

1 The ASK Utility

Writing a GUI (Graphical User Interface) for a GAMS application requires some serious programming, and therefore place a burden on the knowledge and time of the modeler. In this section we show how extremely simple user interfaces can be built using a few simple tools. The main purpose of these tools is to allow a developer quickly put an application together such that an end-user does not have to edit GAMS files. We assume the end-user runs a GAMS model from the GAMS-IDE.

The ASK utility can be used to get input from an end-user and the GDXViewer tool can be used to present end-results. Together these tools allow you to build a minimalist GUI without any programming.

The ASK utility

The ASK utility is a simple tool to ask simple interactive questions to the end-user. For instance, if your model requires a scalar to be changed regularly, instead of letting the end-user change the .gms source file, it may be better to pop up a window, with a question text, where the required number can be entered. The ASK tool allows you to do this. As the ASK tool generates a standard GAMS include file, this file can then be used through a \$include statement:

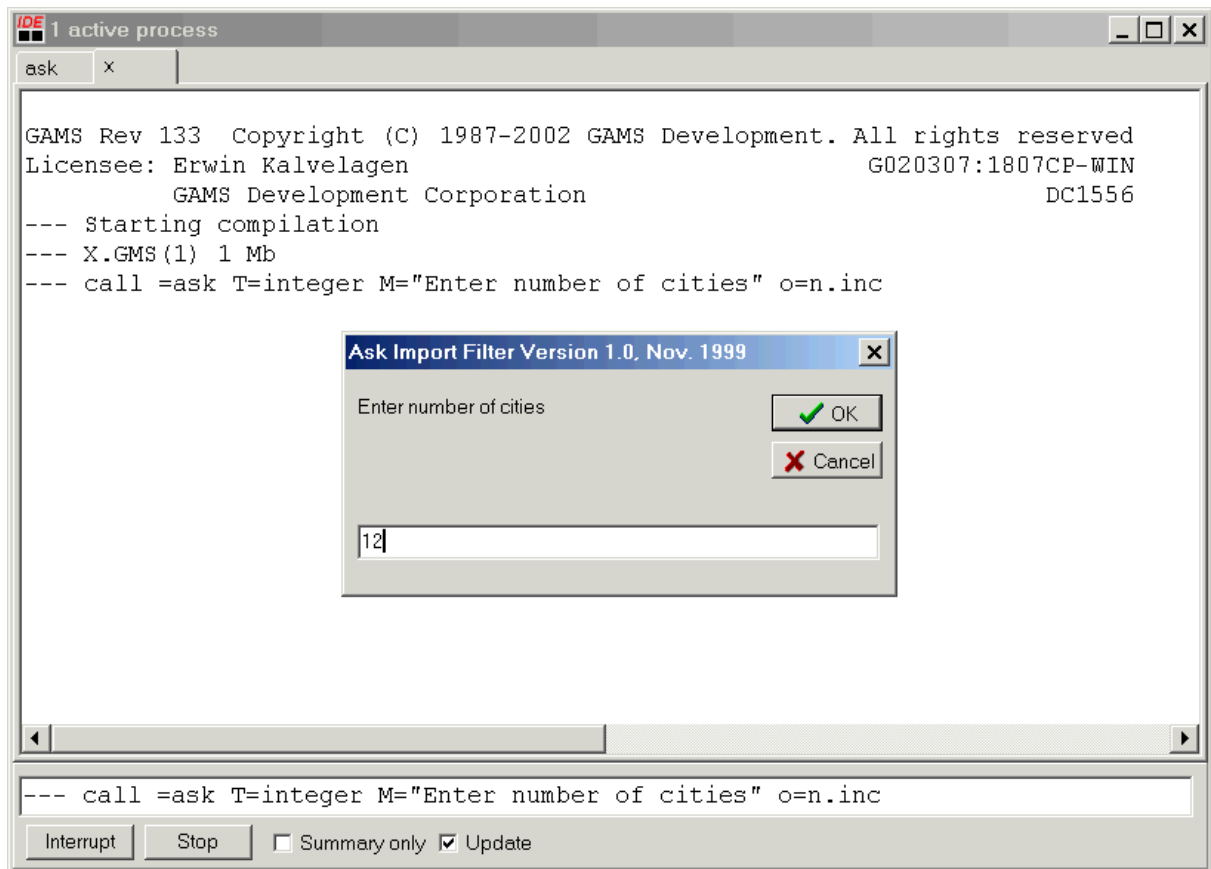
```
$call =ask T=integer M="Enter number of cities" o=n.inc
scalar n 'number of cities' /
$include n.inc
/;
display n;
```

