Barista® Application Framework

Getting Started

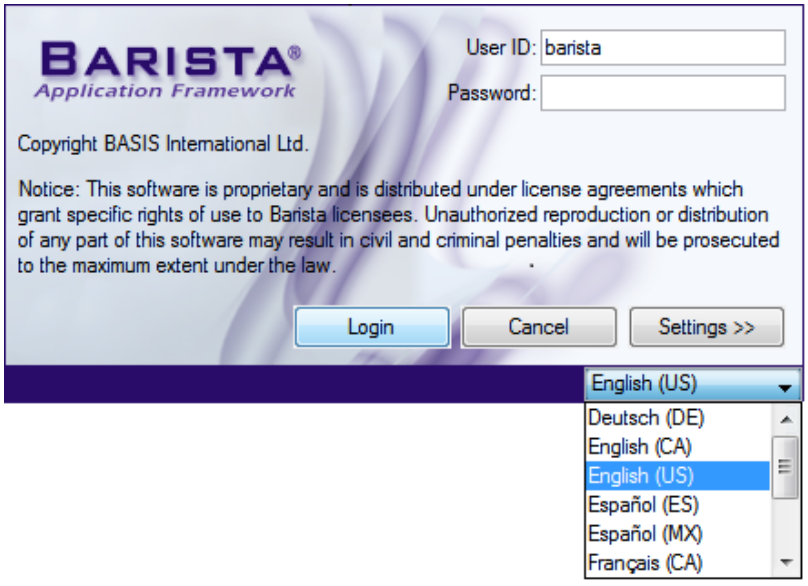
Introduction

Barista is:

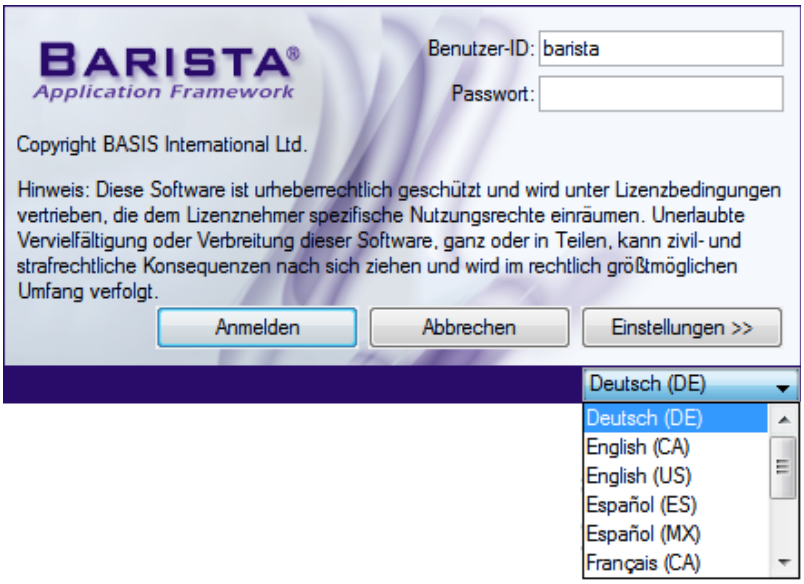
- A dictionary-based GUI development platform
- A workbench for developing data-centric applications
- An application runtime environment for end users

Logging In

The Barista login form will be displayed in the language associated with your BBj installation. To change languages, select an option in the list in the lower right corner of the form.



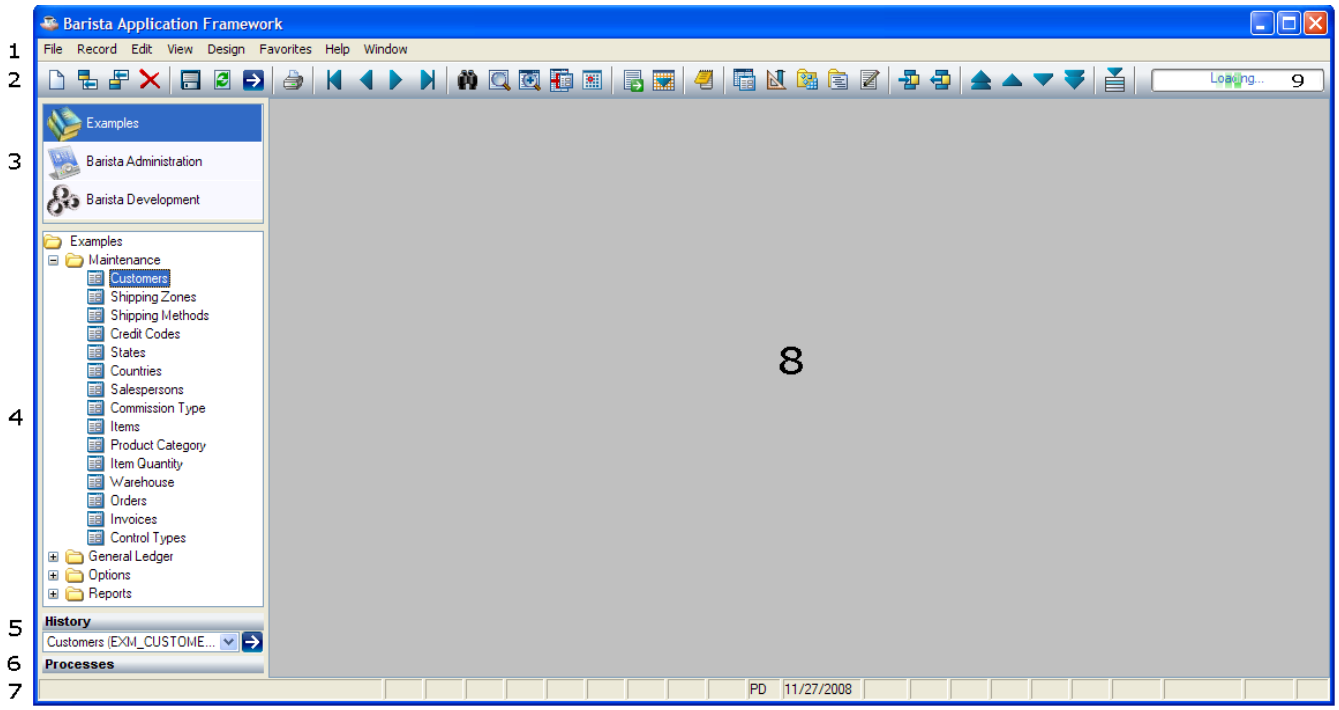
As a new language is selected, the login form redisplays in the new language:



Enter the user ID and password (if any) and click the [Login] button.

Environment

The Barista Application Framework is organized as an MDI (Multiple Document Interface) window with the following sections:

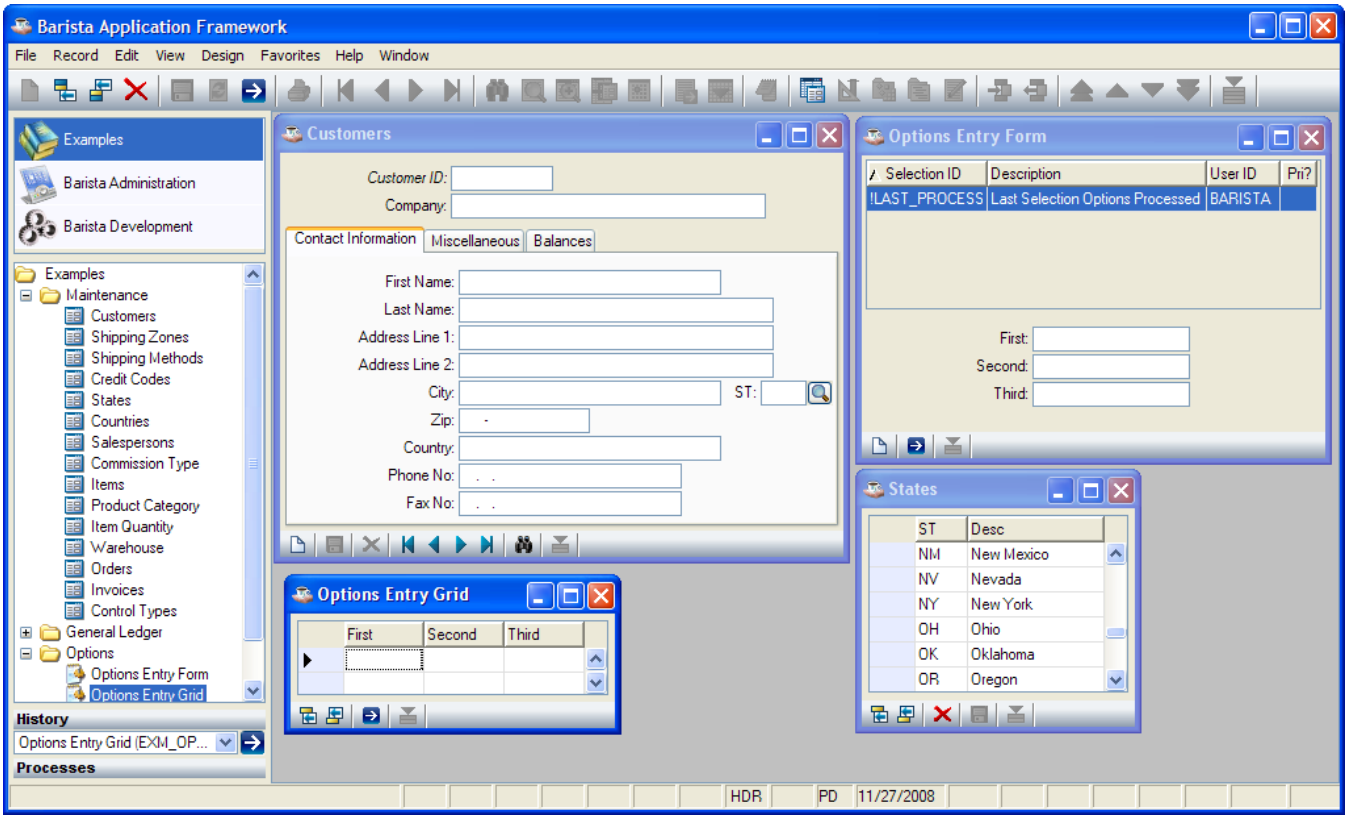


1. Menu Bar. Menu items are dynamically enabled and disabled based on the current task.
2. Toolbar. Toolbuttons are dynamically enabled and disabled based on the current task.
3. Application List. From this list, the user or developer selects the application to work with.
4. Application Menu. From this menu, the user or developer selects a specific task within an application.
5. History. This is a list of all tasks performed during the current session.
6. Processes. Long-running processes display their status here. See Process Status.
7. Status bar. See Status Bar Reference
8. MDI client area. This is where active tasks will appear.
9. Progress. Progress meter displayed when loading and saving tasks.

The panel containing items 3 through 6 can be docked/undocked using File□Dock Applications Menu or hidden using File□Show Applications Menu.

Using the Barista Application Framework

To run a program in the Barista Application Framework, select an application from the application list and a program from the application menu. Multiple programs can be running at the same time. For example:

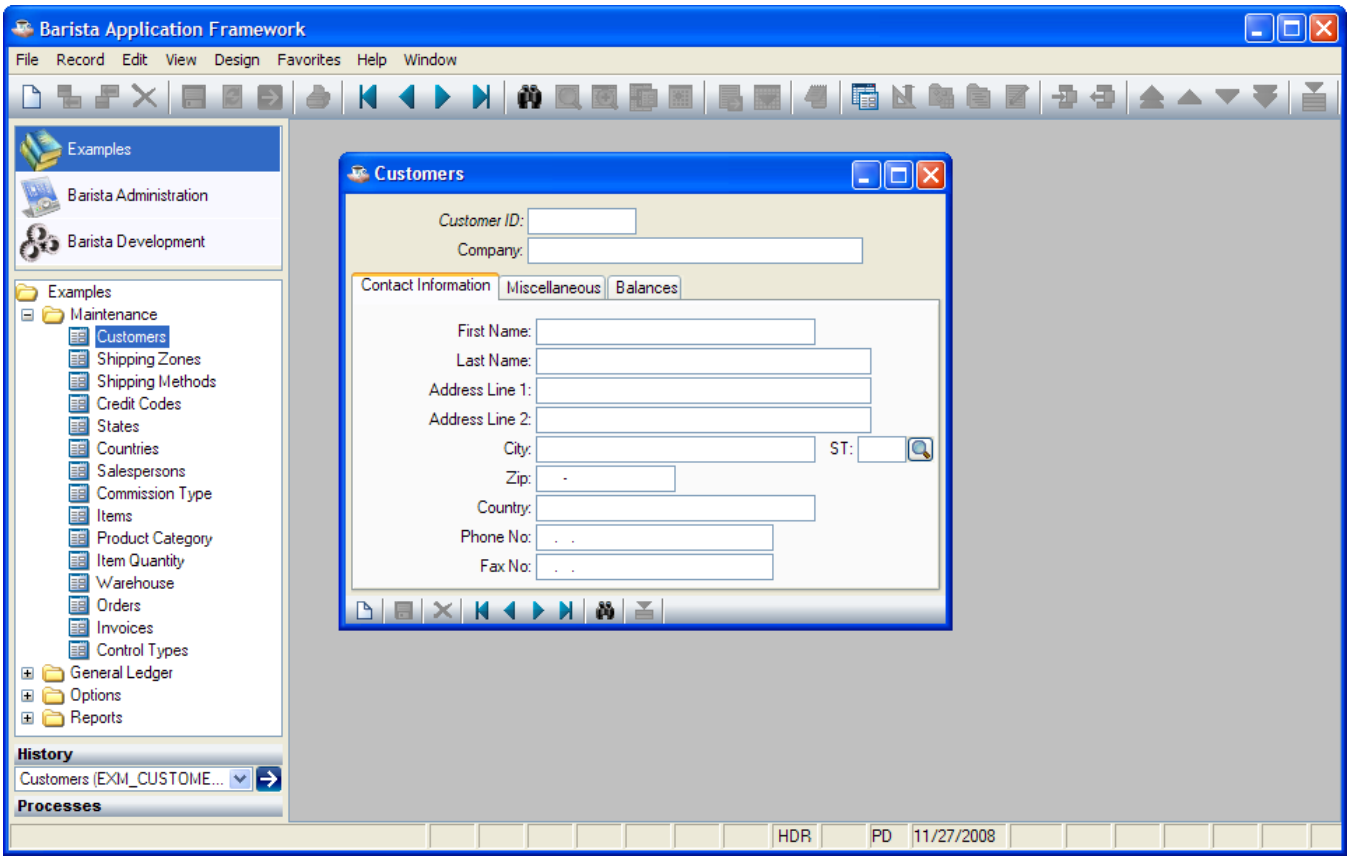


Barista generates forms according to one of four basic styles:

1. **Maintenance Form**. A detail form to maintain a data file.
2. **Maintenance Grid**. A grid-based form to maintain a data file.
3. **Options Entry Form**. A detail form used to enter information that will be passed to a program.
4. **Options Entry Grid**. A grid-based form used to enter information that will be passed to a program.

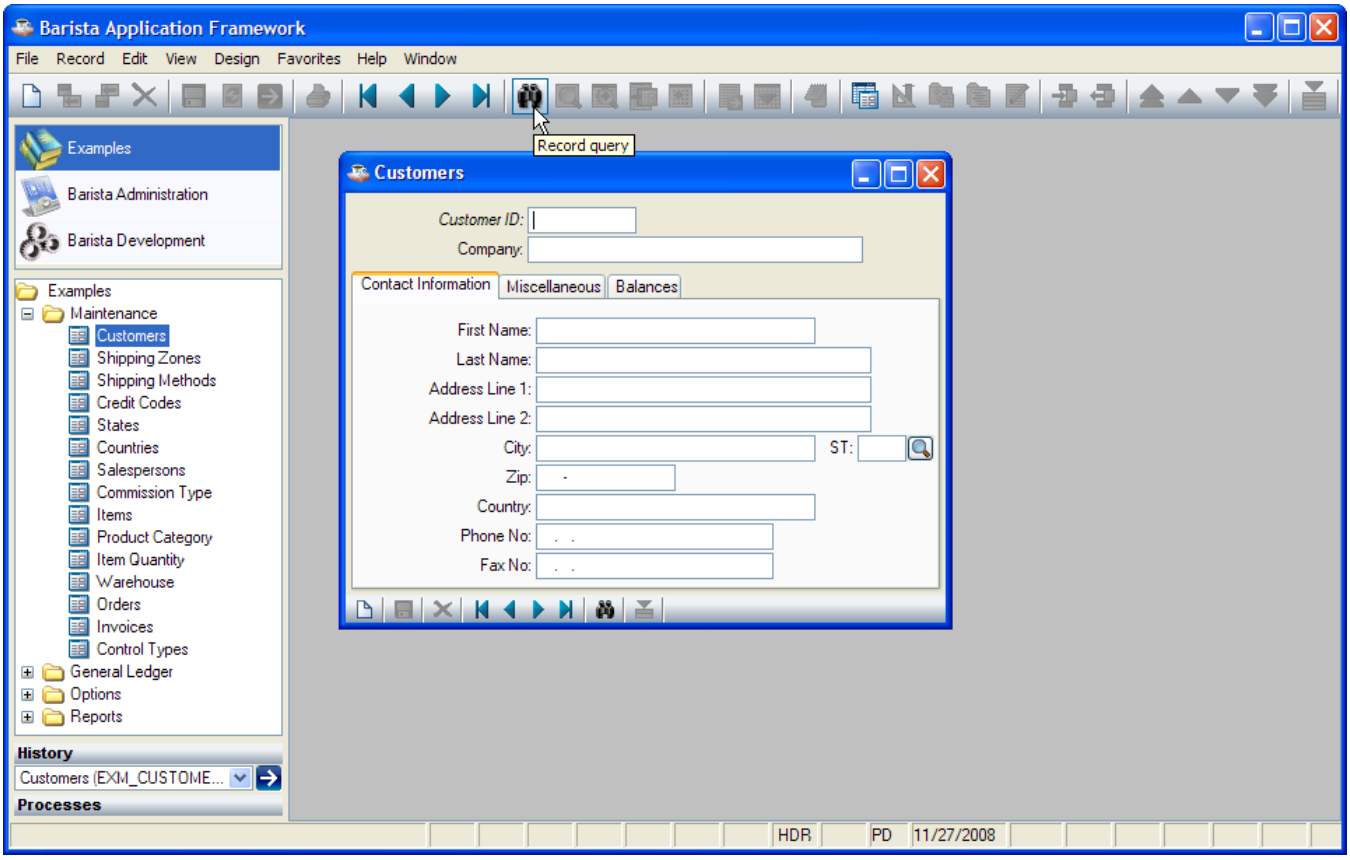
Maintenance Form

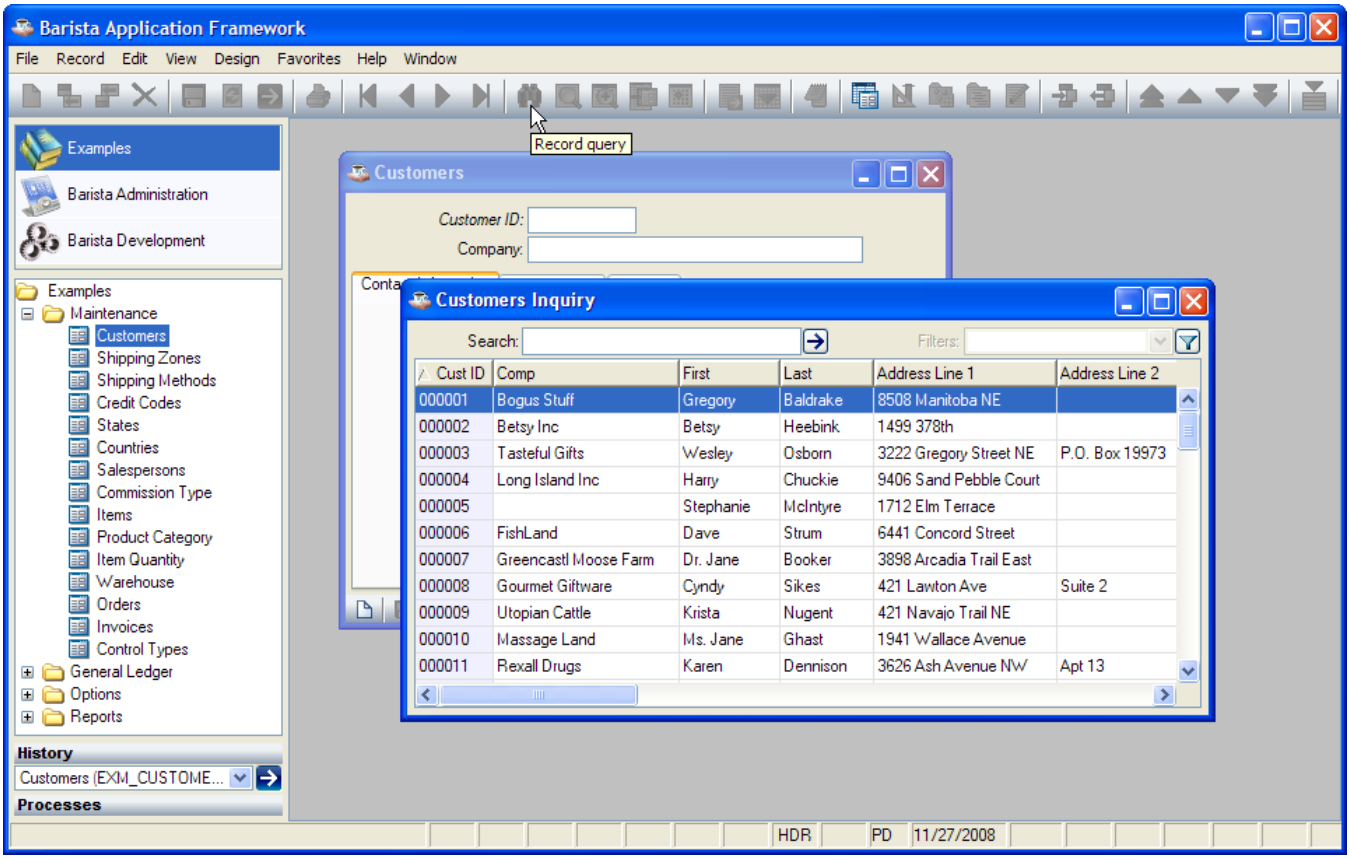
The Maintenance Form is a formatted screen for maintaining records in a table. For example:



Record Query

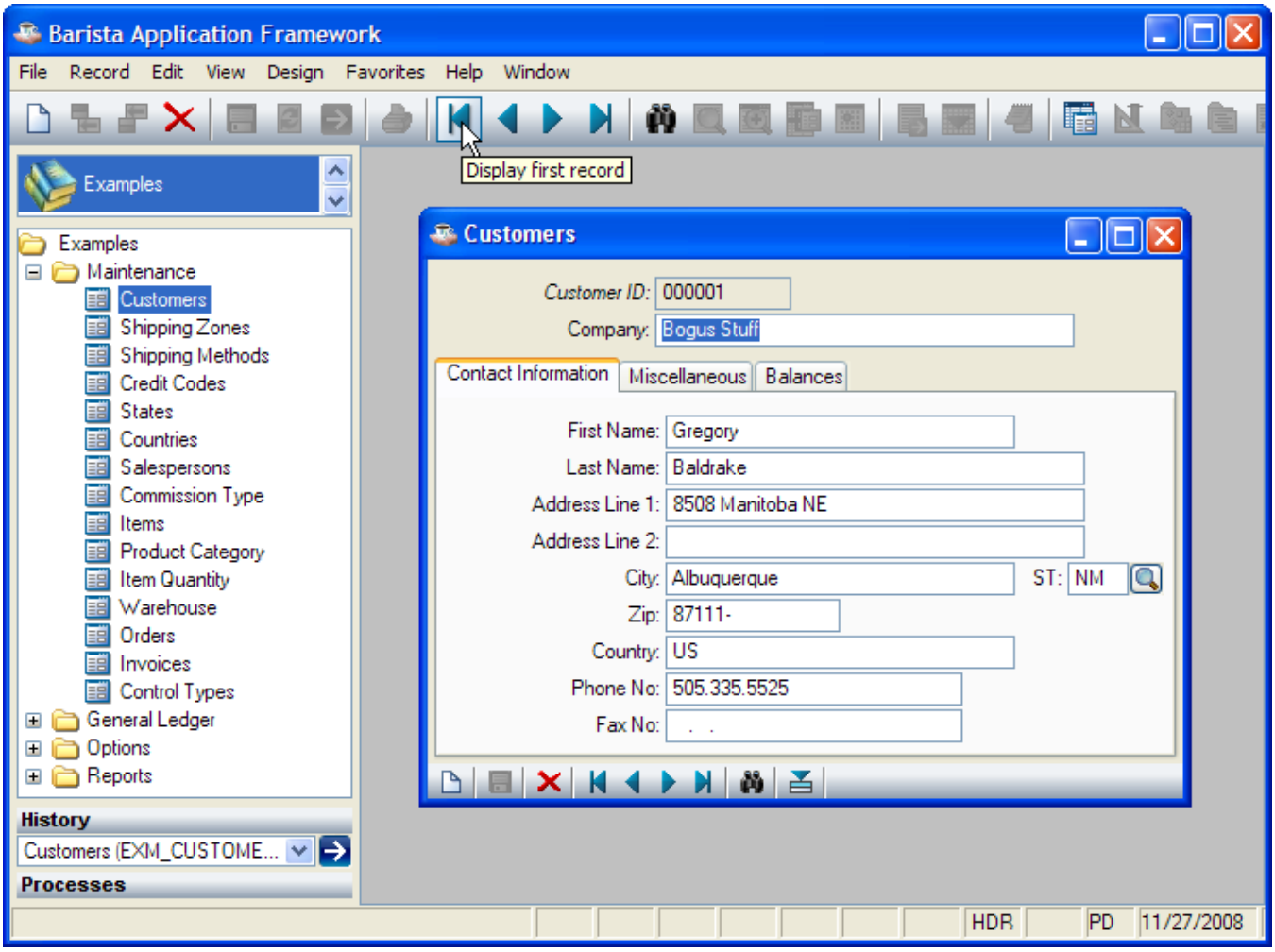
To query existing records, select View□Record Query from the menu, press [Ctrl]+Q, or press the  tool button:





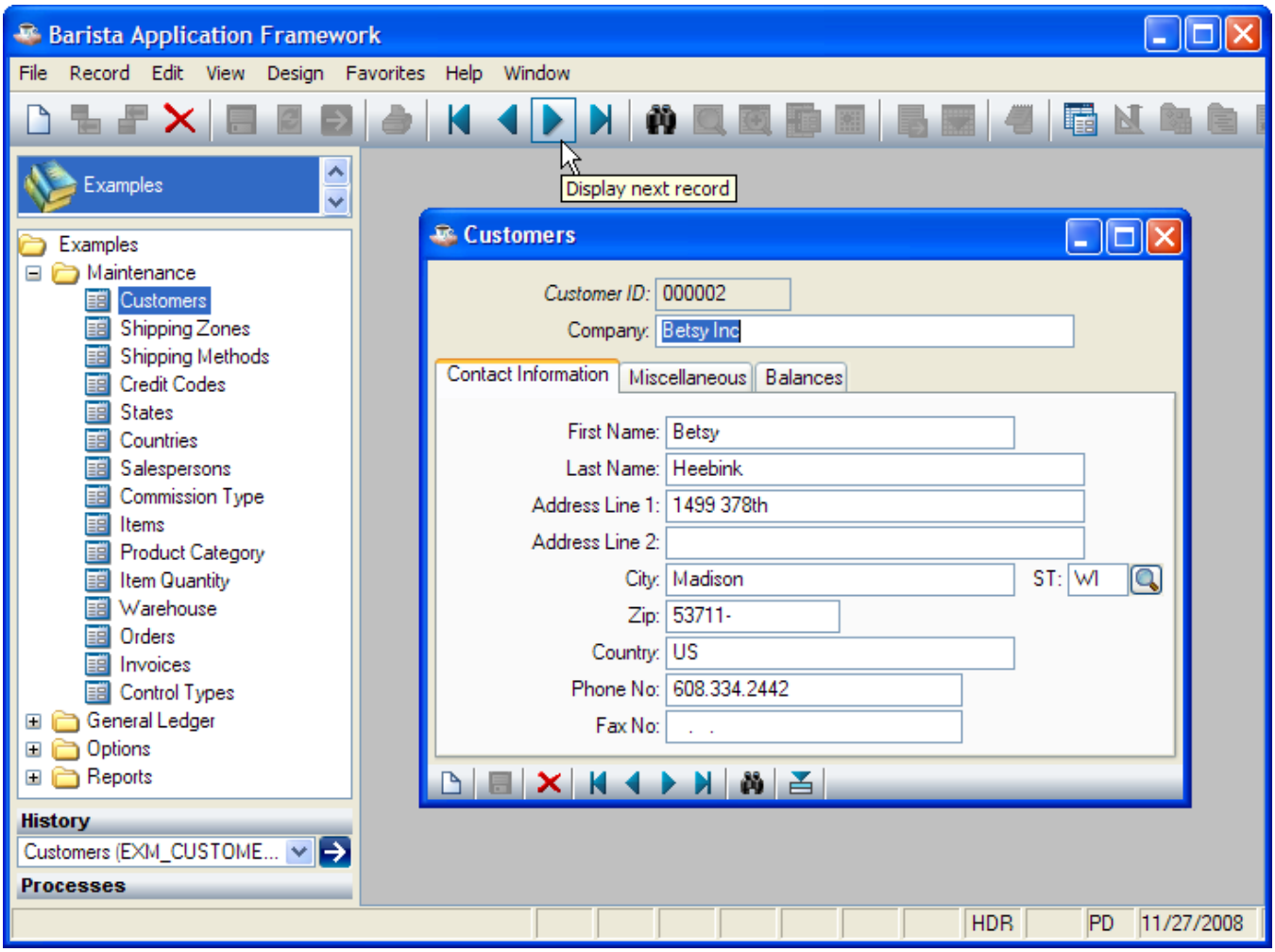
First Record

To go to the first record, select Record First Record from the menu, press [[Ctrl]]+Page Up, or press the  tool button:



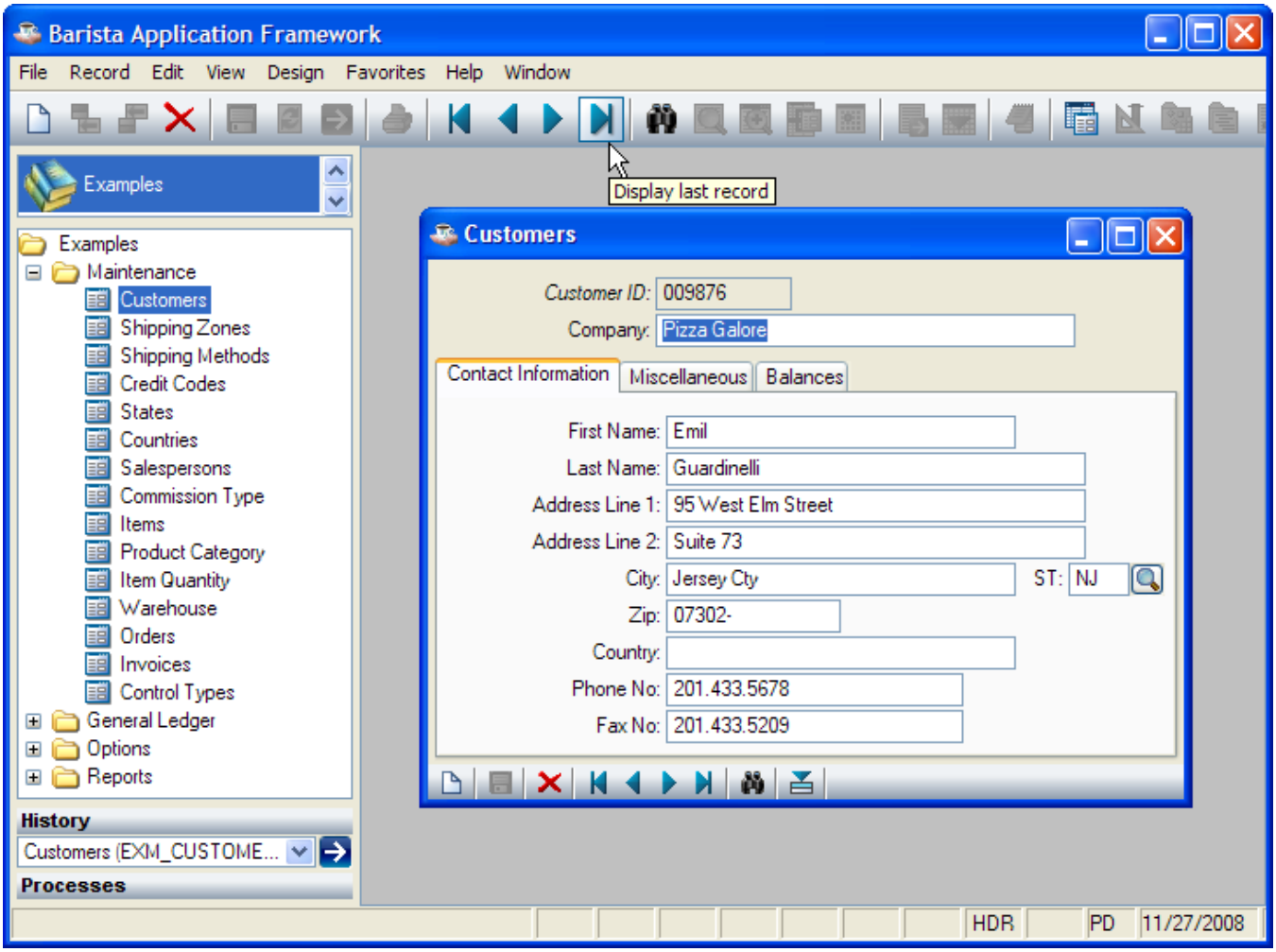
Next Record

To go to the next record, select Record□Next Record from the menu, press the [Page Down] key, or press the  tool button:



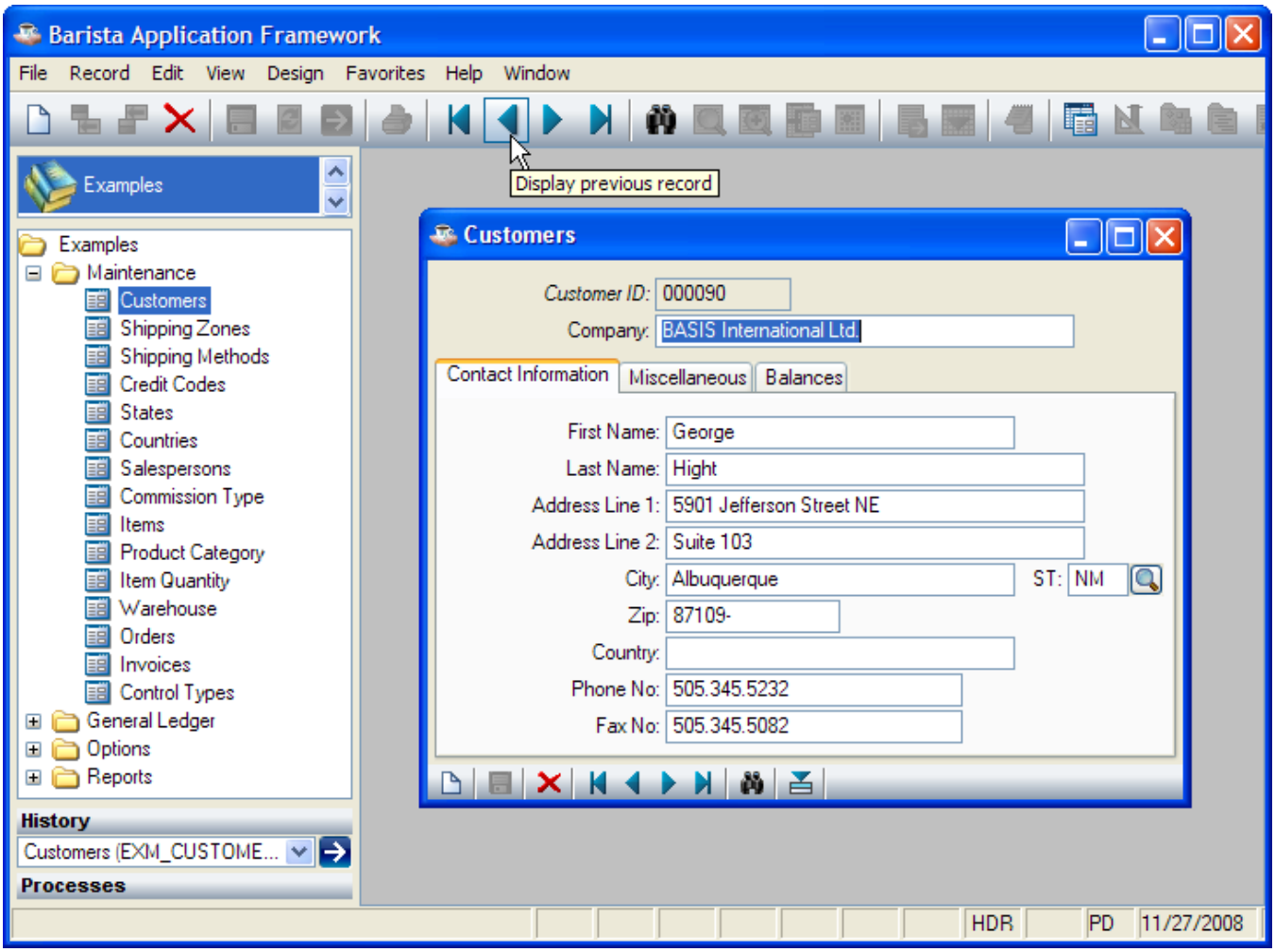
Last Record

To go to the last record, select Record□Last Record from the menu, press [Ctrl]+Page Down, or press the  tool button:



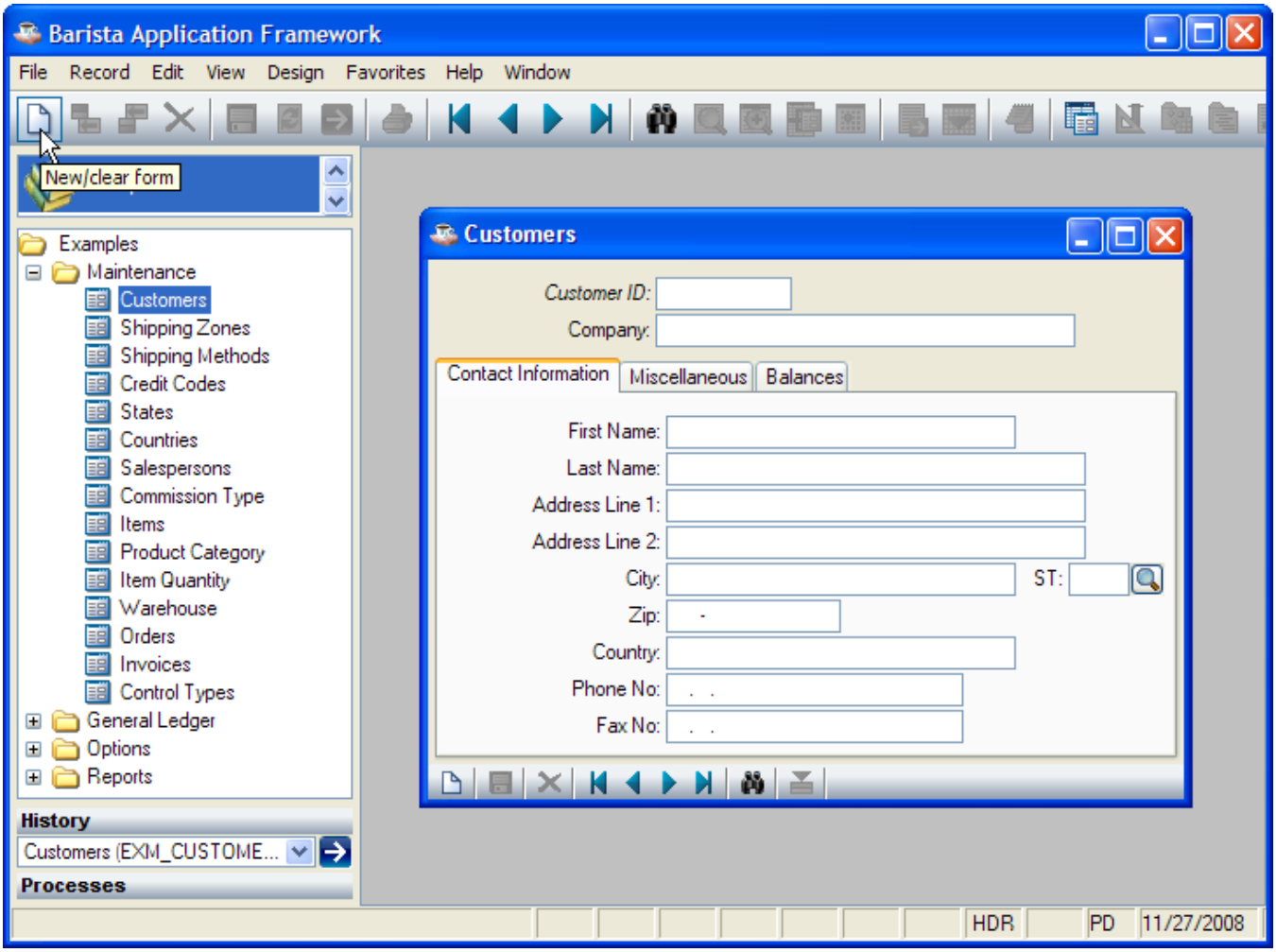
Previous Record

To go to the previous record, select Record□Previous Record from the menu, press the [Page Up] key, or press the  tool button:



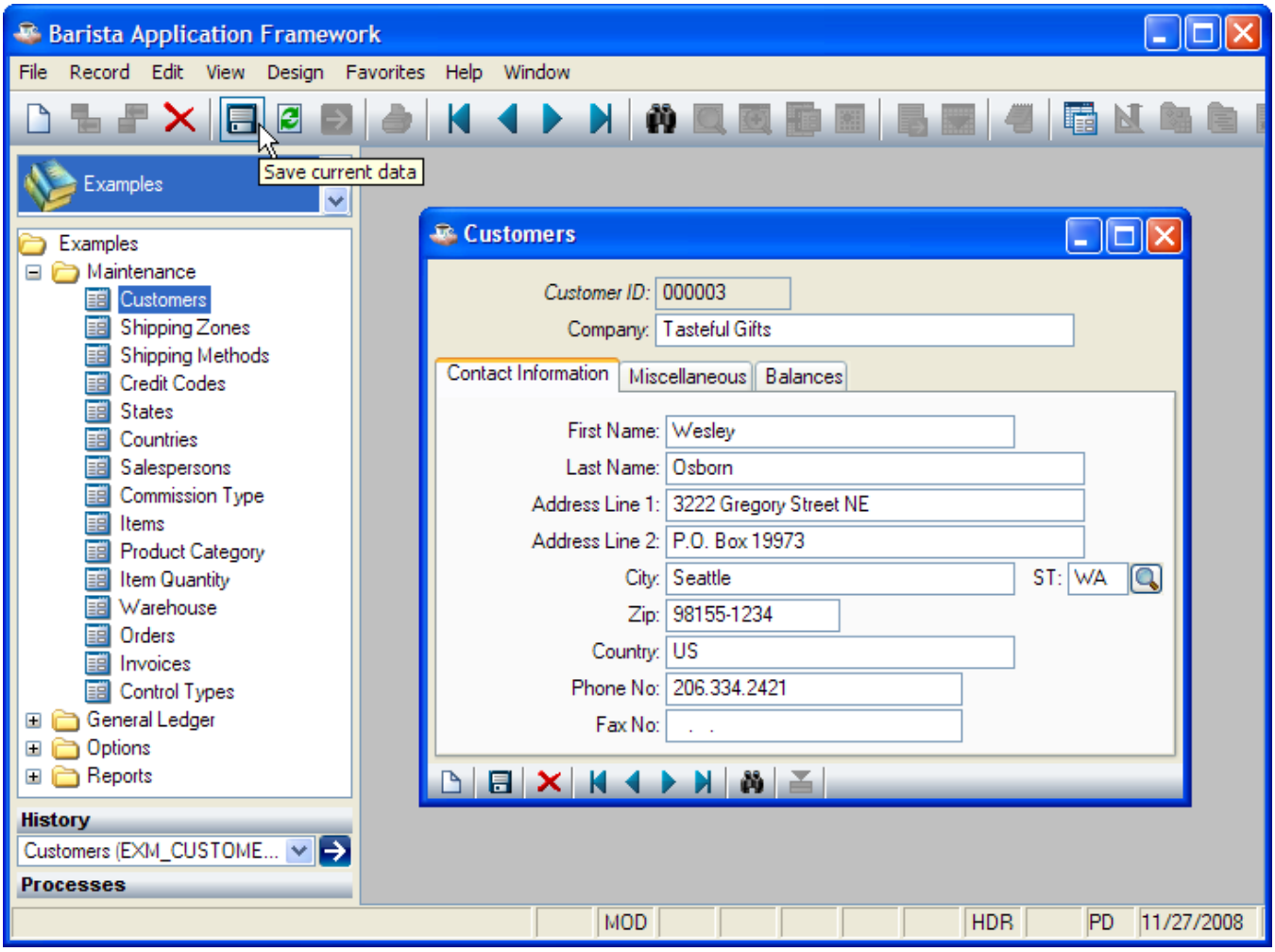
New Record

To clear the current record, select Record□New/Clear from the menu, press [Ctrl]+N, or press the  tool button:



