



Build Generation

Version 1.1
August 16, 2009

PathMATE Technical Notes

Pathfinder Solutions LLC
33 Commercial Drive, Suite 2
Foxboro, MA 02035 USA
www.PathfinderMDA.com
888-662-7284

Table Of Contents

1. Introduction.....	1
Context.....	1
2. Projectfile/Makefile Generation.....	1
3. Build Directory Layout	1
Defaults	1
User Selections	2
Obsolete Properties	3
4. Makefile Command Line	3
5. What's Different from Old Build File Generation?.....	3
Specifying Realized Code Locations	3

1. Introduction

This tech note outlines makefile and project file generation, and how generated code and build directories are arranged in the Q2-2005 release.

Context

The Q2-2005 release eliminates the gencpp.bat/genproject.bat approach with a move to PathMATE GUI-based transformation for both RSM and Rose based developers. This provides an opportunity to replace the genproject.bat approach and refine the build directory structure.

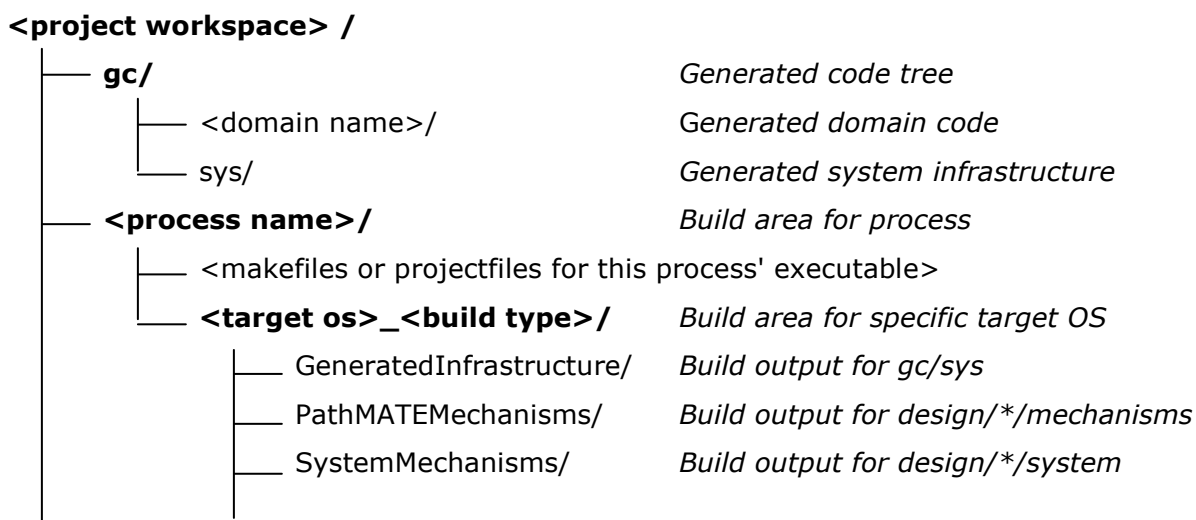
2. Projectfile/Makefile Generation

A new PathMATE platform model is provided with a transformation map for build_top.arc. This template examines the specified TargetOS and the deployed process topology, and then generates the required projectfiles or makefiles. After the Q2-2005 release, the PathMATE GUI will be able to specify TargetOS on a per-process basis, and this structure will allow build_top.arc to easily handle heterogenous deployments.

3. Build Directory Layout

Defaults

Working from the PathMATE GUI from within Eclipse, all transformations are done within a transformation project, with a project workspace directory. Default locations for all generated output and build directories are relative to the project workspace. The tree has the overall structure:



___ <domain name>/

Build output for domain

For single process systems, a single process area for MAIN will result. For multi-process systems, a process area for each process is generated.

