Barista Creations – Add Your App, Stir, and Enjoy

One of the strongest incentives for developing within the Barista® Application Framework is the instant availability of new Barista features and functionality to your existing applications. Read on to learn how you can provide configurable, more sophisticated error handling in your Barista applications, and how to unleash the power of filtering in Barista queries.

Console Access

Console access has long been one of the most amazing benefits of programming in BASIS' interpretive BBx® languages. This is especially true when dealing with unexpected errors. This powerful feature can be particularly handy when dealing with an application runtime framework like Barista. With these new error-handling enhancements, Barista delivers the troubleshooting control that developers have grown to expect from their BBx-based applications.

By Christine Hawkins
Software Developer

Handling the Unexpected

Barista's messaging system already provides a robust mechanism for interacting with users when an application encounters an anticipated error or processing anomaly. Now, the process for handling *unanticipated* errors is enhanced so that Barista application developers can configure and control access to the console as well as error reporting options.

Three levels of error handling are available so choose the level that suits your needs. Make your choice based on the type of system, e.g., production, demonstration, or development, and the desired level of access that should be granted to users who may encounter an unanticipated error.

Strict:	No console access is allowed.
Authorized:	Console access is permitted, but only if a password is supplied.
Open:	Console access is granted.

Strict Error Handling

Strict error handling does not permit any real-time debugging. Users may be able to retry/resume processing if the error is due to a locked record or file, but otherwise they must abort the process or send an error report. This is the mode that is in effect

by default when launching applications via Web Start. **Figure 1** shows an example of what happens when an unanticipated error occurs in this mode.

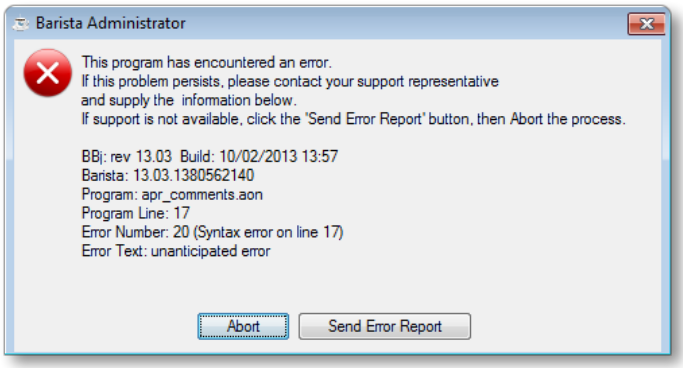

Figure 1. Unanticipated error message showing detailed information about the error

Authorized Error Handling

If you want to permit access to the console in a controlled fashion, configure Authorized error handling with two easy steps. First, remove the Disallow Console setting in BBj using Enterprise Manager (see **Figure 2**).

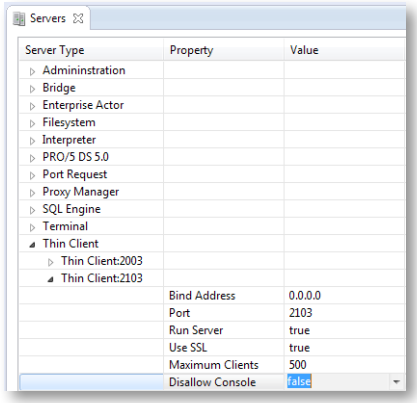

Figure 2. Turn off the Disallow Console setting in the Servers tab of the BBj Services node

Next, configure Barista to display a custom message, and accept a password before allowing access, as shown in these lines from barista.cfg:

```
set !CONEXIT=true
set !CONMESS=Please provide the password for console access, or <enter> to retry:
set !CONPASS=admin123
```

Authorized mode provides an additional [Debug] button on the error dialog so the console can be accessed once the user supplies the correct password.

Open Error Handling

If there are no concerns about console access, use Open mode. Like Authorized mode, Open mode removes the Disallow Console restriction in Enterprise Manager. However, with Open mode you do not enable the !CONxxx globals to prompt for an additional password, so clicking the [Debug] button immediately drops the user to a console prompt.

To help developers in the debugging process, BASIS unprotected the code in several of the Barista form-related runtime programs. This makes the debugging option available if an error occurs in one of these Barista programs when using Authorized or Open mode. Although the code is unprotected for debugging purposes, this is not an invitation to modify it; any modifications render the program unusable. Should a developer accidentally do so, the only recourse would be to restore the original program.

Figure 3 shows code from an AddonSoftware® program that uses the Barista error handler. In this sample, if the code isn't protected (as indicated by **`tcb(2)=0`**), then the text from the error line is fed into the handler. Once in the handler, if console access is allowed and the error line text has been supplied, the [Debug] button will be included on the error message dialog. Barista returns "ESCAPE" or "RETRY" if the user has pushed the [Debug] or [Retry] buttons in the message dialog, so the program can take action accordingly, or do an exit or release if the user has opted to Abort.

```
std_error:  rem --- Standard error handler

    rd_err_text$=""
    if tcb(2)=0 and tcb(5) then rd_err_text$=pgm(tcb(5),tcb(13),err=*next)
    call stbl("+DIR_SYP")+"bac_error.bbj",pgm(-2),str(tcb(5)),str(err),rd_err_text$,rd_err_act$
    if pos("ESCAPE"=rd_err_act$) seterr 0; setesc 0
    if pos("RETRY"=rd_err_act$) retry
    if pgm(-1)<>pgm(-2) status=999; exit
    release
```

Figure 3. Sample code for calling the Barista error handler

Regardless of the error handling mode, users can always generate an error report and add them to Barista's Document Processing Queue to email to the designated recipient. The error report form provides text input so users can describe the run-time conditions surrounding the error, and also a checkbox for including a workspace memory dump as part of

the error report. Developers or system administrators can configure Barista to allow or deny inclusion of workspace memory dumps, or include them if a password is supplied, as seen in **Figure 4.**

WHERE There's a Filter, There's a Way (to get the data you want)!

Inquiry

The inquiry system is perhaps Barista's most popular built-in component. Users can quickly and easily launch inquiries on forms and individual fields to find the data they're looking for, or click hyperlinks to display complete information for a coded field. In addition, applications can tie into the inquiry system with drilldowns and custom queries. The queries themselves are loaded with features for sorting, searching, filtering, and exporting in any of several formats (**Figure 5**).

Filter

If the basic sorting and searching capabilities aren't enough, users can take it further with the built-in filtering tool. The enhanced point and click interface in the filter tool makes it easy to construct more complicated filters across multiple columns, using AND/OR conjunctions, various operators, and parentheses for grouping. Power users may even be allowed to access the WHERE clause for direct editing. And remember, any number of filters can be defined and saved, so a user may quickly recall a filter to run the query again later, rather than having to reconstruct it every time, delivering mini-reportwriter functionality to the user.

Figure 6 shows the filter tool in action. Toggle the filter tool on or off by clicking the filter button at the top right of the query form. Create a filter by selecting the desired column, operator, and value, along with the desired conjunction. Click the [Enter] button to add each filter component of the resulting WHERE clause to the box at the right. When the clause contains all of the desired filter components, press the [Execute] button to run the query using the clause and see the results.

