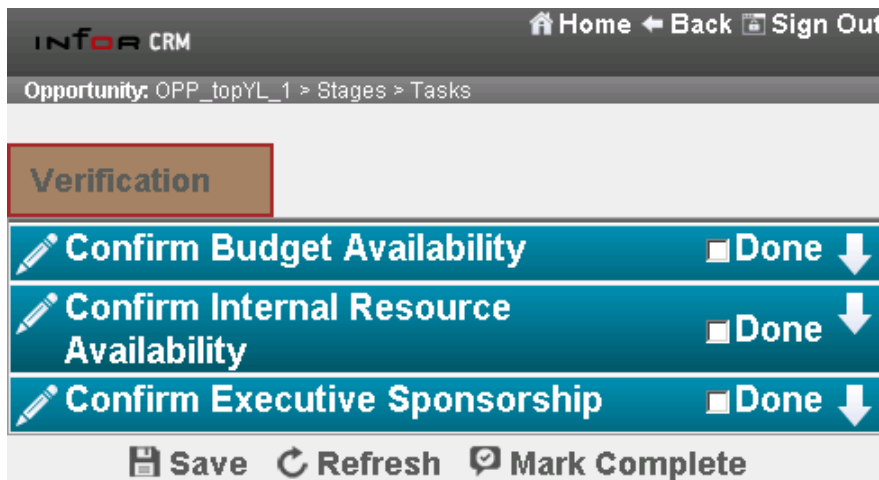
inforErrorMessage

The style for the Error message.

The screenshot shows the Infor CRM 'New Contact' form. The breadcrumb trail is 'Home > Contacts > New Contact'. A red error message box at the top says 'Some required fields are missing'. Below the error message is the 'Name' section, which is expanded. It contains three fields: 'Salutation' (dropdown menu), 'First Name' (text input with 'Mardy'), and 'Middle Name' (text input). The 'Salutation' field has a dropdown arrow. The 'First Name' field has a small icon to its left.

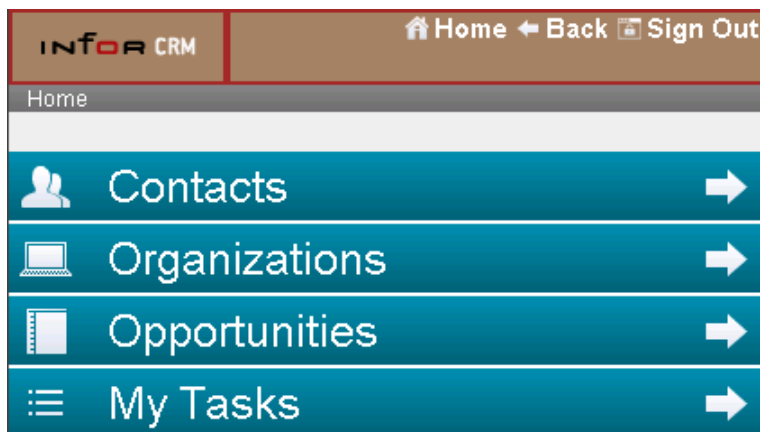
inforFormCaption

The style for the form heading (stage name) on sales process tasks list view.



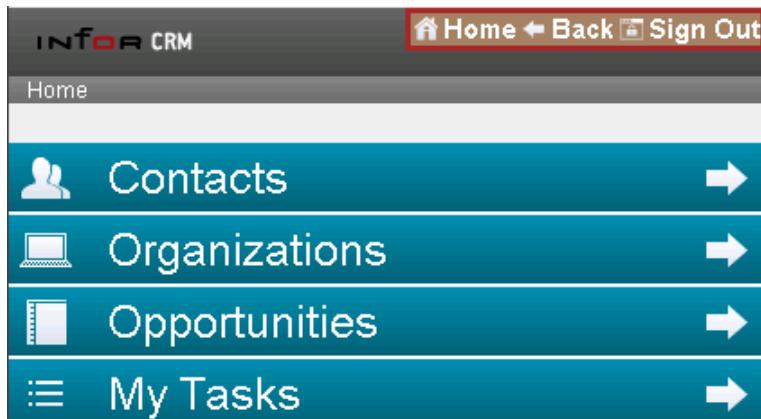
inforHeader

The style for the header slot that appears on every page except the login page.



