March 2012

On Technical/
Customizing EnterpriseOne

Business Function Error Handling in UBEs

by Brian Oster

Editor's Note: *What happens when a Business Function (BSFN) throws an error in a UBE? It's like that proverbial fallen tree in the woods; even if your users don't see it, it's still there. Wouldn't it be nice, though, if you could get a warning on those BSFN errors in a UBE, just like you do in an APPL, before further processing occurs? Brian Oster has a way to make that happen. In this article, he outlines the steps to take and offers plenty of codes to help you get started.*

Overview

Business Functions (BSFNs), both C and NERs, are very tightly coupled with Interactive Applications (APPLs) when it comes to error handling. When a function is called from an APPL, any errors or warnings thrown by that function are immediately displayed to the user in the APPL; and depending on which event the function was called from, may stop processing the action that initiated the event (see Figure 1). In addition to the error message, information about the error or warning is also displayed such as the Error ID and the function's source file and line number where the error was thrown (see Figure 2).

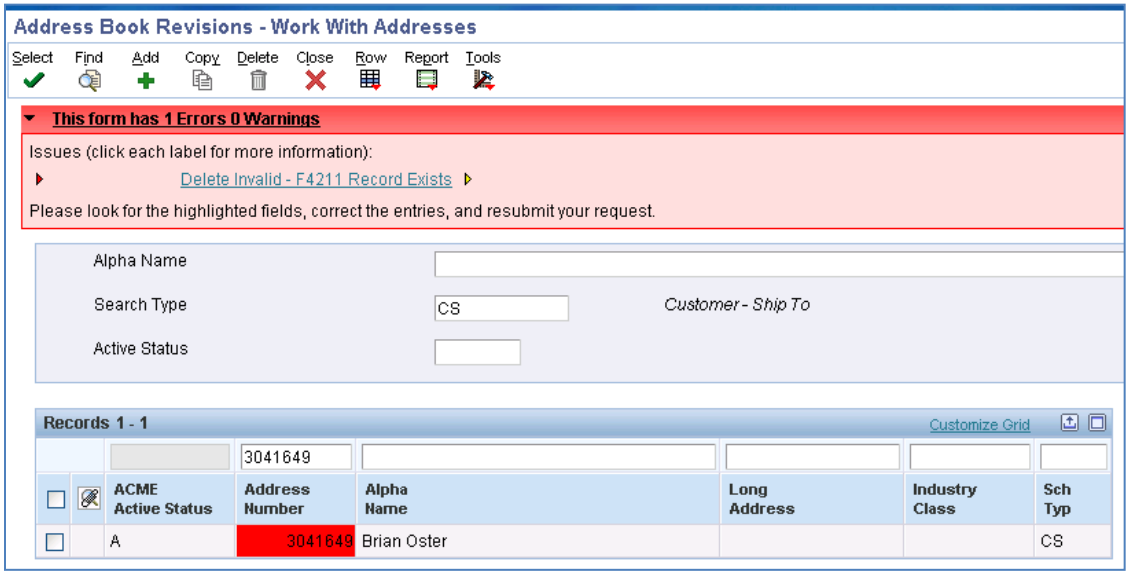

Figure 1: Error Displayed in Address Book When User Tries to Delete a Record that is In Use, and Stopping the Delete Action.

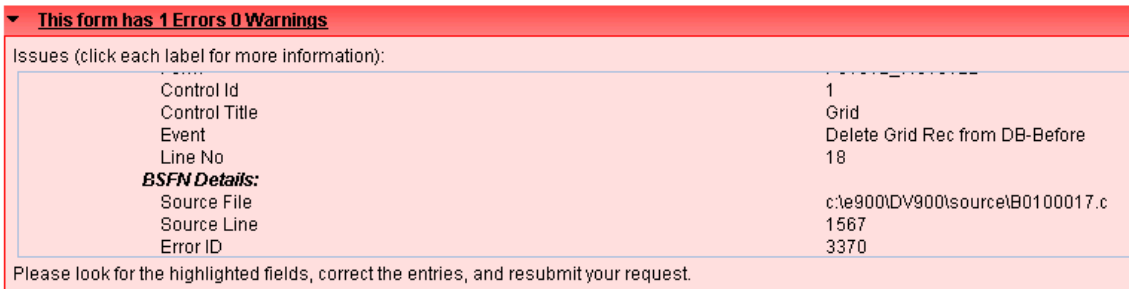

Figure 2: Expanded Error Information for the Error Displayed in Figure 1

Furthermore, when a function is called from an APPL, all this error handling behavior happens by default without any additional code or configuration by the developer. Unfortunately, this default behavior does NOT extend to UBEs. There is not any built in BSFN error handling mechanism in the UBE engine. If an error or warning is thrown from a function that is called from a UBE, nothing is displayed in the output of the UBE to the user and nothing prevents further processing in the UBE's event or the UBE itself. Obviously this can pose a pretty big problem if the developer is calling BSFNs from the UBE to do data validation, or worse, create transactions. Some BSFNs try to address this functionality gap by returning the error ID as a return parameter of the function so that the caller of the function can check to see if an error occurred. However, this is a very poor work around for several reasons:

- This only returns the error ID without any error message or additional error information.
- Only one error ID can be returned; if there were multiple errors there is no simple way to return the entire list of error IDs to the caller.
- The error may not be thrown directly in the called function at all, but instead by a function call made inside of the called function or even further down the call stack. More than likely this error ID will not be returned by the top level function call made from the UBE.
- Finally, and probably more importantly, most functions do not employ this work-around of returning the error ID as a parameter. In other words, you are completely at the mercy of the function you are calling as to whether or not you can check for errors and warnings when the function is called from your UBE.

In this article I will show you how you can implement a simple technique to create an error handling mechanism inside of UBEs that works much the same way the default error handling mechanism works inside of APPLs. Additionally this tip will work with calls to any BSFN, doesn't require any special code or return parameters by the called BSFN, and can catch any error, regardless of where in the call stack the error is thrown.

Example

To illustrate, I have set up a very simple example; we'll start by using this custom table I created with three fields (see Figure 3).

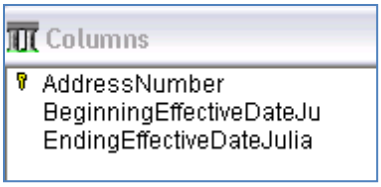

Figure 3: Example Table Structure

I then created a simple APPL with a Search and Select form over the table as well as a Headerless Detail form to add/edit the data in the table. In the "Row Exited & Changed Inline" event of the Headerless Detail form, I call two functions as per the following ER code: