Development manual

Contents

License Information

Download

Git - Version Control System Tool

Description

BUSMASTER is located in a Git repository on the open source hosting platform *https://github.com/*. Git for Windows is the standard git client for Windows. There is also an Windows Explorer extension available called Git Extensions.

There are comprehensive documentation available for any experience grades at GitHub.com:

- *http://help.github.com/*: Start page for GitHub help. This includes a documentation on how to clone the BUSMASTER source codes from GitHub.com.
- *http://help.github.com/win-set-up-git/*: A specific documentation on how to setup Git for Windows.
- *http://pages.github.com/*: Documentation on GitHub project homepages.

The most important steps and the ones necessary for BUSMASTER are described in the following paragraphs. GitHub usually has to repository branches:

- master, which contains the source codes (default checkout)
- gh-pages, which contains the homepage (explicit checkout necessary)

Download

Git for Windows is available at *http://code.google.com/p/msysgit/downloads/list*. Download the most current version of the installer, e.g. `Git-1.7.7-preview20111014.exe` (as of 2011-10-26).

Git Extensions is available at *http://code.google.com/p/gitextensions/*. Download the most current version of the installer, e.g. `GitExtensions225SetupComplete.msi` (as of 2011-10-26).

Installation

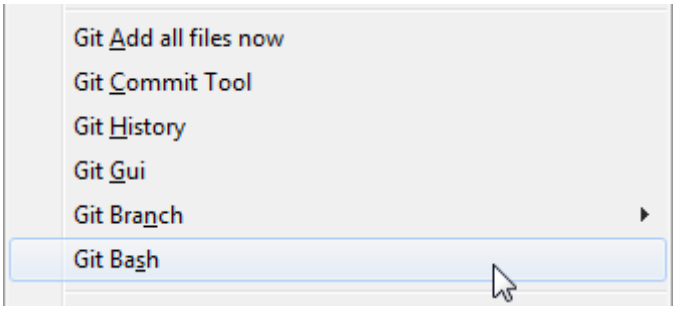
Git for Windows is installed by executing the installer `Git-1.7.7-preview20111014.exe`. The standard installation folder is `%ProgramFiles%\Git`.

Git Extensions is installed by executing the installer `GitExtensions225SetupComplete.msi`. The standard installation folder is `%ProgramFiles%\GitExtensions`.

Configuration

Git requires a configuration regarding your firewall/proxy server settings and regarding your username and e-mail address for commiting patches into the repository.

For configuring Git for Windows, you first have to execute **Git Bash** via the **Windows Start** menu or as shown in the image below via the content menu of the **Windows Explorer**:

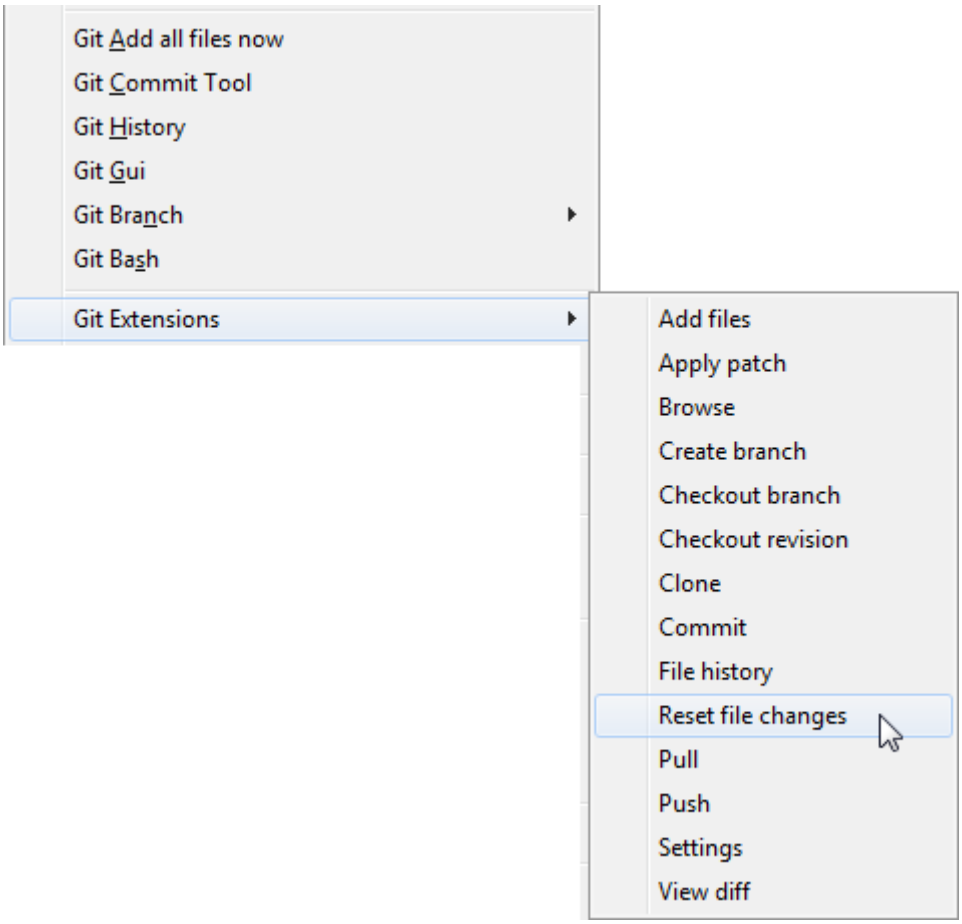


This opens a shell and allows you to execute the usual git command line client. To globally set your proxy, user name and email address do the following and adjust it to your needs:

```
git config --global http.proxy proxy:8080
git config --global user.name "Firstname Lastname"
git config --global user.email "your_email@youremail.com"
```

Usage

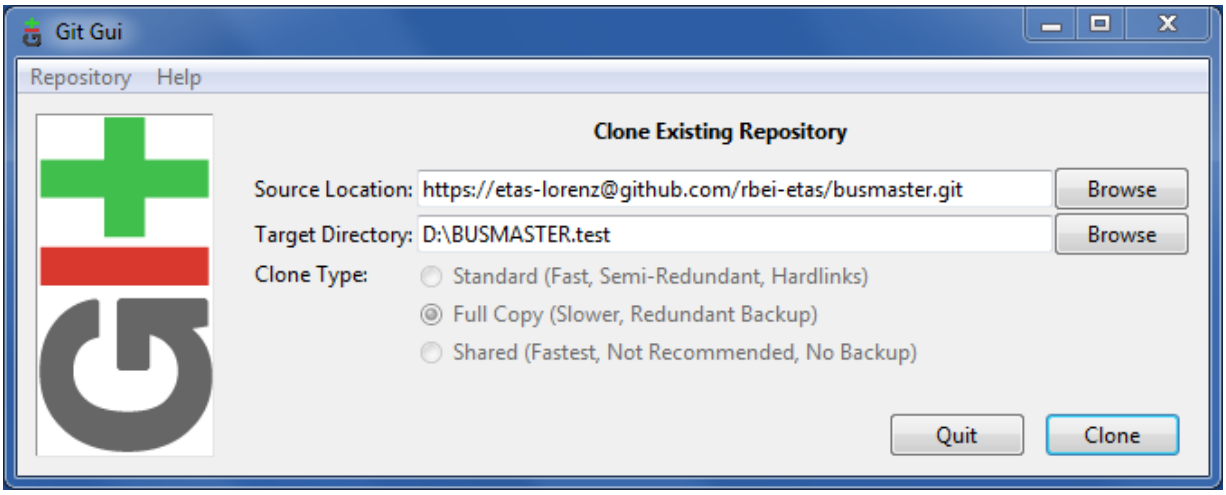
Git for Windows is available via the **Windows Start** menu or as shown in the image below via the content menu of the **Windows Explorer**:



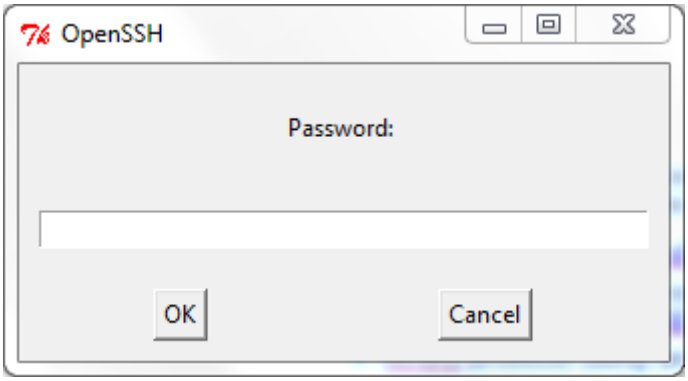
To make an initial checkout execute the **Git Gui** and **Clone Existing Repository**. Select the **Target Directory** according to your needs. The **Source Location** of BUSMASTER is mentioned on the GitHub project page:

- SSH protocol: *git@github.com:rbei-etas/busmaster.git*
- HTTPS protocol using your GitHub account (change accordingly): *https://etas-lorenz@github.com/rbei-etas/busmaster.git*
- HTTPS protocol using anonymous user: *https://github.com/rbei-etas/busmaster.git*

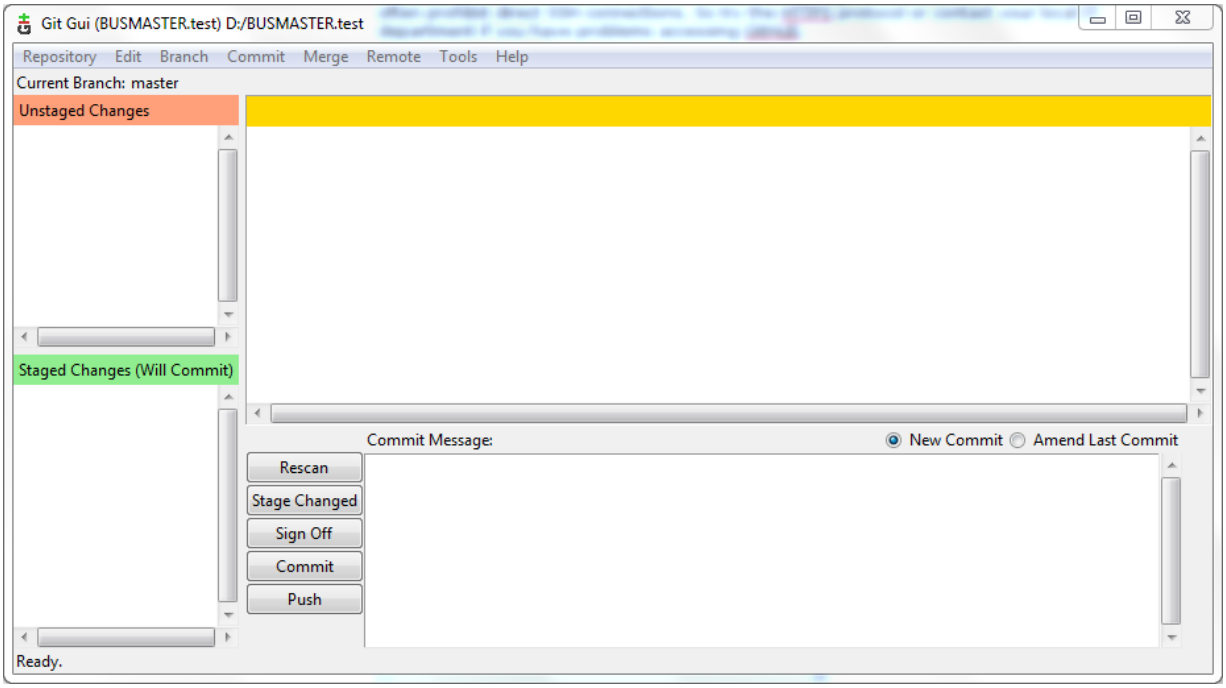
Depending on your firewall and proxy servers not every URL might work. Especially company proxies often prohibit direct SSH connections. So try the HTTPS protocol or contact your local IT department if you have problems accessing GitHub.



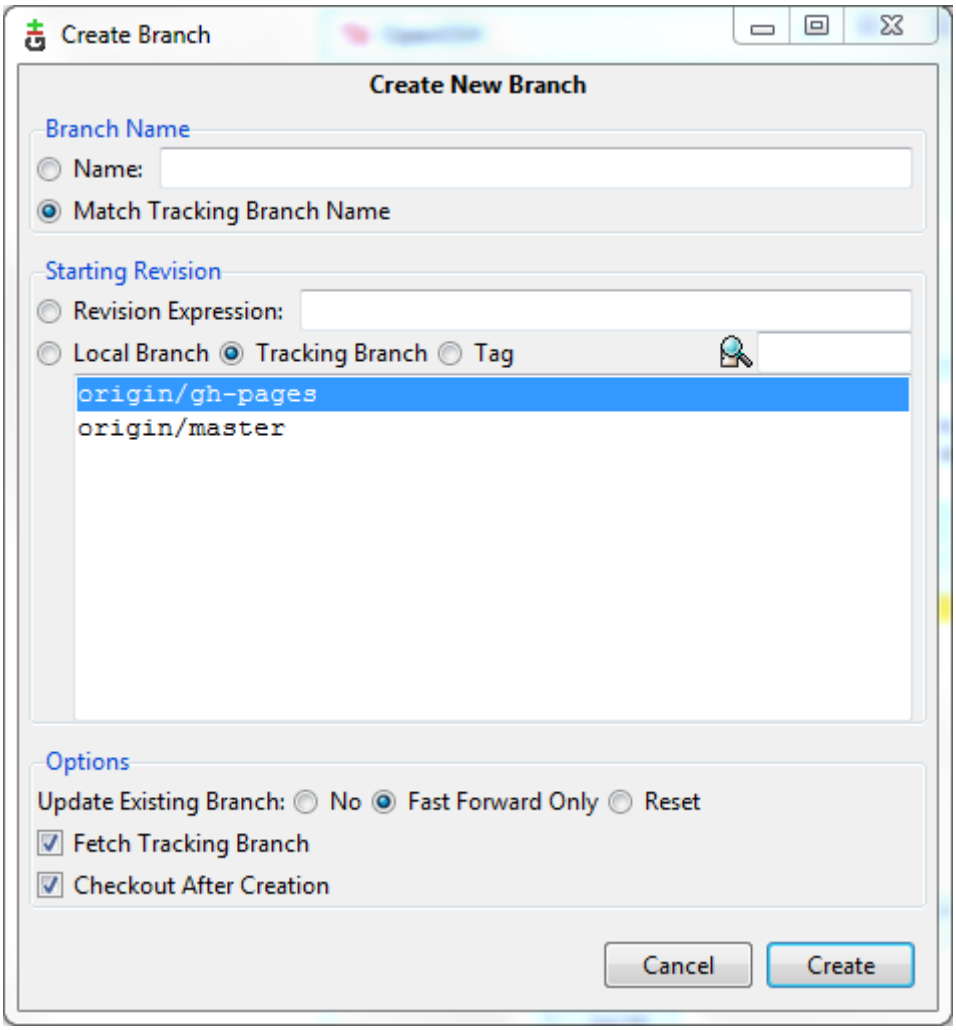
Depending on the URL a window **OpenSSH** will pop up several times and ask you for your GitHub user account password as shown below:



The Target Directory should contain a complete checkout of the master branch. The **Git Gui** starts automatically as shown below:



In case you need to checkout another branch (e.g. from branch to gh-pages) for the first time go to **Branch** > **Create...** A window **Create Branch** should come up. Under **Branch Name** select **Match Tracking Branch Name**. Under **Starting Revision** select **Tracking Branch** and select the respective branch (e.g. origin/gh-pages) as shown below:



Press **Create** to start the checkout from the remote branch (e.g. origin/gh-pages) to the newly created local branch (e.g. gh-pages). Again, depending on the URL the window **OpenSSH** will pop up several times and ask you for your GitHub user account password. Afterwards your Target Directory should contain only the new branch.

The same can be reached via Git Bash if you execute the following command in the Target Directory:

```
git checkout origin/gh-pages -b gh-pages
```

If you open the **Windows Explorer** content menu in the Target Directory you should be able to switch (checkout) between your local branches as shown below:

The Git Gui is pretty helpful. As shown above it consists of four panes:

- **Unstaged Changes**: This shows the modifications you have done since checkout.
- **Staged Changes (Will Commit)**: This shows what you want to upload (commit).
- **Commit Message**: Here you have to inform what your change is about. The first line is a title. All following lines can be used for a detailed description. You should **Sign Off** your message.
- Pane on the upper right side: This shows the differences in the file you have select in one of the left panes.

If you want to keep your checkout up to date, you need to pull from the origin and merge the changes into your local branch. This can be done via **Git Bash** in the Target Directory:

```
git pull origin master
```

The most helpful commands in the beginning can be accessed via **Git Gui**:

- **Commit** > **Stage To Commit** : The file(s) selected under **Unstaged Changes** are selected for the commit under **Staged Changes**.
- Button **Commit**: This will commit your changes into your local branch (e.g. master).
- Button **Push**: This will push your changes from the local branch (e.g. master) to the remote branch (e.g. origin/master).

Build

MS Build - Build Tool

Description

MS Build automates the task to compile the sources codes of the project in the appropriate way.

Download

MS Build is provided as part of the .NET Framework at least in versions 3.5 and 4.0.

Installation

Depending on the version `MSBuild.exe` is installed in the following standard installation folders:

- `%SystemRoot%\Microsoft.NET\Framework\v3.5`
- `%SystemRoot%\Microsoft.NET\Framework\v4.0.30319`

The environment variable `%SystemRoot%` contains the path to your windows folder, e.g. `C:\Windows`.

Configuration

There is no specific configuration of MSBuild necessary. However if your version of .NET and the MS Build tool is different and thus the installation folder, you need to adjust the provided BUSMASTER build script accordingly. Currently it checks for the mentioned versions and folders. The script is located at `Sources\build.bat`.

Usage

Just execute `Sources\build.bat` to build the project in the Release configuration. The output is created in the folder `Sources\BIN\Release`.

If you want to build in Debug configuration, you need to adjust the `Sources\build.bat` accordingly. The output is then created in the folder `Sources\BIN\Debug`.

MS Visual C++ Express 2012 - Compiler and IDE

Description

The MS Visual Studio is a common build environment and IDE on the Windows platform. MS provides cost-free Express editions that are missing the necessary MFC and ATL libraries. This topic mainly describes how to setup a Visual C++ Express 2012 environment and add the necessary MFC and ATL libraries.

With the more feature-rich versions, the compilation should work out-of-the-box.

Download

Go to the Windows Server 2003 driver development kit (DDK) webpage (*DDK 7.1.0 Download Link*), download the DDK ISO file, and burn it to a CD. Most of the time, you can just use the CD burning software that comes with your computer for this task. Alternatively, 7-zip can be used to extract the installation programme from the ISO file.

