NPanday 1.3-incubating

v.

The NPanday Team

1

1.1 Building NPanday on 64bit-Windows

NPanday should be built using x86-toolsets of the .NET Framework - the build will succeed, but tests will fail, if built with Framework64. On 64bit operating systems, however, NPanday defaults to Framework64. The easiest way to workaround the issue, is to change the file path for all frameworks in `.m2\npanday-settings.xml`.

We expect to enable native support for building with 32-bit even on 64-bit operating systems in the next version of NPanday, though. For Details see NPANDAY-369.

2

2.1 NPanday Conventions

The following sections describe the conventions used within NPanday itself. This section is useful for developers wishing to contribute to NPanday, as well as developers looking for a baseline for their own projects. These conventions are evolving and subject to change as better ideas emerge.

- Artifact ID - specified within the pom - is equivalent to the project's module name.

  Project Structure

  ```
   |-- NPanday.Artifact
   |   `-- src
   |       `-- main
   |           `-- csharp
   |               `-- NPanday
   |                   `-- Artifact
   |                       |-- ArtifactContext.cs
   |                       `-- Artifact.cs
   `-- pom.xml
  ```

  pom.xml file

  ```
  <project xmlns="http://maven.apache.org/POM/4.0.0">
    <modelVersion>4.0.0</modelVersion>
    <groupId>npanday.artifact</groupId>
    <artifactId>NPanday.Artifact</artifactId>
    <packaging>library</packaging>
    <version>0.9</version>
    <name>NPanday.Artifact</name>
  </project>
  ```

- If the module does not contain children modules, the Group ID is the same as the artifact ID.

  ```
  <project xmlns=&quot;http://maven.apache.org/POM/4.0.0&quot;>
    <modelVersion>4.0.0</modelVersion>
    <groupId>npanday.artifact</groupId>
    <artifactId>NPanday.Artifact<artifactId>
    <packaging>library</packaging>
    <version>0.9</version>
    <name>NPanday.Artifact</name>
  </project>
  ```

- If a module contains children modules, the child module Group ID should either be equivalent to a pluralized parent module Group ID or be a deriviative of the parent module Group ID.

  ```
  parent Group ID: NPanday.Model
  child Group ID: NPanday.Model, NPanday.Models or NPanday.Model.VSContent
  ```

- The directory structure of the source directory (typically src/main/csharp) will follow the same pattern as the group ID.

```
 |-- NPanday.Artifact
 |   `-- src
 |       `-- main
 |           `-- csharp
 |               `-- NPanday
 |                   `-- Artifact
 |                       |-- ArtifactContext.cs
 |                       `-- Artifact.cs
 `-- pom.xml
```

```
<project xmlns=&quot;http://maven.apache.org/POM/4.0.0&quot;>
  <modelVersion>4.0.0</modelVersion>
  <groupId>npanday.artifact</groupId>
  <artifactId>NPanday.Artifact<artifactId>
  <packaging>library</packaging>
  <version>0.9</version>
  <name>NPanday.Artifact</name>
</project>
```

- If an assembly will only compile under a specific platform, those values should be specified within the compiler-config.

```
 <project xmlns="http://maven.apache.org/POM/4.0.0">
  <modelVersion>4.0.0</modelVersion>
  <groupId>npanday.plugins</groupId>
  <artifactId>NPanday.Plugins</artifactId>
  <packaging>pom</packaging>
  <version>0.9</version>
  <name>NPanday.Plugins</name>
  <build>
    <sourceDirectory>src/main/csharp</sourceDirectory>
    <testSourceDirectory>src/test/csharp</testSourceDirectory>
    <plugins>
      <plugin>
        <groupId>npanday.plugin</groupId>
        <artifactId>maven-compile-plugin</artifactId>
        <extensions>true</extensions>
        <configuration>
          <vendor>MONO</vendor>
          <frameworkVersion>2.0.50727</frameworkVersion>
          <vendorVersion>1.2.3.1</vendorVersion>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

- Use the default setup within the npanday-settings.xml is to configure cross-platform builds.

2.1.1 Plugin Naming Convention

For plugins going forward, NPanday will use the normal convention of *subject* `-maven-plugin`, e.g. `wix-maven-plugin`.

3

3.1 Debugging the Visual Studio Add-in

If you are looking to contribute to the code in the Add-in, you will find it helpful at times to be able to step through it in the Visual Studio debugger. The following steps show how to do so.

1. Check out the source code for NPanday if you haven't already. More information on how to build it is available in Building NPanday.
2. Open solution file: `assemblies\NPanday\NPanday.sln`. You may need to choose 'Load project normally' in a warning dialog. The solution is for Visual Studio 2005 - if you are using a more recent version it may ask you to upgrade the project as well.
3. Build the whole solution using the normal Visual Studio build solution command.
4. Open this file using a text editor (notepad will do): `My Documents\Visual Studio 2005\Addins\NPanday.VisualStudio.AddIn`. This should be the file for the version of Visual Studio you intend to debug it in, so change 2005 to 2008/2010 if desired (it need not be the same as the one you have the main solution open in).
5. Change the value of the `<Extensibility><AddIn><Assembly>` to the following file within your checkout: `assemblies\NPanday.VisualStudio.Addin\target\NPanday.VisualStudio.Addin.dll`. You must use the full path, for example: `c:\Documents and Settings[username]\checkouts\npanday\assemblies\...`
6. Back in Visual Studio, from the NPanday solution, right click on the `NPanday.VisualStudio.Addin` project. Set it to be the *Start up project* of the solution.
7. Right click on `NPanday.VisualStudio.Addin` again and select *Properties*
8. Go to the Debug tab and select *Start External Program* in the *Action* section. Locate the `devenv.exe` of the version of Visual Studio that you want to debug. This is likely to be either `C:\Program Files\Microsoft Visual Studio 10.0\Common7\IDE\devenv.exe` (Visual Studio 2010) or `C:\Program Files\Microsoft Visual Studio 9.0\Common7\IDE\devenv.exe` (Visual Studio 2008) or `C:\Program Files\Microsoft Visual Studio 8.0\Common7\IDE\devenv.exe` (Visual Studio 2005)
9. You can now set any breakpoint on the project and press the "Run" button to open a new instance of Visual Studio with the current Add-in code loaded. `Connect.cs` is the main class of the Add-in and is the place to look for code to set breakpoints in to start with.

4

4.1 NPanday and ASP.NET Projects

4.1.1 Structure of ASP projects

In Visual Studio you can build and publish a web application using right click in a ASP.NET project -> Publish

A simple Web Application in ASP .NET will generate:

```
.
|-- bin/
|   |-- *.dll -- the assemblies referenced by the application
|   |-- artifact.dll -- the project assembly
|   `-- artifact.pbd -- the program debug database
|-- Web.config
`-- *.aspx -- ASP pages
```

This can be deployed to IIS.

The normal compilation using the dotnet compiler plugin already generates proper DLLs.

The plugin created an `aspnet` type that generates the DLL in `target/artifactId/bin` and copies all aspx files to `target/artifactId`

It then zips that folder and install it in the repo (in the future create a msi package - perhaps by using the wix plugin as well).

```
<assembly xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
maven.apache.org/xsd/assembly-1.1.0-SNAPSHOT.xsd">
  <id>dist</id>
  <formats>
    <format>zip</format>
  </formats>
  <fileSets>
    <fileSet>
      <directory>${basedir}/target</directory>
      <outputDirectory>/bin</outputDirectory>
      <includes>
        <include>**/*.dll</include>
      </includes>
    </fileSet>
    <fileSet>
      <directory>${basedir}</directory>
      <outputDirectory>/</outputDirectory>
      <includes>
        <include>web.config</include>
        <include>**/*.aspx</include>
      </includes>
    </fileSet>
  </fileSets>
</assembly>
```

4.1.2 Precompilation of ASP pages

The SDK provides a ASP .NET precompiler `aspnet_compiler` that is used to check for errors in the ASP pages. See also: Precompilation In ASP.NET 2.0.

4.1.2.1 Implementation

Create an `aspx-compiler-mojo` for `aspnet_compiler` executable, and run

```
aspnet_compiler.exe -v /artifactId -p artifactId\ -u -f target\artifactId
```

4.1.3 Visual Studio Addin

The Add-in needs to recognize Web and Webservices projects in project import

4.1.3.1 Implementation

Recognize Web and Webservices projects by checking the packaging

4.1.4 References

- Using IIS with Microsoft Visual Studio 2005 and the New Web Project System
- Precompilation In ASP.NET 2.0

5

5.1 COM References

When Visual Studio compiles code that uses a COM object, it creates a wrapper DLL. For example, compiling code in `MyProject` that uses `Shell32.dll` will result in both `MyProject.dll` and `Interop.Shell32.dll` being created.

This also needs to happen if there is a transitive dependency - if `OtherProject` depends on `MyProject`, and `MyProject` uses `Shell32.dll`, then `Interop.Shell32.dll` also needs to be available when building `OtherProject`.

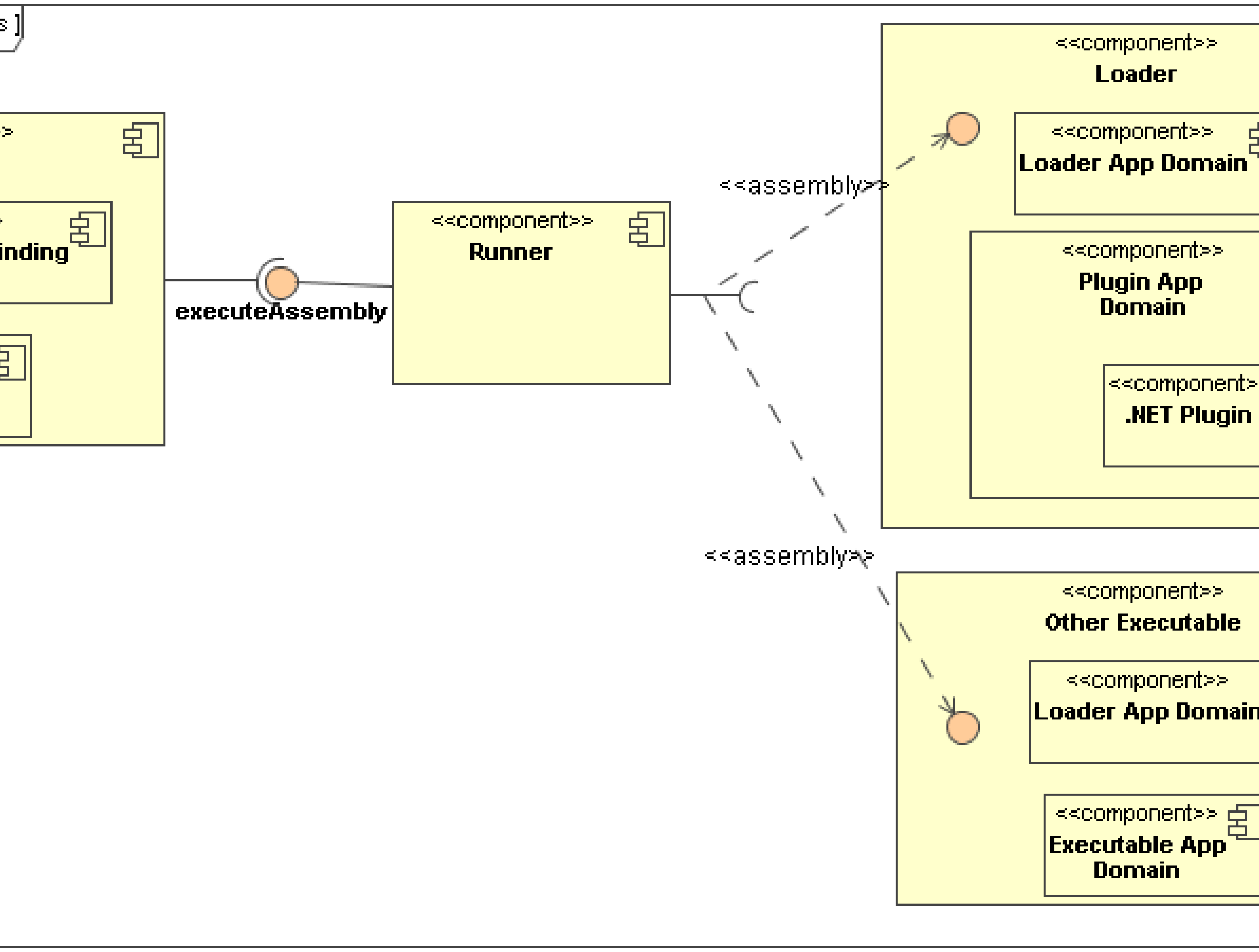
5.1.1 Requirements

- If needed, ActiveX/COM objects must be pre-installed on the build system. NPanday will not do the installation.
- Adding ActiveX/COM from VS "Add Reference" should update the POM information with a `<dependency>` with type `com_reference`.
- Generating POMs from a project containing a COM reference will result in a POM with a dependency with type `com_reference`.
- When a dependency with type `com_reference` is encountered, NPanday will generate the interop wrapper DLL and use it during compilation.
- Deleting ActiveX/COM reference should remove the `com_reference` dependency from the POM.
- Adding ActiveX/COM from "Add Maven Artifact" local or remote Maven repository is not allowed. If this attempted, a meaningful error message should be displayed, and nothing should be added to the POM.
- Building a project with ActiveX/COM references should successfully build if the referenced ActiveX is pre-installed in the system.
- Building a project with ActiveX/COM references but the actual ActiveX/COM is not in the system should prompt a meaningful error message.
- Interop wrapper DLLs may be installed/deployed to the local/remote repository and consumed as normal 'library' DLLs.

5.1.2 References

- http://msdn.microsoft.com/en-us/library/sd10k43k.aspx
- http://msdn.microsoft.com/en-us/magazine/cc301750.aspx

6



Architecture Diagram

7

7.1 NPanday Architecture

This document is currently a placeholder for documenting the design decisions in NPanday. Critical aspects are being reconsidered for future versions, and changes made will be documented here.

7.1.1 Legacy Documents

The following documents are still relevant to the current version, but need revision.

- Plugins written in .NET
- ASP.NET projects
- COM References

8

8.1 NPanday Developer's Guide

This guide is for those wishing to contribute to NPanday, and contains references for current developers.

8.1.1 Contents

- Building NPanday
- Debugging the Visual Studio Add-in
- Conventions to use
- NPanday Architecture

8.1.1.1 References for NPanday Developers

- Releasing NPanday
- Publishing the Documentation
- Issue Handling Workflow

9

9.1 NPanday Developer's Issue Workflow

- After fix is committed, developer would post a comment on the issue and changed the status of the issue to "FIXED".
- Another person would test the fix in another machine and if the issue is already fixed then that issue will be "CLOSED" by that person.

Note: Make sure that issues under the next release version will be updated (e.g if fixed, will be closed if it hasn't been updated).

10

10.1 Publishing the Documentation

The documentation should be published after a release has been tagged.

There are two separate sites to publish:

- the plugins reference
- the main documentation site

This is done with the following command within the release checkout, first from the `plugins` subdirectory, then the `site` subdirectory:

```
mvn site-deploy
```

Note: It's important to do them in this order so that the generic plugins index page will get overwritten by the correct one in the docs for this version.

11

11.1 Pre-release checklist

- Check the build is successful for both NPanday and NPanday-ITs
- Ensure all issues in the jira for that version have been resolved, or are moved to a later version
- NPanday staging repo http://vmbuild.apache.org/archiva/repository/staged-npanday/ should be re-created before the release to ensure that the repo only contains the artifacts for the said release version.
- Make sure that the copyright and licenses are on all source code

  To do this, run the following command:

  ```
  mvn clean install -Ppre-release
  ```

  Ensure that the build is successful before proceeding to the release proper.

11.2 *Note: (This is needed for the signature of the artifacts to be deployed)*

- The release manager should generate a sign key and register the key at http://pgp.mit.edu/
- Be sure that the key was already VERIFIED for integrity puposes
- The following configuration should be added in the settings.xml:

```
<profile>
  <id>npanday-releases</id>
  <properties>
    <gpg.passphrase>PASSPHRASE_CREATED</gpg.passphrase>
  </properties>
</profile>
```

11.3 Creating a Release Candidate

Send a proposal in the discussion forum regarding the release and the issues that would be fix in the said release.

The release will be done using the command line.

11.3.1 Steps in releasing NPanday:

1. Check out the code from https://svn.apache.org/repos/asf/incubator/npanday/trunk//
2. Run the following command:
   `mvn release:prepare`
3. Fill in the following values:
   - the release version should be `1.3.1-incubating` or similar (use the release version)
   - the SCM tag should be `npanday-1.3.1-incubating` or similar (use the release version)
   - check the values for the *Release version* and *Next development version* are correct
4. Check if the tag was successfully created in https://svn.apache.org/repos/asf/incubator/npanday/tags/
5. After successful release:prepare, run the following command:
   `mvn release:perform -Pnpanday-release -Dtag=url_tag`

11.3.2 Steps in releasing NPanday-ITs

1. Check out the code from https://svn.apache.org/repos/asf/incubator/npanday/npanday-its/trunk/
2. Run the following command:
   `mvn release:prepare`
3. Fill in the following values:
   - the release version should ve `1.3.1-incubating` or similar (use the release version)
   - the SCM tag should be `npanday-its-1.3.1-incubating` or similar (use the release version)
   - check the values for the *Release version* and *Next development version* are correct
4. Check if the tag was successfully created in https://svn.apache.org/repos/asf/incubator/npanday/npanday-its/tags/
5. After successful release:prepare, run the following command:
   `mvn release:perform -Pnpanday-release -Dtag=url_tag`

   To verify if the release was successful, check that the artifacts has been populated in http://vmbuild.apache.org/archiva/repository/staged-npanday/

11.4 Testing the Release Candidate

Announce the RC and the community will be given a 72hr window to test. If the RC passes, same process would be done for the final version. eg. `1.3.1-incubating`.

11.5 Voting on the Release

Post the Majority of Vote on the results of the release.

11.6 Finalising the Release

After the approval of the developers:

- Publish the site for the released version
- Binaries and sources should be copied to `//www/www.apache.org/dist/incubator/npanday/`.
- Artifacts should be merged and synched to ibiblio by executing the ff command:
  ```
  mvn stage:copy -Dsource="http://vmbuild.apache.org/archiva/repository/
  staged-npanday/" -Dtarget="scp://[APACHE_USERNAME]@people.apache.org/
  www/people.apache.org/repo/m2-ibiblio-rsync-repository"
  -Dversion=1.3.1-incubating -DrepositoryId=apache.releases
  ```
- Send out an announcement of the release to:
  - npanday-users@incubator.apache.org
  - announce@apache.org

Note: If a serious flaw is found in the release, the release version will not be removed in the distribution list but instead a new release (1.3.2-incubating) should be provided.

12

12.1 NPanday Frequently Asked Questions

12.1.1 Why is NPanday named as such?

Since NPanday is a project that builds .NET Applications, we brainstormed for a name that would symbolize a great builder, a builder that would have more freedom from its predecessor.

`Panday` is a bisaya word for carpenter/builder and at the same time `Panday` is a fictional Filipino comic hero that would fight monsters using a dagger which magically turns into a sword when raised into the sky. The materials of the dagger came from a meteorite that struck down on earth during the reign of the monsters and supernatural beings. `Panday` fights to bring back freedom and peace to the people once more.

12.1.2 What are the requirements needed to run NPanday?

You would need to install the following:

- Java 1.5 or higher
- Apache Maven 2.0.9 or higher

12.1.3 Do you need to have intensive knowledge of Apache Maven or Java in order to run NPanday?

No. NPanday's Visual Studio Add-in can ensure the user will have minimal interaction with Apache Maven if desired, without any need to manipulate the POM.

12.1.4 Can NPanday build projects outside of Visual Studio?

Yes. Since the NPanday Visual Studio Addin creates a POM file from your corresponding .NET Project you can build your .NET Projects using standard Apache Maven commands.

12.1.5 Why use NPanday when you can build .NET Applications in Visual Studio or MSBuild?

By using NPanday, you can take advantage of existing development infrastructure that is compatible with Maven. This is particularly beneficial to organizations that have both Java and .NET development teams that want to share a common infrastructure stack.

In addition, NPanday brings Maven's dependency management and other plugins (such as developer site generation) to .NET projects with little additional work needed.

12.1.6 How can a custom `settings.xml` be used for the Visual Studio Addin?

Add the `-DsettingsFile=[path_to_custom_settings.xml_file]` parameter when executing `mvn npanday.plugin:maven-vsinstaller-plugin:install`. For example,

```
mvn npanday.plugin:maven-vsinstaller-plugin:install -DsettingsFile="C:
\common\settings.xml"
```

12.1.7 How do I set the root namespace for a Visual Basic assembly?

Add the `<rootNameSpace>` element under `<configuration>` inside the `maven-compile-plugin` plugin. Just like the following:

```
<plugin>
  <groupId>npanday.plugin</groupId>
  <artifactId>maven-compile-plugin</artifactId>
  <extensions>true</extensions>
  <configuration>
    <language>VB</language>
    <rootNameSpace>my.company</rootNameSpace>
  </configuration>
</plugin>
```

13

13.1 Features

The following features available in NPanday.

13.1.1 NPanday Visual Studio Add-in Features

- *Maven Builds*: Execute Maven directly from Visual Studio or run standalone from a build server.
- *Dependency Management*: Add references to the project by automatically obtaining them from a shared Maven repository.
- *Project Importer*: Convenient way of converting an existing .NET Applications so that it can be built using Maven outside VS, ready for Continuous Integration.
- *Auto-Sync*: Synchronization while developing the .NET Application for minimal interaction with the POM (Project Object Model) file.

13.1.2 NPanday Maven Integration Features

NPanday's Maven integration features support console, window and web based applications as listed below:

13.1.2.1 C# & VB Project supported project types

- Windows Application
- Class Library
- Console Application
- Device Application
- Crystal Reports Application
- ASP .NET Web Application
- ASP .Net Web Service Application
- Windows Presentation Foundation (WPF)
- Windows Communication Foundation (WCF)

13.1.2.2 Web Site supported project types

- ASP.Net Web Site
- ASP.NET Web Service

13.1.3 NET Frameworks Supported

- .NET 2.0 / Visual Studio 2005
- .NET 3.5 / Visual Studio 2008

Note: There is partial support for Mono, .NET 1.1 and other vendors, but these are currently untested.

13.1.4 Supported Directory Structures

NPanday supported project structures are as follows:

- Flat Single Module Project
- Normal Single Project

- Flat Multi Module Project
- Normal Multi Module Project

14

14.1 Setting up an Artifact Repository

Setting up an artifact repository is highly recommended for working with NPanday and Maven projects in general.

In this example, we configure Apache Archiva, however other repository managers will operate similarly. Instructions on installing Archiva can be found on the Archiva web site.

14.1.1 Creating an NPanday repository

Logged in as the administrator created when installed, go to the *Repositories* tab and add a new managed repository. Use the following values:

- *Identifier*: `npanday`
- *Name*: NPanday Repository
- *Location*: the path where the NPanday release repository was unpacked on your machine

After adding this, you may be prompted due to the content already existing - you can safely request to continue.

Next, you will need to go to the *Users* tab and select the `guest` user, and follow the screens to allocate them with *Repository Observer* permission for the `npanday` repository to allow it to be accessed without a username and password.

This will make all of the NPanday artifacts available at http://localhost:8080/archiva/repository/npanday/.

14.1.2 Configuring Maven to use the NPanday repository

You will next need to configure Maven to use this remote repository. This can be done using the instructions shown on the Archiva web site, Configuring Maven 2 to use an Archiva repository.

14.1.3 Proxying Future Releases

An alternative to downloading the releases from the NPanday site is to automatically obtain them from the NPanday repository on demand using Archiva's repository proxy feature.

To do so, first go to the *Repositories* tab and add a new *remote* repository (below the *managed* repositories at the top). The fields used will be:

- *Identifier*: `npanday.remote`
- *Name*: NPanday Remote Repository
- *URL*: http://repo.npanday.org/archiva/repository/npanday-group

After this is added, you can connect the previous `npanday` repository with the `npanday.remote` repository using a proxy connector. You can find more information on the Archiva web site, under Understanding Proxy Connector Configuration of Apache Archiva.

14.1.4 Separate NPanday Release Repository

The above techniques provide repositories that contain not only the NPanday releases, but also all of the artifacts it depends on.

Alternatively, you can configure NPanday releases into a single managed repository, and other dependencies in separate repositories.

For the NPanday releases, they can be obtained as above either from the `npanday` subdirectory of the downloaded repository, or from the remote URL: http://repo.npanday.org/archiva/repository/npanday-releases.

For the dependencies, there are two needs:

- the Open RDF repository, which can be proxied from http://repository.aduna-software.org/maven2. Only the `info/aduna/**` and `org/openrdf/**` artifacts need to be proxied.
- 3rd party .NET dependencies. Some can be obtained from http://repo.npanday.org/archiva/repository/3rdparty/, or they can be uploaded manually to your own managed repository. Currently, only NUnit is included there.

15

15.1 Using Continuum to release .NET projects

This guide shows how to use Continuum to release NPanday based projects. It is based on the same commands used for Maven. For more information on that topic, see Releasing .NET Projects with Maven.

15.1.1 Pre-release build definition

As described in the general releasing guide, NPanday needs to prepare COM and GAC references (if your project has them) before a release. To do that, you can add a special build definition for the project module(s) in question.

Click the *Add* button under the *Build Definitions* tab of the project, and enter the following values:

Field	**Value**
POM filename	`pom.xml`
Goals	`npanday.plugin:NPanday.Plugin.SysRef.JavaBinding:`
Arguments	`--batch-mode --non-recursive`
Schedule, Build Environment, Type	*default values*
Description	*optional*

First, make sure that the project has had a successful build with its default build definition (using `clean install`).

Next, execute the new build definition for `npanday.plugin:NPanday.Plugin.SysRef.JavaBinding:prepare goal` by clicking the build icon found to the right of the goal.

Once this completes successfully, you can proceed with a release as usual.

For instructions on releasing projects using Apache Continuum, you can refer to Apache Continuum release documentation.

16

16.1 Using NPanday with your Maven Development Infrastructure

This guide shows how to configure development infrastructure to work with NPanday.

16.1.1 Contents

17

17.1 NPanday User's Guide

This guide is broken into the following sections:

- Installation - How to download and install an NPanday release
- Visual Studio Add-in User's Guide - for Visual Studio users
- Maven Command Line Users's Guide - for Maven users or Visual Studio users wishing to run builds outside of Visual Studio, or customize the generated POM
- Using NPanday with your Maven Development Infrastructure
- Reference
    - Maven Plugin Reference - reference for Maven users constructing POMs by hand
    - Troubleshooting - for when it all goes wrong
    - Supported Directory Structures

If you are looking for a particular reference, you can skip to those sections now. If you are new to NPanday or Maven, continue reading the introduction.

18

18.1 Uninstalling NPanday

To remove the NPanday .NET Build Tool, follow these steps:

1 Run the uninstaller from the Control Panel's Add/Remove programs dialog if you used the MSI to install NPanday.

2 Clear the cache by typing the following in a command shell:

`%ProgramFiles%\Microsoft Visual Studio 8\SDK\v2.0\Bin\mscorcfg.msc`

other possible locations for mscorcfg.msc

`%windir%\Microsoft.NET\Framework\v1.1.4322\mscorcfg.msc`

`%windir%\ServicePackFiles\i386\mscorcfg.msc`

`%ProgramFiles%\Microsoft SDKs\Windows\v6.0\Bin\mscorcfg.msc`

3 Go to My Computer > AssemblyCache, click View List. Delete the following two files:

- `Npanday.Model.Pom`
- `NPanday.Plugin`

4 Delete the following directories:

- `C:\Documents and Settings\[user_home]\.m2\pab`
- `C:\Documents and Settings\[user_home]\.m2\uac`

5 Remove the following file `C:\Documents and Settings\[user_home]\.m2\npanday-settings.xml`

6 Remove the following file `C:\Documents and Settings\[user_home]\.m2\repository\npanday.artifacts.resolved`

An alternative to step 3 is to run the following commands from the command prompt as an administrator:

```
gacutil /u NPanday.Model.Pom
gacutil /u NPanday.Plugin
```

18.2 Uninstalling without the Installer

If you didn't use the installer package to install NPanday, you should still perform step 2 onwards above, and then the following additional steps:

1 Locate and delete your previous version of the NPanday .NET Build Tool for Visual Studio. It is typically found at the following location, where [user_home] is your home directory:

`C:\Documents and Settings\[user_home]\My Documents\Visual Studio 2005\Addins\NPanday.VisualStudio.AddIn`

2 Remove the `C:\Documents and Settings\[user_home]\.m2\repository\npanday` directory

19

19.1 Project Dependencies

This section defines the basic configuration of adding dependencies (otherwise known as *references*) into the project. To avoid runtime errors, make sure that the library to be loaded does not have conflicting instance(s) and/or is not yet loaded in the .NET Framework.

Dependency Types:

- normal - Local repository content artifacts that are added as dependencies.
- `gac_msil` - Artifacts that are installed in the GAC and used as references. Configuration similar to the following is used, `<type>gac_msil</type>`
- `system` - Artifact references to standard framework namespaces. Configuration similar to the following is added, `<scope>system</scope>` and the `<systemPath>[path_to_library]</systemPath>`
- `com_references` - Artifact references to Component Object Model that is installed in the system. Configuration similar to the following is used, `<type>com_reference</type>`

19.1.1 Adding Dependencies

Adding dependencies follows the normal Maven pattern, using the NPanday packaging types. For example:

```
<dependency>
  <groupId>NUnit</groupId>
  <artifactId>NUnit.Framework</artifactId>
  <version>2.2.8.0</version>
  <type>library</type>
</dependency>
```

These dependencies can also refer to other modules in a multi-module build, which will be installed in the local Maven repository before building the project that depends on them.

As mentioned above, the special type of `gac_msil` is available to refer to dependencies in the GAC.

For all other dependencies, they should be located in a remote artifact repository. NPanday offers a few dependencies in the repository http://repo.npanday.org/archiva/repository/3rdparty/. More commonly however, you will establish and populate your own artifact repository as described in Setting up an Artifact Repository.

19.1.1.1 Installing an Artifact Locally

If you are testing a new library and do not yet have it available in a remote artifact repository, you may wish to install it locally.

Use the following command:

```
mvn npanday.plugin:maven-install-plugin:install-file -Dfile=[path_to_file] \
  -DgroupId=[group_id] -DartifactId=[artifact_id] -DartifactVersion=[version] \
  -Dpackaging=[packaging_type]
```

and supply appropriate values:

- **file**: the location of the file to be installed.
- **groupId**: the groupId of the file.

- **artifactId**: artifact name of the file.
- **artifactVersion**: the version of file.
- **packaging**: the type of file to be installed.

19.1.2 Adding System Scope Dependencies

To add a system scope dependency, add a configuration similar to the following in the project's `pom.xml`:

```
<dependency>
  <groupId>artifact_group_id</groupId>
  <artifactId>my.lib</artifactId>
  <version>2.2</version>
  <type>library</type>
  <scope>system</scope>
  <systemPath>C:\path\to\your\library.dll</systemPath>
</dependency>
```

Supply the proper values for the dependency's `<groupId>`, `<artifactId>`, `<version>`, `<type>`, `<scope>`, and `<systemPath>`. Exact match of `artifactId` and `version` are required when manually adding dependencies into the project's `pom.xml`.

Note that in general system dependencies are not recommended as they may be located elsewhere on a different machine. If they are required, you may mitigate this problem by using a property value that can be set differently on another system:

```
<dependencies>
  [...]
  <dependency>
    <groupId>artifact_group_id</groupId>
    <artifactId>my.lib</artifactId>
    <version>2.2</version>
    <type>library</type>
    <scope>system</scope>
    <systemPath>${some.property}\library.dll</systemPath>
  </dependency>
  [...]
</dependencies>
[...]
<properties>
  <some.property>C:\path\to\your</some.property>
</properties>
```

The other system can then override it with `mvn -Dsome.property=c:\another\path ...` or by adding it to Maven's `settings.xml` file.

20

20.1 Creating a Simple Project

Before you start this procedure, you must have all Pre-requisites in place and have successfully completed all steps in the previous section entitled Installing and Verifying NPanday .NET Build Tool.

We will use the Maven Archetype Plugin to generate a skeleton project. It will be created as a subdirectory of the current working directory in the command shell.

Execute the following command to create a C# project:

```
mvn archetype:generate -DarchetypeArtifactId=maven-archetype-dotnet-
simple -DarchetypeGroupId=npanday \
  -DarchetypeVersion=[version]>>>
```

When prompted, select the co-ordinates for the project that you wish to use, eg:

- group ID = `com.example`
- artifact ID = `Example.Project`
- version = `1.0-SNAPSHOT`

The command creates the project in the Example.Project directory, which will now contain the following:

```
.
|-- src
|  `-- main
|    `-- csharp
|      `-- Sample
|        `-- MyApp.cs
|    `-- resources
|  `-- test
|    `-- csharp
`- pom.xml
```

From inside the Solution directory (`Example.Project`), execute the following command to build and install the Example.Project DLL files into your repository:

```
mvn install
```

The `mvn install` command typically puts artifacts (installs them) into the repository here, `C:\Documents and Settings\[user_home]\.m2\repository`.

NPanday installs the artifacts in a .NET local repository, located in `C:\Documents and Settings\[user_home]\.m2\uac`.

For the `Example.Project` solution the artifact is placed here, `C:\Documents and Settings\[user_home]\.m2\uac\gac_msil\Example.Project\1.0-SNAPSHOT__com.example\Example.Project.dll`

The contents in `C:\Documents and Settings\[user_home]\.m2\uac\gac_msil` SHOULD NOT be manually modified or the project build will fail due to artifacts that are not properly indexed and are not synchronized with the repository. In case the contents have been modified, delete the `C:\Documents and Settings\[user_home]\.m2\uac` directory then re-install the project.

The sample project contains NUnit tests. If the build fails on `nunit-console`, make sure you have NUnit 2.2+ installed and located in the PATH.

You can clean up the target directory and download dependencies by executing:

```
mvn clean
```

20.1.1 Creating other types of projects

Other types of template projects are available. To use them instead, change the `archetypeArtifactId` option to one of the following:

- `maven-artchetype-dotnet-simple` - simple C# project with unit tests in the Maven directory layout
- `maven-artchetype-vb-simple` - simple Visual Basic project with unit tests in the Maven directory layout
- `maven-artchetype-netexecutable` - simple project for creating a console executable

21

21.1 Maven Command Line Users's Guide

This guide is for those that wish to run builds from the command line using Maven, and to author or alter POMs themselves without the help of the Visual Studio Add-in. It assumes that you already have NPanday successfully installed as shown in the Installation instructions.

21.1.1 Contents

22

22.1 Integration Tests

Integration tests are run on projects that contains the plugin `maven-test-plugin` under the POM build configuration.

Since integration test itself is for testing, the `<testSourceDirectory>` is no longer needed and the `<sourceDirectory>` is enough to be able to run the project test successfully. This matches the Maven convention of placing integration tests in a separate module.

The following POM snippet is an example build configuration of an integration test project.

```
<build>
  <sourceDirectory>./</sourceDirectory>
  <plugins>
    [...]
    <plugin>
    <groupId>npanday.plugin</groupId>
    <artifactId>maven-test-plugin</artifactId>
    <extensions>true</extensions>
    <configuration>
      <integrationTest>true</integrationTest>
    </configuration>
    </plugin>
  </plugins>
</build>
```

To run the test:

```
mvn test
```

Like regular unit tests, the plugin requires that NUnit be installed and on the PATH to execute the tests.

23

23.1 Releasing .NET Projects with Maven

Releasing a project with NPanday follows the same technique as for any Maven project, so the Maven documentation can be consulted on the topic. Here we will provide a brief summary, and give notes on NPanday specific issues. You can refer to the release documentation in Better Builds with Maven for more information.

23.1.1 Setting up the POM

Before preparing the release, the Maven POM must be set up appropriate for release. The version of the project should be a SNAPSHOT, but all SNAPSHOT dependency, parent and plugin references outside the project solution should be set to fixed releases.

The `<scm>` tag should be configured in the project's `pom.xml`. It should look similar to the following:

```
<scm>
  <connection>scm:svn:[project_url]</connection>
  <developerConnection>scm:svn:[project_url]</developerConnection>
  <url>[URL of the project]</url>
</scm>
```

23.1.2 Preparing References

Before starting the release, projects with GAC or COM references should run an additional preparation step:

```
mvn npanday.plugin:NPanday.Plugin.SysRef.JavaBinding:prepare
```

The reference (DLL) will be searched from `C:\WINDOWS\assembly\GAC_MSIL` directory and will be put in `C:\WINDOWS\Temp\NPanday` then renamed following the artifact filename format `[artifactId]-[version].[packaging]`.

After the reference is renamed, it will then be installed in the local repository `C:\Documents and Settings\[user_home]\.m2\repository` for it to be used when releasing the project.

The following are examples of references of different types which are renamed then installed in the local repository,

- **gac_msil** file: `Microsoft.JScript-8.0.0.0-b03f5f7f11d50a3a.gac_msil`
- **com_reference** file: `Acrobat-1.1.0.0-E64169B3-3592-47D2-816E-602C5C13F328-1.1-0.com_reference`

23.1.3 Preparing and Performing the Release

Following this, the Maven release process can be used. First, it needs to be prepared:

```
mvn release:prepare
```

Following the prompts will adjust the versions and manipulate SCM to end up with a tagged release, and trunk ready for more development.

After that (or at a point in the future), the release can be built and published using the perform command:

```
mvn release:perform
```

This will also deploy the built artifacts to an artifact repository. See Setting up an Artifact Repository for information on how to do this and to configure the POM's `distributionManagement` section accordingly.

24

24.1 Supported Directory Structures

The following directory structures are supported by NPanday's Visual Studio integration. Note that using the bare Maven plugins you can support any directory structure, but it may require manual configuration of various plugin fields.

24.1.1 Multi-module Projects

Multi-module project with a parent POM containing one or more modules, then a subdirectory for each module, which equates to a VS "project". The `.sln` file sits beside the parent POM, and each subdirectory contains a `.csproj`, `pom.xml` and source code. Source code is not typically put in a subdirectory under the module, but it might be. NUnit test code may be within each module in a directory named "Tests", or it may be in a separate module.

```
MySolution/
|-- MyClassLibrary/
|   |-- Properties/
|   |   `-- AssemblyInfo.cs
|   |-- Class1.cs
|   |-- MyClassLibrary.csproj
|   `-- pom.xml
|-- MyClassLibraryTest/
|   |-- Properties/
|   |   `-- AssemblyInfo.cs
|   |-- MyAppTest.cs
|   |-- MyClassLibraryTest.csproj
|   `-- pom.xml
|-- MyConsoleApplication/
|   |-- Properties/
|   |   `-- AssemblyInfo.cs
|   |-- MyConsoleApplication.csproj
|   |-- Program.cs
|   `-- pom.xml
|-- MySolution.sln
`-- pom.xml
```

24.1.2 Flat Single Module Project

Flat structure with `pom.xml`, `.sln`, `.csproj` and source code all in the same directory. Source code is not typically put in a subdirectory under the module, but it might be. If present, NUnit test code should be in a directory named "Tests", which is not packaged in the main artifact. See note below about "nested" projects. The "flat" structure is only supported as a single project with no sub-modules, for projects of type `library` (DLL) and `exe`.

```
FlatSingleModule/
|-- Properties/
|   `-- AssemblyInfo.cs
|-- Class1.cs
|-- FlatSingleModule.csproj
|-- FlatSingleModule.sln
`-- pom.xml
```

24.1.3 Visual Studio Web Site Project

Visual Studio "Web Site" (File -> New -> Web Site) project with `.sln` file copied into the project directory and modified to normalize paths. NPanday will build a zip file containing the aspx files and compiled dlls (in a bin/ directory). Note: There is an issue with building this type of project twice in a row: you must 'clean' before building again.

```
MyWebSite
|-- App_Data/
|-- Default.aspx
|-- Default.aspx.cs
|-- MyWebSite.sln
`-- pom.xml
```

24.1.4 Visual Studio ASP.NET Web Application

Visual Studio "ASP.NET Web Application" (File -> New -> Project, "ASP.NET Web Application"). Structure is similar to a multi-module project with a parent POM and `.sln` file at the top, plus a subdirectory for each project/module. NPanday should build a .zip file as in the previous web site example containing pages and dlls.

