Development Mode Features

This is the approved revision of this page, as well as being the most recent.

> The printable version is no longer supported and may have rendering errors. Please update your browser bookmarks and please use the default browser print function instead.

□

Contents

Development Mode Features

Adding Variables to IFs

You will only be able to do this if you have the source code of IFs and a license for its use (and if you have Visual Basic programming skills).

To add a variable to IFs (or to eliminate one):

- Edit IfsVar.mdb. This is an MS Access file. Add a new line for a new variable and provide values for the various fields by looking at similar, already existing variables in the file.
- After adding a variable to IFs, you must rebuild the common block,

rebuild the base case, and re-run the model.

- Be aware that this topic has only described the addition of a variable to the input/output interface of IFs. It will have values of 0 unless you do something with it in the model code:
- Use it as an output variable, calculated from other variables in IFs.
- Use it as an input parameter or variable with specified initial condition. If you want to do this, you would also need to learn how to change the data files of IFs and the code in the pre-processor.

Legacy Approach to Adding a Variable. In older versions of IFs that process was harder. The following description should not be necessary for those with new copies of the source code:

Edit DECLARE2.BAS to change the total number of positions in the vector that stores all results from IFs. Near the bottom of that file you will find code like this:

Type DATAPT2 '- New system

elem2(5000) As Single

End Type

Type DATAPT3

elem3(3456) As Single

' elem3(4698) As Single

' Dim BufElem in RebBase sets total size of buffer; may need to

' expand that also - now set at 70000

End Type

DATAPT2 contains an element which is always dimensioned to 5000, because data is transferred to and from result files in a buffer which is 5000 positions in length. The number of positions needed will change with the number of variables in IFs and their dimensionality. You will need to change the dimensionality of elem3 in DATAPT3 in order to pick up the residual positions after the 5000-position buffer is used as frequently as needed.

When adding a variable of known length (e.g. a scalar of length 1 or a regionalized variable across x regions), it is possible to adjust elem3's dimensionality on the front end of an attempt to rebuild common. Often it is difficult to calculate the number of positions needed (especially for the change in subscripts applied to many variables). In this case, rebuild the common and let that procedure abort with a message that will tell you the new size needed for the total number of positions; adjust elem3 dimensionality accordingly and rebuilt the common again (hopefully now with success).

It is possible to change the dimensionality of variables by edition IFs.grp. Again, you would need similarly to change the number of positions in DECLARE2.BAS and rebuild common.

For your information, the total number of positions (FILEN) in each year of the result file is calculated from IfsVar.mdb when the model is initiated. You need not do so (its length will be elem2 dimensionality * x(as needed) + elem3 dimensionality.

Rebuilding the Common Block

This option will appear as a sub-option of Change Menu on your main menu only if you have:

1. The Professional Edition.

2. Visual Basic and the model code. This option is for model-builders only and requires both technical skills and a site license. Inside the IFsInit.mdb file in the \Code directory, the DevelopmentMode$ variable must have a True specification to activate the option.

You will use this option when you have

1. Changed the variables in IFSVAR.mdb (additions or deletions).

2. Changed dimensionality within the model (either by changing the maximum permissible regionalization in FRM_IFS or by changing other dimensions in IFS.GRP.

After completing a rebuild of common, you must:

1. Exit the model and Visual Basic.

2. Restart Visual Basic and the model (this loads the newly revised files with the altered common block).

3. Rebuild the base (because rebuilding common changes dimensionality and therefore sets all values in IFSBASE.RUN to 0.0 to avoid errors).

4. Rerun the model.

Special Modes of .exe Use

Several special modes have been created for using the .exe file. The modes are:

1. **Normal Mode** (1 argument).

Arg1 = mode number (i.e. 1).

Example: IFs.exe 1 This behaves just like calling IFs.exe without an argument.

2. **Drill Down Mode** (3 arguments). This was created so that a program that was executing IFs (such as the Computer Assisted Reasoning System or CARS of the RAND Pardee Center and Evolving Logics) can identify and load a particular scenario or case from the SensitivityInput.mdb file (see below), run it, and be positioned for further analysis of it using the IFs interface.

Arg1 = mode number (i.e. 2).

Arg2 = run number from the SensitivityInput.mdb file (see below).

Arg3 (optional) = run horizon (e.g. 2050).

