

# How to Generate Build to Build Change History using Visual Build Professional and Team Foundation Server

---

## Introduction

I was recently asked to include build by build change history in our build notification messages by a Senior Developer and the QA Supervisor. Some things to note ahead of time is that the company I work for is on the small side and we do not have a Configuration Manager so everyone shares the responsibility in Development and Quality Assurance. My company boasts around 30 separate applications written in a variety of languages. The source control tree looks somewhat like a tumbleweed due to dependencies between application common components.

## Checklist of steps used

1. Query for and store the current Date and Time in Visual Build Pro.
2. Split the current Date and Time into string format that is usable in TFS.
  - a. I accomplished this using VBP's VBScripting.
3. Query history on TFS using the last known build date and time and the current time.
4. Once a build succeeds, update the last known build date and save the VBP project.

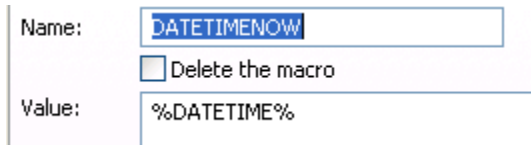
## Step 1: Define and Split Current Date and Time

To start with, we need to obtain the current Date and Time. This will be used later if the build succeeds to update the last known build time. It is important to get it before pulling down all of the source code so that we do not miss any history. It would be better for this report to have more information than needed rather than missing information.

1. Inside of Visual Build, add a "Set Macro" object and name it something appropriate on the General tab (ie: Define DATETIMENOW).
2. Define it as a Temporary Macro from the Macro tab.



3. In the Value field, enter %DATETIME%.
  - a. This is an internal macro that will return the current date and time.
4. Name your temporary macro (ie: DATETIMENOW).



5. Click OK.
6. Next, add a "Run Script" object and name it something appropriate on the General tab (ie: Format DATETIMENOW for Team Foundation).
7. Input the script below on the Script tab (Make sure to select VBScript from the dropdown).
  - a. Read below for a breakdown on what this script is doing.

```
DATETIMENOW = Application.ExpandMacros ("%DATETIMENOW%")
```

```
sYEAR = vbld_FormatDateEx (DATETIMENOW, "YYYY")  
sMONTH = vbld_FormatDateEx (DATETIMENOW, "mm")  
sDAY = vbld_FormatDateEx (DATETIMENOW, "dd")  
sHOOR = vbld_FormatDateEx (DATETIMENOW, "hh")  
sMINUTE = vbld_FormatDateEx (DATETIMENOW, "MM")
```

```
HISTORYSTRING = "D" & sYEAR & "-" & sMONTH & "-" & sDAY & "T" & sHOOR  
& ":" & sMINUTE & "~T"
```

```
Set objMacro = vbld_TempMacros.Add ("HISTORYSTRING", HISTORYSTRING)
```

8. Click OK.
9. You should end up with the following 2 items.



The VBScript above does the following:

1. `DATETIMENOW = Application.ExpandMacros ("%DATETIMENOW%")`
  - a. `Application.ExpandMacros` takes the value you assigned in `DATETIMENOW` and applies it to a variable inside of the script. In this case, I used the same variable name. Note that macro names need to be enclosed in % signs and VBScript variables names do not when in the VBScript editor.
2. `sYEAR = vbld_FormatDateEx(DATETIMENOW, "yyyy")`
  - a. `vbld_FormatDateEx` returns a formatted date string based on a date and time string in the first parameter along with a format string in the second parameter.
3. `sMONTH`, `sDAY`, `sHOUR`, `sMINUTE` all do the same as `sYEAR` except for their respective date/time elements.
4. `HISTORYSTRING = "D" & sYEAR & "-" & sMONTH & "-" & sDAY & "T" & sHOUR & ":" & sMINUTE & "~T"`
  - a. `HISTORYSTRING` is the formatted date/time string required for TFS to retrieve the desired history.
  - b. When supplying a Date element, you precede it with a D
  - c. When supplying a Time element, you precede it with a T.
  - d. If you only specify a T (as I did at the end), it will default to the current date and time.
  - e. `HISTORYSTRING` contains {the last known build date and time in TFS format ~ current date and time}.
5. `Set objMacro = vbld_TempMacros.Add("HISTORYSTRING", HISTORYSTRING)`
  - a. **This line creates a temporary macro with the name of `HISTORYSTRING` that contains the contents of the VBScript variable `HISTORYSTRING`.**
  - b. **By creating a temporary macro, you make it available to other steps in the build script.**

For reference, I have included the help text you can find inside of Visual Build Professional containing the different format strings.

