Compile Utility

Documentation for the Compile Utility is broken into four sections:

1. Goals this tool attempts to achieve
2. Broad overview of how the tool operates
3. Build setup instructions
4. Detailed overview regarding execution of builds

GOALS:

I believe the build process is an extremely important piece of any software development cycle. A flexible and reliable method is needed to create builds nightly (automated), at deployment time, and on demand (when necessary). If the nightly builds fail in some way, QA could waste valuable time testing and logging issues with code that is not “current” due to the build failure(s). Deploying an incorrect build will most likely introduce problems at the client. These problems will result in the need to allocate resources to identify and solve the issues that have arisen. In short, if we cannot produce a deliverable product with a high degree of competence (from the “build” perspective), the end result will be costly to say the least.

Through a combination of configuration flexibility, robust error handling, and comprehensive logging, the Compile Utility tool provides a mechanism for creating “high confidence” builds, thus greatly reducing (if not totally eliminating) errors introduced by the user interface located between the keyboard and the seat.

BROAD OVERVIEW:

The Compile Utility tool is completely data driven. There are no build details (such as propath entries, database connection parameters, directory/file lists, etc.) hard coded into the engine. A configuration file (Compile.cfg) drives the build process and eliminates the need to maintain multiple versions of hard coded compile programs. This approach allows virtually anyone to understand and configure a build without being intimately

familiar with the Progress source code (or programming in general) utilized to execute these tasks behind the scenes.

The configuration file consists of independent entries, each of which represents a build. These entries are referred to as Compile Objects (COs). Being that COs are independent of one another, they may be invoked in any order and at any time without regard to context generated by previous objects (builds). Basically, each CO determines what code will be included in the build, connects to the necessary DBs, sets its propath, executes supporting OS scripts and/or custom Progress logic, and determines where the log file will be written. As each CO executes, a detailed .log file is created that captures exactly what happened during that particular build. This .log is essential in tracking down problems that have occurred.

There is also a Global CO included within the configuration file. This global object controls which COs are included during the build process. It also allows OS scripts and/or custom Progress logic to be invoked on the build process as a whole (before and after any of the individual COs are run). There is a log file created for this object which serves as a digest version for all COs run at that time. The global log can quickly be scanned to determine if the build processes where a success or failure. The user will only need to examine the detailed log for a CO if exceptions are noted in the global log.

BUILD SETUP INSTRUCTIONS:

Setting up a build requires the user to edit the configuration file, Compile.cfg. COs are defined using two types of tags:

1. CO name tag. This tag signifies the unique name of each CO and is represented by enclosing the CO name with "[" and "]". With the exception of the global CO (which is hard coded as "[GLOBALOBJECT]"), each CO name is specified by the user and must be unique within Compile.cfg.

2. CO attribute tag. These tags are used to set attributes of the CO that has defined them and are represented by "<CO attribute name> = <value>". These tags define values such as the description of what this CO represents, the location of the source code to be compiled, a list of DBs that need to be connected at compile time, etc.

The following is an explanation of the Global CO and standard CO settings:

Global CO Settings:

[GLOBALOBJECT]	Name of global CO. This is the only hard coded CO name and cannot be changed. *This tag is required.
objectDescription	Text that briefly describes the purpose of this global CO.
compileObjectList	Comma separated list of CO names to be run during this compilation. COs are executed in the order they appear in the configuration file and not in the order they are specified in this list. A value of "ALL" may be specified signifying that all COs should be compiled as opposed to listing each CO. *This tag is required.
startProgressCode	Path to a file containing custom Progress code that will run before any COs start compiling.
endProgressCode	Path to a file containing custom Progress code that will run after any COs have ended compiling.
startOSScript	Path to an OS script file that will run before any of the COs start compiling.
endOSScript	Path to an OS script file will run after all COs have ended compiling.
logFileDirectory	Path to the directory where the all CO .log files will be written. This serves as the default log file directory and can be overridden by specifying a logFileDirectory value in each standard CO.
fileExtensionList	List of file extensions that can be compiled. Include files may be included in this list so they show up in the log file, but they will never attempt to be compiled. This serves as the default file extension list and can be overridden by specifying a fileExtensionList value in each standard CO.
compileOptions	Comma seperated list of options applied to all COs in this compilation. • NO-COMPILE – Do not compile files and/or generate r-code • NO-RECURSE – Do not recursively search through subdirectories of the directory specified with the compileDirectory CO attribute tags below • NO-PROGRESSCODE – Do not run files containing custom Progress code specified with the startProgressCode and endProgressCode CO attribute tags below • NO-OSSCRIPT – Do not run files containing OS scripting commands specified with startOSScript and endOSScript CO attribute tags below

