



Building 3-Level Entry Programs

Overview

The 3-Level Entry driver is actually much more than its name implies. It can be used to create and modify FACTS entry programs that consist of header, line entry and footer sections or any display or entry that involves a list of things.

Basic 3-level entries include Order Entry, Invoice Entry, Purchase Order Entry, Warehouse Transfer Entry, etc. Any program like these can be created using the 3-Level Entry Driver.

Examples of the other types of programs that can be created with the 3-Level Entry Driver include Deposit/Payment Entry, Past Sales Search, Price Search, Vendor Paid and Open Documents Drill Downs, the Committed Item Display in Order Entry, and the On Order Display from Order Entry.

The meta-data file used by the 3-Level Entry Driver is SMENTRY, and in this program you can indicate the functionality required by the program being created.

The 3-Level Driver allows you to create programs that have a header, line items and a footer, but the only one that is required when using the 3-Level Driver is line items. You can create a program that had no header and no footer.

The 3-Level Driver also allows you to select what line item functionality is available, including whether the user can Add/Insert/Delete/Edit or Select lines, and even specifying that lines may sometimes be edited or deleted. This allows you to create programs that may be read only or fully functional.

► Chapter Outline

- Overview
- Using the Main tab
- Using the Files tab
- Using the Screen 1 tab
- Using the Screen 2 tab
- Using the Main Line tab
- Using the Header Procedure tab
- Using the Header Detail tab
- Using the Line Procedures tab
- Using the Line Detail tab
- Using the Line Types tab
- Using the Lines Flag tab
- Using the Totals tab
- Using the Footer Procedures tab
- Using the Footer Detail tab

Menu bar

Header

Custom Header

Line item browser

Document Totals line

Line-item entry

Action buttons

LN#	Item	Flags	WH	Ordered	UM	Committed	Backorder	Price	UM
001	1100		01	1	EA	1	0	85.00	EA

The 3-Level Driver may also be called from anywhere in the system. It will clean up after itself, closing all channels that were opened, dropping all windows it created, and resetting the screen attributes of the screen that called it.

It also allows for a flexible call/enter list and a return value to tell the calling program what the user did while in the driver. You set the return value in SMENTY, and it will be placed in XS when exiting the driver.

Even though entry programs are often thought of in terms of their three primary sections — header, line entry and footer — a number of other sections comprise entry programs. You can control all of them through the SMENTY meta-data.

The Menu section can be defined using the routines from Main Tab of SMENTY.

The Main header can be defined in SMPRMT with the Main Header Entry program name as defined on Main Tab of SMENTY and the runtime condition of “main screen.”

Custom header – fields are selected from SO Entry Options F/M. The SMPRMT entries originate from the SMPRMT entries for the Header Detail program as defined on Main Tab of SMENTY and the runtime condition from SMENTY.

When the order entry program is run from the menu, a temporary SMPRMT file is created with a name of “data/SM/”+hta(fid(0)). Then all records for the Main header and the Custom header (and the Main Line Entry – see below) are copied to the temporary file. The Main header records are copied as is, but the

Custom header records have to be manipulated for screen position, tab order, description length, etc.

When running the entry driver, it will alternately be using the temporary SMPRMT or the normal SMPRMT based on context.

Custom header pop-up – fields are selected from SO Entry Options F/M. The SMPRMT entries are created from the SMPRMT entries for the Header Detail program as defined on Main Tab of SMENTY and the runtime condition from SMENTY.

These records also get copied to the temporary SMPRMT file with program being the Main Screen Header program from the Main Tab of SMENTY and the runtime condition “req header”.

Main line entry – fields are defined in SMPRMT with Main Line Entry program and runtime condition from SMENTY. These SMPRMT records also get copied to the temporary SMPRMT file with the program name of the Main Screen Header from the Main Tab of SMENTY and the runtime condition “main screen”. The tab order gets assigned sequentially starting with the last tab order from the Main Screen Header and Custom Header.

Required line fields – fields are selected from SO Entry Options F/M. The SMPRMT entries originate from the Line Detail program as defined on Main Tab of SMENTY and the runtime condition from SMENTY.

These records get copied to the temporary SMPRMT as well, with runtime conditions of “req lines 1”, “req lines 2” or “req lines 3”, and adjustments for screen locations, tab order, etc.