Installation

Install the DDK, which provides support to compile the MFC and ATL codes.

Configuration

The changes described below will be executed in a form of a patch file and two scripts, one for patching and one for reverting the patch. These files are located at:

- `Documents/1 Development Environment/files/WinDDK_7600.16385.1.diff`
- `Documents/1 Development Environment/files/WinDDK_7600.16385.1_patch.bat`
- `Documents/1 Development Environment/files/WinDDK_7600.16385.1_unpatch.bat`

The patch scripts assume that you have installed WinDDK in the standard installation folder `C:\WinDDK` and that you have installed Git in the standard installation folder `C:\Program Files\Git`. If any of these paths differ you need to modify the scripts before using them.

The following changes in following 3 files listed below are made in `C:\WinDDK\7600.16385.1\inc\atl71` folder:

- `Atlchecked.h`: Add the following definition in this file which is missing:
  ```
  #define AFX_CRT_ERRORCHECK(expr) AtlCrtErrorCheck(expr)
  ```
- `Atlexcept.h`: Replace `AfxThrowOleException(HRESULT)` with `AtlThrow(hr)` as it is not supported.
- `Cstringt.h`: Substancial parts of the file must be replaced to eliminate errors which creep in while using `CString`.

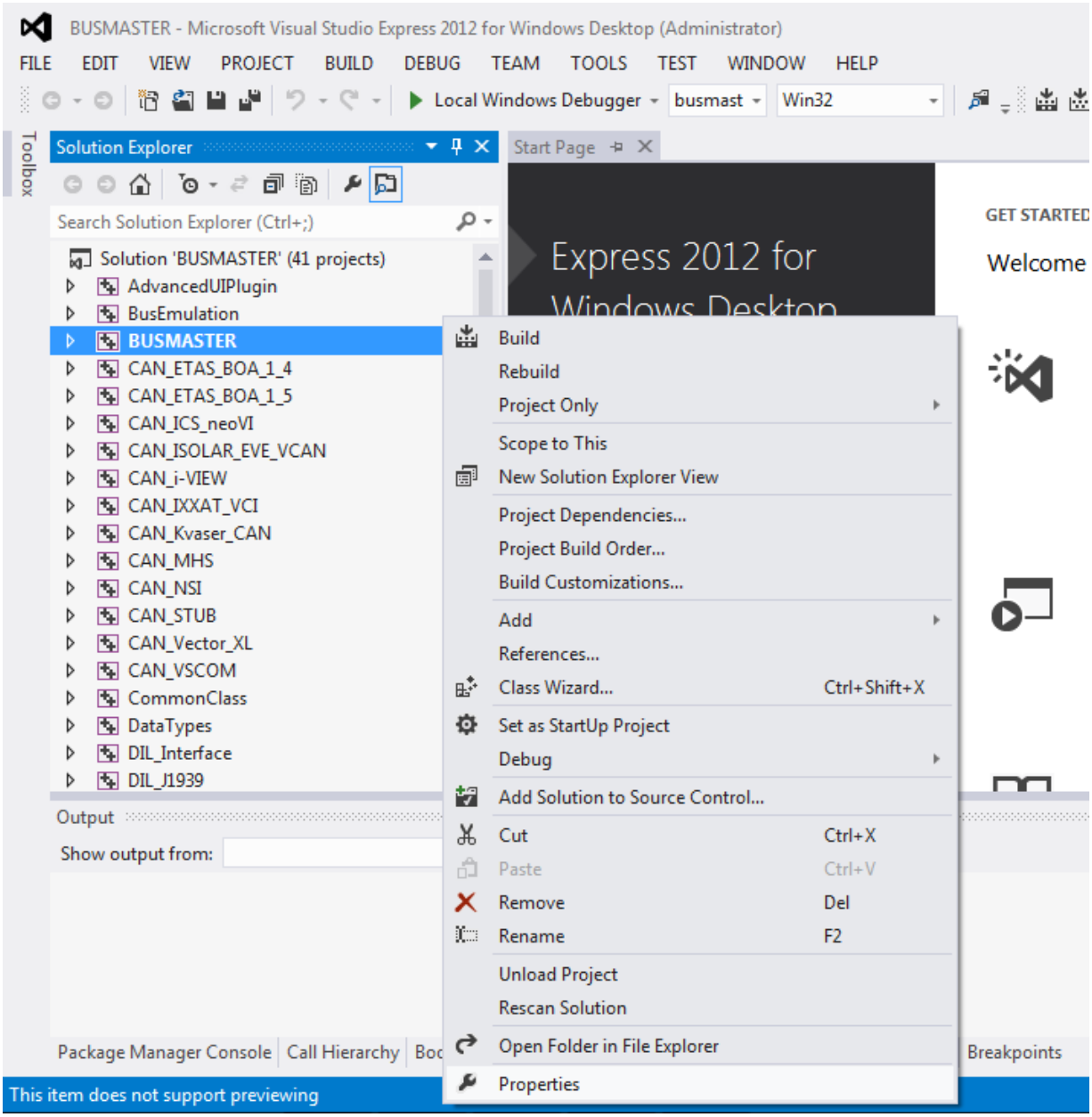
The following changes in following 5 files listed below are made in `C:\WinDDK\7600.16385.1\inc\mfc42` folder:

- `Afx.h`: Comment the function call `::AfxThrowInvalidArgException()` as it is not supported in MFC 6.0 version.
- `Afx.inl`: Add type casting code to avoid warnings for variable *m_timeSpan*. The inline functions to update with this typecasting `(LONG_PTR)m_timeSpan` are `GetDays()`, `GetTotalHours()`, `GetTotalMinutes()` and `GetTotalSeconds()` in `CTimeSpan` class.
- `Afxdisp.h`: Comment the virtual function declarations `GetLicenseKey(...)` and `VerifyLicenseKey(...)` as they are not supported.
- `Atlconv.h`: Substitute the references to the folder `atl30` with `atl71`.
- `Atldef.h`: Substitute the references to the folder `atl30` with `atl71`.

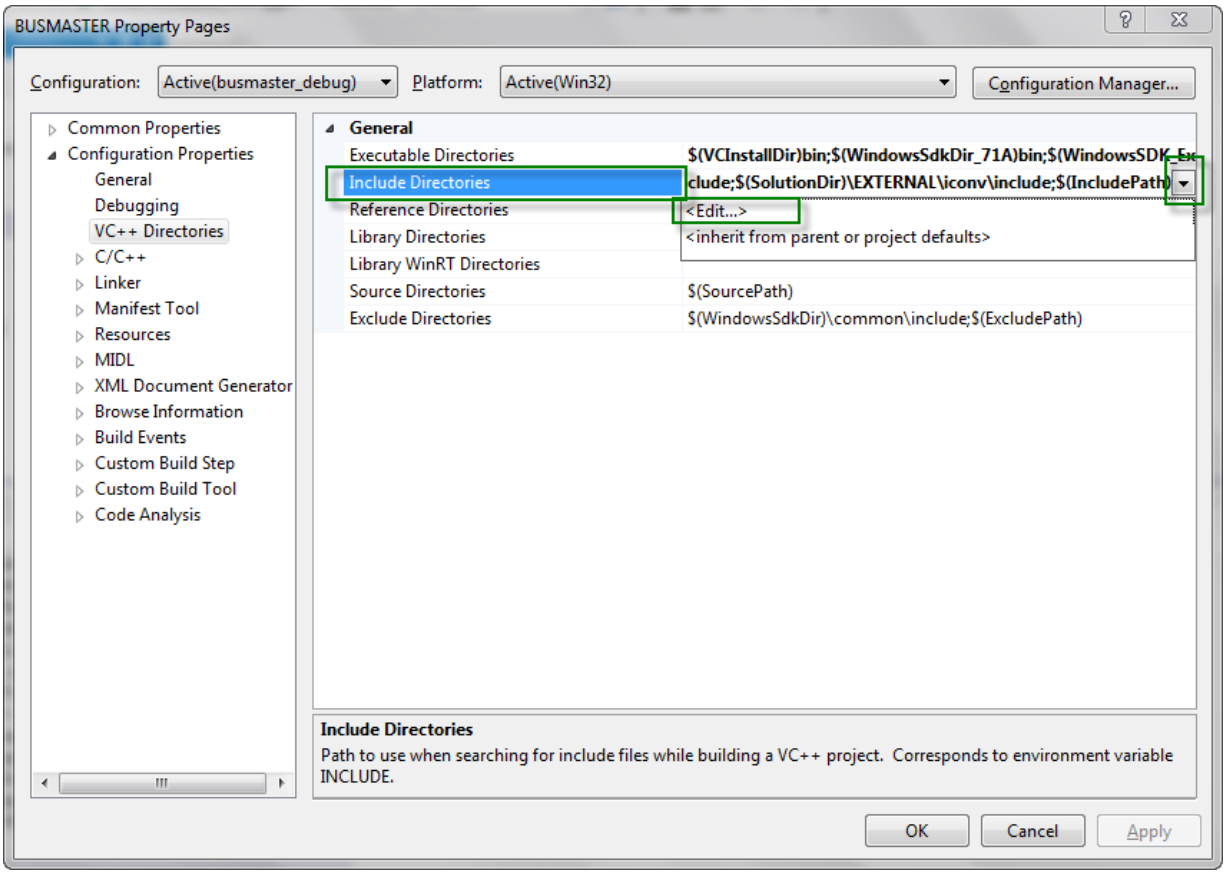
The BUSMASTER project files and the solution file should automatically use WinDDK if it is installed in the standard installation folder. You need to adjust it if necessary.