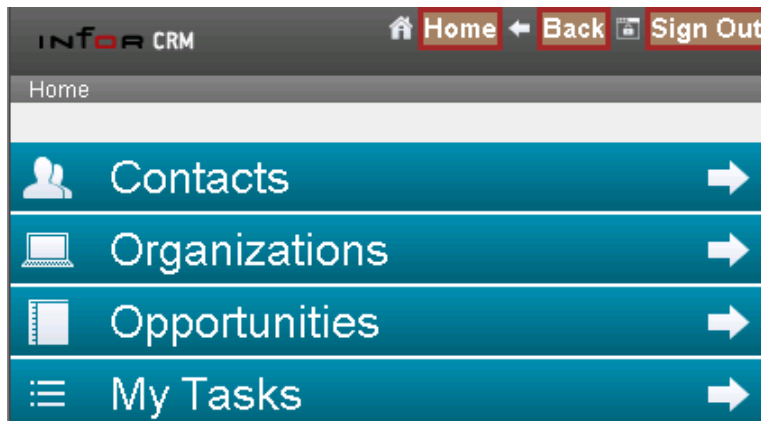
inforHeaderToolbar

The style for the header toolbar which is present at the top right corner of every page except the login page.



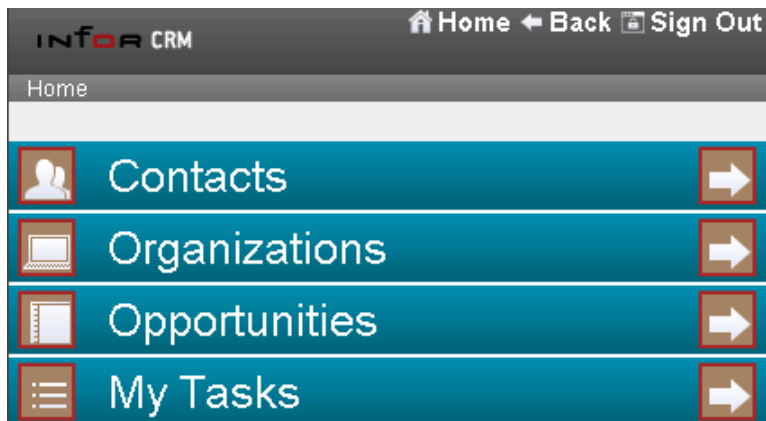
inforHeaderToolBarWidget

The style for the header commandlink widgets.



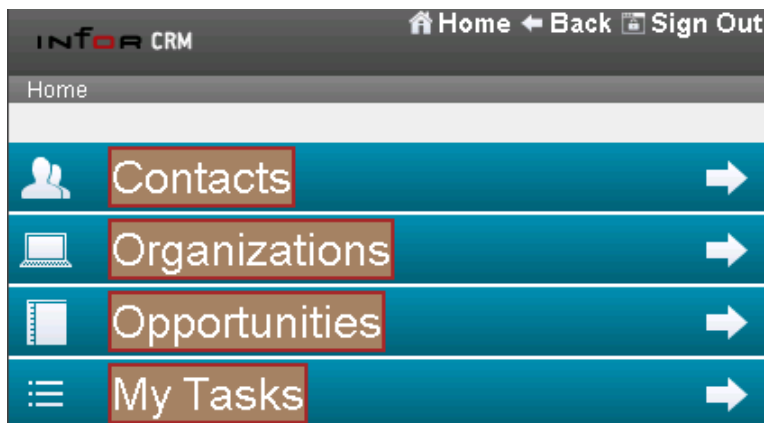
inforHomeButton

The style for the buttons on the home page.



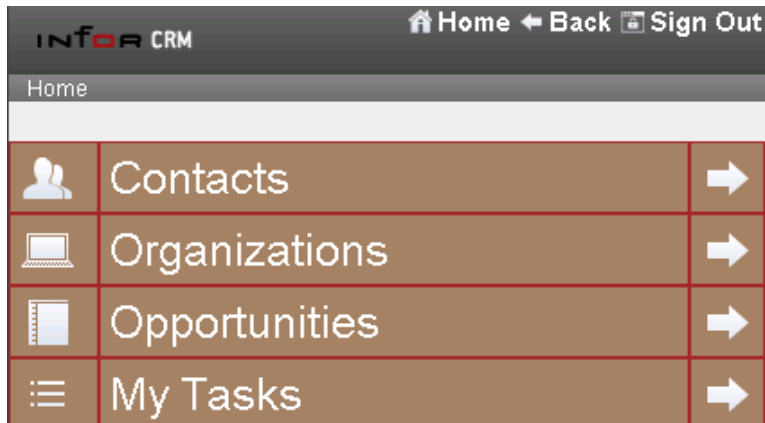
inforHomeLink

The style for the links on the home page.