The \$call statement will invoke the ASK tool. If ASK.EXE is not located in the GAMS system directory but placed somewhere else you may have to provide a path, as in:

```
$call =d:\util\ask T=integer M="Enter number of cities" o=n.inc
```

The parameter *T=integer* indicates we want to ask for an integer number ('T' means type). The parameter *M="text"* specifies the question text. Finally *O=filename* sets the name of the include file that ASK should create.

When we run this GAMS fragment from the IDE, we will see:



After enter a number and hitting the OK button GAMS will continue. The listing file will demonstrate clearly how the include file was formed:

```

2 scalar n 'number of cities' /
INCLUDE D:\GAMS PROJECTS\ASK\N.INC
4 * Ask Import Filter Version 1.0, Nov. 1999
5 * Erwin Kalvelagen, GAMS Development Corp.
6 12
7 /;
8 display n;
Include File Summary
SEQ GLOBAL TYPE PARENT LOCAL FILENAME
1 1 INPUT 0 0 D:\GAMS PROJECTS\ASK\X.GMS
2 1 CALL 1 1 =ask T=integer M="Enter number of cities" o=n.inc
3 3 INCLUDE 1 3 .D:\GAMS PROJECTS\ASK\N.INC

```

In this case '12' was entered and the OK button was pressed. If the user pressed the CANCEL button, an error will be generated and the listing file will show that the CANCEL button was pressed:

```

2 scalar n 'number of cities' /
INCLUDE D:\GAMS PROJECTS\ASK\N.INC
4 * Ask Import Filter Version 1.0, Nov. 1999
5 * Erwin Kalvelagen, GAMS Development Corp.
6 * Cancel was pressed
7 /;
**** $!
**** LINE 4 INPUT D:\GAMS PROJECTS\ASK\X.GMS
**** 1 Real number expected
8 display n;

```

In case you want to limit the integer that the user is allowed to enter you can specify a lower and an upperbound as in:

```

$call =ask T=integer M="Give integer, between 0 and 5" L=0 U=5 O=n1.inc
scalar n1 'an integer 0..5' /
$include n1.inc
/;
display n1;

```

This will only accept values between 0 and 5.

To allow the user to specify a floating point number, we can use T=float. An example is:

```

$call =ask T=float M="Give floating point number, no bounds" O=x.inc
scalar x 'real' /
$include x.inc
/;
display x;