If the usual registration does not work for you, read on and learn how to register WinDDK manually in your Visual C++ 2008 environment.

A couple of directory paths from WinDDK are need to be added to the project configuration, to instruct Visual C++ where the MFC related files are found. This can be done by selecting the project **Properties** menu item displayed when right clicked on the project in **Solution Explorer** view like shown in the image below. The properties are set by default. Please verify with the paths added. This has to be checked for all the projects in the Solution Explorer.



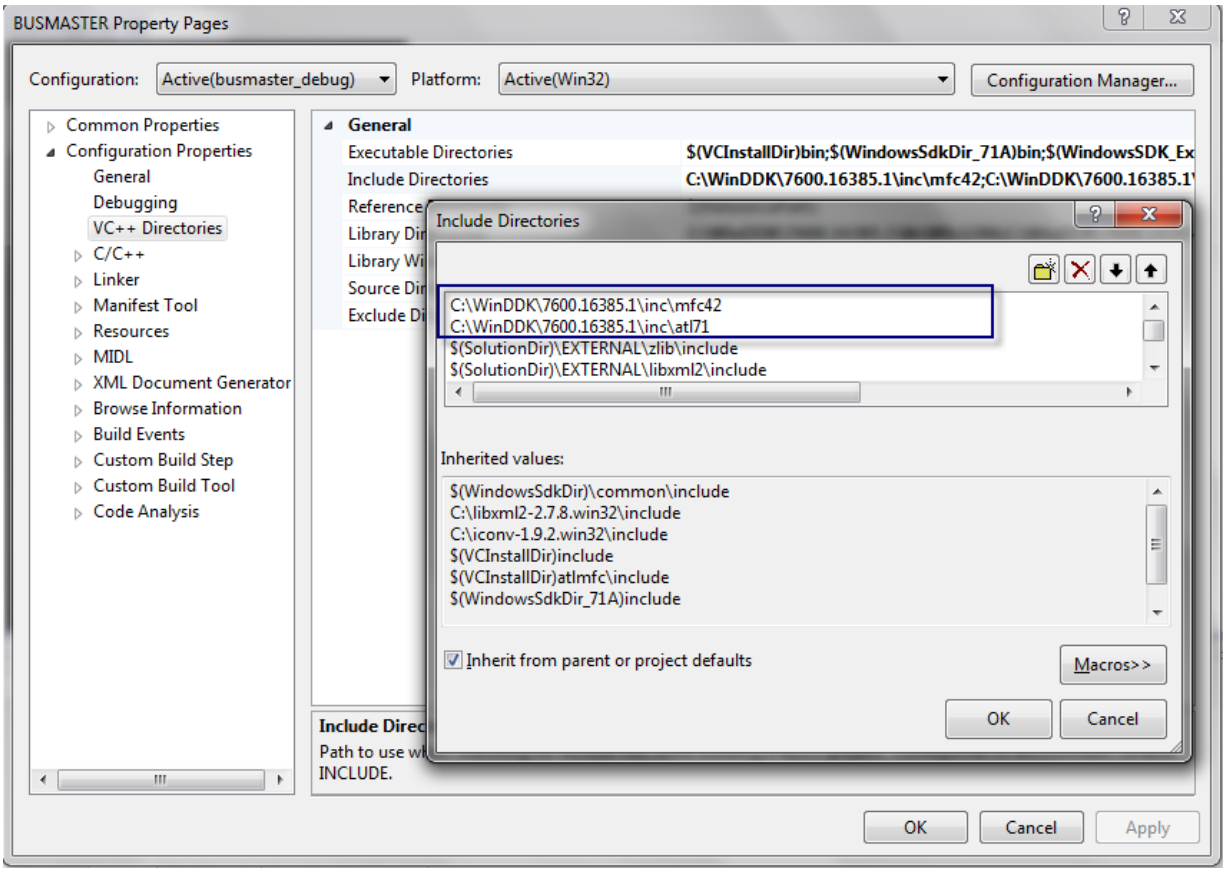
Clicking the menu item **Properties** will display the properties window as shown below. Please select the node **VC++ Directories** under its parent node **Configuration Properties** occurring under the left pane. Please select **Include Directories** entry and then select the drop-down combo box and select **Edit...** as shown below.



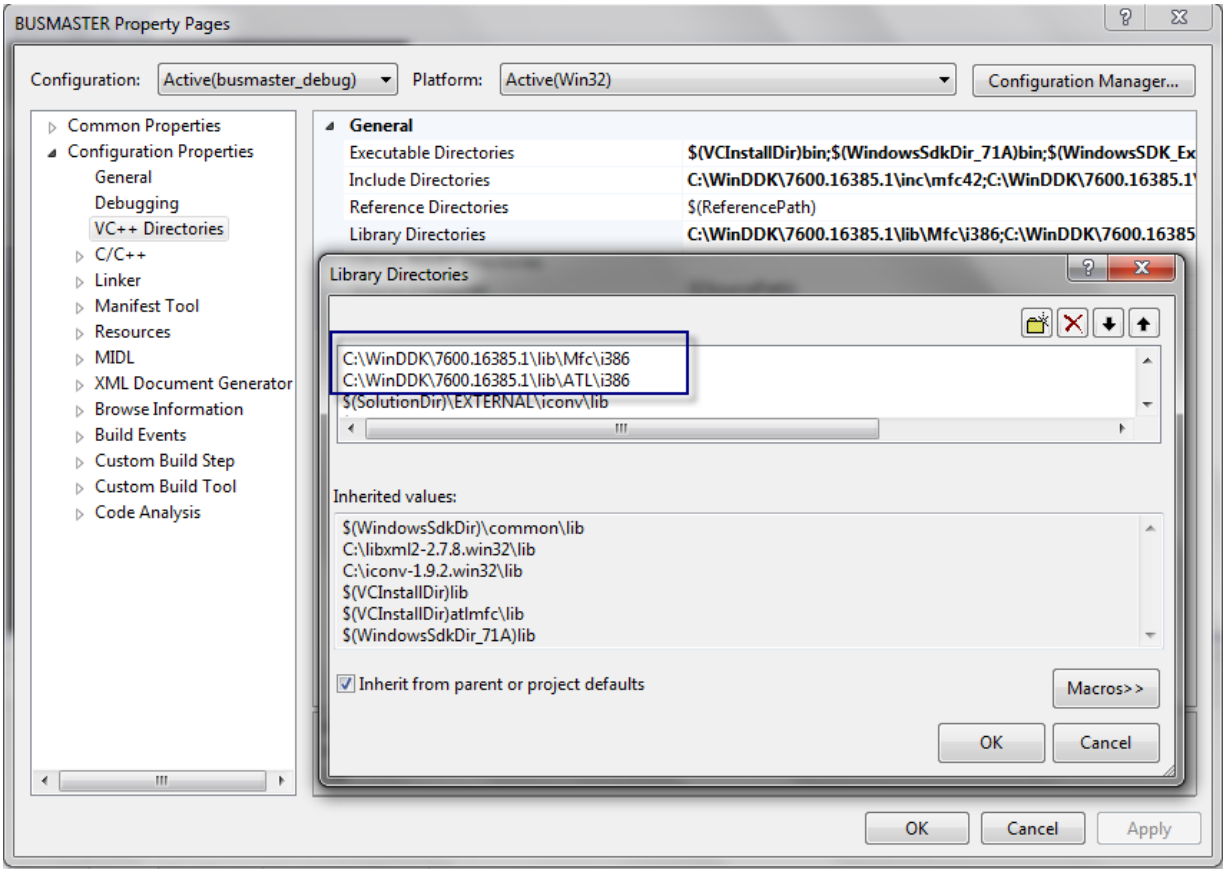
As indicated in the picture below, the following path entries should be listed. If not listed, please add the following entries.

- `$(DDK_directory)\inc\mfc42`
- `$(DDK_directory)\inc\atl71`

`$(DDK_directory)` indicates the directory where DDK has been installed (`C:\WinDDK\7600.16385.1` for example)



Next, similar procedure should be carried out for **Library files** entry as shown below:



As indicated in the picture above, the following path entries should be listed. If not listed, please add the following entries. Please add there:

- `$(DDK_directory)\lib\mfc\i386`
- `$(DDK_directory)\lib\atl\i386`

Usage

It should now be possible to convert and compile an MFC application. The limitations are as follows:

- No plug-in support.
- Resource editing is not possible
- Only up to Visual Studio 6.0 MFC libraries are supported by the DDK.

Project building guidelines:

BUSMASTER project consists of a multitude of sub-projects of various types namely, dynamic link libraries (Win32 DLL, regular DLL using MFC and MFC extension DLL), static link libraries, out-of-proc servers and executables. For obvious reason there are dependencies. For example the static link libraries are the helper libraries employed by all the other modules / components. That's why all the project dependencies must be suitably defined. The following two guidelines must be strictly adhered throughout the development process:

- Ensure every individual sub-project is buildable from a clean state (which means no relevant object files are available prior to the building process).
- The above must hold for both release and debug modes.

Build Configurations

Description

Two configurations are provided so that the users can set the configuration based on their needs

Configurations

Following configurations are available :

- Release Configuration
- busmaster_debug Configuration

Release Configuration

This configuration is provided by the MS Visual Studio IDE itself. In this configuration, the source code debugging cannot be done. This will be used mainly in creating a Release Build and use it for testing. Additionally, the .pdb file generation option is enabled.