When running Graphical Line Entry, all of these fields are combined into “req lines 1”.

Line item browser – the columns are defined in SMENTY Main Line Tab. You can't edit the lines within this window.

Line flags may be displayed in this window. Available flags are defined in SMENTY Line Flags Tab and are selected by the user in SO Entry Options F/M.

Totals line – the totals that display at the bottom of the screen are defined in SMENTY Totals Tab and are selected by the user in SO Entry Options F/M. Up to three may be displayed. Total Extension is always displayed.

Buttons line – the Header, Line Detail, and Done buttons are pre-defined and always appear. Six standard and two custom buttons may be defined on the Main Scrn Tab in SMENTY. These routines are performed, and all the variables of the line in the line browser that is highlighted will be available.

➡ **To access the 3-Level Entry Meta-data, enter the access code SMENTY at any FACTS menu.**

Using the Main tab

File Maintenance for Entry Program Control

Program Name:

Runtime Cond:

Main | Files | Screen 1 | Screen 2 | Main Line | Hd Proc | Hd Det | Ln Proc | Ln Det | Ln Type | Flags | Totals | Ft Proc | Ft Det

Program Title: Return Eval:

Program Identifiers: Main Header Entry: Header Detail: Line Detail:
Main Line Entry: ☒ GUI Line Entry Footer Detail:

Standard Procedures

Initialization:

Menu Bar Definition:

Menu Bar Execution:

Frames Definition:

Custom Procedures

Menu Bar Update Procedures

New Doc - First Prompt:

New Doc - After First Prompt:

Main Screen:

Line Entry First Prompt:

Line Entry After First Prompt:

Write Delete Clear Exit

Program Title –

Return Eval –

Program Identifiers –

GUI Line Entry –

Initialization Procedures –

Menu Bar Definition Procedure –

The menu is defined, maintained and executed based on the above procedures.

Defining the menu is based on the ProvideX command syntax for menu_bar.

The following variables are used to define the menu:

_menu_bar_main\$ defines the main options for the menu.

_menu_bar_subs\$ defines the drop down menu options for the main options.

When the Menu Bar definition custom procedures are hit, to change the menu options available, simply change these variables. All changes will be available to both graphical and character users.

Menu Bar Execution Procedures -

The variable `_menu_sel$` gets set to the menu option selected based on the way the `menu_bar` read command in ProvideX works. The standard menu execution is run first, then the custom menu execution program.

To change the available options on the menu after it has been created, the menu bar update procedures are used. In these procedures, you can set the variable `_menu_action$`. Each entry begins with the selection identifier enclosed in brackets (e.g. `<FX>` for File, Exit) and ends with the same tag with a backslash (e.g. `<\FX>`). Between these two tags, may be any of the following: enable, disable, tick, untick.

Frames Definition -

Menu Bar Update Procedures:

New Doc - First Prompt -

New Doc - After First Prompt -

Main Screen -

Line Entry First Prompt -

Line Entry After First Prompt -

Using the Files tab

File Maintenance for Entry Program Control

Program Name: SOE510

Runtime Cond.: ORDER

Main | **Files** | Screen 1 | Screen 2 | Main Line | Hd Proc | Hd Det | Ln Proc | Ln Det | Ln Type | Flags | Totals | Ft Proc | Ft Det

Standard Header File: SORSOH Template Procedure: "prog/SO/SOSTRN;dim_sorsoh"

Custom Header File: Template Procedure:

Header Key Eval: %a0\$+shfile.doc_num\$

Standard Item File: SORSOL Template Procedure: "prog/SO/SOSTRN;dim_sorsol"

Companion Item File: Template Procedure:

Line Read Key Eval: %a0\$+shfile.doc_num\$+slfile.line Fixed Key: %a0\$+shfile.doc_num\$

Line Display Key Eval: %a0\$+shfile.doc_num\$+slfile.seq kno: 1 Fixed Key: %a0\$+shfile.doc_num\$

Line Data Filters: slfile.order_status\$<>"C"

Line Loading/Positioning

Lines to Load: 999 Z[4] Eval: 0

Line Piece Title: "" Pad Char: ""

Length: 0 Z[17] Eval: 0

Mask: ""

Sequence # Eval (for insert): slfile.seq_num\$

Sequence # Mask Eval: "000"

Std Preferences File: SMEPRF

Cust Preferences File:

Write Delete < > >> Clear Exit

Header and Line Files –

Line Read Key Eval –

Line Display Key Eval –

The difference between Line Read Key Eval and Line Display Key Eval is essentially the difference between using the line number and the sequence number.