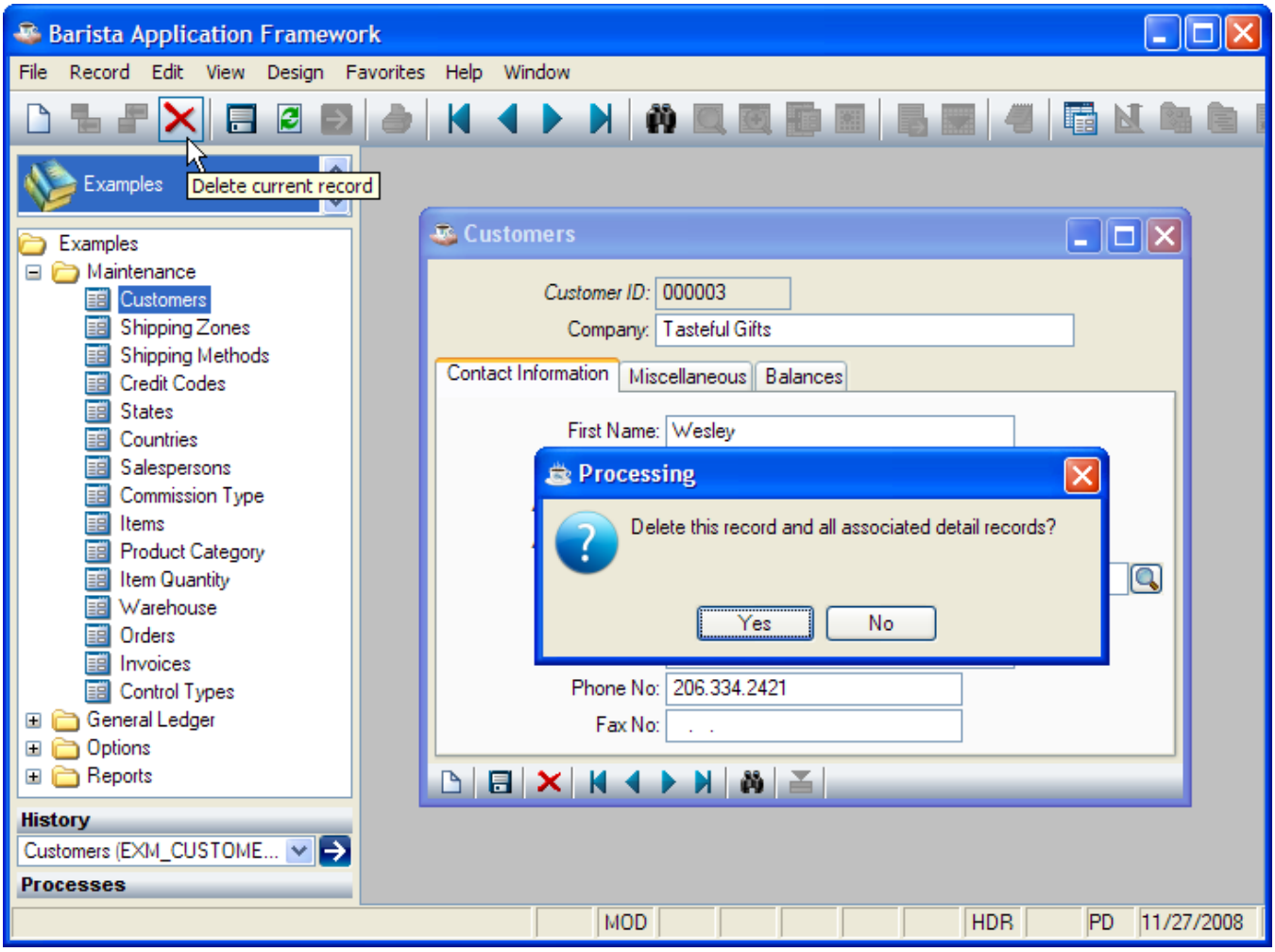
Save Record

To save the current record, select Record□Save from the menu, press [Ctrl]+S, or press the  tool button:



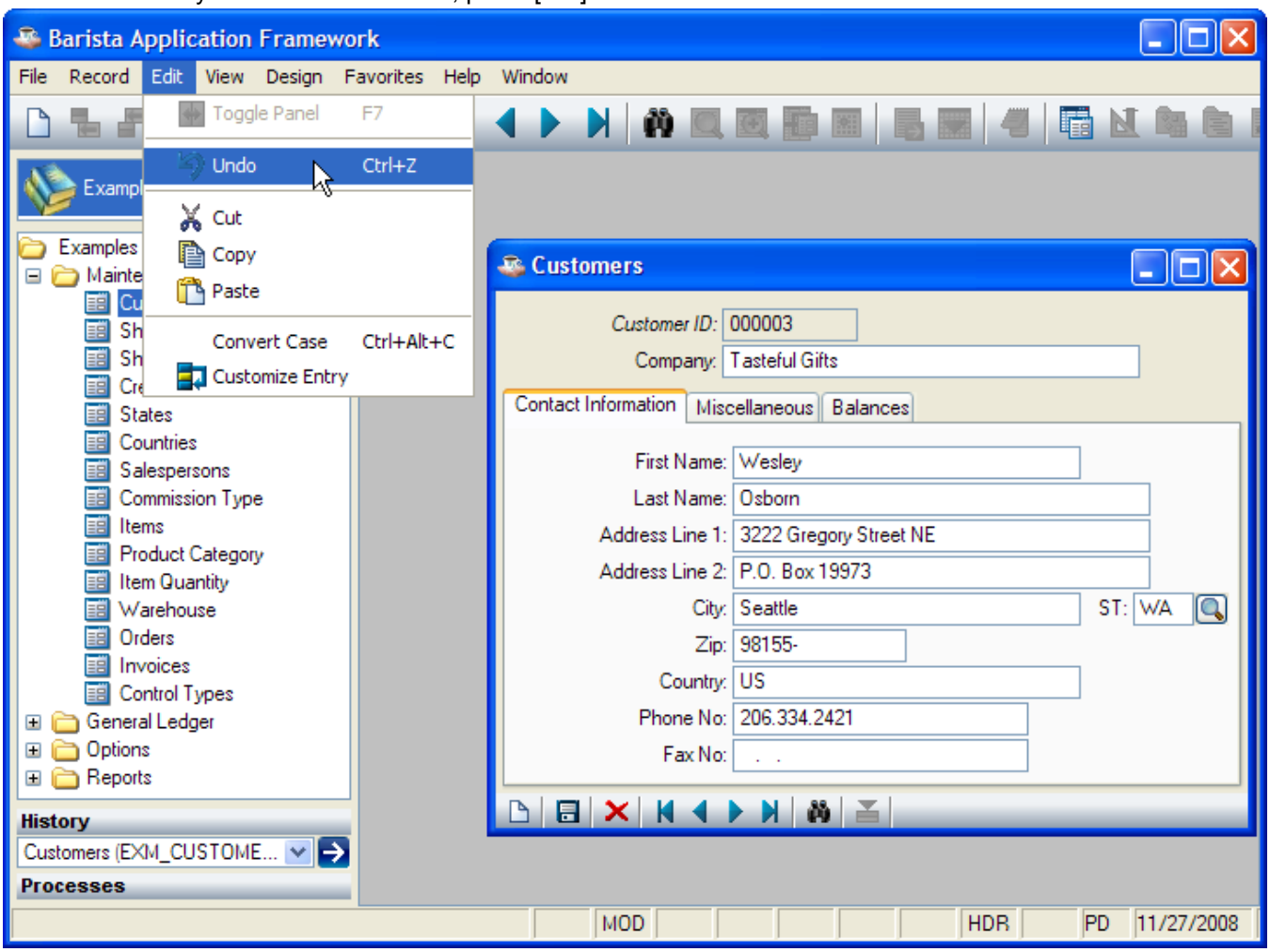
Delete Record

To delete the current record, select Record□Delete from the menu, press [Alt][Ctrl]+Delete, or press the  tool button:



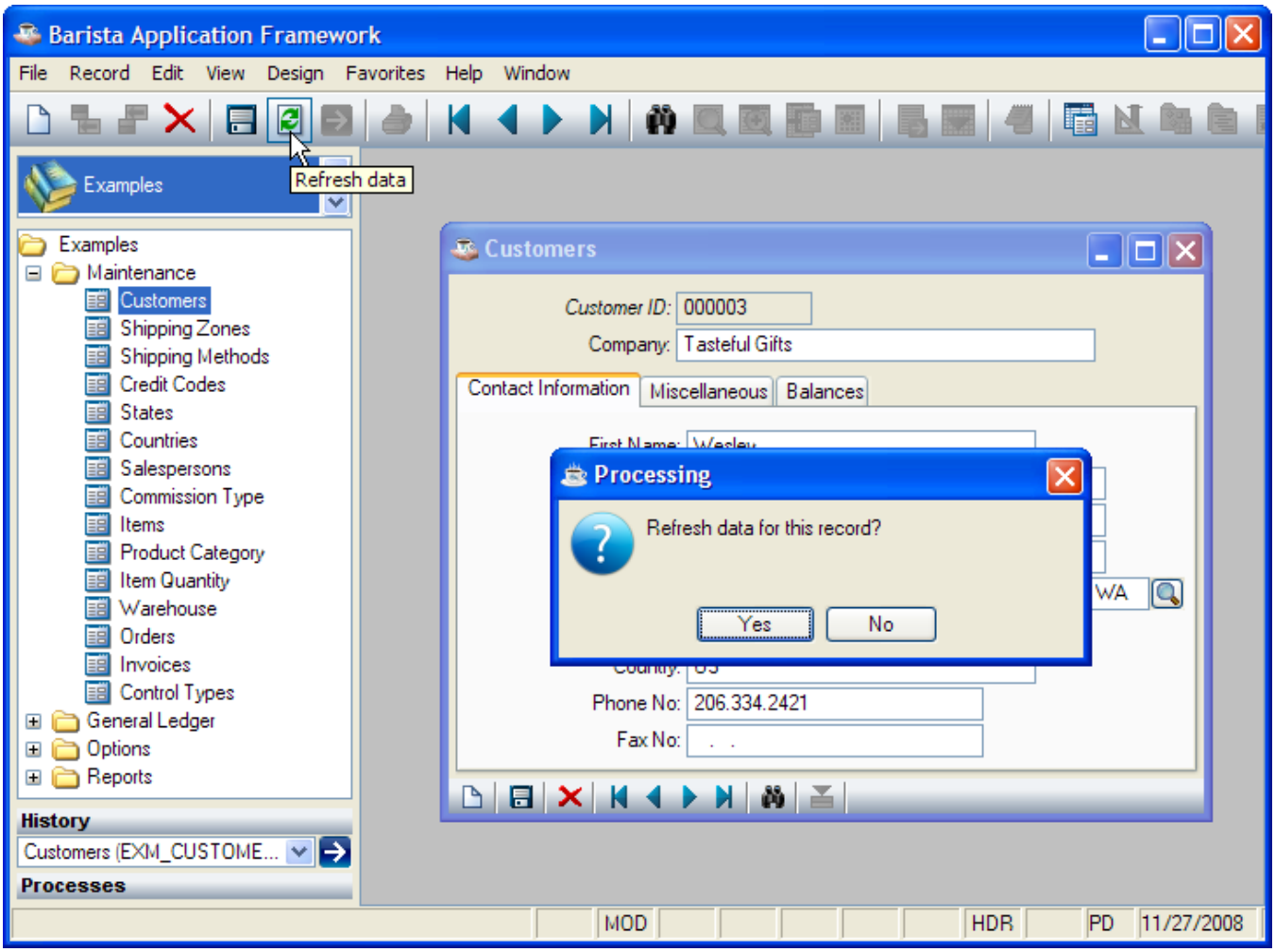
Undo Field Changes

To restore an entry field to its initial value, press [Ctrl]+Z or select Edit□Undo from the menu:



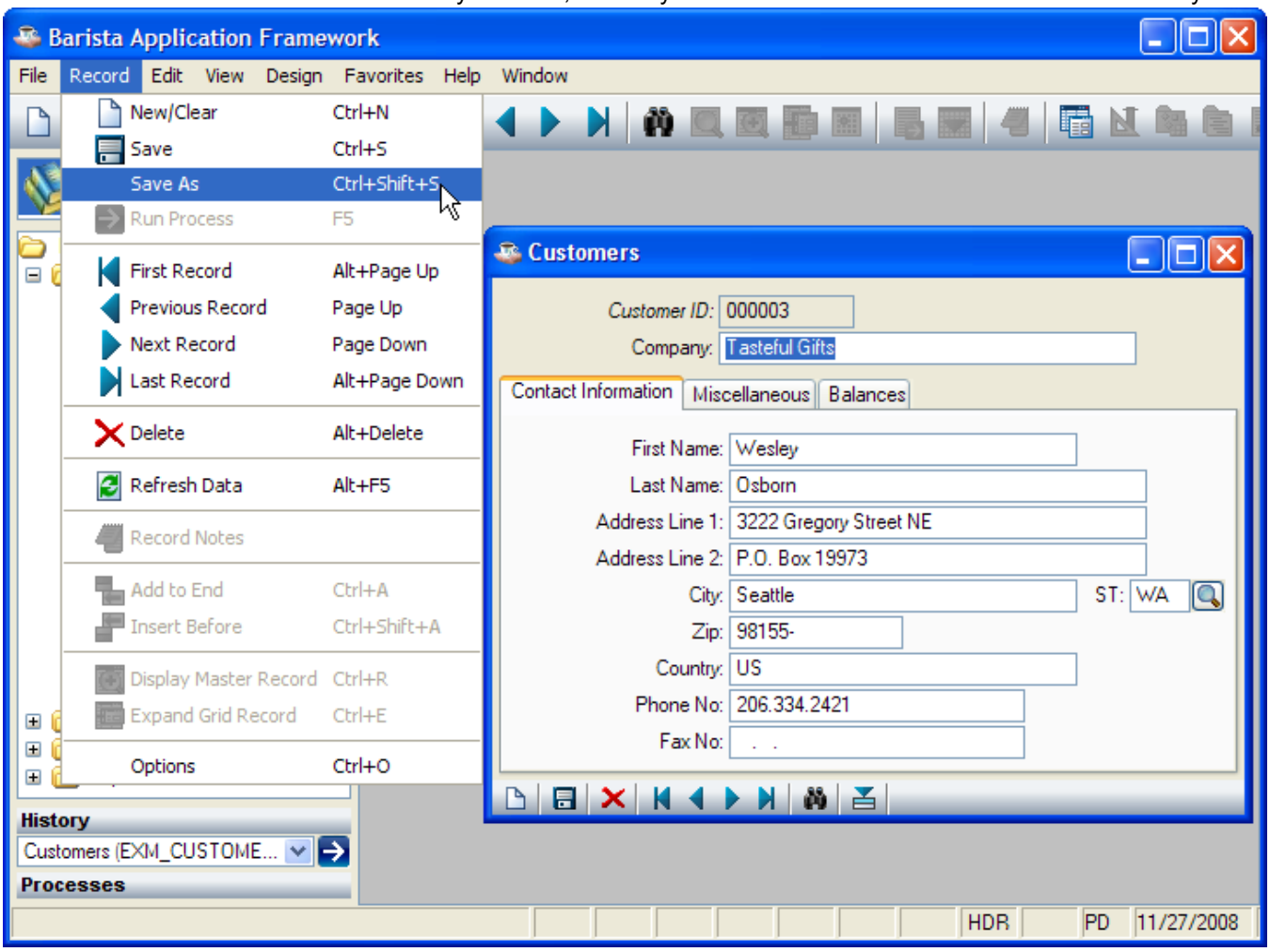
Undo Record Changes

To reload the current record from disk, press [Alt]+[F5], select Record□Refresh Data from the menu, or press the  tool button:



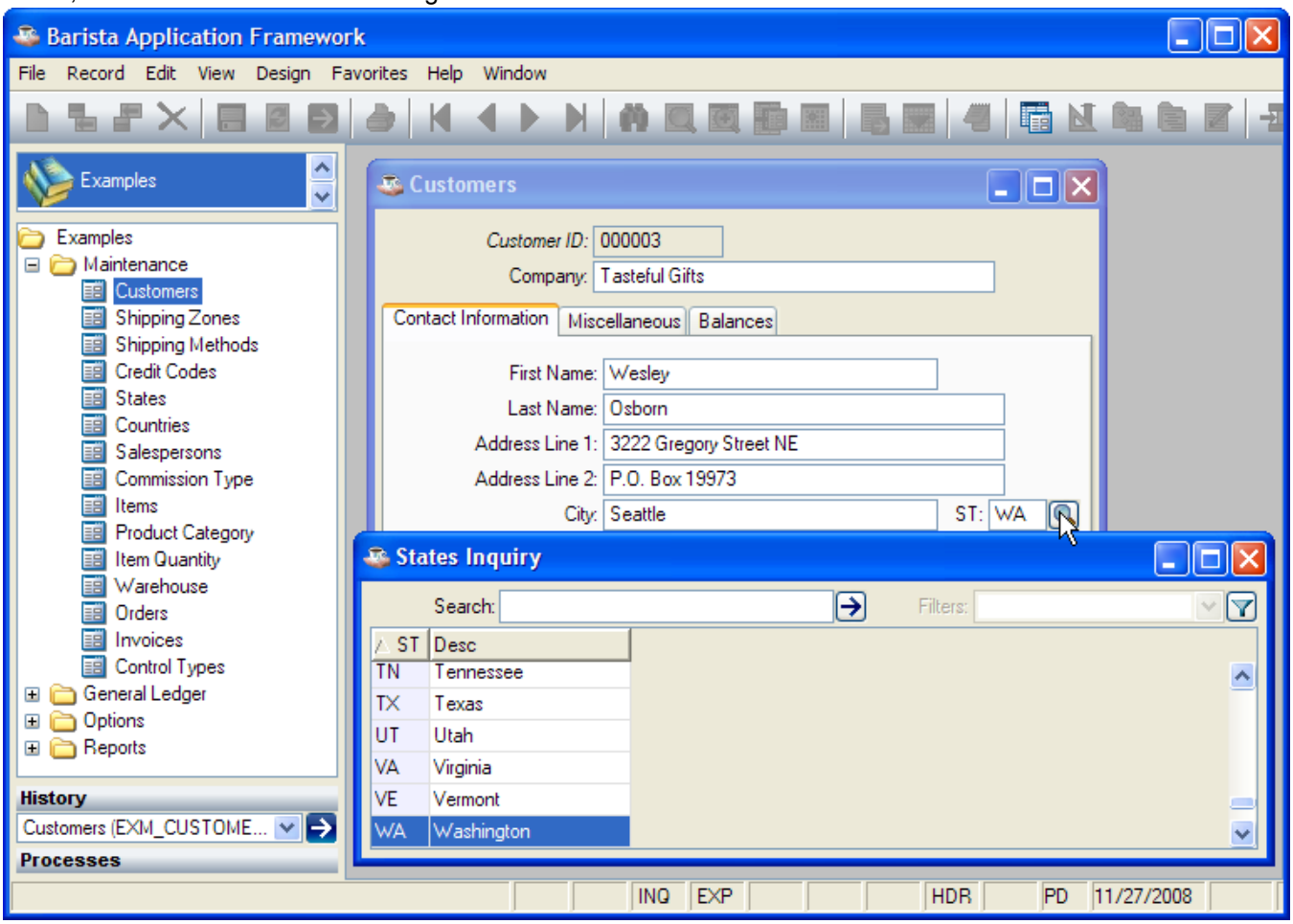
Record Save As

To save the contents of the current record to a new record key, press [Ctrl]+[[Shift]]+S or select Record□Save As from the menu. This makes the record key editable, so that you can save the current record under a new key:



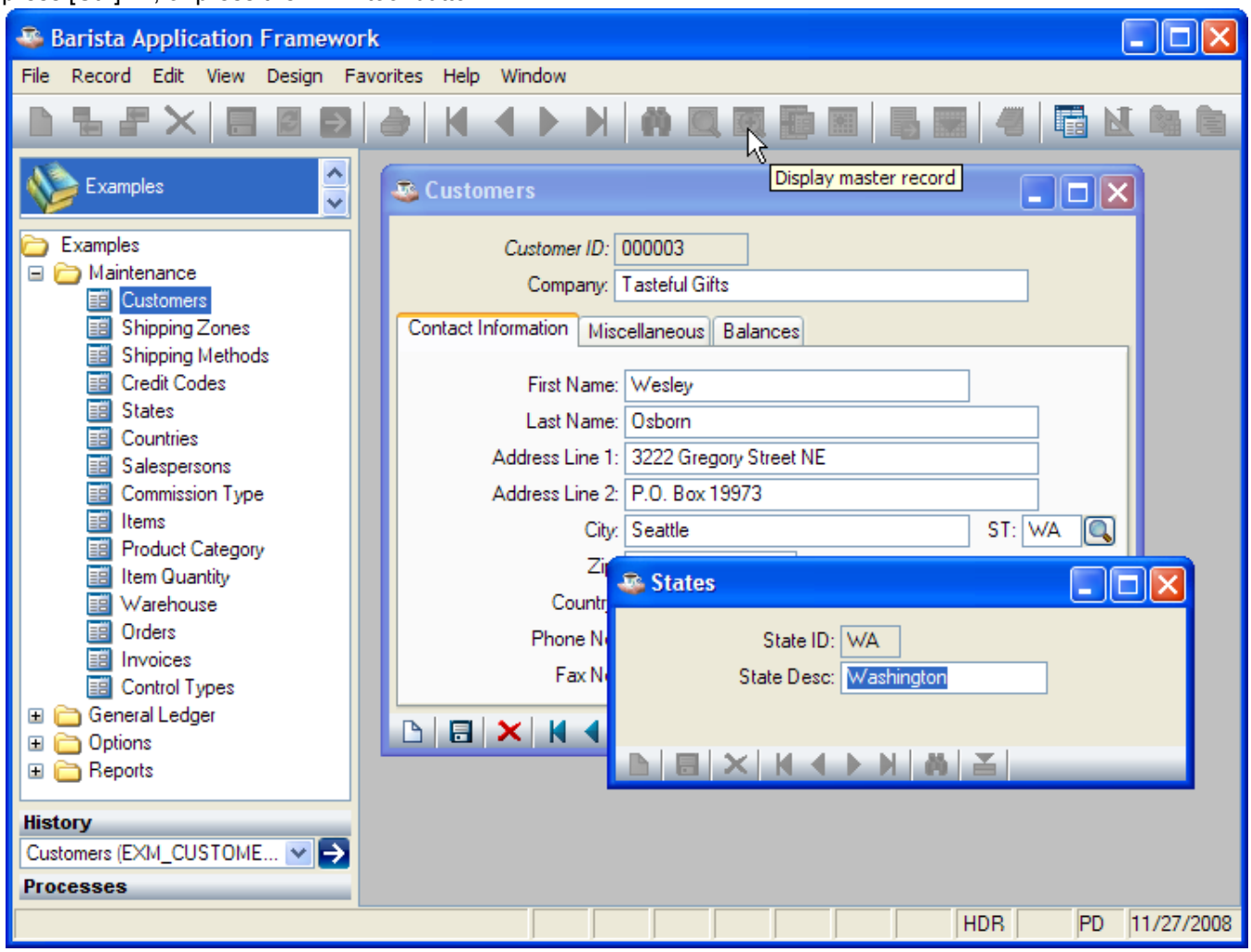
Find Field Records

To find records for a particular field, select View□Find from the menu, press [Ctrl]+F, or press the  tool button, either in the toolbar or to the right of the field:



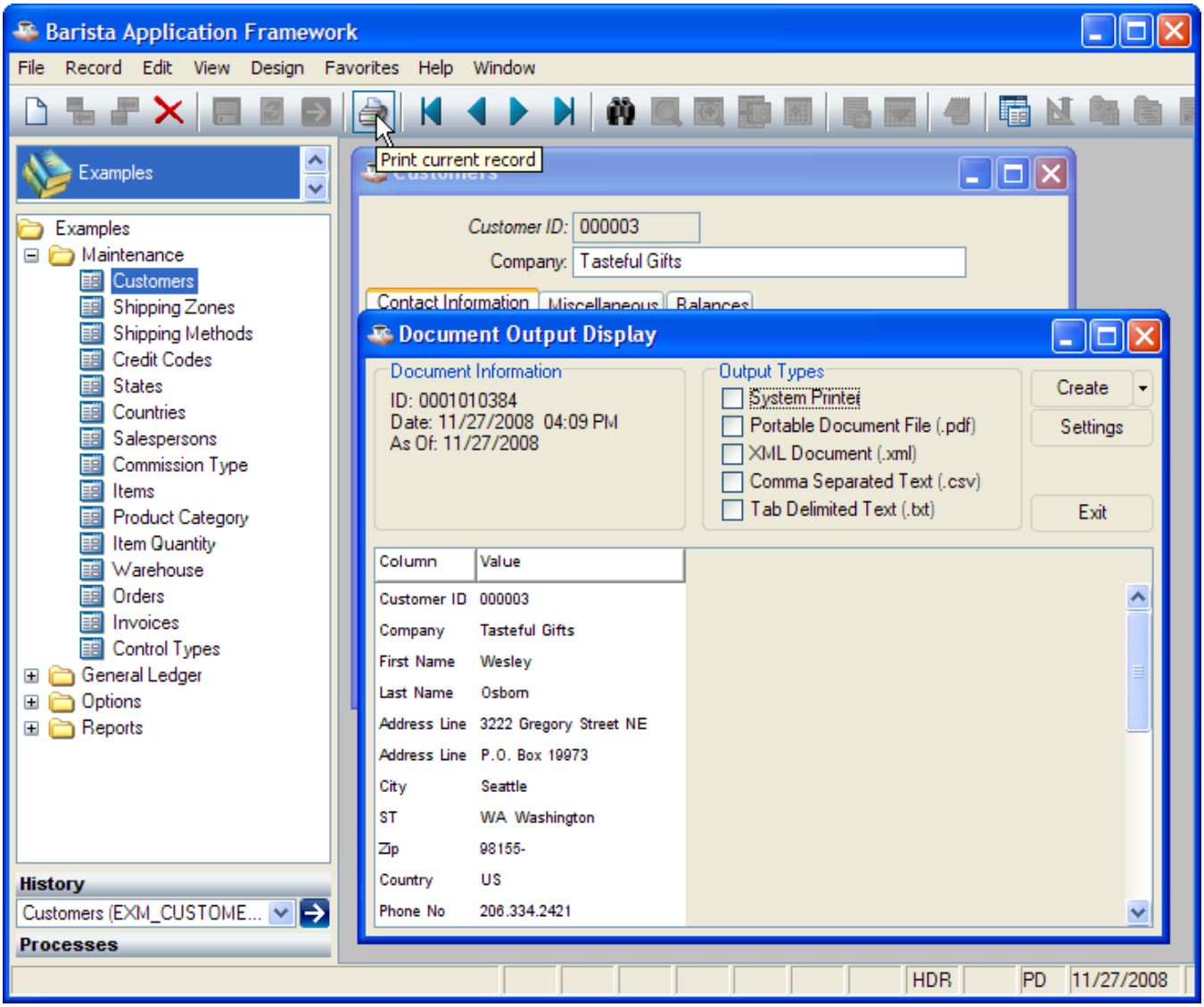
Display Master Record

To display the master record form for the current field, select Record□Display Master Record from the menu, press [Ctrl]+R, or press the  tool button:



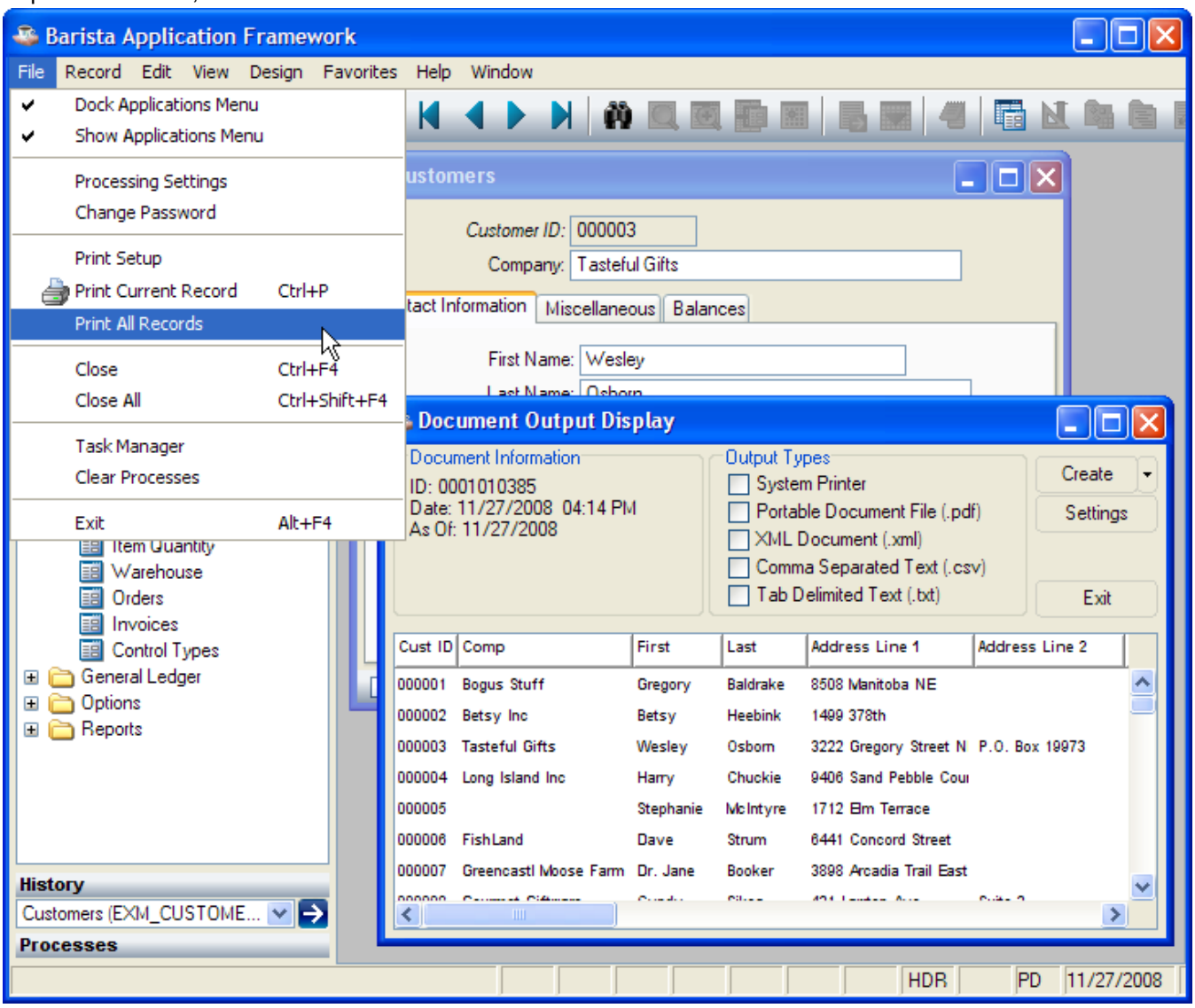
Print Record

To print the current record, select File□Print Current Record from the menu, press [Ctrl]+P, or press the  tool button:



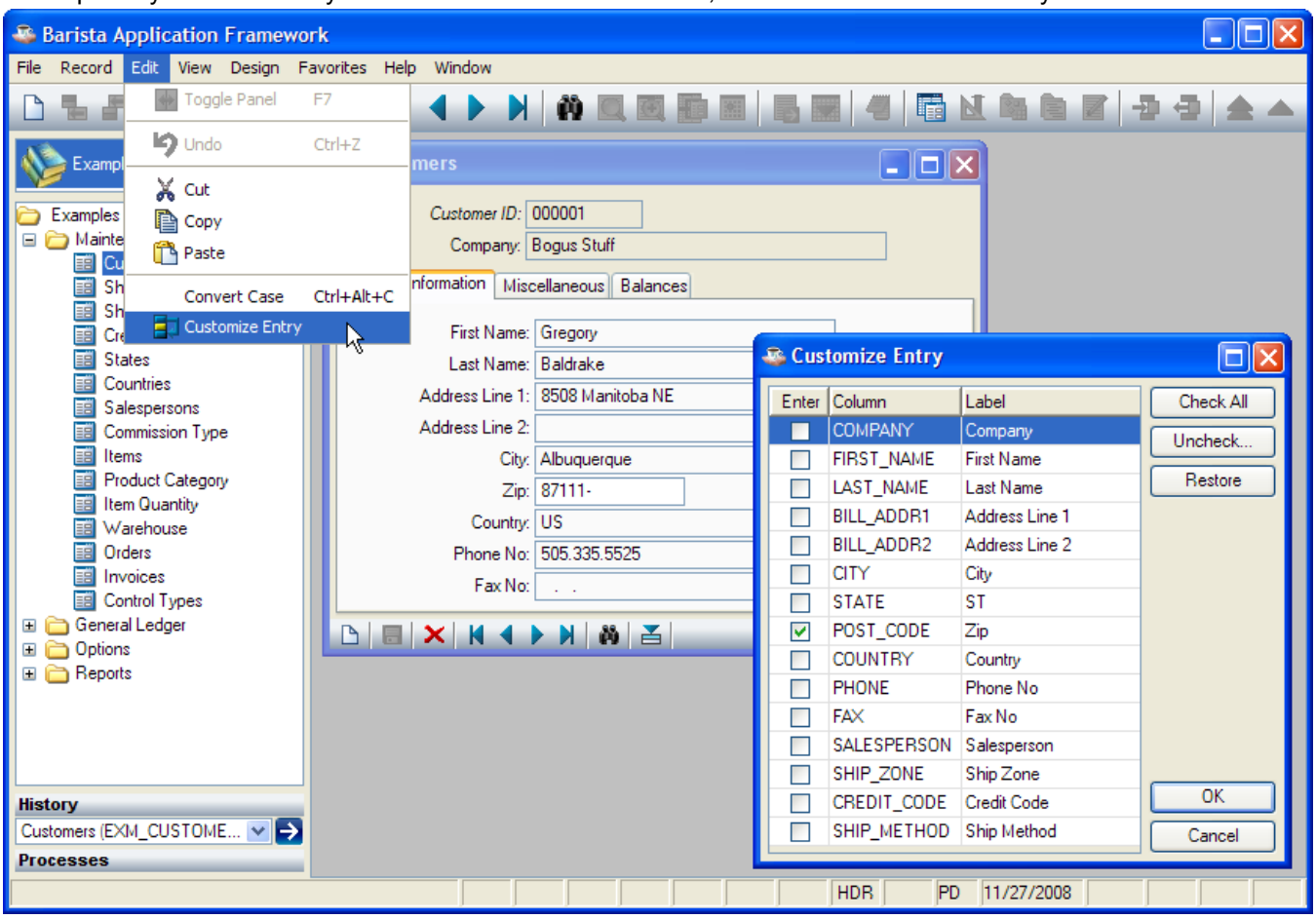
Print All Records

To print all records, select File□Print All Record from the menu:



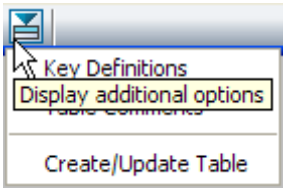
Customize Entry

To temporarily limit data entry to a subset of fields on the form, select Edit□Customize Entry from the menu:



Record Options

To access additional options (not defined for all forms), right-click in the program header area, select Record□Options from the menu, press [Ctrl]+O, or press the  tool button:

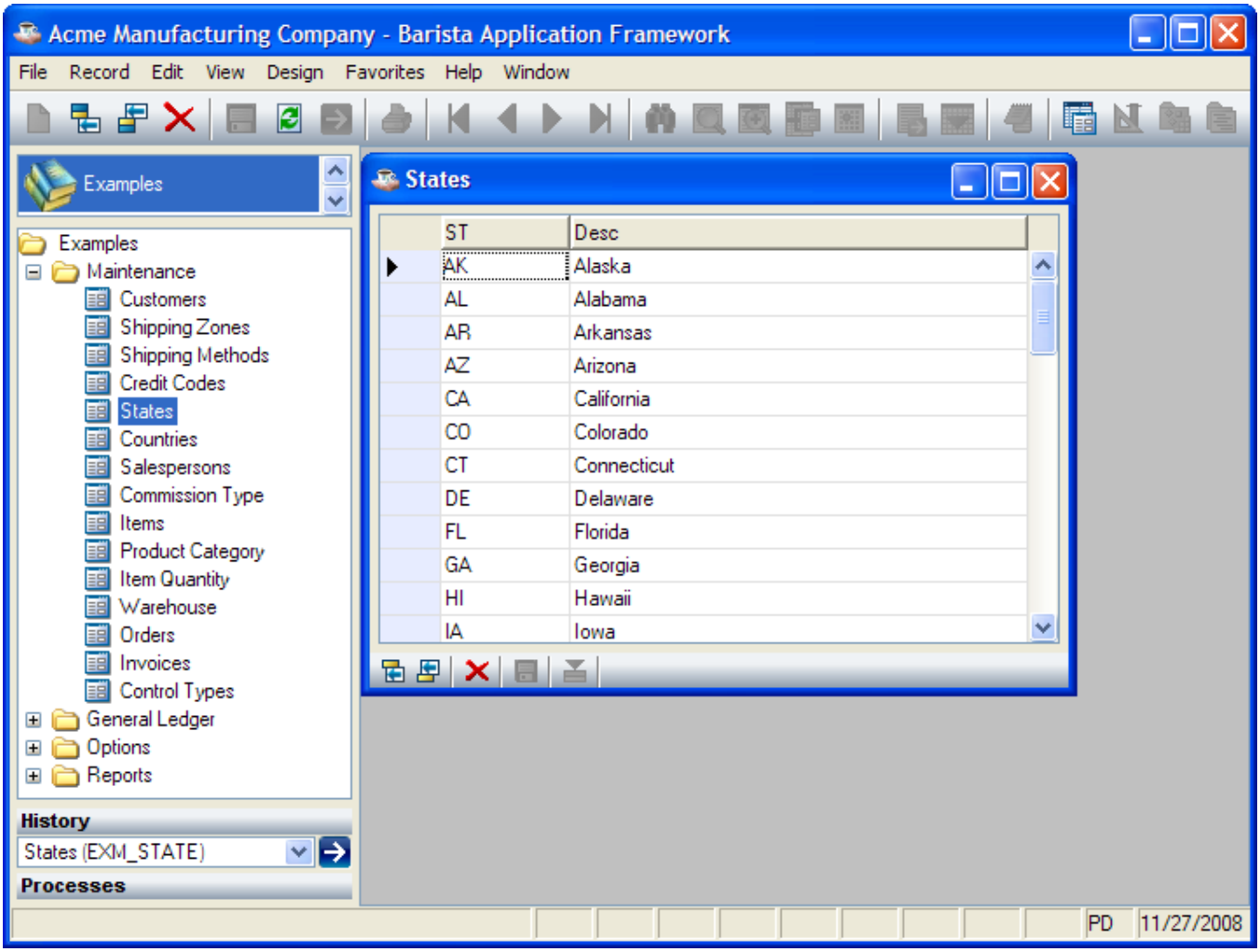


Convert Case

In addition to standard editing features ([Ctrl]+X / [Ctrl]+C / [Ctrl]+V for Cut/Copy/Paste), Barista GUI controls also implement a case conversion command. Press [Ctrl]+[Alt]+C (or select Edit□Convert Case) in any input control to cycle through UPPERCASE, lower case, and Title Case.

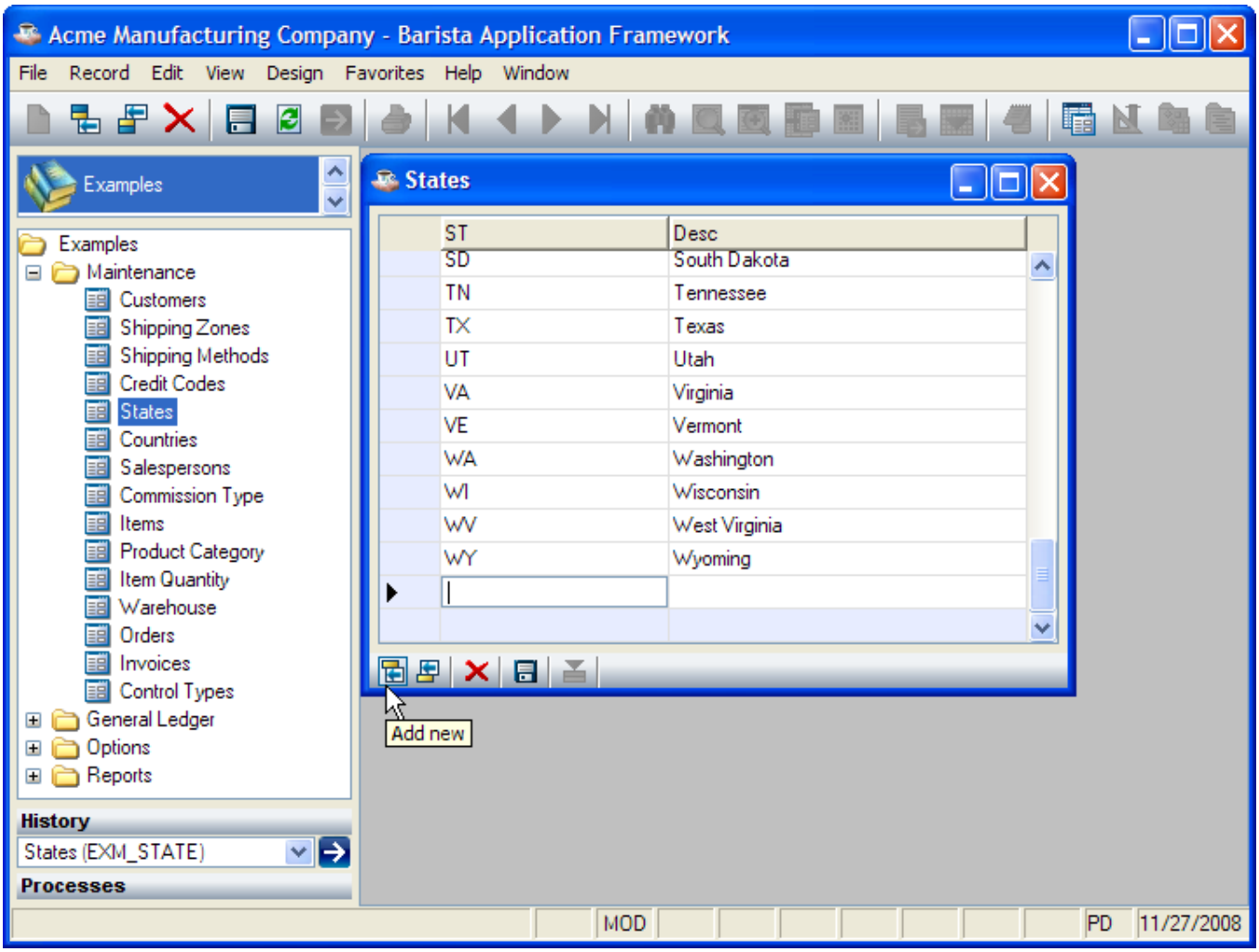
Maintenance Grid

The maintenance grid is used for editing a small set of records (generally less than a few hundred) as a group. The entire file is loaded into memory. If the record key contains a sequence number and records must be written in the entered sequence, the records are edited as a set; changes are not written to the data file until the user selects "Save current data" or closes the program. If the record key does not contain a sequence, the records are written to the data file as the user leaves each row.