Example: IFs.exe 2 54 2020. This would pick the 54th scenario from the input file, load the parameters associated with it, run the model, and turn control to the user.

3. **Invisible Mode** (2 arguments). This was created so that a program that was executing IFs (again like CARS) or the sensitivity option of IFs itself could run multiple scenarios, placing results for selected variables into an input file. The scenario descriptions must again be in the SensitivityInput.mdb file. The variables that the user wants to see as outputs must be specified in the SensitivityOutputVars.mdb. The results generated by IFs will be put for each scenario into a SensitivityOutput.mdb file.

Arg1 = mode number (i.e. 3)

Arg2 = run horizon (e.g. 2015)

Example: IFs.exe 3 2015. If there are many scenarios in the SensitivityInput.mdb file, this

can create an execution of IFs that runs for a very long time (even days). It does not need a run number from SensitivityINput.mdb because it runs all scenarios/cases in that file.

4. **Invisible Single Run Mode** (2 arguments). This mode was created so that a user can load a particular scenario, run it, and put selected results into an output file. The variables that the user wants to see as outputs must be specified in the SensitivityOutputVars.mdb. The results generated by IFs will be put for the selected scenario or case into a SensitivityOutput.mdb file.

Arg1=mode number (i.e. 4)

Arg2 = run number from the SensitivityInput.mdb file.

Example: IFs.exe 4 8

5. **Invisible Web Mode** (3 arguments). This mode was created specifically to run IFs.exe on the web. The web-based version of IFs only includes the interface or GUI elements of IFs. For actual calculations it relies on the .exe file of IFs created in Visual Basic development mode. When a model run is called for on the web, it passes appropriate arguments to the .exe. The result is a run of the model with full results put into the working.run file.

Arg1 = mode number (i.e. 5)

Arg2 = run horizon (e.g. 2020)

Example: IFs.exe 5 2020

6. **Invisible Batch Scenario Mode** (3 arguments). This mode loads and runs all of the .sce files in the Scenario directory of IFs and puts results into .run files with the same name as the .sce files. It processes also all subdirectories of the Scenario directory. If there are many .sce files, this can take a very long time to run.

Arg1 = mode number (i.e. 6)

Arg2 = run horizon (e.g. 2100)

Arg3 = -1 if it is being run on a stand-alone, non-web installation; = session number if it is being run on the web (i.e. the session number where it is going to look for the scenario file directory and subdirectories (e.g. ifs\scenario\session0)

Example: IFs.exe 6 2100 -1 (will run all scenarios for a stand-alone application).

All of the above examples assume that IFs is being initiated from a command line or inside a program (and the full path for the model's .exe must be given). In a VB development environment, the modes can be turned on by going to the Project option, IFs Properties sub-option, the Make Tab, and setting the appropriate arguments in the command line (e.g. 6 2005 -1). Note that the name of the model is not specified in the command line.

Some of the above modes require one or all of three files. These were once text files (.dat) but have been converted to .mdb (Access) files to make them more transparent to the user. See examples in the IFs Sensitivity directory so as to understand the field specifications.

SensitivityInput.mdb file.

SensitivitiyOutputVars.mdb file. It is necessary to select the output variables of interest to you in the set of sensitivity runs. The model stores these in a file named SensitivityOutputVars.dat. You can select up to 500 of these (each country/region counts as one output variable even for the same variable).

SensitivityOutput.mdb file.

Procedures for IFs Development

This topic will be meaningful only to those having a site license for the source code and doing IFs development or for those in similar modeling projects.

Development Topics:

Visual Basic. IFs was developed using Visual Basic.

- Installing IFs Source Code
- Adding Variables
- Rebuilding the Common Block

Installing IFs Source Code

Installation of object code and necessary controls. Before attempting to install the source code, install the object code of the Professional Edition of IFs on the target machine (i.e., installed the Professional Edition). The object code of IFs normally is installed into a subdirectory called International Futures or IFs in the C:\Program Files directory of your hard drive. The installation process will place a number of necessary controls onto other directories (including Windows\System) of the target machine. Among the controls installed are a set of ActiveX controls that were part of VB 4.0 and VB 5.0, that are used by IFs, but that are not installed by VB 6.0 (AniBtn32.ocx; Gauge32.ocx, Graph32.ocs, Gsw32.EXE, Gswdll32.dll, Grid32.ocx, KeySta32.ocx, Spin32.ocx, and Threed32.ocx). Please note, however, that installation of a run-time copy of IFs will not register all of these controls for design-time use (see subsequent steps).