Group

In **Figure 6**, the query results aren't quite as expected. Some grouping is

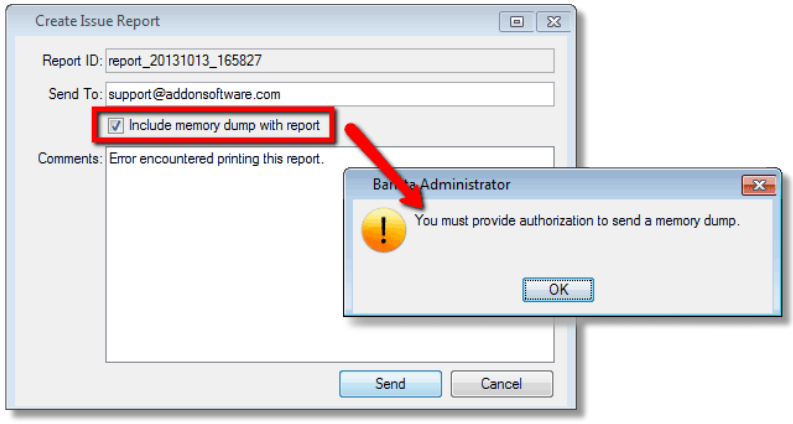

Figure 4. Create Issue Report and optionally allow inclusion of a workspace memory dump

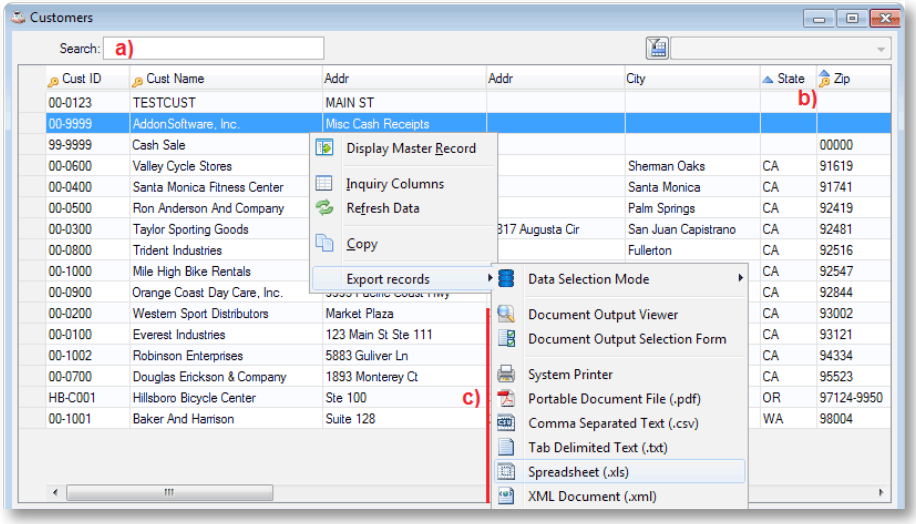

Figure 5. Inquiry grid showing a) Search box for quick filtering, b) multi-column sorting for State and Zip, and c) a variety of ways to output the query results