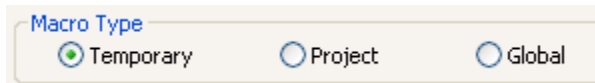
yyyy	4-digit year
yy	2-digit year (left padded with 0s)
y	1- or 2-digit year
mmmm	Month name
mmm	Month name (abbreviated)
mm	Month (left padded with 0s)
m	Month
dd	Day (left padded with 0s)
d	Day
hh	24-hr hour (left padded with 0s)
h	24-hr hour
HH	12-hr hour (left padded with 0s)
H	12-hr hour
MM	Minutes (left padded with 0s)
M	Minutes
SS	Seconds (left padded with 0s)
S	Seconds
AP	AM/PM
A	A/P

ap        am/pm  
a         a/p

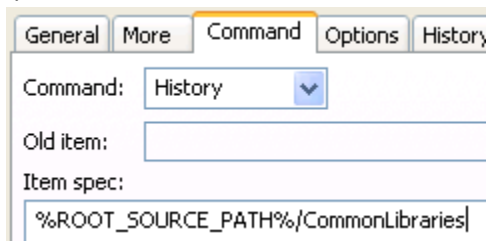
## Step 2: Retrieve History for your Projects

Now that you have the date and time saved in a TFS ready format, it is time to go retrieve the history for all of the projects referenced in your solution. For example, one of our projects contains 5 separate root source locations (Common libraries, Extension libraries, etc). In order to get a complete change history, we need to get the history of all 5 root source locations. I will explain how to do 1, you can repeat the process as many times as you need.

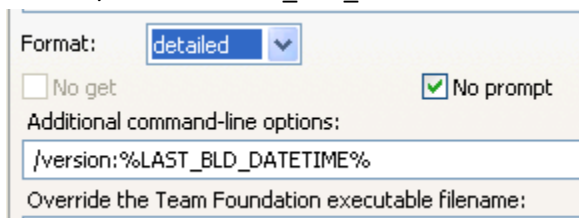
1. Inside of Visual Build, add a "Set Macro" object and name it something appropriate on the General tab (ie: Define CommonLibraryRevisionText).
2. Define it as a Temporary Macro from the Macro tab.



3. Click OK.
4. Add a "Team Foundation" object and name it something appropriate on the General tab (ie: Get History for Common Libraries).
5. Select "History" from the dropdown on the Command tab.
6. Enter the source path to the root source location you wish to retrieve history from in the "Item spec" field.



7. Select "detailed" or leave the option as default depending on the need of history detail returned.
  - a. Note that I found detailed to contain the complete history text of each check in whereas the default or brief could be cut off after a certain length. I opted for the full text by using detailed.
8. Enter the following command in the "Additional command-line options".
  - a. /version:%LAST\_BLD\_DATETIME%



9. We will define LAST\_BLD\_DATETIME in the final step.
10. Click OK.
11. Repeat these steps for each “root source location” that you need to report history for.

### Step 3: Last Build Date and Time Project Macro

You now have the meat of gathering the change history for each project. The last thing you will need to do is create a Project Macro that can contain and persist the last time a successful build was finished. This would go towards the end of your build script just before the save project step. This way, when it is updated from a successful run, the next run will be able to apply that new date and time to the run and retrieve only the changes that occurred between that date and time and now.

1. Inside of Visual Build, add a “Set Macro” object and name it Set LAST\_BLD\_DATETIME.
2. Define it as a Project Macro from the Macro tab.