```
MyProject/
|-- MyWebApplication/
|   |-- Properties/
|   |   `-- AssemblyInfo.cs
|   |-- Default.aspx
|   |-- Default.aspx.cs
|   |-- Default.aspx.designer.cs
|   |-- MyWebApplication.csproj
|   |-- Web.config
|   `-- pom.xml
|-- MyProject.sln
`-- pom.xml
```

24.2 Notes

1. Some versions of NPanday have limited support for a "nested" project-within-project structure with source code in the parent directory. This structure will have a `.sln` *and* `.vbproj` file at the top, then directories for additional modules beneath, each containing a `.vbproj` file. This structure is **not recommended** and not likely to be fully supported by Maven tools such as the Release plugin
2. In the examples, `.vbproj` and `.csproj` are interchangeable, each structure should work for any language, and a solution may be composed of different modules using different languages. In addition, tests within a module may be written in a different language than the main artifact
3. The Visual Studio integration uses the project directory as the main directory for source code. However, NPanday archetypes will generate projects using the more Maven-like convention of `src/main/csharp`, etc.

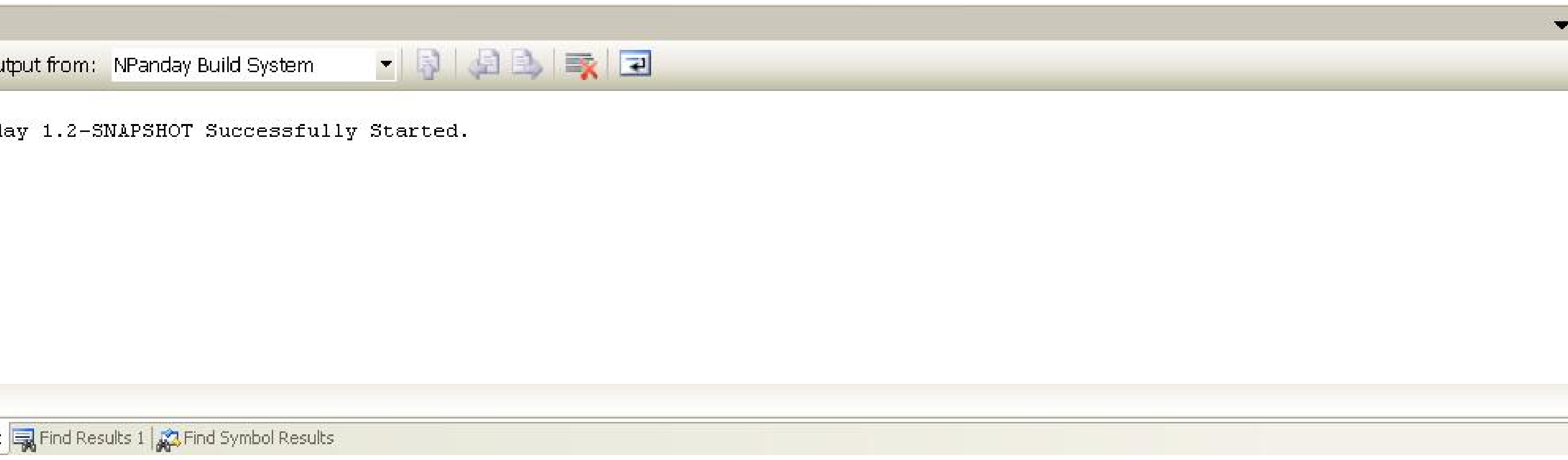
25

25.1 Troubleshooting

If you encounter a problem with NPanday, try some of the following to resolve the issue before contacting the user forums.

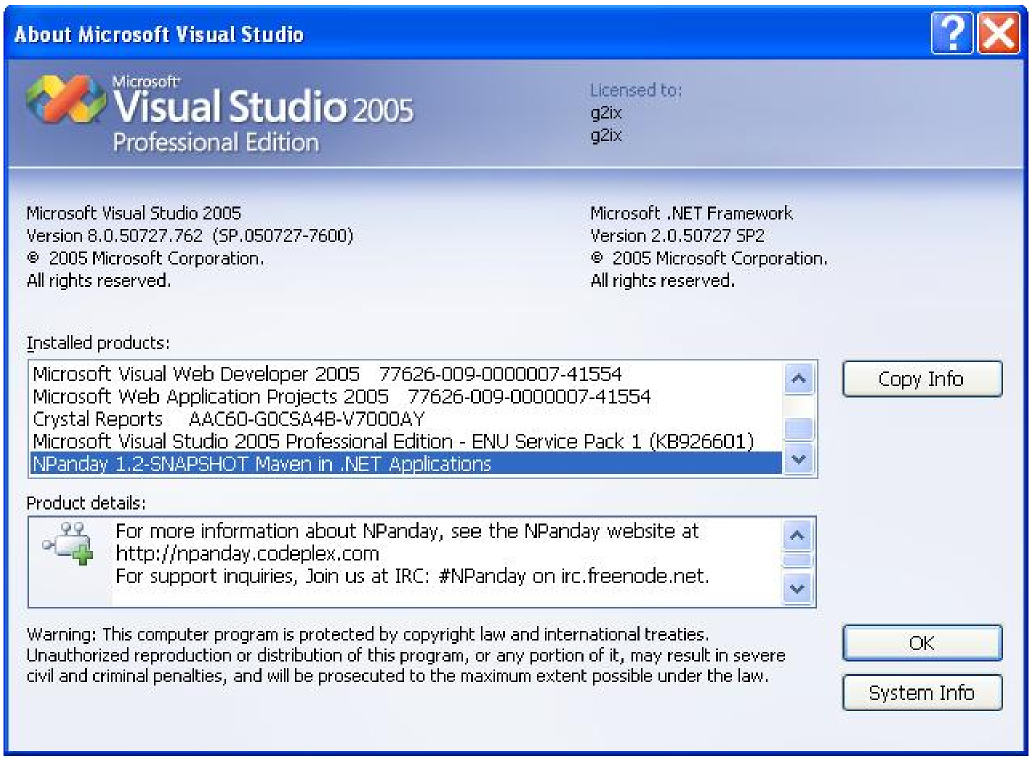
25.1.1 Check what version of NPanday you are currently using

The version can be viewed from the output console once you start NPanday



```
lay 1.2-SNAPSHOT Successfully Started.
```

Another way to check for the version would be to go to **Visual Studio Help** > **About** menu option


About Microsoft Visual Studio
Microsoft Visual Studio 2005 Professional Edition
Copy Info
OK
System Info

Licensed to:
g2ix
g2ix

Microsoft Visual Studio 2005
Version 8.0.50727.762 (SP.050727-7600)
© 2005 Microsoft Corporation.
All rights reserved.

Microsoft .NET Framework
Version 2.0.50727 SP2
© 2005 Microsoft Corporation.
All rights reserved.

Installed products:
Microsoft Visual Web Developer 2005 77626-009-0000007-41554
Microsoft Web Application Projects 2005 77626-009-0000007-41554
Crystal Reports AAC60-G0CSA4B-V7000AY
Microsoft Visual Studio 2005 Professional Edition - ENU Service Pack 1 (KB926601)
NPanday 1.2-SNAPSHOT Maven in .NET Applications

Product details:
For more information about NPanday, see the NPanday website at http://npanday.codeplex.com
For support inquiries, Join us at IRC: #NPanday on irc.freenode.net.

Warning: This computer program is protected by copyright law and international treaties. Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under the law.

25.1.2 Verifying Visual Studio 2005/2008 Service Pack 1 is installed

To make sure you have the Service Pack 1, go to **Visual Studio Help** > **About** menu option. From the About screen, verify that the Service Pack version is enclosed in parenthesis after the Visual Studio Version.

Another way to verify the Service Pack installation is to click **Copy Info** from the **About** screen and paste the information in a text editor. Verify that the SP installation information is in the list.

For Visual Studio 2005

```
Microsoft Visual Studio 2005 Professional
Edition - ENU Service Pack 1 (KB926601)
```

For Visual Studio 2008

```
Microsoft Visual Studio 2008
Version 9.0.30729.1 SP
Microsoft .NET Framework
Version 3.5 SP1
```

25.1.3 Verifying .NET SDK installation

The SDK usually installs automatically when you install Visual Studio, but you might have to download it separately and put it into the right place.

25.1.3.1 .NET 2.0 SDK (Visual Studio 2005)

To make sure you have the SDK, navigate to the following directory on your system: either in `C:\Program Files\Microsoft Visual Studio 8\SDK\v2.0` or `C:\Program Files \Microsoft.NET\SDK\v2.0`

Make sure the \v2.0 directory shown above contains all the pieces of the SDK (for example, the \Bin, \include, \CompactFramework, and \Lib directories, among others).

Note: These are the default directories and it may be installed elsewhere on your system.

If the directory structure/folder shown above does not exist or is empty, download the SDK from the following location: http://msdn2.microsoft.com/enus/downloads/default.aspx

Note: It is not necessary to move the SDK after you download it (for example, to the location where Visual Studio puts it). Just make sure your PATH environment variable points to its location, as described in the PATH environment variable prerequisite information, below.

25.1.3.2 .NET 3.5 SDK (Visual Studio 2008)

Verify that your system includes the following .NET Install Root directory: `C:\WINDOWS \Microsoft.NET\Framework\v3.5`

Note: This directory should be in place by default from your Microsoft installation procedures. If it is not, refer to the appropriate Microsoft documentation for help.

25.1.3.3 PATH

The PATH environment variable must be set such that it can find the following:

- The csc executable directory, the default path for which is: `C:\WINDOWS\Microsoft.NET \Framework\v3.5` (or equivalent for .NET 2.0)
- The devenv executable directory, the default path is typically: `C:\Program Files \Microsoft Visual Studio 9.0\Common7\IDE` (or equivalent for Visual Studio 2005)

Typically, the the `PATH` environment variable will look similar to this:

```
PATH=%PATH%;C:\Program Files\Microsoft Visual Studio 9.0\Common7\IDE;C:\WINDOWS\Mic
C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727;C:\Program Files\Microsoft Visual Stu
C:\Program Files\Microsoft Visual Studio 8\SDK\v2.0\bin
```

Note: To successfully install the NPanday add-in on machines with only Visual Studio 2008 installed, add this in the PATH: `C:\Program Files\Microsoft SDKs\Windows\v6.0A\bin`.

25.1.4 Clearing the Visual Studio Add-in Cache

If you want to clear Visual Studio's cache of any previous versions of the tool before you start Visual Studio, run the following command:

```
devenv /ResetAddin NPanday.VisualStudio.Addin
```

25.1.5 Warnings when opening solutions generated by NPanday

Upon opening your Project/Solution, you may see a window with the selection *Load project normally*. This occurs when you generate a Visual Studio solution from a Maven project (as opposed to importing the existing Solution in Visual Studio).

Select that option and continue. This window may appear for each project, so select *Load project normally* each time, or change the checkbox that indicates not to warn again.

26

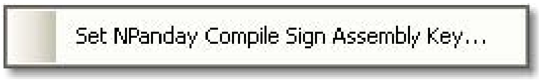
26.1 Setting Assembly Keys

Assembly keys are used to uniquely identify the project to be installed in the Global Assembly Cache or also referred to as GAC.

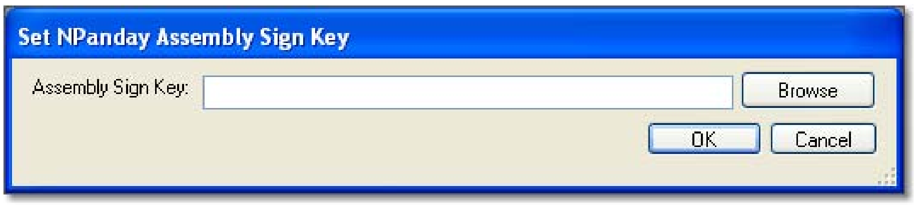
To set an assembly key to a project, generate a strong name key by running the command

```
sn -k [filename] .snk
```

Right click on the project and select *Set NPanday Compile Sign Assembly Key...*



Put the strong name key that was generated in the Assembly Sign Key field.



Setting assembly sign key

26.1.1 Summary

We have now covered much of the functionality of NPanday Visual Studio Add-in. In subsequent sections of the guide, we will examine the available configuration options.

The next topic is Configuring a Remote Repository, or you can return to the index.

27

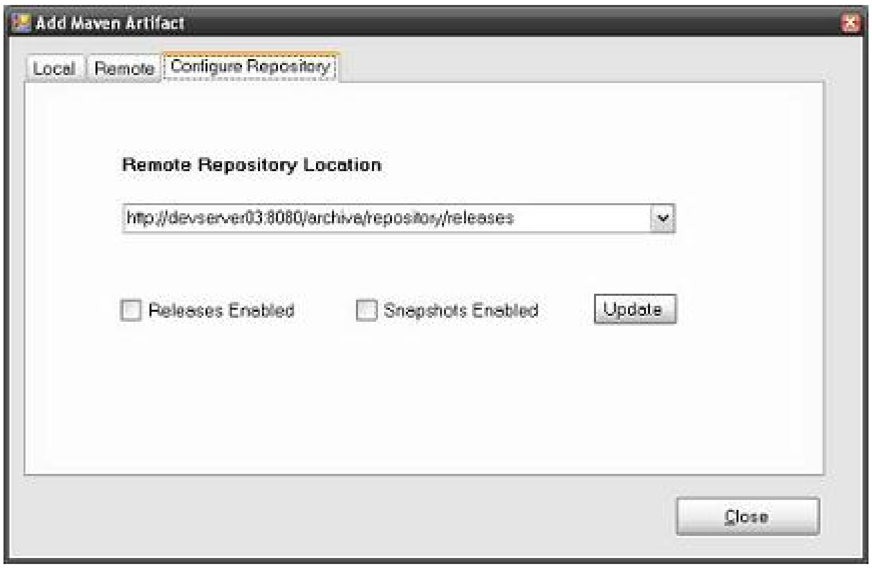
27.1 Remote Repository Configuration

A remote Maven repository can be configured using the Visual Studio IDE or it can be configured manually by adding information to the Maven settings file.

27.1.1 Automatically Configure a Remote Maven Repository via the IDE

To access a remote Maven repository you can configure it from within Visual Studio. Visual Studio must be open, the NPanday Build System must be running, and you must have a project loaded. Then:

1. Right-click on a project and select Add Maven Artifact... from the menu.
2. In the Add Maven Artifact pop-up window, click the Configure Repository tab.
3. Select the URL of the repository in the pop-up window.
4. If the remote repository allows snapshots or released artifacts to be stored there, then check the appropriate box.
5. Click Update. This will save the configuration to `C:\Documents and Settings \[user_home]\.m2\settings.xml`. To edit multiple repositories, repeat steps 3 to 5 by selecting another repository to be configured form the drop down list.
6. Click Close when done configuring the repository.



Remote Repository Location

http://devserver03:8080/archiva/repository/releases

Releases Enabled [ ]
Snapshots Enabled [ ]

Sample configuration for Remote Repository

NOTE: The repository is stored in a profile NPanday.id this profile is then added into the activeProfiles list in `settings.xml` as soon as an NPanday remote repository is added.

27.1.2 Manually Configure a Remote Maven Repository

To manually add a remote repository, add the following lines in your `C:\Documents and Settings\[user_home]\.m2\settings.xml` file within the `<profiles>` `</profiles>` tag. Modify the values for the `<repository>` and `<id>` elements with the repository url you want to access.