Busmaster_Debug Configuration

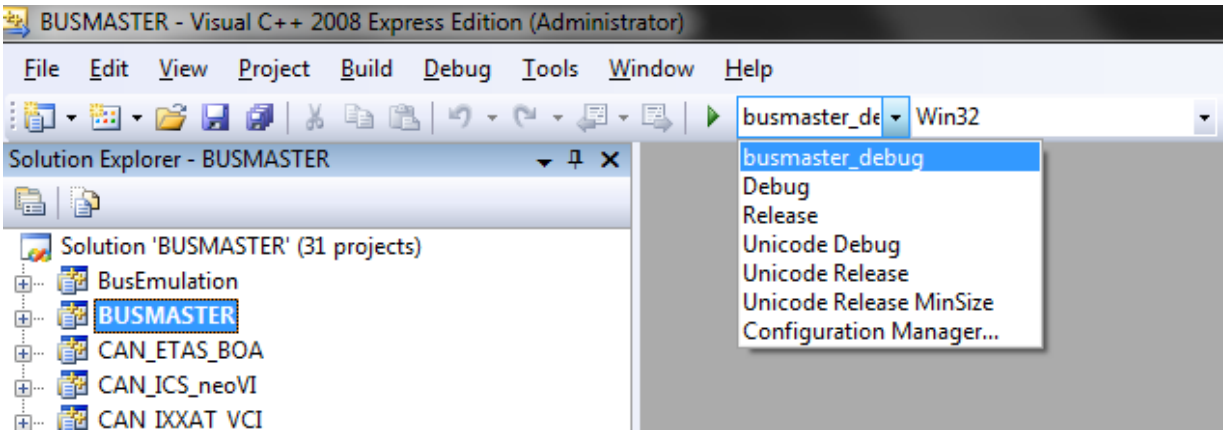
This is a new configuration created and customized to enable the users to debug the source code. This will be used during the development phase to debug the source. Debug format is set to Program Database (/Zi) and .pdb file generation is also enabled. This is not supported for DMGraph and Format Converters.

Debug Configuration

Debug version is not supported as Microsoft don't distribute debug libraries with Express Editions

Unicode Configurations

Unicode Release, Unicode Release MinSize are used with DMGraph project in BUSMASTER solution. Unicode Debug is not supported.



Note: If build fails, select build solution another time without rebuild.

Installation of Runtime Libraries

Description

Describes about the installation of additional libraries required for BUSMASTER compilation.

Support to IXXAT hardware

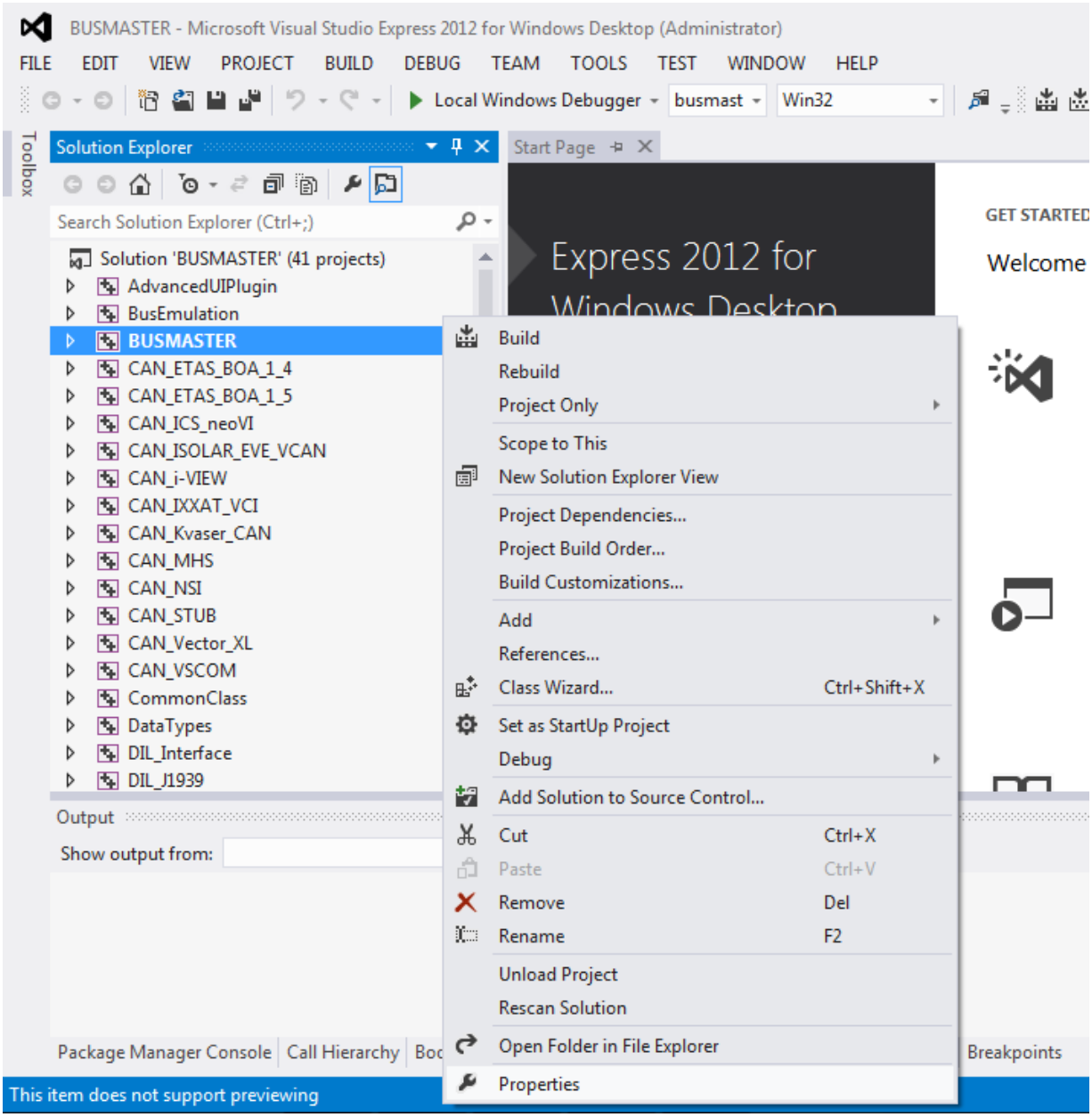
The following steps should be followed to compile CAN_IXXAT_VCI project in BUSMASTER.

- Download and install IXXAT VCI drivers from the following link *http://www.ixxat.com/download_vci_v3_en.html*
- Set Windows environment variable 'IXXAT_VCI_SDK' with the path to the VCI header files (e.g. IXXAT_VCI_SDK = "C:\Program Files\IXXAT\VCI 3.5\sdk\Microsoft_VisualC").
- Restart the computer after performing the above two steps

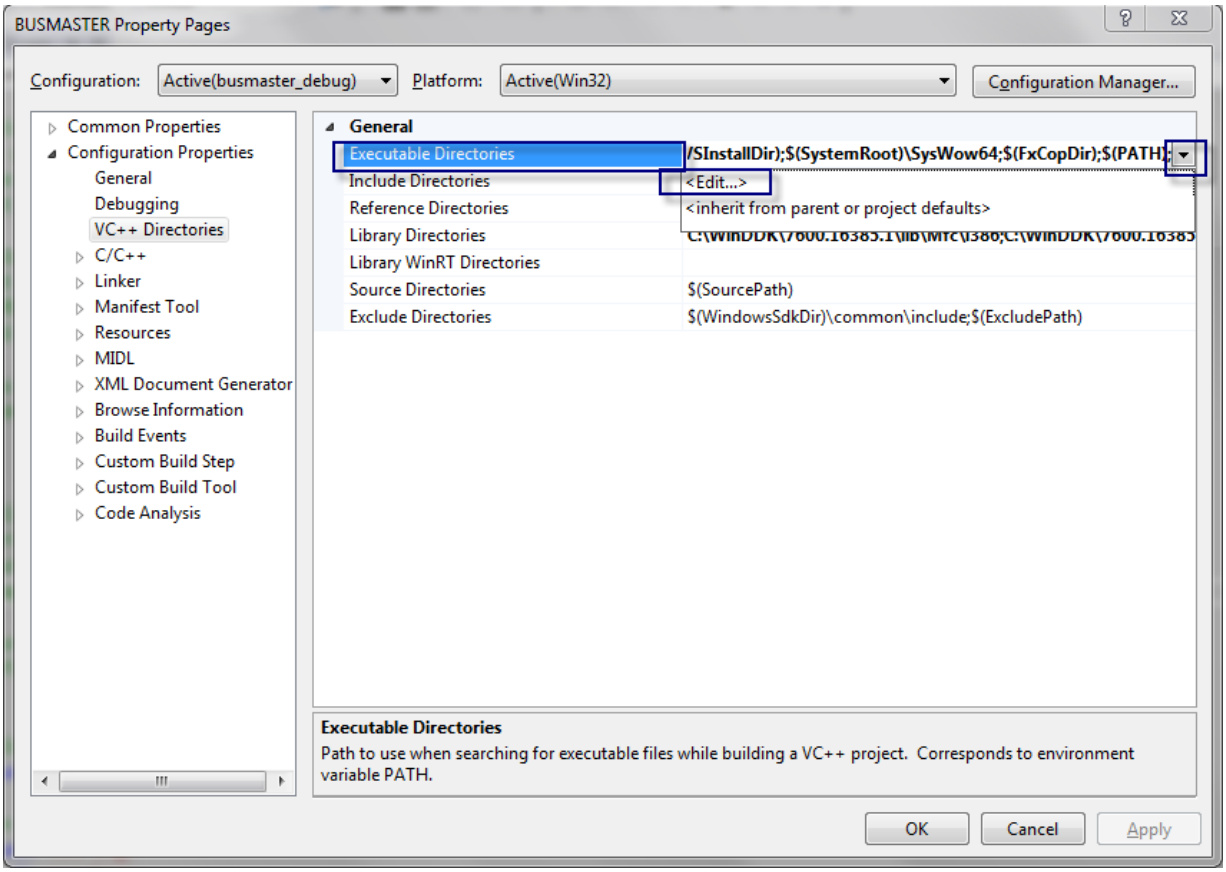
Libxml2 linking support

The following steps should be followed to compile BUSMASTER.