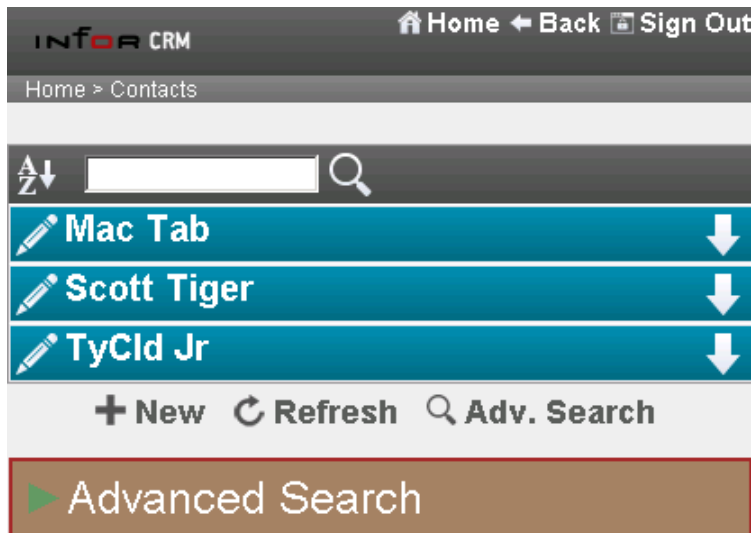
inforHomeNav

The style for the navigation row on the home view.



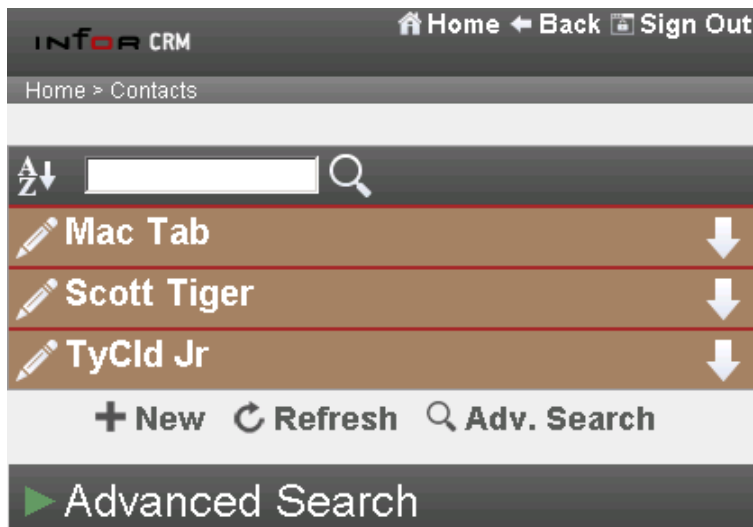
inforListAdvancedSearch

The style for the Advanced Search collapsible slot.



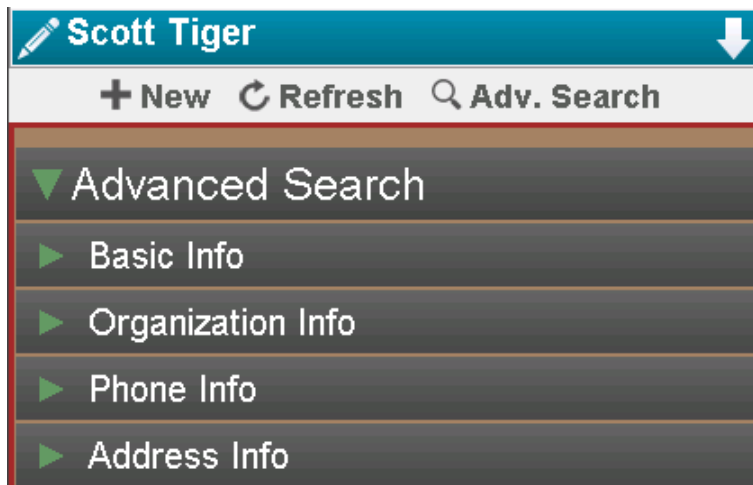
inforListDataColumn

The style for each row of data. Please note that this style is overridden by the style applied to the inner layers like `inforListIconColumn` & `inforListPrimaryAttr`.



