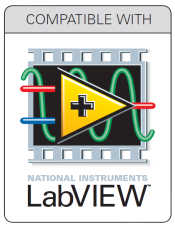
Tedious Task Manager (TTM) User Manual



July 2012 Edition

Worldwide technical support and product information:

www.toolsforsmartminds.com

TOOLS for SMART MINDS Corporate headquarter

Via Padania, 16 Castel Mella 25030 Brescia (Italy)

Contents

FIGURE INDEX

About this Manual

The TTM User Manual describes the functionalities of Tedious Task Manager (TTM) a productivity tools capable to edit, run parametric scripts and provide useful information about the way scripts have completed their job or have fails at a certain point.

Conventions

The following conventions appear in this manual:

▸	The ▸ symbol leads you through nested menu items and dialog box options to a final action. The sequence **Tools** ▸ **Options** directs you to pull down the **Tools** menu, select **Options** item.
Bold	Bold text denotes items that you must select or click on the software, such as menu items and dialog box options. Bold text also denotes parameter names.
italic	Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. This font also denotes text that is a placeholder for a word or value that you must supply.
`monospace`	Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames and extensions, and code excerpts.

`monospace italic`

Italic text in this font denotes text that is a placeholder for a word or value that you must supply.

INTRODUCTION

This document illustrates how to use TTM (Tedious Task manager). If you are a developer, you are reading the right document. TTM is a productivity tool dedicated to automate repetitive activities you encounter every time when you are at work. With TTM you can:

- build executables at specific time,
- save source code into a zip file automatically with a specific name calculated dynamically,
- organize your code files avoiding mistakes
- send emails to your team mates when a build is ready
- many more nice things...

DISTRIBUTIONS

TTM is available in two different distributions:

TTM STANDALONE APPLICATION

This distribution of TTM runs on Windows based PC and is generally used to schedule some repetitive activities on files, folders, etc.

TTM LABVIEW ADD-ON

This distribution of TTM is fully integrated with LabVIEW development environment and can execute some specific LabVIEW commands like building specifications and running custom VI.

REQUIREMENTS

TTM standalone application requires Windows XP/Vista/7.

TTM ADD-ON for LabVIEW requires LabVIEW 2012 or higher. FTP file transfer functionality is available only if Internet Toolkit is properly installed.

INSTALLATION

Download TTM.VIP at the following link:

https://www.toolsforsmartminds.com/products/ttm.php

To install the .vip package use VIPM package manager. You can download a copy at the following URL

www.jki.net

GETTING STARTED WITH TEDIOUS TASK MANAGER (TTM)

In this chapter you learn how to create your own scripts to automate everyday activities. TTM provides a list of built-in functionalities called *commands*; you can extend this set of commands with your VIs and command line instructions.

RUNNING TTM INSIDE LABVIEW

If you have installed TTM for LabVIEW you can launch TTM main window from **Tools ▸ T4SM** menu as shown below.

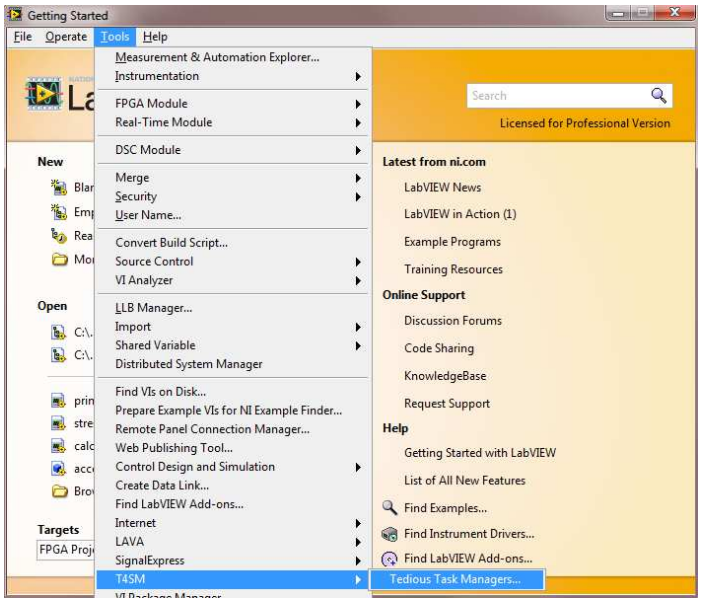


FIGURE 1 – TEDIOUS TASK MANAGER IS LAUNCHED FROM TOOLS ▸ T4SM MENU.

RUNNING TTM STANDALONE APPLICATION

Click on Start ▸ All Programs ▸ TOOLS for SMART MINDS ▸ TTM ▸ TTM from start menu.

At first activation TTM takes longer time to display its main window because it creates default folder used to store scripts and log files. The following figure shows TTM main window:

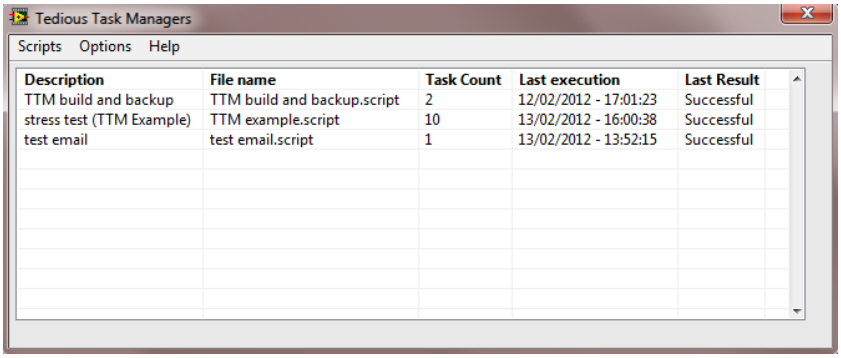


FIGURE 2 – TTM MAIN WINDOW DISPLAYS ALL AVAILABLE SCRIPTS.

CREATING A NEW SCRIPT

To create a new script, click on **Scripts** ▸ **New** from Scripts menu. Type script file name and a new empty script is created.

EDITING A SCRIPT

To edit an existing script click on **Scripts** ▸ **Edit** form Scripts menu. Script Editor's window appears and you can edit script properties. TTM sets script name equal to its file name, you can change script name at any time editing **Name** field.

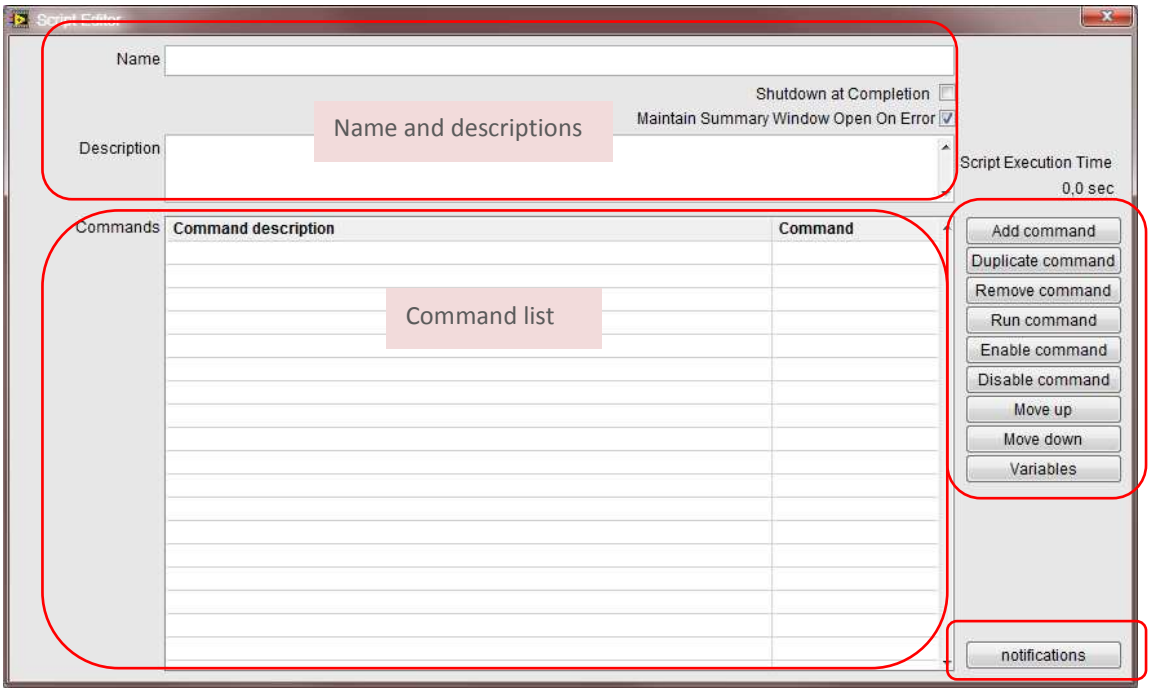


FIGURE 3 – SCRIPT EDITOR WINDOW.

By default, script file **Name** is used as script's name. **Description** is used to describe the script purpose. **Commands** table contains all script's commands. Use button list at the right of Commands table to add/edit/remove commands to your script. Click **Add command** button to add new command to your script. New commands are appended at the end of the list. To change the order of execution of commands, select the command you want to move up or down and click on **Move up** or **Move down** buttons. Click **Remove command** button to remove a command from script. Use **Run command** button during script editing to execute a single selected command. Use **Variables** button to view current variables values.

Below script's name you have two checks:

- Shutdown at completion
- Maintain Summary Window Open on Error

These checks are evaluated after script execution.

Shutting down system after script execution

When your script takes a lot of time to execute you may wish TTM switches off your PC when finishes to execute the selected script. Switching computer off is done regardless script fails of succeeds. Mark this checkbox when, for example, launched script execution is going to end late in the evening and you don't want to keep your PC on all night. By default this checkbox is set to FALSE.

Maintaining Summary window open on error

When script execution is not supervised and you want to have details about script execution in case of fail, set "Maintain Summary Window Open On Error" checkbox to TRUE so that when script execution aborts due to a command error, summary window shows all execution details.

Notifications

Notifications allows TTM to notify the final execution status at the end of script execution by email.

Click **Notification** button to extend Script Editor with notification section as shown in the following figure.