Standard CO Settings:

[CO Name]	Name of CO. Each CO must have a unique name and be enclosed in brackets (“[” and “]”). *This tag is required.
objectDescription	Text that briefly describes the purpose of this CO.
compileDirectory	Path to the directory (and subdirectories) that contains Progress/WebSpeed source code to be compiled. *This tag is required.
saveToDirectory	Path to the directory (and subdirectories) where Progress/WebSpeed compiled source code will be saved to. Leaving this value blank defaults to saving compiled code into the compileDirectory. If this value is different from compileDirectory, then it is assumed that there are identical directory structures under compileDirectory and saveToDirectory.
databaseList	Comma separated list of DB connection parameters. DBs will connected in the order that they are specified.
propathFile	Path to a file containing Propath entries. The original Propath value that was present before any COs are run is preserved and appended to the Propath of each CO after this files’ entries are loaded.
logFileDirectory	Path to the directory where the log file will be written for this CO. The name of the log file is defaulted to <CO name>.log and cannot be changed.
startProgressCode	Path to a file containing custom Progress code that will run before this CO starts compiling.
endProgressCode	Path to a file containing custom Progress code that will run after this CO ends compiling.
startOSScript	Path to an OS script file that will run before this CO starts compiling.
endOSScript	Path to an OS script file will run after this CO ends compiling.
excludeDirectoryList	Comma separated list of directories (located under compileDirectory) that will NOT be included in this compilation. Each entry in this list may specified as a directory name, a full path to a directory, or a relative path to a directory.
excludeFilenameFile	Path to a file containing filenames that will NOT be included in this compilation. Each entry in this file may be specified as a filename, a full path to a file, or a relative path to a file.
fileExtensionList	List of file extensions that can be compiled. Include files may be included in this list so they show up in the log file, but they will never attempt to be compiled.

compileOptions	Options applied to this CO. (see global OPTIONS above for a listing of valid values.

The following file is an example illustrating some possible configuration values for Compile.cfg:

```
# Compile.cfg
#
# This file contains data that will be used to determine what
# files will be compiled. Lines beginning with a "#" are comment
# lines. "\" and "/" will be converted to either a slash or
# backslash depending on the OS to handle portability issues.
# Blank lines are allowed.

[GLOBALOBJECT]
objectDescription       = Global Compile Object
compileObjectList       = TestCompileObject1
startProgressCode       = .\support\ProgressCode.p
endProgressCode         = .\support\ProgressCode.p
startOSScript           = .\support\StartGlobalOSScript.bat
endOSScript             = .\support\EndGlobalOSScript.bat
logFileDirectory        = .\log
fileExtensionList       = i,p,w,htm,html
compileOptions          =

[TestCompileObject1]
objectDescription       = Test Compile Object #1
compileDirectory        = d:\webspeed\sports\wk
saveToDirectory         = d:\webspeed\sports\temp\wk
databaseList            = -db d:\webspeed\sports\db\sports.db -ld sports
propathFile             = .\support\SportsPropath.dat
logFileDirectory        = .\log
startProgressCode       = .\support\ProgressCode.p
endProgressCode         = .\support\ProgressCode.p
startOSScript           = .\support\StartOSScript.bat
endOSScript             = .\support\EndOSScript.bat
excludeDirectoryList    = wk\sharedimages,anotheremptydirectory
excludeFilenameFile     =
fileExtensionList       =
compileOptions          = WRITE-XREF
```