Install Visual Basic 6.0 on the target machine. As of early 2005, Service Pack 6 was available for VB 6 and it should also be installed. Design use of IFs requires a license for Visual Basic 6.0 Professional from Microsoft.

Design-time registration of controls. As noted above, there are some controls used by IFs that were part of VB 4.0 and VB 5.0 but are not installed and registered for design-time use by VB 6.0 (see the list in parentheses in step 1). The contorls can be found in the Common\Tools\VB\Controls directory of the VB 6.0 installation disk and the ReadMe.txt file (reproduced at the end of this write-up) describes their registration. In January 2005 these were also added to a VB Old Controls subdirectory of the IFs development directory. It is necessary to copy these files to your Windows\System directory and register them. There are several ways to register them. It can apparently be done from within Visual Basic. The ReadMe.txt below suggests using RegSvr32.exe (execute this program with each of the controls as a target or drag and drop the controls onto the control) – this procedure seems to work. The ReadMe.txt also says you can also use RegEdit.exe (apparently you can register them all at once by by executing RegEdit with Vbctrls.reg as the target). In January 2005 it seemed to be necessary to drag and drop the Graph32.ocx unto RegEdit.exe (within

/System) to get it registered correctly. Although it should not be necessary to use, the set-up wizard that creates the object code installation CAB file also creates a directory named Support that contains copies of all of the installed files (including the controls from earlier VB versions); these constitute a secondary location for finding components that may somehow be missing.

Installation of source code. The CD-ROM includes all of the source code necessary to run and modify International Futures (subject to provisions of the license agreement). In order to keep the source code separate from the ojbect code, it is highly recommended that you copy all of the directories and files from the CD-ROM into a subdirectory (IFs) of a directory called C:\My Documents on your hard drive. (If you copy the source code into another location, it will be necessary to change code specifying StartDirec$ in the Form-Load routine of the frm_IFs.) The CD-ROM does not include a setup or install routine. You simply copy the IFs directory and all of its files and subdirectories to the C:\My Documents directory.

IFs.ini. The IFs.ini file needs to be on the Windows directory for both development and use of the model.

Inetwh32.dll. This file needs to be on the system32 directory in order to allow use of IFsHelp. The SetBrows file also should be there.

Additional software required. IFs also requires a license for MapObjects 3.0 from ESRI. This software must be installed prior to your use of IFs. IFs also uses components from Total VB Statistics from FMS. If you have installed the object code of IFs successfully and followed the above steps, your machine should have the needed components from FMS.

Initiating use of IFs. Once all of the needed software is installed, start Visual Basic. Initiate a New Project. Select the Existing tab and browse for IFs (presumably on C:\My Documents\). Go to the Code subdirectory of IFs and you should find the file IFs.vbp (the Visual Basic master program file for IFs). Activate it with a double click and all of the files of IFs should be read into Visual Basic. If you receive error messages during this process, make a note of them and continue if possible; you may still having missing components in your projects (see the next step).

Earlier use of ESRI MapObjects. At one time MapObjects was used instead of Blue Marble's GeoObjects. The following instructions will be unnecessary unless IFs has returned to its use: Check to see if ArcExplorer Legend is a selected (checked) component. If it is not (and it probably will not be), use the Browse button to find the needed file, MO2Legend.ocx. If you have installed MapObjects 2.0, you will normally find it in the directory C:\Program Files\Esri\MapObjects2\Samples\Vb\MoView2. After you have selected Project\Components from the VB menu you will be able to use the browser search for it, click on it, and thereby activate it; you should see it appear and checked in the list of components available to your Visual Basic project. You also need to make sure that the component ESRI MapObjects 2.0 is in the project. Scroll down the alphabetical list of components. When you find it listed, make sure it is checked.