```
<profile>
  <activation>
    <activeByDefault>true</activeByDefault>
  </activation>
  <repositories>
    <repository>
      <releases>
        <enabled>true</enabled>
      </releases>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
      <id>http://repo1.maven.org/maven2/</id>
      <url>http://repo1.maven.org/maven2/</url>
    </repository>
  </repositories>
  <id>NPanday.id</id>
</profile>
```

You should change the values of `enabled` for each of the artifact types, dependencing on whether the repository contains that type of version.

27.1.3 Summary

The next configuration options are the Maven settings themselves. You can also return to the index.

28

28.1 Executing Maven Goals

As we learned in the quick-start tutorial, it is possible to run Maven goals from within Visual Studio using NPanday. This section explores the concept of the build lifecycle in more detail.

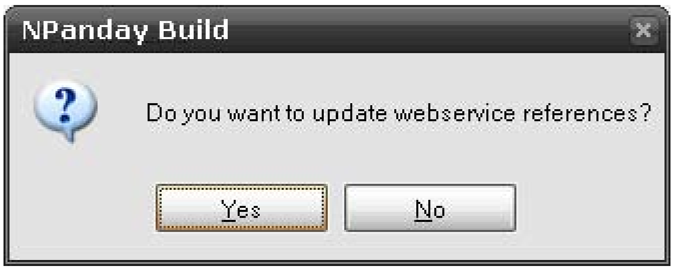
28.1.1 Compile

In the Solution Explorer, right-click on the name of the project you want to compile, then select `Current NPanday Project > Build` to build the selected project. Or, `All NPanday Projects > Build` to build the parent and its sub-projects.

Note: In some circumstances, Maven is currently unable to build (compile) without installing the project first. As a workaround, execute `All NPanday Projects > Install` first before `All NPanday Projects > Build [compile]`.

For projects with dependencies from remote repositories, the build might fail. As a workaround, re-import the project then, execute the build goal (`Current NPanday Project > Build` or `All NPanday Projects > Build`).

For projects with web references, a dialog box is prompted for updating the Web Services Description Language (`wsdl`). Click Yes to continue with the update, or, No to skip updating.



Do you want to update webservice references?

The NPanday .NET Build Tool performs the compile on the project and sends corresponding information to the Output window (including a message saying the build was successful). You can scroll up and down in the output window to display all the text.

You can also execute the normal Visual Studio build on your projects. To invoke Maven to perform `build`/ `install`/ `test`/ `clean`, you must select the option from the All NPanday Projects or Current NPanday Project sub-menu.

When compiling a web application project, the NPanday .NET Build Tool performs an extra step by calling the `Aspnet_compiler` to validate the ASP section of the project. The Build Tool performs this step right after calling the CSharp compiler and Visual Basic compiler.

28.1.2 Test

To use the NPanday .NET Build Tool to test a Project:

1 From the Solution Explorer, right-click on the name of the project you want to test, then select `Current NPanday Project > Test` to perform the test to the selected project. Or, `All NPanday Projects > Test` to perform the test to the parent project and its sub-projects.
2 The NPanday .NET Build Tool performs the tests on the project and sends corresponding information to the Output window (including a message saying the build was successful). You can scroll up and down in the output window to display all the text.

Note: When the test goal output result is not refreshed for C# test project, the project should be reimported, and the value of the test should also be changed. Then, perform `Current Project: Test` (or, `All NPanday Projects > Test`).

28.1.3 Install

From the Solution Explorer, right-click on the name of the project you want to build, then select Current NPanday Project > Install to install the selected project. Or, All NPanday Projects > Install to install the parent project and its sub-projects.

The NPanday .NET Build Tool installs the artifacts into your local repository. For web projects with POM as packaging, the `target` directory is not created and only the parent `pom.xml` is put in the local repository which is the default behavior.

For web application projects, the project zip file is produced during this phase, specifically during the `package` phase.

Corresponding information is sent to the Output window (including a message saying the build was successful). You can scroll up and down in the output window to display all the text.

Next, verify the artifacts were placed into the .NET local repository under the following path: `C:\Documents and Settings\[user_home]\.m2\uac\[gac_architecture]\artifactId\Version__GroupId`

For example: `C:\Documents and Settings\[user_home]\.m2\uac\gac_msil\NPanday.Test\1.0__NPanday`

The following explains what each element in the path means:

- `[user_home]` The user's home directory.
- `gac_architecture` The architecture type such as `gac_msil`, `gac_32`, etc.
- `artifactId` This is similar to the Maven `artifactId`. It is equivalent to the project's module name. In the example above, the artifactId is `NPanday.Test`.
- `Version__GroupId` The version of the artifact and the group it is in. In the example above the version is `1.0` and the GroupId is `NPanday`.

The contents in C:\Documents and Settings\[user_home]\.m2\uac\gac_msil SHOULD NOT be manually modified or the project build will fail due to artifacts that are not properly indexed and are not synchronized with the repository. In case the contents have been modified, delete the C:\Documents and Settings\[user_home]\.m2\uac directory then re-install the project.

Finally, Verify that the following are also created under the following paths:

- `C:\Documents and Settings\[user_home]\.m2\pab`
- `C:\Documents and Settings\[user_home]\.m2\npanday-settings.xml`

Contents of npanday-settings.xml looks similar to the following:

```
<?xml version="1.0" encoding="utf-8"?>
<npandaySettings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd=
  <operatingSystem>
    Microsoft Windows NT 5.1.2600 Service Pack 2
  </operatingSystem>
  <defaultSetup>
    <vendorName>MICROSOFT</vendorName>
    <vendorVersion>2.0.50727</vendorVersion>
    <frameworkVersion>2.0.50727</frameworkVersion>
  </defaultSetup>
  <vendors>
    <vendor>
      <vendorName>MICROSOFT</vendorName>
      <vendorVersion>1.1.4322</vendorVersion>
      <frameworks>
        <framework>
        <frameworkVersion>1.1.4322</frameworkVersion>
          <installRoot>
            C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322
          </installRoot>
        </framework>
      </frameworks>
    </vendor>
    <vendor>
      <vendorName>MICROSOFT</vendorName>
      <vendorVersion>2.0.50727</vendorVersion>
      <frameworks>
        <framework>
        <frameworkVersion>2.0.50727</frameworkVersion>
          <installRoot>
            C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727
          </installRoot>
          <sdkInstallRoot>
            C:\Program Files\Microsoft Visual Studio 8\SDK\v2.0\
          </sdkInstallRoot>
        </framework>
      </frameworks>
    </vendor>
  </vendors>
</npandaySettings>
```

28.1.4 Clean

To use the NPanday .NET Build Tool to clean a project (removes the `target\` directory containing the files that were generated at build-time from the project's working directory):

1. From the Solution Explorer, right-click on the name of the project you want to clean, then select Current NPanday Project > Clean to clean the selected project. Or, All NPanday Projects > Clean to perform the clean goal to the parent project and its sub-projects.
2. The NPanday .NET Build Tool performs the clean on the project and sends corresponding information to the Output window (including a message saying the build was successful). You can scroll up and down in the output window to display all the text.

28.1.5 NPanday's Maven Phase Algorithm

Algorithm that differs the Visual Studio Execution from Console Command.

1. All Project - Is triggered when the user right click in the solution explorer and select `All NPanday Projects > Build` or `All NPanday Projects > Clean` or `All NPanday Projects > Test` or All NPanday Projects > Install>>>
   - A Save All Documents that are open in Visual Studio.
   - B Retrieve the solution that is open in Visual Studio.
   - C Loop through all the projects in the solution.
   - D Check if it has a Web Reference, if it has it will update the Web Reference.
   - E Execute the pom.xml of the solution may it be Build/Clean/Test/Install.
2. Current Project - Is Triggered when the user right click in the solution explorer and select `Current NPanday Project > Build` or `Current NPanday Project > Clean` or `Current NPanday Project > Test` or `Current NPanday Project > Install`
   - A Save All Documents that are open in Visual Studio.
   - B Retrieve CURRENT pom.xml and CURRENT project.
     - a If pom.xml does not exist, set errorMsg to "Pom Not Found" Error.
     - b If the packaging in the pom.xml is "pom", set errorMsg to "Pom may not be Project's Pom" Error.
   - C Check if it has a Web Reference, if it has it will update the Web Reference.
   - D Execute the pom.xml of the solution may it be Build/Clean/Test/Install.

28.1.6 Summary

Continue on to Setting Assembly Keys, or return to the index.

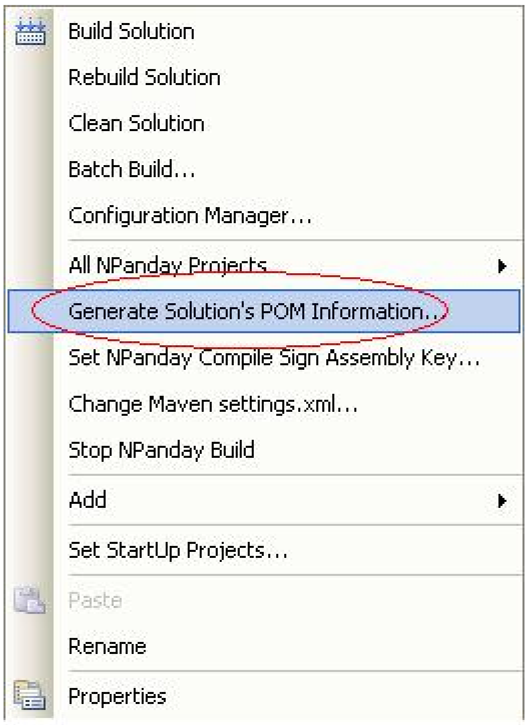
29

29.1 Importing Projects

This section provides more details on how on importing a Visual Studio project using the *Generate Solution's POM Information...* feature works. As we saw in the Quickstart example, this feature will generate `pom.xml` files from an existing Visual Studio solution file. The generated `pom.xml` file will contain the latest versions of the artifact dependencies used for the project.

Note: ASP.NET MVC should be installed to get the proper behavior when importing projects especially those projects with unsupported project types. See MVC projects for more details.

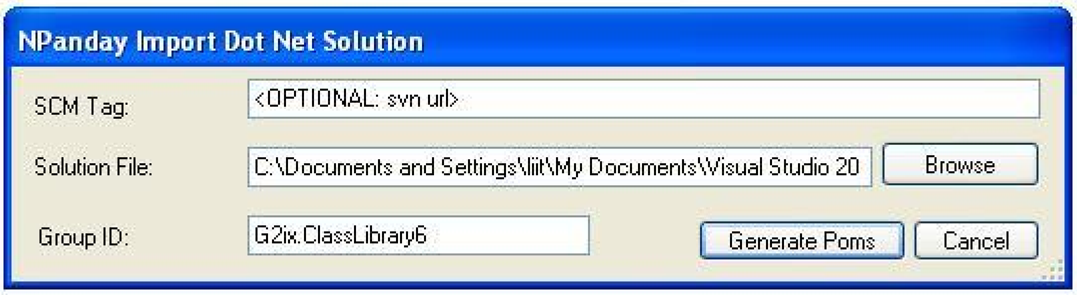
To import a project, first select *Generate Solution's POM Information...* from the project's context menu:



In the pop-up screen, Browse to the directory location of the parent solution file to use.

The *Group ID* is generated by default with the format of `[company_id].[solution_name]`.

The *SCM Tag* is optional. If you provide a url that is not reachable (e.g: it is authenticated and the credentials are not supplied or incorrect) by the Addin, you will see a warning, but the url will still be added to the parent pom.



SCM Tag: <OPTIONAL: svn url>
Solution File: C:\Documents and Settings\liit\My Documents\Visual Studio 20
Group ID: G2ix.ClassLibrary6

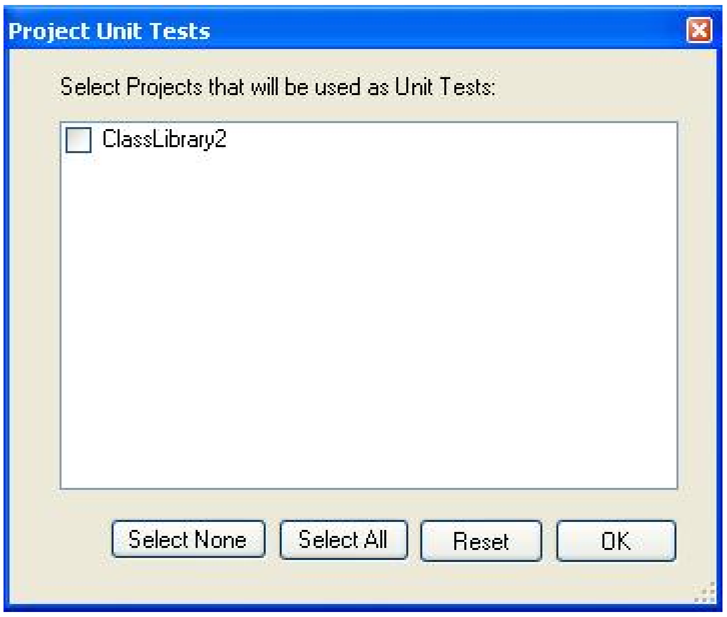
Importing parent solution file

When you are ready, click *Generate POMs*.

Underscores and spaces in the artifact's Group ID are automatically deleted when added to the project's `pom.xml`. The table below shows the conversion of certain values of groupIds.

Artifact Group Id	**Converted Group Id**
The.Project	The.Project
the.project	the.project
the project	TheProject
the_project	TheProject

Next, the project unit tests window will appear. If you have modules that are for unit tests only, select them and click OK.



Select Projects that will be used as Unit Tests:

☐ ClassLibrary2

Project unit tests list

`pom.xml` files will be generated for all the projects that are included in the solution file, including the parent project and its sub-projects, if there are any.

Note: The Units Tests window is not displayed when importing unsupported project types. Hence, an unsupported project type warning is displayed. For multi-module projects with supported and/ or unsupported project types, the supported projects are included in the parent POM while for the unsupported project types, these are not included in the parent POM, and a warning is displayed.

For projects with web references, the `pom.xml` generated will contain `maven-wsdl-plugin` configuration similar to the following:

```
<plugin>
  <groupId>npanday.plugin</groupId>
  <artifactId>maven-wsdl-plugin</artifactId>
  <extensions>true</extensions>
  <executions>
    <execution>
      <goals>
        <goal>wsdl</goal>
      </goals>
    </execution>
  </executions>
  <configuration>
    <webreferences>
      <webreference>
        <namespace>AnotherWebService</namespace>
        <path>Web References/AnotherWebService/AnotherWS.wsdl</path>
        <output>Web References/AnotherWebService/</output>
      </webreference>
    </webreferences>
  </configuration>
</plugin>
```

The wsdl goal is the one responsible for building and creating webservice proxy classes. And, all the web references used in the project are listed inside the `<webreferences>` tag. For more instructions on adding, renaming, and deleting web references, refer to Web References section of this guide.

Other project types will generate corresponding plugins needed to build that type of project (for example, the compile plugin, aspx plugin, or msbuild plugin).

29.1.1 Importing Projects with Resource Files

If the project you are importing includes a resource file, the `maven-resgen` plugin is added to `pom.xml` during import.

In the plugin's configuration, you can find the list of resource (`resx`) files with corresponding names. By default, `RootNamespace` is the assembly name and `RootNamespace.Filename` is the filename of the `resx` files.

To ensure that the plugin works, the following are required:

- The classname and the filename of the item that contains the resx files must have the same name. Example: `"public class frmMaestroName" = frmMaestroName.resx = frmMaestroName.cs`

  Note: Changing the codefile(.cs/.vb) automatically updates the resource(resx) file. In some instances, however, this does not hold true.
- The namespace of the project should be equal to the DefaultNamespace of the project assembly. To check the default name,

- Right-click on the project and select Properties. This displays a project property window.
- Select the Application tab.
- In the DefaultNamespace field, you will find `RootNamespace`.

Once the above requirements are met, the importer automatically does the work and there is no need to edit `pom.xml`.

Removing the configuration part of the `maven-resgen` plugin allows the plugin to locate all the resx files and compile them as `ArtifactId.Filename`. If you set the namespace as the artifactId, you do not need to list all the resx files in the `pom.xml`.

Note: This feature does not support cultured esx files such as `eu-US`, `it`, `jp`, `de`, `eu-UK`, and even custom culture.

29.1.2 Importing WPF/WCF Projects with XAML files

If the project you are Importing is a WPF/WCF project and it contains a XAML file the MSBuildPlugin will be automatically added into that project's pom file.