```

The floating point popup window can be told to make sure the number entered is within certain bounds, using the *L=lowerbound* and *U=upperbound* syntax:

```

$call =ask T=float M="Give floating point number, between 0 and 5" L=0 U=5.0 O=x1.inc
scalar x1 'real' /
$include x1.inc
/;
display x1;

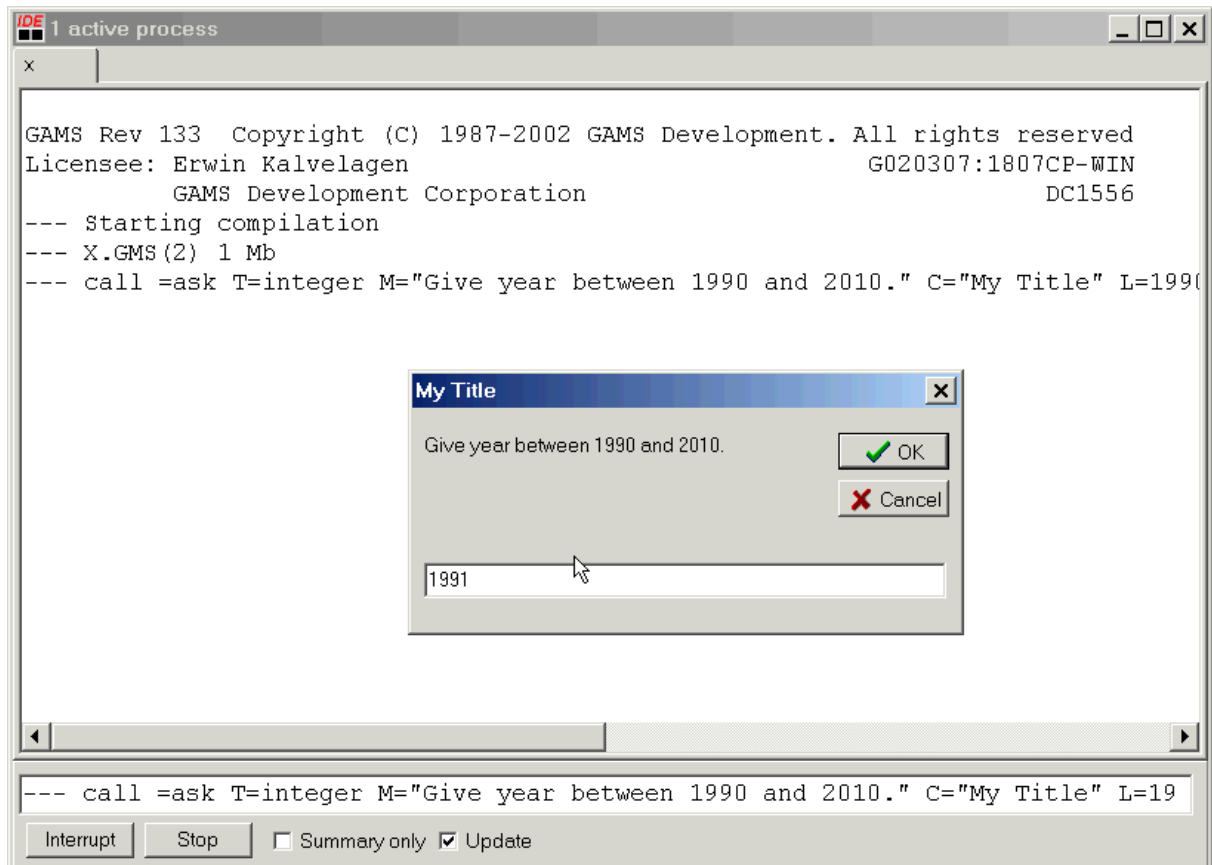
```

Up to now, the include file generated by ASK just contained a single number. ASK can generate more complicated include files. Consider the example:

```

* import a set
$call =ask T=integer M="Give year between 1990 and 2010." C="My Title" L=1990 U=2010 R="set
i /1990*%s/;" O=i.inc
$include i.inc
display i;

```



The parameters *C="caption"* and *R="resultstring"* are new. The caption is simple: it sets the name of the window. The result string is a string that is returned after ASK has substituted %s for the result. I. e. if we enter '1991', then the result written to the include file is "set i /1990*1991/;"