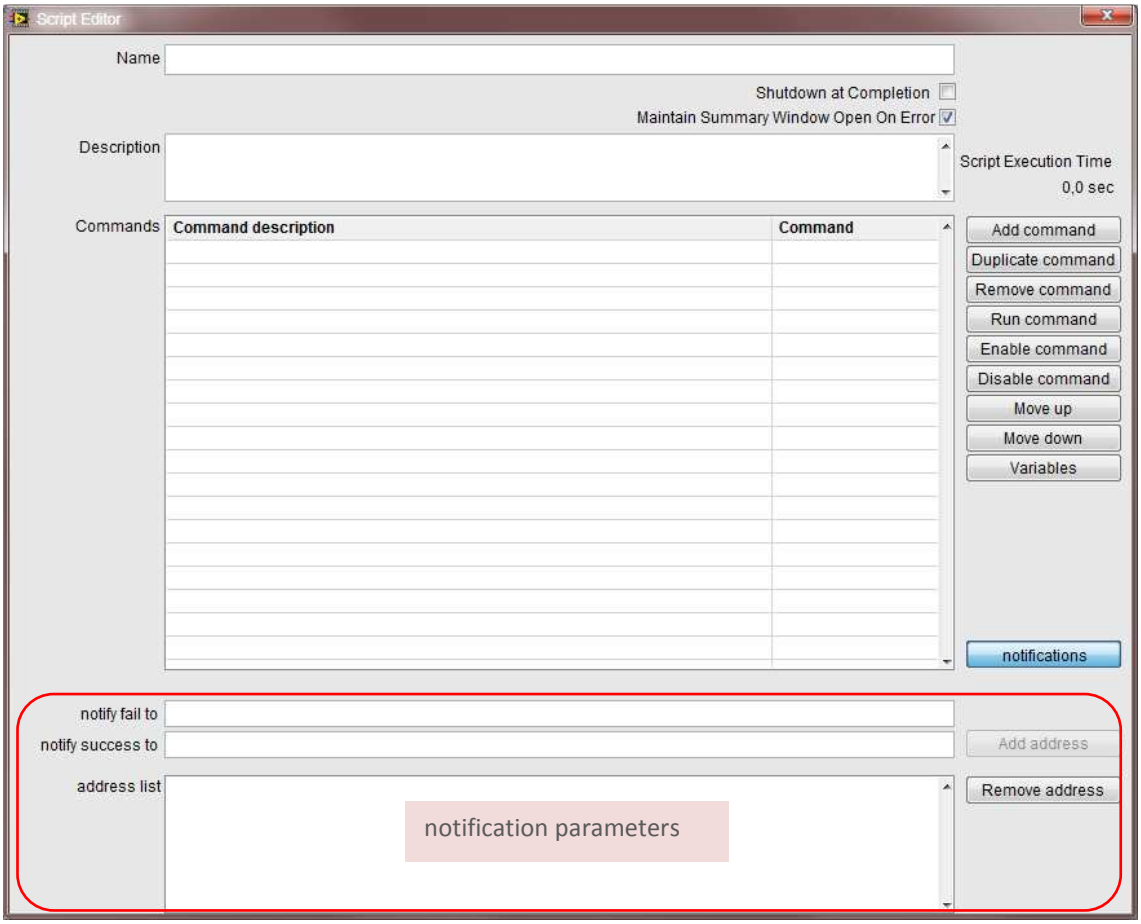


Figure 4 – Script editor window has "notification" section at the bottom, to specify email addresses.

If TTM completes script without errors, a successful notification is sent to all recipients indicated in **Address List**. If Script execution aborts a single email is sent to recipient indicated in **Notify fail to** field.

Important TTM can send emails only if mail server is properly configured. Please read Paragraph "Parameters" at the end of this manual, to setup SMTP parameter and allow TTM to send email.

ADDING A COMMAND

There are two buttons that allows to add a command to a script:

ADD COMMAND

Use **Add command** button to add a new command. Commands can be selected among a set of existing command with the command selection window.

DUPLICATE COMMAND

Use **Duplicate command** button to create a new command as copy of an existing one. new (duplicated) command is inserted before existing one.

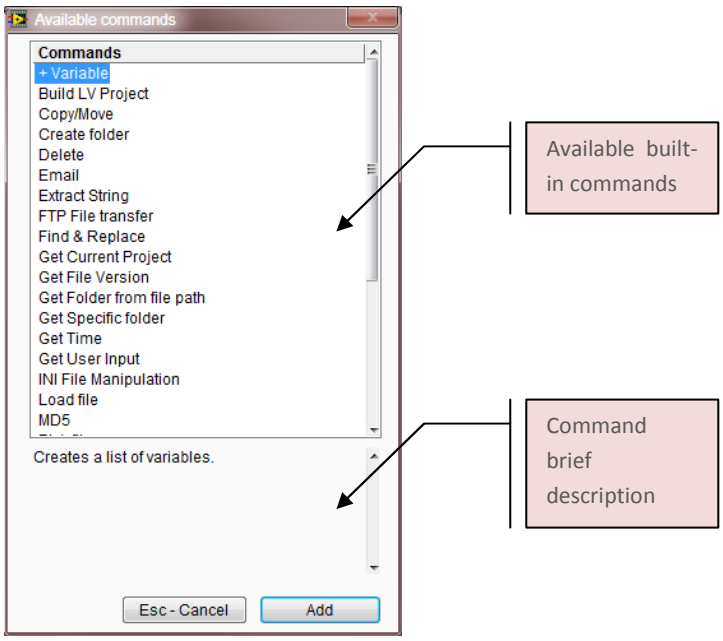


FIGURE 5 – COMMAND SELECTION WINDOW.

When you click on a row of commands table, TTM displays a brief description. To proceed with command selection press **Add** button or double-click on command row. TTM scripts are composed by a sequence of commands: in most cases, built-in commands can be used to fulfill user's needs. If you need to integrate a script with special features not available as external application, you can create your custom commands as LabVIEW VIs. TTM is provided with a template you can use to create your own commands. See **Run VI** command description the get more details.

Available commands are:

Command name	Command description
+ Variable	Creates a script variable
Build LV Project	Build selected specifications
Copy/Move	Copies/moves file
Create folder	Creates a new folder
Delete	Deletes a file or folder
Email	Sends email with an attachment
Extract string	Extracts a token from a string
FTP file transfer	Transfers a file on a FTP host
Find & Replace	Find a string into a text file and replace it with a new string
Get current project	Returns current LabVIEW project's name
Get File version	Returns executable file version
Get Folder form file path	Returns a folder path which contain a specific file
Get Specific folder	Returns a system folder
Get Time	Returns current time with a custom format
Get User input	Returns user input
INI File Manipulation	Reads and writes keys into a .IN configuration file
Load file	Loads a text file into a variable
Manipulate VIs	Clean up block diagram and set password to a list of selected VIs
MD5	Computes MD5 code of specific file
Pick file	Selects a file path according to custom rules
Print document	Print documents on a selected printer
Run VI	Runs external VI, to execute custom commands
Save file	Saves a file
Start LabVIEW	Runs latest LabVIEW version installed in your PC
System	Executes external commands
Unzip	Unzip existing zip file
Wait file count	Waits a specific file count into a folder
Wait for User	Stops script execution until user presses OK button
Wait for file	Waits for a specific file arrives or leaves a folder
Wait for time	Waits until a specific time is reached
Zip	Zip a specific file or folder

DISABLING/ENABLING A COMMAND

Every command can be disabled or Enabled. When a script runs, only enabled command are executed. Use **Enable command** and **Disable command** to change selected command's state. Disabled commands have a special symbol at the left of their name as shown below:

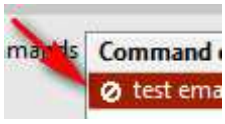


FIGURE 6 – DISABLED COMMANDS ARE MARKED WITH A CROSSED CIRCLE.

DELETING A SCRIPT

To delete an existing script select its row from script table and click on **Script ▸ Delete** form Scripts menu. Confirm your choice to delete script form disk.

DUPLICATING A SCRIPT

To duplicate an existing script, select its row from script table an click on **Scripts ▸ Duplicate** from Scripts menu.

AVAILABLE COMMANDS

In this chapter TTM's available commands are described. A script is composed by a sequence of commands. TTM executes commands in the same order they are placed into the sequence. Commands are defined by a description and some parameters which depend on command type. Description doesn't affect command execution, simply describes the command purpose in your script. If Description is left empty TTM fills it automatically with a default value and most significant field's values. Some command's editors show *Hide error* checkbox in the upper right corner. This field avoid TTM to stop script execution if command fails.

SCRIPT VARIABLES

Except Description field and variable names, all other text fields can be set to static values (for example "C:\myData") or dynamic values (for example <MY DATA>) that TTM evaluates during script execution. Variables are defined during script editing phase and are evaluated during script execution. Commands are executed in the same order they are displayed in script editor. So variable's values are calculated in that order. A variable's value can be composed with other variables.

DECLARING VARIABLES

Variables are declared during script editing and are initialized at script loading, with empty values, or during script execution. Variables can be declared inside command's editors with **[+]** button as shown in the following figure:

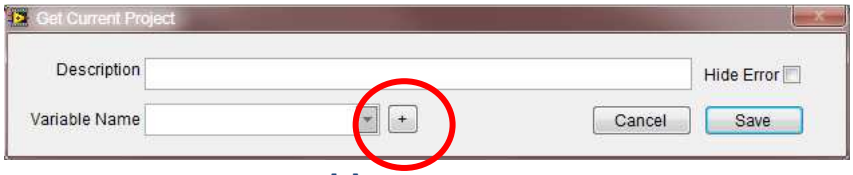


FIGURE 7 – USE [+] BUTTON TO DECLARE A VARIABLE.

FIGURE 8 – "ADD VARIABLE" WINDOW LET YOU DECLARE A VARIABLE.

IMPORTANT: Variable names are case sensitive hence ***Build***, ***BUILD*** and ***build*** are treated as three different variables. Variable names can be compose with the following set of characters: a..z, A..Z, 0..9, space.

When a script runs, TTM executes commands in the planned order, if finds a variable name, TTM proceeds as follow:

If variable name is not found in TTM lookup table, TTM creates variable and initialize the value according to command result, otherwise TTM updates variable's value according to command result.

IMPORTANT: When a script is loaded into memory with a run or edit command, a new name space is created so all existing variables are deleted. Remember that every time you run a script all existing variables are destroyed so a script cannot make use of variables created or modified by other scripts.