inforListFields

The style for the panel row layout associated with the Advanced Search fields in a list view.



inforListFieldsLabel

The style applied for the label column of the more fields in a list view.

Note: A combination of `inforListFieldsLabel` and `inforListFieldsValue` should be used to get the final desired result.

▼ Advanced Search

▼ Basic Info

Status	---
Job Title	
Employer	
E-mail	

inforListFieldsValue

The style for the value column of the Advanced Search fields in a list view.

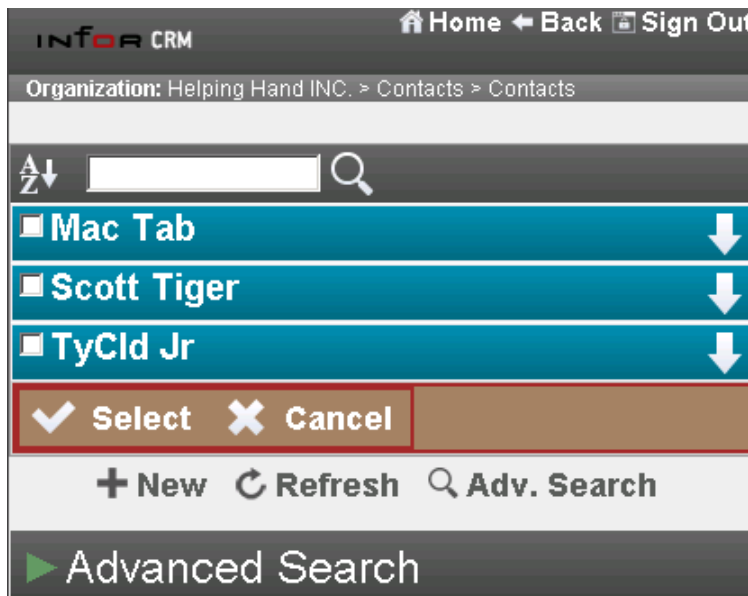
▼ Advanced Search

▼ Basic Info

Status	---	
Job Title		
Employer		
E-mail		

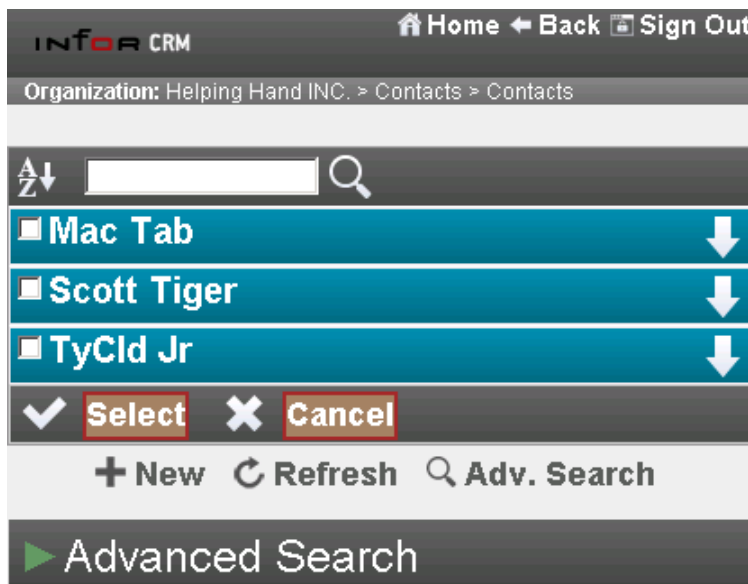
inforListFooter

The style applied to multi-select list view footer row.