```
<plugin>
  <groupId>npanday.plugin</groupId>
  <artifactId>NPanday.Plugin.Msbuild.JavaBinding</artifactId>
  <executions>
    <execution>
      <goals>
        <goal>compile</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

This will generate the needed *.g.cs/*.g.vb files that are needed for those XAML files.

Note: The user would encounter double dependency errors, if the project was created with 3.0 .NET framework. We suggest that you change your framework to 3.5 and then delete the references that are already included as defaults in NPanday. This will then allow your project to run successfully.

29.1.3 Importing ASP.NET MVC Projects

As of NPanday 1.1, MVC projects are not support for import as Maven projects from the Visual Studio Add-in. However, in some situations they may be incorrectly imported as a normal C# or VB project.

To avoid this problem, make sure you have installed ASP.NET MVC Release Candidate 2 which can be obtained from http://www.microsoft.com/downloads/details.aspx?FamilyID=ee4b2e97-8a72-449a-82d2-2f720d421031&displaylang=en.

This must be installed to skip the unsupported project types and be able to get the correct project type GUID.

29.1.4 Summary

This section has described how to import a project, and the nuances of various different supported project types.

You can now read on about executing Maven goals, or go back to the index.

30

30.1 Using the NPanday Visual Studio Add-in

By this point, you should have NPanday successfully installed as shown in the Installation instructions, and have Visual Studio started with the NPanday Build Tool running.

This guide will walk you through a basic introduction to the Visual Studio Add-in, how to use it for typical project tasks, and what configuration options are available.

Finally, we will discuss the use cases where you might need to manipulate the Maven POM directly and other advanced use cases.

After completing this guide, you may want to read the guide to Using NPanday with your Maven Development Infrastructure.

If you encounter any problems, don't forget to consult the Troubleshooting guide.

30.1.1 Contents

- Quick Start Guide - starting a new project and making it buildable by Maven
- Working with references - how adding references impacts the project, and how it works with Maven depedencies
- Working with web references - how web references in a web application work with NPanday
- Keeping references in sync - how to ensure project files are portable and references are always available
- Creating Maven POM's from Visual Studio projects - more detail on the import process covered in the quick start guide, including different project types supported
- Executing Maven Goals - more detail on the integration of the Maven lifecycle with Visual Studio by the NPanday Add-in
- Setting the Assembly Key - how to set a strong name on your build using NPanday
- Configuring a Remote Repository - the steps necessary to have a remote repository for downloading Maven artifacts from
- Maven settings - how to configure an alternate Maven settings file
- Releasing .NET projects with Maven - how to use the Maven release mechanism from the command line

31

31.1 Quick Start Guide

This section is provides a quick overview of the NPanday .NET Build Tool by creating a sample project. You must first have installed the Add-in according to the instructions in the installation guide.

31.1.1 Starting the Add-in

Start Visual Studio ensuring that your PATH is set correctly, as described in the installation guide.

Next, in Visual Studio start the NPanday Build System from Tools>NPanday Build System. You will see NPanday starting in the Visual Studio Output pane.

Note: After you start the NPanday Build System and it is running, it no longer appears as an option on the Tools menu (and reappears again when Visual Studio is stopped and started). Therefore, if you had previously started the NPanday Build System and it is still running, you do not need to start it again in this step.

31.1.2 Creating a Project

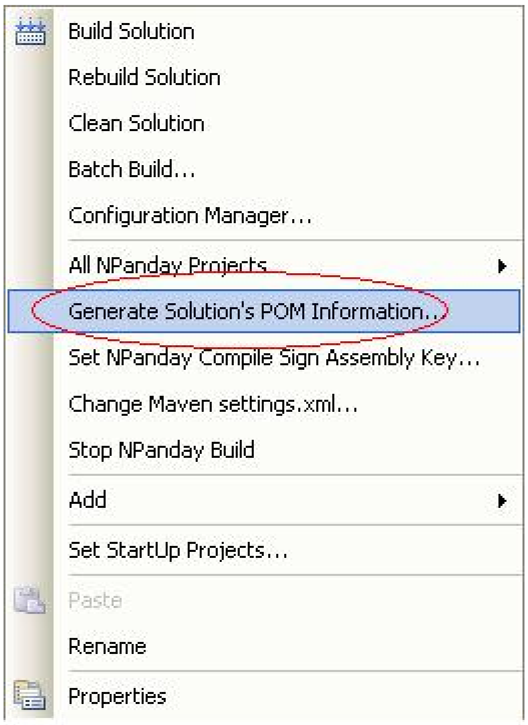
Creating a project for use with NPanday is the same as you would normally do so for any project within Visual Studio. In this case, we will create a class library.

1. Got to File > New > Project...
2. Retain the default of a C# Class Library (if you are using a different language variant of the IDE that will also work)
3. In the *Name* field, type `NPandayTestQS`
4. Keep the other options the same (using the same name for the Solution and creating a directory for the solution) and press OK.

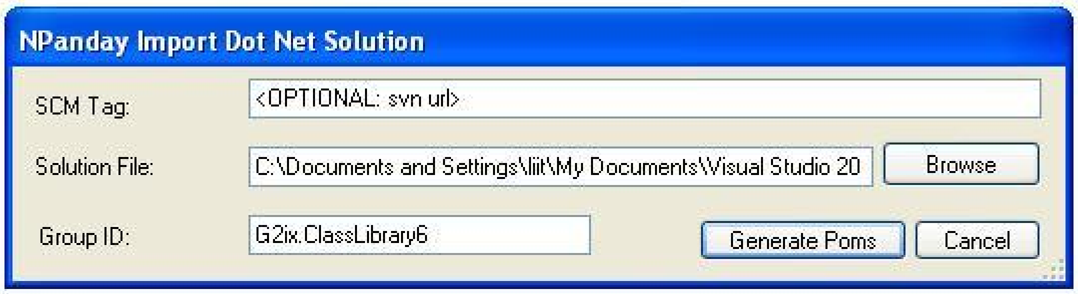
You now have a simple, standard, project. However, nothing has been generated for Maven to be able to build it, so for that we need to generate a `pom.xml` file.

31.1.3 Generating the POM

NPanday makes creating the POM file(s) easy. To do so, right click on the project solution (or any of the projects) in the Solution Explorer and select *Generate Solution's POM Information...*:



Next, you will see the following dialog:



NPanday Import Dot Net Solution

SCM Tag: <OPTIONAL: svn url>

Solution File: C:\Documents and Settings\liit\My Documents\Visual Studio 20

Group ID: G2ix.ClassLibrary6

We'll look at this in more detail shortly, but for now accept the default settings, and do likewise on the next dialog asking about test projects. Finally, NPanday will inform you that it has created two `pom.xml` files: one for the solution, and one for the class library project. These files won't appear in the Solution Explorer - NPanday keeps them hidden as they don't need to be modified directly. However, they should be checked into your source control system to ensure that they can be used by other developers and the shared build infrastructure. In the next section we will take a closer look at the POM files.

31.1.3.1 About the POM file

First, we need to understand what the `pom.xml` file is. This is the sole descriptor used by Maven to build an entire project. In some respects, it is similar to the `.sln` and `.csproj` (or `.vbproj`) files, however Maven uses a different approach - while these are 'scripts', the POM is 'declarative', indicating where parts of the project are located, and which Maven plugins are used to build the project, utilizing that project information. By being declarative in this fashion, many plugins can operate on the same information without having to reconfigure each specifically.

Let's look at POM for the solution first:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
         xmlns="http://maven.apache.org/POM/4.0.0">
  <modelVersion>4.0.0</modelVersion>
  <groupId>MyCompany.NPandayTestQS</groupId>
  <artifactId>NPandayTestQS-parent</artifactId>
  <packaging>pom</packaging>
  <name>MyCompany.NPandayTestQS : NPandayTestQS-parent</name>
  <version>1.0-SNAPSHOT</version>
  <modules>
    <module>NPandayTestQS</module>
  </modules>
</project>
```

You will notice a few important elements here. First, there is the *Group ID* which is the value that you entered into the previous dialog. There is also an *Artifact ID* that is the name of the solution with 'parent' appended. Further down, there is a *Version* that is set to the default of `1.0-SNAPSHOT`. These 3 elements constitute the project *co-ordinate*, which is a unique identifier across all projects you can access to this particular project. This is the reason that the solution appends `-parent` to the end, to distinguish it from the child project with the same name.

Within the group/artifact ID combination, each project is then versioned to distinguish between different releases. The default is `1.0-SNAPSHOT`, which means "in development towards the 1.0 release". Maven places special significance on the `SNAPSHOT` addition to a version, which means that the project is changing. When a version is set to a concrete one such as `1.0`, Maven considers it to always remain the same and changes to go under a new version (such as `1.1-SNAPSHOT`).

We will later use these co-ordinates to locate and utilize dependencies on other projects from within our own project, even if we don't have a copy locally.

We also see the *Modules* section that shows which subproject exist. This contains a reference to the class library's project directory - so let's now look at the POM for that:

```
<?xml version="1.0" encoding="utf-8"?>
<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
         xmlns="http://maven.apache.org/POM/4.0.0">
  <parent>
    <artifactId>NPandayTestQS-parent</artifactId>
    <groupId>MyCompany.NPandayTestQS</groupId>
    <version>1.0-SNAPSHOT</version>
    <relativePath>..\pom.xml</relativePath>
  </parent>
  <modelVersion>4.0.0</modelVersion>
  <artifactId>NPandayTestQS</artifactId>
  <packaging>library</packaging>
  <name>MyCompany.NPandayTestQS : NPandayTestQS</name>
  <build>
    <sourceDirectory>./</sourceDirectory>
    <plugins>
      <plugin>
        <groupId>npanday.plugin</groupId>
        <artifactId>maven-compile-plugin</artifactId>
        <extensions>true</extensions>
        <configuration>
          <frameworkVersion>2.0.50727</frameworkVersion>
          <includeSources>
            <includeSource>Class1.cs</includeSource>
            <includeSource>Properties\AssemblyInfo.cs</includeSource>
          </includeSources>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

This project starts with the *Parent* definition, which points back to the project for the solution. This is used to inherit any shared settings among subprojects (though by default, there is nothing saved in the parent POM that will be shared).

Next we have the artifact ID for this project, which uses the library name. The group ID and version can be omitted, since by default they will use the same one as the parent project.

In this project, we have altered the *Packaging* element to be of type `library`. While the parent was a `pom`, indicating it didn't build anything additional, this type indicates that Maven and NPanday should include all the standard build steps for a .NET class library.

Along with this we now have a *Build* section. This contains two further elements - first, the *source directory*, that says the Visual Studio convention of storing source files in the project directory (and subdirectories) will be followed. Next is the *Plugins* section, which is where Maven is told to use NPanday to compile, and what framework and source code to use. For other project types, additional plugins may be added (see the Plugins Reference for more available plugins).

While it is not necessary to be able to create these POM files to be able to use NPanday, understanding how they work, and in particular the Maven co-ordinate system, will help to understand what NPanday is doing "behind the scenes" when you use some of its other functionality.

31.1.4 Building the Project

With this in place, we can now check that Maven is able to build the project. Right click on the solution or project and select *All NPanday Projects*, then *Build [compile]*.

In the Output Window (if the NPanday Execution Output view is selected from the dropdown), you will see something similar to the following:

```
------------------------------------------------------------------
Executing Maven
Pom File: C:\...\Visual Studio 2008\Projects\NPandayTestQS\pom.xml
Goal: compile
Arguments: compile
NPanday Command: c:\...\apache-maven-2.2.1\bin\mvn.bat compile
------------------------------------------------------------------
[INFO] Scanning for projects...
[INFO] Reactor build order:
[INFO]   MyCompany.NPandayTestQS : NPandayTestQS-parent
[INFO]   MyCompany.NPandayTestQS : NPandayTestQS
[INFO] ------------------------------------------------------------------------
[INFO] Building MyCompany.NPandayTestQS : NPandayTestQS-parent
[INFO]    task-segment: [compile]
[INFO] ------------------------------------------------------------------------
[INFO] No goals needed for project - skipping
[INFO] ------------------------------------------------------------------------
[INFO] Building MyCompany.NPandayTestQS : NPandayTestQS
[INFO]    task-segment: [compile]
[INFO] ------------------------------------------------------------------------
[INFO] [compile:initialize {execution: default-initialize}]
...
[INFO] [resolver:resolve {execution: default-resolve}]
...
[INFO] [NPanday.Plugin.Settings.JavaBinding:generate-settings {execution: default-g
[INFO] [compile:generate-assembly-info {execution: default-generate-assembly-info}]
...
[INFO] [compile:process-sources {execution: default-process-sources}]
...
[INFO] [compile:process-test-sources {execution: default-process-test-sources}]
...
[INFO] [resgen:copy-resources {execution: default-copy-resources}]
[INFO] [resgen:generate {execution: default-generate}]
[INFO] [resgen:generate-existing-resx-to-resource {execution: default-generate-exis
[INFO] [compile:compile {execution: default-compile}]
...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO] ------------------------------------------------------------------------
[INFO] MyCompany.NPandayTestQS : NPandayTestQS-parent ........ SUCCESS [0.005s]
[INFO] MyCompany.NPandayTestQS : NPandayTestQS ............... SUCCESS [10.547s]
[INFO] ------------------------------------------------------------------------
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 18 seconds
[INFO] Finished at: Wed Feb 03 12:58:47 EST 2010
[INFO] Final Memory: 12M/24M
[INFO] ------------------------------------------------------------------------
NPanday Execution is Successful!
```

Here, both projects are built (since it was run from the solution level). The solution POM has no steps to perform to compile. However, you can see several steps (`compile:initialize`, `resolver:resolve`, and so on) performed for the library. These are all defined by the POM's `packaging` element.

Of course, the existing built-in commands are still available for building from Visual Studio, so development need not change day to day - but the build is there to be sure that the POM works as expected, so that you have the same build as other systems that will use it.

You may have noticed that there were other options that *Build* - namely *Test*, *Install* and *Clean*. There are other parts of what is known as the Maven *Build Lifecycle*.

31.1.4.1 The Build Lifecycle

Maven uses the POM to drive builds that a constructed as *patterns*. No matter what type of project you are building it can be shaped to Maven's build lifecycle, which is a sequence of phases always run in the same order, and for which a given phase can be sure that the previous phases have always executed as part of this build.

There are quite a number of phases that Maven plugins can participate, and that you can run from the `mvn` command, but the Visual Studio Add-in exposes the primary ones:

- `compile` - compile source code into binary output
- `test` - run unit tests (such as NUnit) against the code to verify that the package will work
- `install` - produce the package and install it into Maven's *local repository*.