USING VARIABLES

Variables can be used in every field of a command except command description that must be a static text. When you specify a variable name you have to write variable name inside "<" and ">" brackets as shown in the following example:

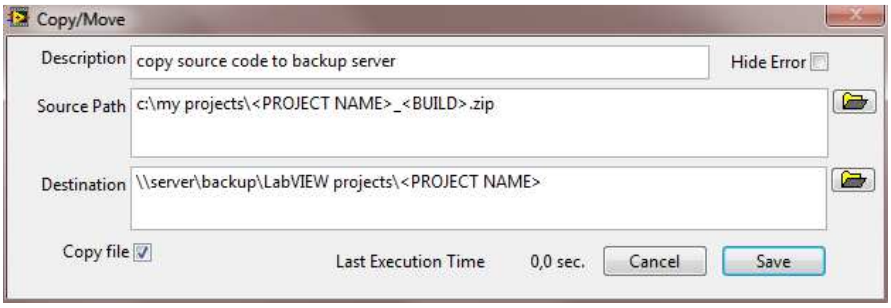


FIGURE 9 – A COMMAND WITH VARIABLES IN ITS FIELDS.

Source Path field contains two variables: PROJECT NAME and BUILD, their values have to be calculated in other commands executed before the one in above example. TTM stores in a lookup table all variable names and their values so, in above example, TTM can evaluate **Source Path** substituting <PROJECT NAME> and <BUILD> with respective values.

CREATE A VARIABLE

This command creates a variable and sets a value. If **Variable value** contains references to other variables, when script is executed TTM calculates a run-time value. **Variable name** can be composed with any character in this set: a..z, A..Z, 0..9, space.

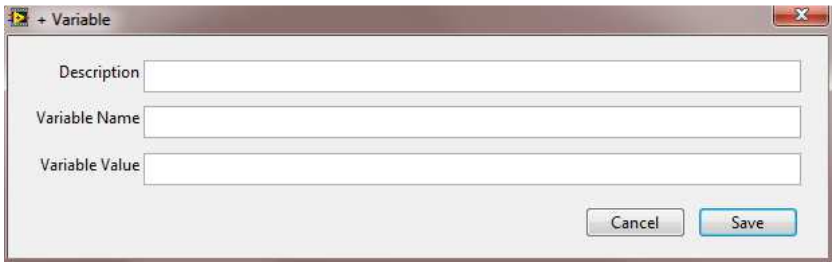


FIGURE 10 – "CREATE A VARIABLE" EDITOR WINDOW.

BUILD A LABVIEW PROJECT

This command builds one or more build specifications of a selected LabVIEW project. Building specifications have to be present into LabVIEW project at the time this command executes.

IMPORTANT: This command is available in TTM ADD-ON for LabVIEW.

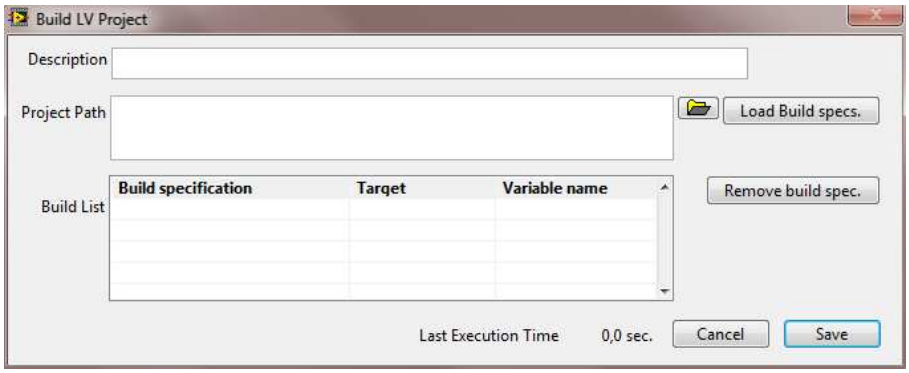


FIGURE 11 – "BUILD LV PROJECT" EDITOR WINDOW.

To properly configure this command, follow the following steps:

1. select LabVIEW project
2. Click **Load build specs.** button to load specifications. **Build list** is compiled with available build specifications. TTM associates a new variable to every build specification, with names EXE 1, EXE 2, etc.
3. Selected unwanted build specifications and click Remove **Build specs.** to remove them from table.
4. Use triple click on a Variable name to edit its value, if required.

Build list must contain at least a building specification. Build specifications are compiled in the same order they are indicated in Build **List**.

IMPORTANT: Set Build specification value to an empty string to allow TTM to search into build specification list of selected project and use an existing build specification.

When LabVIEW starts building process, interaction with TTM windows is suspended up to the end of building process. Trying to stop building process can drive TTM to unpredictable behavior.

UNZIP A COMPRESSED FILE

This command unzips compressed files and folders contained into *Zip file*. All created files and folders are created into *Target folder*. If *target folder* doesn't exist, TTM creates it before unzipping.

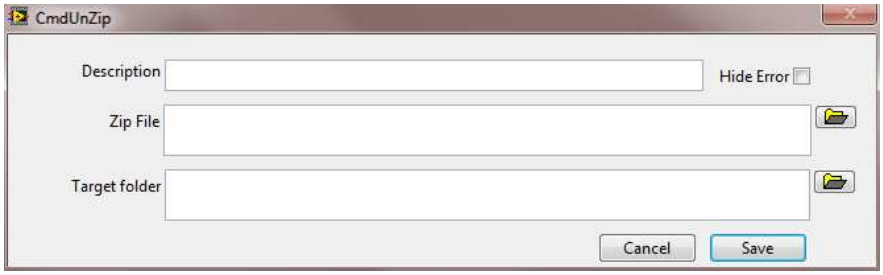


FIGURE 12 – "UNZIP" EDITOR WINDOW.

If **Target folder** doesn't exist, it is created automatically.

COPY/MOVE/RENAME FILES AND FOLDERS

This command copies or moves file indicated in *Source Path* field. File is copied onto *Destination* path. Both *Source Path* and *Destination* are evaluated at run-time and can contain VARIABLE NAMES. Use *Copy file* flag to choose if file has to be copied or moved.

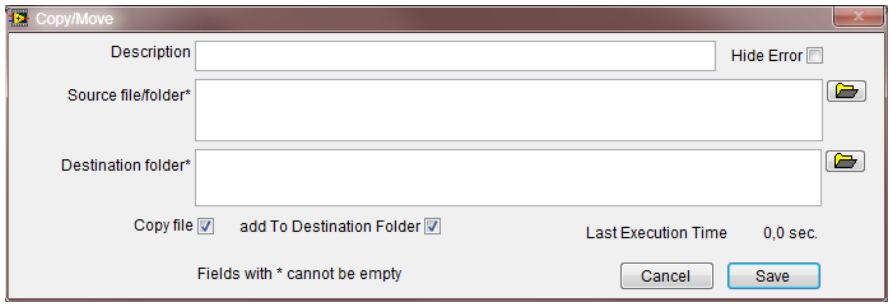


FIGURE 13 – "COPY/MOVE" EDITOR WINDOW.

If file exists before command execution, it is overwritten. If **Destination folder** does not exist, Copy/Move command creates it. Set **Add to Destination Folder** checkbox to TRUE if you want to move/copy **Source file/folder** into **Destination folder**. Set **Add to Destination Folder** checkbox to FALSE if you want to rename **Source file/folder** with **Destination folder** as new name.

CREATE A FOLDER

This command creates a folder as specified in **Folder name**.

FIGURE 14 – "CREATE FOLDER" EDITOR WINDOW.

If folder exists before command execution, no error is returned

DELETE A FILE/FOLDER

This command deletes folder/file specified in *Path* field.

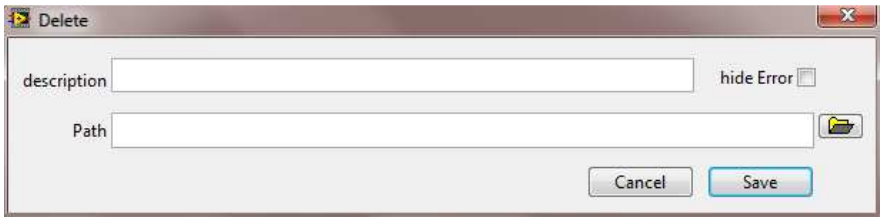


FIGURE 15 – "DELETE" EDITOR WINDOW.

If *Path* doesn't exist, no error is signaled.

EMAIL

This command sends email to specified list of recipients. Email can has one attached file. All fields are evaluated at run-time and can contain variable names.

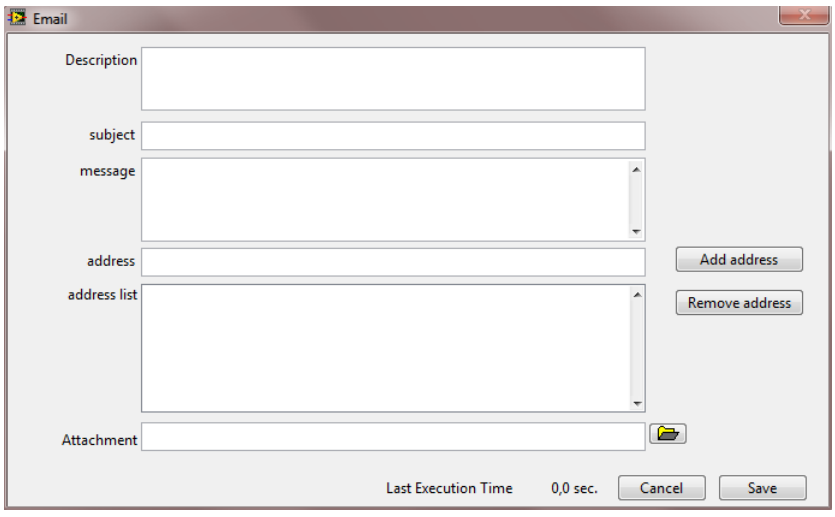


FIGURE 16 – "EMAIL" EDITOR WINDOW.

EXTRACT STRING

This command gets a string part from an input string, provided by a variable value. Extracted string is set to an output string value.