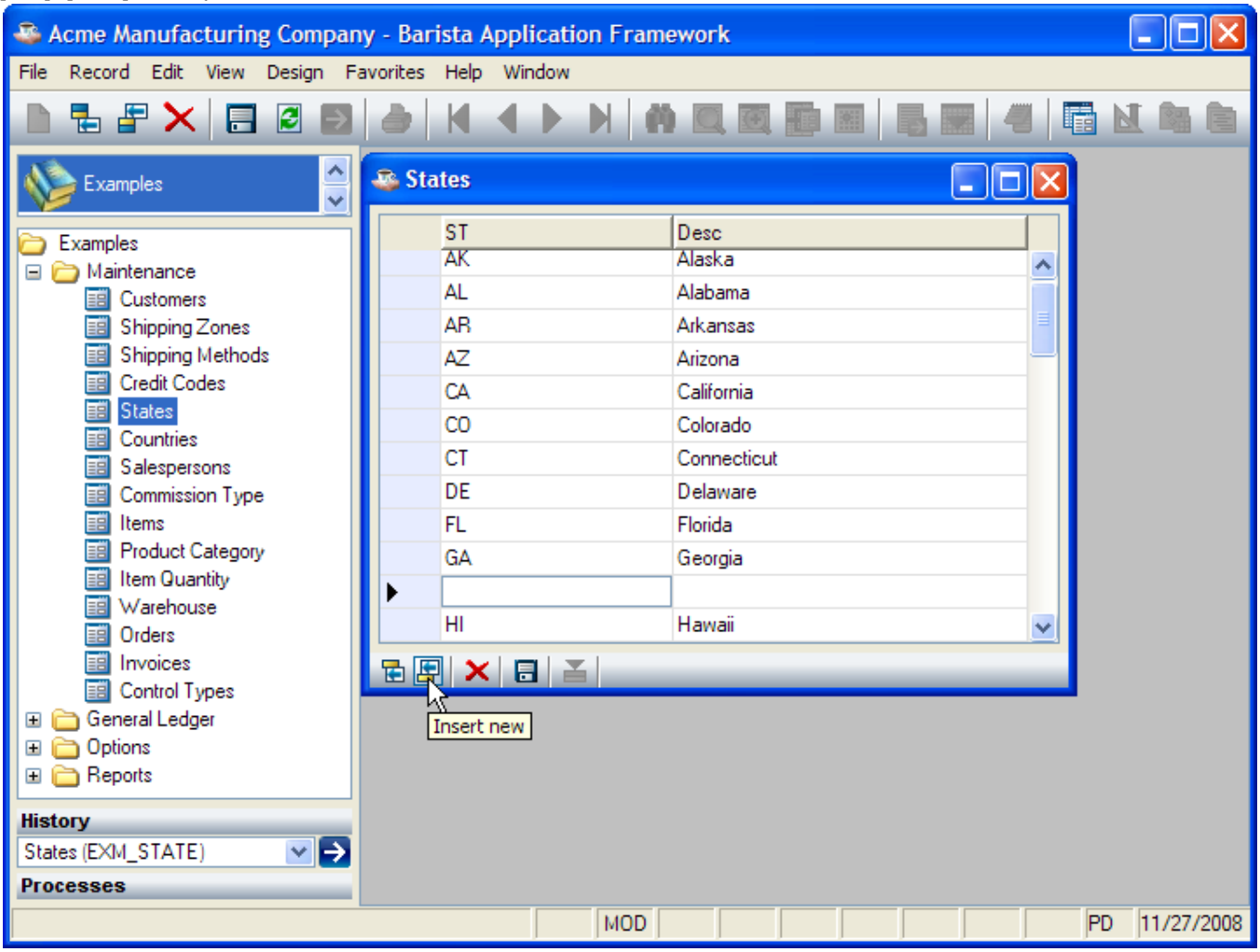
Add New

To add a new record to the end of the grid, select Record□Add New from the menu, press [Ctrl]+A, or press the  tool button:



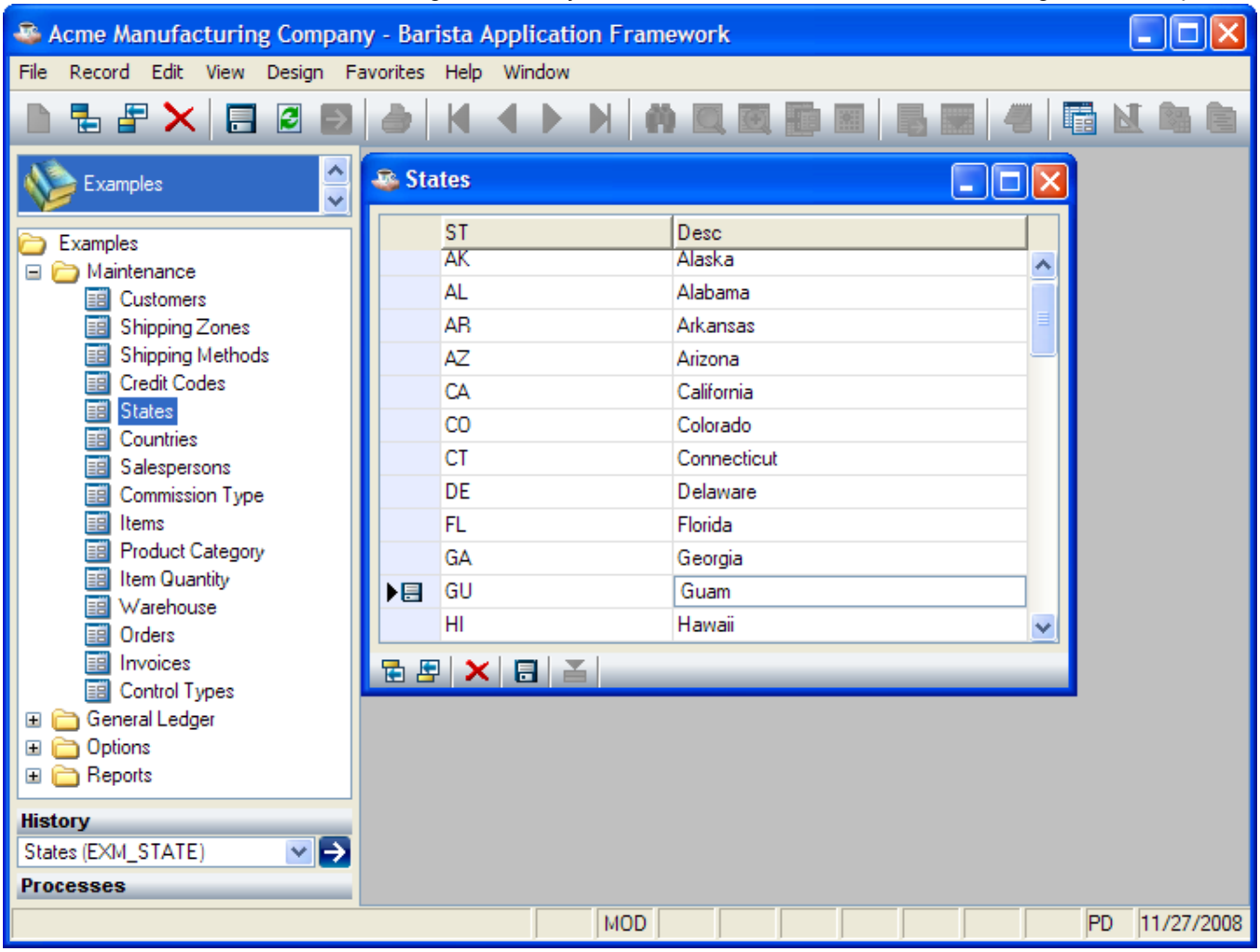
Insert New

To insert a new record before the current grid row, select Record□Insert New from the menu, press [Ctrl]+[Shift]+A, or press the  tool button:



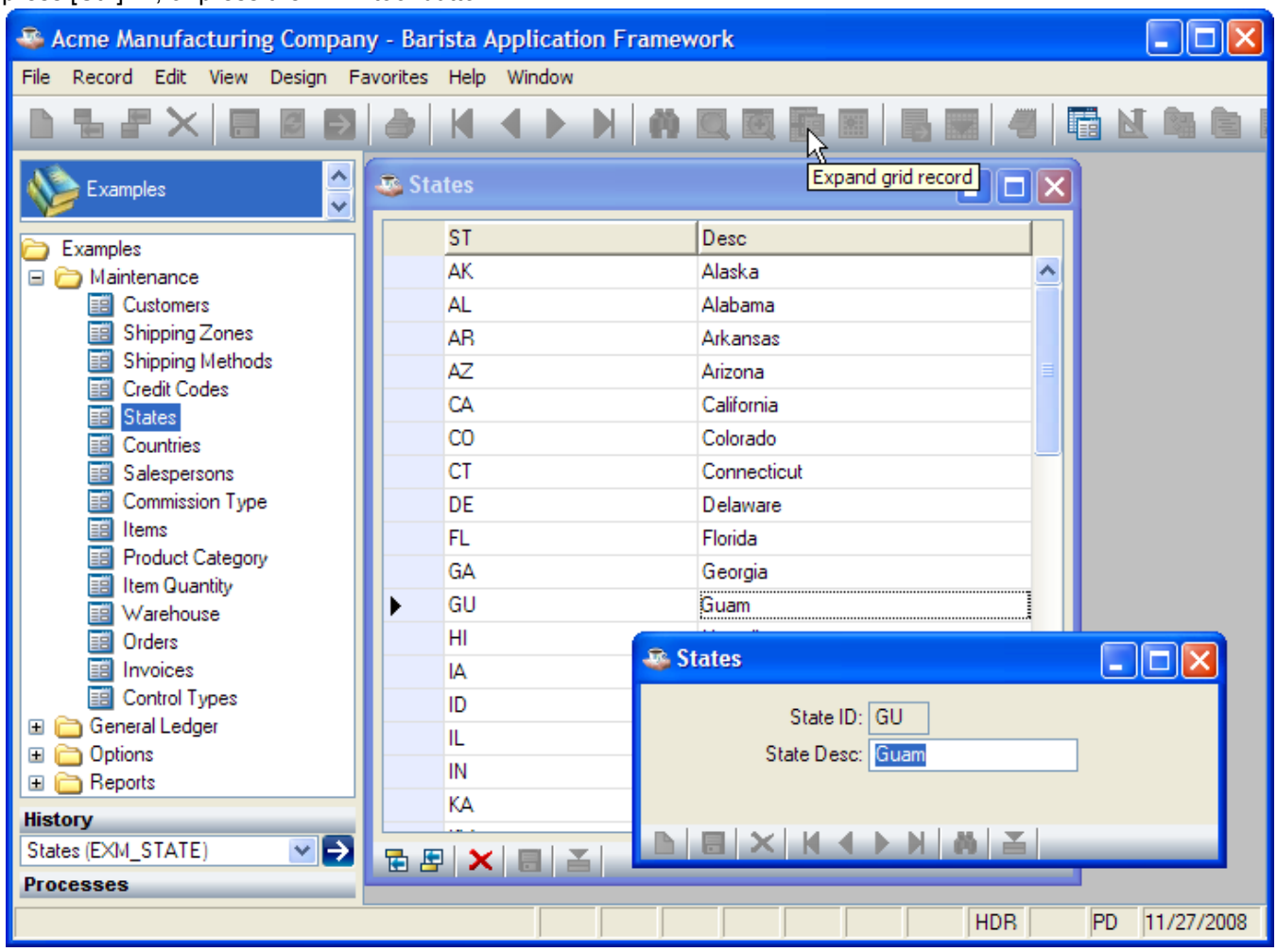
Pending Changes

Records that have been added or changed, but not yet saved to disk, show  in the left margin, for example:



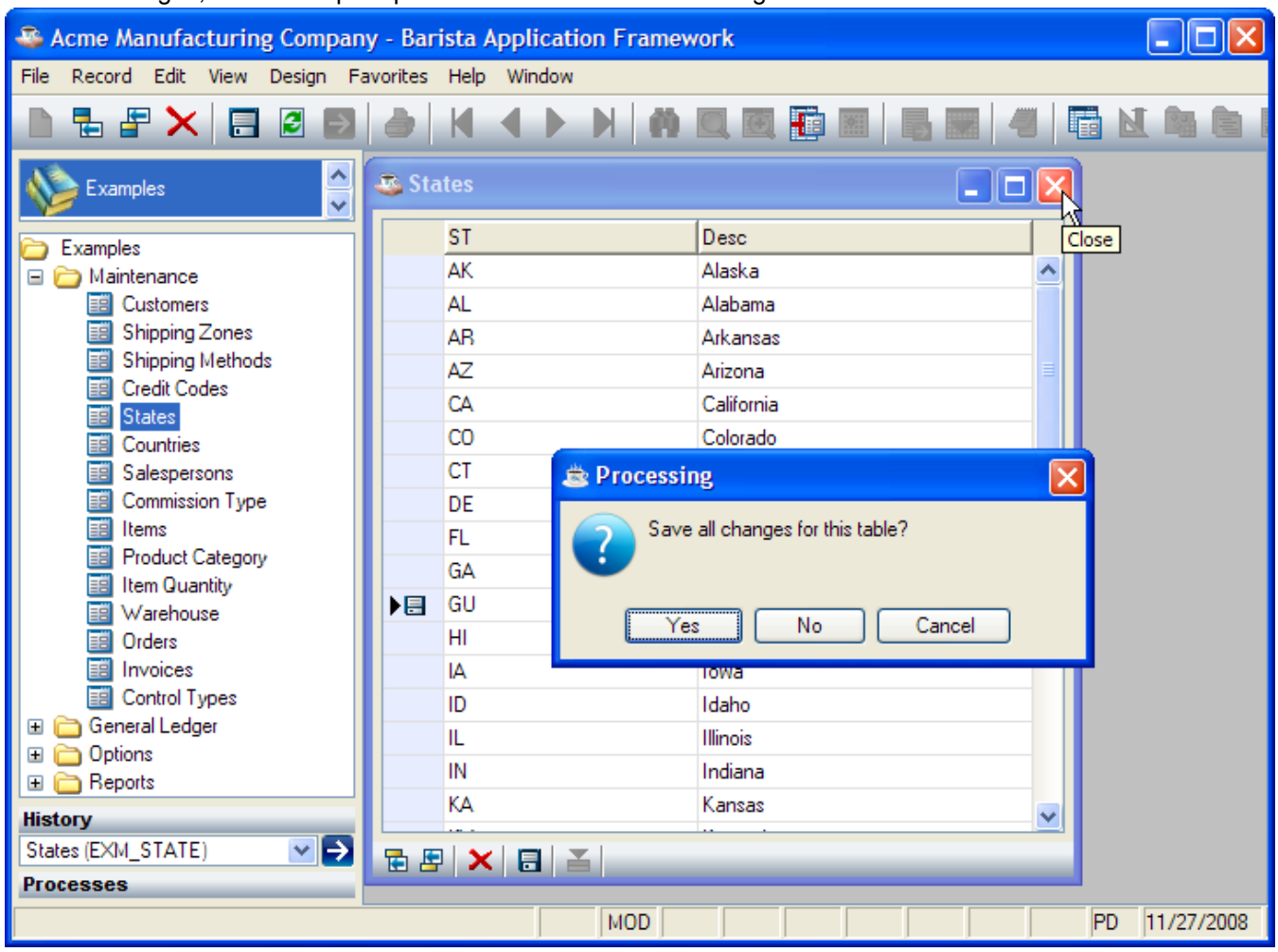
Expand Grid Record

To view or edit the current record in a Maintenance Form, select Record□Expand Grid Record from the menu, press [Ctrl]+E, or press the  tool button:



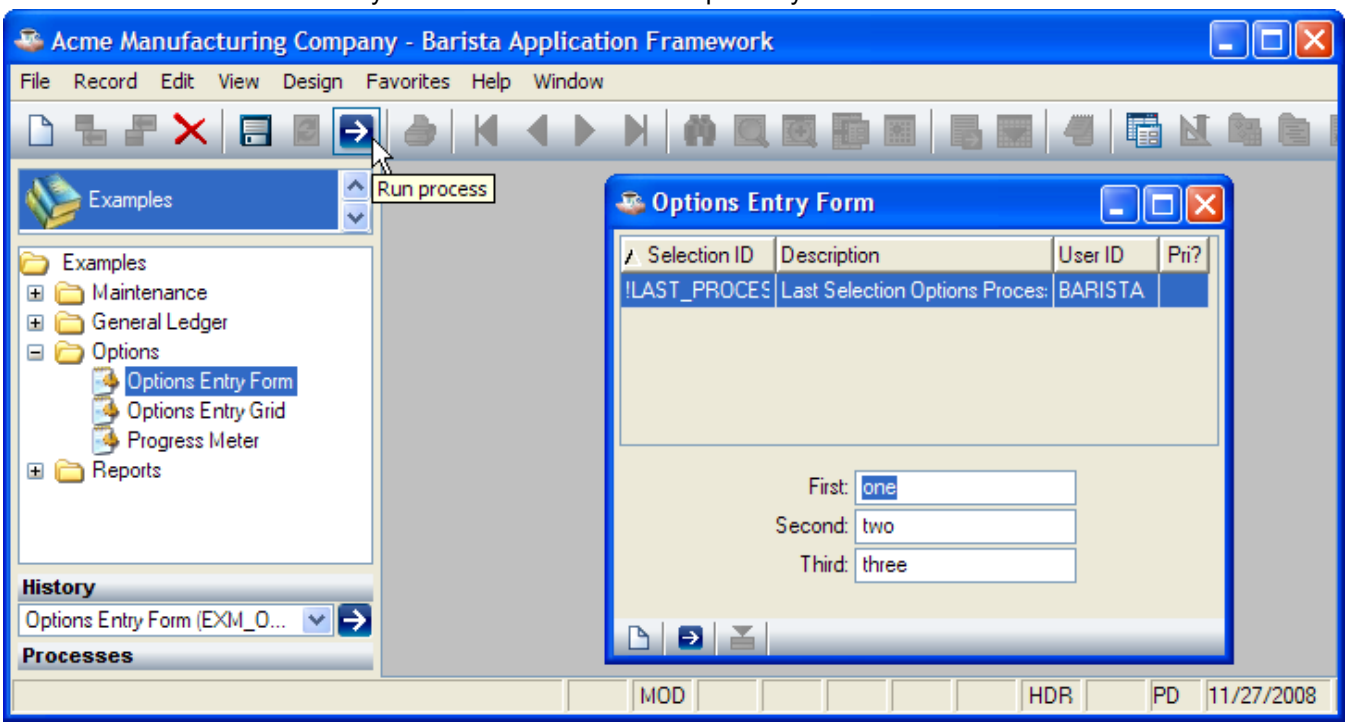
Closing a Grid Maintenance Program

To close the program, select File□Close from the menu, press [Ctrl]+[F4], or click the close box. If there are any unsaved changes, the user is prompted to save or discard the changes:



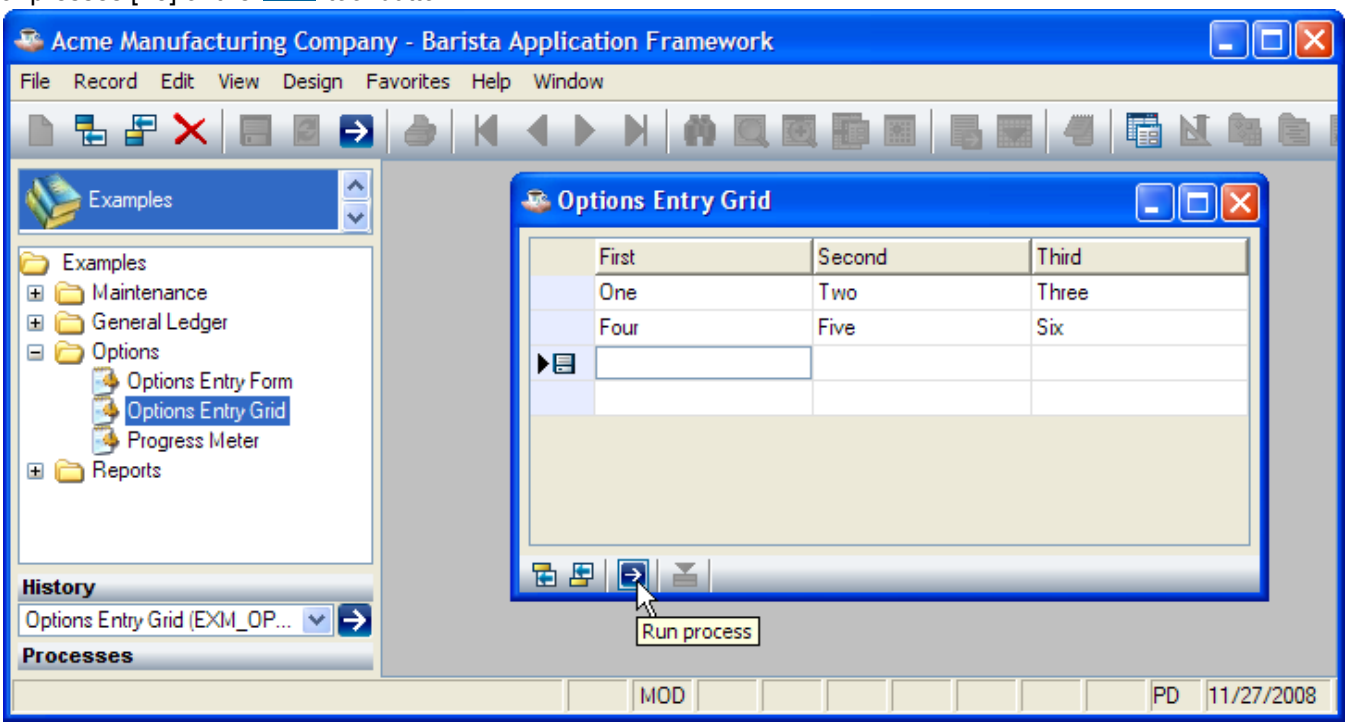
Options Entry Form

Options Entry forms prompt the user for a set of data. When the user selects Record□Run Process or presses [F5] or the  tool button, Barista passes that data on to an overlay program (usually a report or batch update). It also saves the data entered by the user so that it can be optionally reused the next time the form is used.



Options Entry Grid

The Options Entry Grid is used to prompt the user for multiple rows of data. As with the Options Entry Form, that data is passed on to a report or batch update program for processing when the user selects Record□Run Process or presses [F5] or the  tool button.



Building Applications With Barista

Follow these steps to build a Barista form and add it to the menu system:

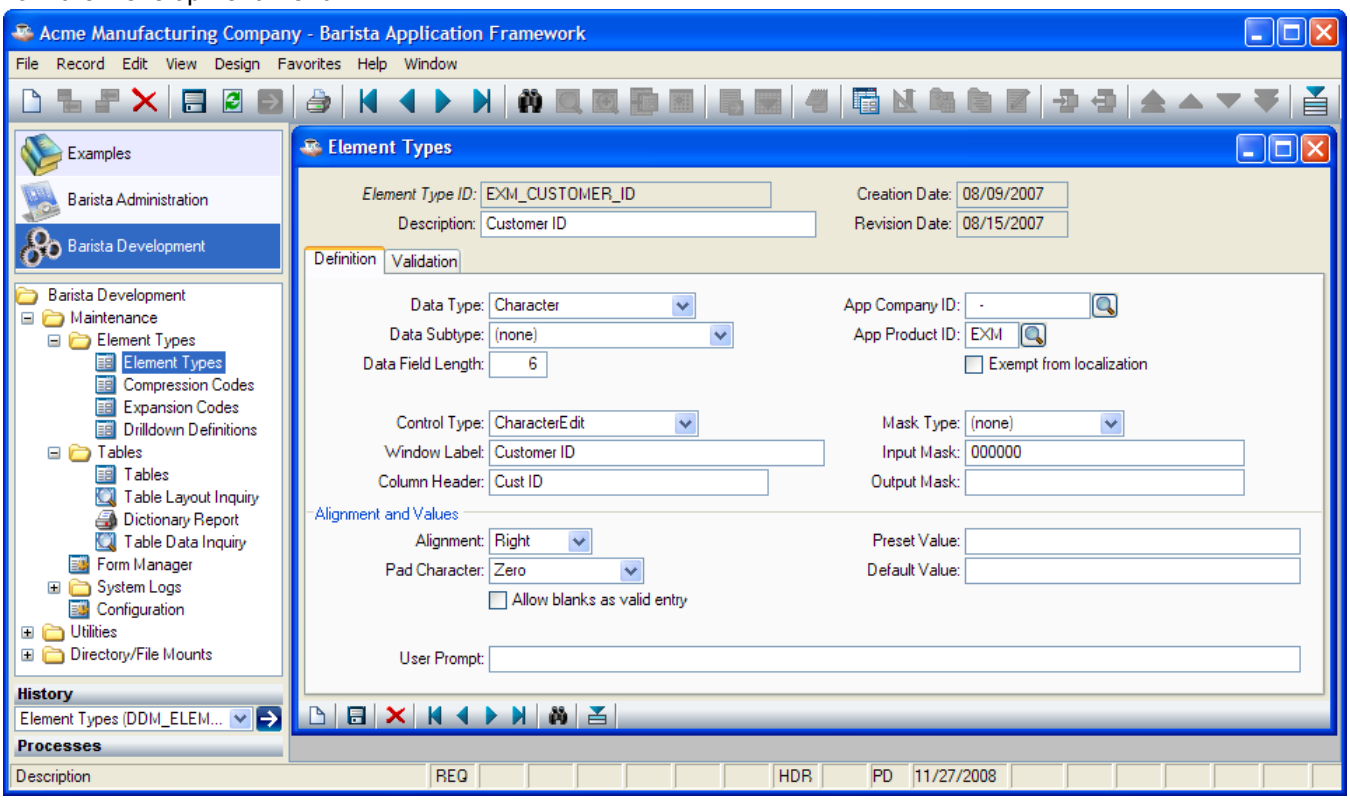
1. Element Types. Define the element types that will be used by a table. An element type can be defined in a general way (e.g. TEXT_30 can be used for many different kinds of text fields), or it can correspond to a single data element (e.g. CUSTOMER_ID always references a customer ID).
2. Tables. Define the physical layout of a table (file) and describe the data elements in the table.
3. Form Designer. Build the data entry form that will be used to maintain the table.
4. Menu Designer. Add the newly developed program to the menu system.

Documentation Conventions

Bold Italic	Key field
Bold	Required field
Underlined	Conditionally required
Bold	field
Shaded	Display-only field

Element Types

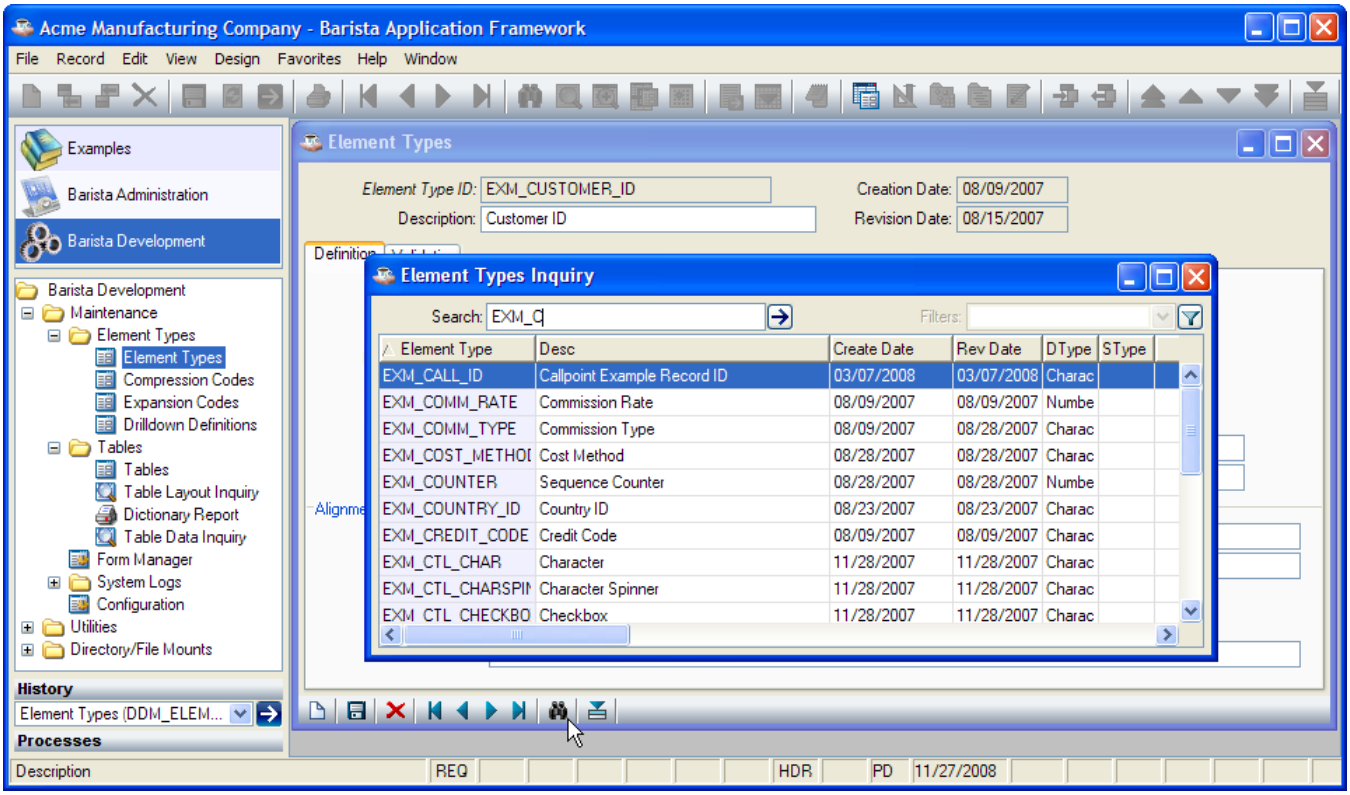
An element type defines the structure of one or more data elements. To edit element types, select Element Types from the Development menu.



Header

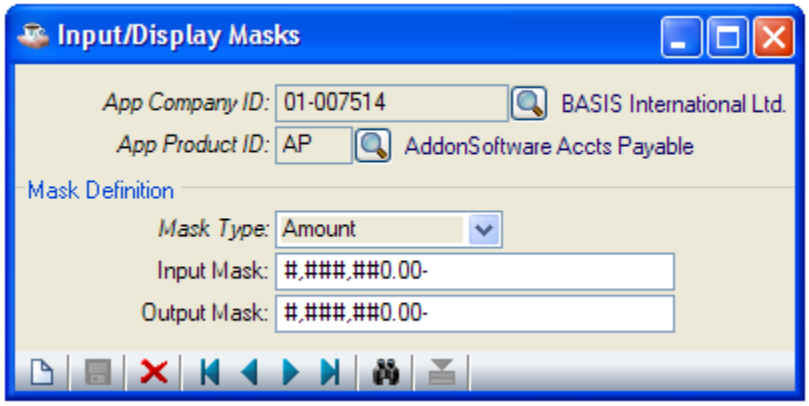
Element Type ID	The Element Type ID must contain only letters, numbers and underscores ("_"); it is automatically converted to uppercase. To see a list of existing Element Types, select View□Record Query from the menu, press [Ctrl]+Q, or press the  tool button.
Description	Describe the purpose of this data element type.
Creation Date	Displays the date on which this element type was first created.
Revision Date	Displays the date on which this element type was last changed.

Element Types Inquiry

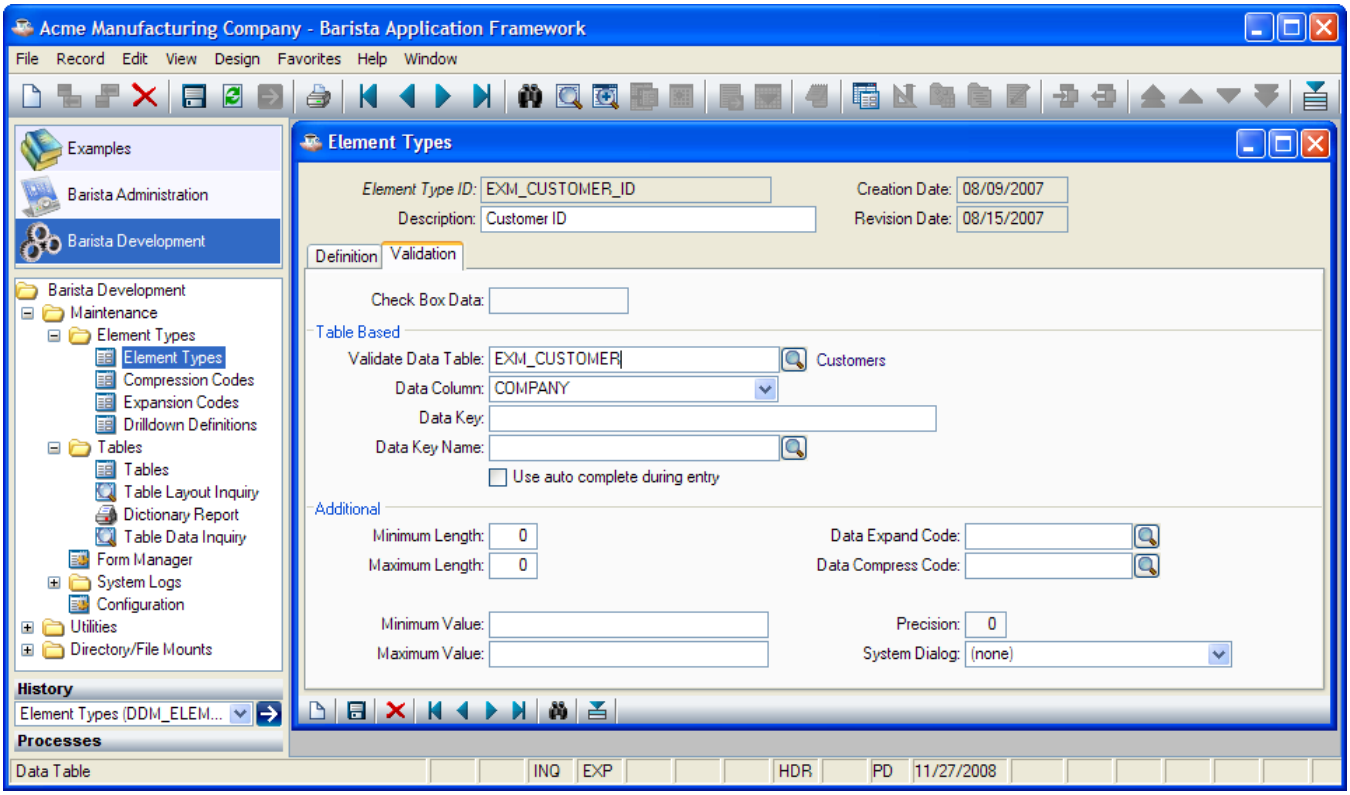


Definition

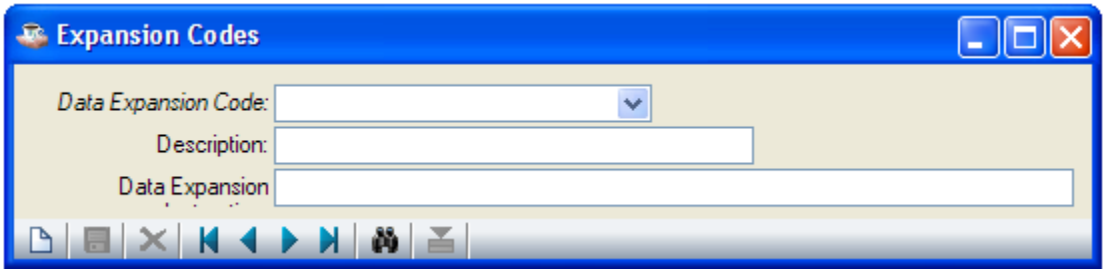
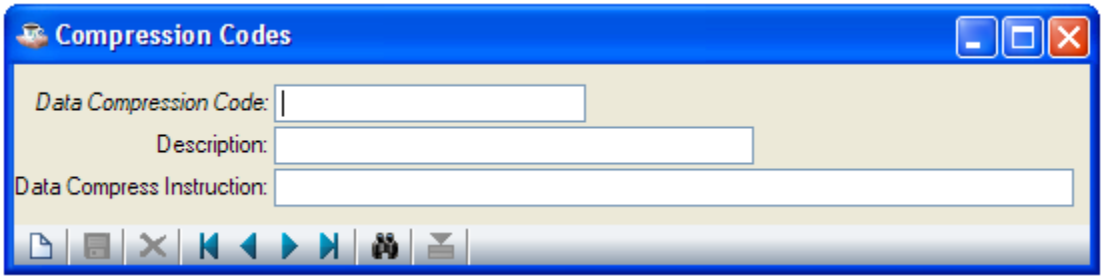
Data Type	The data type is one of the following BBj types: • Character (C) • Number (N) • Integer (I) • Unsigned Integer (U) • Business Math (B) • Blob (O)
Data Subtype	Several predefined data types, mostly involving standard date and time formats: • Date (YYYYMMDD) • Date (YYYYMM) • Date (YYYY) • Date (YYMMDD) • Date (MMDD) • Date (Julian) • Created Date Stamp (YYYYMMDD). Automatically set by Barista when a record is created. • Revised Date Stamp (YYYYMMDD). Automatically updated by Barista when an existing record is modified. • Created Time Stamp (HHMMSS). Automatically set by Barista when a record is created. • Revised Time Stamp (HHMMSS). Automatically updated by Barista when an existing record is modified. • Time (HHMMSS) • Time (HHMM) • Time (MMSS) • Postal Code • Telephone Number • Sequence Counter. Can only be used as a key field; must be defined as Character. Barista automatically increments this value on the highest key currently in the table. • Image Path. The value is entered as a filename, or any string expression that resolves to a filename. For example: [+IMAGE]+cvs([ITEM_ID],2)+".jpg"

Data Field Length	Length of this field: • 1..9999 (Character and Number) • 1..8 (Integer and Unsigned Integer) • 8 (Business Math)
Control Type	GUI control type to be used to manipulate data elements of this type: • CharacterEdit (BBjInputE or password-style BBjEditBox) • CharacterSpinner (BBj InputESpinner) • NumericEdit (BBjInputN) • NumericSpinner (BBjInputNSpinner) • DateEdit (BBjInputD) • DateSpinner (BBjInputDSpinner) • MultiLineEdit (BBjCEdit) • ListButton (BBjListButton) (Requires Element List Definitions) • ListBox (BBjListBox) (Requires Element List Definitions) • CheckBox (BBjCheckBox) • Slider (BBjSlider) • RadioButtons (BBjRadioButton) (Requires Element List Definitions) • None – Hidden. Used for data elements that will be stored to the file, but not displayed on the form. • None – Ref Only. Used to enter information for documentation (reference) purposes only. Does not cause data to be stored to the file, and is not displayed on the form.
Window Label	Enter the default label to be used when adding a data element of this type to a Maintenance Form or Options Entry Form. The initial default value corresponds to the Description from the header.
Column Header	Enter the default column header to be used when adding a data element of this type to a Maintenance Grid or Options Entry Grid, or when it appears on a columnar report. The initial default value corresponds to the Description from the header.
Solution Company ID	This alphanumeric code identifies the company that is defining this data element type.
Solution Product ID	This alphanumeric code identifies the primary product that will use data elements of this type. For example, an inventory code is used by multiple applications, but its Solution Product ID would be the Inventory Control application.
Exempt From Localization	Excludes this element when creating locale property files used in internationalization.
Mask Type	The Mask Type is a standard predefined mask that can be selected from the following list: • Amount • ID • Units • Percent • Rate • Extension • Quantity • Cost • Price • Conv Factor • Mtl Factor • Ovhd Factor • Hours • Zip Code • Telephone The actual value of each mask is defined in the "Input/Display Masks" program, which can be accessed from the Administration menu: 
Input Mask	This is the mask to be used for data entry.
Output Mask	This is the mask to be used for displaying and reporting data elements of this type. If not set, the input mask is used.
Alignment	Select one of:

	• None (default – don't justify/pad data) • Left (Justify left – text style) • Right (Justify right – numeric style)
Pad Character	Select from the list: • None (default) • Space ( ) • Quote (") • Pound (#) • Asterisk (*) • Period (.) • Zero (0) • Caret (^) • Underscore (_)
Allow blanks	Allows blanks (spaces) as valid entry.
Preset Value	If a value is entered here, it will be used as the enforced (display-only) value for data elements of this type. Preset Value and Default Value are mutually exclusive; if both are set, Preset Value is used.
Default Value	If a value is entered here, it will be used as the default (editable) value for data elements of this type. Preset Value and Default Value are mutually exclusive; if both are set, Preset Value is used.
User Prompt	The user prompt (help message) appears in the first segment of the status bar when data elements of this type are selected.



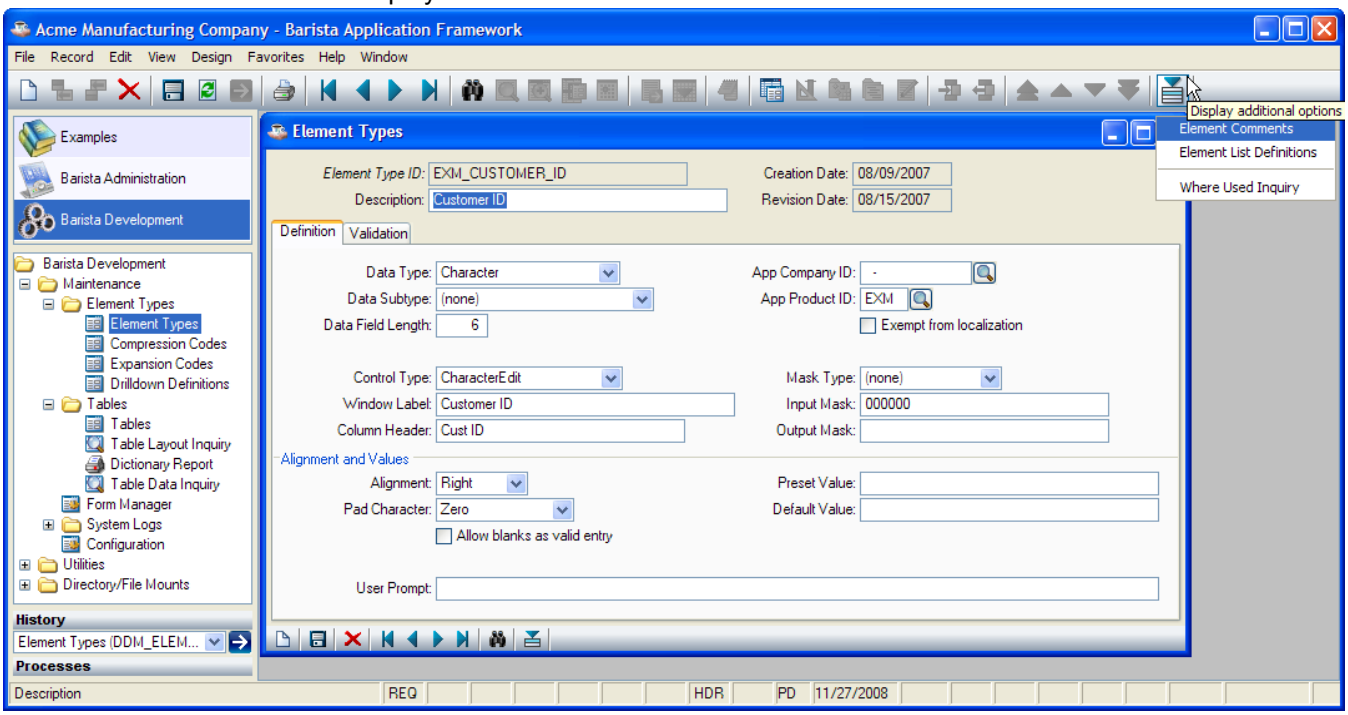
Validation

Check Box Data	If Control Type is CheckBox, this specifies how checked and unchecked values are written to the data file, in the format "*checked;unchecked*". Typical values might be "Y;N" or "1;0".
Table Based	
Validate Data Table	To validate a code against a lookup table, enter the table name here.
Data Column	Select the data element from the validation table to be displayed as a description to the right of the entered code.
Data Key	If the lookup key is a complex expression, enter it here. The expression can include any combination of BBj string functions, string literals, and [stbl] references, with @ acting as a place holder for the data value itself. For example: [+FIRM_ID]+"C"+@
Data Key Name	If the validation key is not the primary key, select the secondary key here.
Use Auto Complete	Displays an auto complete drop down list during entry.
Additional	
Minimum Length	To enforce a minimum length, enter it here.
Maximum Length	To enforce a maximum length, enter it here.
Minimum Value	To enforce a minimum value, enter it here.
Maximum Value	To enforce a maximum value, enter it here.
Data Expand Code	If a transformation must be applied to the data when reading it from the file, select an expansion code here. The expression is defined in the Expansion Codes maintenance program; % is the place holder for the raw value read from the file: 
Data Compress Code	If a transformation must be applied to the data when writing it to the file, select a compression code here. The expression is defined in the Compression Codes maintenance program; % is the place holder for the value as entered by the user: 
Precision	For NumericEdit (BBjInputN) fields, specify the decimal precision in the range -1..99.
System Dialog	To override the default Find lookup function and map it to a File Open or File Save dialog, select a system dialog here. The choices are: • None (default) • File Open • File Save

Element Type Options

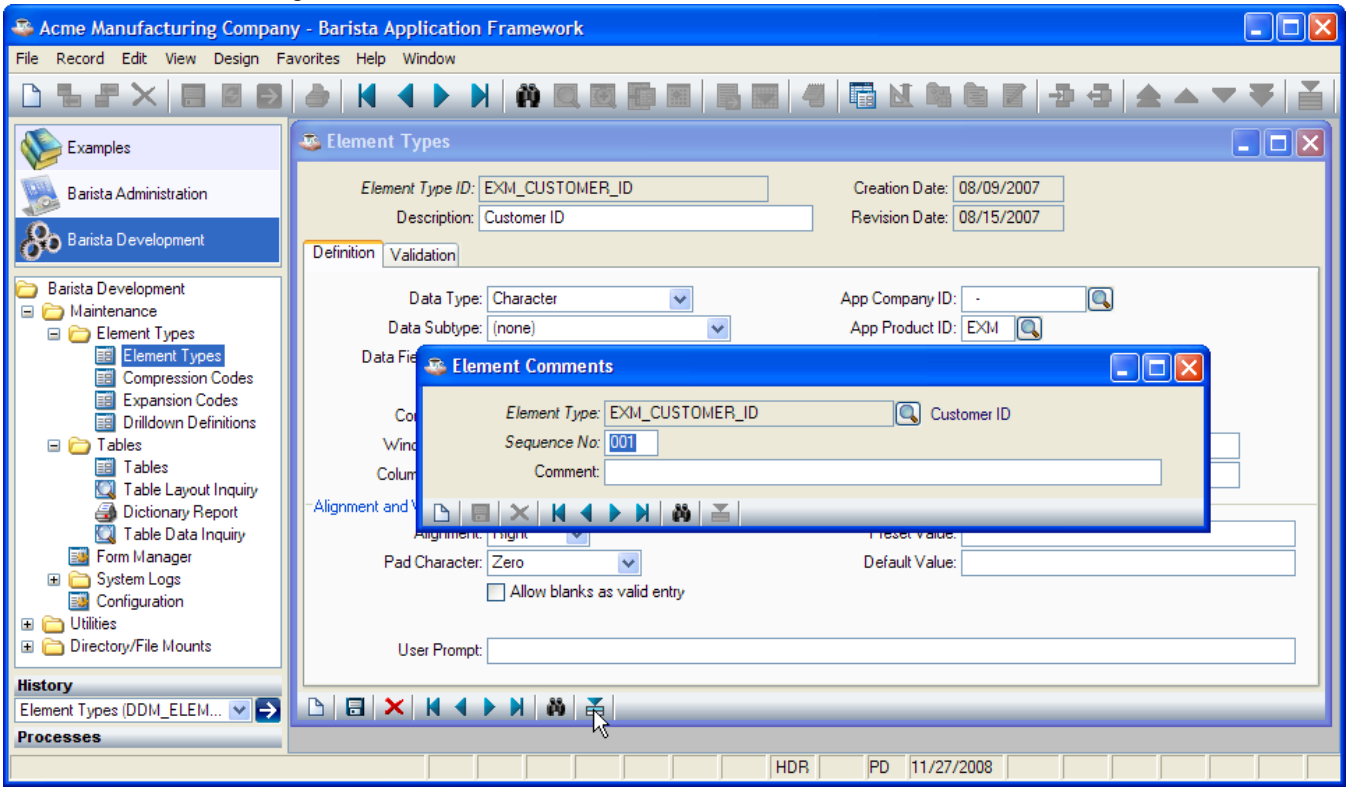
To access Element Type Options, right-click in the header or click the  tool button. The options are:

- Element Comments
- Element List Definitions
- Element Where-Used Inquiry



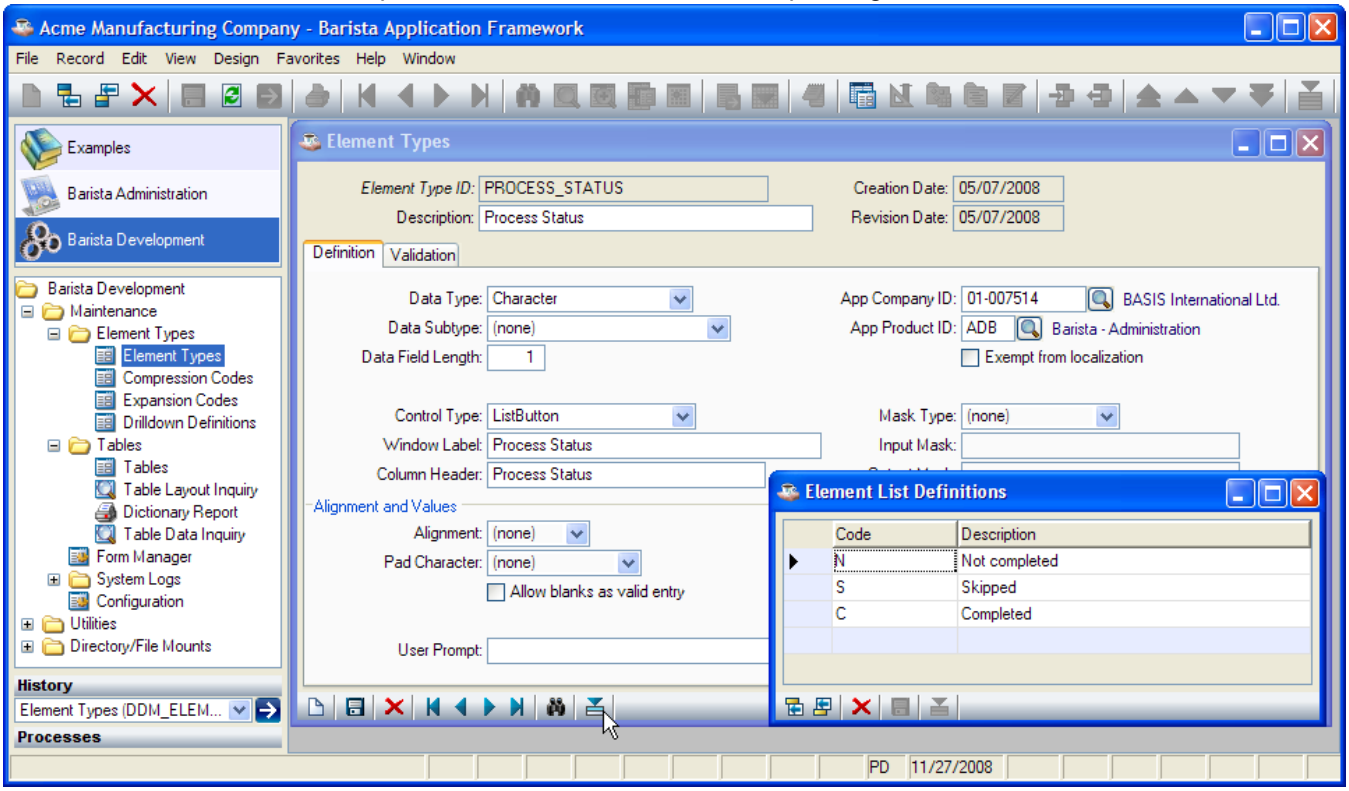
Element Comments

Free-format notes relating to this element.