The listing file can be used to check the correct behavior:

```

1 * import a set
INCLUDE      D:\GAMS PROJECTS\ASK\I.INC
4 * Ask Import Filter Version 1.0, Nov. 1999
5 * Erwin Kalvelagen, GAMS Development Corp.
6 set i /1990*1991/;
7 display i;
Include File Summary
SEQ  GLOBAL TYPE      PARENT  LOCAL  FILENAME
  1          1 INPUT      0       0  D:\GAMS PROJECTS\ASK\X.GMS
  2          2 CALL        1       2  =ask T=integer M="Give year between 1990 and
2010." C="My Title" L=1990 U=2010
  3          3 INCLUDE     1       3  .D:\GAMS PROJECTS\ASK\I.INC
                                     R="set i /1990*%/;" O=i.inc

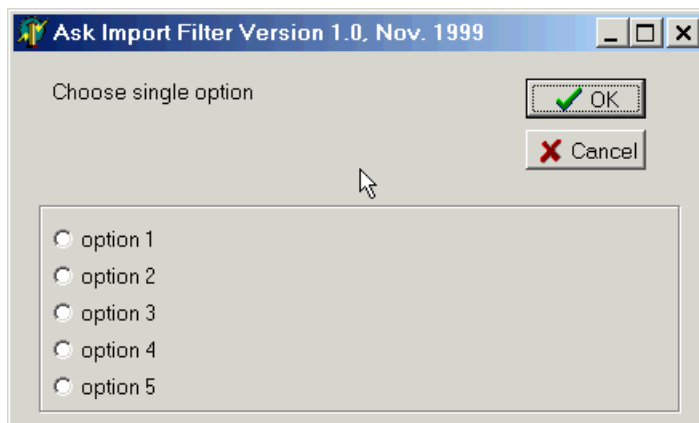
```

Radio buttons can be used through the parameter *T=radiobuttons* as in:

```

* import a number through radio buttons
$call =ask T=radiobuttons M="Choose single option" D="option 1|option 2|option 3|option 4|
option 5" E="1|2|3|4|5" R="scalar n2 option /%/;" o=n2.inc
$include n2.inc
display n2;

```



The parameter *D="option 1|option 2|option 3|option 4|option 5"* specifies the text of the options shown. The list *E="1|2|3|4|5"* gives the return (exit) strings when a certain option is chosen. I.e. if the second option is chosen in the list specified with the D parameter, then the second string in the E list is returned. The result is again substituted in the string specified in the R parameter if it exists. In this example the command line became rather long and difficult to handle. In addition some Windows systems have restrictive maximum lengths for the command line. Therefore we offer the possibility to specify the command line arguments in a separate external text file. This text file is passed using @filename.