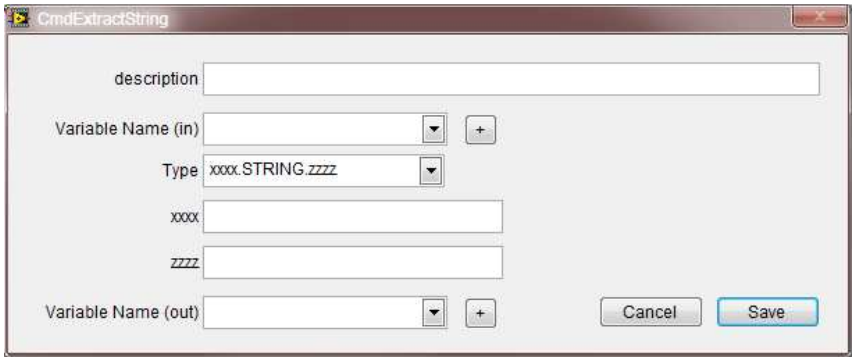


FIGURE 17 – "EXTRACT STRING" EDITOR WINDOW.

This command handles three different types of extraction:

Type	Description	Example
xxxx STRING	Extract STRING value after xxxx token	Input string: c:\projects\my file.zip xxxx: c:\projects\ STRING: my file.zip
xxxx STRING zzzz	Extract STRING value between two different tokens, called respectively xxxx and zzzz.	Input string: my_package_1.0.12.vip xxxx: my_package_ zzzz: .vip STRING: 1.0.12
STRING zzzz	Extract STRING value before zzzz token	Input string: my file.zip zzzz: .zip STRING: my file

FIND AND REPLACE STRING

This command loads in memory a file, finds **Search string** value and replaces all its occurrences with **Replace string** value.

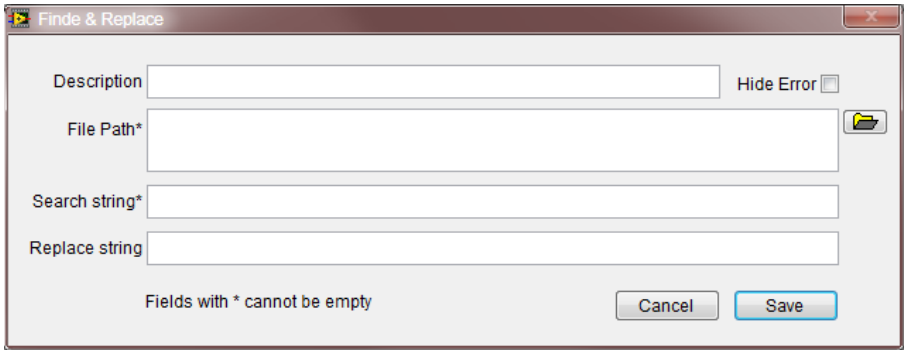


FIGURE 18 – "FIND & REPLACE" EDITOR WINDOW.

If file is not found, an error is returned. If **Replace string** is left empty, this command removes all occurrences of Search string from file. String replacement is not case sensitive. You can specify variables in both **Search string** and **Replace string** fields.

FTP, TRANSFER A FILE ON A HOST

This command transfers a file to a remote host using FTP protocol.

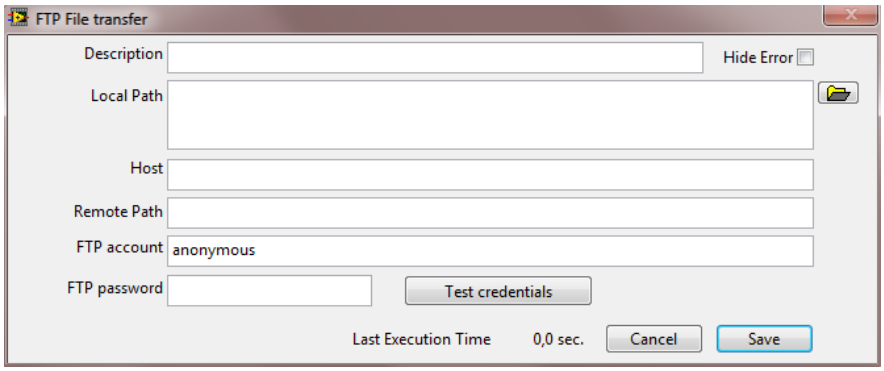


FIGURE 19 – "FTP FILE TRANSFER" EDITOR WINDOW.

Host specifies the destination where TTM has to copy file selected in **Local Path** field. Both fields can be composed with variables.

Click **Test credentials** button to verify FTP connection with specified credentials. If **remote path** value ends with a backslash character "/", TTM considers this path as a folder and appends the file name extracted from **Local path** value as illustrated in the following examples.

Example 1

Local path = c:\mydata\myfile.txt

Remote path =/sharedData/XYZ

TTM tries to create the file /sharedData/XYZ

In the above case, XYZ is used as remote file name instead of myfile.txt

Example 2

Local path = c:\mydata\myfile.txt

Remote path =/sharedData/XYZ**/**

In this second case, TTM tries to create the file /sharedData/XYZ/myfile.txt

In the above case, XYZ is considered as a folder name.

FTP, DELETE A FILE ON A HOST

This command deletes a file on a remote host using FTP protocol.

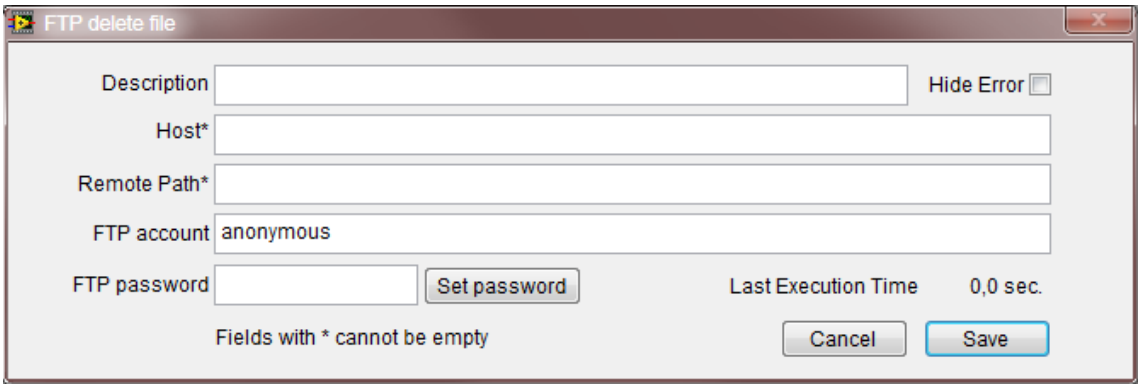


FIGURE 20 – "FTP DELETE" EDITOR WINDOW.

Host specifies the destination where TTM has to delete file indicated in **Remote Path** field. Both fields can be composed with variables. Click **Test credentials** button to verify FTP connection with specified credentials. If **remote path** value ends with a backslash character "/", TTM considers this path as a folder and abort command, because TTM doesn't delete remote folder.

GET CURRENT PROJECT

This command retrieves current project file name. You have to specify a **Variable Name** that is set at run-time with open project full path. If multiple projects are open it returns the first project file name

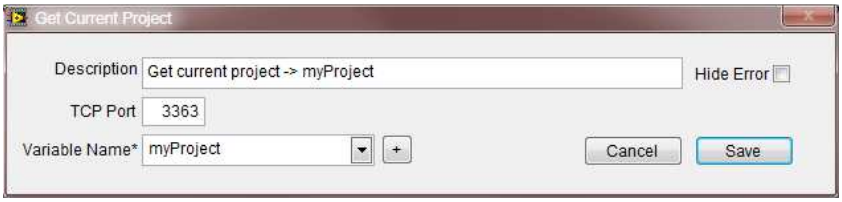


FIGURE 21 – "GET CURRENT PROJECT" EDITOR WINDOW.

If no projects are open, an error is returned. Ensure VI server is enabled in LabVIEW and specify the right **TCP port** value. The following figure illustrates the LabVIEW option page where you can enable VI server:

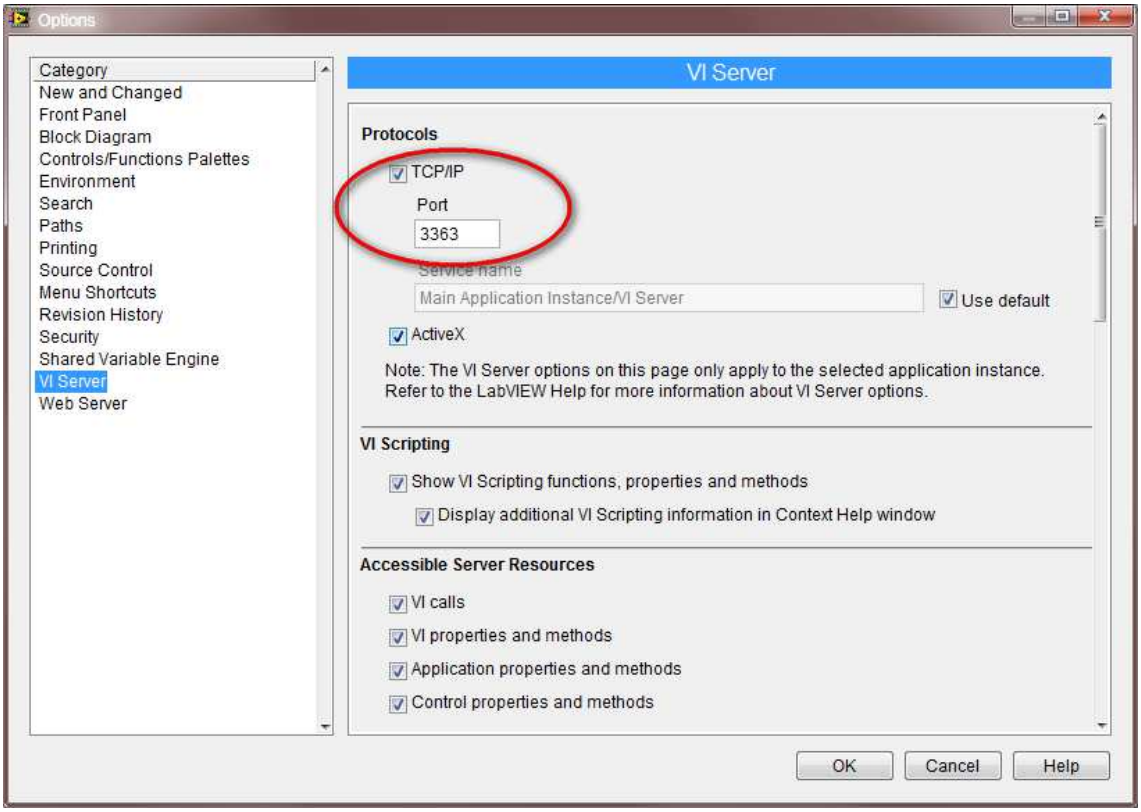


FIGURE 22 – LABVIEW OPTION WINDOW, WITH VI SERVER SETTINGS.

