Zonnon Command Line Compiler Parameters

Build 1.0.0.36, December 14, 2004

1. Common form

ETH.Zonnon.CommandLineCompiler.exe <parameters and source files>

Parameters and source file names come in an arbitrary order and should be separated by blanks.

2. Source file names

- Complete path to the file should be specified - either in absolute or in a relative form, e.g.

  c:\testsuite\declarations\test5.znn OR
  ..\random.znn

- If a file name contains spaces then it should be represented as a string, e.g.,

  "file 20.znn"

- Several source files can be specified in the command line. The compiler compiles them sequentially composing the single output assembly.

3. External file with parameters/sources

A file can be determined as the source of parameters and file names.
The construct of the form

@filename

can be specified among other parameters and file names. 'filename' here should be a name of a text file (perhaps together with the directory path) containing additional parameters and/or source file names. Compiler treats the contents of the file as normal parameters together with the ones specified directly in the command line.

4. /out parameter

Parameter has the following form:

/out:assembly-name-without-extension

Compiler generates the output assembly with the name taken from this parameter. If parameter is not specified then the name of the (first) source file is taken (without extension). Notice that assembly-name should be specified without extension (compiler decides by its own which kind of assembly to generate - .exe or .dll - depending on /exe, or /entry parameters, see below).

5. /exe parameter

This parameter looks like as follows:

/exe

Parameter (as well as /entry parameter) specifies the behaviour of the output assembly. For the case of /exe parameter compiler generates executable assembly (.exe-file). Being invoked, the assembly initializes all modules (just runs **begin...end** parts of them), and then starts the simple command dialogue in the command window. The dialogue supports four commands:

- <module-name>.<procedure-name>
  - invokes the parameterless procedure with the specified name;
- list (or l)
  - outputs the names of all runnable (i.e., public and parameterless) procedures from the source program;
- help (or h or ?)
  - outputs the short help info;
- exit (or quit or q)
  - completes the dialogue.

REMARK 1
/exe and /entry parameters are mutually exclusive, that is, only one of them can appear in the command line.

REMARK 2
/exe or /entry parameter can appear only once in the command line.

REMARK 3
If there is no /exe or /entry parameter specified then compiler generates non-runnable assembly (.dll-file).

There are two ways of using the dll-assembly:

- Make references to assembly's public components (modules, objects etc.) from other programs (either Zonnon programs, see /ref parameter below or .NET-programs written in other languages);

- Directly invoke parameterless procedures from the dll-assembly using dialogue.exe utility from the distribution bundle. The utility organizes the command dialogue which is identical to the dialogue described above for /exe parameter.

REMARK
The utility is not provided yet.

6. /safe parameter

This parameter affects the way of handling exceptions in the executable programs generated by the compiler. Normally (if no /safe parameter is given) an exception which is not catched by a corresponding **on** clause issues the standard .NET message (which includes the current state of the execution stack). If /safe parameter is specified then no stack is printed out but just the message about the exception which has been thrown and not handled.

REMARK /safe parameter makes sense only together with /exe parameter.

7. /entry parameter

This parameter has the following common form:

```
/entry:<startup-module-name>
```

<startup-module-name> <startup-module-name> specifies the name of existing program module (in Zonnon terminology) the execution should start from. This parameter implements the conventional C/C++/C# way of executing programs. Notice however that the startup module can have any name (not necessarily "main" or "Main").

8. /ref parameter

This parameter has the following common form:

```
/ref:<full-path-to-the-assembly-file>
```

This parameter specifies the assembly which is used in the program (via import declarations). The syntax and semantics of the parameter is identical to the same parameter of the C# compiler. Notice that it doesn't matter in which language the assembly has been written. The only requirement is that the file should contain a valid .NET assembly. There can be several /ref parameters in the command line.

9. /quiet parameter

This parameter has the following form:

```
/quiet
```

If this parameter is specified then the compiler doesn't output its title, version and copyright information.