Assume the file 'ask.opt' looks like:

```

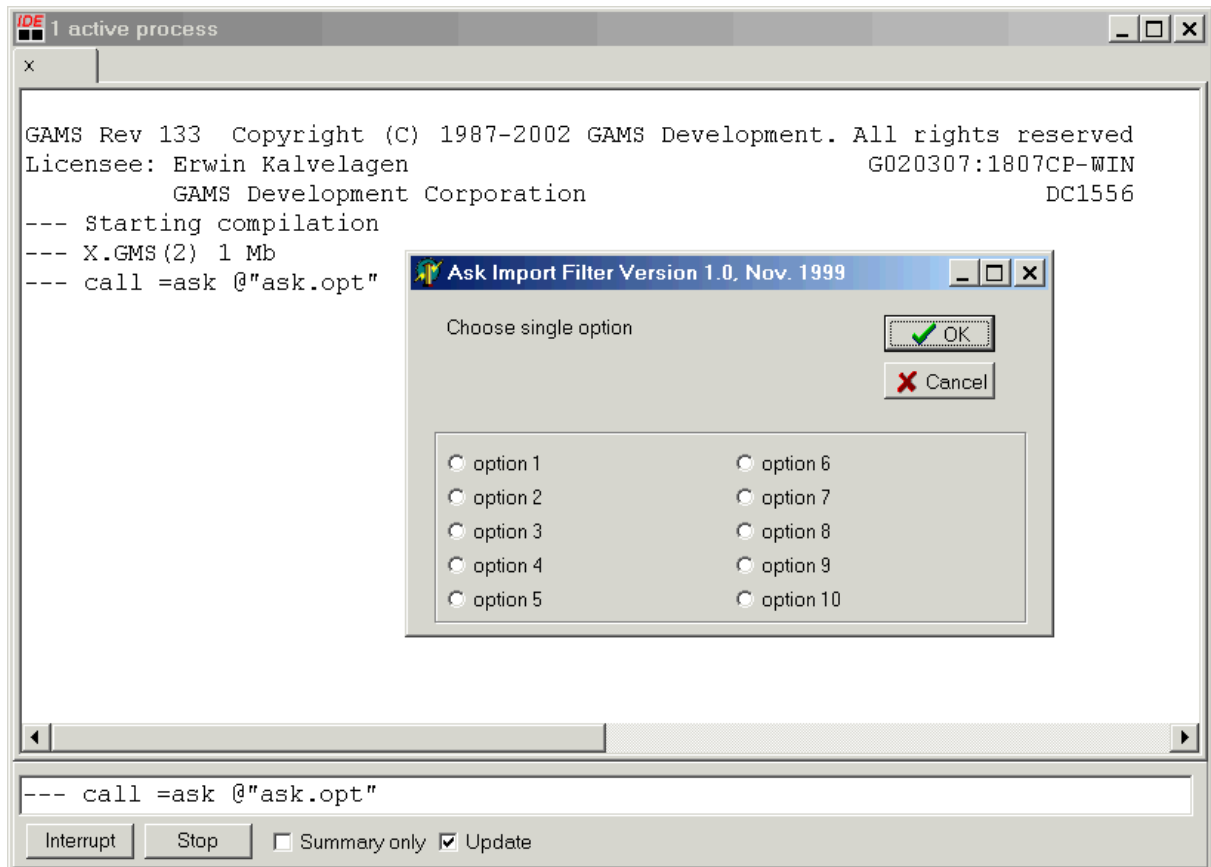
T=radiobuttons
M=Choose single option
D=option 1|option 2|option 3|option 4|option 5|option 6|option 7|option 8|option 9|option
10
E=1|2|3|4|5|6|7|8|9|10
2
R=scalar n3 option /%/;
O=n3.inc

```

Every command line parameter is specified on a separate line. Notice the strange option '2'; this tells ASK to display the radio buttons in two columns. We can use this option file as follows:

```
* id, now 2 columns and using a parameter file
$call =ask @"ask.opt"
$include n3.inc
display n3;
```

The quotes around the filename are optional, and really only needed if the filename contains blanks.



If you want to keep all the logic in one place, then one can use GAMS to generate the option file. It is noted that it is not possible to use the PUT facility for this. I.e.:

```
file f /asktest.opt;
put f;
put 'T=checkboxlistbox'/
    'M=Choose multiple options'/
    'D=option 1| option 2| option 3| option 4| option 5' /
    'E=1| 2| 3| 4| 5' /
    'R=%s checked list box choice' /
    'O=k2.inc' /;
putclose;
$call =ask @asktest.opt
set k2 /
$include k2.inc
/;
display k2;
```

is not correct: the \$call is handled at compile time, before the PUT statement has done its work.

However, one could use the following:

```

$echo ' T=checkboxlistbox' > asktest.opt
$echo ' M=Choose multiple options' >> asktest.opt
$echo ' D=option 1| option 2| option 3| option 4| option 5' >> asktest.opt
$echo ' E=1| 2| 3| 4| 5' >> asktest.opt
$echo ' R=%s checked list box choice' >> asktest.opt
$echo ' O=k2.inc' >> asktest.opt
$call =ask @asktest.opt
set k2 /
$include k2.inc
/;
display k2;

