User Defined Functions

User Defined Functions (UDF) allow you to add fields to Lookup Grids using SU-A for calculated values or for data fields in files other than the primary file opened by the grid. A simple UDF contains up to 5 sections. Define variables, Open file, Find correct record, perform calculation, return results.

Template UDF

Create a text file using Notepad named UDF1.SRC (or use whatever number is the next available if you already have UDF#.SRC files for other functions) that contains the following. The number in the file name UDF#.SRC must match the number in the line func UDF# and the number entered on the SU-A screen as described below. Numbers 1 through 40 are available. Lines in the UDF beginning with // are comment lines. Any blue text is something that you would replace depending on the specifics of the UDF you are creating.

```
//This is a Template UDF containing sample lines for the various sections of a UDF
//This first line defined the UDF by a number.  Every UDF must have a unique number
        func UDF1
//This next line is used if you will be performing any sort of calculation and need a field defined to store
//the results. The example given is a 9 place number with 2 decimal.  The size includes the decimals plus
//the decimal point itself.  You can define multiple variables.
        Define variablename type N size 9 dec 2
//This next line is only needed if you need to open a file that the grid has not already opened.  You are
//defining a File Handle, or Alias for the program to use to open the file.  Replace filehandle.h with the
//name you want to use for your file
        define filehandle.h type I size 5
//Check to see if the file is already open so you only open it once, then open it
        if filehandle.h=0
        open filename fnum filehandle.h lock N
        endif
//Now find the correct record in the file.  Index tells the program which index to use on the new file and
//field is the field in the primary file that links to the index in the new file.
        Findv M fnum filehandle.h key index val field
//Now perform any necessary calculation
        variablename = calculation
//Finally, return the results which will typically be the Variable calculated or a field in the other file that
//was opened
        ret variablename
//or
        ret fieldfromfile
```

Adding a UDF to a Grid

When you are on a lookup grid screen, the name of the grid is on the lower left corner.
In Queries - SU-A for the appropriate grid, add a column for the UDF with the Data field named UDF#() where the # is the UDF number. The program will automatically populate the UDF column with U. Enter type A for an Alpha field or N for Numeric and the size you want the column to be.

Sample UDFs

```
// UDF1 will get the Cust PO for Shipments grid
func UDF1
define invhead.h type I size 5
// File handle to open BKARHINV
 if invhead.h = 0
   openv 'bkarhinv' fnum invhead.h lock N
// this will open the file once
 endif
 findv M fnum invhead.h key bkar.inv.num val bkar.invl.invnm
// this finds the correct record
// return the value
ret bkar.inv.cusord
```

****

The UDF below will calculate the Net Discounted price of a shipment if there is a discount on the line.

```
// UDF2 calculates discounted unit price
 func UDF2
 define discprice type n size 13 dec 4
 discprice=bkar.invl.pprce * (1-(bkar.invl.pdisc/100))
 ret discprice
```

****

This UDF is extremely simple. It is getting a field from a different file that is already open and on the correct record

```
// UDF3 will get the SO Number from the Sales Order header
func UDF3
ret BKAR.INV.SONUM
```

****

This UDF assumes a previous UDF was already called by another column on the grid and has already defined the file handle

```
 // UDF4 assumes UDF1 was called and running uses same file handle.  It returns the Invoice total
func UDF4
findv M fnum invhead.h key bkar.inv.invnum val bkar.invl.invnm
ret bkar.inv.total
```

****
This UDF needs to convert a text field to a numeric value to find the associated record in the other file

```
// UDF5 will get the Invoice number for the PO Receiver grid
func UDF5
 define apinv.h type I size 5
 if apinv.h = 0
   openv 'MKICLASS' fnum apinv.h lock N
 endif
 findv M fnum apinv.h key mkeclass.num val val(bkap.pol.invnum)
ret mkeclass.desc
```