GET FILE VERSION

This command retrieves version of executable file (.exe) specified in **File Path**. You have to specify a **Variable Name** that is set at run-time with file version.

FIGURE 23 – "GET FILE VERSION" EDITOR WINDOW.

File version can be viewed with file's property dialog box into resource explorer: browse to your executable folder and right click on your file, choose properties and select version tab, as shown in the following figure.

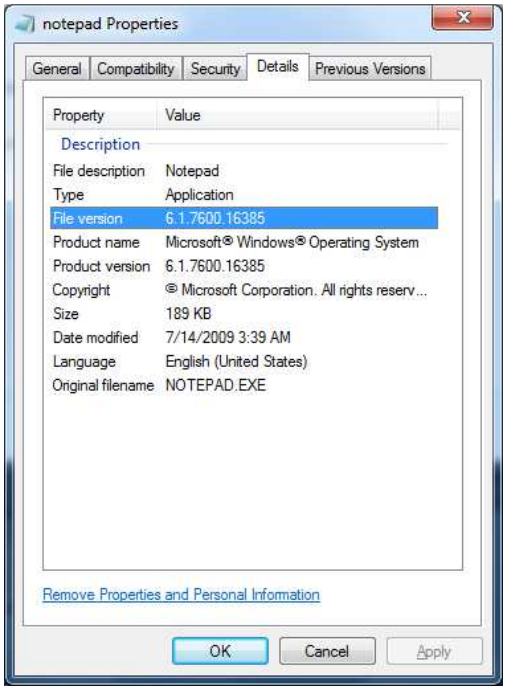


FIGURE 24 – USE PROPERTY DIALOG TO VIEW FILE VERSION.

File version is defined during executable creation. If your executable files are made with LabVIEW application builder, you have to check Auto increment as shown in the following figure:

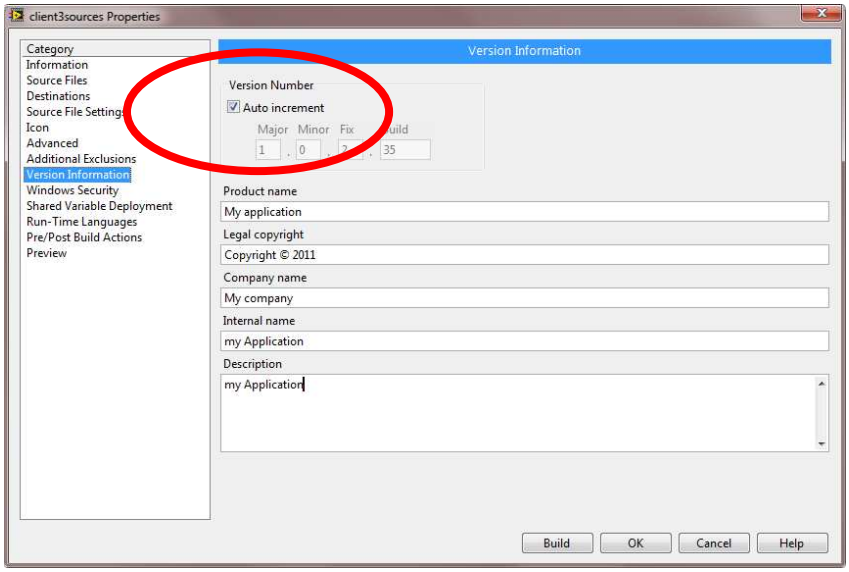


FIGURE 25 – ENABLING AUTO INCREMENT OPTION INTO BUILD SPECIFICATION PROPERTIES.

GET FOLDER PATH FROM A FILE FULL PATH

This command sets a variable value equal to a full path of the folder that contains **File Path**. Use **Levels Up** to select the number of elements removed from **File Path** value,

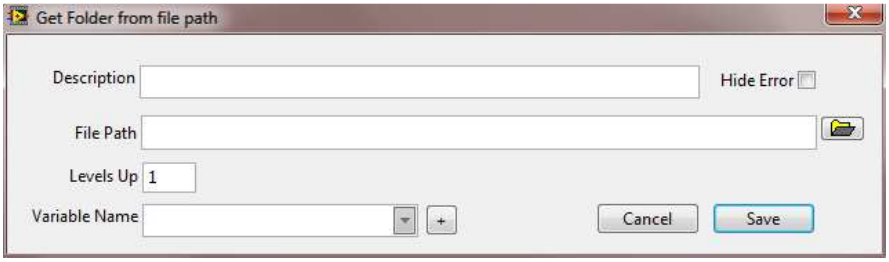


FIGURE 26 – "GET FOLDER FROM FILE PATH" EDITOR WINDOW.

Example:

If **File Path** is `c:\my projects\LabVIEW\Customer ABCD\Application A\A.proj`

When **Level Up** is set to 1,
command returns `c:\my projects\LabVIEW\Customer ABCD\Application A`

When **Level Up** is set to 2,
command returns `c:\my projects\LabVIEW\Customer ABCD`

When **Level Up** is set to 3,
command returns `c:\my projects\LabVIEW`

GET SPECIFIC FOLDER

This command returns a folder's full path among a list of available specific folders.

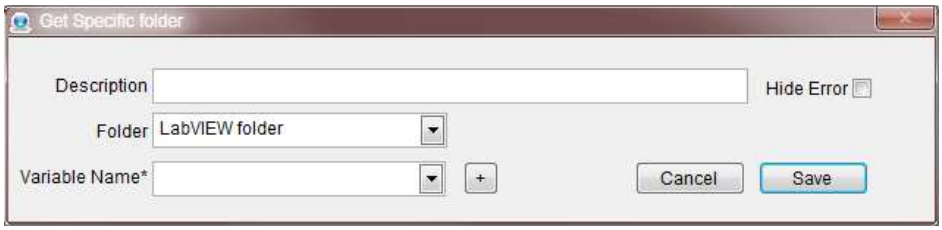


FIGURE 27 – "GET TIME" EDITOR WINDOW.

You can choose one of the following folder's full path:

Folder	Description
LabVIEW folder	LabVIEW folder that contains labview.exe
User.lib	User.lib folder inside LabVIEW folder
vi.lib	vi.lib folder inside LabVIEW folder
Root	Local disk root (i.e. c:\)
Temporary folder	Current system folder for temporary files
Default user data	Default user data folder
Windows	Windows main folder (i.e. c:\windows)

GET TIME

This command sets a variable value equal to the PC's time at the moment it is executed. You can specify the format of time with **Time format** field.

FIGURE 28 – "GET TIME" EDITOR WINDOW.

If Time format is left empty, TTM assumes default time format %Y%m%d-%H%M%S that means data and time info are represented as YYYYmmdd-HHMMSS. Notice that you can use fixed string inside time format, to create special string. For example you can use the following Time format

Year-%Y Month-%m and TTM calculates at runtime the following value ***Year-2012 Month-03***

The following table illustrates available format codes:

Code	Description
%X	locale-specific time
%H	hour, 24-hour clock
%I	hour, 12-hour clock

%M	minute
%S	second
%<digit>u	fractional seconds with <digit> precision
%p	a.m./p.m. flag
%x	locale-specific date
%y	year within century
%Y	year including century
%m	month number
%b	abbreviated month name
%a	abbreviated weekday name

Format can be composed by any combination of format codes, string constant and VARIABLES. TTM evaluates **Format** in two steps:

1. Variable evaluation: variable names are changed with their values
2. Format codes evaluation: format codes are changed with their values at the time of execution.

GET USER INPUT

This command sets a variable value with an input text. User input can be empty text.

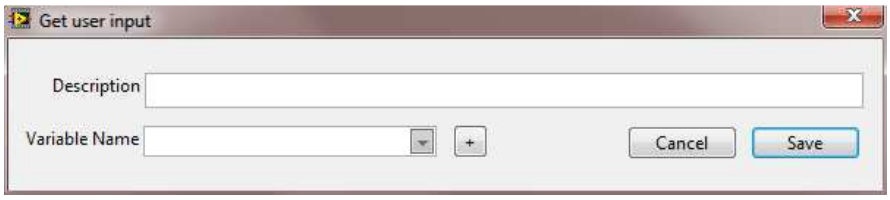


FIGURE 29 – "GET USER INPUT" EDITOR WINDOW.

When this command runs, script execution is stopped until user confirms input and presses OK button on a message box.

INI FILE MANIPULATION

This command allows you to manipulate a .INI configuration file. In a single command you can group different actions on the same file. Command opens configuration file, performs all actions and then saves it.

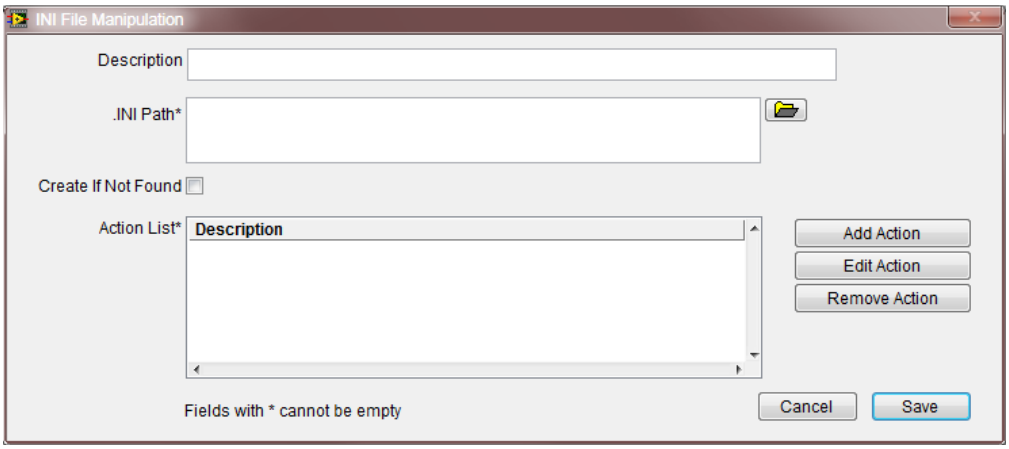


FIGURE 30 – "INI FILE MANIPULATION" EDITOR WINDOW.