3. On the Macro tab, provide the name LAST\_BLD\_DATETIME.
4. In the Description box, supply the temporary macro from the VBScript named %HISTORYSTRING%.

The image shows a form with the following fields:
 

- Name:** A text box containing "LAST\_BLD\_DATETIME".
- Delete the macro:** An unchecked checkbox.
- Value:** A text box containing "%HISTORYSTRING%".

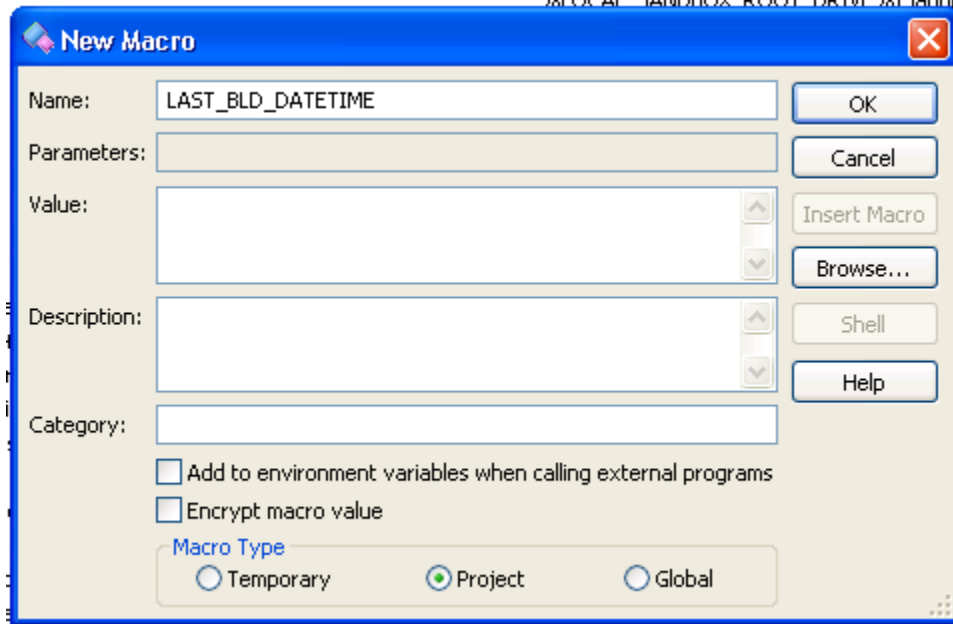
5. Now LAST\_BLD\_DATETIME has the date and time string from the beginning of the build script in a TFS recognized format for querying history.
6. Click OK.
7. Add a “Run Script” object and name it something appropriate on the General tab (ie: Save Project when changed).
8. On the script tab, insert the following script. Make sure to select VBScript from the dropdown.  
`If Project.IsModified Then Project.Save`
9. Click OK.

### Step 4: Create LAST\_BLD\_DATETIME Macro and Update the value

The last thing you will need to do is add the Project Macro of “LAST\_BLD\_DATETIME” to the Macro section, populate it with the current date and time and save the file. Then the next build you create will be ready.

1. From the Macros tab, right click on Project Macros and click Insert.
2. Supply the name of LAST\_BLD\_DATETIME in the name field.
3. Add a description if you like.

4. Make sure that Project Macro is selected.
5. Click OK.



6. Go back to the Project Steps Tab and build the “Define DATETIMENOW” and “Format DATETIMENOW for Team Foundation” steps.



7. Build the “Set LAST\_BLD\_DATETIME” step.
8. Save your project.
9. Now the next time you build your project it will be ready to pull a limited change history that you can report to others any way you wish. I include it in an email but your organization may do things differently.

## Acknowledgements

Team Foundation Server (TFS) is owned by Microsoft. To find out more about TFS, visit them here <http://msdn.microsoft.com/en-us/vstudio/ff637362.aspx>.

Visual Build Professional is owned by Kinook. To find out more about VBP, visit them here <http://www.kinook.com/VisBuildPro/>.