Electric Cloud ElectricAccelerator

version 7.0

Technical Notes
MS Visual Studio Solution Support Add-in

version 3.2.3

May 2013

This document contains information about the ElectricAccelerator Solution Support Add-in. Topics include:

Overview

The ElectricAccelerator® Solution Support Add-in is a command line add-in used by Electric Make® (eMake) to convert Visual Studio projects into NMAKE makefiles.

IMPORTANT: The ElectricAccelerator Solution Support Add-in is different from the ElectricAccelerator Visual Studio IDE Add-in, which allows you to build Visual Studio solutions and projects from within the Visual Studio IDE using eMake.

Support Information

The ElectricAccelerator Solution Support Add-in supports all .NET versions of Visual Studio (2002, 2003, 2005, 2008, 2010, and 2012).

New Features and Improvements

for 3.2.3

- Corrected a compile failure issue. (VSP-632)
- Removed sticky from ECADDIN_RUN_LOCAL_LINK/LIB. (VSP-630)
- If ModuleDefinitionFile contained spaces only, it would cause a build failure. This issue was fixed. (VSP-629)
- Fixed a missing LIBPATH paths issue. (VSP-628)
- Corrected a build failure that could occur if a space is present in custom build step output. (VSP-627)
- Fixed a linker error that occurs when the lib file does not exist. (VSP-626)
- User macros that contain '-' would cause the build to fail. This issue was fixed. (VSP-625)
- Fixed an issue that caused a user-defined manifest to be excluded from the resultant executable. (VSP-621)
- Fixed a C# build failure issue that was caused by a space in the solution name. (VSP-614)

for 3.2.2

- Added the environment variable `ECADDIN_NORMALIZE_PATHS`, allowing you to normalize all paths in the makefile (VSP-577).
- Fixed an issue that resulted in referenced assemblies not being copied (VSP-576).
- Corrected an issue that resulted in files being copied to an incorrect target directory (VSP-575).
- Specifying a directory as an output file for browse information now resolves correctly and no longer causes eMake to throw an error (VSP-572).
- Fixed the registration of the add-in for Visual Studio 2012 (VSP-570).

for 3.2.1

- Added support for Visual Studio 2012 (VSP-538).
- Corrected an issue where user macros were not passed to the environment (VSP-558).

Known Issues

This version of the add-in has the following known issues:

- For Visual Studio 2012, the project build order under eMake may be different to Visual Studio if project dependencies are not fully defined.

  Workaround: If a build fails because a prerequisite project has not been built, add an explicit project dependency in the solution.

- Visual Studio 2008 builds may break or may not be optimized after upgrading from an earlier version.

  Workaround: See knowledge base article KBEA-00065.

- Microsoft Visual C++ 2010 projects that contain “custom build rules” will not be parallelized at the project item level.

Prerequisites

The Microsoft Visual C++ 2005 SP1 Redistributable Package is required for all versions of Visual Studio.

Getting Started

Before you can use ElectricAccelerator to build your Visual Studio project, you must have already installed and run Visual Studio on each agent host for each user (all *ECloudInternalUsers*).

If you currently invoke Visual Studio from inside a makefile, you are ready. If you invoke Visual Studio directly from the command line or through a batch file, you must create a makefile for Electric Make to run. For example:

```
all:
        devenv /build Release foo.sln
```

-- or --

```
all:
        devenv /build Release foo.sln /project bar.vcproj
```

The makefile must invoke `devenv` with whatever options you currently use. Ensure the correct version of `devenv` is in your path:

```
devenv /?
```

and usual Visual Studio environment variables are set.

You may want to ensure the add-in is installed on the agent hosts. To check this, start Visual Studio on the agent host and go to Tools > Add-In Manager. The dialog should contain the add-in, which should be enabled for general and command-line use.

The Solution Support add-in may not be compatible with other build-related add-ins.

If running a local build, the Solution Support add-in recognizes it is a local build and does not perform any operations. Visual Studio behaves as if the add-in were not there.

If your builds fail with messages about not finding certain files (header files, DLLs, libraries, and so on):

Ensure you explicitly define all inter-project dependencies in your solution. Visual Studio uses an internal build order for projects within a solution. This “order” allows projects to be built in a certain order even without explicit declaration of project dependencies. Verify that all source files are in the eMake root definition.

If your build fails with a message that an application folder could not be created:

For example:

```
The Application Data folder could not be created. make: *** [all] Error -1
```

The build runs on agent hosts using the user account that owns the agents. This error message usually means this user does not have an initialized Visual Studio environment. Visual Studio must be invoked on each agent host by the user who owns ElectricAccelerator agents on that host. You may need to contact your ElectricAccelerator administrator.