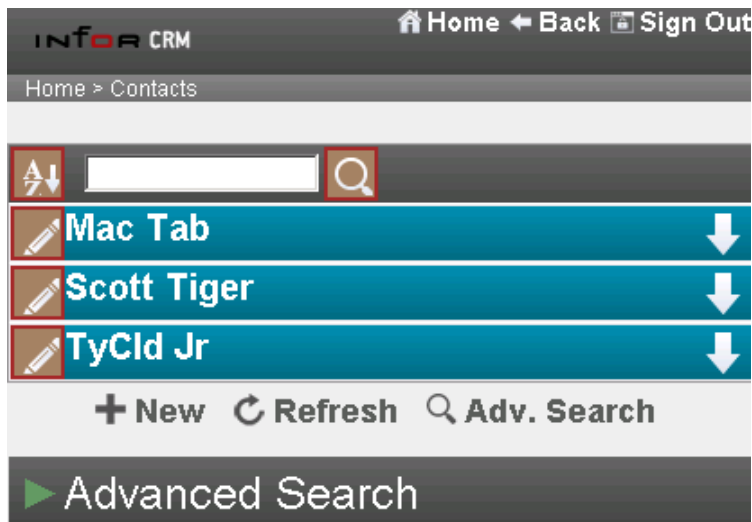
inforListFooterWidget

The style applied to multi-select list view footer commandlinks.



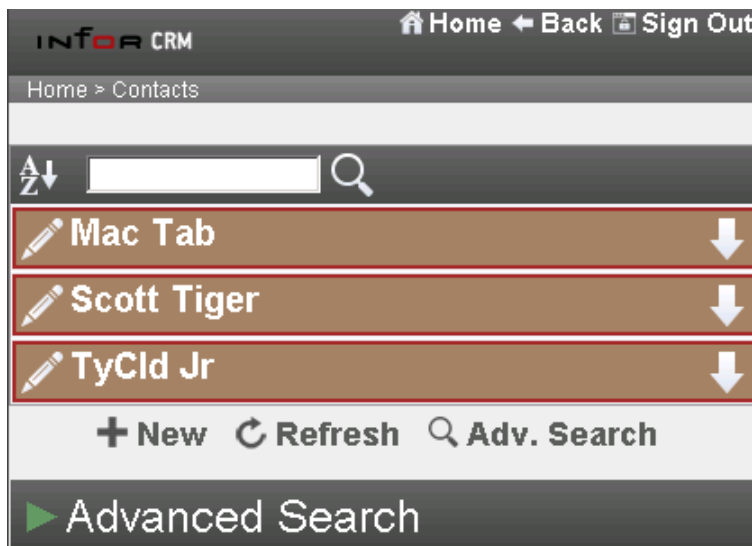
inforListIconColumn

The style for the view detail button and search button.



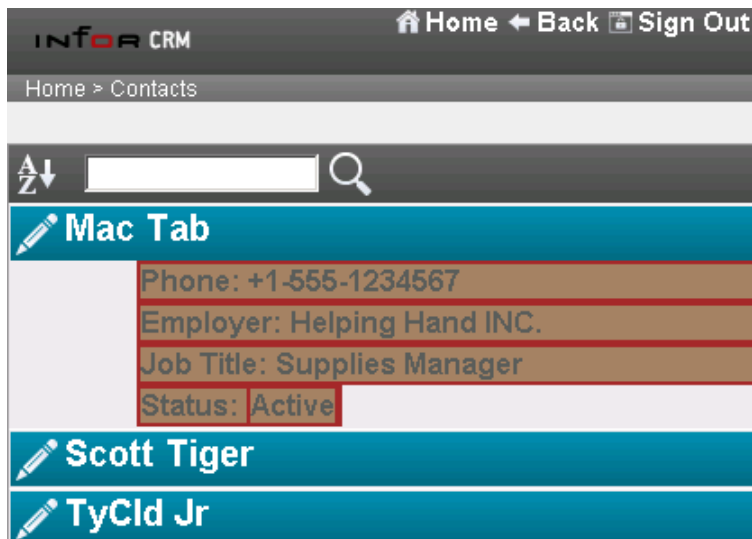
inforListPrimaryAttr

The style for the primary attribute of the list view. This style is applied to the nested table within each row.