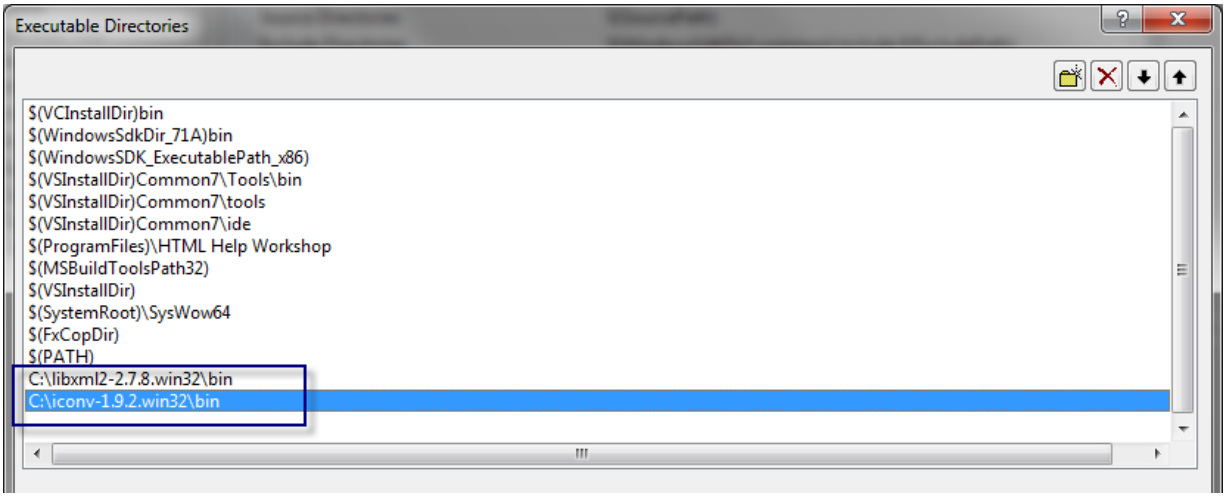
- Download and Copy libxml2-2.7.8.win32, iconv-1.9.2.win32 and zlib-1.2.5.win32 from the following link *ftp://xmlsoft.org/libxml2/win32/* to C: drive
- Add libxml2, iconv and zlib binaries, libraries and include files in VC++ directories in Visual Studio **Properties** page for each project. This can be done by selecting the **Properties** menu item displayed on right click of a project in **Solution Explorer** as shown in the image below:



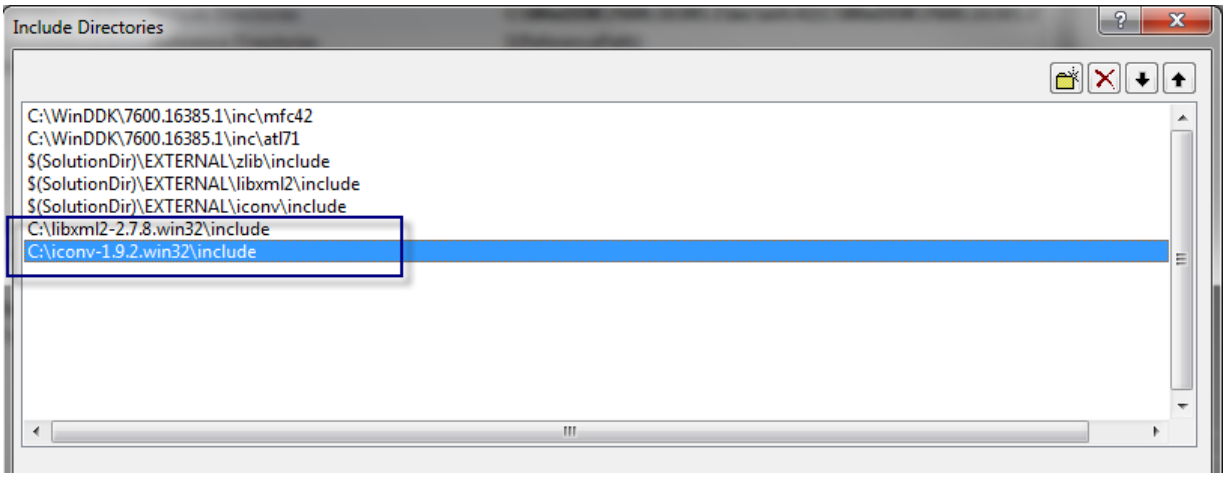
Clicking the menu item **Properties** will display the properties window as shown below:



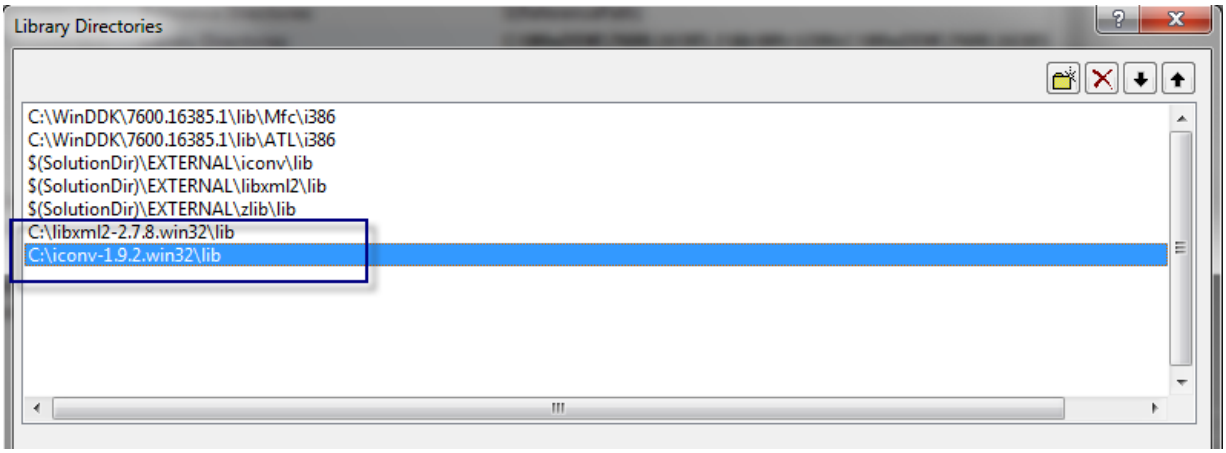
To add executable path for libxml2, select the **Executable Directories** option and select the combo box and click the **Edit...** option. Properties window will be displayed. Add the path for **bin** as shown below:



To add include path for libxml2, select the **Include Directories** option and select the combo box and click the **Edit...** option. Properties window will be displayed. Add the path for **include** as shown below:



To add library path for libxml2, select the **Library Directories** and select the combo box and click the **Edit...** option. Properties window will be displayed. Add the path for **lib** as shown below:



MinGW Installation

The following steps should be followed to install MinGW folder and use it to successfully compile CPP files in Node Simulation using BUSMASTER.

- Download latest mingw executable from the following link *http://sourceforge.net/projects/mingw/files/Installer/mingw-get-inst/* and then use it to download the actual MinGW and copy the MinGW folder to C: drive
- Set the environment variable path ‘C:\MinGW\bin’.

Documentation

DITA-OT - Documentation Publishing Tool

Description

DITA (Darwin Information Typing Architecture) is used as documentation format. It has many beneficial features compared to other documentation formats as the following table shows:

Feature	Word	HTML	Latex	Docbook	DITA
Textual format	No	Yes	Yes	Yes	Yes
Separation of content and style	No	No	Partial	Yes	Yes
XML format and checks	No	No	No	Yes	Yes
Separation of topics and content maps	No	No	No	No	Yes

DITA-OT (Open Toolkit) is used as processing tool.

Serna Free is used as authoring tool.

Download

DITA-OT is available for download at *http://dita-ot.sourceforge.net/*. The package is named `DITA-OT1.5.3_full_easy_install_bin.zip`. The current version (Jun 2011) is 1.5.3.

Installation

DITA-OT requires the installation of the Java Development Kit (JDK). It is not sufficient to have the Java Runtime Environment (JRE) installed. The current version (Oct 2011) is 1.6.0_27. The standard installation folder is `C:\Program Files\Java\jdk1.6.0_27`.

DITA-OT is installed in the old versions 1.3.1 and 1.4 by the authoring tool Serna Free 4.3. They get installed as plugins in the folder `%ProgramFiles%\Syntext\Serna Free 4.3\plugins\dita\DITA-OT1.3.1` and at `%ProgramFiles%\Syntext\Serna Free 4.3\plugins\dita\DITA-OT1.4`.