If you are setting `--emake-root`, do not include Microsoft Visual Studio directories.

For example:

```
emake --emake-cm=<MyCM> --emake-root="C:/Program Files/<MyCom>;C:/Program Files/
Common Files/<MyCom>"
```

Important Notes

Upgrading the add-in

When you upgrade from Solution Support Add-in v3.0 to v3.0.2 or later, "debug" builds using PDB files will generate many conflicts on the first build. Subsequent builds will be fine. This occurs because the build order may have changed between v3.0 and v3.0.2.

Electric Cloud recommends regenerating history whenever you upgrade the add-in.

Setting the path for 64-bit or Xbox builds

To run 64-bit or Xbox builds, you must set the path manually.

Solving common issues

If you encounter issues, make sure you have done the following:

- Initialize Visual Studio

  Use the psexec method to initialize Visual Studio as shown:

  ```
  psexec -u ECloudInternalUser1 "C:\Program Files\Microsoft Visual Studio 8\
  Common7\IDE\devenv.exe"
  ```

  As an alternative, disable profiles for Visual Studio by running this regedit script:

  ```
  REGEDIT4
  [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\VisualStudio\8.0\Profile]
  "AppidSupportsProfiles"="0"
  ```

- Disable the Windows error reporting service on the agent/EFS hosts. This avoids popup windows for crashed applications.
- Set the maximum number of parallel project builds to 1.
- Initialize the Customer Experience Improvement Program.

If you continue to encounter issues, go to the Electric Cloud ElectricAccelerator Knowledge Base and search for "Visual Studio".

Incremental Linking

Using the add-in

Visual Studio supports incremental linking with the /INCREMENTAL linker option. This does not function in eMake because eMake updates the timestamp of the exe/dll when it copies it back to the build machine (from the agent) to prevent any problems due to clock skew.

To work around this problem, we can “touch” the exe after the link with its current timestamp. This explicit modification of the timestamp instructs eMake to preserve the timestamp, hence keeping the validity of its incremental status.

To enable this feature with the add-in, set `ECADDIN_ENABLE_INCREMENTAL_LINK=true`. This inserts a call to `ectouch.exe`, which performs the action stated above. `ectouch.exe` should be installed in the `PATH`.

Not using the add-in

If you are not using the add-in, you can still use this feature. You can rename `ectouch.exe` to `eclink.exe` and replace occurrences of `link.exe` with `eclink.exe`. `eclink.exe` should be in the `PATH`. Alternatively, you can rename `link.exe` to `link_ec.exe` and copy `eclink.exe` to `link.exe`. (If you want something other than `link_ec.exe`, set `EC_ORIGINAL_LINK_PATH` to the location of the “real” `link.exe`.)

PDB Splitting

Using the add-in

By default, Visual Studio puts all debugging information in a centralized database (PDB) called vc80.pdb (this is Visual Studio version-specific). Because each compilation modifies this file, everything in the project is serialized. A workaround is to group debug information into multiple PDB files. You can accomplish this automatically if you use the add-in.

`ECADDIN_MAX_PDB_FILES` is set to 16 by default. You can change this value to be equal to or less than the number of agents, but you may need to increase or decrease this for optimal efficiency. `ECADDIN_MAX_PDB_FILES` specifies the maximum number of PDB files produced. Each file is placed into a PDB determined by a hash of its filename. This method ensures that a particular file is always placed in the same PDB. This is necessary to ensure eMake's history file remains valid.

For example, if a project contains 4 files, `File1.cpp`, `File2.cpp`, and so on, and they are all serialized on PDB file `vc80.pdb`. Set `ECADDIN_MAX_PDB_FILES=2` will create (at most) 2 PDB files:

File1.cpp --' <*ProjectName*>_0.pdb
File2.cpp --' <*ProjectName*>_1.pdb
File3.cpp --' <*ProjectName*>_0.pdb
File4.cpp --' <*ProjectName*>_1.pdb

In this example, File1 and File3 will be serialized against each other but will build in parallel from File2 and File4 (which will be serialized against each other).

You can change this variable in the Visual Studio IDE Add-in solution or global settings. Go to the Performance section of the Add-in pane.

The history file must be deleted when adding or changing the value of `ECADDIN_MAX_PDB_FILES`. You can also set `--emake-history=create`.

Not using the add-in

This technique can be used without using the add-in. This distribution contains the application `hashstr.exe`, which hashes the filename and returns the bucket number. You can use this in your makefile to set the PDB filename (using /fd) in the same manner as above. Precompiled headers must be switched off for this to work.