```

or better:

```

$onecho > asktest.opt
T=checkboxlistbox
M=Choose multiple options
D=option 1| option 2| option 3| option 4| option 5
E=1| 2| 3| 4| 5
R=%s checked list box choice
O=k2.inc
$offecho
$call =ask @asktest.opt
set k2 /
$include k2.inc
/;
display k2;

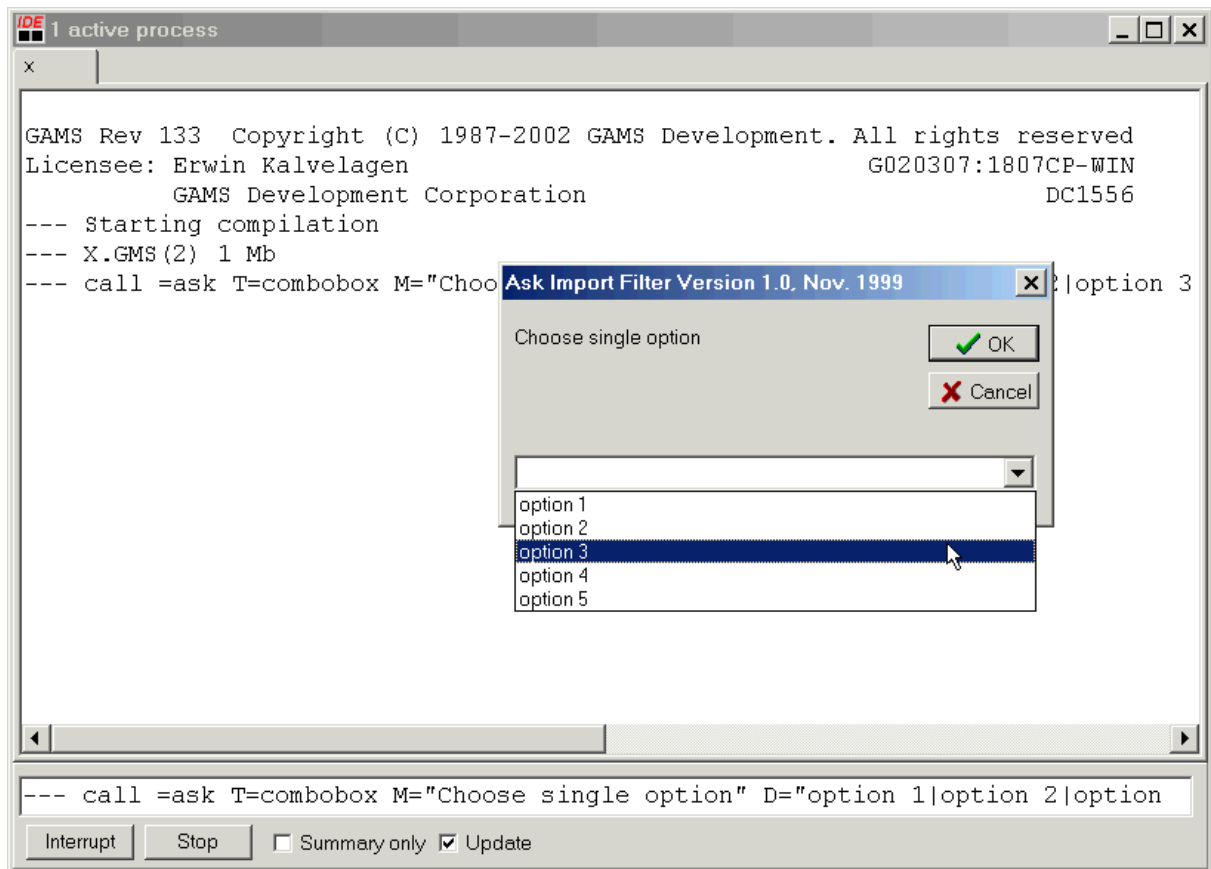
```

The next type is T=combobox which also allows a single selection:

```

* import a number through a combo box
$call =ask T=combobox M="Choose single option" D="option 1| option 2| option 3| option 4|
option 5" E="1| 2| 3| 4| 5" R="scalar n4 option /%s/;" O=n4.inc
$include n4.inc
display n4;