Following the directory structure of Serna, it's best to install DITA 1.5.3 by un-zipping it to the folder `%ProgramFiles%\Syntext\Serna Free 4.3\plugins\dita\DITA-OT1.5.3`.

Configuration

The is no special configuration of DITA-OT necessary.

Usage

BUSMASTER comes with a convenient `build.bat` script in the following folders:

- `Documents\1 Development Environment`
- `Documents\2 Requirement Specification`

- `Documents\3 Design Document`
- `Documents\4 Help`

If you change the JDK standard installation folder or the DITA-OT standard installation folder, then you also have to adjust the `build.bat` files in each of the mentioned `Documents` folders accordingly.

Doxygen - Source Code Documentation

Description

Doxygen is a source code documentation tool. It uses special formatted comments in the source code to generate a complete documentation in several different formats. In BUSMASTER only HTML output is generated.

The doxygen manual is available at: *http://www.testcocoon.org/coveragescanner.html#toc77*.

Download

Doxygen is available at *http://www.stack.nl/~dimitri/doxygen/*. In the blue box on the right side is a "Download" link. Download the latest installer available, e.g. `doxygen-1.7.5.1-setup.exe` (2011-10-25).

Installation

Execute `doxygen-1.7.5.1-setup.exe` to install the application. The standard installation folder is `%ProgramFiles%\doxygen`. It will also be added to the %PATH% environment variable.

Configuration

Doxygen requires no specific configuration beside the files provided in the `BUSMASTER\Documents\5 Source Code Documentation\Doxyfiles\*`.

It is helpful to add Doxygen to the list of external tools in the **Tools** menu within the Visual C++ IDE.

For BUSMASTER we defined two specific tags (see below for examples):

- `copyright`: This contains a copyright note
- `req`: This contains a requirement tag and a brief tag description. The requirement tag must be character-identical to the tags in the requirement specification.

Usage

BUSMASTER comes with a `build.bat` script that automatically calls Doxygen in the right way. However if doxygen is not in your `%PATH%` environment variable you need to adjust the script accordingly. After running Doxygen the output for each module is placed in the folder `Documents\5 Source Code Documentation\out`. An `index.html` is provided as convenient entry point to the generated documentations.

The preferred Doxygen file header contains:

```
/**
 * \file      <filename>
 * \brief     <brief description>
 * \authors   <author>                    [if there is only one author]
 * \authors   <author 1>, <author 2>, ... [if there are multiple authors]
 * \copyright Copyright (c) 2011, Robert Bosch Engineering
 *            and Business Solutions. All rights reserved.
 *
 * <long description>
 */
```

The preferred Doxygen function header contains:

```
/**
 * \brief         <brief description>
 * \req           <req tag> - <brief tag description>
 * \param[in]     <param name> <description> [if that is inbound]
 * \param[out]    <param name> <description> [if that is outbound]
```

```
 * \param[in,out] <param name> <description> [if that is bidirectional]
 * \param         <param name> <description> [if direction is unspecified]
 * \return        <description>
 *
 * <long description>
 */
<return type> foobar(<param type> <param name>, ...)
```

The shortest version in case you have no brief description, no parameters (`void`) or no return (`void`) contains just:

```
/**
 * <long description>
 */
void foobar(void)
```

Detailed example:

```
/**
 * \brief     Converter from foo to bar
 * \req       RS_12_34 - Converter from foo to bar
 * \param[in] fooFile Input file in foo format
 * \param[in] barFile Output file in bar format
 * \return    Result code
 *
 * This function converts the contents of fooFile in foo format to barFile
 * in bar format. fooFile and barFile need to be opened.
 */
int converter_foo2bar(fstream& fooFile, fstream& barFile)
{
    /* do something */
}
```

Serna - Documentation Authoring Tool

Description

DITA (Darwin Information Typing Architecture) is used as documentation format.

DITA-OT (Open Toolkit) is used as processing tool.

Serna Free is used as authoring tool.

Download

Serna is available for download at *http://www.syntext.com/products/serna-free/*. The current version (Jun 2011) is 4.3. Just download the file `serna-free-4.3.0-20110207.exe`.

Installation

Execute the Serna installer `serna-free-4.3.0-20110207.exe`. The standard installation folder is `%ProgramFiles%\Syntext\Serna Free 4.3\`.

Configuration

The is no specific configuration of Serna Free necessary. Serna is automatically associated with the file types `.xml`, `.dita` and `.ditamap`.

Usage

Just execute Serna by double clicking any of the `.dita` or `.ditamap` files located under one of the following folders or subfolders:

- `Documents\1 Development Environment`
- `Documents\2 Requirement Specification`
- `Documents\3 Design Document`

- `Documents\4 Help`

Tests

Artistic Style - Automatic Formatter Tool

Description

AStyle is an automatic source code formatter/beautifier tool. It can be adjusted to create an output consistent throughout the project files.

Download

AStyle is available for download at *http://astyle.sourceforge.net/*. Download the most current version, e.g. `AStyle_2.02_windows.zip` (as of 2011-10-25).

Installation

Extract the `AStyle_2.02_windows.zip` file to `%ProgramFiles%\AStyle`. It will create the following folders:

- `bin`: Contains the program `AStyle.exe`.
- `build` : Contains files to build the source code of AStyle. This is not necessary for BUSMASTER.
- `doc`: Contains the documentation, which could be helpful for the usage.
- `src`: Contains the source code of AStyle. This is not necessary for BUSMASTER.

Only the program `AStyle.exe` is necessary for the usage with BUSMASTER. It's recommended to add the `bin` folder to your %PATH% environmental variable.

Configuration

For BUSMASTER certain AStyle options have been selected based on the ETAS profile:

```
# Bracket Style Options
--style=ansi

# Tab Options

# Indentation Options
--indent-switches
#--indent-namespaces# deleted from ETAS profile
--indent-preprocessor
--indent-col1-comments# added to BUSMASTER profile

# Padding Options
--break-blocks
--delete-empty-lines

# Formatting Options
--add-brackets
--convert-tabs
--align-pointer=type

# Other Options
```

Options can either be given on the command line or via an options file.

AStyle is also used as part of the static code test bench. The options file with the contents given above is located at `Tests/AStyle/astyle.txt`.

Usage

Go to your command line (e.g. by executing `cmd` in the start menu) and type (path to options file need to be adapted):

```
astyle --options=astyle.txt --recursive *.cpp *.h
```

This will recursively run over all source code files of BUSMASTER and will adjust the style as necessary. The modifications will be directly done in the respective file. The original unmodified files are available with the file name extension `.orig`.

Before you commit these file to the project repository, make sure that the modifications are useful or adjust them if necessary.

AutoIt - GUI Automation and Testing Tool

Description

AutoIt v3 is a freeware BASIC-like scripting language designed for automating the Windows GUI and general scripting.

It uses a combination of simulated keystrokes, mouse movement and window/control manipulation in order to automate tasks.

AutoIt is also very small, self-contained and will run on all versions of Windows out-of-the-box with no annoying "runtimes" required!

Features (as described on their homepage):

- Easy to learn BASIC-like syntax
- Simulate keystrokes and mouse movements
- Manipulate windows and processes
- Interact with all standard windows controls
- Scripts can be compiled into standalone executables
- Create Graphical User Interfaces (GUIs)
- COM support
- Regular expressions
- Directly call external DLL and Windows API functions
- Scriptable RunAs functions
- Detailed help file and large community-based support forums
- Compatible with Windows 95 / 98 / ME / NT4 / 2000 / XP / 2003 / Vista / 2008
- Unicode and x64 support
- Digitally signed for peace of mind
- Works with Windows Vista's User Account Control (UAC)

AutoIt has been designed to be as small as possible and stand-alone with no external .dll files or registry entries required making it safe to use on Servers.

Scripts can be compiled into stand-alone executables with Aut2Exe.

A manual is available at *http://www.autoitscript.com/autoit3/docs/*.

Download

AutoIt is available for download at *http://www.autoitscript.com/site/autoit/downloads/*. Download the installed `autoit-v3-setup.exe` (2011-10-25).

Installation

Just execute the installer `autoit-v3-setup.exe`. The standard installation folder is `%ProgramFiles%\AutoIt3`.

Configuration

The is no specific configuration necessary.

Usage

BUSMASTER provides test script in the AutoIt format in the folder `Tests\AutoIt`. Their is a sub folder for each test group, e.g. `Tests\AutoIt\Installer`. The central `build.bat` script executes all tests and summarizes failures in an HTML report.

If you intend to develop further AutoIt test scripts, please follow the following general guidelines to make the usage of the AutoIt tool for our application testing more maintainable:

- Test cases have the file extension `.au3`. The base name of the script and all necessary files should be identical and reflect the name of the test.
- Test cases should run independently from each other. Thus a tester is able to execute specific individual test cases.
- Test cases should run on different system, e.g. different working directory, language environments. Thus don't use absolute paths in AutoIt scripts. Instead use the AutoIt global variables, e.g. `@ScriptDir`, `@ScriptName`, `@ProgramFilesDir`.
- Files that get generated during the test execution should be placed in `out` folders, e.g. `Tests\AutoIt \Format Converter\out`. This makes it easier for the central `.gitignore` file to keep such results out of the source code management system.
- Informations during the test run should be reported to the `stdout`. Use the `ConsoleWrite` command for it.
- Errors during the test run should be reported to `stderr`. Use the `ConsoleWriteError` command for it. If anything goes to `stderr` the test case is assumed as failed.
- Try to avoid using absolute paths in AutoIt scripts as much as possible. Instead use the AutoIt global variables to determine path information.
- Try to avoid runtime specific behavior. Thus try to avoid using the `Sleep` command, but try to use the `WinWaitActive` command instead.
- Add your test case to the central `Tests\AutoIt\build.bat` script, if you feel confident that all aforementioned points are fulfilled.