Line Data Filters –

Line Data Filters are conditions that must be true for the line to be displayed. All line data filters must be true, and you can use and/or conditions within a single data filter to expand the number and complexity of the filters.

Lines to Load –

Line Piece Title –

Length –

Z[4] Eval, Pad Char, Z[17] Eval, Mask –

Sequence Number Eval (for insert) –

Sequence Number Mask Eval –

Preferences File -

The preferences files are read when entering the program and written to when returning to the menu. These files must have embedded data dictionaries, and the key to the standard file is Company + User Code.

Using the Screen 1 tab

File Maintenance for Entry Program Control

Program Name:

Runtime Cond.:

Main Files **Screen 1** Screen 2 Main Line Hd Proc Hd Det Ln Proc Ln Det Ln Type Flags Totals Ft Proc Ft Det

Standard Procedures Custom Procedures

Screen Creation:

Data Display:

Load List View:

Screen Layout

Screen Height: Screen Width:

Last Header Line Number:

Main Line Entry First Line: Last Line:

Main Line Entry Break Point Z[9]:

Line Browser First Line: Last Line:

Totals Line: Buttons Line:

Functionality

☒ Add Lines?

☒ Insert Lines?

☒ Auto Add?

☒ Edit Lines?

☒ Delete Lines?

☒ Load Lines?

☒ Auto Scroll?

Disable Cond

Inquiry Mode

☐ Search UI? Type Code:

Load Procedure:

Write Delete < > <> Clear Exit

Screen Creation Procedures - get executed immediately following the screen being created. After that, they are not accessed again.

Data Display Procedures - get executed each time the data on the main screen gets refreshed. Anything that needs to be done to the screen that isn't included in meta data (SMPRMT, SMENTRY, etc.) should be done in this procedure.

Load List View Procedure -

Screen Layout:

Screen Height -

Screen Width -

Last Header Line Number -

Main Line Entry First Line -

Main Line Entry Last Line -

Main Line Entry Break Point Z[9] -

Line Browser First Line –

Line Browser Last Line –

Totals Line –

Buttons Line –

Functionality:

Add Lines –

Insert Lines –

Auto Add –

Edit Lines –

Edit Lines Disable Condition –

Delete Lines –

Delete Lines Disable Condition –

Load Lines –

Auto Scroll –

Inquiry Mode:

Search UI –

Type Code –

Load Procedure –

Button titles must include a unique hot-key definition.

User-defined fields line number is the line number at which the Custom Header fields selected in SO Entry Options F/M will be displayed. For GUI, this must calculate to an even number. The available screen real estate in the Main Header can be increased by increasing this line number, allowing you to add more fields to the main screen. The line browser will be reduced to accommodate the changes.

Main Line Entry Break Point is the z[9] value for the field in the Main Line Entry before and after which the required fields 2 and 3 appear. 15 is the z[9] value for the price field.

Custom header and line field definitions allow you to add custom fields to SO Entry Options F/M. First enter the SMPRMT entries for Header Detail or Line Detail, then place the z[9] value and a title here. They will then appear in SO Entry Options F/M so the user can manipulate them.

Using the Screen 2 tab

File Maintenance for Entry Program Control

Program Name:

Runtime Cond.:

Main Files Screen 1 **Screen 2** Main Line Hd Proc Hd Det Ln Proc Ln Det Ln Type Flags Totals Ft Proc Ft Det

	Standard Titles	Procedures	Disable Cond	Exclude Cond
Button 1:	"De&posits"	"prog/SO/SOC504,p		
Button 2:				
Button 3:				
Button 4:				
Button 5:				
Button 6:				

	Custom Titles	Procedures	Disable Cond	Exclude Cond
Button 1:				
Button 2:				

Footer Button:

Exit Button Title:

Exit Procedure:

What does CR do?:

CR Procedure:

Custom Header Field Definitions

Z[9] Val	Title (not an eval)