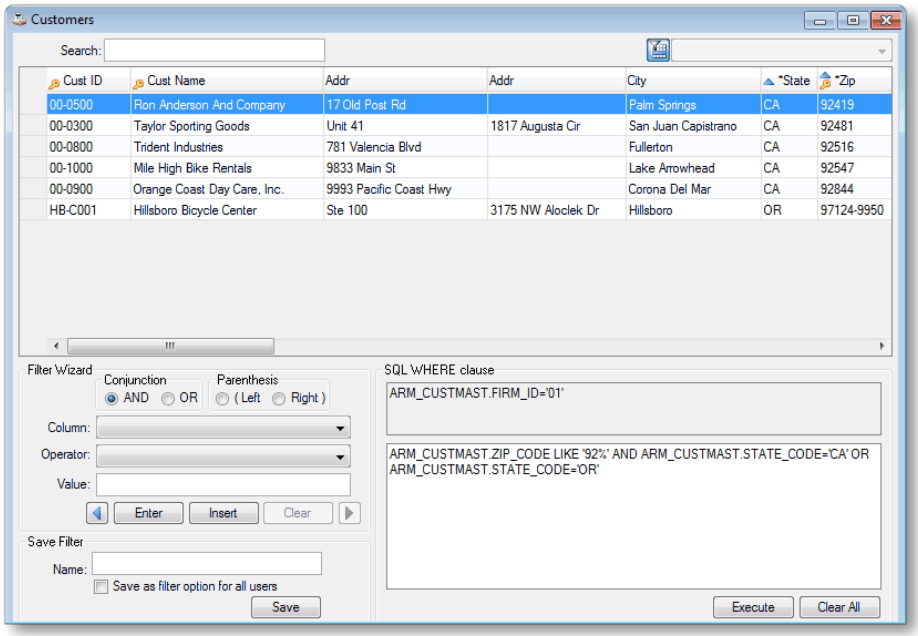

Figure 6. Filter for customer zip codes beginning with "92" and in the state of "CA" or "OR"

necessary to see to it that the value specified for the zip code applies across both states. Use the left and right arrow buttons (**Figure 7a**) to move back and forward through the WHERE clause one unit or filter component at a time. The wizard fields automatically populate with the column, operator, value, etc. of the filter component with the "focus." Then click the *Left* or *Right* radio buttons followed by the [Enter] button to add a parenthesis at the desired location (**Figure 7b**).

In addition to being able to specify the AND/OR conjunction and add parentheses, the point and click filter tool permits insertion or deletion of filter components. Use the arrow buttons in the Filter Wizard group box to "scroll" to the desired filter component, then press [Clear] to remove it from the clause, or [Insert] to create a new filter component in front of the one displayed.

All users can see the WHERE clause taking shape when working with the point and click interface, but Barista security options also make it possible for power users or administrators to edit the WHERE clause directly. An additional STATE_CODE filter component has been added directly to the clause in **Figure 8** by copying one of the other STATE_CODE filter components, then pasting and changing the state to "WA." Even with these permissions, the most efficient way to build a filter is usually a combination of point and click to do the main construction, and then direct edit for fine-tuning.

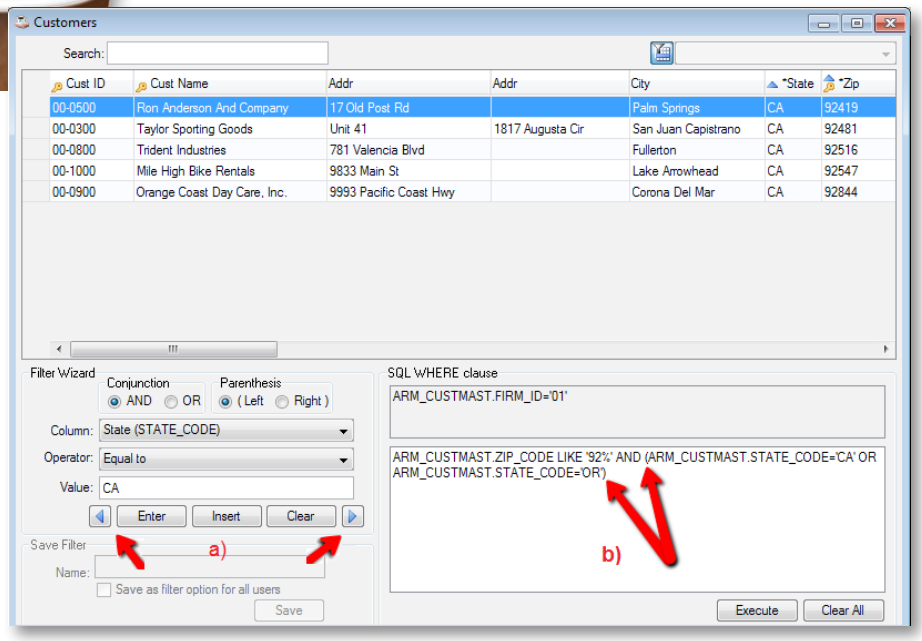

Figure 7. Refine the query results by a) using the arrow buttons to "scroll" through the WHERE clause and b) add parentheses for grouping