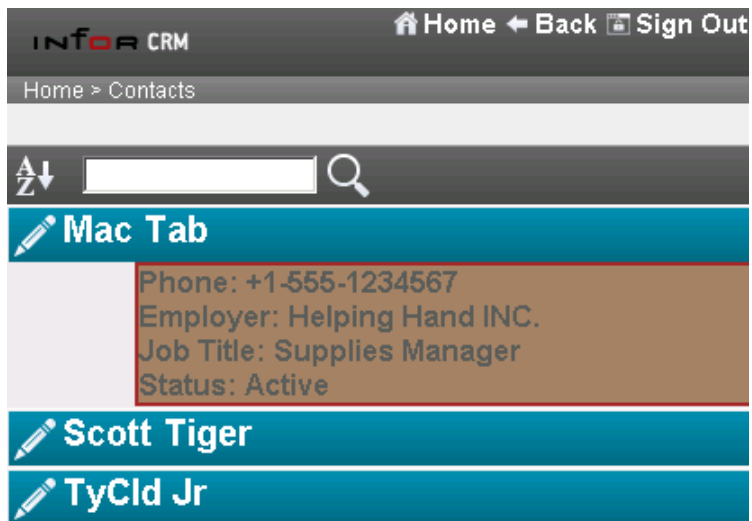
inforListSecondaryAttr

The style for the Secondary attribute line items.



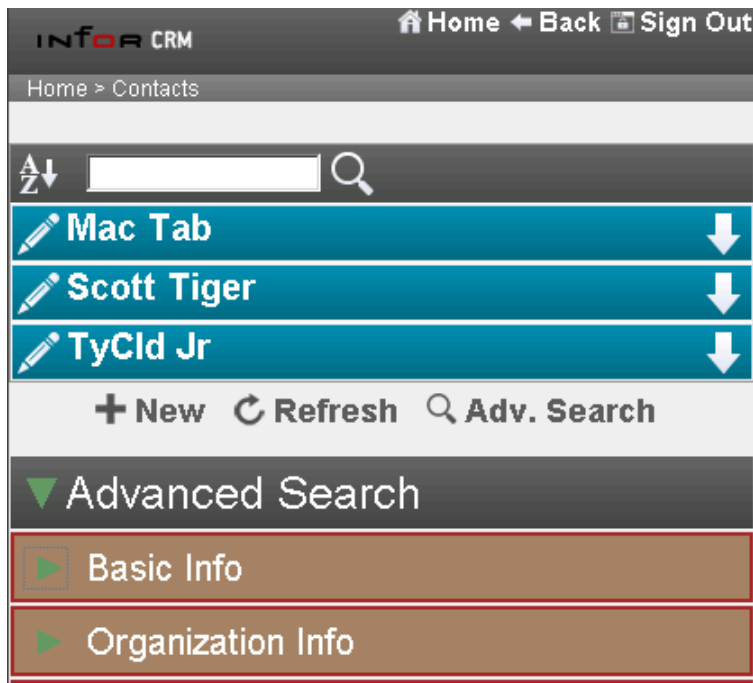
inforListSecondaryAttrs

The style for the table holding the Secondary attributes of the list view.



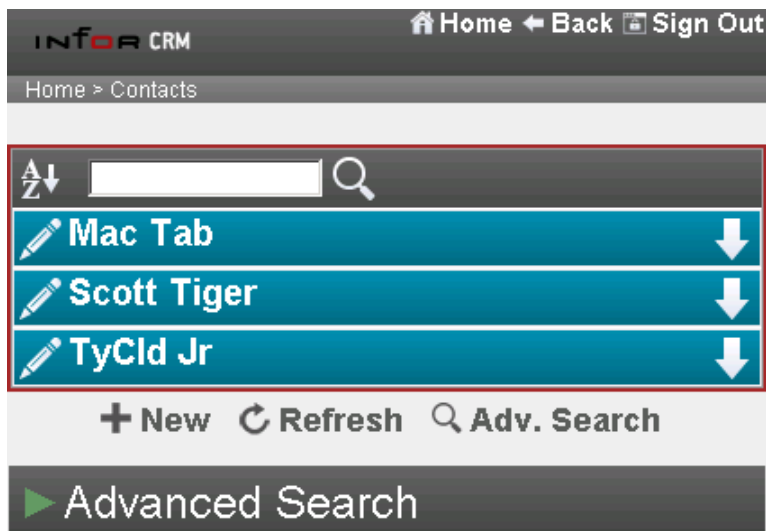
inforListShowDetail

The style for the Advanced Search sub menus.



inforListTable

The style for the table that holds the basic search and search results.



inforLoginButton

The style for the Sign-In button.