Custom Line Field Definitions

Z[9] Val	Title (not an eval)

Write Delete < > Clear Exit

Button Titles –

Button Procedures –

Button Disable Conditions –

Button Exclude Conditions –

Footer Button –

Exit Button Title –

Exit Procedure –

What Does CR Do? –

CR Procedure –

Custom Header Field Z[9] Vals –

Custom Header Field Titles –

Custom Line Field Z[9] Vals –

Custom Line Field Titles –

Using the Main Line tab

File Maintenance for Entry Program Control

Program Name: SOE510

Runtime Cond.: ORDER

Main | Files | Screen 1 | Screen 2 | **Main Line** | Hd Proc | Hd Det | Ln Proc | Ln Det | Ln Type | Flags | Totals | Ft Proc | Ft Det

Entry Field Order Eval: "1,8,10,2,3,4,5,6,7"

-Standard-				-Custom-					
	Variable Eval	Wide	Just	Title		Variable Eval	Wide	Just	Title
Field 1:	{n%if\$(slfile.item_num	20	{n%if\$	"Item"	Field 11:		0	"L"	
Field 2:	{n%if\$(slfile.item_num	11	"R"	"Ordered"	Field 12:		0	"L"	
Field 3:	{n%if\$(slfile.item_num	2+	"L"	"UM"	Field 13:		0	"L"	
Field 4:	{n%if\$(slfile.item_num	11	"R"	"Committed"	-Standard-				
Field 5:	{n%if\$(slfile.item_num	11	"R"	"Backorder"	Field 14:			"L"	
Field 6:	{n%if\$(slfile.item_num	11	"R"	"Price"	Field 15:			"L"	
Field 7:	{n%if\$(slfile.item_num	2+	"L"	"UM"	Field 16:			"L"	
Field 8:	_line_flags\$	5	"L"	"Flags"	Field 17:			"L"	
Field 9:	{n%if\$(slfile.item_num	30	"L"	"Description"	Field 18:			"L"	
Field 10:	{n%if\$(slfile.order_han	4	"L"	"WH"	Field 19:			"L"	
					Field 20:			"L"	

Write | Delete | < | > | Clear | Exit

These field definitions are the available items that can appear in the line item browse window.

Entry Field Order Eval – The Entry field order is the order in which they are displayed within the browse window.

Variable Evals –

Wide –

Justification –

Titles –

Using the Header Procedures tab

File Maintenance for Entry Program Control

Program Name:

Runtime Cond.:

	Standard Procedure	Custom Procedures
Initialization:	<input type="text" value='"prog/SO/SOC502;set_sorsh_defaults"'/>	<input type="text"/>
Create Procedure:	<input type="text" value='"prog/SO/SOC502;create_doc"'/>	<input type="text"/>
Read Procedure:	<input type="text" value='"prog/SO/SOC502;prep_doc"'/>	<input type="text"/>
OK to Write:	<input type="text"/>	<input type="text"/>
Pre Write Procedure:	<input type="text" value='"prog/SO/SOC502;pre_write_header"'/>	<input type="text"/>
Post Write Procedure:	<input type="text" value='"prog/SO/SOC502;post_write_header"'/>	<input type="text"/>
OK to Delete:	<input type="text" value='"prog/SO/SOC502;ok_to_delete"'/>	<input type="text"/>
Delete Procedure:	<input type="text" value='"prog/SO/SOC502;document_delete"'/>	<input type="text"/>
Header Display Procedure:	<input type="text"/>	<input type="text"/>

These are the procedures used when writing or reading the document header.

Initialization sets the defaults for the new line. Create is used when creating the header for the first time, and includes the logic for assigning the new document number, etc.

The read procedure occurs when opening an existing document.

Ok to Write is checked before allowing the user to end out of the document.

Pre Write occurs immediately before the writes to the standard and companion data files. If you need to set or change any of the data in these files, you should do it here.

Post Write occurs immediately following the writes to the data files. If you need to execute some logic which requires the records to already be in the files, you should do it here.

Ok to Delete determines if anything on the document level prevents the document from being deleted. This is used in conjunction with the Ok to Write under the line procedures. To be able to delete the document, all lines must

clear the line level Ok to Delete, and the header level Ok to Delete must be cleared too.