If file not exists and **Create if not found** is NOT checked, an error is returned. Actions on configuration file are indicated in action list. Available actions are indicated in the following table

Action	Description
Read Key	
Remove Section	
Remove Key	
Write Key	

When you press **Add Action** button, actions are listed in a specific window as shown in the next figure.

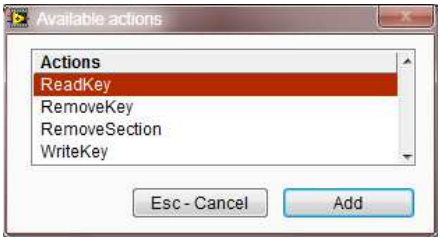


FIGURE 31 – "AVAILABLE ACTIONS" SELECTION WINDOW

Important: all key values are treated as string values regardless of their format.

READ KEY ACTION

This action reads **Key** value from **Section**. Key value is saved into **Variable Name** variable

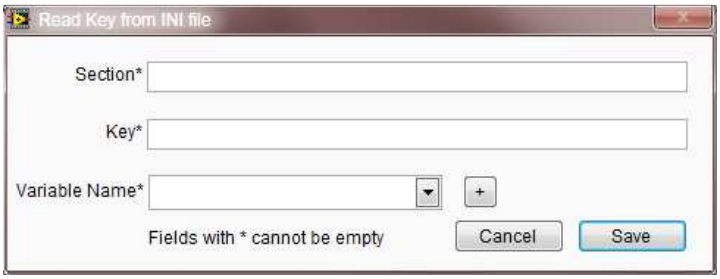


FIGURE 32 – "READ KEY" ACTION WINDOW

Important: Notice that Section value has to be indicated without square brackets.

REMOVE KEY

This action removes **Key** from the key set in **Section**, if present.

FIGURE 33 – "REMOVE KEY" ACTION WINDOW

REMOVE SECTION

This action removes **Section** and all keys associated, if present.

FIGURE 34 – "REMOVE SECTION" ACTION WINDOW

Write Key

This action writes a Key value into **Section**. If **Key** is not present among key set in Section, it creates a new key.

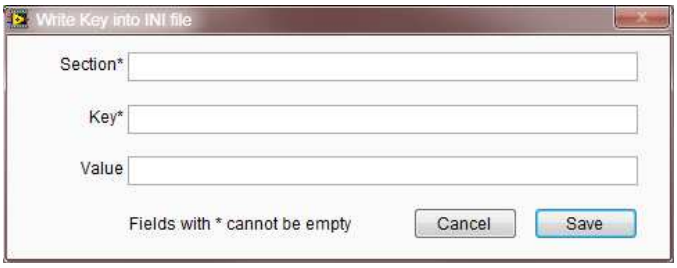


Figure 35 – "Write Key" action window

Important: This command is helpful when you need to maintain variable values calculated by a script, for other scripts or for future executions of the same script or to exchange these valuese with third party applications.

Load file

This command loads a file into a variable.

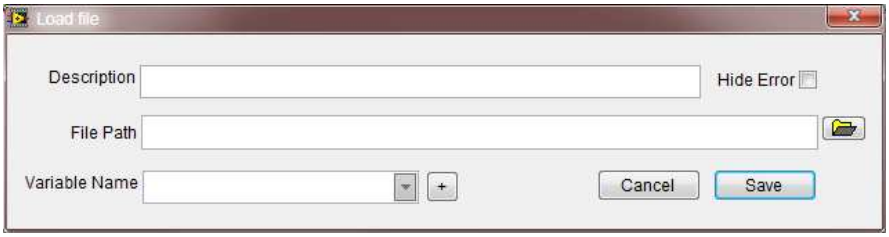


Figure 36 – "Load file" editor window.

If file not exists an error is returned.

Manipulate VIs

This command selects a list of VIs into selected folder and performs the following task on each VI:

- Block diagram clean up
- change password
- change "allow debugging" VI property
- custom manipulation

This command is also useful to analyse a set of VIs and create a comprehensive report where VIs are grouped by their state (idle, Run, bad). The report can be customized with VI's metrics, VI's help details and VI's password.

Set **Current VIs password** value to open password protected VIs. If C**hange VIs password** checkbox is marked, every VI is saved with **New VIs password** value.

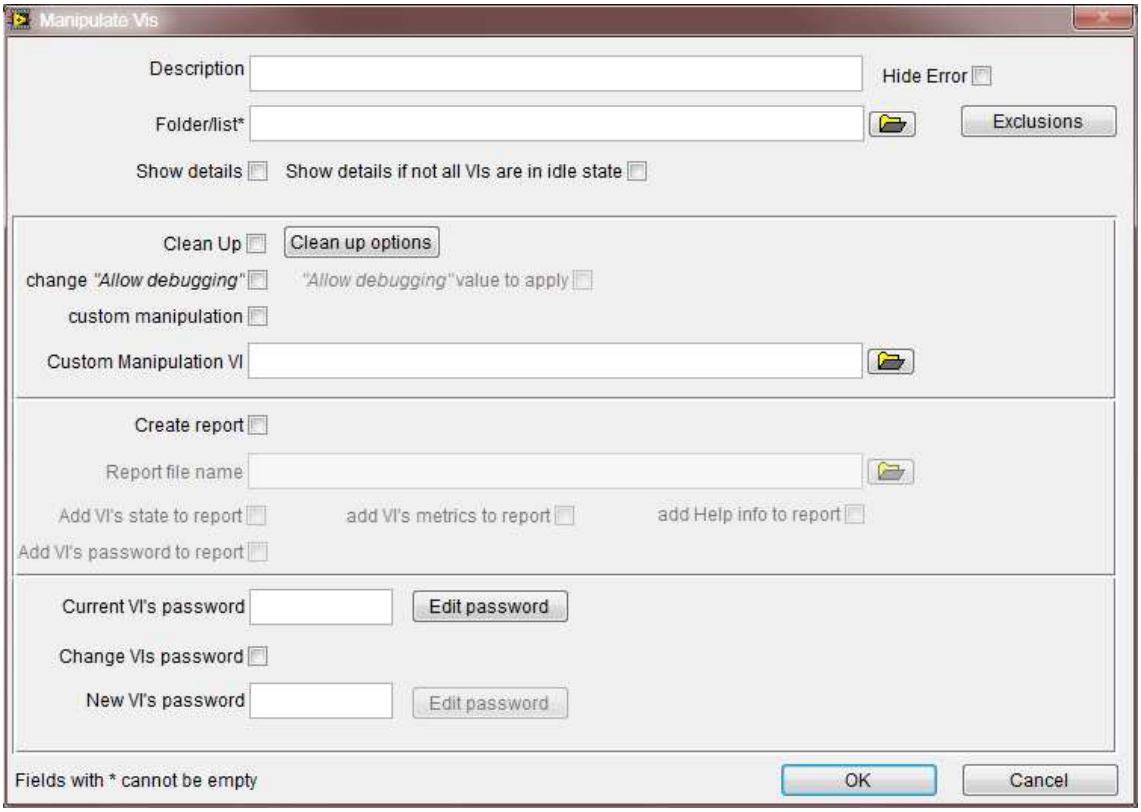


FIGURE 37 – "MANIPULATE VIS" EDITOR WINDOW.

Use **Clean Up options** button to customize the way clean up is performed. the following figure illustrates default values for block diagram clean up.

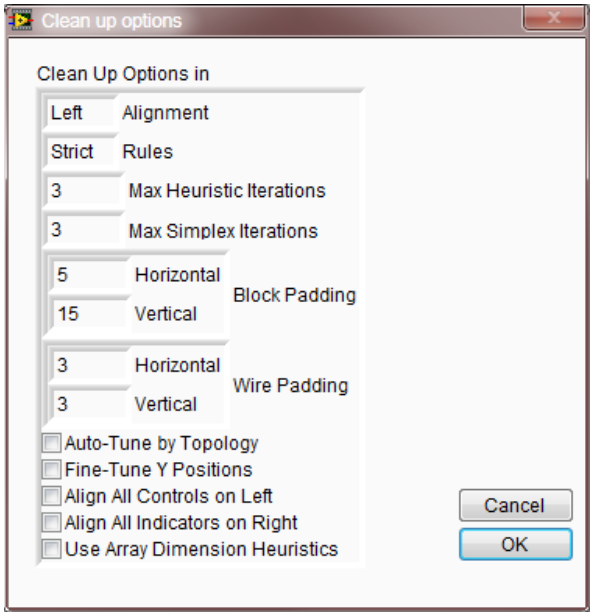


FIGURE 38 – "CLEAN UP" OPTIONS WINDOW.

You can exclude subfolders and VIs with **Exclusions** button.

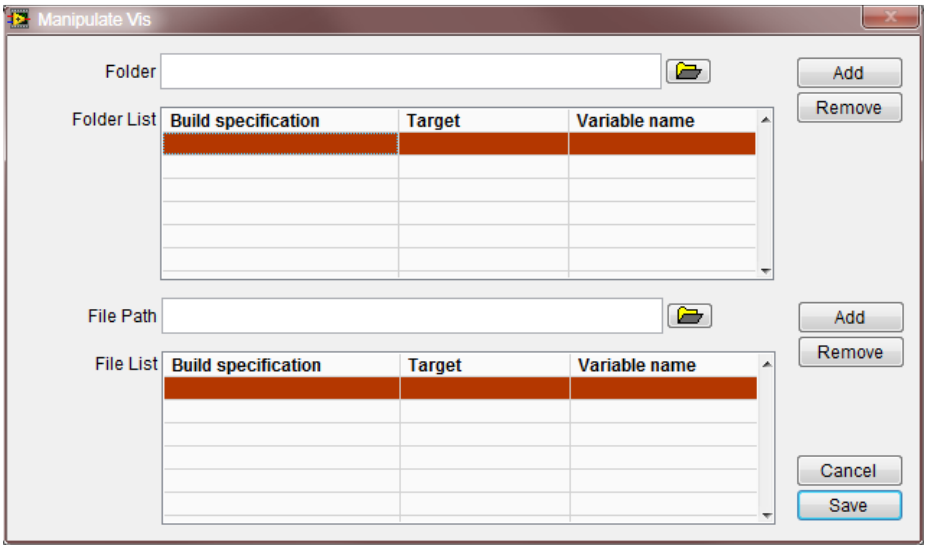


FIGURE 39 – "EXCLUSIONS" WINDOW.

MD5

This command computes the MD5 message-digest of a file. The MD5 message-digest is a 128-bit number displayed in lowercase hexadecimal format.