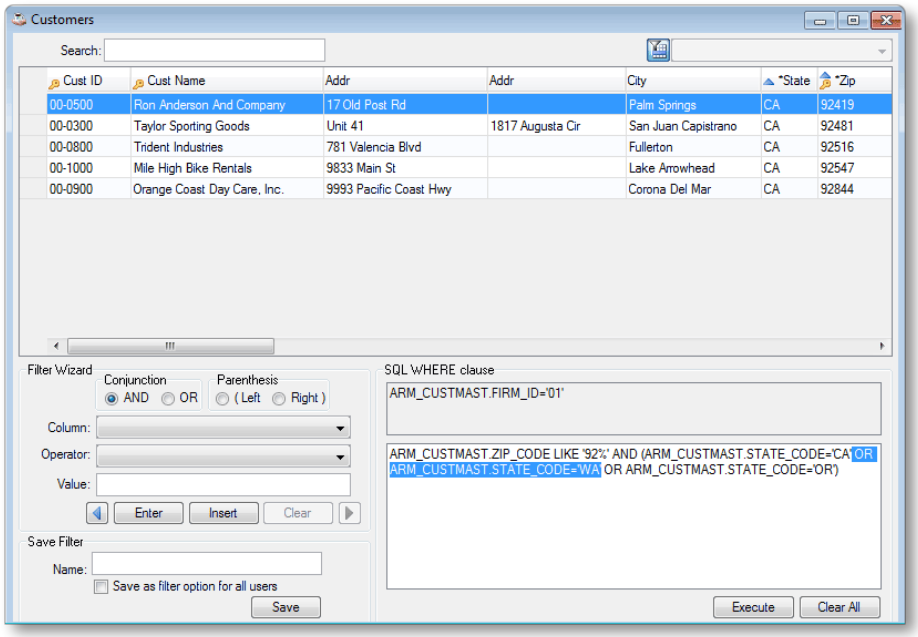

Figure 8. With direct edit permissions, users can edit the text in the WHERE clause control

Save a Filter

Remember, once users create and test a filter to verify that it produces the desired results, they can give the filter a name and save it for future use. If security settings allow the user to create global filters, the *Save as filter for all users* checkbox is enabled, and if checked, all users will see the new filter the next time they launch the inquiry. Each saved filter appears in the listbutton at the top right of the filter form (**Figure 9**). Once saved, apply a named filter by selecting it from the listbutton, or clear filters by selecting the first (blank) row.

To delete saved filters, select the filter and then press [Delete] in the MDI toolbar, or press <Ctrl+D>.

New! Search all Columns

Last but not least, is a new BBj 14.0 feature, previewed in 13.10. Barista inquiry grids now offer accelerated searching with a single click. Select the *Search all columns* checkbox to look for the specified *Search* text in any column, as shown in **Figure 10**. This can be a great time saver compared to constructing a WHERE clause when looking for the same text in multiple columns. As with normal searches, the full text search honors the configuration setting for case sensitivity.