The Delete procedure is performed last, after all lines have been deleted and the header record has been removed from the data file.

The Header Display procedure is performed when the user selects the header detail screen, after the window has been created.

Using the Header Detail tab

File Maintenance for Entry Program Control

Program Name:

Runtime Cond.:

Caption Eval:

Tab Definition:

Frames:

	Standard Button Titles	Procedures	Disable Cond
Button 1	<input type="text"/>	<input type="text"/>	<input type="text"/>
Button 2	<input type="text"/>	<input type="text"/>	<input type="text"/>

	Custom Button Titles	Procedures	Disable Cond
Button 1	<input type="text"/>	<input type="text"/>	<input type="text"/>
Button 2	<input type="text"/>	<input type="text"/>	<input type="text"/>

These fields all apply to the header detail window. See the documentation for the file maintenances for details on using Tabs and Frames.

Buttons can be added to the header detail screen, and the procedures defined will be performed and have access to all of the data.

Using the Line Procedures tab

File Maintenance for Entry Program Control

Program Name:

Runtime Cond:

	Standard Procedures	Custom Procedures
Initialization:	<input type="text" value="prog/SO/SOC503;initialize_line"/>	<input type="text"/>
Create Procedure (add line):	<input type="text" value="prog/SO/SOC503;create_line"/>	<input type="text"/>
Read Procedure:	<input type="text" value="prog/SO/SOC503;read_line"/>	<input type="text"/>
Pre Totals:	<input type="text" value="prog/SO/SOC503;set_org_totals"/>	<input type="text"/>
Line Pre-Edit:	<input type="text" value="prog/SO/SOC504;edit_line"/>	<input type="text"/>
Line Completion:	<input type="text"/>	<input type="text"/>
OK to Write:	<input type="text" value="prog/SO/SOC503;ok_to_write"/>	<input type="text"/>
Pre Write Procedure:	<input type="text" value="prog/SO/SOC503;pre_write_line"/>	<input type="text"/>
Post Write Procedure:	<input type="text" value="prog/SO/SOC503;post_write_line"/>	<input type="text"/>
Post Totals:	<input type="text" value="prog/SO/SOC503;set_totals"/>	<input type="text"/>
Entry/Post Display Proc:	<input type="text"/>	<input type="text"/>
OK to Delete:	<input type="text" value="prog/SO/SOC503;ok_to_delete"/>	<input type="text"/>
Delete Procedure:	<input type="text" value="prog/SO/SOC503;line_delete"/>	<input type="text"/>
Display Line Detail:	<input type="text"/>	<input type="text"/>

The initialization procedure is performed first, before reading a line and before creating a line.

The create procedure is used when creating a line for the first time, and includes the logic to assign the next line number, etc. This is used for both inserting a line and adding a line.

Pre-totals is executed immediately following reading or creating a line. In this routine, you should set variables to hold the values of the original line as they have impacted the totals. During the Post-totals routine, the new totals will be calculated as $TOTAL = TOTAL - ORG_VARS + CUR_VALS$.

Line Pre-edit is executed when editing a line. Since the read procedure gets executed when loading the browse window, there is code that should be executed to edit a line that is not needed when simply reading the line. This includes setting values needed while editing the line that aren't necessary when displaying it.

Line completion is used when editing a line on the main screen. It is performed after the final fields have been entered and provide you with one final location

to get user input. Setting `z[11]=4` or `OK=0` will cause the flow to return to the last field in the line entry.

Ok to Write, Pre-write, Post-write, Ok to Delete, Delete, and Display Line Detail all function exactly like those for the header.

There is a new procedure in SMENTY Line Procedures: Entry/Post Display Procedure

This procedure is executed immediately after adding or inserting a line and the new line is displayed in the line item browse region.

Its primary purpose is to allow automatically adding additional lines to the document, like memo lines that always go with the line or other items that always get sold with the item.