FIGURE 40 – "MD5" EDITOR WINDOW.

If file not exists an error is returned. MD5 value is saved into **Variable Name** and, if **Create file** is checked, in a file with same name and path of **File Path** field, but with extension equal to .MD5.

PICK FILE

This command sets a variable with the selected file full path. File selection is done according to selected **Criteria** and **Pattern** among available file in **Folder** at the time command runs. If no files are present, an error is returned.

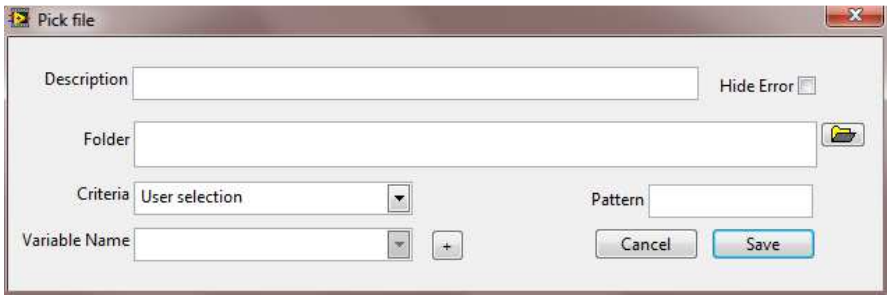


FIGURE 41 – "PICK FILE" EDITOR WINDOW.

Available criteria are illustrated in the following table:

Criteria	Description
Newest	Select file with most recent date and time, among files which match specified pattern
Oldest	Select file with most old date and time, among files which match specified pattern
Biggest	Select file with biggest file size, among files which match specified pattern
Smallest	Select file with smallest file size, among files which match specified pattern
First (Alphabetical order)	Select file among them which match specified pattern, that comes first in alphabetical order (not case sensitive)
Last (Alphabetical order)	Select file among them which match specified pattern, that comes last in alphabetical order (not case sensitive)
User Selection	Let user select a file. In this case **Folder** is used as start path for file dialog and user can choose a file path in every folder. In this case only, a non existing file can be specified.

PRINT DOCUMENT

This command prints a document, using windows shell, to a selected printer. if no printer is selected, default printer is used.

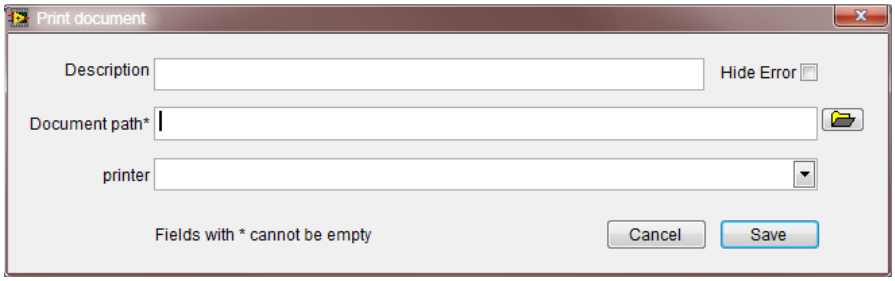


FIGURE 42 – "PRINT DOCUMENT" EDITOR WINDOW.

RUN VI

This command runs a VI which path is specified in **VI Full Path** field. Selected VI must fulfil connector pane requirements.

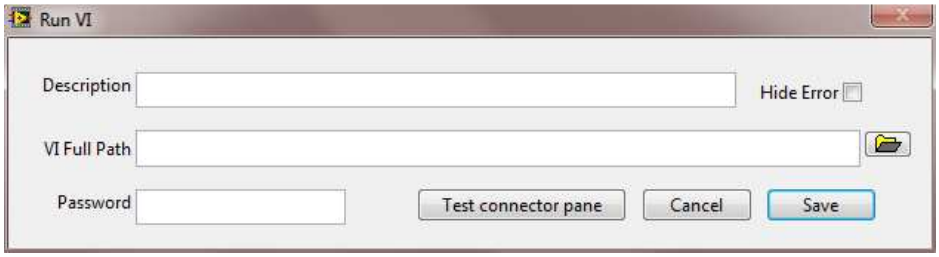


FIGURE 43 – "RUN VI" EDITOR WINDOW.

If VI is password protected, use **Password** field to allow TTM to load and run selected VI. You can find a template that shows how to properly setup connector pane for your VIS in **Addons ▸ TOOLS for SMART MINDS ▸ TTM palette.**

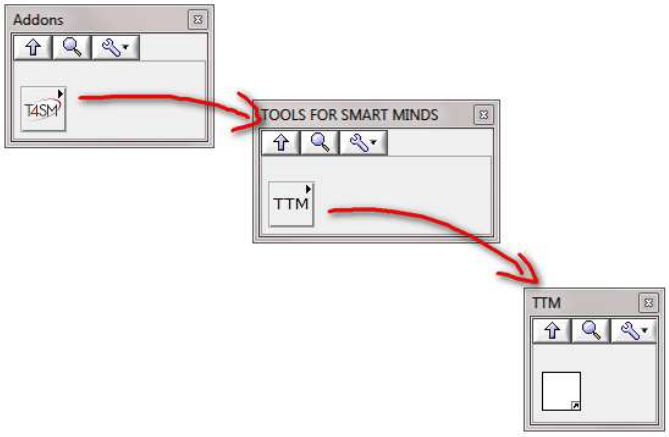


FIGURE 44 – TEMPLATE IS AVAILABLE IN LABVIEW PALETTE.

Create a copy of template file and put your code in its diagram. Input parameters are

- Variables in — it's an array of clusters, with all script's variables and their respective values.
- Error in — for future use

Output values are:

- Variables out — it's an array of clusters with all script's variables and their (new) values.
- Error out — returns VI error after its execution

Notice that custom VIs cannot create new variables; they can only change existing variable's values. If custom VI has a visible Panel and interaction with user is required, you have to set VI's Panel to **modal** behavior.

SAVE FILE

This command save a text into a file

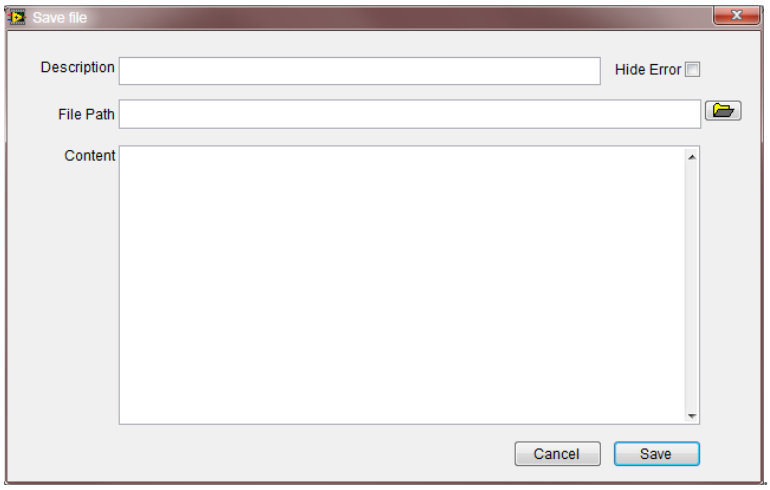


FIGURE 45 – "SAVE FILE" EDITOR WINDOW.

Content field is saved into the file specified by **File path**. Content can contain variable names closed in "<", ">" brackets.

START LABVIEW

This command starts current version of LabVIEW of the same machine where TTM is running.

FIGURE 46 – "START LABVIEW" EDITOR WINDOW.

This command has no parameters. Instead of this command, use System command to run a older version of LabVIEW on machines where multiple versions are installed.

IMPORTANT: This command is available only in TTM standalone application.

SYSTEM

This command executes a system command. TTM waits until *Command* terminates its execution.

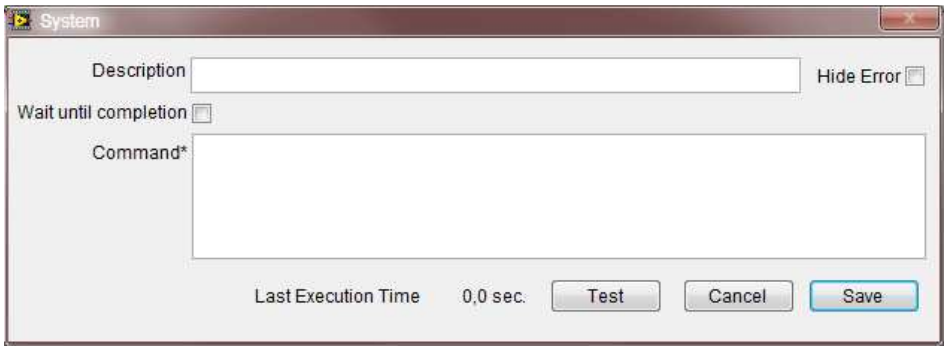


FIGURE 47 – "SYSTEM" EDITOR WINDOW.

If **Wait until completion** is selected, the system-dependent exit code returned by command is used to evaluate if Command completed successfully or not. Exit code equal to zero means no errors. When you have type **Command** field, can use Test button to run system command. Use **Test** button to execute **Command**.

WAIT FOR USER

This command stops script execution until user presses OK button in a message box, displayed with **Message**.

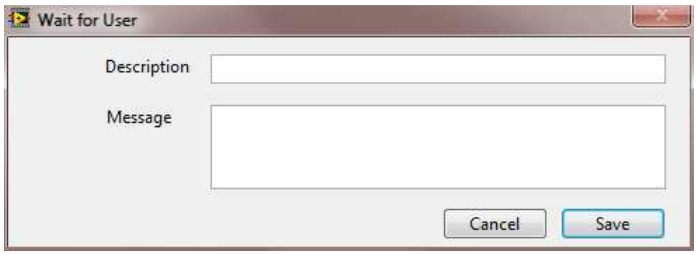


FIGURE 48 – "WAIT FOR USER" EDITOR WINDOW.

WAIT FOR FILE

This command stops script execution until *Target File* performs action specified in *Action* field.

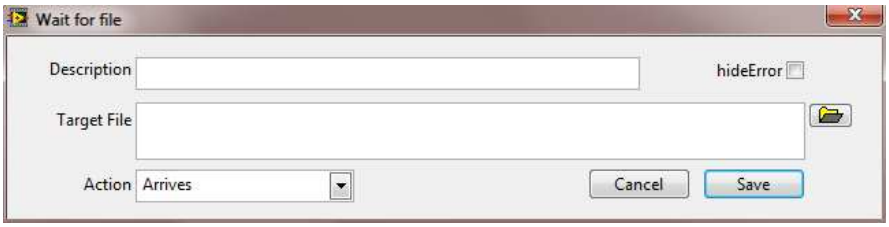


FIGURE 49 – "WAIT FOR FILE" EDITOR WINDOW.