What this means is that if you run `mvn install`, you will actually run compile, then test, then install (and the intermediate steps). Maven views testing as important, so this structure ensures that you run any unit tests you have before using the final DLL, for example. Running the *install* phase is the most common one used when building a project with Maven.

The installation phase is different to the traditional meaning of installing software. Maven maintains a *local repository* on the current machine (by default in `%HOMEDRIVE%%HOMEPATH%\.m2\repository`) where all build output is stored. This common storage allows sharing between different projects on the same machine when they express a dependency on another project. You have already encountered this when installing NPanday, as Maven uses the local repository to house it's own functionality as well.

We'll also learn later that Maven utilizes *remote repositories* to obtain dependencies from other sources, and to publish build output for sharing with other users. In this case, the local repository acts as a local intermediatary and cache between Maven and the remote repositories.

In addition to the above phase, there is the *Clean* option, which uses a different 'lifecycle' for cleaning up any build output such as the `target`, `Bin` and `Obj` directories.

31.1.5 Summary

In this simple example, we've seen how we can turn an existing .NET project into a functional Maven project, and learned some of the basic terminology and interactions with the new build system. In further parts of the guide, we will learn more about how Visual Studio and Maven interact through NPanday.

You can now read on about working with references, or go back to the index.

32

32.1 Project Dependencies

This feature allows the Visual Studio user to browse managed repositories and add and delete Maven dependencies to and from the repositories from within the Visual Studio IDE.

A remote repository must be configured before attempting to use this feature. Please see the Remote Repository section for instructions.

Note: Adding dependencies should be done using Visual Studio while manually adding is not recommended. At this time, the Visual Studio add-in will not detect changes to the Maven dependencies and add them to your Visual Studio project. If you manually add a dependency and re-import the project in Visual Studio, the recently added dependencies are deleted from the `pom.xml` file.

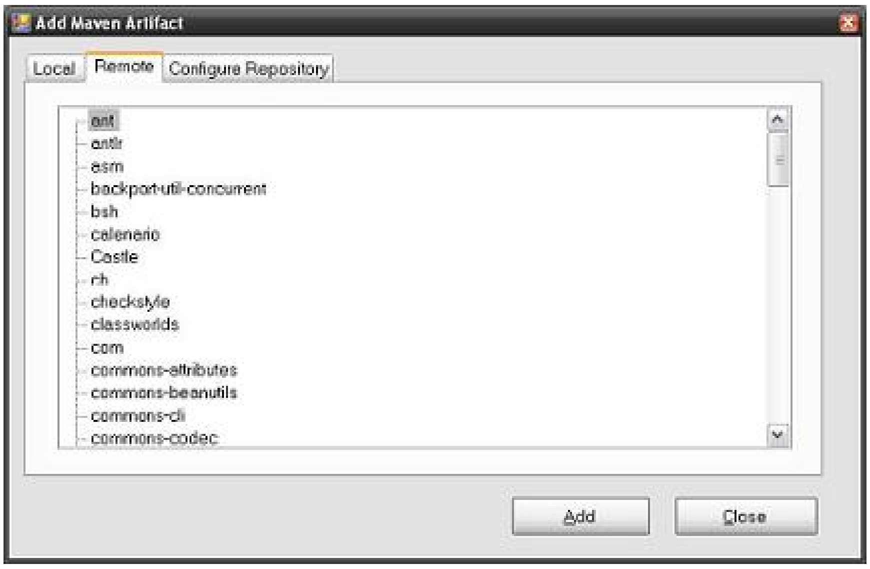
32.1.1 Adding Maven Artifacts

Adding a Maven artifact allows you to refer to a DLL or other dependency that may not yet be installed on the local machine. This is done by consulting a Maven *remote repository* for available dependencies, and adding a reference to the Maven `pom.xml`.

When adding an artifact dependency into a web site project, there will be no changes in the `pom.xml`. This is because the artifacts are put directly into the project's `bin\` directory (one is created when this is missing), thus, `<dependencies>` in the `pom.xml` is not needed.

To add an artifact:

1 Right-click on a project and select Add Maven Artifact... from the menu.
2 In the Add Maven Artifact pop-up window there are tabs for each repository you have configured. Click the tab for the repository in which the desired artifact is located.
3 Scroll to select the desired artifact (must be a `DLL` file, not a `JAR` nor an `OCX` file):
4 Click Add to add the selected artifact and exit the form. Or, double click on the artifact to be added. Repeat this step to add multiple artifacts. Then, click Close to exit the form.

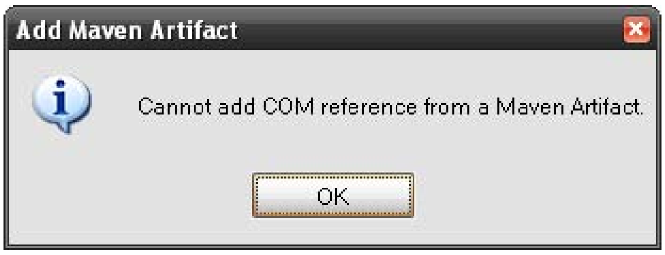


Add Maven Artifact screen

The artifacts will be added as a reference to the project and added as a dependency in the POM file. It is also downloaded to the `C:\Documents and Settings\[user_home]\.m2\uac` directory.

32.1.1.1 COM References

When attempting to add an `.ocx` file into the repository, a warning message similar to the following will be displayed:



Add Maven Artifact

Cannot add COM reference from a Maven Artifact.

COM reference error message

See below for adding COM references directly.

32.1.1.2 .NET References

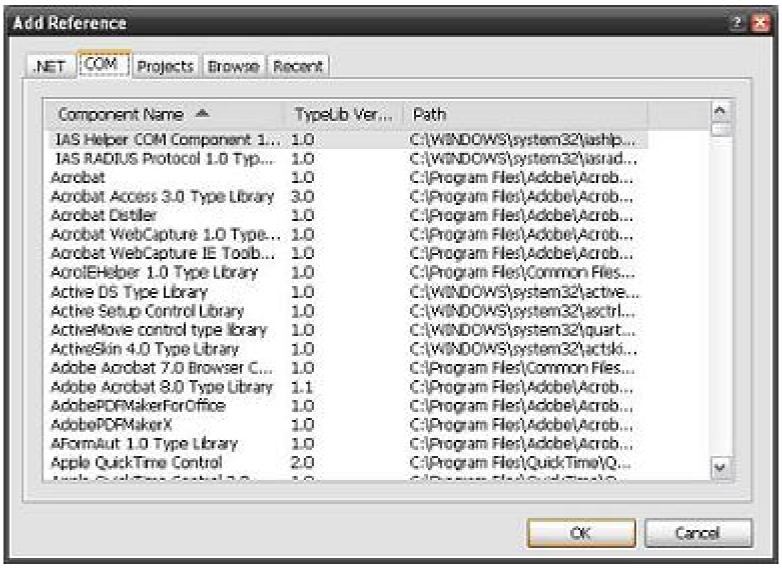
As of version 1.2 when adding .NET References NPanday will no longer add system scope and system path properties on the .NET dependency.

32.1.2 Adding References

You can also add traditional references to the project and have NPanday translate them into Maven dependencies.

To add references to the project:

1 Right click on the project that you want to add a reference to.
2 Select Add Reference... from the menu list.
3 Select the tab to which the reference belongs (**.NET**, **COM**, **Projects**). Or, Browse to look for the specific artifact to be reference and Recent for artifacts recently accessed.
4 Select the references to be added from the list as shown in the figure above. Scroll to select the desired object reference.
5 Click OK.



Component Name	TypeLib Ver...	Path
IAS Helper COM Component 1...	1.0	C:\WINDOWS\system32\iashlp...
IAS RADIUS Protocol 1.0 Typ...	1.0	C:\WINDOWS\system32\iasrad...
Acrobat	1.0	C:\Program Files\Adobe\Acrob...
Acrobat Access 3.0 Type Library	3.0	C:\Program Files\Adobe\Acrob...
Acrobat Distiller	1.0	C:\Program Files\Adobe\Acrob...
Acrobat WebCapture 1.0 Type...	1.0	C:\Program Files\Adobe\Acrob...
Acrobat WebCapture IE Toolb...	1.0	C:\Program Files\Adobe\Acrob...
AcroIEHelper 1.0 Type Library	1.0	C:\Program Files\Common Files...
Active DS Type Library	1.0	C:\WINDOWS\system32\active...
Active Setup Control Library	1.0	C:\WINDOWS\system32\asctrl...
ActiveMovie control type library	1.0	C:\WINDOWS\system32\quart...
ActiveSkin 4.0 Type Library	1.0	C:\WINDOWS\system32\actski...
Adobe Acrobat 7.0 Browser C...	1.0	C:\Program Files\Common Files...
Adobe Acrobat 8.0 Type Library	1.1	C:\Program Files\Adobe\Acrob...
AdobePDFMakerForOffice	1.0	C:\Program Files\Adobe\Acrob...
AdobePDFMakerX	1.0	C:\Program Files\Adobe\Acrob...
AFormAut 1.0 Type Library	1.0	C:\Program Files\Adobe\Acrob...
Apple QuickTime Control	2.0	C:\Program Files\QuickTime\Q...

Sample display of list of references

NPanday will take care of adding the reference to the Maven POM automatically. In most cases this will also be made available on any other system building from the Maven project. In the case of an inter-project dependency, it will rely on the project being built on the target machine before the project that has the dependency on it - but all that is needed here is to build the whole solution as Maven will then take care of the correct ordering.

However, some references are non-portable when they are added.

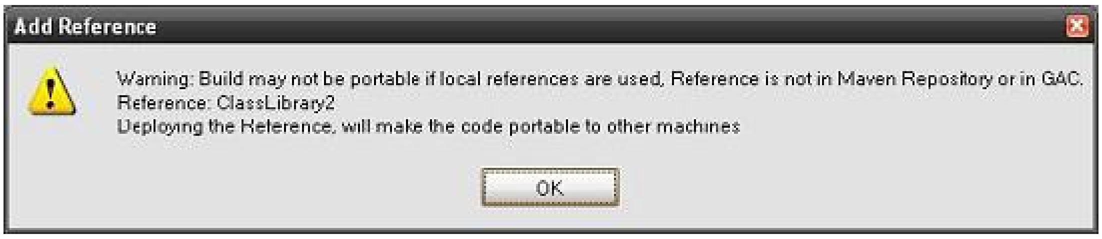
32.1.2.1 Intra Solution References

When Adding Intra Solution references use the Visual Studio "Add Reference" and not the "Add Maven Artifact" command. This sets the csproj/vbproj file & the pom.xml correctly simultaenously, and allows VS to build the project without needing to resolve the reference.

32.1.2.2 Non-portable References

When adding a COM reference, the actual reference (DLL) is copied to `C:\WINDOWS\assembly\GAC_MSIL`. Wherein, the path to the reference found in this directory is used as the reference's `<systemPath>` in the `pom.xml`. References such as this rely on the given DLL being available at the same location on each machine.

When adding an artifact reference that is not installed in the system, NPanday will prompt a warning message then adds the reference in the `pom.xml` as a system scope dependency.



Warning: Build may not be portable if local references are used, Reference is not in Maven Repository or in GAC.
Reference: ClassLibrary2
Deploying the Reference, will make the code portable to other machines

Reference not installed in the system

However, the above warning message is not displayed for web site projects even if the artifact reference to be added is not installed in the system. This is because the artifact references for web projects are directly added in the project's `bin\` directory (one is created when this is missing), thus, no need to add it in the `pom.xml`.

The following POM snippet shows a system dependency:

```
<dependency>
  <groupId>artifact_group_id</groupId>
  <artifactId>my.lib</artifactId>
  <version>2.2</version>
  <type>library</type>
  <scope>system</scope>
  <systemPath>C:\path\to\your\library.dll</systemPath>
</dependency>
```

32.1.3 Removing Artifacts and References

The Remove Artifact feature removes the artifact as a reference to the project and as a dependency in the POM file.

To remove an artifact, you can select the artifacts to remove and click the Delete key. This works for both Visual Basic and C# projects.

Alternatively, to remove Visual Basic project artifact:

1. Right-click on a project and select Properties from the menu.
2. Click on the References tab, and select the desired artifact.
3. Click Remove.

Alternatively, to remove C# project artifact:

1. Inside the project, browse through the References folder.
2. Right click on the desired artifact and select Remove.

32.1.4 Summary

This has covered all options for adding one of Maven's most powerful features - dependency resolution - to your .NET projects by using NPanday.

The next section continues with web references. You can also go back to the index.

In the next section we revisit importing a project into Maven. Other topics are available on the index.

33

33.1 Making VS Project Files Portable

To be able to make VS project files portable, NPanday 1.1 and above provides the Resync References functionality. This will synchronize the references so that the project will still run in other user's machine. Added references will be stored in hidden folder named ".references". When the project is used in other user's machine, the Resync References functionality will download and store the references in the ".references" folder.

To use Resync References button, right click on the project and go to 'All NPanday Projects' or 'Current NPanday Project' in the context menu and select 'Resync References'.

Intra-project references are skipped on Resync. Since this behavior was just applied recently in Version 1.0.2, old POMs with intra-project references might generate an error during Resync or Import. To fix this, remove the reference and add it back again. Or simply delete the POM and re-import the project.

How Resync Works:

- First NPanday will check for the dependency in the local repository, if the dependency is not found NPanday proceeds to the next possible location.
- If NPanday was not able to find the dependency in the local repository NPanday will proceed to check the remote repositories found in the user's settings.xml.
- For SNAPSHOT Dependencies NPanday bases from the maven-metadata.xml and gets the latest time stamp versioned dependency.

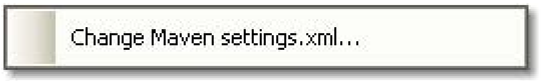
  In the next section we revisit importing a project into Maven. Other topics are available on the index.

34

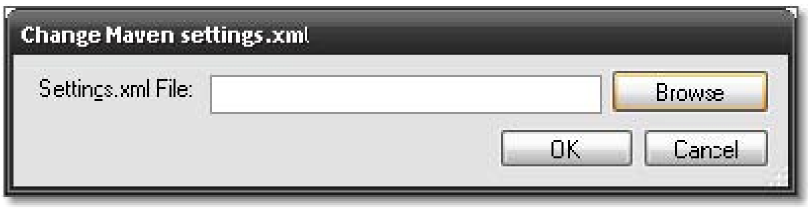
34.1 Using Custom Settings

This feature allows the Visual Studio user to customize the location of the `settings.xml` file that points to the repository of NPanday artifacts.

To change the location, right click on the project and select the *Change Maven settings.xml...*



In the pop-up screen, click Browse to browse through the directory location of the `settings.xml` to use.