Usage: `hashstr "mystring" [modulus]`

Where mystring is the string from which to generate the hash value, and modulus is the number of hash bins you want to use.

You can add this to a pattern rule for builds that suffer from performance degradation due to PDB serialization, with something similar to the following:

```
%.o: %.c
```

```
$(CC) /c $(cflags) $(PCH_USE_FLAGS) $(cvars) $(cplus_flags) $(LOCAL_INCLUDE)    $(P
CB_INCLUDE) $< /Fo$@ /Fd$(shell ${path-to-hashstr}/hashstr.exe "$@"     ${hashstr-mo
dulus}).pdb
```

Removing Project References and Dependencies

Visual Studio will always attempt to build project references and dependent projects before building the target project. This behavior causes problems for eMake because it attempts to build projects on different agents simultaneously. It can result in a particular project being built multiple times and a longer build time with eMake than with Visual Studio alone.

To resolve this, the add-in removes project references and dependencies. To keep project references and dependencies, set the following environment variable:

```
ECADDIN_REMOVE_DEPENDENCIES=false
```

You can change this variable in the Visual Studio IDE Add-in solution or global settings. Go to the Performance section of the Add-in pane.

Setting Environment Variables for Visual Studio

You can control the way the Solution Support add-in works by setting these environment variables on the Electric Make machine.

Note: Environment variables can be true or false. Valid boolean values are “0”, “no”, “false”, “off” and “1”, “yes”, “true”, “on”. Case is not significant.

Environment Variable	Description	Usage
ECADDIN_DEBUG	Setting this variable to any value causes debug log files to remain in C:\ecdebug<ID>.log on the agent host. These files are used for troubleshooting by Electric Cloud engineers. Normally, you do not need to set this value.	debugging
ECADDIN_DEBUG_LOG_FILENAME	Specifies the debug log name. Requires ECADDIN_DEBUG. Use '$1' in the file specification to insert a unique ID. For example, C:\ecdebug_$1.log. Use a file location outside of emake root. The log file is stored on the agent.	debugging
ECADDIN_DONT_RM_TMP_MAKEFILES	Retains makefiles created during the build but normally deleted when the build finishes. This environment variable can have any value; it just needs to be set.	debugging
ECADDIN_DONT_USE_UNIQUE	Does not use unique names for temporary makefiles. Use with ECADDIN_DONT_RM_TMP_MAKEFILES.	debugging
ECADDIN_ECBREAKPOINT	Determines whether to invoke ecbreakpoint on failed jobs.	debugging

Environment Variable	Description	Usage
ECADDIN_ECBREAKPOINT_PROJECTS	Determines whether to invoke ecbreakpoint for specified projects. Use a semi-colon to delimit projects.	debugging
ECADDIN_ADD_IMPLICIT_PDB_DEPENDENCIES	Adds dependencies to improve first-time build speed.	performance
ECADDIN_MAX_PDB_FILES	Specifies the maximum number of PDB files produced (set to 16 by default). See "Overview" on page 2.	performance
ECADDIN_REMOVE_DEPENDENCIES	Removes project dependencies (on by default). See "Removing Project References and Dependencies" on page 6.	performance
ECADDIN_USE_DEVENV_FOR_PROJECT	Uses devenv (instead of MSBuild) to build specific projects. Supply a list of projects (separated by a semicolon) to be built with devenv.	
ECADDIN_CONTINUE_ON_ERROR	Allows the build to continue after an error has occurred. Off by default for the Solution Support Add-in. On by default for the Visual Studio IDE Add-in.	switch
ECADDIN_CREATE_MISSING_DEPENDENCIES	Creates missing dependencies to avoid missing dependency warnings.	switch
ECADDIN_DISALLOW_BSC	Does not generate browse information files.	switch
ECADDIN_DISALLOW_PCH	Does not generate/use precompiled header files (implied by ECADDIN_MAX_PDB_FILES)	switch
ECADDIN_DISALLOW_PDB	Does not generate PDB files.	switch
ECADDIN_DISALLOW_SBR	Does not generate browse information files from sources.	switch
ECADDIN_DONT_ADD_PCH_LOCATION	Prevents the add-in from adding the location of the PCH file in all cases. This variable is relevant only if ECADDIN_MAX_PDB_FILES or ECADDIN_DISALLOW_PCH is switched on.	switch