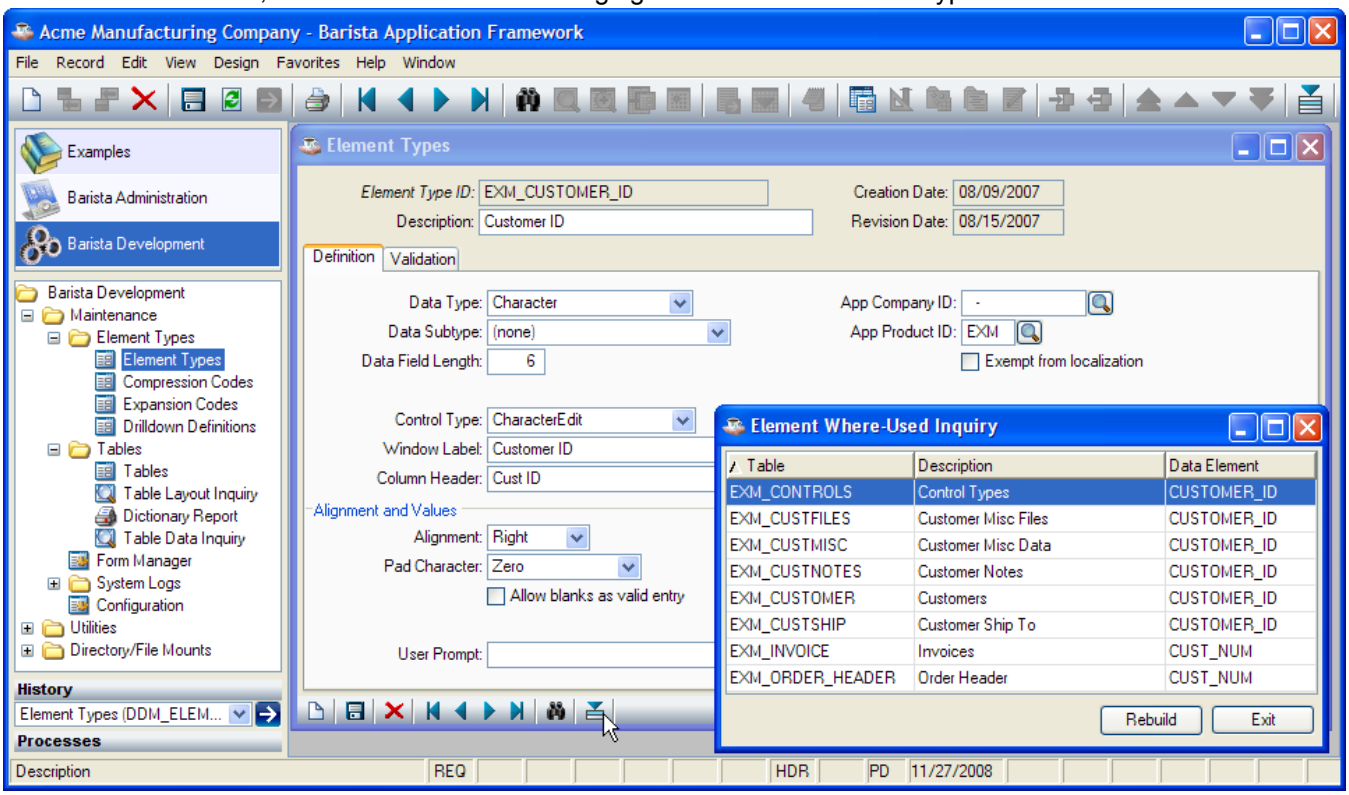
Element List Definitions

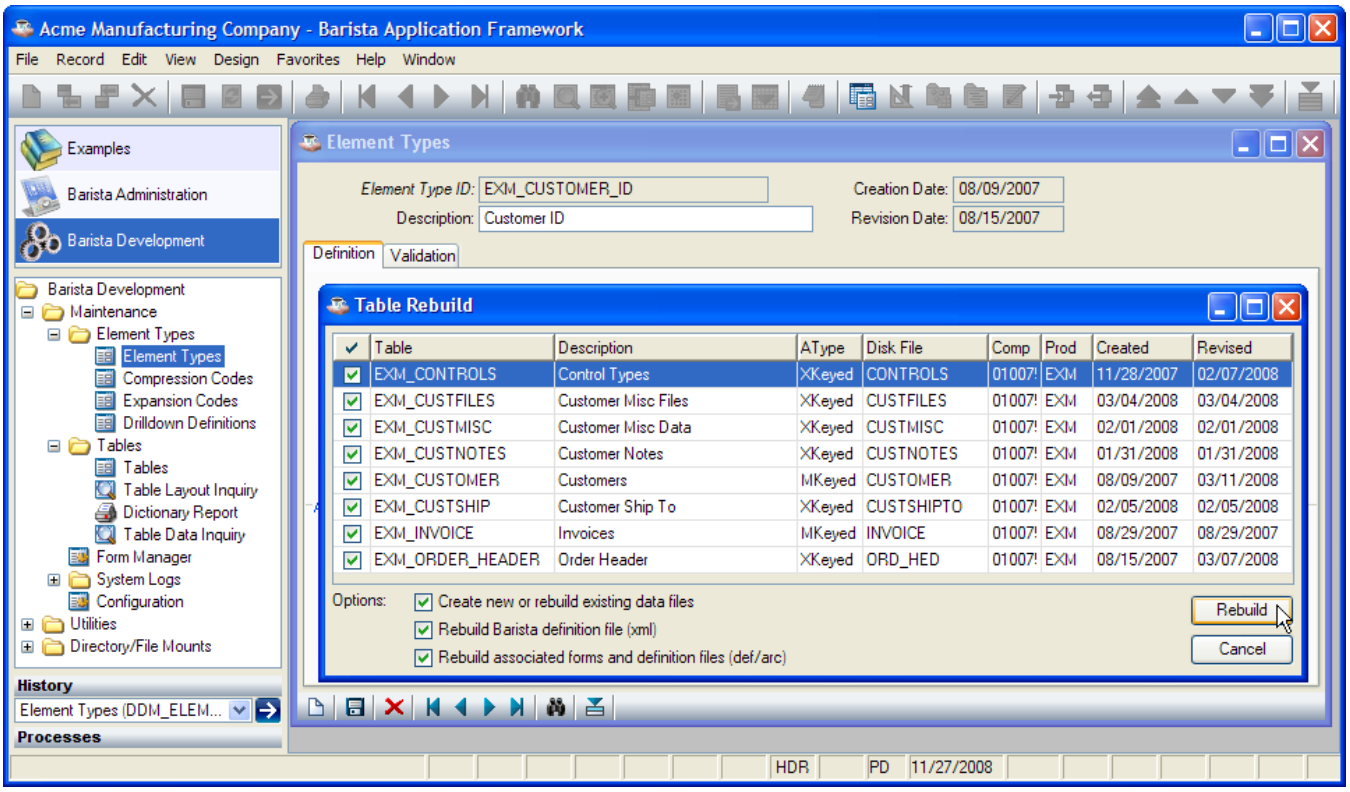
For list-oriented GUI controls (ListButton, ListBox, and RadioButton), fill in the Element List Definition to populate the list. Barista shows the description values to the user; the corresponding codes are written to the record.



Element Where-Used Inquiry

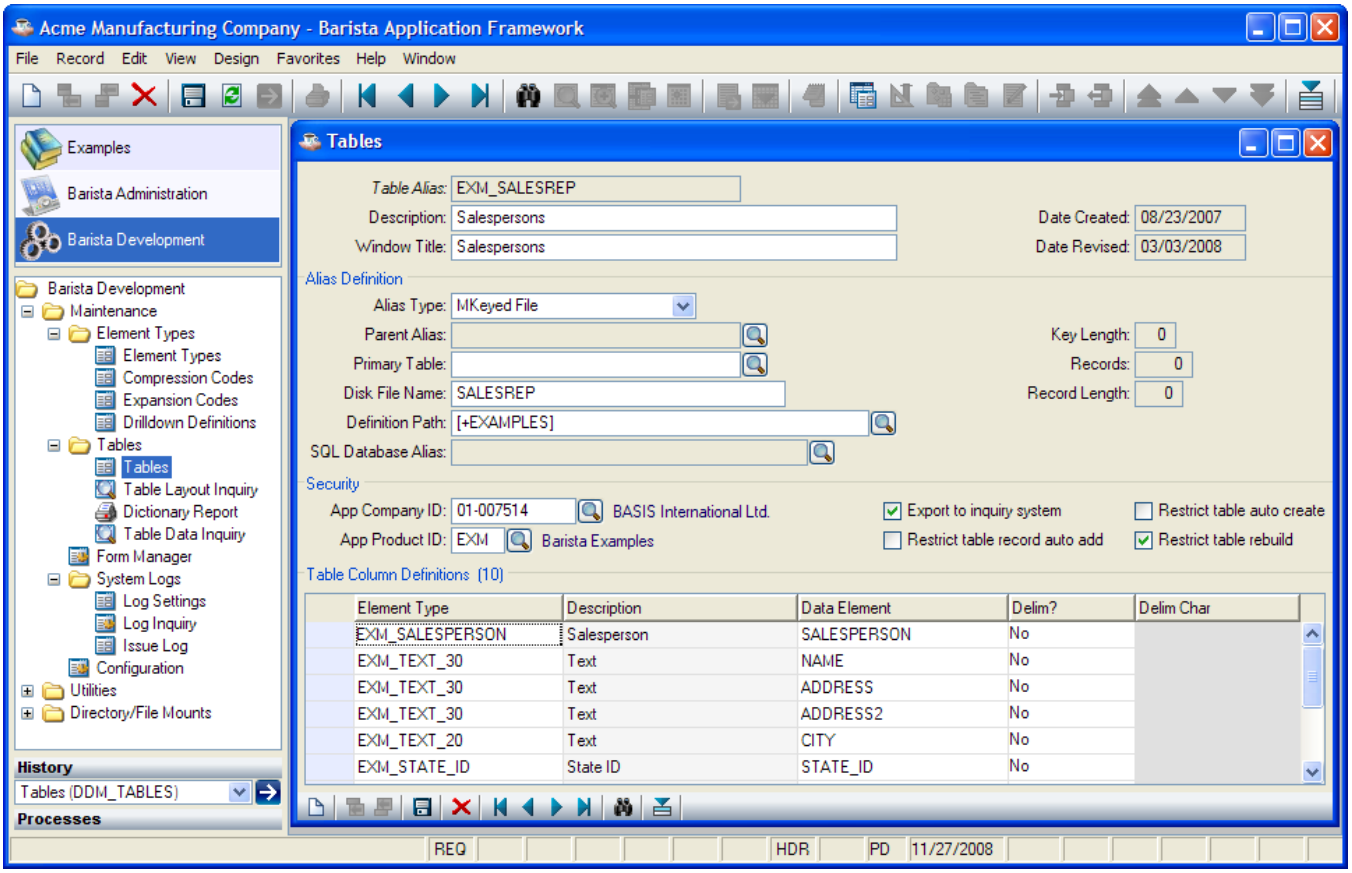
The Element Where-Used Inquiry reports all tables that use this Data Element Type. It also provides a quick way to rebuild those tables, a useful feature when changing the size of the element type.





Tables

The Table Maintenance program is used to define aliases for data files as well as records that don't map to data files.



Header

Table Alias	The Table Alias must follow standard identifier naming rules: ● It must start with a letter. ● It must contain only letters, numbers and underscores ("_"). ● It is automatically converted to uppercase. To see a list of existing Tables, select View□Record Query from the menu, press [Ctrl]+Q, or press the  tool button.
Description	Enter a short description describing the purpose of this table.
Window Title	Enter the title to be used on forms generated for this table.
Created	Displays the date on which this table was first created.
Revised	Displays the date on which this table was last changed.

Alias Definition

Alias Type	Alias type is selected from the following list: ● Mkeyed File ● Xkeyed File ● Vkeyed File ● Single Mkeyed File ● Serial File ● Direct File ● Indexed File ● Sort File ● String File ● Child Alias - Indicates that this alias is an alternate view of an existing (parent) alias. If the Child Alias type is selected, "Parent Alias" must be filled in. ● Options Entry - Indicates that this alias defines data to be entered using an Options Entry Form or an Options Entry Grid; it doesn't write data permanently to a data file. ● Reference - Indicates this alias is a reference only, or “documentation” record not associated with any disk table.
Parent Alias	If defining a "Child Alias" (an alternate view), the "Parent Alias" is the controlling alias that defines the file.
Primary Table	If defining a detail table, this reference field can contain "Parent Table".
Disk File Name	The base filename on disk.
Table Path	The path to the file.
Key Length	Valid key sizes are: Single Mkeyed 1..12 File 0 Direct File 1..64 Sort File 1..64
Records	A non-zero number of records must be specified for Direct and Sort Files.
Record Length	A record length in the range 1..32767 must be specified for Direct, Indexed, and Single Mkeyed files.

Security

Solution Comp ID	This alphanumeric code identifies the company that is defining this table.
Solution Prod ID	This alphanumeric code identifies the primary product that will use this table. For example, an inventory master file is typically used by multiple applications, but its Solution Product ID would typically be the Inventory Control application.
Export to Inquiry System	The inquiry system allows ad hoc SQL queries. To disable ad hoc queries for this table, deselect "Export to Inquiry System".
Restrict Table	When entering data validated to this table, Barista will ask if the user wishes to add the new record if the entered data is not found.

Record Auto Add	
Restrict Table Auto Create	Barista will automatically create the file if not found. This stops the default behavior.
Restrict Table Rebuild	Disallows table rebuild.

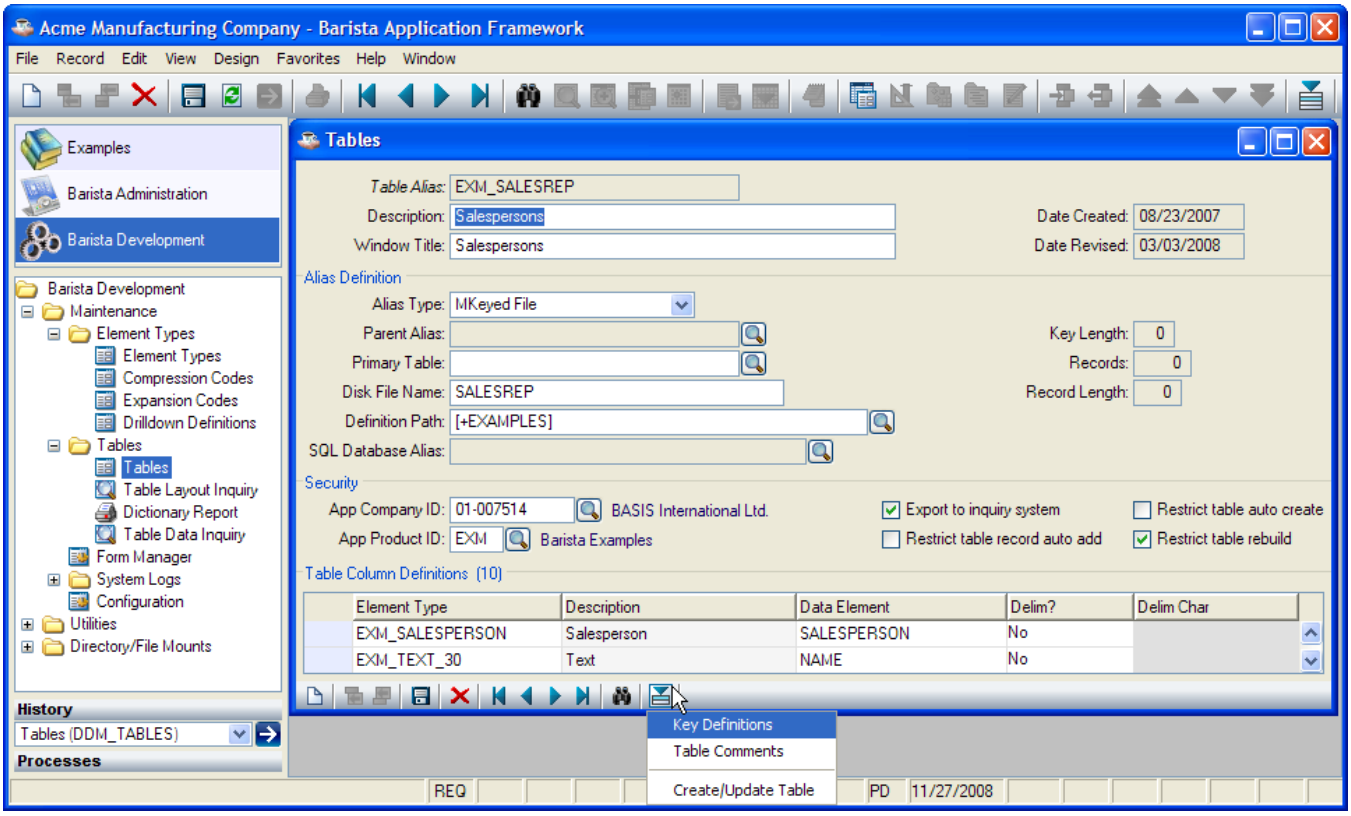
Element Detail

Element Type	Enter an existing Element Type ID.
Description	The description is displayed from the Element Type.
Data Element	Defaults to the same as the Element Type. The Data Element is the actual variable name used in the record and template for table access. This value must be unique within the Element Detail records for a specific table.
●Mult	Specifies the multiplier for the length of the Element Type within a field. For example, if an Element Type/Data Element ("RESERVE_TEXT") with a length of 5 bytes is entered, along with a Mult value of 4, the actual field length in the table will be 20 bytes (5 bytes * 4).
●Occ	S the number of occurrences for a specfic Element Type/Data Element within a table. For example, if an Element Type/Data Element ("SALES_AMT") with a length of 10 bytes is entered, along with a Occ value of 12, the table will actually contain 12 fields ("SALES_AMT_01"- "SALES_AMT_12"), each with an length of 10 bytes.
Delim?	Indicates whether this field is fixed-length or delimited.
Delim Char	If this is a delimited field, select the delimiter character from the list.

●To edit the "Mult" or "Occ" values, select Record□Expand Grid Record from the menu, press [Ctrl]+E, or press the  tool button.

Table Options

To access Table Options, right-click in the form header, select Record□Options from the menu, press [Ctrl]+O, or press the  tool button.



Key Definitions

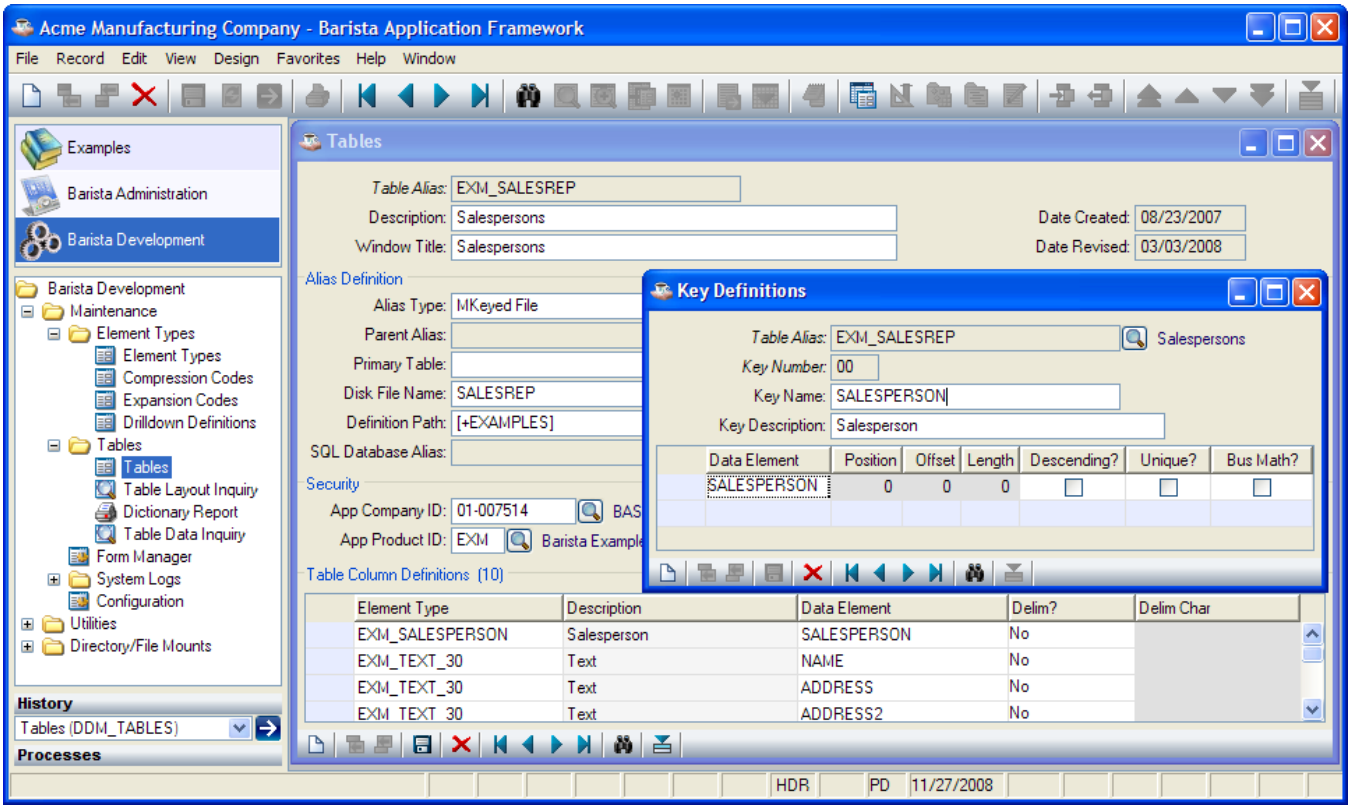
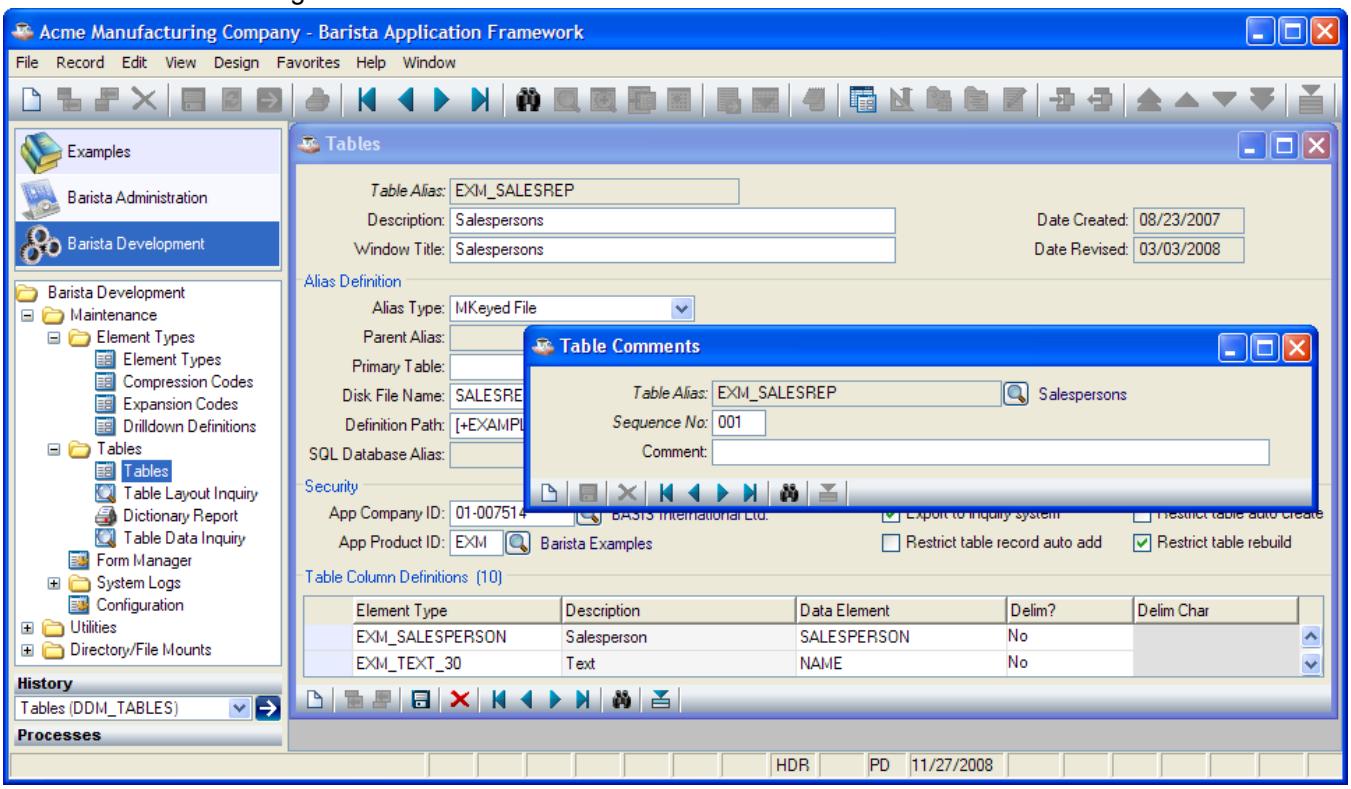


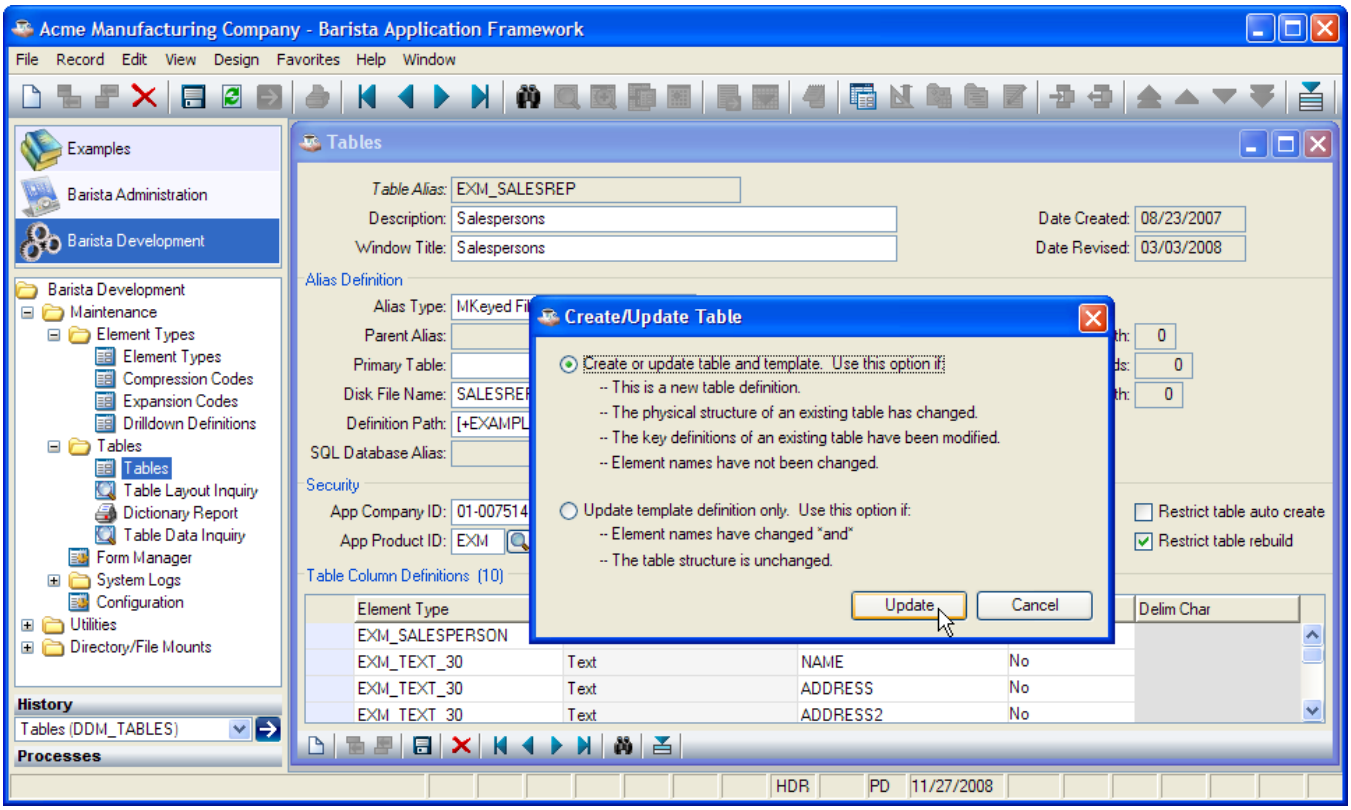
Table Comments

Free-format notes relating to this table.



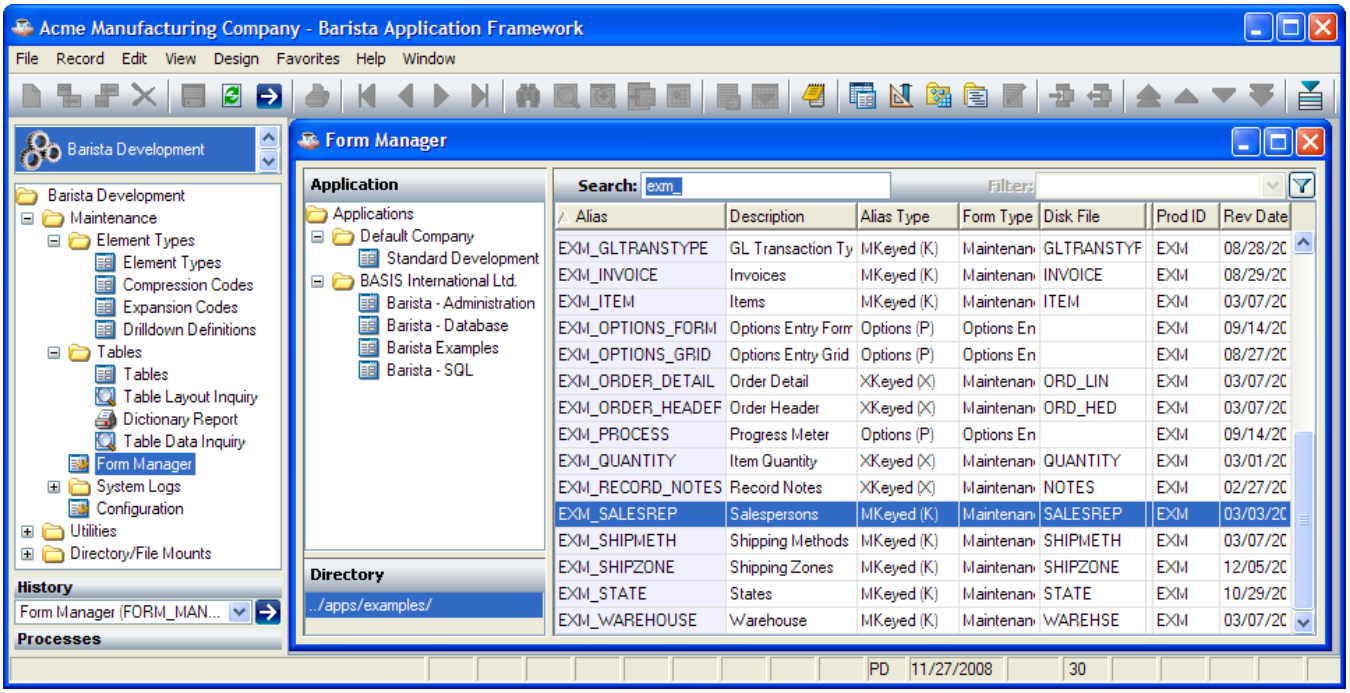
Create/Update Table

Used to rebuild the table after making changes to the Table Definition. During this process, the table template is rebuilt, and the definition is exported to the Inquiry System, if specified in the Security section of Table Maintenance.



Form Manager

The Form Manager is the entry point to the Form Designer. To run the Form Manager, select Design□Form Manager from the menu bar, press F8, select Form Manager from the Development application menu, or press the  tool button.



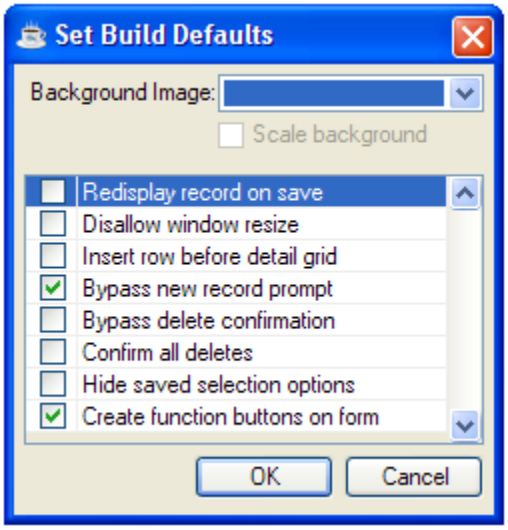
The Form Manager presents a list of Table aliases with the following information:

ID	Table/Alias ID

Description	Table Description
Atype	Alias type
Ftype	Form type
Disk File	Disk file name
Sys	System code
Comp	Company code
Prod	Product code
Rev Date	The last time the table or form information was revised
Build Date	The last time the form was rebuilt

Build Defaults

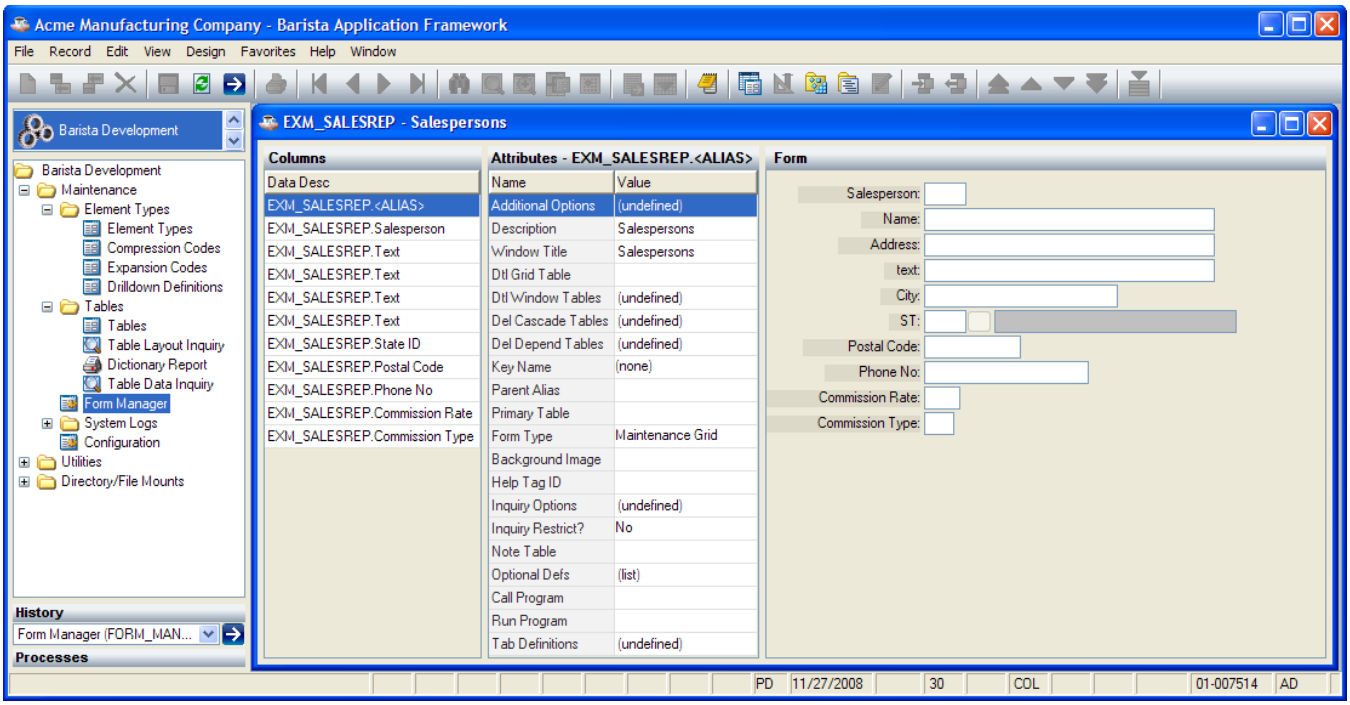
To set default values for some form options, select Design□Set Build Defaults from the menu bar.



Options selected here act as overrides; they supercede options selected in the Form Designer.

Form Designer

The Form Designer is the central workbench for creating and editing forms. To run the Form Designer, press ENTER or double-click on an alias in the Form Manager list, select Design□Form Designer from the menu bar, or press the  tool button.



The Form Designer is organized around three panels:

1. ***Columns*** shows the individual components that make up the form, including the table alias, the data elements within that table, and optional derived components, described below.
2. ***Attributes*** shows the attributes of the currently selected column.
3. ***Form*** shows a preview of the form; this is also the form editor.

Columns

A database table is a set of data elements organized into horizontal rows (records) and vertical columns (fields). In the following sample table, ***Description*** is a column and the record for ***Item 000002*** is a row:

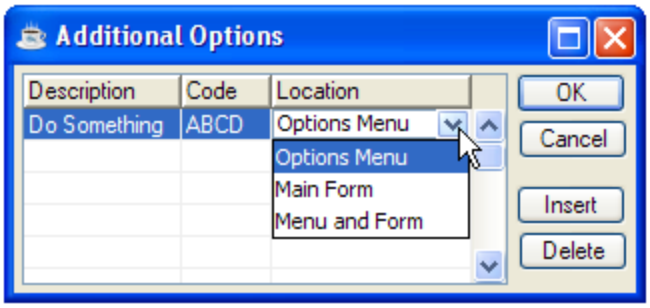
ITEM	DESCRIPTION	CAT	UOM	COST	WEIGHT	UNIT	PRICE
000001	Ancho - Pods, small	CH	EA	1925	113.4	GR	3.5
000002	Ancho - Pods, large	CH	EA	3575	226.8	GR	6.5
000003	Blue Corn Meal	FL	EA	11	0.45	KG	2.75
000004	Pepitas	SN	EA	275	85.05	GR	2.25
000040	Blue Corn Chips	CO	BAG	22	0.45	KG	4

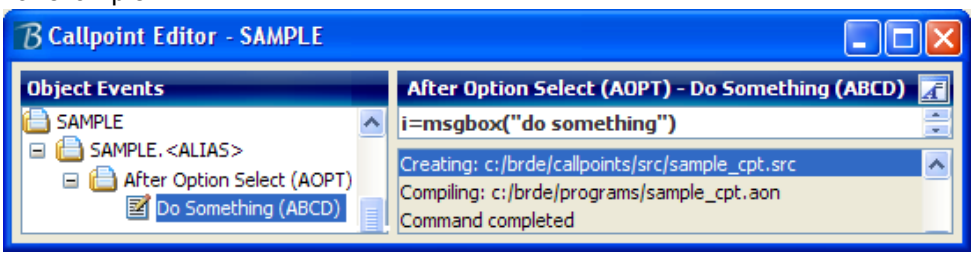
Within the Barista Form Designer, the Columns panel lists the individual table columns. It also includes a reference to the table itself and (optionally) derived elements, described below. A column can be dragged and dropped within the list to move it on the form. Select an item on the Columns panel to view or edit that item's attributes.

Form Attributes

To view or edit overall form attributes, select the *ALIASNAME*.<<ALIAS>> item at the top of the Columns panel. The following form attributes are available:

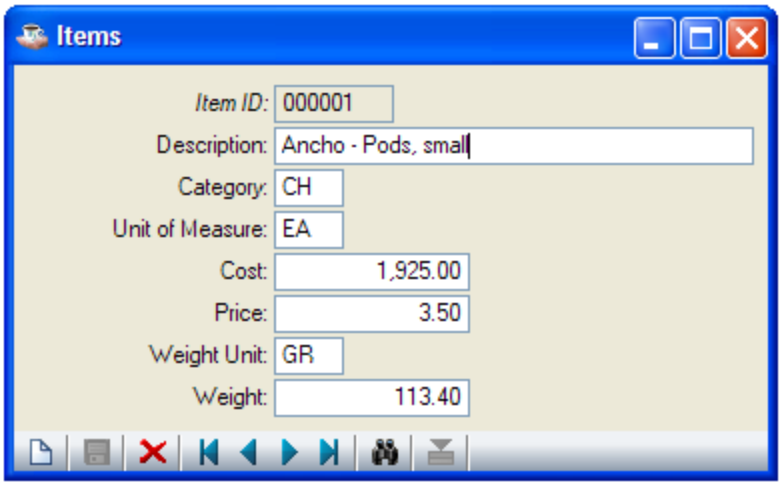
Add Options	Click here to define program options to be accessed using the Options menu, or to be associated with buttons added to the bottom of the form:

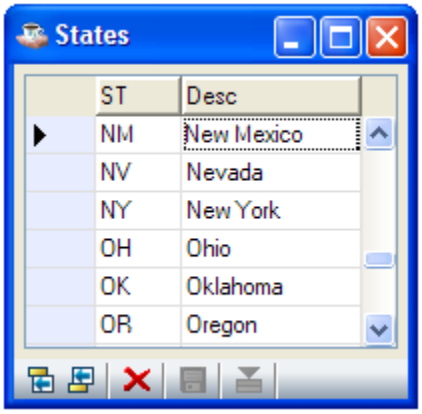


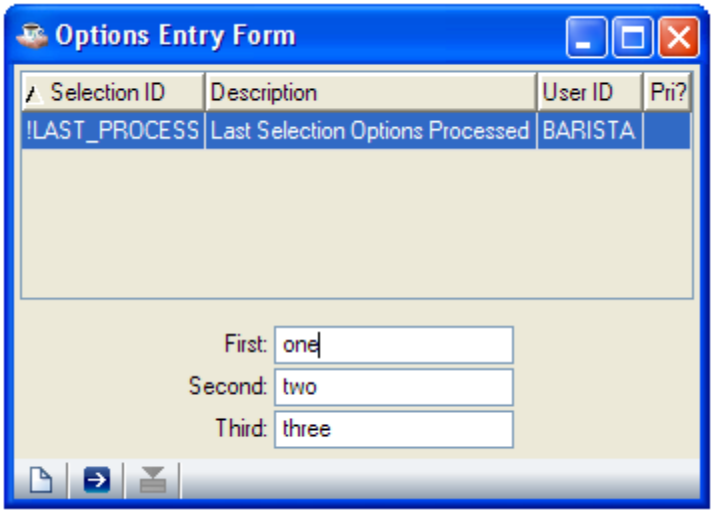
Description	The option description will appear on the options menu and/or the button label.
Code	A unique callpoint code to identify this option. When the user selects this option, Barista runs the code defined for the "After Option Select" callpoint for this code. For example:  Callpoint code is edited in the Callpoint Editor, described below.
Location	Select one of: • Options Menu – to add this option to the Options tool button. This is the default. • Main Form – to add a button to the bottom of the generated form. • Menu and Form – to add this option to the Options tool button and also add a button to the bottom of the generated form.