Configure settings.xml location

34.1.1 Summary

This concludes the configuration section of the guide. Return to the index.

35

35.1 Web References

Web references are artifacts that refer to the web service. These are usually in web format.

The following procedures in adding, renaming, and deleting web references assumes that the project in which the web references will be added, renamed, or deleted has an existing `pom.xml` file from a previous import. Executing the following procedures will also make changes to the `pom.xml` file of the project.

Luckily these procedures are identical to normal Visual Studio development.

35.1.1 Adding Web References

1. Right click on the project that you want to add a web reference to.
2. Select Add Web Reference... from the menu list.
3. Select the web service you want to add, or you can choose to type the URL.
4. Supply a Name for the reference. This will be used to call the reference later in your code.

35.1.2 Renaming Web References

1. Right click on the web reference of the project application that you want to rename. Or, right click on the folder containing the wsdl file of the web service of the web project that you want to rename.
2. Select Rename.
3. Supply the new Name for the reference. This will be used to call the reference later in your code.

35.1.3 Removing Web References

1. Right click on the web reference of the project application that you want to delete. Or, right click on the folder where the wsdl file of the web service of the web project that you want to delete.
2. Select Delete.

35.1.4 Summary

As you can see, working with web references is very familiar - all of the procedures are the same, with the `pom.xml` changes handled by NPanday behind the scenes. You may, out of interest, want to review the POM after the addition of a web reference.

Having understood references, the next topic is maintaining them across different checkouts in the document making project files portable. You can also return to the index.

36

36.1 NPanday

NPanday is a project to integrate Apache Maven into .NET development environments. It includes both a Visual Studio add-in to integrate Maven, and a set of plugins for Maven that can build .NET projects uniformly from the command line.

NPanday enables you to use Visual Studio with a Maven-based development infrastructure, including:

- continuous integration servers for scheduled or triggered automated builds
- source control systems for assistance with the release process
- build artifact repositories for inter-project dependency management and centralised management of build output

You can read more about Maven's features on the Maven web site.

NPanday doesn't intend to replicate MSBuild's (or Visual Studio's build) behavior exactly, or necessarily serve as a replacement for Visual Studio users, but rather to:

- augment Visual Studio with dependency management and make the build reproducible in clean environments
- offer a consistent toolchain to build environments with projects in other languages (particularly those using Maven for Java projects)
- offer a solution for declarative builds for .NET projects that will work consistently whether Visual Studio is used or not
- expose other Maven features and plugins to .NET projects

For more information about NPanday, refer to the following guides:

- Features
- FAQs
- Release Notes

36.1.1 User's Guide

- Introduction
- Installation
- Visual Studio Add-In User's Guide
- Maven Command Line User's Guide
- Using with Maven-based Development Infrastructure
- Maven Plugin Reference

36.1.2 Contributing to NPanday

NPanday is an open source project under the Apache License 2.0. You are free to use the application and source code as you wish under those terms.

We welcome you to contribute to the project by submitting your requests and feedback, patches to the code and participating in development discussions in the discussion area.

We are looking for active contributors to join the project, and use the Apache meritocracy model by nominating and voting for committers from those that regularly discuss development and submit patches for fixes and enhancements.

36.1.2.1 Developer's Guide

The NPanday developer's guide contains information on how to checkout and build NPanday for yourself, debug issues, and contribute fixes. It also covers committer's procedures such as the release process and publishing documentation.

36.1.2.2 Reporting Bugs and Requesting Features

You can also help by contributing high quality bug reports and feature requests via the issue tracker, and questions and ideas via the discussion forums:

- issue tracking
- mailing list information

36.2 Documentation PDF

This documentation is also available in a single pdf.

37

37.1 Release Notes for NPanday 1.3-incubating

NPanday .NET Build Tool 1.3-incubating release is now available for download.

The NPanday FAQs provides answers to common questions regarding NPanday .NET Build Tool.

37.1.1 New in NPanday 1.3-incubating

Visual Studio 2010 and .Net 4.0 Framework support

NPanday support for VS 2010 and Net 4.0 Framework.

Removal of UAC and PAB directories

UAC and PAB directories were removed thus making NPanday more maven-like.

37.1.2 Release Notes

37.1.2.1 Changes in NPanday 1.3-incubating

Date staged: 22 January 2011

- NPANDAY-69 NPanday needs to be more synchronized and automated
- NPANDAY-186 Remove the uac and pab directories
- NPANDAY-239 VS Add-in messes up settings.xml
- NPANDAY-288 Support for VS 2010 and .NET Framework 4.0
- NPANDAY-316 Trying to import WPF-project on 64-bit system (VS 2008) throws FileLoadException
- NPANDAY-326 misc improvements
- NPANDAY-328 Support for VS2010 WPF Project
- NPANDAY-329 Support for VS2010 WCF Project
- NPANDAY-330 Support for VS2010 MVC Project
- NPANDAY-333 Dependency of type 'gac' is not resolved correctly
- NPANDAY-334 Update Integration Tests in relation with the UAC & PAB removal enhancements
- NPANDAY-335 NPanday on Visual Studio only starts on Debug Mode
- NPANDAY-336 Add Integration Test for VS2010 supported projects (WPF, WCF, MVC)
- NPANDAY-337 NPanday unusable under linux
- NPANDAY-339 Npanday.Plugin size 0 in repo
- NPANDAY-341 command execution fails if paths with space
- NPANDAY-342 deploy sources artifacts to npanday asset repo
- NPANDAY-343 Can't successfully build projects (VS 2005 and VS 2008) - Build Error due to Access is denied
- NPANDAY-347 compile fails on unix if sources in subfolder
- NPANDAY-348 Import Error when importing WPF projects in VS 2010 on a 64-bit system
- NPANDAY-349 NPanday not building resource files correctly
- NPANDAY-351 NPanday does not respect the configured repository in the pom

- NPANDAY-354 Addin 'resync references' feature fails if using remote repositories but there is no 'mirrors' section in Settings.xml file.
- NPANDAY-355 Addin 'resync references' feature hard-wires "http://repo1.maven.org/maven2' as remote repository causing network security violations.
- NPANDAY-356 Addin 'resync references' feature fails to download latest SNAPSHOT from remote repo if older snapshot file already exists locally.
- NPANDAY-357 Add 'deploy' option to 'Current NPanday Project' menu options
- NPANDAY-360 Local Repository not displaying correct artifacts from the local repository
- NPANDAY-365 Error when deploying documentation for plugins

37.1.3 New in NPanday 1.2

Improved type names available

Improved type names available (dotnet-library, etc.), and previous types are deprecated (library, etc.)

MSI installer

MSI-based installer for the Visual Studio Addin and Maven plugins

MVC project type support

The Visual Studio Addin now creates POMs for project using MVC.

Updated Documentation

Rewrite of documentation to be more complete.

37.1.4 Compatibility Changes

- Most previous project packaging and dependency types have been deprecated in favour of more specific `dotnet-*` alternatives. For more information, see Available Project Types.

37.1.5 Release Notes

37.1.5.1 Changes in NPanday 1.2

Date released: 22 June 2010

- #9011 - Trunk fails to compile when not having VisualStudio
- #9019 - java packages not matches with classes package declaration
- #9622 - error building npanday on linux
- #9627 - Visual Studio Addin Installer
- #10276 - Incompatibilty between NPanday and maven-flex-plugin
- #10388 - Path separator issue in ArtefactInstallerImpl of dotnet-artifact
- #10389 - Specify nunit-console binary for maven-test-plugin
- #10603 - Build with embedded resource files
- #10710 - Docs on the Automation of NPanday Startup
- #10803 - Create Addin Folder during installation if not present
- #10813 - Migrate relevant wiki content to bundled documentation
- #11453 - vsinstalller:install does not generate the AddOn when My Document is not under $ HOME
- #11480 - NPanday should support MVC
- #11524 - Complete integration tests for wix plugin

- #11614 - refresh bundled documentation
- #11615 - migrate integration-tests to npanday-its
- #11635 - Resync Reference unable to download snapshot artifact from Archiva
- #11637 - MSBuild output is not shown and errors are ignored
- #11649 - Add Maven Artifact dialog may not appear if settings can not be read
- #11673 - GAC versions of NPanday.Model.Pom and NPanday.Plugin are not shipped in the 1.1 repository
- #11695 - MSBuild Plugin fails to resolve references in a clean environment
- #11697 - Visual Studio Add-In needs to maintain the includeSources compile plugin configuration
- #11711 - Renaming a web reference would result to adding a new *webreference* node
- #11733 - installing an artifact with a snapshot dependency causes NPE
- #11746 - show the NPanday version in console when it starts up, and/or have an about box
- #11803 - generated ID for remote repository is too long
- #11837 - VS crashes when adding a new remote repository
- #11938 - VS-Installer-Plugin only supports english environments
- #11941 - Doku on Uninstalling NPanday, mscorcfg.msc location
- #11946 - Update Documentation: Install + Build NPanday locally with VS 2008 only
- #11949 - Update Documentation: NUnit-Console must be configured or in Env Path
- #11955 - 1.2-SNAPSHOT referencing 1.1-SNAPSHOT
- #12004 - Unable to recognised GAC dependencies
- #12011 - Deploy plugin deploys artifacts with a classifier with a different snapshot build number
- #12110 - Unable to build VB WebApp projects with a .Net 3.0 Version
- #12120 - Misleading warning message when unable to resolve an SCM URL
- #12231 - Unable to build NPanday projects on Maven 3
- #12239 - NUnit doesn't work on Mono
- #12287 - Overzealous project structure check in project importer
- #12379 - WiX plugin ITs depend on .NET 3.5 SDK
- #12402 - Supporting alternative packaging types
- #12549 - WPF applications can miss including required resources
- #12686 - Remove the deploy plugin
- #12782 - Remove Requirement to use system path for Gac Dependencies
- #12901 - Visual Studio Addin should support *mirrors* in settings.xml
- #12940 - NPanday Fails to Resolve Dependencies if mixed versions of -SNAPSHOT and Released
- #12951 - Document how to add intra solution references
- #13018 - NPanday Transitive Dependency Error
- #13092 - Logger on Resync
- #13199 - compile-plugin calls different goals on deploy for ArtifactType.LIBRARY and ArtifactType.EXE/WINEXE
- #13203 - Completely enable npanday for new dotnet-* types
- #13446 - Include in Installation docs the instructions on how to install NPanday using the installer
- #13449 - vsinstaller-plugin:install fails if the npanday remote repos are not accessible even if artifacts are already in the local repository

- #13541 - unable to successfully install NPanday when running the vsinstaller command
- #13542 - "Optioninfer" compiler option is not recognized by NPanday

37.1.5.2 Changes in NPanday 1.1

Date released: 24 January 2010

- #9272 - Creation of .msi files
- #11579 - NPE when project inherits group ID or version
- #9289 - Check/Fix the Build Process
- #9290 - Check/Fix the Release Process
- #9293 - Integration Tests for NPanday
- #11274 - Documentation for NPanday Plugins
- #10760 - Error Parsing NPanday.Plugin.Msbuild
- #10582 - Problem with Projects with Multiple folder levels in Continuum
- #10675 - In Project import the Resync References should be executed after directory structure support check
- #10654 - SCM tag is not validated correctly when prefixed with "http://"
- #10640 - Add Reference in Visual Studio should behave like Add Maven Artifact
- #11010 - missing NPanday.Plugin.SysRef dependency
- #10985 - Without POM file, double-clicking a DLL in Add Maven Artifact -> Remote tab will crash VS
- #10410 - Cannot read settings.xml using -DSettingsFile
- #10576 - Source distribution for NPanday 1.0.2 includes .svn directories
- #10578 - Source distribution for NPanday 1.0.2 does not match svn tag
- #10577 - Source distribution for NPanday 1.0.2 requires latest WinZip
- #9271 - Support for WCF project type
- #9270 - Support for WPF project type
- #9053 - Generate Solution's POM Information doesn't generate the Company Name in GroupId field
- #9038 - Problem with case of group ids in repository
- #10685 - Misleading options in popup dialog when Adding Maven Artifact without POM
- #10551 - Project Importer incorrectly capitalizes groupId
- #9389 - Configuring Remote Repository successful but artifacts aren't displayed in Remote tab
- #10570 - Fix formatting of includeSource in generated POM
- #10716 - Deleting a Web Reference does not totally updates the pom file
- #10717 - Renaming web reference is successful but the updated pom file is malformed
- #9644 - Build Error when building a newly created project in VS 2008
- #10550 - Attempting to Add Maven Artifact without configuring a remote repo crashes VS 2005
- #10372 - Adding an artifact from a file protocol repository prompts an error.
- #10411 - Adding a artifact that is already in the project prompts an unnecessary warning message "Cannot add ant not an artifact assembly."
- #10376 - NPanday Builder fails to be created if built using Maven 2.1.0
- #10643 - ASPX plugin can copy files to the Bin directory with the wrong filename

37.1.5.3 Changes in NPanday 1.0.2

Date released: 21 August 2009

- #9673 - NPanday copies dependencies to the bin directory on Install
- #9802 - Error when doing a new importing and reference does not exist.
- #9887 - Project Importer does not respect the activeProfile when downloading for artifacts.
- #9931 - Project Importer SCM Option not working with flat projects
- #10000 - Mouse click on SCM tag input box of NPanday Import window clears the input box contents.
- #10011 - NPanday Import; Failed SCM tag add doesn't stop the import process
- #10053 - Repository change in configure repository tab adds duplicate entry to the repository list.
- #10123 - When building .Net 3.0 project duplicate references are reported during mvn install
- #10152 - Directory structure not supported error message is wrongly placed
- #10162 - Project names should be listed in Project Unit Tests window
- #10275 - Error message appears multiple times when selecting an invalid URL in the configure repository tab
- #10273 - Replace 'Select Assemblies' with 'Select Projects' in Project Unit Tests window.
- #10326 - Update NPanday docs for the repository

37.1.5.4 Changes in NPanday 1.0.1

Date released: 12 July 2009

- #8386 - Only the DLL files gets released when releasing a Web Application project type
- #9749 - Versioning of Dll's
- #8962 - Example project for Compiling resources and other file types - Compiling resources and other file types
- #9070 - Build Error on new Project
- #9556 - Problem "Command Too Long"
- #9602 - Tests for projectimporter are failing
- #9645 - Deleting/Removing web references with VB project does not remove the deleted reference in pom file
- #9646 - Detect if there are already existing scm tag in the parent pom.
- #9660 - Build fails if Key File specified

37.1.5.5 Changes in NPanday 1.0

Date released: 30 March 2009

Initial release.