```



As an example consider the case where the model comes with three data sets: a small one, a medium sized one and a large data set. Each data set is stored in a separate include file: *small.inc*, *medium.inc* and *large.inc*. We want to ask the user which data set should be used and the correct include file should be used. This can be accomplished by:

```
$echo 'T=combobox' > ask.opt
$echo 'M=choose data set' >> ask.opt
$echo 'D=Small data set|Medium data set|Large data set' >> ask.opt
$echo 'E=small.inc|medium.inc|large.inc' >> ask.opt
$echo 'R=$include %s' >> ask.opt
$echo 'O=dataset.inc' >> ask.opt
$call =ask @ask.opt
$include dataset.inc
```

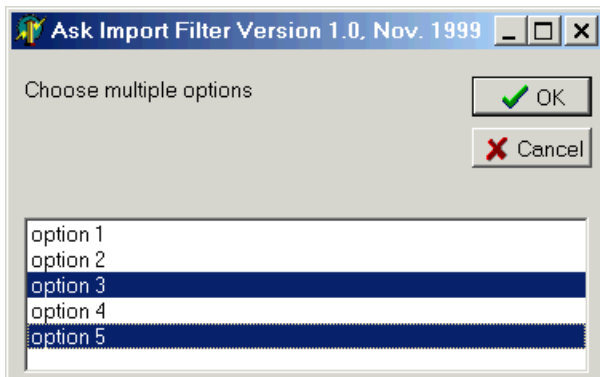
Newer GAMS systems allow:

```
$onecho > ask.opt
T=combobox
M=choose data set
D=Small data set|Medium data set|Large data set
E=small.inc|medium.inc|large.inc
R=$include %s
O=dataset.inc
$offecho
$call =ask @ask.opt
$include dataset.inc
```

In this case *dataset.inc* will contain a single line: another include statement, which is either *\$include small.inc*, *\$include medium.inc* or *\$include large.inc*.

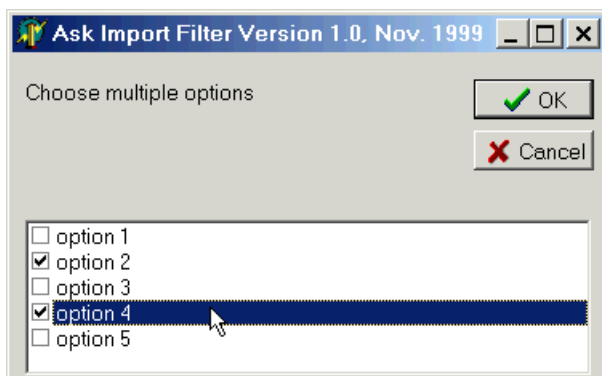
In some cases it may be useful to be able to select multiple items from a list. This can be done with `T=listbox`:

```
* import a set through a listbox
$call =ask T=listbox M="Choose multiple options" D="option 1| option 2| option 3| option 4|
option 5" E="1| 2| 3| 4| 5" R="%s list box choice" O=k.inc
set k /
$include k.inc
/;
display k;
```



Selecting multiple entries involves holding down the CTRL key. Sometimes a convenient alternative is

```
T=checkboxlistbox:
* import a set through a checked listbox
$call =ask T=checkboxlistbox M="Choose multiple options" D="option 1| option 2| option 3| option
4| option 5" E="1| 2| 3| 4| 5" R="%s 'checked list box choice' " O=k2.inc
set k2 /
$include k2.inc
/;
display k2;
```