The image shows a login form for Infor. At the top is a dark gray header bar. Below it is the Infor logo, where 'in' is black, 'fo' is red, and 'r' is black. The form consists of two text input fields, one labeled 'User Name' and one labeled 'Password'. Below these is a brown button with the text 'Sign In' in white. The entire form is set against a light gray background.

inforLoginDataLabel

The style for the label of the data row in the login page.

Note: A combination of inforLoginDataLabel and inforLoginDataValue should be used to get the final desired result.



The image shows the same login form as before, but with the 'inforLoginDataLabel' style applied. The labels 'User Name' and 'Password' are now highlighted with a brown background and a red border. The 'Sign In' button is also highlighted with a blue background and a red border. The input fields are highlighted with a white background and a red border. The entire form is set against a light gray background.

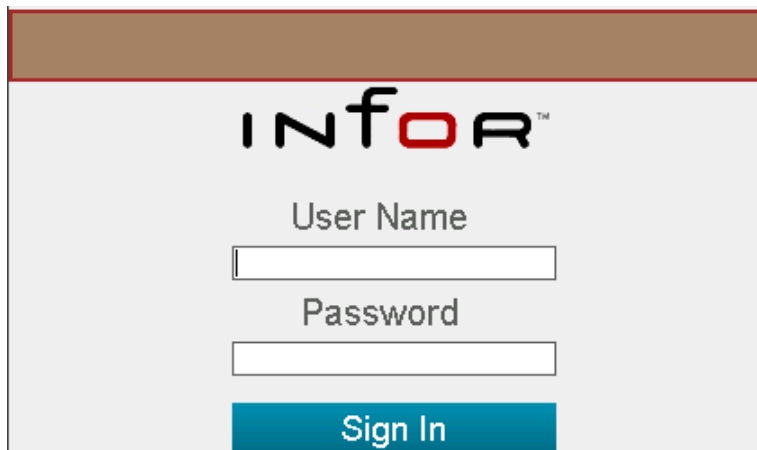
inforLoginDataValue

The style for the input fields in the login page. In addition, this style is also applied to the cells holding the logo and the sign-in button.



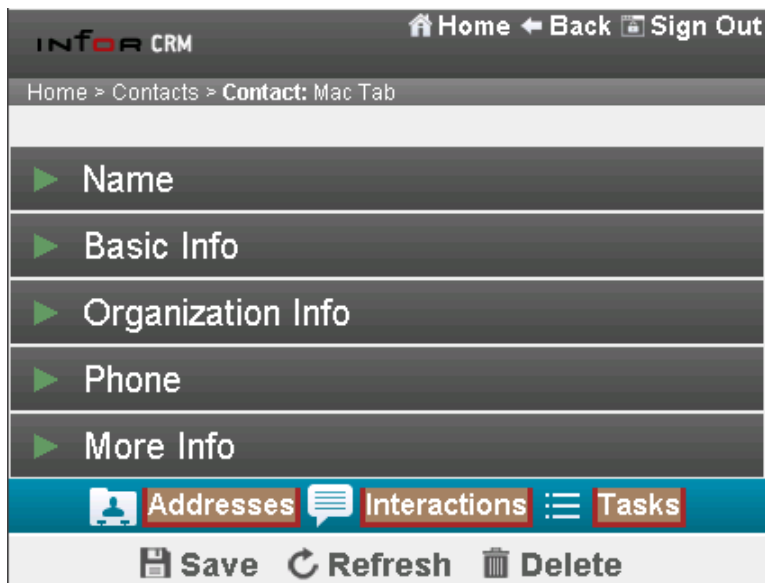
inforLoginHeader

The style for the header row in the login page.