Environment Variable	Description	Usage
ECADDIN_DONT_PARSE_PROJECT	This variable takes a list of project names separated by semi-colons and without white spaces. This variable is useful for deploying the add-in. If for any reason the add-in cannot build some of your projects, this variable allows you to work around the problem. When using this variable, you may experience an the warning MSB4098. You can ignore this warning because any project references are now converted into additional dependencies. MSBuild, however, does not provide a mechanism to turn off this warning.	switch
ECADDIN_DONT_PARSE_PROJECTS	This variable takes any non-blank value and its behavior is similar to ECADDIN_SERIALIZE. It calls devenv on each project (the add-in does not convert each project into individual compile/link steps).	switch
ECADDIN_DONT_USE	Disables the add-in. This environment variable can have any value, it just needs to be set. Also, you can disable the add-in on each host by using the Visual Studio Add-in Manager (on the Tools menu). **Note:** This is a “light-weight” uninstall program that disables one individual machine at a time.	switch
ECADDIN_DISABLE_MINIMAL_REBUILDS	Disables minimal rebuilds (off by default).	switch
ECADDIN_ENABLE_INCREMENTAL_LINK	Inserts a call to ectouch.exe. See "Incremental Linking" on page 4.	switch
ECADDIN_EXPAND_LINKER_OBJECTS	Expand linker objects to one line per object. Prevents errors when link line length is exceeded.	switch
ECADDIN_FORCE_Z7	Enables /Z7 compiler options for all C++ files.	switch
ECADDIN_INCLUDE_CMAKELISTS	Excludes any source file with the name CMakeLists.txt (false is default). Set ECADDIN_INCLUDE_CMAKELISTS=true to execute the file.	switch
ECADDIN_MSBUILD_PARAMETERS	Add extra parameters to msbuild command line.	switch
ECADDIN_NORMALIZE_PATHS	Normalizes all paths in the makefile. The default value is false.	switch
ECADDIN_RUN_DEPLOYMENT_PROJECTS_LOCALLY	Runs deployment projects locally using #pragma runlocal.	switch
ECADDIN_RUN_LOCAL_LIB	A list of projects where the librarian tool should be run locally (using #pragma runlocal).	switch

Environment Variable	Description	Usage
ECADDIN_RUN_LOCAL_LINK	A list of projects where the linker should be run locally (using #pragma runlocal).	switch
ECADDIN_RUN_LOCAL_PROJECT	Use this variable if your build uses a local resource (for example, a resource only on the Electric Make host (for example, a database). You do not need to set this variable if your project build includes web deployment; this is handled by the add-in. The value of this variable is a list of project names separated by semi-colons. Each project name must be the unique Visual Studio identifier for the project (for example, solution1/project1.vcproj). Do not add quotation marks or white spaces.	switch
ECADDIN_SERIALIZE	Causes each project to be built serially. It inserts '#pragma allserial' into each makefile. This variable is equivalent to setting ECADDIN_DONT_PARSE_PROJECTS.	switch
ECADDIN_UP_TO_DATE_CHECK	Pre-parses the projects to determine whether there is anything to build. Prevents unnecessary rebuilding of static build steps.	switch
ECADDIN_USE_DEVENV	Use devenv for all unparsed projects (the default is msbuild).	switch
ECADDIN_USE_MSBUILD	Allows you to use MSBuild internally for projects that the add-in cannot parse (on by default).	switch
ECADDIN_USE_RELATIVE_PATHS	Use relative paths in the makefile to reduce line lengths.	switch

Using the ecdevenv.exe Utility

Incremental Visual Studio builds under ElectricAccelerator can have a long parse time before the build commences. This process can take several minutes for large solutions.

You can use the ecdevenv.exe utility to bypass parsing the solution if the NMAKE makefiles have already been created. If the solution, projects, environment, or command-line have changed since the last parse, devenv is called to regenerate the makefiles. If nothing has changed, eMake is called on the previously generated makefiles.

ecdevenv.exe creates a file, <*solution*>.last_build, which contains details about the solution, projects, environment, and command-line at the time of the last build. Deleting this file forces ecdevenv to call devenv to regenerate the makefiles.

ecdevenv.exe is in the Electric Cloud bin directory (C:\ECloud\i686_win32\bin).

To use ecdevenv, simply replace existing devenv.exe / devenv.com calls with ecdevenv.

To debug ecdevenv, set ECDEVENV_DEBUG to turn debugging on and send debug to stdout, or set ECDEVENV_DEBUG_LOG to send debug to a file.

Troubleshooting and Getting Help

Contacting Technical Support:

Before you contact our technical support staff, please have the following information available.

- Your name, title, company name, phone number, fax number, and email address
- Operating system and version number
- Product name and release version
- Problem description

Hours: 8AM - 5PM PST (Monday-Friday, except Holidays)

Phone: 408-419-4300, Option #2

Email: support@electric-cloud.com