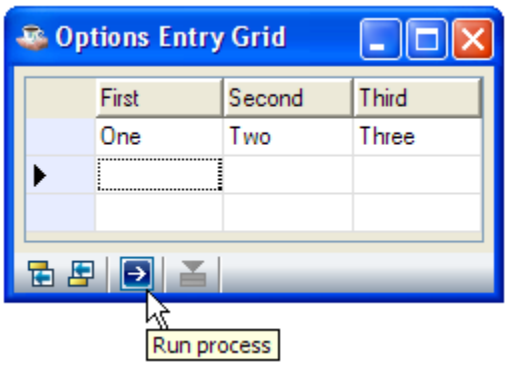
Description	Enter a description for this form. The description will appear in the Form Manager list.
Window Title	The Window Title defaults to the same as the Description.
Dtl Grid Table	If this is a header/detail form, select the table to use in populating a detail grid.
Dtl Window Tables	Optionally select one or more associated tables. These will be added to the Options tool button.
Del Cascade Tables	Optionally select one or more associated tables. If any delete cascade tables are specified here, deleting a record from this table will also delete all related records (records with keys that start with the key to this table) from the delete cascade tables.
Del Depend Tables	Optionally select one or more associated tables. If any delete dependency tables are specified here, any attempt to delete a record from this table will be disallowed if any of the delete dependency tables contain related records (records with keys that start with the key to this table).
Data Key Name	To view data in other than primary key sequence, select an alternate key here.
Parent Alias	Imported from the Table.
Primary Table	Imported from the Table.
Form Type	The Form Type describes the form's basic structure and behavior. Barista implements four form types.

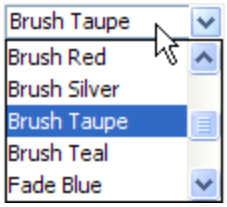
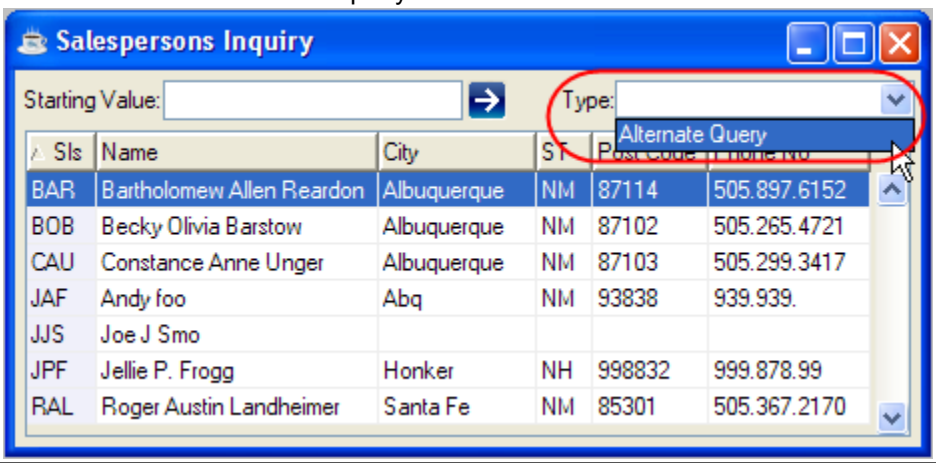
	Maintenance Form. A formatted screen for maintaining a table; it displays a record at a time. The Barista default form presents the data elements in a single column, with labels down the left side. This basic design can be customized using the form editor and enhanced by adding labels, tabs, group boxes, and buttons. **Maintenance Grid.** This is also used for maintaining a table, displaying the data in a grid. **Options Entry Form.** An options entry form is used for entering a single page of information and passing it on to a program for processing. The overlay program is typically a report or batch update. **Options Entry Grid.** This is used for entering multiple rows of information in a grid, then passing that information over to a report or update program for processing.
Background Image	Select an optional background image for the form; for example:

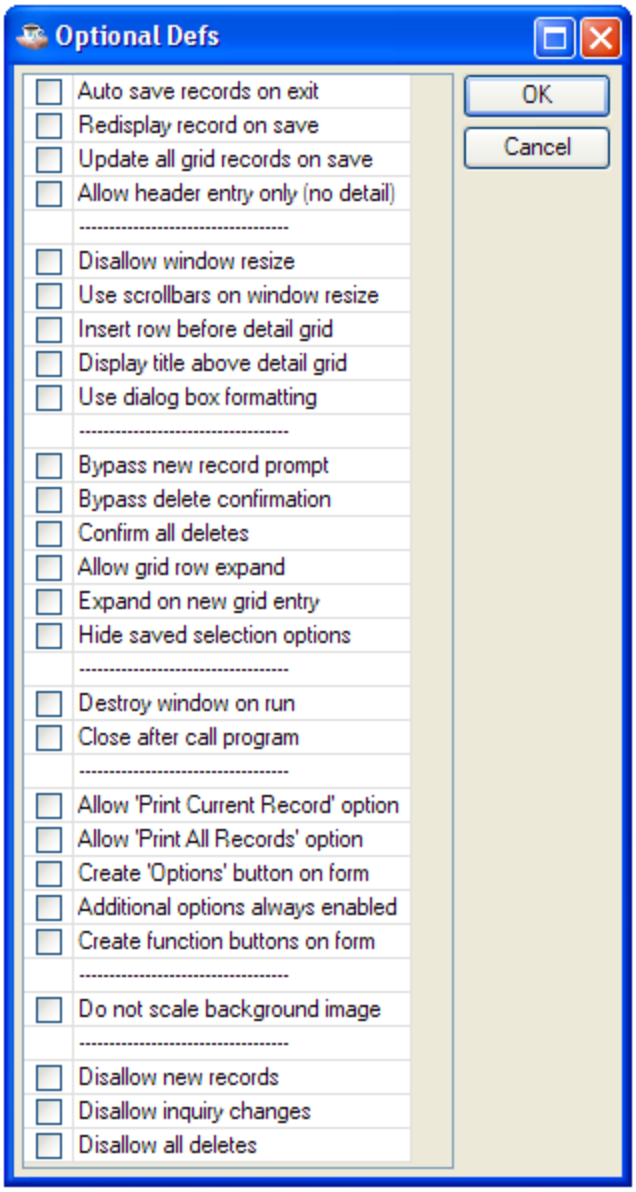




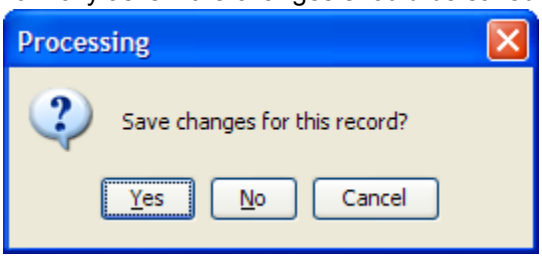


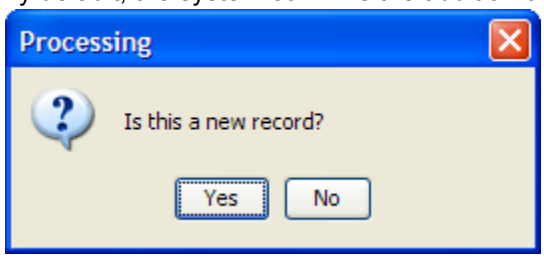
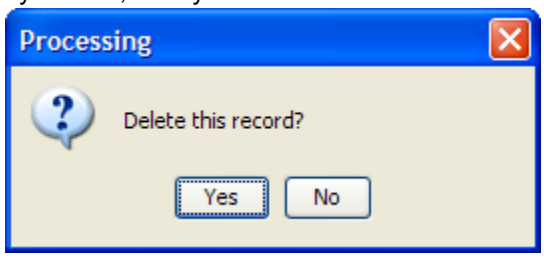
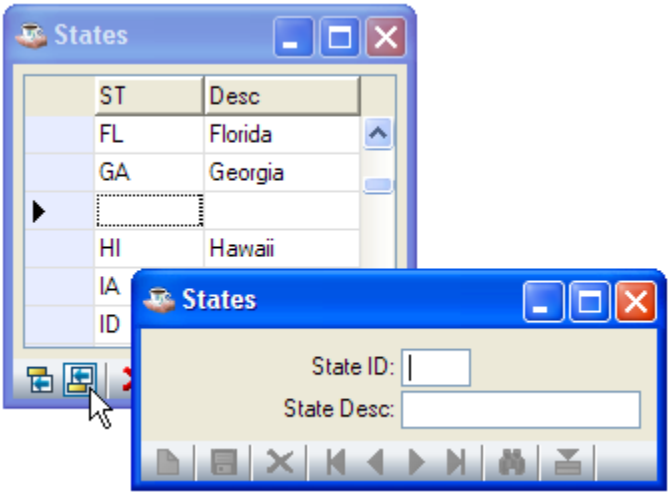


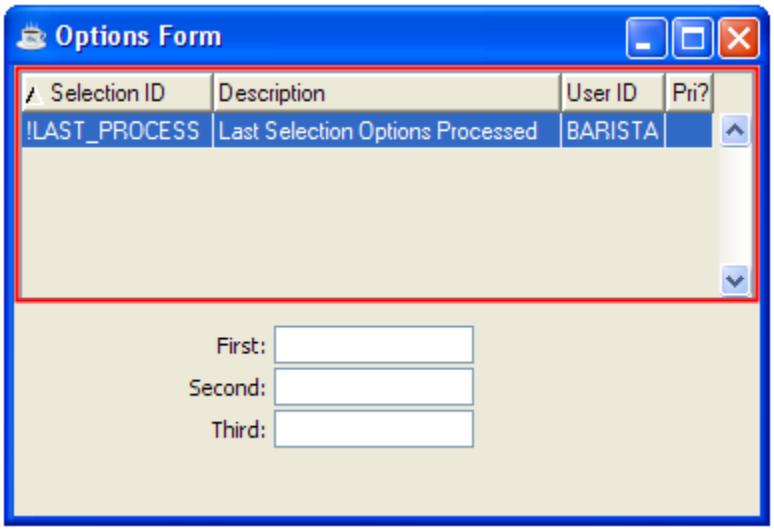
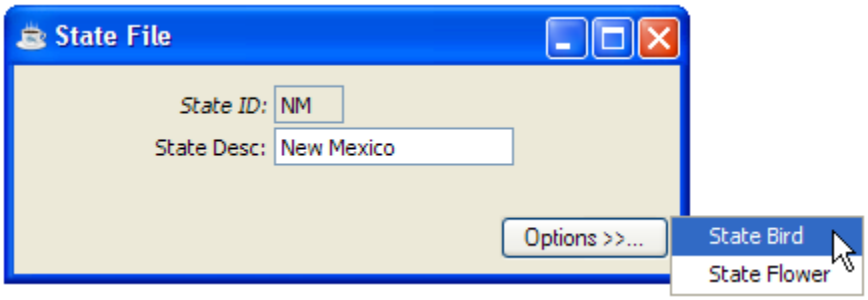
	 Barista comes with a standard set of images; to install additional background images, copy them to **Barista\sys\images\im_back_*.***.
Help Tag ID	Not yet implemented.
Inquiry Options	In a Form Maintenance program, Record Query presents a list of records in the table. [Alt]ernate query options can be added here. Description — A short description of the alternate query option. This will appear in the "Type" dropdown list on the Inquiry form, as shown in the sample below. Alias — The alternate query table. Column — The alternate query column. 
Inquiry Restrict?	By default, grid-based forms allow the user to sort on any column by clicking in the column header. When this option is selected, grid sorting is restricted to key fields to ensure optimal performance.
Note Table	Barista provides the option to store notes for any specific record entered for any defined table. To configure this option, the developer must create a table definition with the following key format: DD_TABLE_ALIAS – Specifies the table id for the record being created. NOTE_KEY – The maximum length of a record key to be stored in the note table. This allows note records for multiple tables to be store in the same Note Table. Example: We want to create a single Note Table to store user notes for the following tables: TUT_CUSTOMER – Key length = 6 TUT_SALESPERSON – Key length = 3 The Note Table would be defined as: DD_TABLE_ALIAS (16*) NOTE_KEY (6*) – the maximum possible length of the key from TUT_CUSTOMER and TUT_SALESPERSON NOTE_TEXT (512*)
Optional Defs	Click here to customize the form. The available options are:

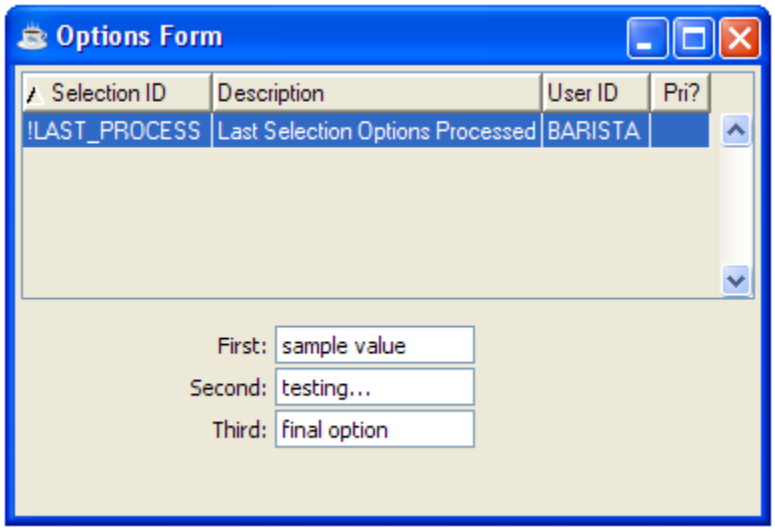
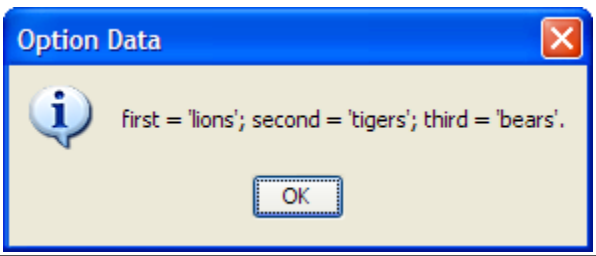


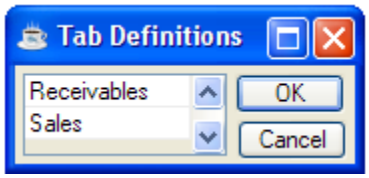
Several of these options can be set automatically using Design□Set Build Defaults.

Auto save records on exit	If the user attempts to exit from a program with unsaved changes, the system normally asks if the changes should be saved:  If "Auto save records on exit" is selected, the user doesn't see a confirmation dialog; unsaved changes are automatically saved.
Redisplay record on save	After the user saves a modified record, the system normally clears the Maintenance Form and moves input focus to the first field on the form. If "Redisplay record on save" is selected, the form is not cleared, and input focus is unchanged.
Disallow window resize	Forms are user-resizable by default. If "Disallow window resize" is selected, forms are not user-resizable.

	Insert row before detail grid	If this is a header/detail form, "Insert row before detail grid" inserts ROW_HEIGHT pixels (default 21) between the header area and the detail grid.
	Bypass new record prompt	By default, the system confirms the addition of new records with:  If "Bypass new record prompt" is selected, this confirmation message is omitted.
	Bypass delete confirmation	By default, the system confirms record deletions with:  If "Bypass delete confirmation" is selected, this confirmation message is omitted.
	Confirm all deletes	In rare cases, when a deletion will have significant consequences, it might be advisable to reconfirm it before proceeding, with a message like the following: 
	Expand on new grid entry	By default, when inserting a row into a Maintenance Grid using the  or  tool buttons, the grid is edited in place. When "Expand on new grid entry" is selected, the system opens a Maintenance Form – equivalent to pressing the  tool button – for typing in the new record: 

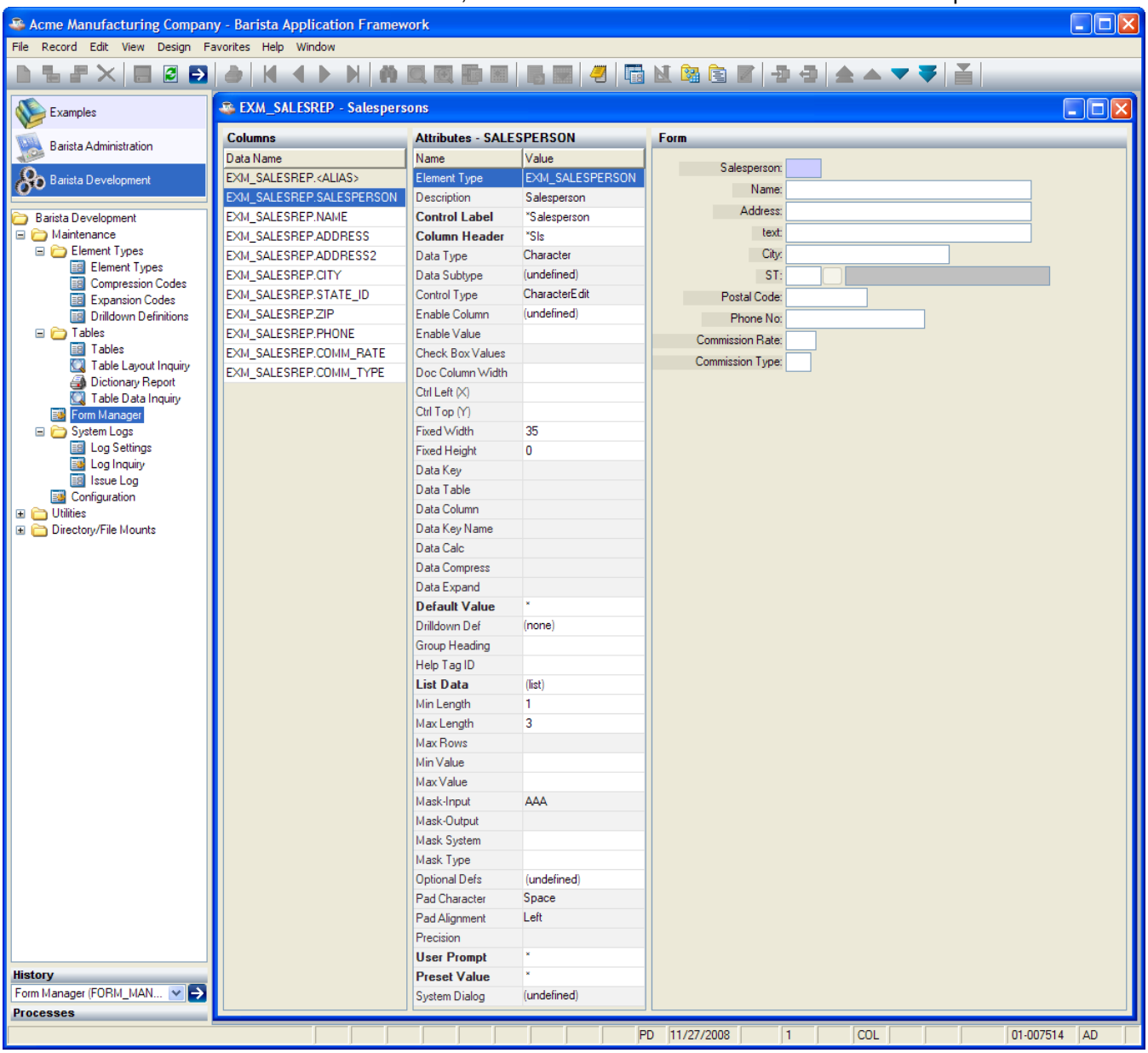
	Hide saved selection options	By default, Options Entry Forms include a list of saved and recent options, so that users can recall previously used options. If "Hide saved selection options" is selected, this list is omitted: 
	Destroy window on run	By default, the Options Entry Form remains visible after control is passed from the Options Entry Form to the overlay program. If "Destroy window on run" is selected, the Options Entry Form is destroyed when the overlay program is started.
	Close after call program	By default, the Options Entry Form retains control once the called program is completed. If “Close after call program” is selected, the Options Entry Form will process a normal exit.
	Allow Print record option	Selecting "Allow Print record option" enables the File□Print Current Record menu item and the  tool button.
	Allow Print all records option	Selecting "Allow Print all records option" enables the File□Print All Records menu item.
	Create Options button on form	By default, any options defined under "Add Options" can be accessed from the toolbar using the  tool button. When "Create Options button on form" is selected, an equivalent "Options >> …" button is added to the bottom of the form. For example: 
	Additional options always enabled	By default, the  tool button is only enabled when there is an active record on the screen. When "Additional options always enabled" is selected, the  tool button is always enabled.
	Create function buttons on form	As the user navigates through a form, either the  or  tool button is enabled if the current data element supports a "Find" or "Lookup" option. When "Create function buttons on form" is selected, the appropriate tool button is also added to the right of the data element on the generated form.
	Do not scale background image	If a background image is selected, by default it is resized when the window is resized. If "Do not scale background image" is specified, the image size is fixed.

	Disallow new records	By default, if the user enters a key that is not on file, the system assumes that the user is creating a new record. If "Disallow new records" is specified, new records cannot be created; the system generates the following dialog: 
	Disallow inquiry changes	By default, views in the Inquiry System can be modified by a user. This option specfies the table has a preset view that cannot be modified.
	Disallow all deletes	This option disables the Record□Delete menu item and the  tool button.
Call Program **Run Program**	Options Entry forms prompt the user to enter some parameters, then press [F5] or the  tool button to call or run an overlay program. For example, this Options Entry form prompts the user for three values (first, second, third):  When the user presses [F5] or the  tool button, control is passed to the specified overlay program (either a Call Program or a Run Program). Here's a sample overlay program: **rem ' Overlay for Options Entry samples** **rem ' If CALLed, share the workspace** **if tcb(13) then enter** **msg$="first = '"+Option!.getOptionData("first")+"'; "** **msg$=msg$+"second = '"+Option!.getOptionData("second")+"'; "** **msg$=msg$+"third = '"+Option!.getOptionData("third")+"'."** **i=msgbox(msg$,64,"Option Data")** **release** That overlay program produces the following dialog: 

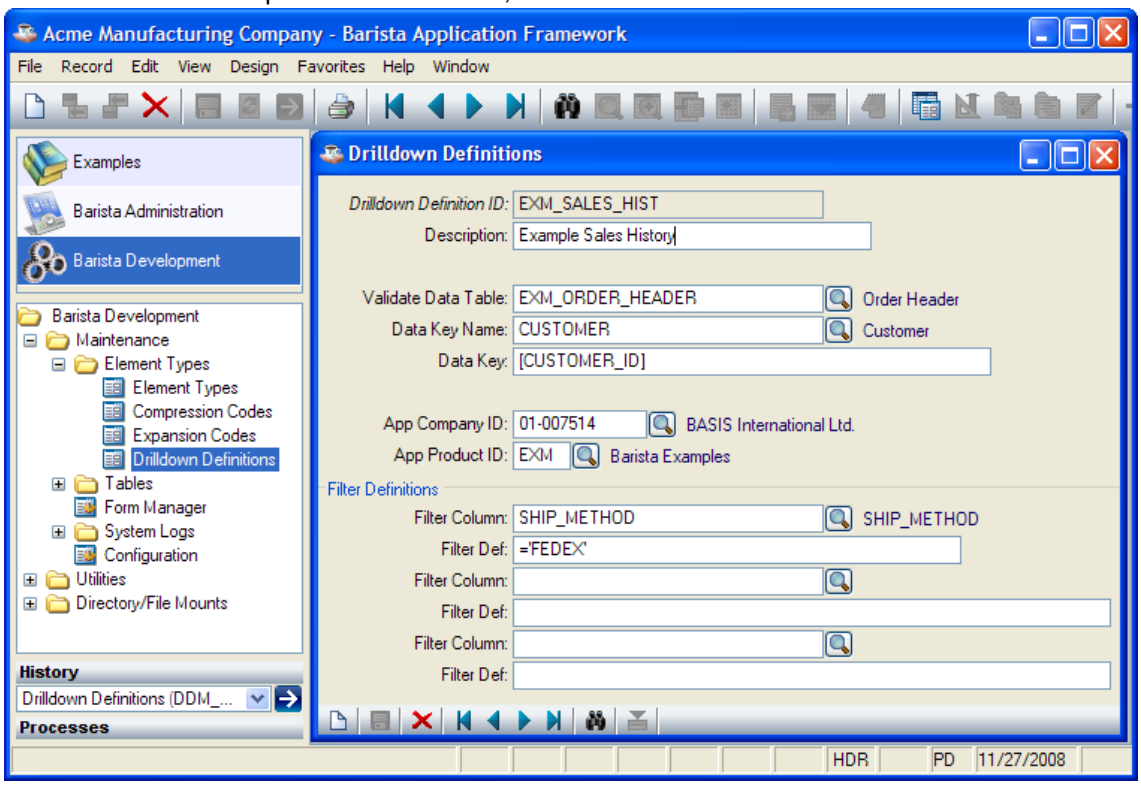
Tab Definitions	If any tabs have been defined on this form, this brings up an editable list of tab labels, for example: 

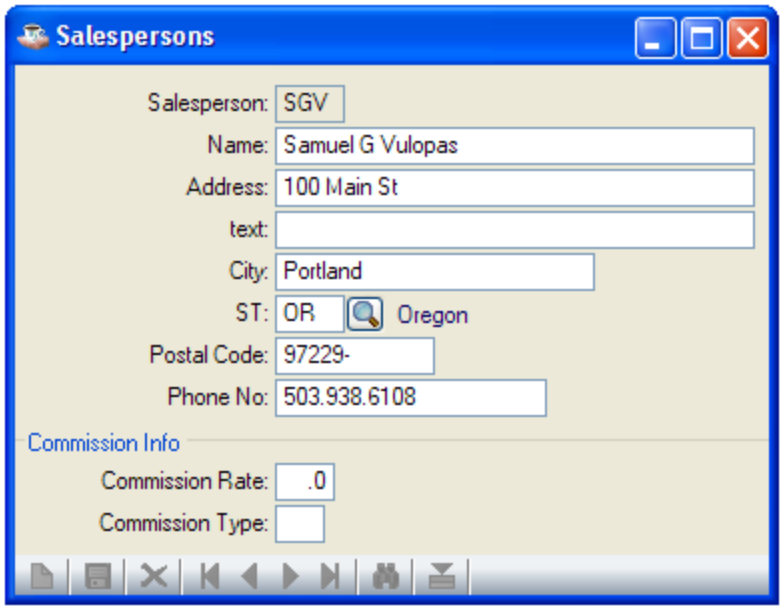
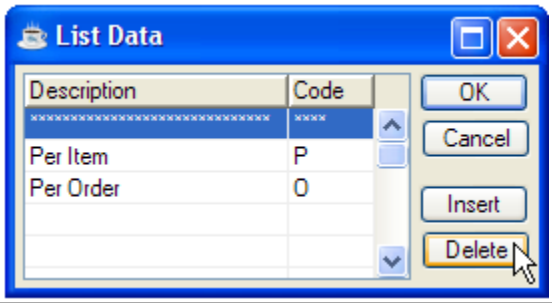
Column Attributes

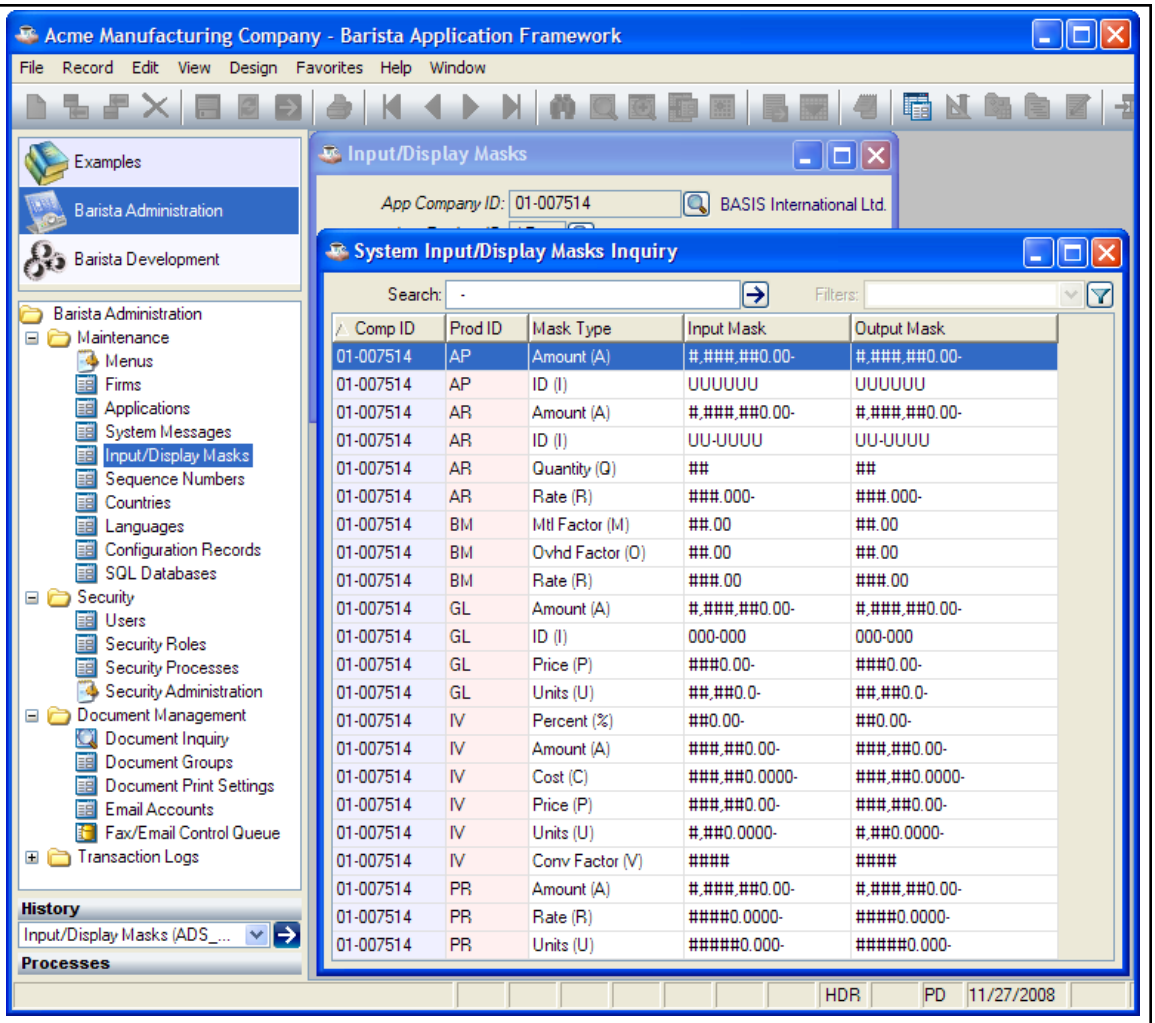
To view or edit data element column attributes, select an *ALIAS.COLUMN* item in the Columns panel.



Element Type	Displays the Element Type for this Data Element. To view the Element Type details, right-click on the data element in the form editor and select "Maintain Element Type".
Description	Displayed from the Element Type record.
Window Label	This is the label that will be used on a generated "Maintenance Form" or "Options Entry Form".
Column Header	This is the column header that will be used on a generated "Maintenance Grid" or "Options Entry Grid".
Data Type **Data Subtype**	Displayed from the Element Type record.

Control Type	
Enable Column **Enable Value**	To selectively enable this control based on the value of another column, set "Enable Column" to the column to be checked, and set "Enable Value" to the required value.
Check Box Values	Displayed from the Element Type record.
[Ctrl] Left (X) **[Ctrl] Top (Y)** **Fixed Width** **Fixed Height**	For Maintenance Forms and Option Entry Forms, these values can be adjusted to fine-tune the placement of a control on the form. The X, Width, and Height attributes are measured in pixels, while the Y attribute is based on logical rows, with the default row height being 21 pixels. The X and Y values can be specified as relative values (+*n* or -*n*). This format is interpreted as *n* columns (X) or rows (Y) relative to the previous data element.
Data Key **Data Table** **Data Column** **Data Key Name** **Data Calc** **Data Compress** **Data Expand**	Displayed from the Element Type record.
Default Value	If a value is entered here, it will be used as the default (editable) value for this column. (See also Preset Value.)
Drilldown Def	To define a drilldown option for this column, select a drilldown code from the list. 
Group Heading	To insert a section heading, enter a title here. For example, entering a Group Heading of "Commission Info" for the "Commission Rate" column would generate this form layout:

<table>
<tr><td></td><td> 
</td></tr>
<tr><td>Help Tag ID</td><td>Not yet implemented.</td></tr>
<tr><td>List Data</td><td>For list-oriented controls, enter the list values here.
<table>
<tr><td>Description</td><td>This is the value that the user will see on the screen.</td></tr>
<tr><td>Code</td><td>This is the value that will be stored on disk.</td></tr>
</table>
When defining a custom list, the place-holder entry must be deleted:

</td></tr>
<tr><td>Min Length
Max Length</td><td>To enforce a minimum and/or maximum length for this column, enter the length value(s) here.</td></tr>
<tr><td>Max Rows</td><td>Displayed from the Element Type record.</td></tr>
<tr><td>Min Value
Max Value</td><td>To enforce a minimum and/or maximum value for this column, enter the value(s) here.</td></tr>
<tr><td>Mask-Input
Mask-Output</td><td>Displayed from the Element Type record.</td></tr>
<tr><td>Mask System
Mask Type</td><td>To select a standard mask, enter a System ID and Mask Type code. This resolves to a mask defined in Input/Display Masks:</td></tr>
</table>

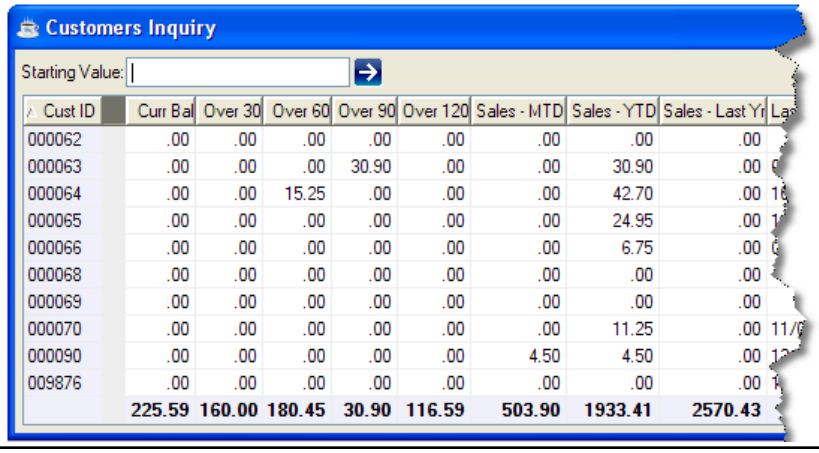
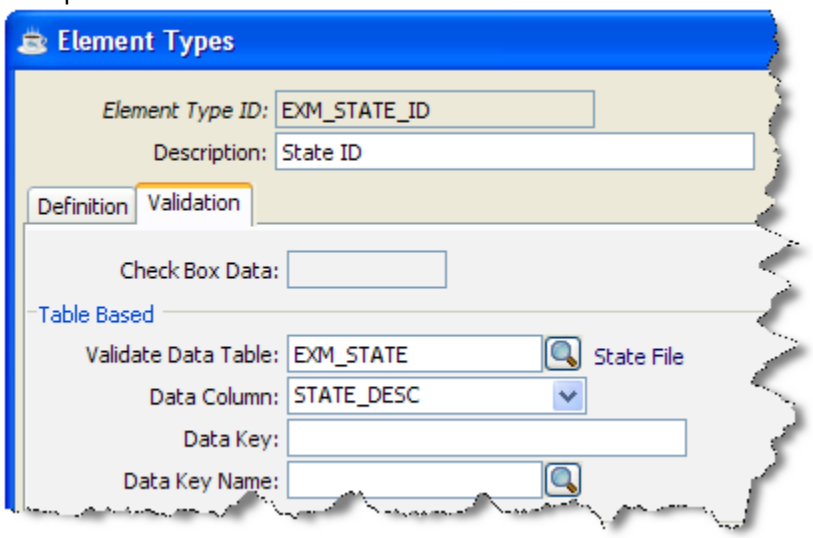
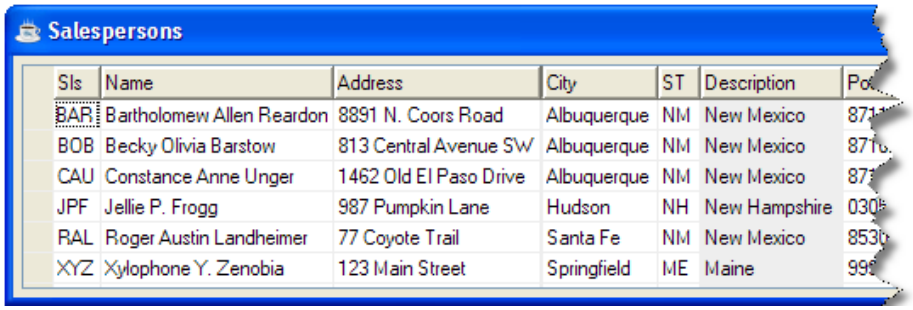


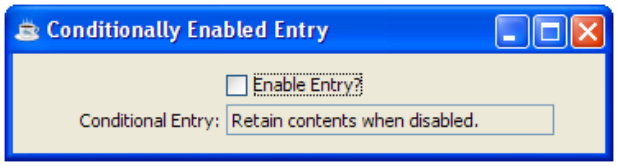
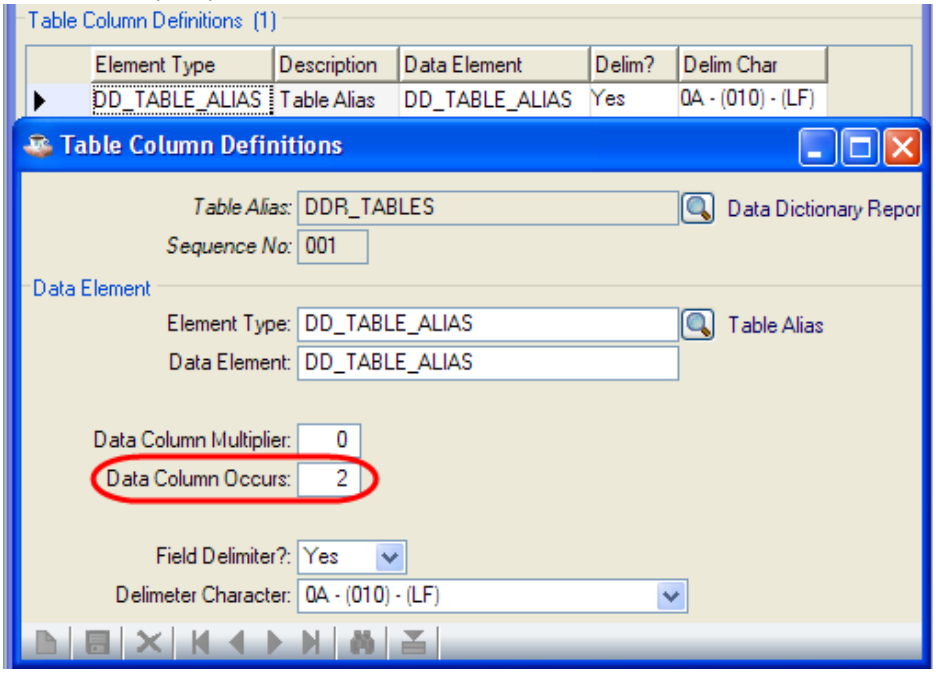
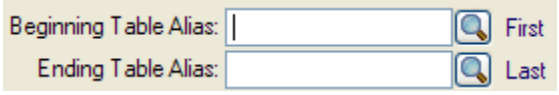
For masking rules, refer to Numeric Masking.

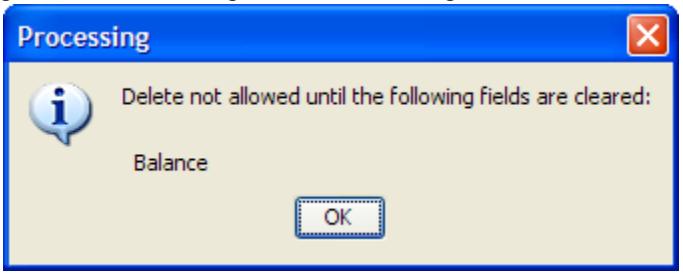
Optional Defs

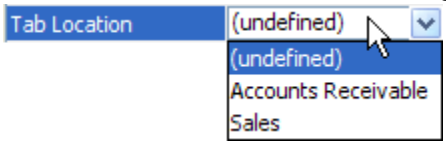
Click here to customize the behavior of this column. The available options are:

Hide data in maintenance form	Omits this column from Maintenance Form and Options Entry Form programs.
Hide data in maintenance grid	Omits this column from Maintenance Grids and Options Entry Grid programs.
Hide data in inquiry system	Omits this column from inquiry display.
Hide data in all forms	Omits this column from all forms.
Hide data in Print All option	Omits this column from the Print All output.
Hide data in Print Record option	Omits this column from the Print Record output.
Hide corresponding control label	Don't show the label for this column on generated Maintenance Form and Options Entry Form programs.

	Create total for inquiry column	Adds a totals line to the bottom of the Inquiry screen showing the total for all data elements that have this option selected. For example, this can be used to show totals for the various sales figures and accounts receivable balances in the customer table: 
	Display description in grid	When generating a Maintenance Grid program, this option causes the grid to display the description that corresponds to this code value. This feature assumes that table based validation rules are defined for the element type, for example:  If this option is set for the state code column in the salespersons table, and the table is defined to generate a Maintenance Grid, the resulting form looks like this: 
	Save contents on conditional disable	If this column is conditionally enabled (see *Enable Column* and *Enable Value* above), the default behavior is for the contents to be reset to the default value when the column is disabled. If this option is selected, the contents are retained when the column is disabled.

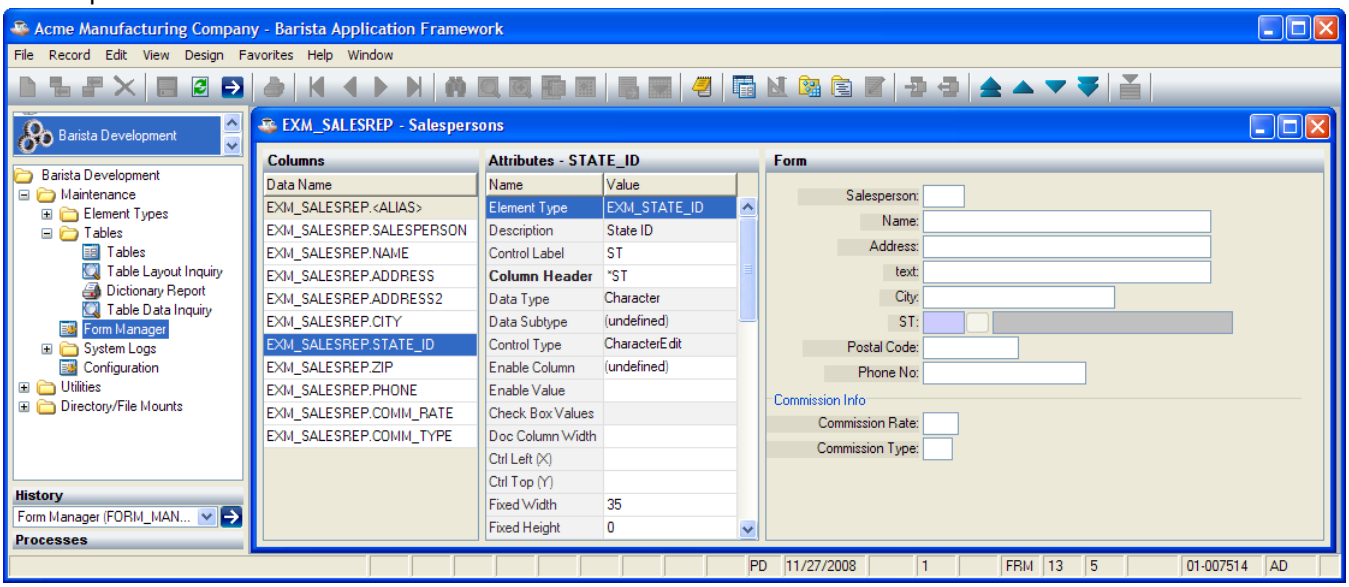
	
Period/year entry control	Used in Addon legacy code to specify if the Data Element should use the Addon period/year parameter settings.
From/to entry control set	Used when defining from/to entries in an Options Entry Form. To use this option, a single Data Element is defined in Table Maintenance with two occurrences (Occ):  The "From/to entry control set" option tells Barista to create two controls labeled "Beginning *Value*" and "Ending *Value*":  These controls implement special validation rules to ensure that the Ending value is greater than or equal to the Beginning value. In this example, the Data Element is named "RANGE". The overlay program sees the two values as Option!.getOptionData("RANGE_1") and Option!.getOptionData("RANGE_2"). Note: The "Beginning" and "Ending" strings are taken from the barista.ini file; they can be localized or adjusted based on personal preference: FROM_TO_LABEL_BEG=Beginning FROM_TO_LABEL_END=Ending
Null entry=First or Last	This option is used with a From/To entry control set based on a range of values taken from a file. When this option is selected, Barista displays descriptions of "First" and "Last" when the values are blank:  This doesn't have any effect on the values that are returned in Option!.getOptionData(); it's assumed that the application will interpret the blank values correctly. Note: The "First" and "Last" strings are taken from the barista.ini file; they can be localized or adjusted based on personal preference: FROM_TO_DESC_FST=First FROM_TO_DESC_LST=Last

	Null entry=All	This option is only meaningful for data elements that are based on a file lookup. When this option is selected, Barista displays a default description of "All" when the value is blank:  This doesn't have any effect on the value that is returned in Option!.getOptionData(); it's assumed that the application will interpret the blank value correctly. Note: The "All" string is taken from the barista.ini file; it can be localized or adjusted based on personal preference: FROM_TO_DESC_ALL=All
	Limit to variable naming rules	Entry will be forced to uppercase. It must start with a letter and must contain only letters, digits, and underscores.
	Resize control on window resize	On Maintenance Form and Options Entry Form programs, resize the control to use all available space. This option is only valid for the last visible control on the form.
	Wildcard data entry	Limits entry to the characters specified in the **+WILDCARD_CHARS** global string. By default, this is: **ABCDEFGHIZKLMNOPQRSTUVWXYZ0123456789 ?**
	Validate entries on save only	The default is to validate entry as the user tabs out of the field. Select this option to defer validation until the record is saved.
	Control contains external link	The entry field is a command or document to be handed over to the system command processor. When this option is set, Barista adds a  tool button to the form after the control. When the user clicks that tool button, the text from the control is passed to the system command processor.
	Use last valid entry as default	For the duration of a program run, the system will use the most recent valid entry as the default value for this column.
	Do not resolve STBL reference	Overrides the default behavior where Barista attempts to resolve any text within [square brackets] as an STBL.
	Display only – all cases	Sets this column to display-only.
	Display only – new records only	When creating a new record, sets this column to display-only.
	Display only – existing records only	When editing an existing record, sets this column to display-only.
	Value must be zero for record delete	Disallows delete on any records containing non-zero values in the specified columns. Attempting to delete a record with a non-zero value in this column generates a message like the following: 

	Password entry control	On Maintenance Form and Options Entry Form programs, this column is a password entry field; each input character is echoed as '*'. (Note: This option is only meaningful for Character Edit controls.)
	Spell check control	Activates spellchecking for this control.
	Zero based sequence number control	If this is a sequence number control, start counting at zero. By default, sequence numbers start at one.
	Assign next sequence on null entry	If this is a sequence number control, increment it on null entry.
	Multiple language data control	Not yet implemented.
Pad Character	Displayed from the Element Type record.	
Pad Alignment		
Precision		
User Prompt	The user prompt message is displayed in the status bar when the user enters this field.	
Preset Value	If a value is entered here, it will be used as the enforced (display-only) value for this column. (See also Default Value.)	
System Dialog	Displayed from the Element Type record.	
Tab Location	If any tabs have been defined on this form, this dropdown list shows a list of available tabs; select a tab to move this control, along with all subsequent controls, to it. 	