Available actions are:

Action	Description
Arrives	Waits until file is detected
Leaves	Waits until file is move or deleted by third party applications or users

WAIT FOR TIME.

This command stops script execution until specified *Time* is reached. Use *Include date* and *Include time* to determine the exact behaviour of command

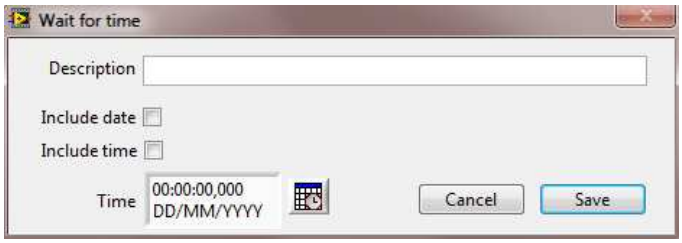


FIGURE 50 – "WAIT FOR TIME" EDITOR WINDOW.

Notice that if **Include date** is checked, command can run only one time and further executions don't wait for specified date and time.

ZIP FOLDER/FILE

This command zips file or folder specified in **Source (file or folder)** field into **Destination File**. If **Destination File** exists, TTM deletes it before creating zip file.

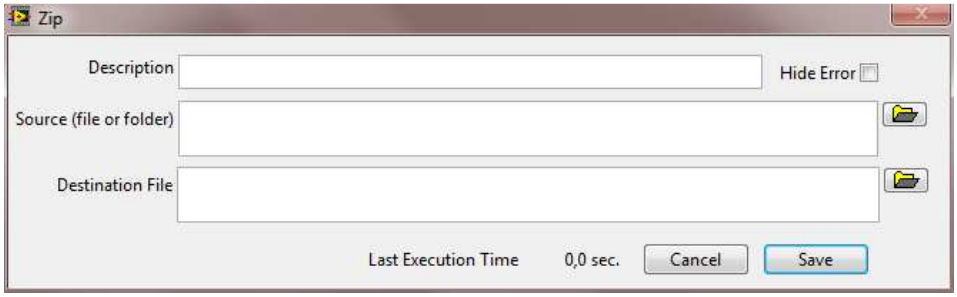


FIGURE 51 – "ZIP" EDITOR WINDOW.

If a zip file with the same name exists, it is deleted before starting creation of new zip file.

RUNNING A SCRIPT

To execute a script you have to right-click on the script's row in script table as show in the following figure:

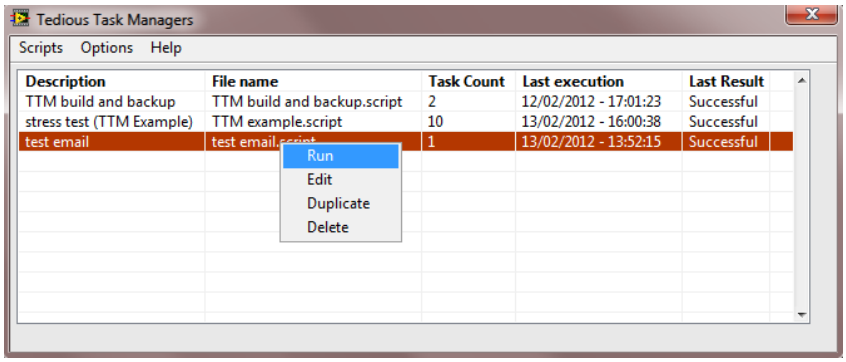


FIGURE 52 – RIGHT-CLICK ON A ROW TO RUN A SCRIPT.

Alternatively you can run a script from **Scripts ▸ Run** menu. When a script is running a log page is displayed as shown below.

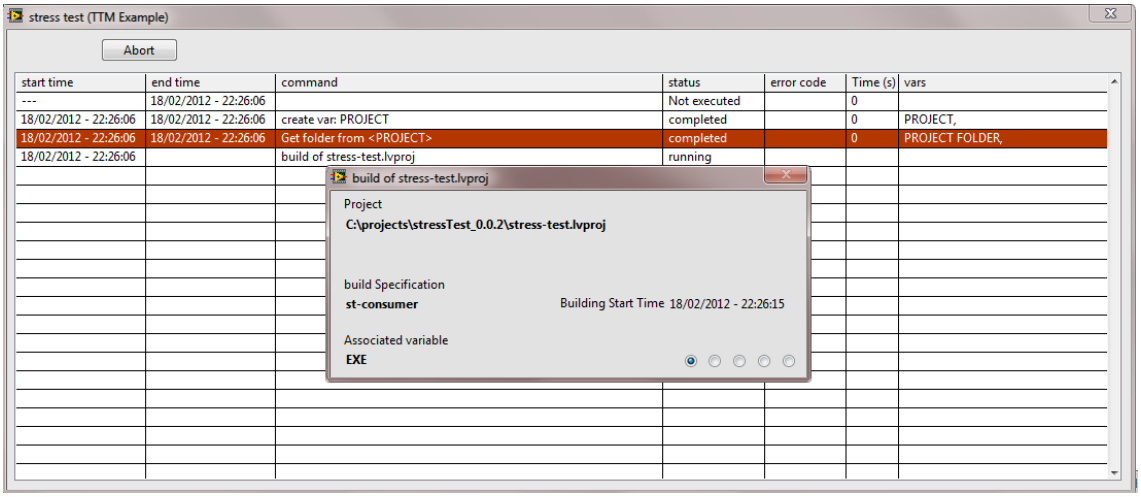


FIGURE 53 – WHEN SCRIPT RUNS, A LOG PAGE DISPLAY PROGRESS OF EXECUTION.

Some commands open a monitor window during their execution. These monitor windows display details about execution status and optionally a progress bar. If a command fails its execution, script aborts and stop executing the commands which follow the one who failed. TTM keeps track of start time and end time for every command and, after first run, uses this information to estimate execution duration for single commands and whole script.

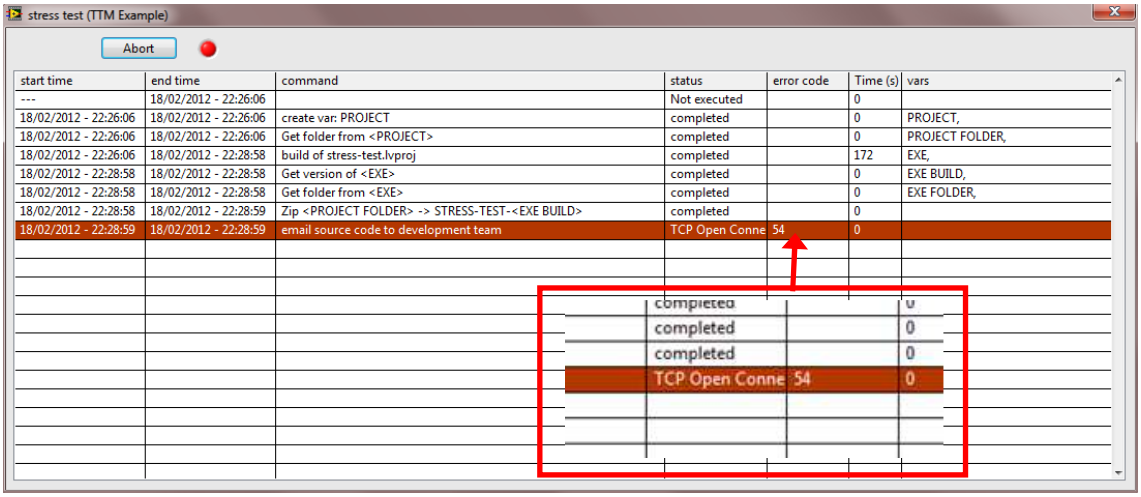


FIGURE 54 – IF A COMMAND FAILS A RED LIGHT SIGNALS THAT SCRIPT EXECUTION HAS BEEN ABORTED.

When a command fails, TTM indicates returned error code in log table as shown above. Error codes refer to LabVIEW error coding tables. Refer to LabVIEW error code help to get more details about errors and their meaning.

PARAMETERS

This window let you specify TTM parameters to send emails, store scripts and log information. To open Parameters window, click on **Option** ▸ **Parameters** for **Options** menu. The following figure appears:

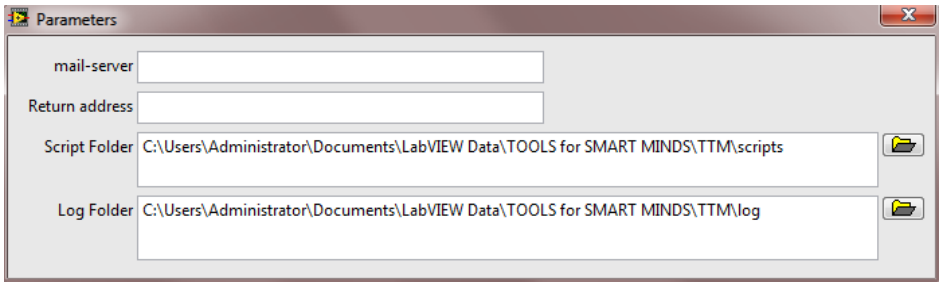


FIGURE 55 – PARAMETERS WINDOW.

Mail-server is your SMTP mail server required to send emails.

Important if mail-server is not properly configured, TTM cannot notify emails at the end of script execution and Email command fails execution.

Return address is sender mail address of all email sent by TTM.

Script Folder is folder where TTM stores script files. At first run, TTM creates a folder named *LabVIEW\TOOLS for SMART MINDS\ scripts* into current user's data folder.

Log folder is folder where TTM store log files. At first run, TTM creates a folder named *LabVIEW\TOOLS for SMART MINDS\ log* into current user's data folder.

ONLINE SCRIPTS LIBRARY

You can browse online script library at the following address:

http://www.toolsforsmartminds.com/products/labview_script_library.php

In this page you find useful scripts created and TTM community that you can download without registration and have to copied into TTM Script Folder.

Script library page can be open also from **Help ▸ Script library (online)** .

FIGURE 56 – ONLINE SCRIPT LIBRARY WEB PAGE.

INDEX