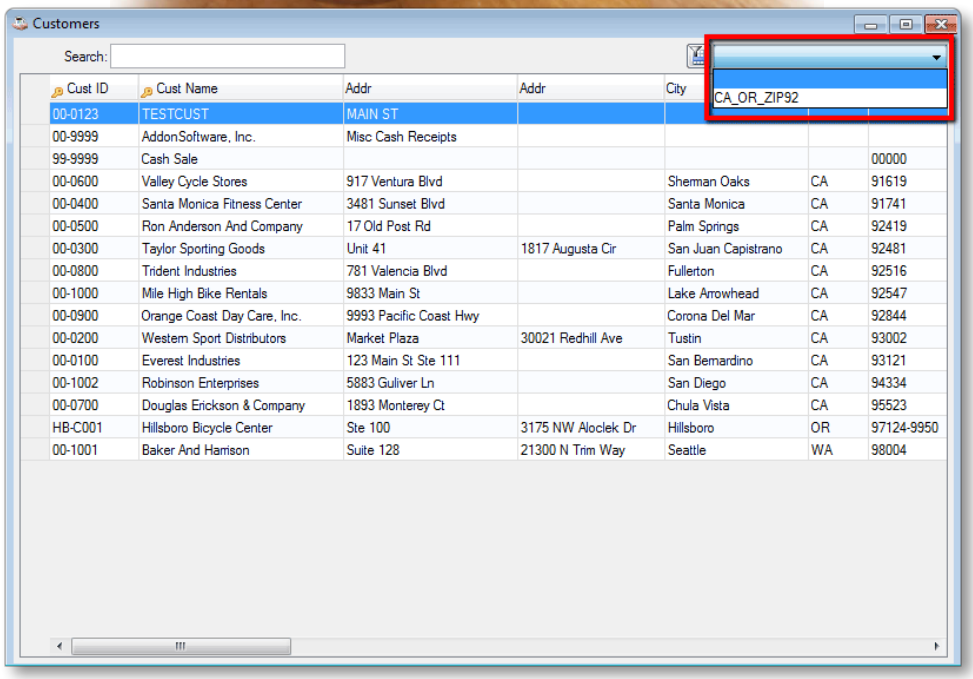

Figure 9. Example of a saved filter

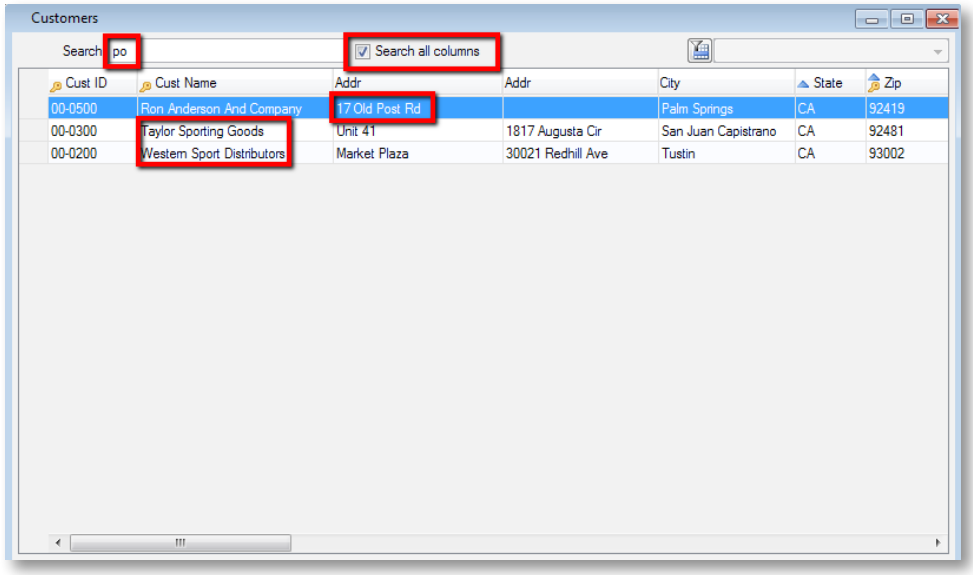

Figure 10. Case-insensitive search for "po" using new *Search all columns* option

Summary

As Barista continues to evolve and gain new functionality, so do your Barista-built applications. The recent addition of advanced and flexible error handling give developers the information they need to effectively support their application. And the addition of advanced query building and editing capabilities to the often-used inquiry system means that users will be even more effective at finding the exact data they need. Finally, the text search across all columns feature will boost your users' productivity when using your Barista-designed applications. These compelling new additions add value to your new or existing application without any development effort on your part, further solidifying Barista's position as a powerful application development and runtime framework. ■

links.basis.com/13code

For more information, refer to *Barista Error Handling* at links.basis.com/errorhandling