

# Build Solutions for the Microsoft Developer

## OpenMake Meister uses Microsoft Technology to extend the build functionality for the Microsoft Developer

OpenMake Meister uses Microsoft Technology to extend the build functionality for the Microsoft Developer. OpenMake Meister extends the build features of Visual Studio and Microsoft Team Build to support multi-language builds. Meister allows VS developers to build multi-language applications including older versions of Microsoft Studio and Java in the same build. In addition, using innovative Build Forensics, Meister can bridge production binaries back to their development origins, ensuring that source code managed in Team Foundation Server was used to create the production objects.



### Key Benefits

- Multi-language Build Support for building Visual Studio applications that span version of Visual Studio and call Java components
- Build best practices regardless of the development language, or Microsoft IDE version.
- Dependency Discovery and Build Forensics bridging production binaries to matching source code in Team Foundation Server.
- Consolidation of MSBuild scripts, Nant Scripts and Nmake scripts into a set of standardized and reusable build methods.
- Centralized user management, on-demand workflow management, and self service information for viewing build logs and metric reports.

### Extending Builds for the Visual Studio Suite

Microsoft Visual Studio 2005 provides streamlined functionality for executing builds outside of the IDE using MSBuild to create C#, J# and VB.Net objects. For development teams who use multiple versions of Visual Studio, they need a build process that supports more than just C#, J# and VB.Net objects. OpenMake Meister extends the build functionality to all different kinds of build types.

For a large university, OpenMake Meister improved the development efforts by weaving together the compile and link process for both Visual Studio 2005 and Visual Studio .Net objects, as if they came from the same IDE. Enhancing the build process without the need for manual scripting allowed developers to easily work within the different Microsoft IDEs and submit builds in a standard and traceable way, improving productivity and reducing development costs.

Table 1 shows what objects are supported by MSBuild and how OpenMake Meister extends the Visual Studio Build capabilities.

| Language                                      | MSBuild/Team Build      | Meister |
|---|-------------------------|---------|
| C#  | Yes                     | Yes     |
| J#  | Yes                     | Yes     |
| VB.NET  | Yes                     | Yes     |
| C .Net  | Yes, via VCBuild.exe    | Yes     |
| ASPX  | Yes, via ASPCompile.exe | Yes     |
| C – VS 6                                      | No                      | Yes     |
| C++ – VS 6                                    | No                      | Yes     |
| VB – VS 6                                     | No                      | Yes     |
| MSI Installers                                | No                      | Yes     |
| Other .Net Project file formats, Biztalk 2005 | No                      | Yes     |
| Java  | No                      | Yes     |
| Cobol   | No                      | Yes     |

Table 1: Multi-language Builds

# Build Solutions for the Microsoft Developer

## Build Decision Making when Building Multiple Solutions

In larger enterprise efforts, where it is necessary to manage large applications, multiple Visual Studio solution files are frequently used to create a single application. For this effort, developers must write their own MSBuild XML scripts. This is where OpenMake Meister enhances the process.

OpenMake Meister allows you to generate a build.xml file, called a Build Control File, that builds an application that uses multiple Visual Studio solution files with dependency linking. During the build process, Meister's deep dependency scanning links all of the dependent parts together. For a major retailer, this solution management feature allowed the development team to separate application components into unique VS solution files, creating smaller solution files, without losing the ability to build all of the solutions as a single unit. When applications become more complex, it is often not possible to manage the entire application as a single VS solution. OpenMake Meister allows for teams to use as many VS solutions as needed, and not lose critical build functionality. OpenMake Meister can accurately determine the order in which objects between solution files should be built. We call this good build "decision making."

## Build Decision Making for Build Avoidance

Building faster is critical for large projects. The most efficient way to build faster is through *build avoidance*. Build avoidance simply means that your builds will be done incrementally, re-compiling only the objects that are out of date. With MSBuild you can define "Transform" relationships between objects and MSBuild will check the time and date of the relationships to determine if an object is out of date. To do this, you must code the "Transform" statements manually in the MSBuild script to define how you want to handle the date/time checking process.

OpenMake Meister uses machine intelligence to automatically track these dependencies and perform the date/time checking. For a major financial institution, the build avoidance feature of OpenMake Meister allowed them to easily define a process for handling Emergency Build procedures. To build their full application required over 8 hours. To support an Emergency "fix" procedure, they wanted only the changes to be re-built. Meister's automatic build avoidance supported this need decreasing the build times from 8 hours to less than 30 minutes. In addition, this same process was introduced into their Agile development practices allowing continuous integration builds to occur every time a check-in of source was performed, not just on a nightly basis. With OpenMake Meister, 10 minute builds are possible, making continuous integration builds a reality.

## Support for Building a VS Web Project

A common build challenge for VS 2005 developers is the Web Project. The web project involves not only source and assemblies, but also a requirement to interact with the IIS web server. MSBuild can call the process that creates the Web Project, but developers must manually create the MSBuild scripts. In some cases, developers choose Nant, an Apache open source language, to create the build logic for VS 2005 Web Projects. Meister provides an alternative. It supports the building VS 2005 Web Projects by automatically generating their build scripts, similarly to the way VS 2005 generates MSBuild scripts for C# and J#.

For sales information and demo request, call:  
Toll Free: 800.359.8049  
312.440.9454



213 W. Institute Place, #404  
Chicago, IL 60610  
Tel: 312.440.9545  
Toll Free 800.359.8049