Making Sure Components are Available. Before starting to run IFs, select Project from the VB menu and Components from the submenu. Components that you may find checked or should attempt to check yourself are: Desaware Animated Button Control; MicroHelp Guage Control; MicroHelp Key State Control; Microsoft Comm Control 6.0; Microsoft Common Dialog Control 6.0; Microsoft Data Bound Grid Control 5.0 (SP3); Microsoft FlexGrid Control 6.0; MicroSoft Grid Control; Microsoft Masked Edit Control 6.0; Microsoft Picture Clip

Control 6.0; Microsoft Window Commons Control 6.0; Outrider SpinButton Control; Pinnacle-BPD Graph Control; SHERIDAN 3D CONTROLS. If you attempt to check a control and are unable to do so, make a note of it, but continue. Some of these may be obsolete for your version of IFs.

Making Sure References are Selected. The following references are selected/checked in my project file (see yours by selecting Project from the VB menu and References from the submenu): Visual Basic for Applications; Visual Basic runtime objects and procedures; Visual Basic objects and procedures; Microsoft Excel 9.0 Object Library; Standard OLE Types; FMS Total VB Statistics 6; Microsoft DAO 3.6 Object Library (before upgrading to Office 2000, it was Microsoft DAO 2.5/3.51 Compatibility Library). I do not believe it is necessary to have FMS Total VB Statistics 6 in the development environment, but I am not certain.

Starting IFs. After loading IFs and activating the controls it needs in the VB project, select Run from the main VB menu and Start from the submenu. This should initiate IFs. If it does not, make a note of the nature of any error messages or the point at which problems arise. If IFs does start properly, test it thoroughly to make sure that all files and components are available and working. Then you can proceed to edit the source code and change IFs.

Using RoboHelp for Help System updates. The RoboHelp software (company acquired by MacroMedia in 2004) has been used for Help system development. It needs to be licensed, installed, and linked to the latest version of MS Word for development involving updates of the Help system.

The Wise installation software. It has been used for creating installations of IFs on stand-alone machines. In 2004 version 9.02 was licensed. The latest .wse files will identify a number of additional components needed for creation of setup .exe files. These include a substantial number of files for Proessentials graphics. They can be put into the System32 directory.

Although they are not required for the use of source code of IFs, the following software has been very useful in the IFs project:

Blue Sky Software, RoboHelp Office. Used for the construction/update of the Help system.

FMS, Total VB Statistics. Used for creation of the code of the cross-sectional statistics capability.

Visio Corporation, Visio Standard. Used for creation of the flow charts in the Help system.

Readme.txt (from the VB installation disk)

\Tools\Controls

This directory contains all of the ActiveX Controls that shipped with Visual Basic 4.0/5.0 Professional and Enterprise Editions, which are no longer shipping with Visual Basic 6.0.

AniBtn32.ocx

Gauge32.ocx

Graph32.ocx

Gsw32.EXE

Gswdll32.DLL

Grid32.ocx

KeySta32.ocx

MSOutl32.ocx

Spin32.ocx

Threed32.ocx

MSChart.ocx

The \Tools\Controls\BiDi directory contains a Bi-directional version of Grid32.Ocx.

If you have Visual Basic 5.0 Professional or Enterprise Editions installed on your machine, you should already have these ActiveX controls available to you in Visual Basic 6.0.

Graph32.ocx has been updated to work properly in Visual Basic 6.0 and it requires two additional support files: gsw32.exe and gswdll32.dll. You must place the three files together in the \Windows\System directory or the control will not function properly.

If you do not have these controls and wish to use these in Visual Basic 6.0, you can install them by:

1. Copy all of the files in this directory to your \WINDOWS\SYSTEM directory.

2. Register the controls by either Browsing to them in Visual Basic itself, or manually register them using RegSvr32.Exe. RegSvr32.EXE can be found in the \Tools\RegistrationUtilities directory. The command line is:

regsvr32.exe grid32.ocx

3. Register the design time licenses for the controls. To do this, merge the vbctrls.reg file found in this directory into your registry. You can merge this file into your registry using RegEdit.Exe (Win95 or WinNT4) or RegEd32.Exe (WinNT3.51):

regedit vbctrls.reg (or other reg files associated with the controls)

Screen Captures for Documentation

To capture a screen image, display it on the screen and depress Alt-Print Screen. To place it in a graphics file, start Visio and use Control-V to paste it on a Visio Page. Then after sizing, etc., Save As will allow saving in a wide variety of forms. WMF format saves it as a large file that can be imported to Word (using Insert-Picture), etc. Visio also allows printing of the screen image directly.