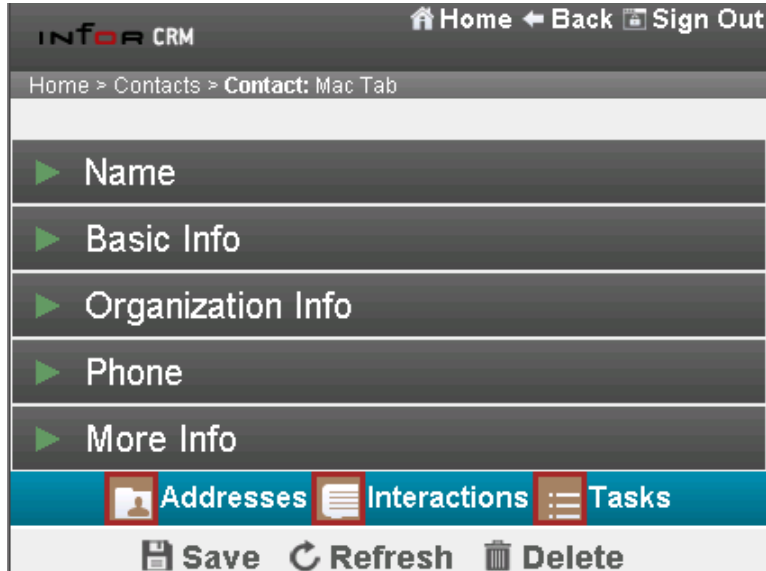
inforRelatedEntity

The style for the Related Entity links in a detail view.



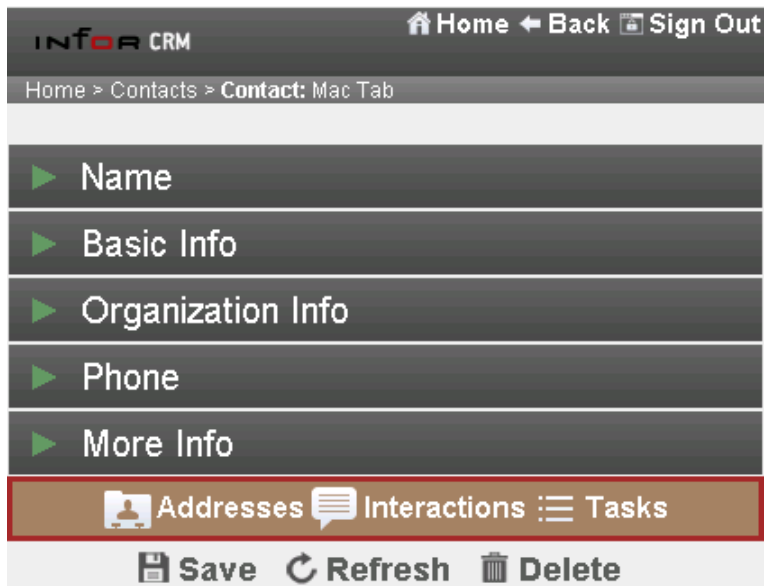
inforRelatedEntityIcon

The style for the Related Entity icons in a detail view.



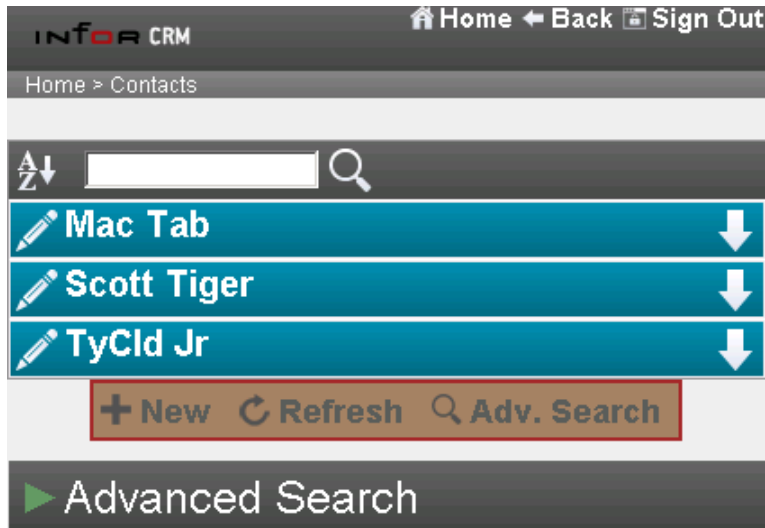
inforRelatedLinks

The style for the Related Entities row in a detail view.



inforToolbar

The style applied for the toolbar.



inforToolbarWidget

The style applied for each widget in the toolbar.