In the example, the affiliate would put "custom;add_memos" in the custom Entry/Post Display Procedure in SMENTY, and the following is the code they would create in the program `custom1`:

Program `CUSTOM1`:

```
0010 ADD_MEMOS:
0020 if not(nul(_ALT_LINE_ENTRY$)) then goto *return
0030 let _ALT_LINE_ENTRY$=quo+"custom1;create_memo"+quo
0040 if _EMODE=_ADD then perform "prog/SM/SME997;add_line"
    else perform "prog/SM/SME997;insert_line"
0050 return

0060 CREATE_MEMO:
0070 perform "prog/SM/SME997;setup_line"
0080 perform "prog/SM/SME997;create_line"
0090 perform "prog/SO/SOC504;memo_line"
0100 let SLFILE.DESCRPTION_1$="This is the new memo line"
0110 perform "prog/SM/SME997;write_line"
0120 return
```

`ADD_MEMOS` gets executed immediately following any line being inserted or added. That procedure sets the variable `_ALT_LINE_ENTRY$` to the name of a procedure that will setup all the variables for the new line item to be added, instead of having the user key in the data. It then performs either the standard `ADD_LINE` or `INSERT_LINE` procedures from `SME997`.

When the standard `ADD_LINE` or `INSERT_LINE` procedures are performed, the `ADD_MEMOS` procedure will be executed again, allowing a recursive call if the programmer chooses. To prevent an unintentional loop, we check to see

if the `_ALT_LINE_ENTRY$` variable is already set, and if so, we bypass the loop.

The `CREATE_MEMO` procedure is specified in `_ALT_LINE_ENTRY$`. The standard `ADD_LINE` and `INSERT_LINE` procedures in `SME997` are coded to look for this variable, and if it is set, they will perform it instead of providing the user interface as is normal for adding or inserting a line. When that happens, `CREATE_MEMO` performs the standard procedures for initializing a new line, creating a new line, and for setting up a new memo line, then it sets the description (the actual text of the new memo line). Finally, it performs the standard procedure for writing the new line.

Since `custom1` is using all of the standard procedures from `SME997`, the companion files will automatically be handled as well as all writes to the standard files, and all write procedures as specified in `SMENTY` will be executed as well.

Using the Line Detail tab

File Maintenance for Entry Program Control

Program Name:

Runtime Cond:

Caption Eval:

Tabs Definition:

Frames:

	Standard Titles	Procedures	Disable Cond	Custom Titles	Procedures	Disable Cond
Button 1	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Button 2	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

The top portion of this screen functions exactly like the corresponding fields for Header Detail.

Using the Line Types tab

File Maintenance for Entry Program Control

Program Name: SOE510

Runtime Cond: ORDER

Main Files Screen 1 Screen 2 Main Line Hd Proc Hd Det Ln Proc Ln Det **Ln Type** Flags Totals Ft Proc Ft Det

Line Item Conditions for Alternate Required Fields

Std Cond 1: slfile.item_num\$=i\$ Required Field Z[9] Values (comma separated): 1,2,3,4,5,6,14,18

Cust Cond 1: Required Field Z[9] Values (comma separated):

Alternate Line Item Types

	Condition	Title	Program Identifier	Line Entry Procedure
Type 1:	slfile.item_num\$=i\$	"Memo Line Entry"	SOC511	
Type 2:				
Type 3:				
Type 4:				
Type 5:				
Type 6:				
Type 7:				
Type 8:				
Cust Type:				

Write Delete Clear Exit

Line Item Condition for Alternate Required Fields is used to handle special situations like Temporary Items. When a temporary item is edited, the fields in the line detail with the z[9] values specified are also required. If any of them were already listed are required, they will not be duplicated. These alternate required fields will appear immediately following the Item Number prompt.

Alternate Line Item Types are used for different kind of line items, where the standard entry procedures do not apply. Memo lines are the obvious example. When the condition is true, the normal procedures for line detail and main line entry are bypassed, and the Answer/Entry driver is employed, using the program specified and the runtime condition from SMENTY. The title is used as the caption for the window that gets created.

Using the Line Flags tab

File Maintenance for Entry Program Control

Program Name:

Runtime Cond.:

Main Files Screen 1 Screen 2 Main Line Hd Proc Hd Det Ln Proc Ln Det Ln Type **Flags** Totals Ft Proc Ft Det

	Title	Char	Condition
Flag 1	Backorders	B	num(slfile.amount_backordered\$)>0
Flag 2	Direct Ship	D	slfile.order_handling_code\$="D"
Flag 3	Temporary	T	slfile.temporary_item\$="Y"
Flag 4	Alt Whse	A	slfile.ship_whse\$<>slfile.init_whse\$
Flag 5	Nonstock Item	N	slfile.nonstock\$="Y"
Flag 6	On Hold	*	slfile.ship_status\$="H"
Flag 7	Rejected	*	slfile.ship_status\$="R"
Flag 8			
Flag 9			
Flag 10			
Custom Flag 1			
Custom Flag 2			

Write Delete < > <> Clear Exit

Line Flags are made up of a condition that is either true or false concerning a line item, a title or description of what that condition means, and a display character that will show up in the line item browse window.

