

# **Using Custom AppBuilder Generator Programs**

May 2009

By

Jon Bradley

The purpose of this guide is to explain the new configurability of AppBuilder's program generation process. It covers the use of BASIS provided example generator programs, and general design guidelines for creating a custom generator program.

# Using Custom AppBuilder Generator Programs

## Introduction

AppBuilder, like GUIBuilder, uses a BBJ® program to generate a BBJ program based on the contents of a .gbf file. The original (and default) generator is *gb\_func*. In BBJ 8.0, AppBuilder introduced beta support for custom program generators. These custom generator programs access the .gbf's contents through the [AdvancedGBFWrapper](#) object. BBJ 8.0 and above includes two sample generator programs: *ProcessEventsBuilder.src* creates a minimal PROCESS\_EVENTS-based program and *LegacyPEBuilder.src* generates a similar program, but also includes support for some of the more commonly used [control variables](#) from *gb\_func*.

## How to Select a Custom Generator Program in AppBuilder

Select the generator program via **Tools > Options**. Next, select the **FormAppBuilder** options, and then choose the generating program from the list. Select the BASIS-provided programs from the list or type in the name of your own custom generator program.

## Compile Without Output Support

Selecting any generator program other than *gb\_func* configures the AppBuilder system to respect the “Compile without output” setting in **BBJ Compiler Settings**. This causes the back end to compile the generated program to check for errors, but it does not create a .bbj file. Pressing the [RUN] button for the AppBuilder project runs the .src file if “Compile without output” is enabled.

## Design Guidelines for Creating Your own Generator Program

Modifying the BASIS-supplied generators is strongly discouraged since future installs will overwrite them. It is better to copy the supplied generators into a new file or start from scratch. There are very few things one **MUST** do when creating a generator program. The following example, which produces a simple “hello world” program, demonstrates the necessary functionality and interface any generator program must contain.

```
0001 build_program:
0002 enter gbf!
0003 programName$ = gbf!.getProperty("Program Name") + ".src"
0004 program = unt
0005 open (program,mode="O_CREATE,O_TRUNC")programName$
0006 print (program)"print ""Hello, World!""
0007 print (program)"escape"
0008 close (program)
0009 exit
```

Line	Comments
1	The <code>build_program</code> label is required; this is the entry point.
2	An <code>enter objvar!</code> statement is required. Your generator program will retrieve information about the <code>.gbf</code> project file from this <a href="#">AdvancedGBFWrapper</a> object.
3..5	The program must create the program file specified by the <b>Program Name</b> property. The generated program must have a <code>.src</code> extension.
8	Close the channel of the file to which you are writing.
9	It is important to <b>exit</b> , not <b>release</b> , so the AppBuilder backend has a chance to compile your generated program and report any errors.

## ***AdvancedGBFWrapper***

The back-end system passes an [AdvancedGBFWrapper](#) object to all generators (other than `gb_func`). It provides easy access to the contents of the `.gbf` file. The `ProcessEventsBuilder.src` program can be read as a simple example of how to work with this object.

## ***Working With Existing .gbf Files***

The BBx® language has evolved over the years to provide object-oriented interfaces for functionality that the `gb_func` generator program implements with large blocks of generated BBx code. The `PROCESS_EVENTS` verb dispatches events automatically, removing the need for a generated `READ RECORD` loop with individual `GOSUBs` to event handler subroutines. All BBj events implement a `BBjEvent` object, which allows for more readable code.

If your `.gbf` files do not depend on any of the code or [variables](#) generated by `gb_func`, or code merged in from the `*.cod` files, then your program should build and run using any of the BASIS-provided generator programs. The `LegacyPEBuilder.src` provides support for some of the more commonly used variables, including `gb__args$`, `gb__args`, `gb__sysgui$`, `gb__sysgui`, `gb__win$`, `gb__event$`, and `gb__notice$`.