When we select options 1, 3 and 5, the following include file is generated:

```
* Ask Import Filter Version 1.1, Aug. 2002
* Erwin Kalvelagen, GAMS Development Corp.
1 'checked list box choice'
3 'checked list box choice'
5 'checked list box choice'
```

which will populate the set k2 as follows:

```
1 * import a set through a checked listbox
```

```

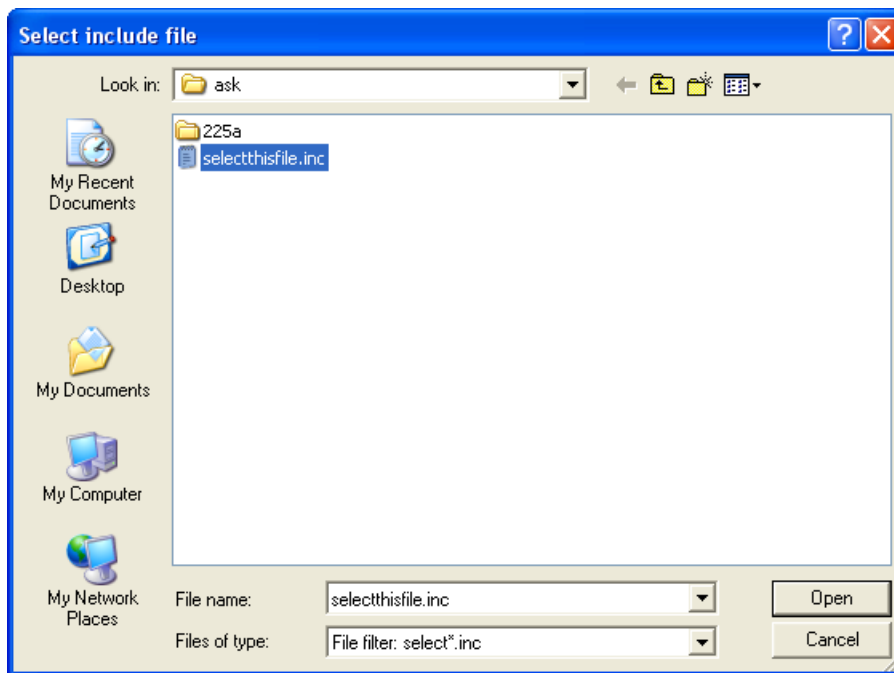
3 set k2 /
INCLUDE D:\GAMS PROJECTS\ASK\K2.INC
5 * Ask Import Filter Version 1.1, Aug. 2002
6 * Erwin Kalvelagen, GAMS Development Corp.
7 1 'checked list box choice'
8 3 'checked list box choice'
9 5 'checked list box choice'
10 /;
11 display k2;
----      11 SET k2
1,      3,      5

```

The last type is the generic string type: $T=string$. This allows the user to enter any string, which is then copied to an include file as is.

FileOpenBox

The type $T=fileopenbox$ will display a file open dialog box from which the user can select a file.



Related options are $I=InitialDirectory$ and $F=Filter$. A complete example to ask for an include file could be:

```

$call =ask T=fileopenbox I="%system.fp%" F="select*.inc" o=fln.inc R="$include '%s'"
C="Select include file"
$include fln.inc

```

This will open a file open dialog box with the starting directory being the GAMS project/working directory (this is also where GAMS looks for include files by default). Only files with mask $SELECT*.INC$ are shown. The file $FLN.INC$ will contain an include statement with the file the user has selected.

A related method would be:

```
$call =ask T=fileopenbox I="%system.fp%" F="select*.inc" o=fln.inc R="$setglobal incfile '%s'" C="Select include file"
$include fln.inc
$include %incfile%
```

where we use a *\$setglobal* to set the macro *incfile* to contain the user-specified file name. To let the user choose from a set of related GDX files, one could use something like:

```
$call =ask T=fileopenbox I="%system.fp%" F="myproject*.gdx" o=setgdxname.inc R="$setglobal gdxfile '%s'" C="Select GDX file"
$include setgdxname.inc
$gdxin %gdxfile%
$load i
$load j
```

In case you want to ask for a filename of a file to be written, use the type *T=filesavebox*. E.g.:

```
$call =ask T=filesavebox I="%system.fp%" o=fln.inc R="$setglobal gdxfile '%s'" C="Specify gdx file"
$include fln.inc
set i /a,b,c/;
execute_unload '%gdxfile%',i;
```