The titles that are entered here will be displayed in SO Entry Options F/M, where the user can indicate which flags they want to show up while running the entry program.

Using the Totals tab

File Maintenance for Entry Program Control

Program Name:

Runtime Cond.:

Standard Totals

Standard Line Condition for Inclusion in All Totals:

Title (not an eval)	Total Eval	Security Code
Weight	cvs(str[s[t_weight]]:"#####)	
Units	cvs(str[s[t_units]],3)	
Cost	cvs(str[s[t_cost]]:"#####)	ics_s1\$(7,1)
Mdse	cvs(str[s[t_merch]]:"#####)	
Taxable	cvs(str[s[t_tax]]:"#####)	
Misc	cvs(str[s[t_misc]]:"#####)	
Tax	cvs(str[s[t_tax]]*num[sfile.co	

Custom Totals

Custom Line Condition for Inclusion in All Totals:

Title (not an eval)	Total Var. Eval	Security Code

Because of the use of line data filters, it is possible that there are lines which do not appear in the browse window but which should be included in the totals, so there are two conditions that may be set which must be true for the line to be included in the totals. These conditions control whether the Pre- and Post-totals procedures are executed for a particular line.

Totals may be assigned a security code, and if the user doesn't have that security code at runtime, the total will not be displayed. A maximum of three totals may be selected at any one time.

The variables you select to maintain totals in should be initialized to zero in the header initialization procedure then accumulated in the Post-Totals procedure. Don't forget to have a separate set of variables that get set to the original values in the Pre-Totals procedure and also get initialized in the header initialization procedure. *Example:*

Initialization procedure sets org_val=0, val=0

Pre-Totals procedure sets org_val=ship qty * price

Post Totals procedure sets $val = val - org_val + (ship\ qty * price)$

Using the Footer Procedures tab

File Maintenance for Entry Program Control

Program Name: SOE510

Runtime Cond: ORDER

Main Files Screen 1 Screen 2 Main Lne Hd Proc Hd Det Ln Proc Ln Det Ln Type Flags Totals **Ft Proc** Ft Det

	Standard	Custom
Access OK Procedure:	"prog/SO/SOC504;ok_to_end"	
Read Procedure:		
OK to Write:	"prog/SO/SOC504;ok_to_save"	
Pre Write Procedure:	"prog/SO/SOC504;pre_write"	
Post Write Procedure:	"prog/SO/SOC504;post_write"	
Footer Display Procedure:		

Write Delete [Navigation Arrows] Clear Exit

The footer procedures are accessed similarly to the header procedures. Since the footer data is stored in the same data file as the header data, there is not a specific requirement for two sets of procedures, but it is useful to keep them conceptually separate.

The Access Ok Procedure is performed when the user selects “Done” from the main screen. It is currently used just to ensure that lines have been entered.

Using the Footer Detail tab

File Maintenance for Entry Program Control

Program Name: SOE510

Runtime Cond.: ORDER

Main Files Screen 1 Screen 2 Main Lne Hd Proc Hd Det Ln Proc Ln Det Ln Type Flags Totals Ft Proc **Ft Det**

Caption Eval: "Footer for Document "+cvs(shfile.doc_num\$,3)+" "+cvs(arcust_b2\$,3)

Standard Tabs Custom Tabs

Tabs Definition: Frames:

	Standard Titles	Procedures	Disable Cond
Button 1	"De&posits"	"prog/SO/SOC504.pay	
Button 2			

	Custom Titles	Procedures	Disable Cond
Button 1			
Button 2			

Write Delete navigation arrows Clear Exit

The fields here function the same way the corresponding fields in Header Detail and Line Detail function.

Since the standard product does not have tabs defined in the footer, it would be more difficult to add a separate tab in the footer. It is recommended that you not do so.