The testing with AutoIt automation can be divided into two varieties:

- Fully Automatic testing requires no manual interaction and is based on the Simulation interface.
- Semi Automatic System Testing: Involves some manual procedures (e.g. attaching hardware devices) mentioned in related guideline.

Cppcheck - Static Source Code Checker

Description

There are many static source code checkers available and used by Open Source projects.

Cppcheck is another Open Source software, which is able to parse C++ code and is therefore used for BUSMASTER.

Download

Cppcheck is available under: *http://cppcheck.sourceforge.net/*. Just download the most current version, e.g. `cppcheck-1.50-x86-Setup.msi` (as of 2011-10-25).

Installation

Execute the installer. The standard installation folder is `%ProgramFiles%\Cppcheck`. Cppcheck should automatically be added to your %PATH% environment variable.

Configuration

The test script in `Tests\Cppcheck` contains all necessary configuration. Beside that no specific configuration necessary.

Usage

Execute the test script in `Tests\Cppcheck\build.bat`. It automatically generates an HTML report with the results.

PMD - Code Redundancy Checker

Description

PMD's Copy/Paste Detector (CPD) is used to check for redundant code fragments. It requires the Java Runtime Environment and is able to parse at least Java, C and C++ codes.

Download

PMD is available for download at *http://pmd.sourceforge.net/cpd.html*. The current version (Mar 2012) is 5.0 Alpha. Just download the file `pmd-bin-5.0-alpha.zip`.

Installation

Extract the archive `pmd-bin-5.0-alpha.zip` in the recommended installation folder `%ProgramFiles%\PMD\`.

Configuration

The configuration is contained in the test script `Tests\PMD\build.bat`. Beside that there is no specific configuration of PMD necessary.

Usage

The test script `Tests\PMD\build.bat` automatically rund on the `Sources` folder and generates HTML report files.

TestCocoon - Source Code Coverage Tool

Description

TestCocoon is a plugin for certain compilers including Visual C++. It evaluates covered source code lines upon execution of the BUSMASTER application.

In combination with the test automation tool AutoIt, it reports the corresponding test coverage.

Download

TestCocoon is available for download at *http://sourceforge.net/projects/testcocoon/*. Download the latest version of the installer, e.g. `TestCocoonSetup_1_6_14_x86.exe` (as of 2011-10-25).

Installation

Execute the installer `TestCocoonSetup_1_6_14_x86.exe`. The standard installation folder is `%ProgramFiles%\TestCocoon`.

Configuration

To integrate TestCocoon into the Visual C++ IDE, use the following documentation: *http://www.testcocoon.org/coveragescanner.html#toc77*

Usage

TestCocoon generates for each module an instrumentation file with the extension `.csmes`.

After executing BUSMASTER, an execution report is available in a file with the extension `.csexe`.

TestCoverage comes with a tool to visualize the corresponding source code coverage. It is called CoverageBrowser and is available in the Windows start menu under **TestCocoon** > **CoverageBrowser** . Use this tool to open the `.csexe` files.

Installation

NullSoft Scritable Install System - Installation Tool

Description

NullSoft Scriptable Install System (NSIS) is the *software installation framework* used. It is a script-driven Windows installation system. The following links may be accessed for details:

Download

The NSIS installer is available for download at *http://nsis.sourceforge.net/*. The latest version is 2.46, dated December 06, 2009.

Installation

Just run the `nsis-2.46-setup.exe`. The standard installation folder is `%ProgramFiles%\NSIS`.

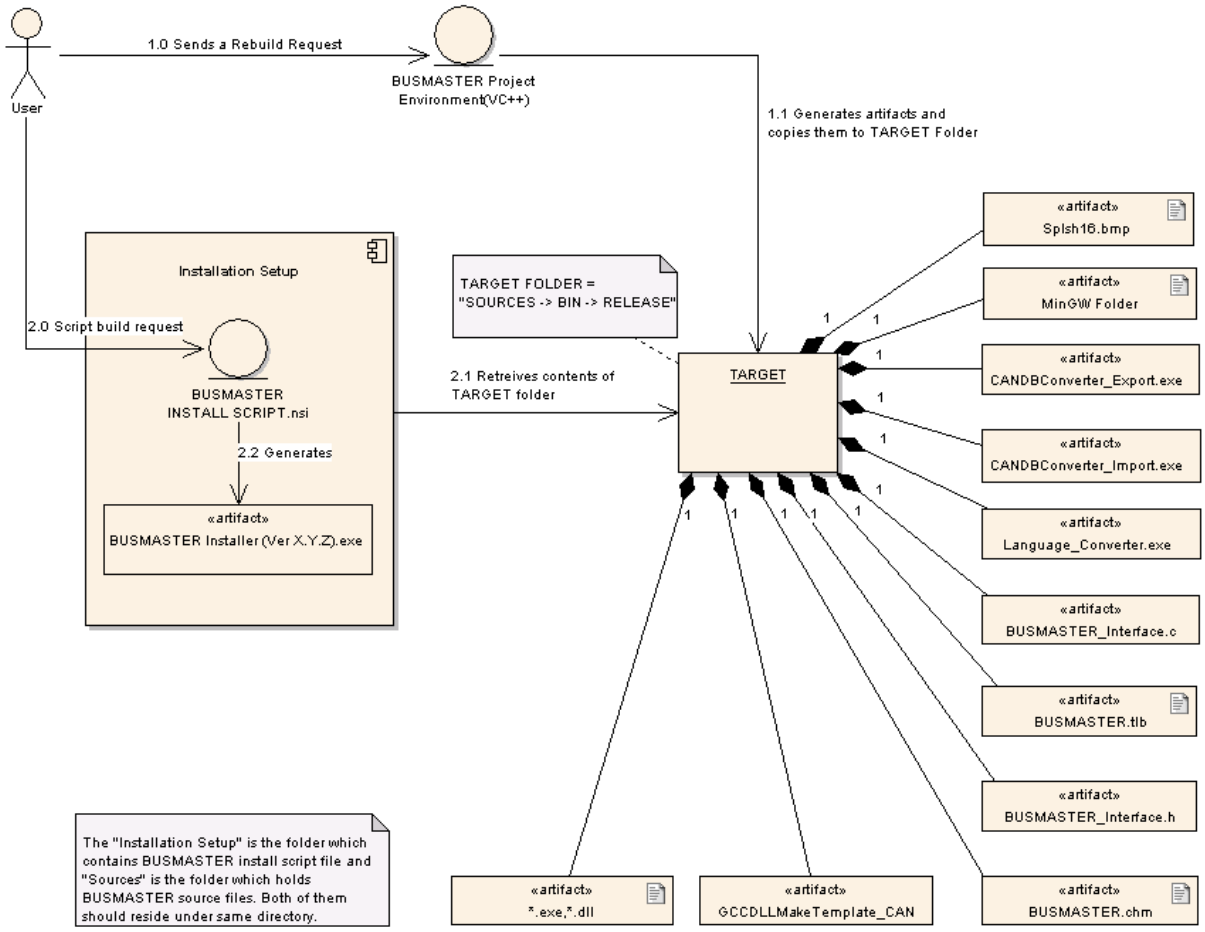
Configuration

There is no specific configuration necessary.

Usage

The BUSMASTER install script is located under `Installation Setup\BUSMASTER Install Script.nsi`.

The setup is depicted in the diagram below which shows how the installer build generation process is initiated by User and how the script accesses the artifacts from various locations under the `Sources` and `Documents\4 Help` folder. Prior to building the script, the artifacts needs to be generated by building the BUSMASTER application in Visual studio environment (VC++) and DITA-OT.



Dependency

Install script expects Visual Studio 2012 Redistributable package to be available under the folder Tools\VC++ 2012 Redistributable\vcredist_x86.exe

Continuous Integration Server

Jenkins - Continuous Integration Server

Description

Continuous Integration environments fulfills the following tasks:

- Automatic building (incl. download of source code)
- Automatic testing (incl. upload of test results)
- Automatic delivery (incl. upload of installer, e.g. in form of nightly builds)

There are many continuous integration systems available and used by Open Source projects.

Jenkins is nowadays the most advanced and most often used continuous integration system. It only requires a JDK installation on the target system to run. If the jobs within the system require tools they need to be installed too, e.g. Visual C++.

As Jenkins is written in Java it runs on any platform, e.g. Windows and Linux.

We've choosen Jenkins for BUSMASTER due to the aforementioned arguments.

Download

Jenkins is available under: *http://jenkins-ci.org*. Just download the most current version, e.g. version 1.457 (as of 2012-04-17).

Installation

Jenkins only need to be installed on the master node. The slaves can be started via Java Web Start.

On the master start the installer. The standard installation folder is `%ProgramFiles%\Jenkins`.

Configuration

All BUSMASTER components incl. builds, tests and the installer can be executed with a `build.bat` script. Create a Jenkins job for any of these scripts and bring them in dependency.

Usage

The Jenkins server is accessable on port 8080 of the server it is running on.

Jenkins can be usually configured to automatically poll a source code management system or a timely manner. It can also get automatically triggered by a platform like *http://github.com*. In this case this needs to be configured on the platform.