User Selections

The following properties control the layout and population of the build area:

Marking name	PIM Element	Default Value	Effect
AdditionalIncludes	System	<blank>	Realized include file directories (list separated by ";").
AdditionalLibraries	System	<blank>	Additional included library files (list separated by ";").
AdditionalLibraryPaths	System	<blank>	Additional library search path directories (list separated by ";").
Compiler	System	gcc	For makefiles, defines compiler program name.
CompilerInstall	System	c:/cygwin	Compiler installation location; (Currently Cygwin only.)
Defines	System	<blank>	Additional compiler symbol definitions.
GeneratedPath	System	<project workspace>/gc	Specifies where the generated code is.
ImplementationLanguage	System	none	Sets up language extensions, and appropriate defaults for mechanism and system files. Use c, cpp, or java.
MechanismsPath	System	PATHMATE_MAP_<language>/mechanisms	Indicates location of SW mechanism files.
MechanismsTargetPath	System	<MechanismsPath>	Directory where makefile builds mechanisms files from in the case of a remote build tree.
RealizedPath	System,Domain	<blank>	Realized include and implementation file directories (list separated by ";"). All implementation files found in these directories are added to the project compartment for the system/domain as appropriate.
SysUmlPath	System	PATHMATE_MAP_<language>/system	Indicates location of realized system files.
SysUmlTargetPath	System	<SysUmlPath>	Directory where makefile builds system files from in the case of a remote build tree.
TargetIDE	System	VS7	When TargetOS == Win32, project files for Visual Studio are generated. This property determines what version of Visual Studio the generated files are for: "VS7", "VS8", "VS9", for targets version 7+ (.NET), and "VS6" targets version 6. For TargetOS values other than "Win32", this property is ignored.
TargetOS	System	Win32	Win32, Cygwin, Solaris, Mercury, Linux
Undefines	System	<blank>	Explicit compiler symbol

		undefinitions.
--	--	----------------

Obsolete Properties

Obsolete elements	Notes
AnalysisDir	no longer needed
BuildType	spotlight, debug, or release; set via makefile variable (from command line) or within compiler IDE; default is spotlight
PathMATEInstallDir	no longer needed
CygwinRoot	Renamed to CompilerInstall

4. Makefile Command Line

Generated makefiles (TargetOS == Cygwin, Solaris, Mercury or Linux) have standard targets to help you manage your system:

```
make -f <makefile>: Builds the default target - <system>.exe
make -f <makefile> <system>.exe : Builds <system>.exe
make -f <makefile> libs : Builds the library file for each domain
make -f <makefile> dirs : Create required output directories in build area
make -f <makefile> clean : Delete all output files (.o, .a)
```

Command line control over target type:

```
make -f <makefile> BUILDTYPE=spotlight {targets...}: Turn on -g and
Spotlight debugging (default)
make -f <makefile> BUILDTYPE=debug {targets...}: Turn on -g debugging only
(no Spotlight instrumentation)
make -f <makefile> BUILDTYPE=release {targets...}: Turn off all debug and
instrumentation
```

5. What's Different from Old Build File Generation?

If you are familiar with language-specific project file or makefile generation from PathMATE Maps version 5.02 or earlier, here's a summary of the marking changes required to use build_top:

- Specify your ImplementationLanguage and TargetOS
- Add ../ to any relative RealizedPath markings (see below)

Specifying Realized Code Locations

The RealizedPath marking for domains is used to specify an optional directory to get realized code files. This is a pathname relative to the *build directory* - where the projectfile or makefile is generated to, and where the build is done.

Historically for single-process systems the default build directory has been the <system>/project/<lang>. For multi-process systems each process has its own build directory: <system>/project/<lang>/<process>.

The new build generation capabilities have unified the default build directory location by having single process systems generate to the MAIN process area - just like a multi-process system. This change for single process systems requires all RealizedPaths that were originally specified relative to the <system>/project/<lang> directory to add a ../ to the RealizedPaths value to account for the build directory mode to <system>/project/<lang>/<process>.

For example, in the SimpleOven sample in C, the build directory was pathmate/samples/SimpleOven/rose/project/c, and the ExternalDeviceControl domain RealizedPath was:

```
Domain,SimpleOven.ExternalDeviceControl,RealizedPath,realized_c
```

Now the build directory is pathmate/samples/SimpleOven/rose/project/c/MAIN, and marking changes to:

```
Domain,SimpleOven.ExternalDeviceControl,RealizedPath,../realized_c
```