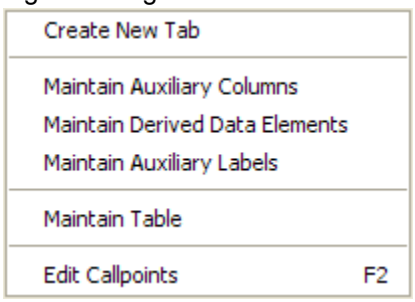
Form Editor

The Form Editor is the last panel in the Form Designer. It shows a preview of the form and implements various editing features. To move a field, for example, select it and click on  ,  ,  , or  , or simply drag and drop it to a new location.



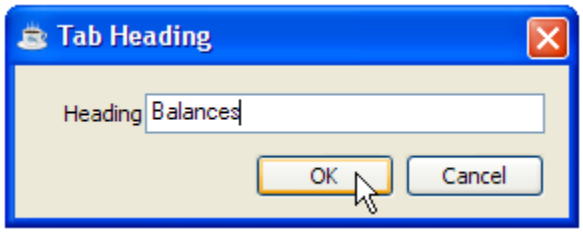
Form Editor Options

Right-clicking in the Form Editor brings up the following options:



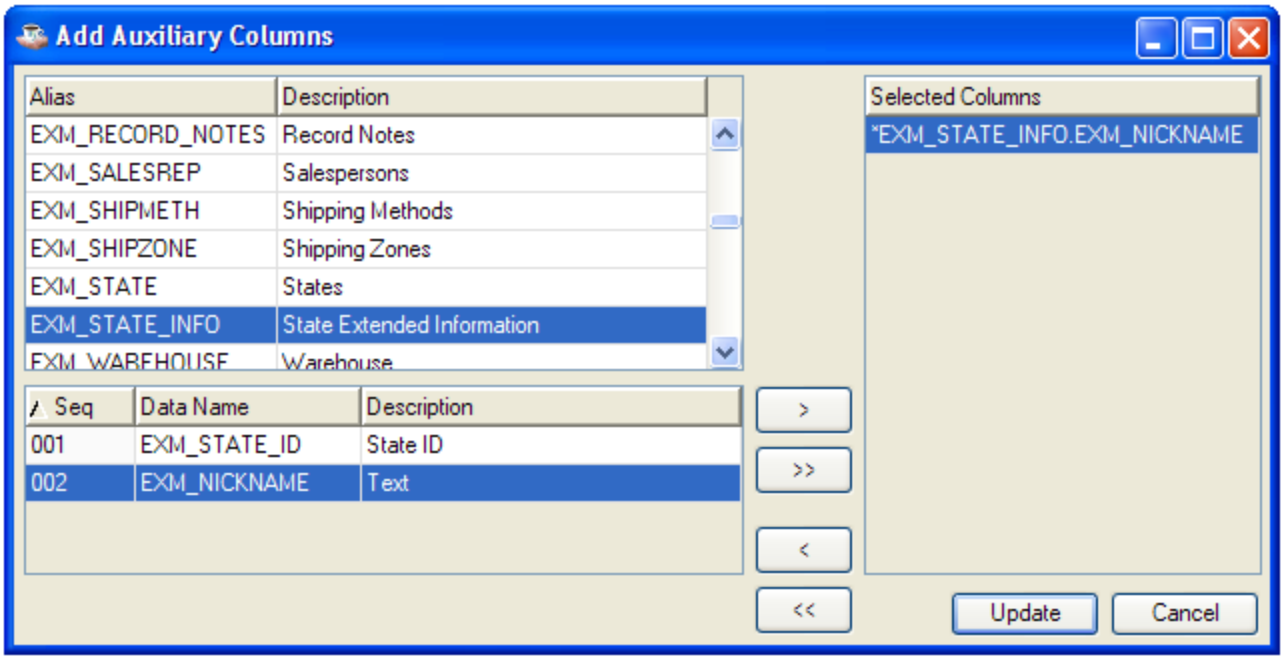
Create New Tab

To add a tab control to the form, or to add a new tab to the existing tab control, select a column, then select this option and enter a tab heading.



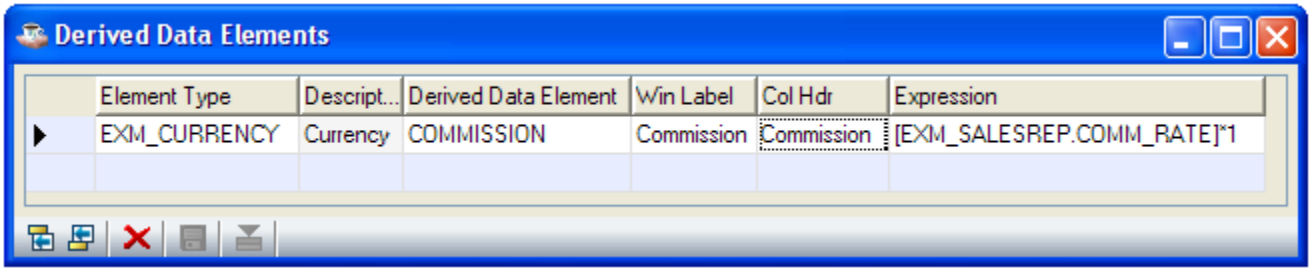
Maintain Auxiliary Columns

To add columns from related tables to the form, select this option (also available via Design□Maintain Auxiliary Columns) and select the required columns from the list. Auxiliary columns must come from a table having the same key as the form's primary table.



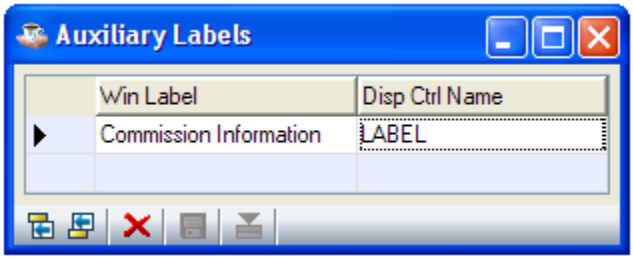
Maintain Derived Data Elements

Derived data elements, as the name implies, are derived from other data elements. To add derived data elements (sometimes called calculated fields), select this option (also available via Design□Maintain Derived Data Elements). After entering the name of the derived data element, select an element type (which determines its overall behavior), then enter an expression that describes how the element is derived. In the expression, other columns in the table are referenced as [*tablenamename.columnname*].



Maintain Auxiliary Labels

To add text labels to the form, select this option (also available via Design□Maintain Auxiliary Labels) and enter the label text.



Maintain Table

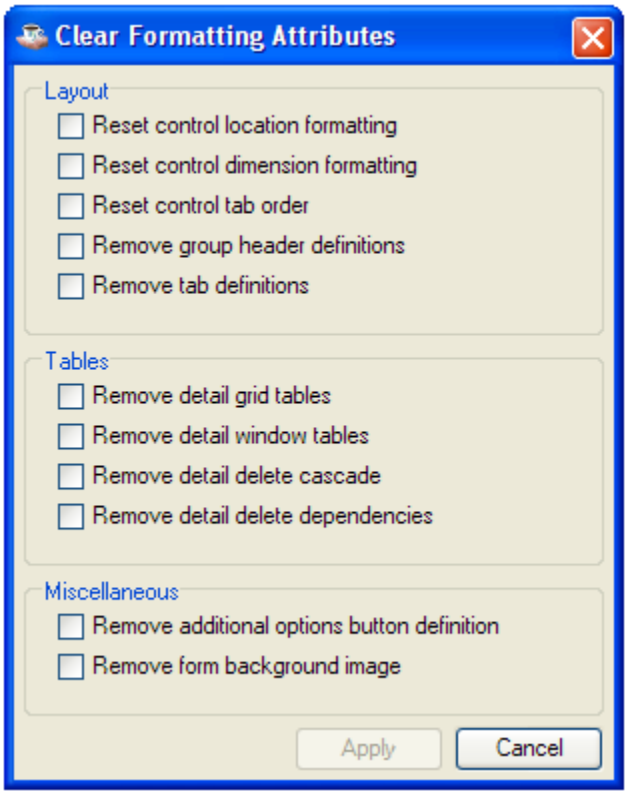
To edit the table for the form, select this option.

Edit Callpoints

To edit callpoint code, select this option. The Callpoint Editor can also be accessed using Design□Edit Callpoints, the F2 key, or the  tool button.

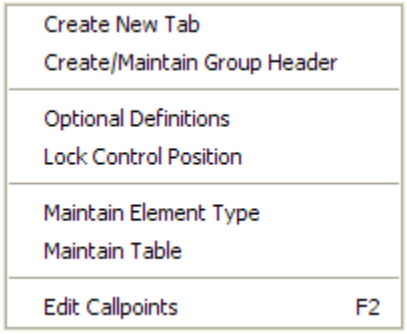
Clear Formatting Attributes

To remove various formatting attributes from the form, select Design□Clear Formatting Attributes from the menu bar, select the attributes to be cleared, and press the Clear button.



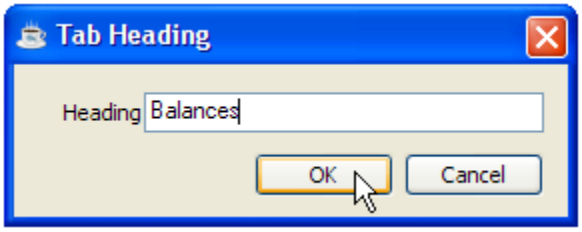
Form Editor Control Options

Right-clicking on a GUI control in the Form Editor brings up the following options:



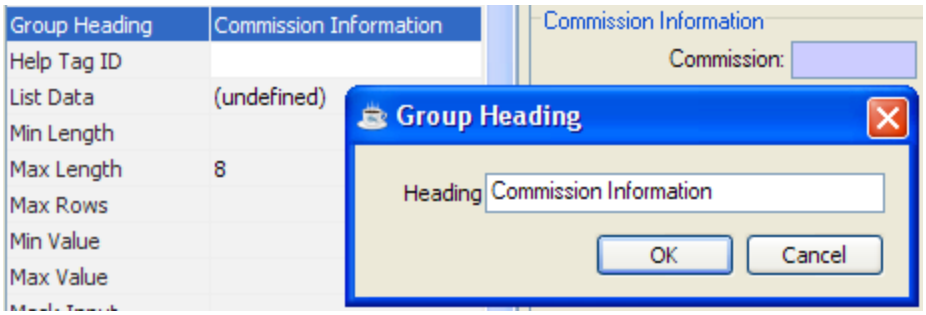
Create New Tab

To add a tab control to the form, or to add a new tab to the existing tab control, select this option and enter a tab heading.



Create/Maintain Group Header

This option is an alternative way to edit the Group Heading attribute.



Optional Definitions

This option is an alternative way to edit the Optional Definitions attributes.

Lock Control Position

Locks the control to its current position and ignores relative formatting rules.

Maintain Element Type

To edit the element type information for the column, select this option.

Maintain Table

To edit the table for the form, select this option.

Edit Callpoints

To edit callpoint code (see below), select this option. The Callpoint Editor can also be accessed using Design□Edit Callpoints, the F2 key, or the  tool button.

Callpoints

The Barista runtime engine takes care of most standard entry, validation and read/write operations automatically. Callpoints can be used to embed call custom code to perform non-standard operations.

During normal processing of entered data, the engine scans the alias attributes, looking for selected callpoint triggers. If a trigger is encountered, the engine calls the specified callpoint routine with a predefined argument list. The callpoint routine performs the necessary operations and returns control to the runtime engine.

Callpoints can also be used to prevent a certain process from occurring. For example, a "Before Record Delete" callpoint can be used to determine if a record deletion is allowed. If the developer-defined callpoint code determines that the deletion should be disallowed, it can pass back an “ABORT” status to the runtime engine.

Table Callpoint Codes (Mainline Processing)

Description	Proc Code	Where Triggered	Uses	Allow Abort?
Before Enable Map	BENA	Before creating the enable/disable controls string.	Adding existing controls to the idle controls string.	
After Enable Map	AENA	After creating the enable/disable controls string.	Enabling/disabling controls outside of normal scope	
After Form Callbacks	ACAL	After settings the event callbacks for all controls on a form.	Adding aditional events	
After Window Creation	AWIN	After issuing the 'WINDOW' mnemonic.	Adding new button or entry controls.	
Before Window Show	BSHO	Just prior to showing the built initial screen.	Adding manual controls or objects.	
After Window Show	ASHO	After completed window is issued a SHOW command. All field arrays have been filled with template data and all applicable fields have been disabled.	Modifying FLD array contents and adding new buttons or fields.	

Before New Record	BREC	Before the field arrays are cleared of data from the previous record.	Manipulating FLD array data and enabling/disabling fields.	
After New Record	AREC	After all arrays and controls have been set for a new record.	Manipulating FLD array data and enabling/disabling fields.	
After Grid Clear	AGCL	After the grid is cleared of all data.	Special grid display or totaling of entered data within the grid.	
Before Record Read	BREA	Prior to the attempted record read.	Read key manipulation.	
After Record Read	AREA	After a successful record read.	Non-standard data retrieval, dynamic field manipulation and field disabling.	
After Key Build	AKEY	After building the key from the contents of the rec$ array.	Manipulating the key.	
After Record Array Transfer	ARAR	After transferring the read data (rd_file_rec$) into record array (rd_rec_data$[]).	Non-standard data retrieval and dynamic field manipulation.	
After Record Display	ADIS	After the record or setup has been initially displayed.	Warning messages for displayed records.	
After Record Removal	AREM	After removal of deleted detail records during save.	Removal of secondary file information.	
After Grid Display	AGDS	After all detail records are displayed in a grid.	Special grid display or totaling of entered data within the grid.	
Before Grid Entry	BGRD	Prior to first entry when focusing on a detail grid.	Grid array manipulation.	
Before Write Array	BWAR	Prior to transferring the contents of the REC$[x] array into the record template (FILE1$)	Modifying the contents of REC$[x] prior to writing.	
Before Record Write	BWRI	Prior to an attempted record write.	Non-standard record validation.	Yes
After Record Write	AWRI	After a successfully completed record write. Bypassed on write errors.	Associated record updating.	
Before Detail Record Save	BSAV	Prior to saving a detail record into the detail record array (DREC$[x])	Data manipulation	Yes
Before Record Removal	BREM	Prior to removal of deleted detail records during save.	Removal key manipulation.	Yes

Table Callpoint Codes (Record Delete)

Description	Proc Code	Where Triggered	Uses	Allow Abort?
Before Record Delete Query	BDEQ	Prior to delete record prompt.	Processing to ensure delete is allowed.	Yes
Before Record Delete	BDEL	Prior to attempted record deletion.	Delete key manipulation.	Yes
After Record Delete	ADEL	After an attempted or successful record deletion.	Cascading record deletion.	

Table Callpoint Codes (Miscellaneous)

Description	Proc Code	Where Triggered	Uses	Allow Abort?
Before Next Record Key	BNEK	Prior to retrieving the key for the next record.	Read position and key manipulation. Tables with mixed multiple record types.	
Before Next Record	BNEX	Prior to retrieving the next record.	Read key manipulation.	
Before Previous Record Key	BPRK	Prior to retrieving the key for the previous record.	Read position and key manipulation. Tables with	

			mixed multiple record types.	
Before Previous Record	BPRI	Prior to retrieving the previous record	Read key manipulation.	
Before File Overview	BOVE	Prior to calling the standard inquiry system for a file overview.	Non-standard record prefix manipulation.	Yes
After File Overview	AOVE	After calling the standard inquiry system for a file overview.	Retrieved data manipulation.	
After Print All Selected	APRA	After selecting the “Print All” option from the menu bar.	Specialized reports formats. Note: If a custom output program is called, a status of “ABORT” must be used stop Barista from attempting to print its own version.	Yes
After Print Record Selected	APRT	After selecting the “Print Record” option from the menu bar.	Specialized reports formats. Note: If a custom output program is called, a status of “ABORT” must be used stop Barista from attempting to print its own version.	Yes
Before Record Copy	BRCO	Prior to the record copy setup.	Current record manipulation or copy validation checking.	Yes
After Record Copy	ARCO	After a successful record copy.	Copied record manipulation.	
After Grid Exit	AGRD	After a user tabs out of a detail grid.	Extra grid validation or totaling of entered data within the grid.	
Before Program Exit	BEND	Prior to exiting the program. All associated windows for the program have already been closed.	Erasing temporary files and setting processing flags.	Yes
After Window Resize	ASIZ	After receiving a window resize event.	Resizing added tabs and child windows.	

Column/Field Callpoint Codes

Description	Proc Code	Where Triggered	Uses	Allow Abort?
After Field Input	AINP	After losing focus and performing base validation (length/field type).	Non-standard field validation.	
After Field Inquiry	AINQ	After calling the standard inquiry system for a field inquiry.	Retrieved data manipulation.	
After Control Modification	AMOD	After receiving a modified event for a BBjSlider or BBjInputE/N/DSpinner,	Using the current modified contents of a field for subsequent data or display manipulation.	
After Field Validation	AVAL	After performing ALL defined validation.	Using the validated contents of a field for subsequent data manipulation.	
Before Drill Down	BDRL	Prior to displaying drill down inquiry	Filter manipulation.	Yes
Before Field Input	BINP	When a specific control gains focus.	Non-standard field defaults.	
Before Field Inquiry	BINQ	Prior to calling the standard inquiry system for a field inquiry.	Non-standard record prefix manipulation.	Yes

The Callpoint! Object

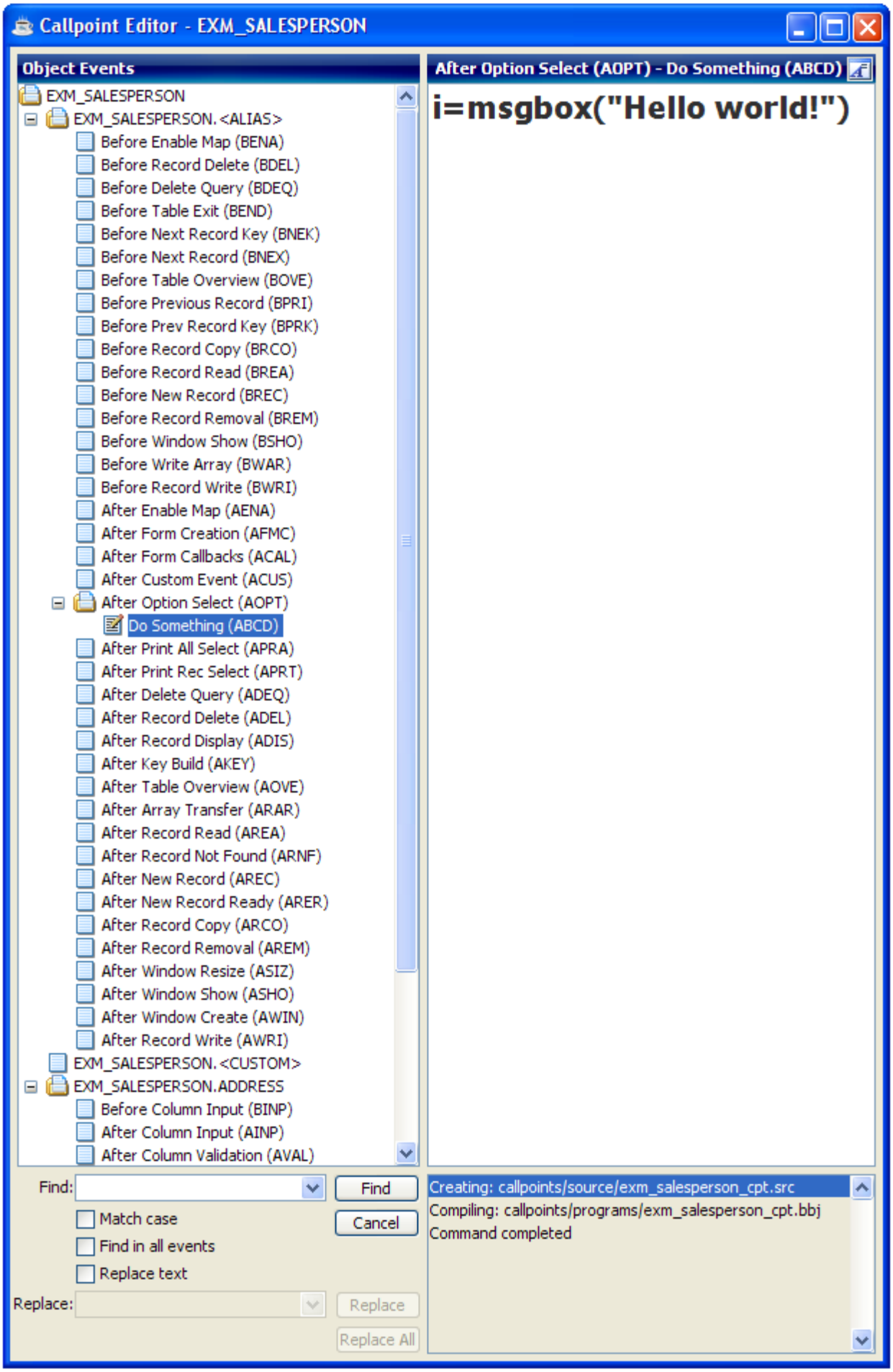
The developer can include any BBj code in a callpoint routine, but most callpoints will make use of the Callpoint! object, which provides access to commonly used information. For a list of available methods, refer to Callpoint Object Methods (CMTH), below.

Callpoint Editor

The Callpoint Editor is a text editor for entering BBj program code to be executed at a particular point during the form execution. To use the Callpoint Editor, select Design□Edit Callpoints, press the F2 key, or click the  tool button. The Callpoint Editor includes:

- the ability to increase or decrease the indent level for a block of code using the  and  tool buttons.
- search and replace options;
- a font selector to change the preferred display font; and
- a right-click menu for selecting common code blocks from a list.

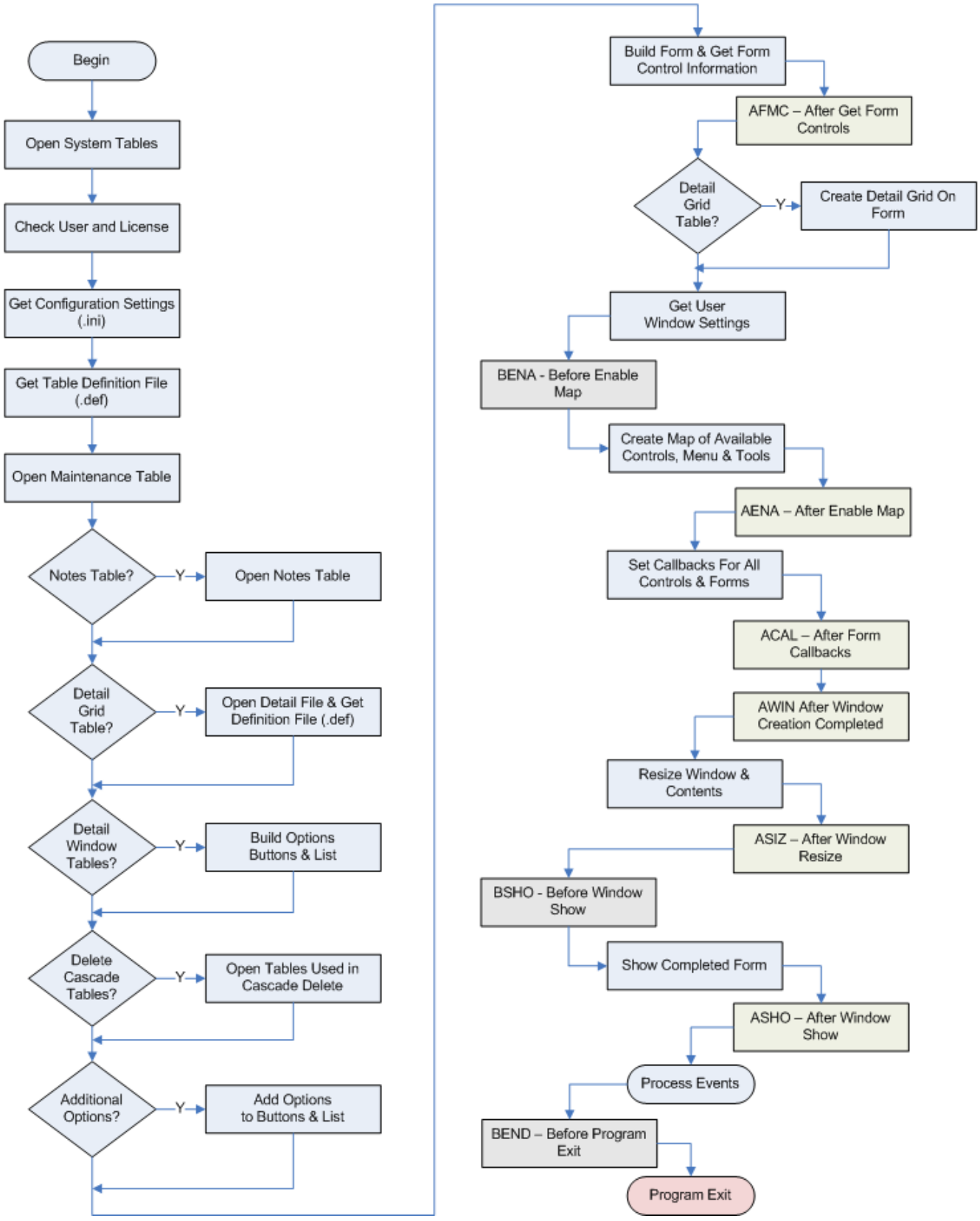
The following sample callpoint code might be invoked in response to the user selecting a particular option using the  tool button:


Callpoint Editor - EXM_SALESPERSON
Object Events
EXM_SALESPERSON
EXM_SALESPERSON.<ALIAS>
Before Enable Map (BENA)
Before Record Delete (BDEL)
Before Delete Query (BDEQ)
Before Table Exit (BEND)
Before Next Record Key (BNEK)
Before Next Record (BNEX)
Before Table Overview (BOVE)
Before Previous Record (BPRI)
Before Prev Record Key (BPRK)
Before Record Copy (BRCO)
Before Record Read (BREA)
Before New Record (BREC)
Before Record Removal (BREM)
Before Window Show (BSHO)
Before Write Array (BWAR)
Before Record Write (BWRI)
After Enable Map (AENA)
After Form Creation (AFMC)
After Form Callbacks (ACAL)
After Custom Event (ACUS)
After Option Select (AOPT)
Do Something (ABCD)
After Print All Select (APRA)
After Print Rec Select (APRT)
After Delete Query (ADEQ)
After Record Delete (ADEL)
After Record Display (ADIS)
After Key Build (AKEY)
After Table Overview (AOVE)
After Array Transfer (ARAR)
After Record Read (AREA)
After Record Not Found (ARNF)
After New Record (AREC)
After New Record Ready (ARER)
After Record Copy (ARCO)
After Record Removal (AREM)
After Window Resize (ASIZ)
After Window Show (ASHO)
After Window Create (AWIN)
After Record Write (AWRI)
EXM_SALESPERSON.<CUSTOM>
EXM_SALESPERSON.ADDRESS
Before Column Input (BINP)
After Column Input (AINP)
After Column Validation (AVAL)
After Option Select (AOPT) - Do Something (ABCD)
i=msgbox("Hello world!")
Find:
Find
Cancel
Match case
Find in all events
Replace text
Replace:
Replace
Replace All
Creating: callpoints/source/exm_salesperson_cpt.src
Compiling: callpoints/programs/exm_salesperson_cpt.bbj
Command completed

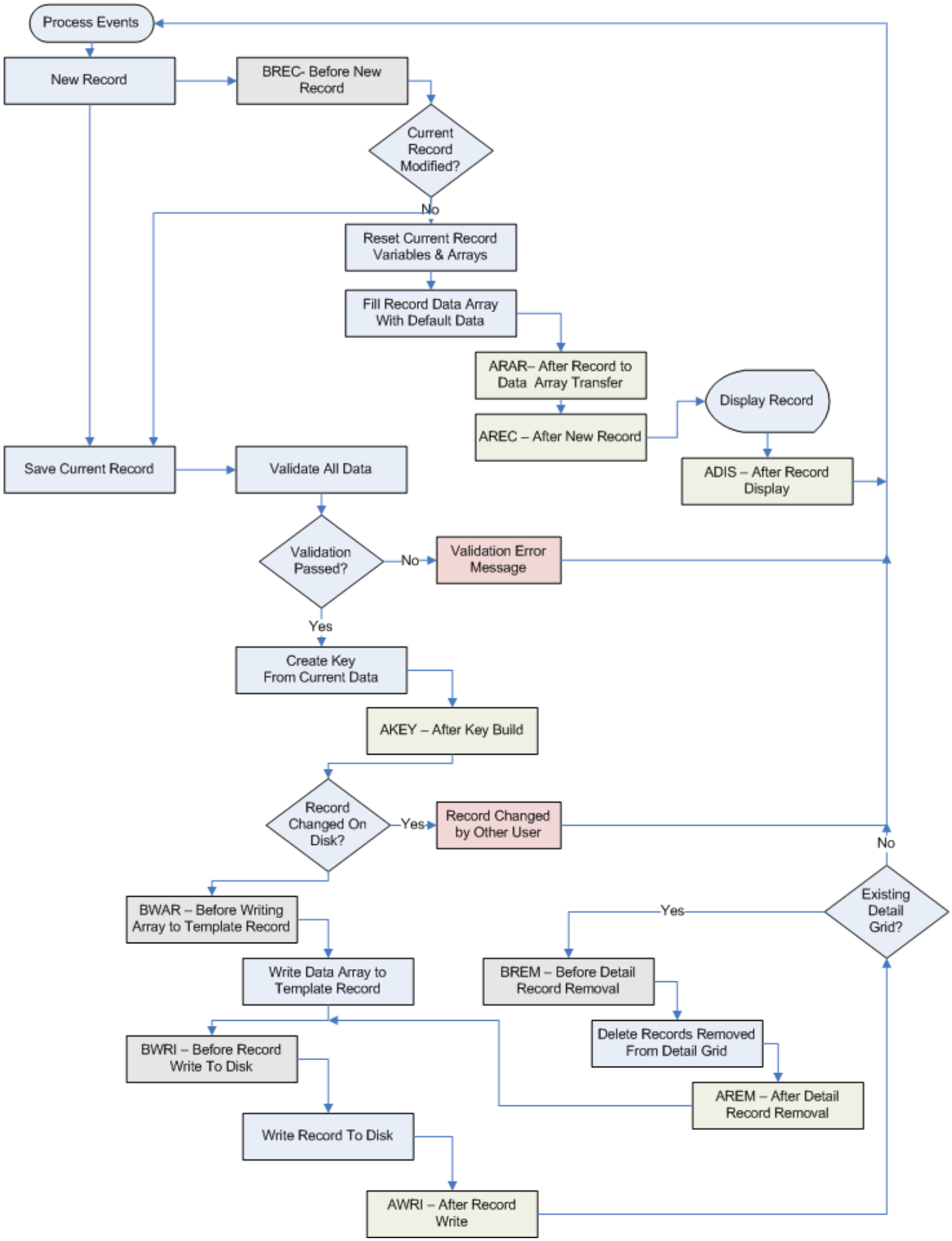
Flow Diagrams

The following diagrams show where callpoint code is executed within the execution of a Barista form.

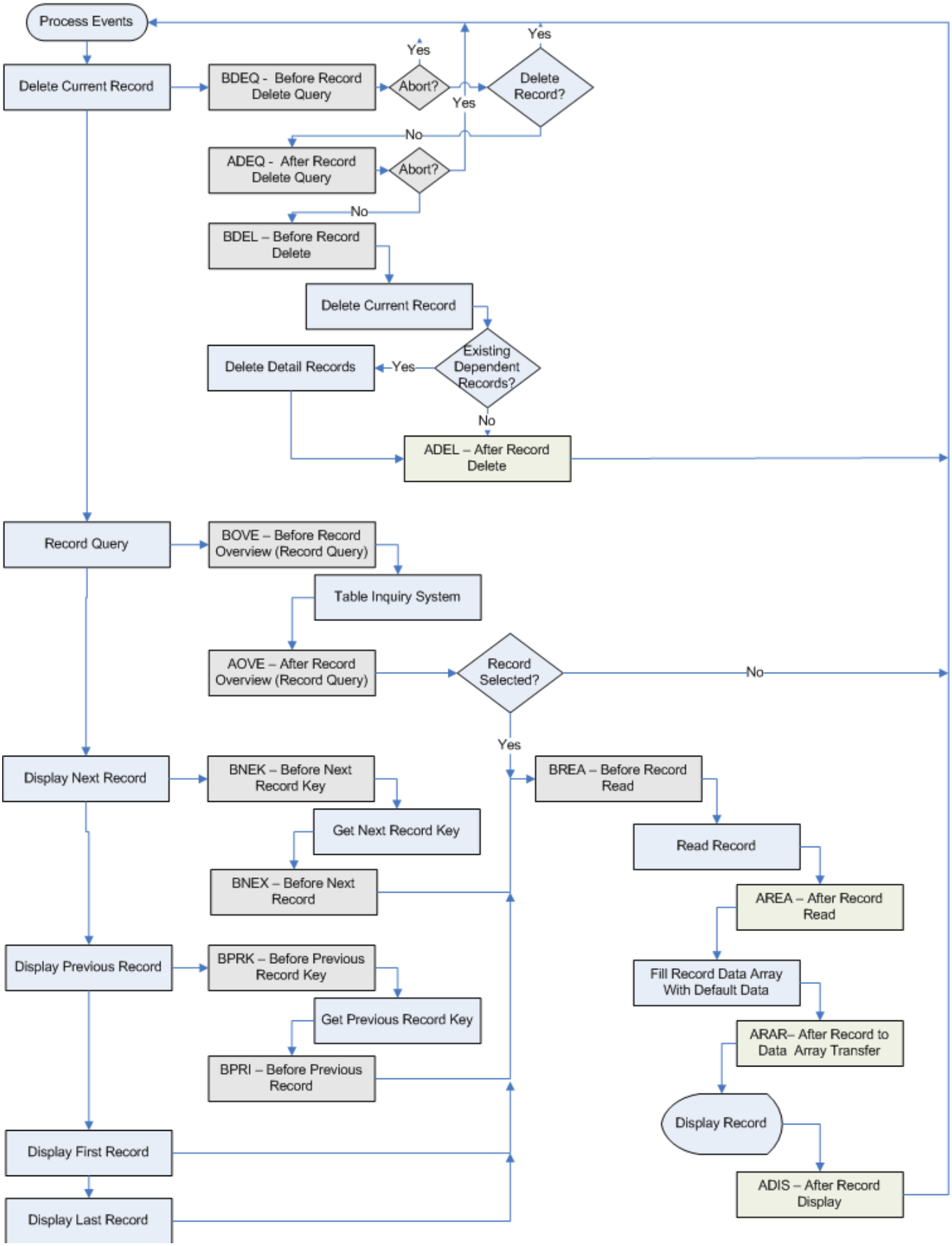
Form Maintenance Flow – Overview



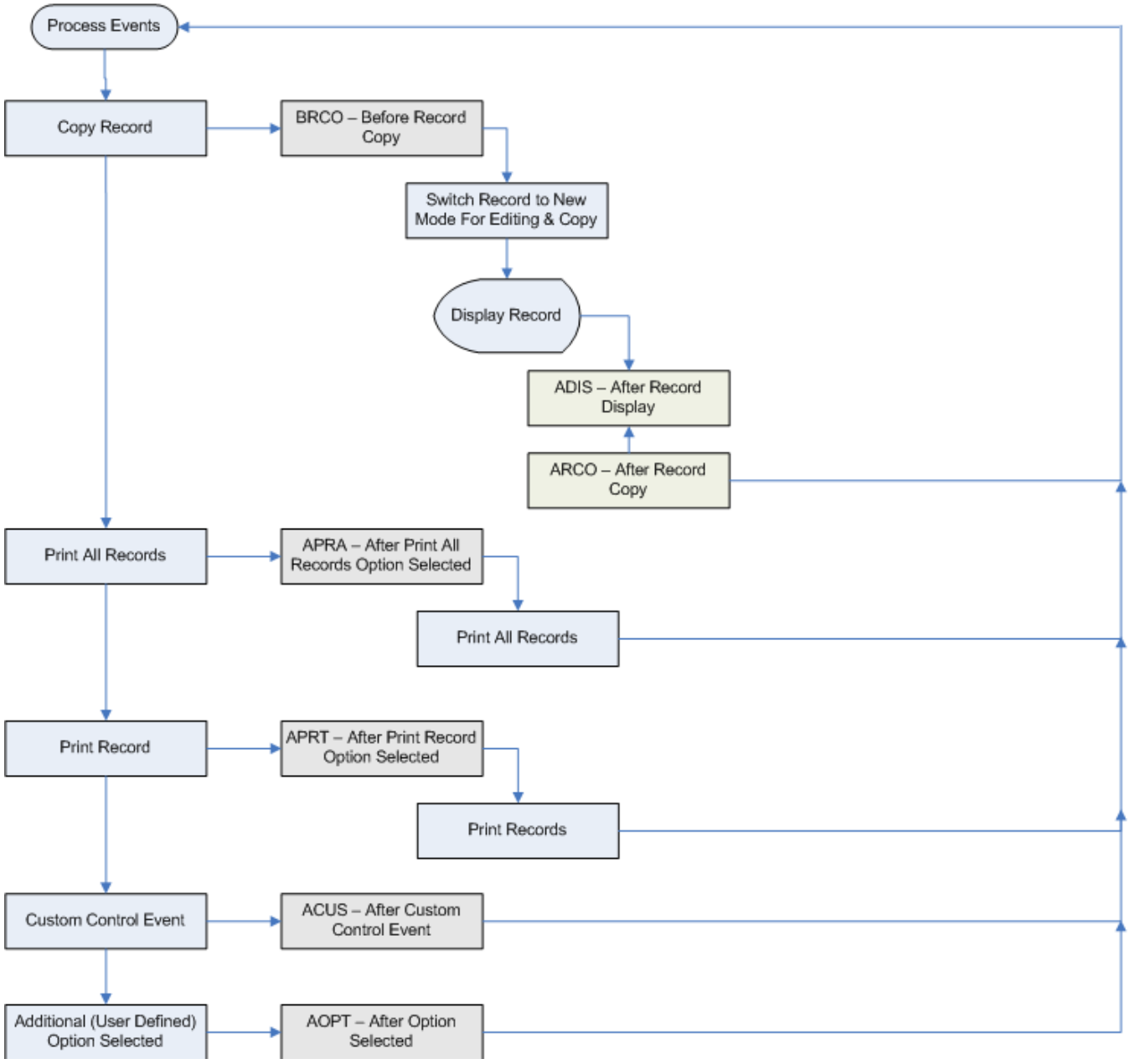
Form Maintenance Flow – Create or Update Record



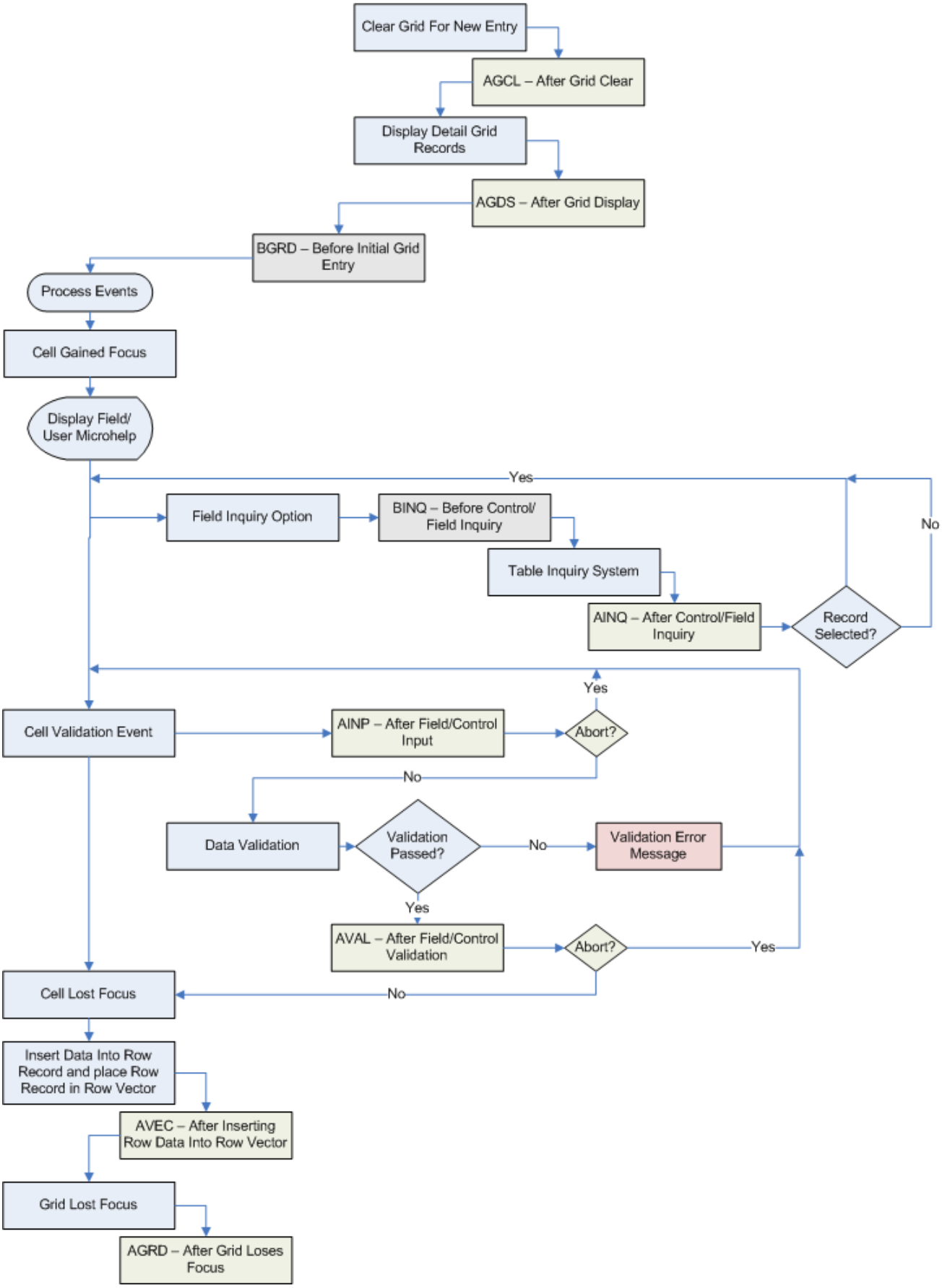
Form Maintenance Flow – Delete, Query, Next, Previous, First, Last



