Subject: Re: Update 4
Posted by AJM on Tue, 10 Jul 2007 17:54:19 GMT
View Forum Message <> Reply to Message

I have just tried modifying a MULTI4 transaction to reproduce the solution I have described, and it works exactly as I expected. I tested it using Radicore version 1.25.0 plus the updated version of 'include.general.inc' which I attached to an earlier post in this thread.

To make sure you are not missing something simple this is what I did:

- the outer entity of the MULTI4 function contains a field called 'sel_publish_key' which appears on the screen and is amendable by the user. This means that when a button is pressed the $_POST array will contain the current value for 'sel_publish_key'.

- when I select entries from the inner entity of the MULTI4 function and press a navigation button, the contents of the $_POST array is then merged with the contents of both the outer and inner entities before the button is processed.

- the inner entity of the MULTI4 function does not contain a field called 'sel_publish_key', so its value will not, by default, be added to $fieldarray within the inner entity. I can change this behaviour by adding the following code to the inner entity:

```
function _cm_changeConfig ($where, $fieldarray)
{
    $this->fieldspec['sel_publish_key'] = array('type' => 'string', 'nondb' => 'y');

    return $fieldarray;

} // _cm_changeConfig
```

- when the navigation button is pressed the following lines in 'std.multi4.inc' will be executed:

```
116:    $inner_post = getPostArray($_POST, $dbinner->getFieldSpec());
117:    $inner_data = array_update_indexed($inner_data, $inner_post);
```

You need to inspect this with your debugger to ensure that $_POST contains a value for 'sel_publish_key', and that $inner_data is updated so that every row now contains the value for 'sel_publish_key'.

- the next line in 'std.multi4.inc' calls the childform() function, which will pass control to the form associated with the navigation button which has been pressed. The processing should go through the code appearing at line 179:

```
    if (isset($post['select'])) {
        // convert selection into SQL where format
        $pkey_array = $dbobject->getPkeyArray(null, $task_array);
```

```
        $selection = selection2where($pkey_array, $post['select']);
    } else {
```

The call to getPeyArray() contains the call to _cm_getPkeyNames() which should contain the following:

```
function _cm_getPkeyNames ($pkey_array, $task_id, $pattern_id)
{
    $pkey_array[] = 'sel_publish_key';

    return $pkey_array;

} // _cm_getPkeyNames
```

This ensures that when the $selection string is built it will contain the value for 'sel_publish_key'. This string will then be passed to whatever function was activated by the navigation button.

All I had to do to get this to work was include 'sel_publish_key' in the _cm_changeConfig() and _cm_getPkeyNames() methods of the inner entity, so unless you are doing something really peculiar it should be just as easy for you.