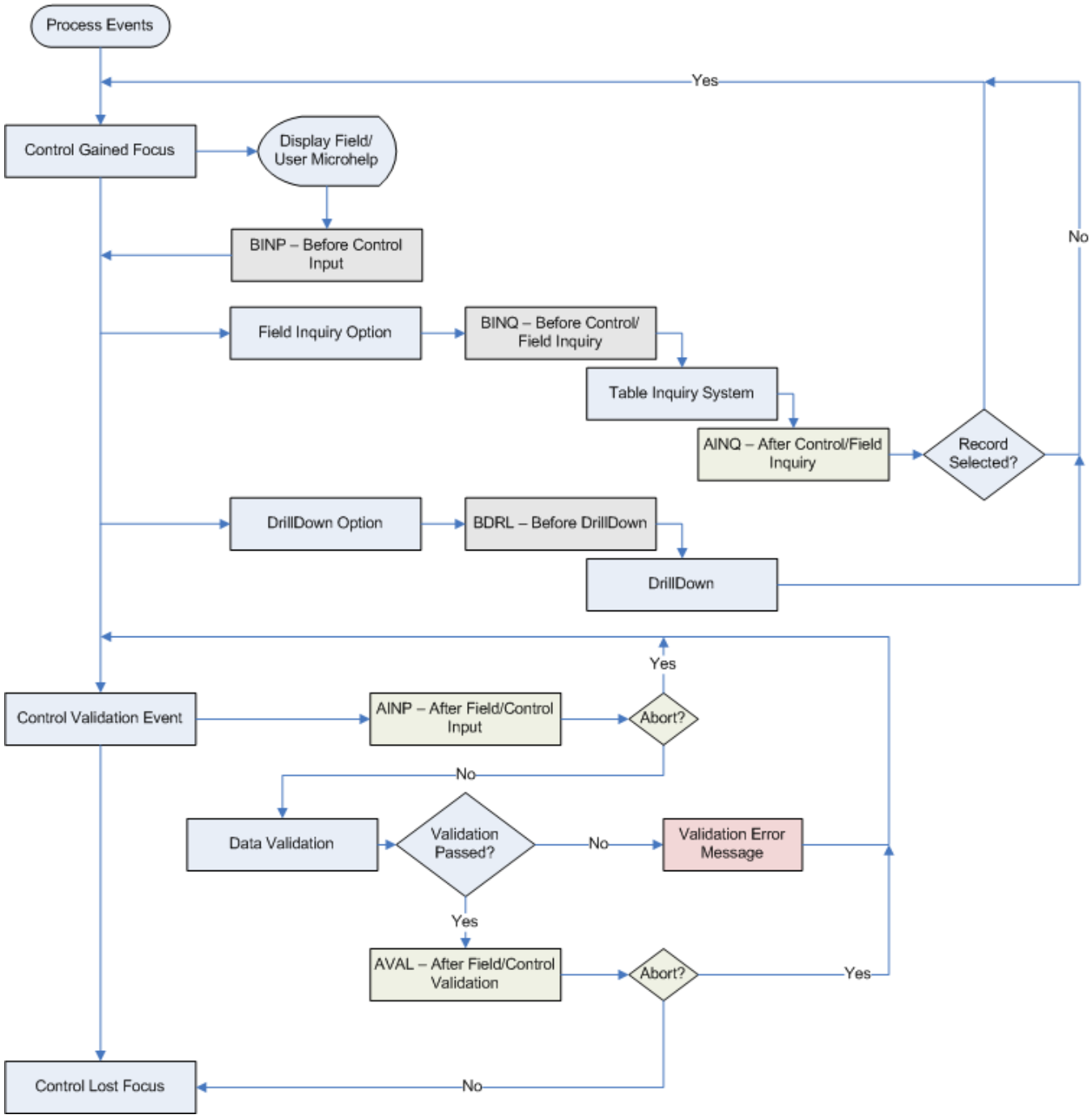
Form Maintenance Flow – Copy, Print, and Custom Options



Grid Maintenance Flow – Overview

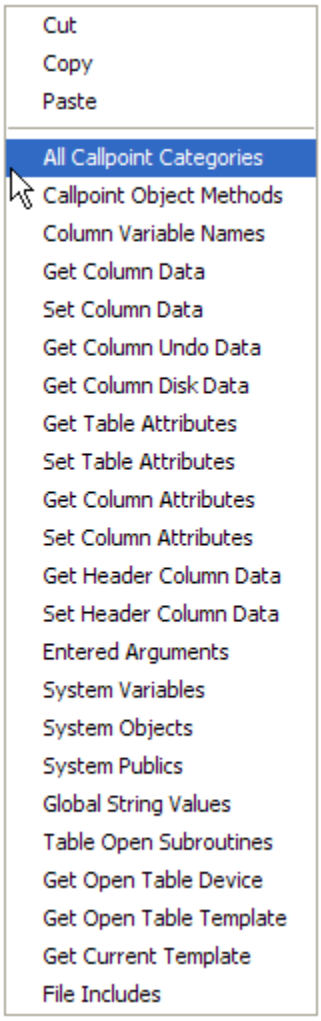


Column Entry Flow Detail



Callpoint Code Fragments

For easy access to commonly used code fragments, click the  tool button or right-click in the Callpoint Editor to see the following list of callpoint code categories:

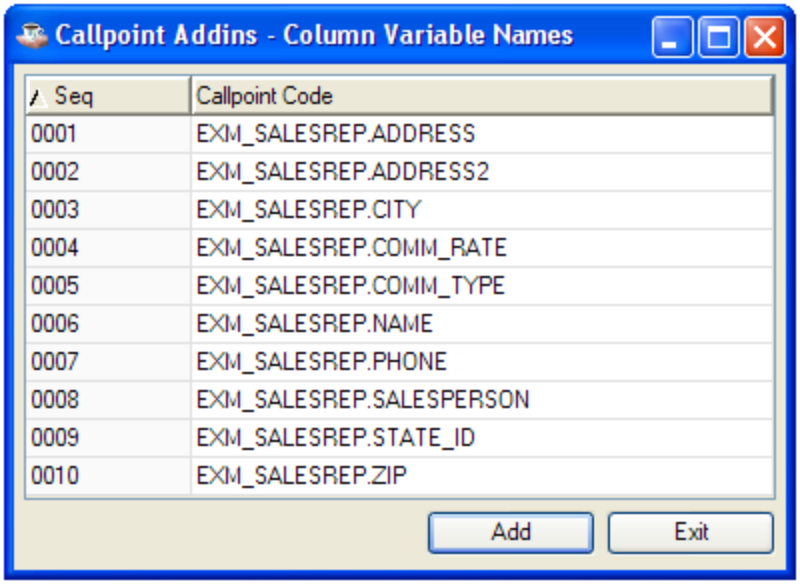


Callpoint Object Methods (CMTH)

Code	Notes
=callpoint!.getRecordMode()	Contains current record mode: (<A>dd mode, <C>hange mode)
=callpoint!.getRecordStatus()	Set to 'M' if current record is modified
=callpoint!.getAlias()	Alias triggering the callpoint.
=callpoint!.getType()	Callpoint type being triggered: (<T>able, <C>olumn, <D>etail, <G>rid)
=callpoint!.getEvent()	<alias_name>.<variable_name>.<event> triggering the callpoint
=callpoint!.getControlID()	ID of the control triggering the callpoint
=callpoint!.getVariableName()	<alias_name>.<variable_name> of the column triggering the callpoint
=callpoint!.getUserInput()	User input data for the control triggering the callpoint
=callpoint!.getRecordKey()	The primary key of the record triggering the callpoint
=callpoint!.getKeyPrefix()	The primary key prefix of the the record triggering the callpoint
=callpoint!.getRecordTemplate()	The template of the record triggering the callpoint
=callpoint!.getAbleMap()	Contains the control enable/disable map
callpoint!.setAbleMap(<_>)	Returns the control enable/disable map
callpoint!.setStatus(<_>)	Sets return status: (Any combination of: <ABORT> process, <MODIFIED>, <CLEAR> form, <REFRESH> display)
callpoint!.setMessage(<_>)	Display indicated message when callpoint exits (<message id:token 1;token2>)

Column Variable Names (COLS)

A list of all data elements in the table, in the format ALIAS.DATA_ELEMENT. For example:

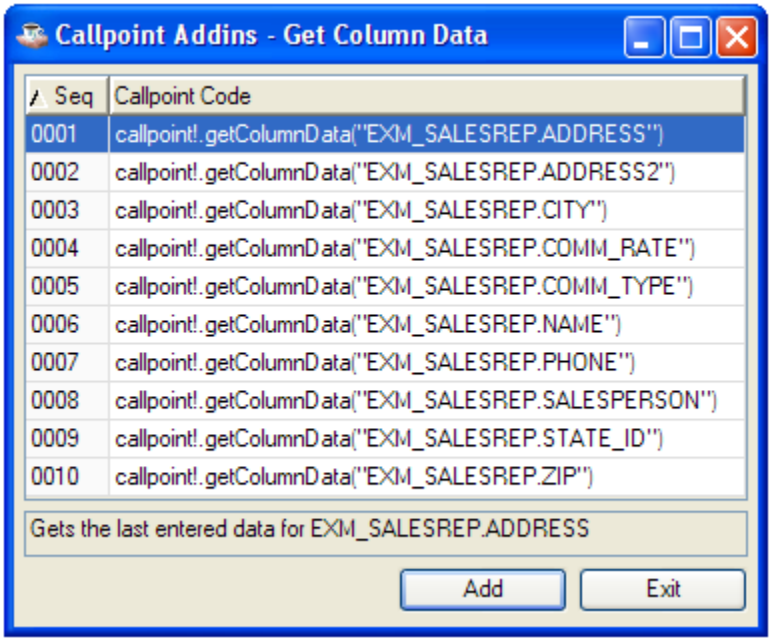


Get Column Data (GETC)

The code used to retrieve column data for a particular field, in the format:

callpoint!.getColumnData("*fieldname*")

For example:

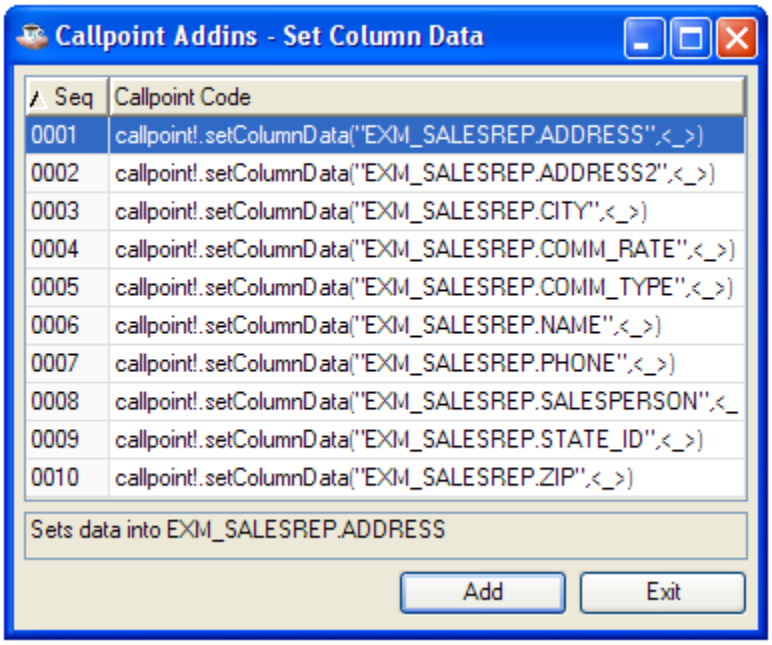


Set Column Data (SETC)

The code used to update column data for a particular field, in the format:

callpoint!.setColumnData("*fieldname*",*value*)

For example:

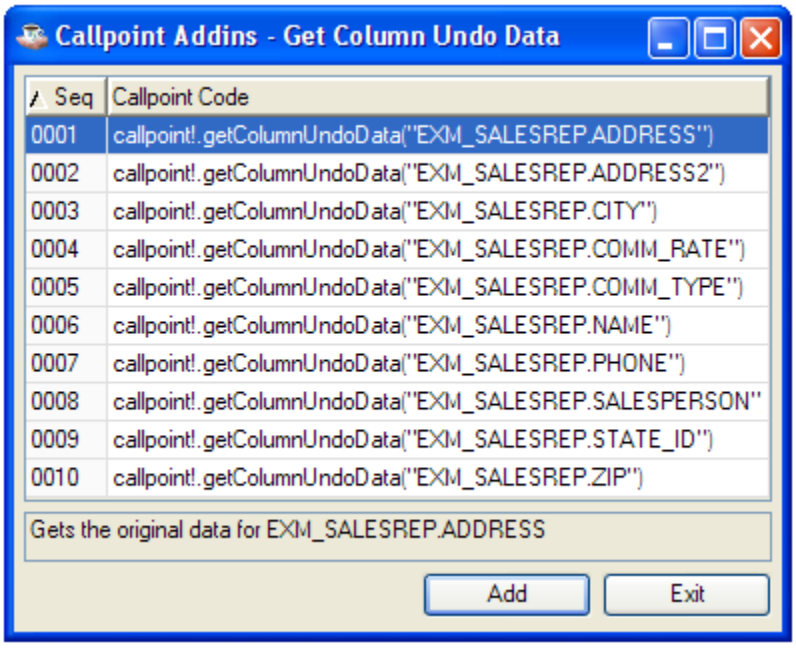


Get Column Undo Data (GETU)

The code used to retrieve the original value of a particular field, in the format:

callpoint!.getColumnUndoData("*fieldname*")

For example:

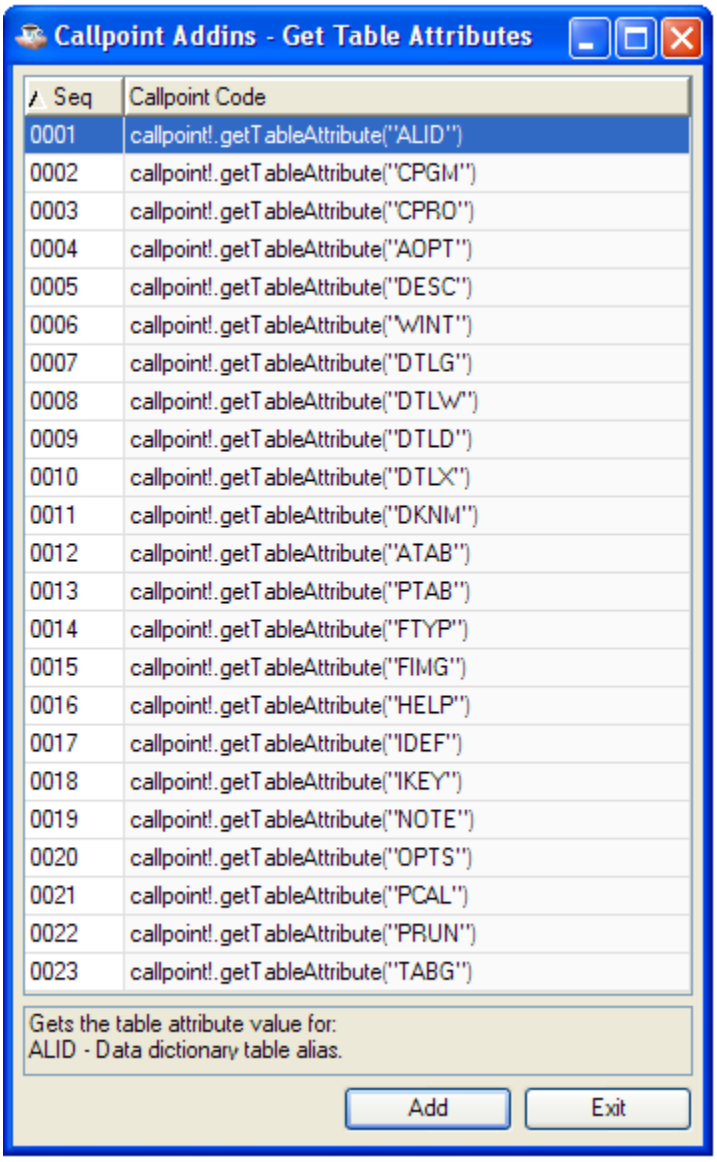


Get Table Attributes (GTAV)

The code used to retrieve a table attribute, in the format:

callpoint!.getTableAttribute("*code*")

For example:



For a complete list of table attribute codes, see

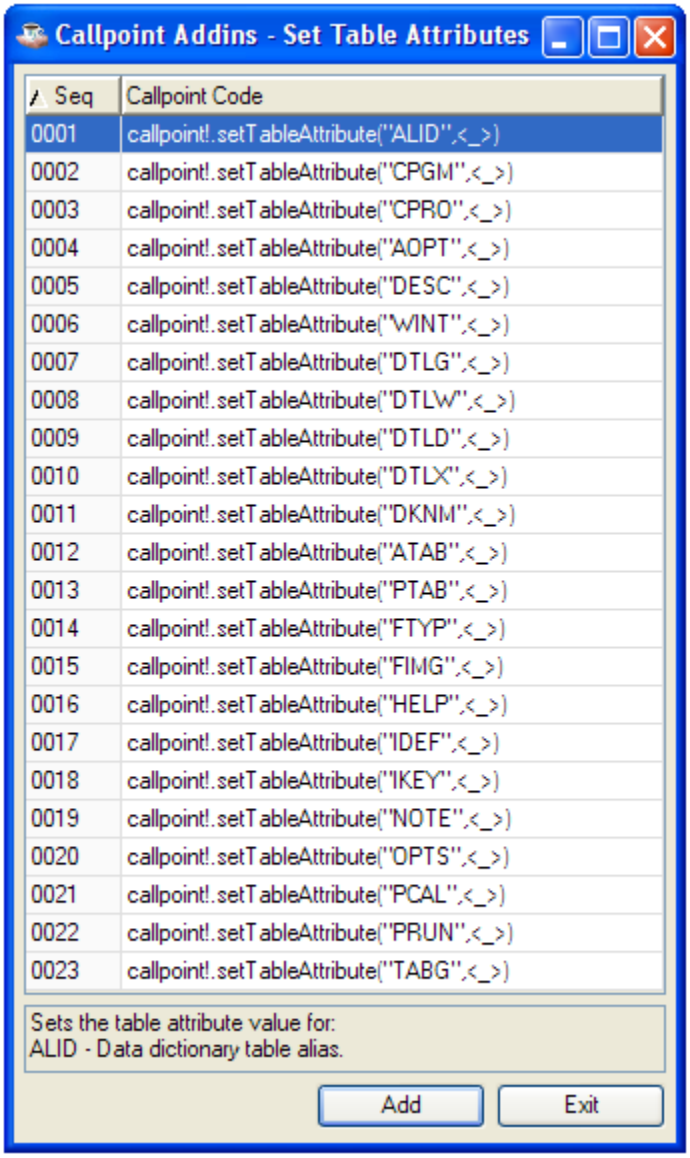
Table Attribute Codes.

Set Table Attributes (STAV)

The code used to update a table attribute, in the format:

callpoint!.setTableAttribute("*code*","value")

For example:



For a complete list of table attribute codes, see

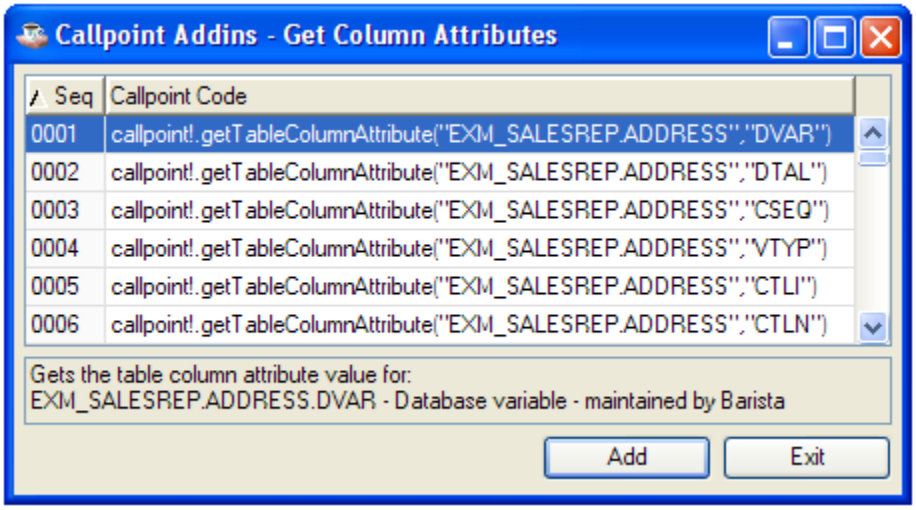
Table Attribute Codes.

Get Column Attributes (GCAV)

The code used to retrieve a column attribute, in the format:

callpoint!.getTableColumnAttribute("*fieldname*","*code*")

For example:



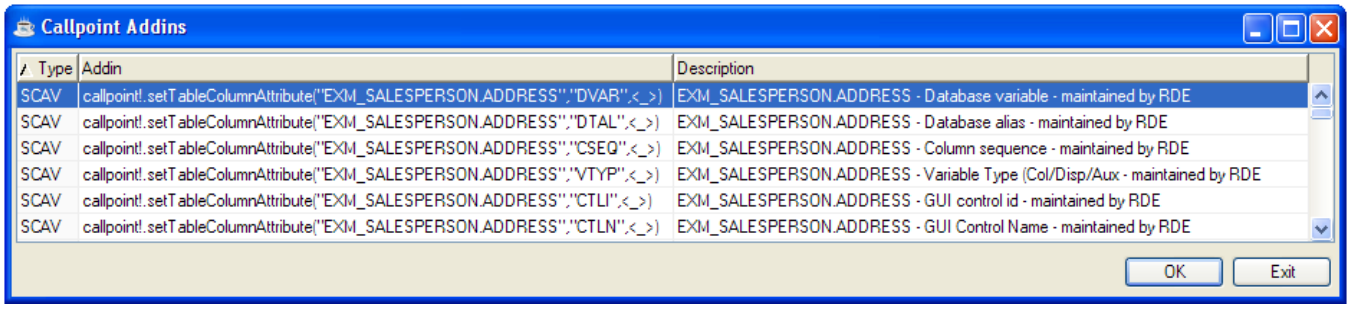
For a complete list of column attribute codes, see Column Attribute Codes.

Set Column Attributes (SCAV)

The code used to retrieve a column attribute, in the format:

callpoint!.getTableColumnAttribute("*fieldname*","*code*")

For example:



For a complete list of column attribute codes, see Column Attribute Codes.

Entered Arguments (ENTA)

Code	Notes
bar_tpl$	Template defined and reserved by Barista
BarObj!	Object defined and reserved by Barista
user_tpl$	User defined template. May be used to store data such as opened channel numbers, etc
UserObj!	User defined object. May be used to store vectors, controls or other objects
gui_dev	Variable specifying the opened sysgui channel
SysGUI!	The SysGUI object retrieved from the BBjAPI()
Form!	The alias form
rec_data$	String template containing the current record data

table_chans$[ ]	Array containing all opened tables and the associated string templates
dtlg_param$[]	
GridVect!	
hrec_data$[]	

System Variables (SVAR)

Code	Notes
bar_tpl$	Template defined and reserved by Barista
user_tpl$	User defined template. May be used to store data such as opened channel numbers, etc
gui_dev	Variable specifying the opened sysgui channel
rec_data$	String template containing the current record data
table_chans$[ ]	Array containing all opened tables and the associated string templates
dtlg_param$[]	Array containing information details for the Detail grid in a Header/Detail form. dtlg_param$[1:#tabs+1,10]

Index	Description	Variable name
0	Table ID (table name)	rd_dtlg_tbl
1	Tab ID (index)	rd_dtlg_tab
2	File channel	rd_dtlg_chn
3	File template	rd_dtlg_tpl
4	Future use	
5	Future use	
6	Future use	
7	Future use	
8	Detail grid control ID	rd_dtlg_ctlid
9	Detail grid header control ID	rd_dtlg_hdr_ctlid
10	Detail grid window control ID	rd_dtlg_parent_ctlid

System Objects (SOBJ)

Code	Notes
BarObj!	Object defined and reserved by Barista
UserObj!	User defined object. May be used to store vectors, controls or other objects
SysGUI!	The SysGUI object retrieved from the BBjAPI()
Form!	The alias form
GridVect!	A vector containing grid data vectors. getItem(0) contains a vector of all row data contained in the detail grid.

Global String Values (STBL)

Variable	Notes
+ADDATA	Data location for Barista Administrator files
+BACKGROUND_CTL	
+BLOCK_CTL	Starting control number sequence for block controls
+BLOCK_TEXT_CTL	Starting control number sequence for block text controls
+BUTTON_CTL	Starting control number sequence for options button controls
+CHILD_WIN	
+CUSTOM_CTL	
+DATE_FORMAT	
+DATE_GRID	
+DATE_MASK	
+DBNAME	
+DDDATA	Data location for Barista Data Dictionary files
+DIR_ADM	Location of Barista administrative files

+DIR_BWU	Location of Barista Web Utility files
+DIR_CFG	
+DIR_CPT	
+DIR_DAT	
+DIR_DDB	
+DIR_DEF	Location of Barista definition files
+DIR_FRM	Location of Barista files
+DIR_IMG	Location of Barista image files
+DIR_PGM	Location of programs files
+DIR_SYP	Location of Barista programs files
+DIR_SYS	Location of Barista system forms
+DIR_USR	
+DIR_WRK	Location of Barista work files
+DISPLAY_CTL	Starting control number sequence for secondary data display controls
+DOC_DIR_CSV	Document warehouse folder for Comma Separated Value text files (.csv)
+DOC_DIR_HTM	Document warehouse folder for HTML text files (.htm)
+DOC_DIR_PDF	Document warehouse folder for Portable Document File text files (.pdf)
+DOC_DIR_TXT	Document warehouse folder for Tab Delimited text files (.txt)
+DOC_DIR_XML	Document warehouse folder for XML text files (.xml)
+DOC_FONT_FIXED	Default font name for non-proportional output in the Document Output Display
+DOC_FONT_PROP	Default font name for proportional output in the Document Output Display
+DOC_FONT_SIZE	Default font size for output in the Document Output Display
+ENTRY_CTL	Starting control number sequence for entry controls
+FIELD_DELIM	
+FILE_ADD	Location of callpoint addins definition file
+FILE_CFG	Location of Barista configuration file
+FILE_CPT	Location of callpoint stub program
+FILE_HLP	Location of help system
+FILE_MEN	Location of menu system file
+FILE_SET	Location of user window settings file
+FILE_TPM	Location of data dictionary definition file
+FILE_USR	
+GRID_CTL	
+GRID_LIST_CTL	
+GUI_DEVICE	
+IMAGE	
+IMAGE_CTL	
+LANGUAGE_ID	
+MASTER_USER	
+MILESTONE	Default milestone for updating progress meters
+PROGRESS_CTL	
+STATUS_CTL	
+STATUS_IND_CHAR	Character used to indicate a 'checked' value in columnar displays
+TAB_CTL	
+USER_ID	The current user id
+TEXT_CTL	Starting control number sequence for label controls
+WILDCARD_CHARS	
+WINDOW_RES	

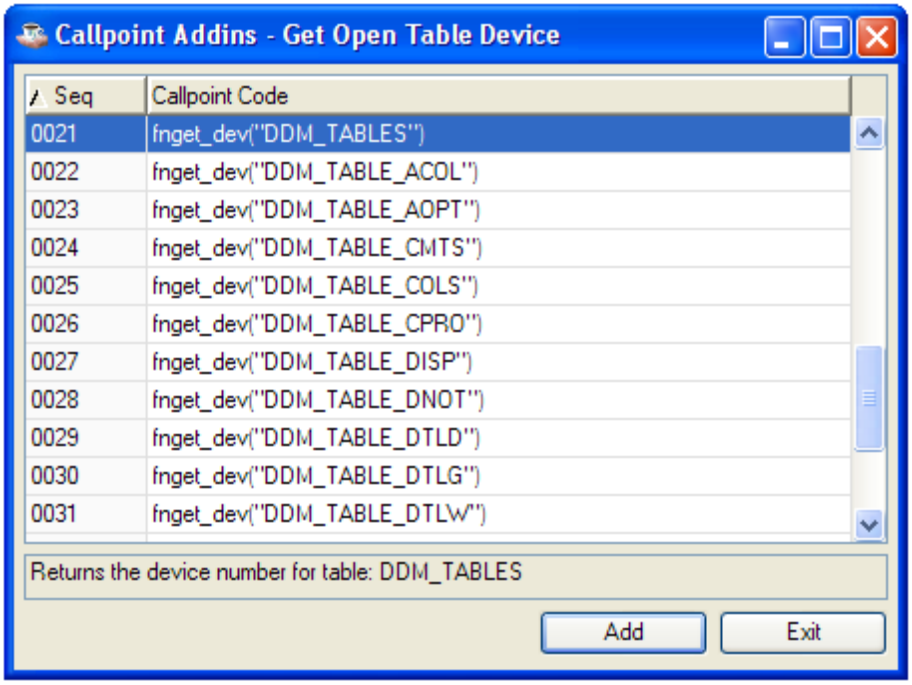
Table Open Subroutines (TBLO)

Code	Notes
num_files=1	Specified the number of table open elements to create
dim open_tables$[1:num_files],open_opts$[1:num_files], open_chans$[1:num_files],open_tpls$[1:num_files]	Dims the required table open elements
open_tables$[1]="<TABLE_ID>",open_opts$[1]="OTA"	Assigns the table id and open parameters
gosub open_tables	Accesses the table open subroutine

<table_id>_chn=num(open_chans$[1]),<table_id>_tpl$=open_tpls$[1]	Assigns the return open information to local variables

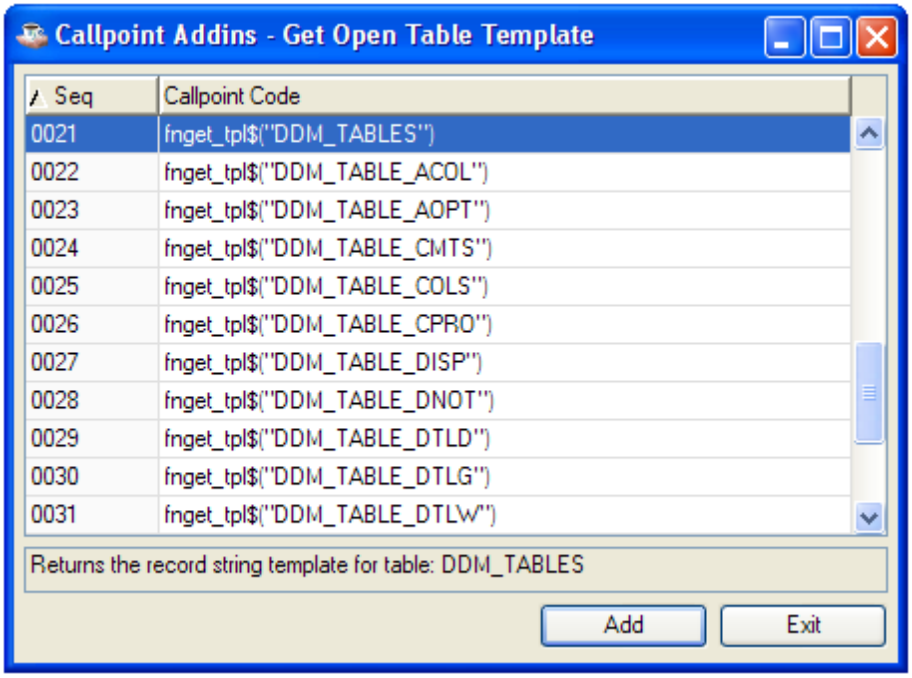
Get Open Table Device (TDEV)

Returns the channel number of a selected internal control file. For example:



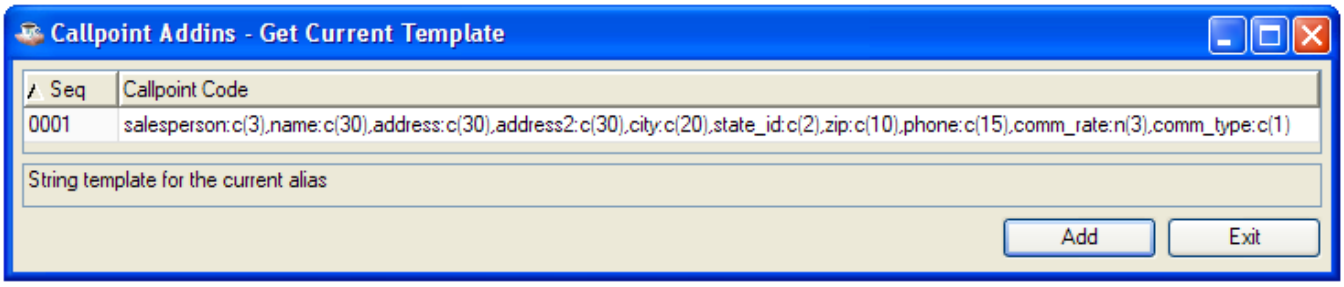
Get Open Table Template (TTPL)

Retrieves the template for a selected internal control file. For example:



Get Current Template (CTPL)

Retrieves the template for this table. For example:

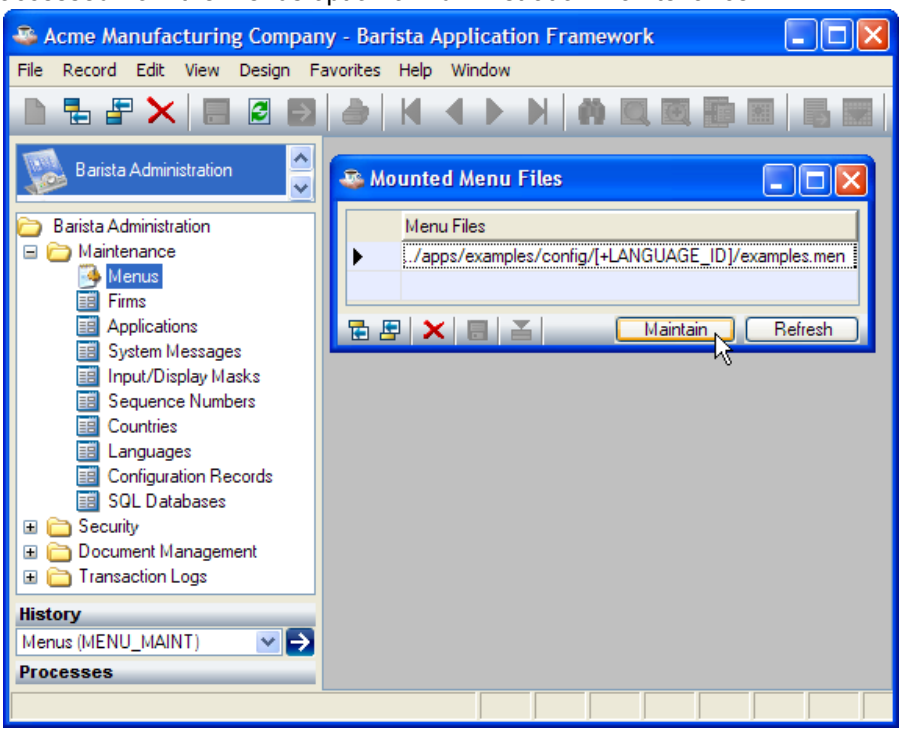


File Includes (INCS)

Code	Notes
#include std_missing_params.src	Standard Barista routine for handling missing parameters
#include std_function.src	Standard Barista form functions
#include std_error.src	Standard Barista error routine
#include std_exit.src	Standard Barista routine for exiting programs

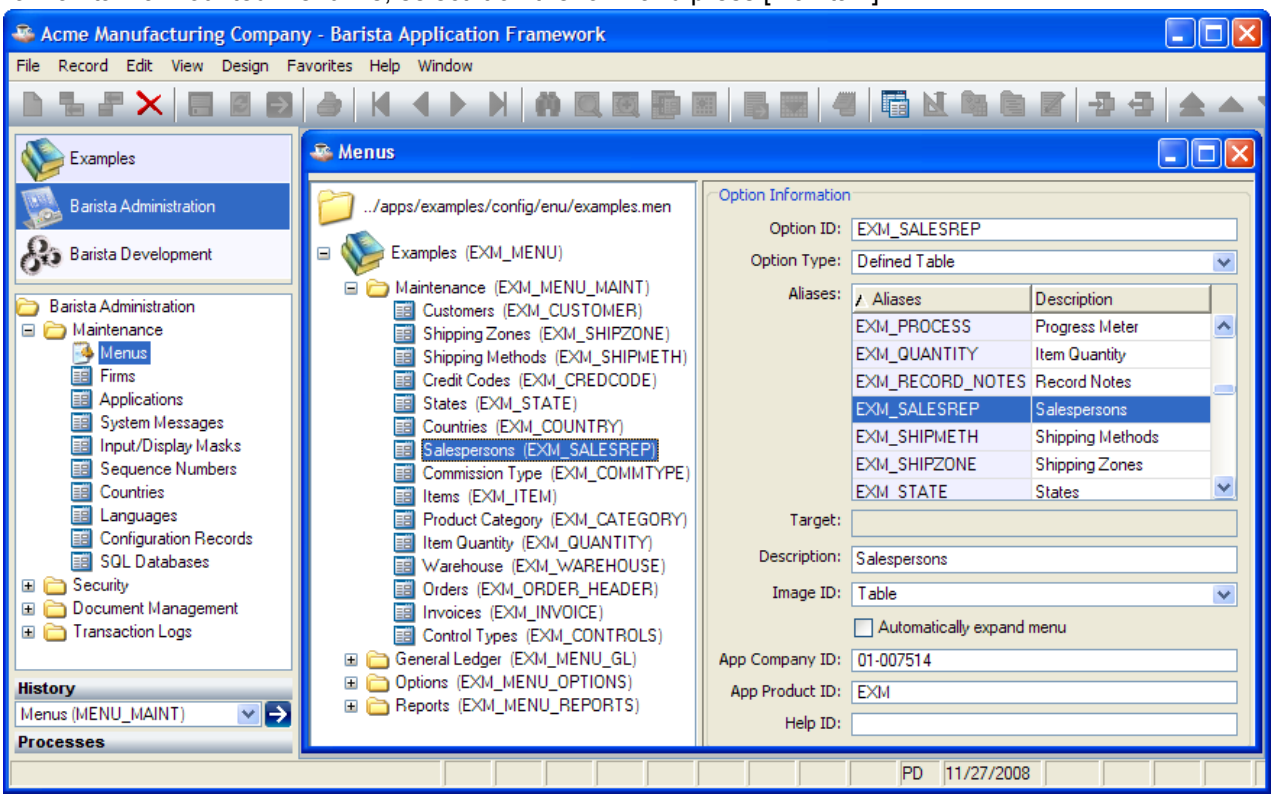
Menu Designer

The application menu system is the panel that usually appears on the left side of the MDI window. This is distinct from the Barista menu bar, which appears at the top of the MDI window, above the tool bar. To make a Barista form available to the end user, it must be added to the menu system. This is done in the Menu Designer, which is accessed from the Menus option of Administration Maintenance:



Barista uses multiple menu files when compiling and displaying the application menu system. The Mounted Menu Files form allows addition and deletion of these menu files. Note, the entry order of the menu files on this form dictates the display sequence in the application menu.

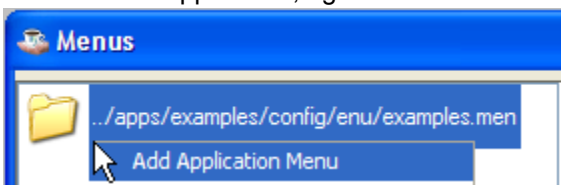
To refresh the application menu display after maintaining the menu files, press [Refresh].
To maintain a mounted menu file, select it on the form and press [Maintain]:

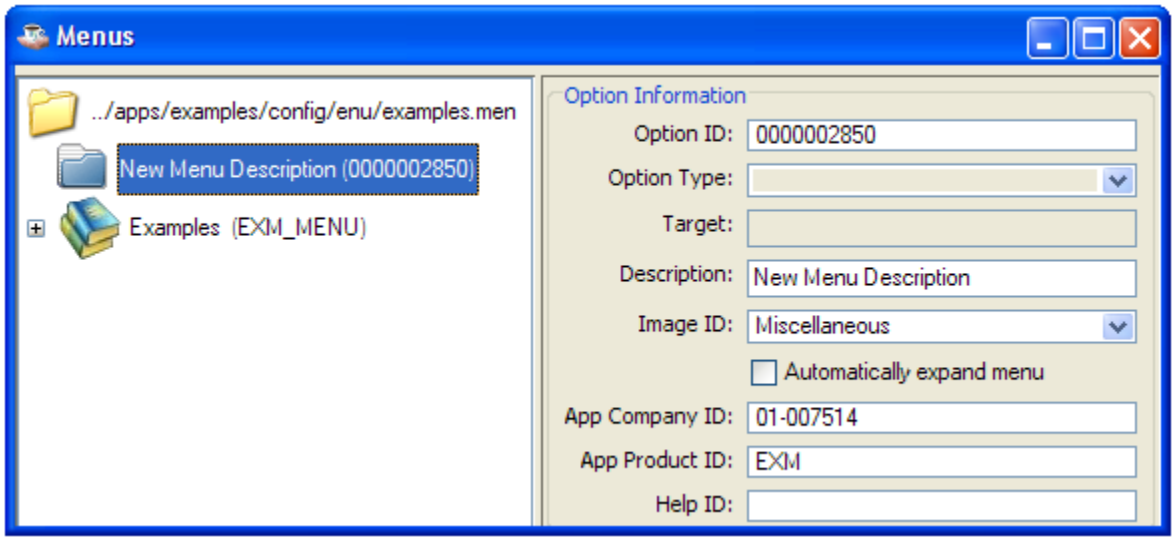


The top level of the application menu system, Applications, is shown in the top of the panel. The menu system for a given application is shown in the middle part of the panel. To add a new item to the menu system, either right-click on an existing item, or select an existing item and click the  tool button.

Add Application Menu

To add a new application, right-click on "Barista Application Framework" and select "Add Application Menu":

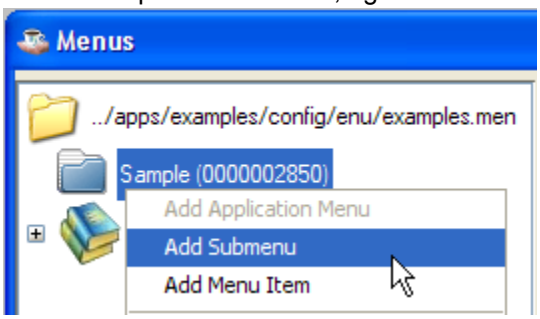




Option ID	The Option ID is a unique reference, up to 20 characters, that identifies a menu system entry. When defining a new Application Menu or Menu Group, just use the default ID (a 10-digit number).
Description	Enter a short name for the new application.
Image ID	Select an image from the list. Barista comes with a standard set of images. To install additional images, copy them to **barista\sys\images\im_ftype_*.***.
Automatically Expand Menu	
Company ID	
Product ID	
Help ID	

Add Submenu

To add an optional submenu, right-click on the newly added application menu and select "Add Submenu":

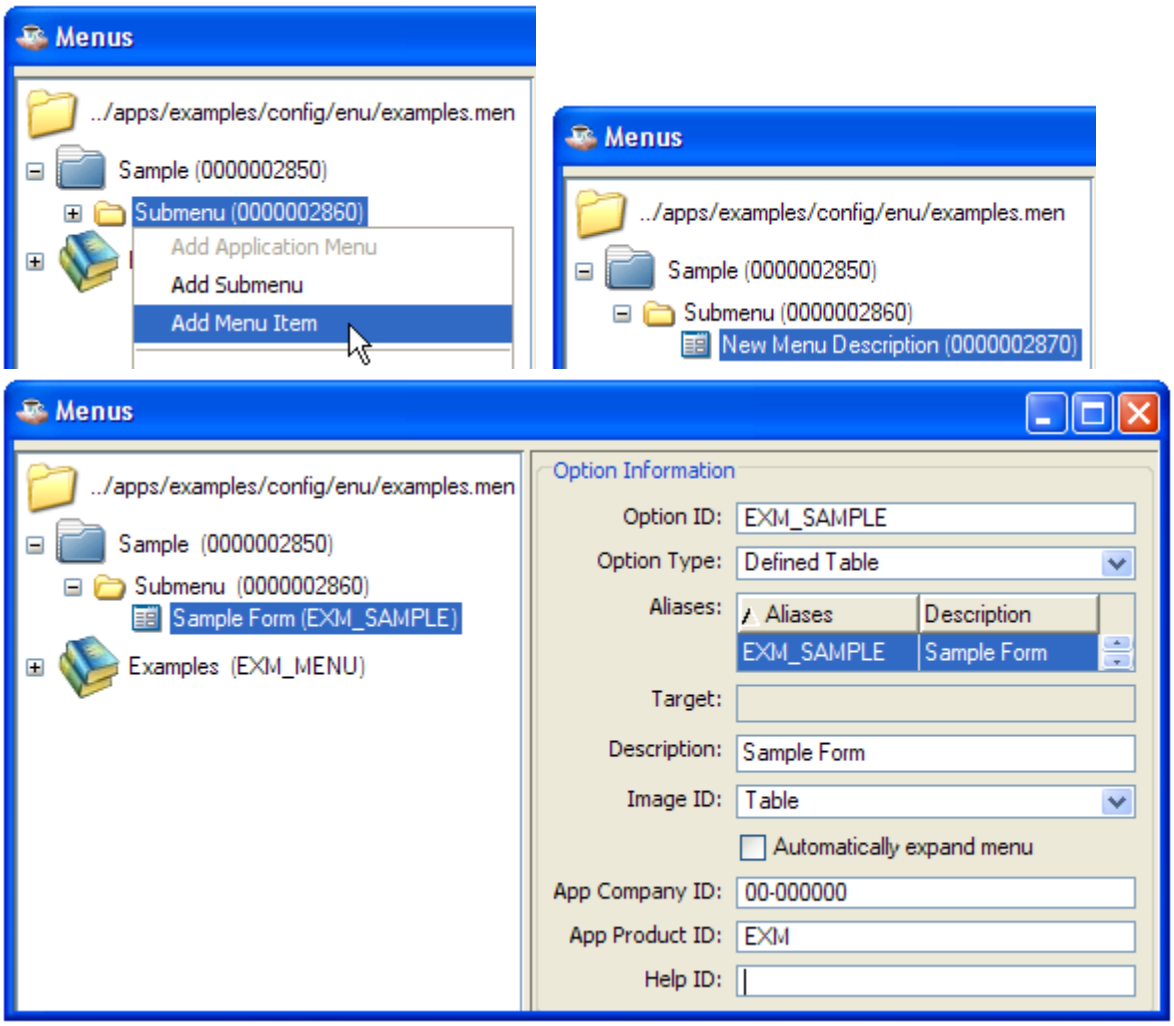


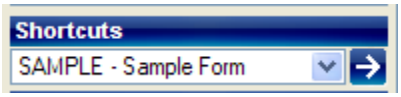
Option ID	When defining a new Menu Group, just use the default ID (a 10-digit number).

Description	Enter a short name for the submenu.
Automatically expand menu	Check this to expand this submenu automatically whenever the application menu is opened.
Company ID	
Product ID	

Add Menu Item

To add a menu item, right-click on the application menu or submenu and select "Add Menu Item":



Option ID	The Option ID is a unique reference, up to 20 characters, that identifies a menu system entry. This reference will appear in the Shortcuts and History lists, for example: 
Option Type	Select one of the following Option Types: • Defined Alias=Barista Form • BASIC Program=Standalone BBj Program • Application Program=Barista-Aware BBj Program • System Call=System-level command
Target	If the Option Type is a program, enter the program name here. If the Option Type is a system call, enter the system command here.
Description	Enter a short name for the menu item.
Image ID	Select an image from the list. Barista comes with a standard set of images; to install additional menu images, copy them to **barista\sys\images\im_mtype_*.***.
Company ID	
Product ID	
Help ID	

Other Options

To move a menu item up or down in the list, select it and either press [Alt]+□ / [Alt]+□, or right-click on it and select Move Up or Move Down from the popup menu.

To delete a menu item, select it and press [Ctrl]+[Delete], press the  tool button, or right-click on it and select Delete from the popup menu.

To update the menu system after making changes in the Menu Designer, either right-click or click on the  tool button and select Refresh Menu.

Reference

System Publics (SPUB)

These are called programs that perform commonly used functions within the Barista environment.

Code	Notes
call stbl("+DIR_SYP")+"bam_inquiry.bbj",gui_dev,Form!,"*<table_id>*","VIEW", table_chans$[all],"*<key_prefix>*","*<key_id>*","","","","","",""	Calls the Barista inquiry system for the specified table and key prefix
call stbl("+DIR_SYP")+"bam_run_prog.bbj","*<table_id>*",user_id$,"","", table_chans$[all],"",dflt_data$[all]	Calls the Barista runtime and launches the form for the specified table

<table>
<tr><th>Program Name</th><th>Description</th><th>Parameter List</th></tr>
<tr><td>bac_create_table.bbj</td><td>Create a work file based on an existing Barista table alias.</td><td>
<table>
<tr><th>Parameter</th><th>In</th><th>Out</th></tr>
<tr><td>rd_table_alias$</td><td>Table alias, C(16), uppercase.</td><td>Unchanged</td></tr>
<tr><td>rd_disk_file$</td><td>"" to use the file name from the alias record, or a fully qualified filename.</td><td>File name.</td></tr>
<tr><td>rd_table_chans$[]</td><td colspan="2">This array always exists in the Barista environment; pass it as given.</td></tr>
<tr><td>rd_create_action$</td><td>"ERASE" to erase existing file if it already exists, otherwise ""</td><td>Unchanged.</td></tr>
<tr><td>rd_create_status$</td><td>""</td><td>Returns "" if successful, otherwise returns a human-readable error message.</td></tr>
</table>
</td></tr>
<tr><td>bac_error.bbj</td><td>Report a program error to the user.</td><td>
<table>
<tr><th>Parameter</th><th>In</th><th>Out</th></tr>
<tr><td>rd_err_prog $</td><td>Program name</td><td>Unchanged</td></tr>
<tr><td>rd_err_line$</td><td>Error line number</td><td>Unchanged</td></tr>
<tr><td>rd_err_num $</td><td>BBx error number (ERR)</td><td>Unchanged</td></tr>
<tr><td>rd_err_text$</td><td>Error message, typically passed as ERRMES(-1)</td><td>Unchanged</td></tr>
<tr><td>rd_err_act$</td><td>N/A</td><td>Returns the user's chosen action:
"EXIT"
"RETRY"
"ESCAPE"</td></tr>
</table>
</td></tr>
<tr><td>bac_key_template.bbj</td><td>Get Template For Selected Key.</td><td>
<table>
<tr><th>Parameter</th><th>In</th><th>Out</th></tr>
<tr><td>rd_table_id$</td><td>Table alias, C(16), uppercase.</td><td>Unchanged</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>rd_table_key$</td><td>Key name, C(16), uppercase.</td><td>Unchanged</td></tr>
<tr><td>rd_key_template$</td><td>N/A</td><td>Template that describes the structure of this key.</td></tr>
<tr><td>rd_table_chans$[]</td><td colspan="2">This array always exists in the Barista environment; pass it as given.</td></tr>
<tr><td>rd_status$</td><td></td><td>Returns "" if successful, otherwise returns a human-readable error message.</td></tr>
</table>
</td></tr>
<tr><td>bac_message.bbj</td><td>Display or Query System Message.</td><td>
<table>
<tr><th>Parameter</th><th>In</th><th>Out</th></tr>
<tr><td>rd_msg_id$</td><td>Message ID defined in Administration, Maintenance, System Messages.</td><td>Unchanged.</td></tr>
<tr><td>rd_msg_tokens$[]</td><td>Single-dimension array of replacement tokens, to be inserted into the message based on place-holders %1, %2, etc.
rd_msg_tokens$[1] replaces %1, rd_msg_tokens$[2] replaces %2, and so on.</td><td>Unchanged.</td></tr>
<tr><td>rd_msg_opt$</td><td>Pass "" to display a dialog window for the user, or "<<TEXT>>" to retrieve the message text without displaying a dialog.</td><td>If "<<TEXT>>" was passed in, returns the formatted message, otherwise returns the user's selection.</td></tr>
<tr><td>rd_table_chans$[]</td><td colspan="2">This array always exists in the Barista environment; pass it as given.</td></tr>
</table>
Note: Assumes the current language, STBL("+LANGUAGE_ID"). These messages are maintained in Administration, Maintenance, System Messages.
</td></tr>
<tr><td>bas_sequences.bbj</td><td>Get Next Sequence Number. Assign defined sequence numbers during updates, reports, etc.</td><td>
<table>
<tr><th>Parameter</th><th>In</th><th>Out</th></tr>
<tr><td>rd_sequence_id$</td><td>Sequence Number ID defined in Administration, Maintenance, Sequence Numbers.</td><td>Unchanged.</td></tr>
<tr><td>rd_sequence_no$</td><td>N/A</td><td>Next sequence number.</td></tr>
<tr><td>rd_table_chans$[]</td><td colspan="2">This array always exists in the Barista environment; pass it as given.</td></tr>
</table>
Sequence numbers are maintained in Administration, Maintenance, Sequence Numbers.
</td></tr>
<tr><td>bac_open_tables.bbj</td><td>Open/Create Tables</td><td>
<table>
<tr><th>Parameter</th><th>In</th><th>Out</th></tr>
<tr><td>rd_open_beg</td><td>Index of the first element of</td><td>0</td></tr>
</table>
</td></tr>
</table>

			rd_open_tables$[] to be processed, or 0 to start at the beginning.	
		rd_open_end	Index of the last element of rd_open_tables$[] to be processed, or 0 to process through the end.	0
		rd_open_tables$[]	Array of table aliases.	If rd_open_opts$[i] includes "A", rd_open_tables$[i] returns the disk file name, otherwise it's unchanged.
		rd_open_opts$[]	● F=Find existing channel for file if open ● A=Get actual disk file name based on alias ● O=Open disk file ● N=Force open on new channel ● T=Retrieve template record ● L=Lock file ● C=Close file ● S=Skip error messages if file open unsuccessful ● I=Initialize file ● D=Define file if not found	
		rd_open_chans$[]	Pass an empty array.	Each element returns the channel number of the corresponding file.
		rd_open_tpls$[]	Pass an empty array.	Each element returns the record template for the corresponding file.
		rd_table_chans$[]	This array always exists in the Barista environment; pass it as given.	
		rd_open_batch	Unused	
		rd_open_status$	N/A.	Returns "" for success, or a human-readable error message.
bac_winsize.bbj	Barista stores persistent information about windows in the [Windows] section of .usr files in the barista/settings/enu/ directory (assuming the English – U.S. language code). The rdc_winsize utility saves and reads that information. Developers can use rdc_winsize to manage persistent information for their custom-built windows.			

Parameter	In	Out
rd_win_key$	Unique Window ID.	Unchanged.
rd_win_type$	"U" = User-defined custom window, not associated with a Barista form. "W" = Maintenance or Options Entry Form. "G" = Maintenance or Options Entry Grid. "I" = Inquiry Window. *The "W", "G", and "I" codes are reserved for internal Barista use only.*	Unchanged
rd_col_str$	Unused.	
rdWindow!	A BBjWindow object.	Unchanged
rd_action$	"SAVE" or "READ"	

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>rd_win_coord[]</td><td>N/A</td><td>Returned as a four-element array of X, Y, W, H.</td></tr>
</table>
</td></tr>
<tr><td>bam_config.bbj</td><td>Read or write configuration information in the barista .cfg, .ini, or .usr files.</td><td>
<table>
<tr><th>Parameter</th><th>In</th><th>Out</th></tr>
<tr><td>rd_cfg_file$</td><td>Fully-qualified filename.</td><td>Unchanged.</td></tr>
<tr><td>rd_cfg_type$</td><td>"CFG"
"INI"
"USR"</td><td>Unchanged.</td></tr>
<tr><td>rd_cfg_proc$</td><td>"READ"
"WRITE"</td><td>Unchanged.</td></tr>
<tr><td>rd_attr_data$[]</td><td colspan="2">RD_ATTR_DATA$[0] is a string of attribute names, each padded out to 20 bytes.
RD_ATTR_DATA$[1:n] contains the data corresponding to each of the attribute names listed in RD_ATTR_DATA$[0]</td></tr>
<tr><td>rd_convert_0a$</td><td colspan="2">By default, rdm_config converts all occurences of "$0A$" in rd_attr_data$[] to $0A$. To disable this behavior, pass rd_convert_0a$ as "NO".</td></tr>
</table>
</td></tr>
<tr><td>bam_enable.bbj</td><td>Enable/Disable MDI tool buttons, menu items and controls on forms.</td><td>
<table>
<tr><th>Parameter</th><th>In</th><th>Out</th></tr>
<tr><td>rd_gui_dev</td><td colspan="2">This variable always exists in the Barista environment; pass it as given.</td></tr>
<tr><td>rd_able_proc$</td><td colspan="2">● "CREATE"=Create rd_able_map$ of all controls listed in rd_able_ctls$ (See below)
● "ENABLE"=Enable all controls listed in rd_able_ctls$ (See below)
● "DISABLE" =Disable all controls listed in rd_able_ctls$ (See below)
● "CHECK"=Check menu controls listed in rd_able_ctls$ (See below)
● "UNCHECK" =Uncheck menu controls listed in rd_able_ctls$ (See below)
● "ACTIVE" =Set status to active for all controls listed in rd_able_ctls$ (See below)
● "UPDATE" =Update status of all controls based on current values in rd_able_map$ (See below)</td></tr>
<tr><td>rd_disp_mode$</td><td colspan="2">● "INIT"=Initializes any previous rd_able_map$ string and settings."
● "CLEAR"=Clear control contents when enabling or disabling.
● "FORCE"=Override current rd_able_map$ status and perform function specified in rd_able_proc$
● "MDI"=Specifies the program is running within the Barista Framework MDI. (Mandatory)
● "IDLE"=When used in conjunction with rd_able_proc$=
● "CREATE"=Sets all controls to idle (always disabled)</td></tr>
</table>
</td></tr>
</table>

		rd_able_ctls$	Variable length string consisting of: Control number (00000) Control type: B = Button M = Menu E = Entry control Semi-colon delimited. Used in conjunction with rd_able_proc$ commands to manipulate various commands.	Unchanged
		rd_able_map$	Variable length string containing the current enable/disable status of all controls on the user interface, including menu and toolbar items Consists of: Control number (00000) Control type: B = Button M = Menu E = Entry control Current control enable/disable status: ‘’ = Enabled D = Disabled I/X = Always disabled. Semi-colon delimited.	Current control enable/disable status is updated.
		rd_able_ctx$	Variable length string defining the context for each control on a form, and consisting of: Control number (00000) Control type: B = Button M = Menu E = Entry control “-“ (Dash) Window context of control (0000).	Unchanged

			Semi-colon delimited.	
		rd_able_ctx	Context of controls on a form if the form consists of one context	Unchanged
		rdSysGUI!	This variable always exists in the Barista environment; pass it as given.	
		rd_rec_data$[]	Internal Barista use only	Unchanged
		rd_ctl_xref_str$	Internal Barista use only	Unchanged
		rd_attr_def_tbl$[]	Table Attribute Definition Array. (See Barista Data Structures)	Unchanged
		rd_attr_def_col$[]	Column Attribute Definition Array. (See Barista Data Structures)	Unchanged
		rd_attr_tbl$[]	Table Attribute Array. (See Barista Data Structures)	Unchanged
		rd_attr_col$[]	Column Attribute Array. (See Barista Data Structures)	Unchanged
bam_enable_pop.bbj	Enable/Disable Popup Menu Items			
		Parameter	**In**	**Out**
		Form!	This standard Barista variable identifies the current form.	
		rd_enable_str$	List of popup menu items to enable.	Unchanged.
		rd_disable_str$	List of popup menu items to disable.	Unchanged.
bam_grid_init.bbj	Initialize and configure a custom-created BBjGrid control.			
		Parameter	**In**	**Out** / **Notes**
		rd_gui_dev	This variable always exists in the Barista environment; pass it as given.	
		rdGridTemp!	A custom-defined BBjGrid control.	Unchanged. / This grid control is created using custom BBj code.
		rd_flags$	One or more of the following: ● AUTO=Enable auto-sort. ● CALENDAR= ● CELL=Clicking in the grid selects a individual cell, as opposed to the complete row. ● CHECKS=Signifies the grid contains checkboxes in one or more columns. ● COLH=Add column headers at the top.	

			● DATES= Signifies the grid contains dates in one or more columns. ● DOCV=Internal Barista code used for Document Output System. ● DRAG=Enable drag-and-drop. ● EDIT=Editable. ● HIGHO=Selects outline highlighting, as opposed to color. ● ICONS=Internal Barista code used for Framework Menu System. ● LIGHT=Lightens the color of horizontal and vertical grid line per Barista standards. ● LINES=Show horizontal and vertical grid lines. ● MULTI=Enable multi-cell selection. ● NOSCROLL=Suppress scrollbars. ● NOSIZE=Disallow resizing. ● PROP=Formats grid to Barista property sheet standards. ● ROWH=Add row headers on the left edge. ● SIZEC=Auto-resize last column. ● SORT=Make the columns sortable. ● TIGHT=Reduce the row height. ● VBOTTOM=Set vertical alignment to bottom. ● VCENTER=Set vertical alignment to center. ● VTOP=Set vertical alignment to top.	
		rd_num_rows	Number of rows.	Unchanged
		rd_attr_def_col$[ ]	Column Attribute Definition Array. (See Barista Data Structures)	Unchanged
		rd_inq_disp_col$	Variable length string defining which column ids are initially displayed. Each segment contains <table_id.column_id> and is padded to 40 bytes.	Unchanged
		rd_attr_col$[]	Column Attribute Array. (See Barista Data Structures)	Unchanged
		rdListVect!	Unused.	
bam_inquiry.bbj	File Inquiry System			
		Parameter	**In**	**Out**
		rd_gui_dev	This variable always exists in the Barista environment; pass it as given.	
		Form!	This standard Barista variable identifies the current form.	Unchanged
		rd_alias_id$	Table alias to query.	Unchanged
		rd_inq_mode$	● "DRILL"=Places inquiry into drilldown mode (future)	

<table>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td>● "MULTI"=Allows selection of multiple rows
● "VIEW"=View only. Does not allow selections.
● "ALL"=Overrides hide/show status and displays all columns.</td><td></td></tr>
<tr><td>rd_table_chans$[]</td><td colspan="2">This array always exists in the Barista environment; pass it as given.</td></tr>
<tr><td>rd_key_pfx$</td><td>Key prefix.</td><td>Unchanged</td></tr>
<tr><td>rd_key_id$</td><td>Key name.</td><td>Unchanged</td></tr>
<tr><td>rd_selected_key$</td><td>Start value.</td><td>User-selected value.</td></tr>
<tr><td>rd_filter_defs$[]</td><td>Unused</td><td></td></tr>
</table>
</td></tr>
<tr><td>bam_prog_bar.bbj</td><td>Display standard Barista progress meter during long-running processes (updates or reports)</td><td>
<table>
<tr><th>Parameter</th><th>In</th><th>Out</th></tr>
<tr><td>rdSysGUI!</td><td colspan="2">This variable always exists in the Barista environment; pass it as given.</td></tr>
<tr><td>rdBaseWin!</td><td>Identifies the calling form.</td><td>Unchanged</td></tr>
<tr><td>rdBarWin!</td><td></td><td>Identifies the progress form.</td></tr>
<tr><td>rd_meter_title$</td><td>Progress Bar Title</td><td>Unchanged</td></tr>
<tr><td>rd_total_recs</td><td>Total number of records.</td><td>Unchanged</td></tr>
<tr><td>rd_proc_recs</td><td>Current number of records.</td><td>Unchanged</td></tr>
<tr><td>rd_curr_data$</td><td>The text to display if “TXT” is specified in rd_action$ (below)</td><td>Unchanged</td></tr>
<tr><td>rd_action$</td><td colspan="2">One or more of the following:
● "WIN"=Display progress bar window.
● "MTR"=Display progress bar on window.
● "TXT"=Display text specified in rd_curr_data$ (above)
● "LST"=Display list box on progress bar window.
● "LSR"=Display contents of list box in reverse order. (last item first.)
● "END"=End of job. Close progress bar.
● "OK"=Add end of job “OK” button.</td></tr>
</table>
</td></tr>
</table>

Barista Data Structures

When launching a Barista defined form, grid or inquiry, all information about a form (controls, entry and validation rules, etc.) is internally stored and referenced by a predefined set of "Attribute Codes". The two categories of attributes are listed below (see Table Attribute Codes & Column Attribute Codes).

During normal processing, these attributes are invisible to the end user. When in Callpoints, these attributes are available via the callpoint! object. Developers, however, may wish to access, and in some cases create selected attributes when calling selected Barista publics from within custom standalone programs.

The first set of arrays needed are the Attribute Definition Arrays. These arrays contain information about the purpose of the attributes and are required to access and manipulate the Table Attribute Codes & Column Attribute Codes. The Attribute Definition Arrays, one for the table attributes (_tbl$[]), and one for the column attributes (_col$[]), are created with a call to the Attribute Definition public:

call "bam_attr_init.bbj",attr_def_tbl$[all],attr_def_col$[all]

Both arrays have the same format:
[0,0] – Contains a “;” delimited string of all attribute codes (“ALID;CPGM;CPRO;… “)
[<attribute_no>,0] – Contains the attribute information definition string:

```
     +---------------------------------------------(01,04) - Attribute Code
     |    +----------------------------------------(06,20) - Attribute Description
     |    |                    +-------------------(27,01) - Data Type
     |    |                    |                             C - Character
     |    |                    |                             N - Numeric
     |    |                    | +-----------------(29) - Internal Barista Information
     |    |                    | |
     AAAA-BBBBBBBBBBBBBBBBBBB-C-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

It is important to note the Attribute Definition Arrays arrays are intended to be read-only and should not be modified at any time.

The second set of arrays needed are the Attribute Arrays. These arrays contain the attribute value information needed by Barista to process the desired objects. The Attribute Arrays, one for the table attributes (attr_tbl$[]), and one for the column attributes (attr_col$[]), are normally maintained by Barista, but can be created manually.

attr_tbl$[]
[0] – Contains a “;” delimited string of all table attribute codes (“ALID;CPGM;CPRO;… “)
[<attribute_no>] – Contains the corresponding table attribute value.

attr_col$[]
[0,0] – Contains a “;” delimited string of all column attribute codes (“DVAR;DTAL;CSEQ;VTYP;… “)
[0,1] – Contains a “;” delimited string of 40 byte padded segments, each containing <table.column>.
[<table.column>,<attribute_no>] – Contains the attribute value for the column.

The following example shows how to use column attributes to create a Barista look and feel grid on a custom window.

```
Rem Custom Grid

rem --- Open GUI device and retrieve the SysGUI object
gui_dev=unt
open(gui_dev)"X0"
SysGUI!=bbjAPI().getSysGui()

rem --- Create Window & Grid Control Shell
Window!=SysGui!.addWindow(20,20,350,400,"Custom Grid Sample")
SampleGrid!=Window!.addGrid(101,5,5,340,390)

rem --- Retrieve the Attribute Definition Arrays
call stbl("+DIR_SYP")+"bam_attr_init.bbj",attr_def_tbl$[all],attr_def_col$[all]

rem --- Define 2 Column Grid Using Attributes
dim attr_col$[2,len(attr_def_col$[0,0])/5];rem Barista Column Attribute Array
attr_col$[1,(pos("DVAR"=attr_def_col$[0,0])+4)/5]="COL_1";rem Sets first column variable name to COL_
1
attr_col$[1,(pos("LABS"=attr_def_col$[0,0])+4)/5]="Column One";rem Sets first column header text
attr_col$[1,(pos("CTLW"=attr_def_col$[0,0])+4)/5]="160";rem Sets first column width
attr_col$[2,(pos("DVAR"=attr_def_col$[0,0])+4)/5]="COL_2";rem Sets second column variable name to COL
_2
attr_col$[2,(pos("LABS"=attr_def_col$[0,0])+4)/5]="Column 2";rem Sets second column header text
attr_col$[2,(pos("CTLW"=attr_def_col$[0,0])+4)/5]="160";rem Sets second column width
attr_col$[0,1]=pad("GRID_SAMPLE.COL_1",40)+pad("GRID_SAMPLE.COL_2",40);rem String of defined columns
variables
attr_disp_col$=attr_col$[0,1]

rem --- Call Barista Grid Initialization Public
call stbl("+DIR_SYP")+"bam_grid_init.bbj",gui_dev,SampleGrid!,"COLH-MULTI-AUTO-LINES-LIGHT",20,attr_d
ef_col$[all],attr_disp_col$,attr_col$[all]

Window!.setVisible(1)

event_ctl:rem --- Event Control
Window!.setCallback(Window!.ON_CLOSE,"exit_prog")
process_events,err=*same

exit_prog:rem --- Exit Program
release
```

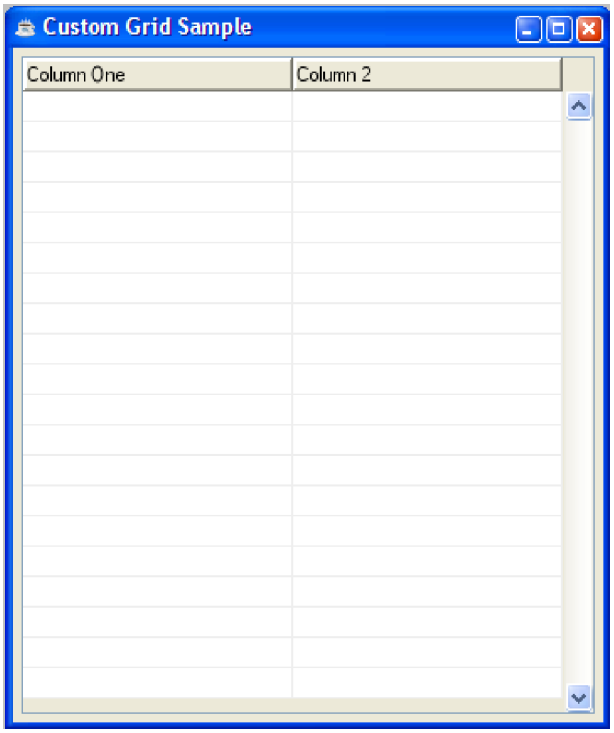


Table Attribute Codes

ID	Attribute Name	C/N	R/W	Notes
ALID	Table Alias	C(16)	R	Data dictionary table alias.
CPGM	Callpoint Program	C(60)	R	Callpoint program to run for selected processes.
CPRO	Callpoint Process	C(20)	R	Semicolon-delimited list of callpoint processes that trigger the callpoint program (e.g.: "AOPT;AREA;AREC;BDEQ;BSHO;"). ● BENA=Before Enable Map ● BDEL=Before Record Delete ● BDEQ=Before Delete Query ● BEND=Before Table Exit ● BGRS=Before Grid Display ● BGRD=Before Grid Entry ● BNEK=Before Next Record Key ● BNEX=Before Next Record ● BOVE=Before Table Overview ● BPRI=Before Previous Record ● BPRK=Before Prev Record Key ● BRCO=Before Record Copy ● BREA=Before Record Read ● BREC=Before New Record ● BREM=Before Record Removal ● BSAV=Before Detail Save ● BSHO=Before Window Show ● BWAR=Before Write Array ● BWRI=Before Record Write ● AENA=After Enable Map ● AFMC=After Form Creation ● ACAL=After Form Callbacks ● ACUS=After Custom Event ● AOPT=After Option Select ● APRA=After Print All Select ● APRT=After Print Rec Select ● ADEQ=After Delete Query ● ADEL=After Record Delete ● ADIS=After Record Display ● AGCL=After Grid Clear ● AGDS=After Grid Display ● AGRD=After Grid Exit ● AKEY=After Key Build ● AOVE=After Table Overview ● ARAR=After Array Transfer ● AREA=After Record Read ● ARNF=After Record Not Found ● AREC=After New Record ● ARCO=After Record Copy ● AREM=After Record Removal ● ASIZ=After Window Resize ● ASHO=After Window Show ● AWIN=After Window Create ● ASVA=After Save Validation ● AWRI=After Record Write
AOPT	Add Options	C(10)	R	Additional option definitions, specified as any number of 35-character semicolon-delimited segments in the following format: ● Option description (30) ● Option ID (4) ● Option location o M=Option menu o F=Form o B=Both
DESC	Description	C(30)	R	Alias description

WINT	Window Title	C(40)	R	Window title bar description
DTLG	Dtl Grid Table	C(16)	R	Detail table to place on the main form
DTLW	Dtl Window Tables	C(16)	R	Semicolon-delimited list of detail tables to access as separate windows. (e.g.: "DDM_ELEMENT_CMTS;DDM_ELEMENT_LDAT;")
DTLD	Del Cascade Tables	C(16)	R	Semicolon-delimited list of detail tables to include in the cascading delete.
DTLX	Del Depend Tables	C(16)	R	Semicolon-delimited list of detail table records checked to prevent deletion of current record.
DKNM	Data Key Name	C(16)	R	Key name to use for maintenance display
ATAB	Parent Alias	C(16)	R	Parent data definition table
PTAB	Primary Table	C(16)	R	Subset of which primary table
FTYP	Form Type	C(1)	R	Type of interface form to generate: ● X=No Form Used ● T=Maintenance Form ● G=Maintenance Grid ● S=Options Entry Form ● R=Options Entry Grid
FIMG	Background Image	C(64)	R	Background image to display on the form
HELP	Help Tag ID	C(10)	R	Help tag linking to system help
IDEF	Inquiry Options	C(16)	R	Defines multiple table inquiries and sorts
IKEY	Inquiry Restrict?	C(1)	R	Restricts inquiry sorts to columns with indexes defined in Table Key Definitions ● Y=Yes ● N=No
NOTE	Note Table	C(16)	R	Note table for records
OPTS	Optional Defs	C(1)	R	Semicolon-delimited list of table definition options. ● S=Auto save records on exit ● V=Redisplay record on save ● G=Update all grid records on save ● X=Destroy window on run ● Q=Launch inquiry on run ● F=Close after call program ● X=Destroy window on run ● Q=Launch inquiry on run ● F=Close after call program ● P=Allow Print record option ● A=Allow Print all records option ● O=Create 'Options' button on form ● W=Additional options always enabled ● U=Create function buttons on form ● 1=Do not scale background image ● B=Disallow new records ● I=Disallow inquiry changes ● D=Disallow all deletes ● L=Disallow dependent delete ● Z=Disallow window resize ● L=Use scrollbars on window resize ● N=Bypass new record prompt
PCAL	Call Program	C(60)	R	Exit call program
PRUN	Run Program	C(60)	R	Exit run program
TABG	Tab Definitions	C(60)	R	Semicolon-delimited list of tabs on the main form.

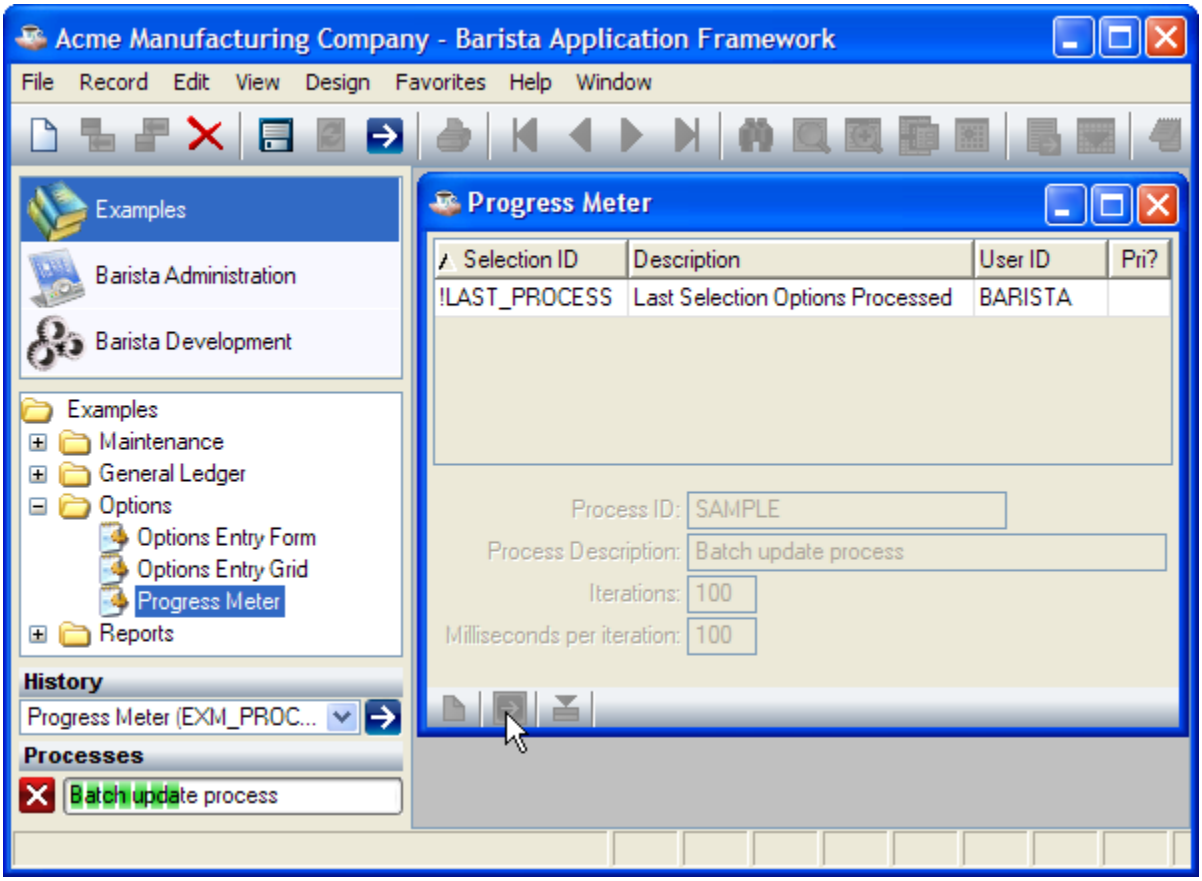
Column Attribute Codes

ID	Attribute Name	Format	R/W	Notes
DVAR	Data Name	C(16)	R	Database variable. Maintained by Barista.
DTAL	Source Alias	C(16)	R	Database alias. Maintained by Barista.
CSEQ	Column Seq	C(3)	R	Column sequence. Maintained by Barista.
VTYP	Variable Type	C(1)	R	Variable Type. Maintained by Barista. ● C = Defined Column ● A = Auxiliary Column ● L = Auxiliary Label ● D = Derived Data Element
CTLI	GUI Control ID	C(5)	R	GUI control id. Maintained by Barista.
CTLN	GUI Control Name	C(16)	R	GUI Control Name. Maintained by Barista.
CTLC	GUI Control Context	C(5)	R	GUI control context. Maintained by Barista.
DCHN	Data Table Channel	C(5)	R	Opened channel for DTAB. Maintained by Barista.
TIND	Tab Order Idx	C(3)	R	Tab order index of an entry object on its parent form.
RKEY	Primary Key?	C(1)	R	Included in primary key definition. ● Y=Yes ● N=No
CPGM	Callpoint Program	C(60)	R	Callpoint program to run for selected processes.
CPRO	Callpoint Process	C(20)	R	Semicolon-delimited string of process codes that trigger the callpoint program (e.g.: "BINP;AVAL;"). ● BINP=Before Column Input ● BINQ=Before Column Inquiry ● AINP=After Column Input ● AINQ=After Column Inquiry ● AVAL=After Column Validation ● AVEC=After Grid Vector Update
BTNS	Form Buttons	C(5)	R	Form function buttons. Maintained by Barista. May contain one or more of the following: ● F=Find ● D=Drilldown ● C=Calendar ● L=Document Link
ETYP	Element Type	C(16)	R	Element Type. Maintained by Barista
DESC	Description	C(30)	R	Column description
LABL	Window Label	C(30)	R	Control label text to display on form
LABS	Column Header	C(20)	R	Column header to display on reports and grid style forms
DTYP	Data Type	C(1)	R	Data type ● C=Character ● N=Numeric ● I=Integer ● U=Unsigned Integer ● B=Business Math ● O=BLOB
STYP	Data SubType	C(1)	R	Subtype definition for data types ● 1=Date (YYYYMMDD) ● 2=Date (YYYYMM) ● 3=Date (YYYY) ● 4=Date (YYMMDD) ● 5=Date (MMDD) ● 6=Created Date Stamp ● 7=Revised Date Stamp ● 8=Created Time Stamp ● 9=Revised Time Stamp ● A=Time (HHMMSS) ● B=Time (HHMM) ● C=Time (MMSS) ● S=Sequence Counter

				● I=Image Path
CTYP	Control Type	C(1)	R	Type of control to display on the form E=CharacterEdit (InputE) e=CharacterSpinner (InputESpinner) N=NumericEdit (InputN) n=NumericSpinner (InputNSpinner) D=DateEdit (InputD) d=DateSpinner (InputDSpinner) U=MultiLineEdit (CEDIT) C=CheckBox L=ListButton O=ListBox I = Spinner R = RadioButtons H=None (Hidden) X=None (Ref Only)
ABLC	Enable Column	C(40)	R	Enable/disable column. Specifies the <table.column_id> containing the data to use when calculating if the current column should be enabled or disabled. Value is specified in “ABLV”
ABLV	Enable Value	C(30)	R	Value trigger to enable/disable column. May contain one or more semi-colon delimited values to search when testing if current column should be disabled.
CDAT	Check Box Values	C(20)	R	CheckBox on/off value definition in the format "checked value;unchecked value" (e.g.: "Y;N").
CTLX	[Ctrl] Left (X)	N(4)	R	Entry control left edge fixed location (in pixels)
CTLY	[Ctrl] Top (Y)	N(4)	R	Entry control top edge fixed location (in rows)
CTLW	Fixed Width	N(4)	R	Entry control fixed width (in pixels)
CTLH	Fixed Height	N(4)	R	Entry control fixed height (in pixels)
DKEY	Data Key	C(40)	R	Special validation key for non standard record find
DTAB	Data Table	C(16)	R	Validation/display table
DCOL	Data Column	C(16)	R	Validation/display column
DKNM	Data Key Name	C(16)	R	Key name to use for validation
CALC	Data Calc	C(99)	R	Display data calculation. Calculation expression is defined in Form Manager □ Form Designer □ Derived Data Elements.
DCOM	Data Compress	C(12)	R	Data compression formula record ID.
DEXP	Data Expand	C(12)	R	Data expansion formula record ID.
DFLT	Default Value	C(80)	R/W	Default column value
DRLE	Drilldown Def	C(16)	R	Drill down definition ID
GHDR	Group Heading	C(30)	R	Header text for control entry group
HELP	Help Tag ID	C(10)	R	Help tag linking to system help
LDAT	List Data	C(10)	R	Semicolon-delimited list of values to display in a list control, specified in the format: ● List item description (30) ● “~” ● List item ID (4) e.g.: Product Level ~P ; Item Level ~I ; No Sales Analysis ~N ;
MINL	Min Length	N(3)	R/W	Minimum data input length
MAXL	Max Length	N(3)	R/W	Maximum data input length
MAXR	Max Rows	N(2)	R	Maximum number of rows for entry
MINV	Min Value	C(20)	R/W	Minimum data entry value
MAXV	Max Value	C(20)	R/W	Maximum data entry value
MSKI	Input Mask	C(20)	R	Mask used during data entry
MSKO	Output Mask	C(20)	R	Mask used during data display
MSKS	Mask System	C(3)	R	System ID for defined mask
MSKT	Mask Type	C(2)	R	Mask type to use from system record

OPTS	Optional Defs	C(1)	R	Semicolon-delimited list of column definition options taken from the following list (e.g.: "I;N;O;U;"). ● W=Hide data in maintenance form ● G=Hide data in maintenance grid ● I=Hide data in inquiry system ● O=Hide data in all forms ● P=Hide data in 'Print All' option ● R=Hide data in 'Print Record' option ● L=Hide corresponding control label ● +=Create total for inquiry column ● V=Display description in grid ● E=Save contents on conditional disable ● Y=Period/year entry control ● 1=From/to entry control set ● 2=Null entry='First' or 'Last' ● 3=Null entry='All' ● T=Limit to variable naming rules ● X=Resize control on window resize ● ?=Wildcard data entry ● S=Validate entries on save only ● N=Control contains external link ● D=Use last valid entry as default ● F=Do not resolve STBL reference ● U=Grid entry value must be unique ● C=Display only-all cases ● A=Display only-new records only ● B=Display only-existing records only ● Z=Value must be zero for record delete ● p=Password entry control ● s=Spellcheck control ● 0=Zero based sequence number control ● #=Assign next sequence on null entry ● M=Multiple language data control
PADC	Pad Character	C(2)	R	Pad character. ● 20=Space ● 22=Quote ● 23=Pound (#) ● 2A=Asterisk (*) ● 2E=Period (.) ● 30=Zero (0) ● 5E=Caret (^) ● 5F=Underscore (_)
PADJ	Pad Alignment	C(1)	R	Entry data alignment ● L=Left ● R=Right
PREC	Precision	N (1)	R	Numeric precision
PROM	User Prompt	C(80)	R/W	User help text to display on status bar
PVAL	Preset Value	C(30)	R	Preset (locked) value
SDLG	System Dialog	C(1)	R	System dialog allowed for inquiry ● O=File Open ● S=File Save ● P=Printer
TABG	Tab Location	C(1)	R	Tab location for this and subsequent controls Format: Numeric tab number.

Process Status




```
rem ' Update process status

if tcb(13) then enter

id$ = Option!.getOptionData("process_id")
task$ = Option!.getOptionData("description")
n = num(Option!.getOptionData("iterations"))
ms = num(Option!.getOptionData("milliseconds"))

Progress! = bbjapi().getGroupNamespace()
Progress!.setValue("+process_task",id$+"^C^"+task$+"^CNC^"+str(n)+"^")

for i=1 to n
  wait ms/1000
  Progress!.setValue("+process_task",id$+"^U^"+str(i)+"^")
next i

Progress!.setValue("+process_task",id$+"^D^")

release
```

Toolbar Reference

Tool Button	Menu Equivalent	Keyboard Equivalent	Notes
	Record New/Clear	[Ctrl]+N	
	Record Add New	[Ctrl]+A	
	Record Insert New	[Ctrl]+[Shift]+A	
	Record Delete	[Ctrl]+[Delete]	Delete current record.
	Record Save	[Ctrl]+S	Save current record.
	Record Refresh Data	[Alt]+[F5]	Reload record.
	Record Execute Process	[F5]	Run process.
	File Print Current Record	[Ctrl]+P	Print current record.
	Record First Record		First record.
	Record Previous Record	[Page Up]	Previous record.
	Record Next Record	[Page Down]	Next record.
	Record Last Record		Last record.
	View Record Query	[Ctrl]+Q	Record query.
	View Find	[Ctrl]+F	Find field records.
	Record Display Master Record	[Ctrl]+R	Display master record.
	Record Expand Grid Record	[Ctrl]+E	Expand grid record.
	View Calendar	[Ctrl]+L	Display calendar.
	View Launch Link		Launch link.
	View Drilldown	[Ctrl]+D	Drilldown.
	Record Record Notes		Record notes.
	Design Form Manager	[F8]	Form manager
	Design Form Designer		Form Designer
	Design Build Object	[Ctrl]+B	Build object.
	Design Edit Callpoints	[F2]	Edit callpoints.

	Design Edit External Program		Edit call/run program
	Design Increase Indent	[Alt]+ □	
	Design Decrease Indent	[Alt]+ □	
	Design Move To Top of Tab Order	[Ctrl]+[Home]	
	Design Move Up One in Tab Order	[Home]	
	Design Move Down One in Tab Order	[End]	
	Design Move To Bottom of Tab Order	[Ctrl]+[End]	
	Record Display Additional Options	[Ctrl]+O	

Status Bar Reference

Seg	Code	Description
0		Help message for the current field.
1	REQ	Displayed if the current field is required.
2	MOD	Displayed if the current record has been modified since the last save.
3	INQ	Displayed if the current field has an inquiry function.
4	EXP	Displayed if the current field has an expand function.
5	LNK	Displayed if the current field is an external link, such as an image file or a Microsoft Office document.
6	CAL	Displayed if the current field has a calendar option.
7	DRL	Displayed if the current field has a drilldown function.
8	HDR DTL	Indicates which panel (header or detail) is active. To toggle between header and detail, press F7 or select Edit□Toggle Panel from the menu.
9	EDT	When in a header/detail maintenance form, indicates that the detail grid is in edit mode.
10		The default printer, set in File□Processing Settings.
11		The current processing date. Defaults to the system date; can be changed in File□Processing Settings.
12		The product version number, from [+VERSION_ID].
13		When in the Form Designer, reports the number of tables defined in the data dictionary.
14		When in the Form Designer, reports the number of tables defined in the data dictionary that have been built.
15	COL ATT FRM	When in the Form Designer, indicates the active panel (Columns, Attributes, or Form Editor).
16		When in the Form Designer, indicates the calculated x coordinate (column) of the selected control. The column is calculated as the number of x pixels divided by the basis_rde.ini COL_WIDTH value, which defaults to 10 pixels.
17		When in the Form Designer, indicates the calculated y coordinate (row) of the selected control. The row is calculated as the number of y pixels divided by the basis_rde.ini ROW_HEIGHT value, which defaults to 21 pixels.
18		When in the Callpoint Editor, shows the current column and row (col:row)

Mask Reference

Numeric Masking

Char	Description
0	A zero is always replaced by a digit (0..9).
#	The pound sign is used to suppress leading zeroes. It is replaced by the fill character for leading zeroes to the left of the decimal point. For trailing zeros to the right of the decimal point it is replaced by a space or a zero. Any other time it is replaced by a digit. See SETOPTS byte 4, bit 04 for more information.
,	To the left of the decimal point, the comma is replaced by the fill character if no digits have yet been placed. Any other time, it results in a comma.
-	The minus sign creates a "-" in the result if the number is negative; otherwise, it is replaced by the fill character.
+	The plus sign becomes a "+" in the result if the number is positive, or a "-" if the number is negative.
$	The dollar sign always results in a dollar sign.
(	A left parenthesis results in a "(" if the number is negative, or the fill character if positive.
)	A right parenthesis results in a ")" if the number is negative, or the fill character if positive.
CR	The characters "CR" are inserted into the number if the number is negative. Two spaces are inserted if the number is positive.
DR	The characters "CR" are inserted into the number if the number is negative. The characters "DR" are inserted if the number is positive.
*****	The asterisk ("*") is inserted into the number.
.	The decimal point is replaced by a decimal point if any digits appear in the output mask. Otherwise, it is replaced by the fill character. After the decimal point, the fill character becomes a space.
B	The uppercase "B" always becomes a space. Any other character is simply copied to the result.
@	The @ sign is replaced by the three-character international currency code, based on the current value of STBL("!LOCALE").
&	The & sign is replaced by the local currency symbol, based on the current value of STBL("!LOCALE").

Some of the above characters can float within the mask. These are "-", "+", "$", "@", "&" and "(". If any of these characters is present in the mask, the first one encountered will be moved to the last position where a "#" or "," was replaced by the fill character. If no such position exists, the float character is left where it is.

String Masking

Char	Accepts
X	Any printable character.
a	Any alphabetic character.
A	Any alphabetic character. Converts lower-case alphabetic characters to uppercase.
0	Any digit.
U	Any digit, alphabetic, space, or punctuation character. Converts lower-case alphabetic characters to uppercase.
z	Any digit or alphabetic character.
Z	Any digit or alphabetic character. Converts lower-case alphabetic characters to uppercase.