WinActor®

WinActor Note

Terminal Function Scenario Creation Manual

NTT ADVANCED TECHNOLOGY CORPORATION

Copyright © 2013-2025 NTT, Inc. & NTT ADVANCED TECHNOLOGY CORPORATION

Contents

1. Introduction

1.1. About this document

This document is the manual for creating scenarios for terminal functions that work in cooperation with WinActor Note.

The following terminal functions are provided:

- Windows shell function
  Executes commands on Windows PowerShell or Command Prompt

- SSH/SCP client functions
  Connects to, executes commands on, and disconnects from an SSH server
  Sends/receives a file with an SSH server (SCP)

- Telnet client function
  Connects to, executes commands on, and disconnects from a Telnet server

By using the terminal functions in combination with the macro functions of WinActor Note, it is possible to perform detailed controls according to the result of executing commands.

In Chapter 2, you will create a scenario to execute commands by connecting to an SSH server built on CentOS and get log files according to the output result of the commands. Through the tutorial in this manual, you can learn the terminal functions of WinActor Note.

For how to use WinActor Note, see the materials No.3 and 4 in Table 5-1. Reference materials.

1.2. Trademarks

The names described below and other names of companies and products in this document are trademarks or registered trademarks of their respective companies. The TM, ®, and © marks are omitted in this document.

・Windows and Windows PowerShell are trademarks or registered trademarks of Microsoft Corporation in the United States and other countries.
* The official name of Windows is Microsoft Windows Operating System.

・The name of CentOS is a trademark or registered trademark of CentOS ltd.
・Mac OS and OS X are trademarks of Apple Inc.
・Linux is a trademark or registered trademark of Mr. Linus Torvalds in Japan and other countries.

1.3. Notes on this manual

・The copyright notice "Copyright © 2013-2025 NTT, Inc. & NTT ADVANCED TECHNOLOGY CORPORATION" attached to this manual and the provided software cannot be changed or deleted.
The copyright of this manual belongs to NTT, Inc. and NTT ADVANCED TECHNOLOGY CORPORATION.
・The descriptions in this manual assume that users understand Windows operations and functions. For information that is not described in this manual, see the documents provided by Microsoft.

2. Scenario creation tutorial

2.1. Preparation

The environment used in this tutorial is shown in Table 2-1. Environment used in this tutorial. It is assumed that the SSH protocol connection confirmation has been completed in advance between the SSH server and the computer on which a WinActor scenario runs.

Table 2-1. Environment used in this tutorial

Target	Item	Description	Settings in this tutorial
Server	Server OS	CentOS Linux release 7.6.1810	-
	SSH server	OpenSSH 7.4p1	IP address: 192.168.56.2 Port number: 22
	Operation verification account	Any	Username: user Password: ax12bc=9
Computer on which WinActor runs	IP address	Any	IP address: 192.168.56.1
	Scenario folder	Any	C:\Terminal_function_scenario
	Password file	Any However, store the file in the scenario folder.	Store the file in the scenario folder with the password: ax12bc=9 and the filename: cihperPassword.json. For details, see 2.3.
	Windows	Edition: Windows 10 Pro Japanese version	

2.2. How to create a scenario

There are two ways to create a scenario using the terminal functions.

① Create a scenario using the user libraries only
② Use the macro functions of WinActor Note

As in the scenario you will create in Section 2.3, when processing operations continuously on WinActor Note, you can efficiently create the scenario by recording a macro on WinActor Note, checking the operations, and then running the created macro using the "Note_ReadAndRunMacro" library.

2.3. Scenario creation procedure

Details and operations of the scenario to be created are shown below.

[Details of the scenario to be created (①② in Figure 2-1)]

- Open the SSH client, and if a file with the extension "log" exists under the /home/user/log folder (⑦ in Figure 2-1), get the file with the extension "log" (⑥ in Figure 2-1).
- Save the SSH client operation as a log (⑤ in Figure 2-1).
- Create a folder with the name including date and time (④ in Figure 2-1) on the computer on which WinActor runs and store the log.

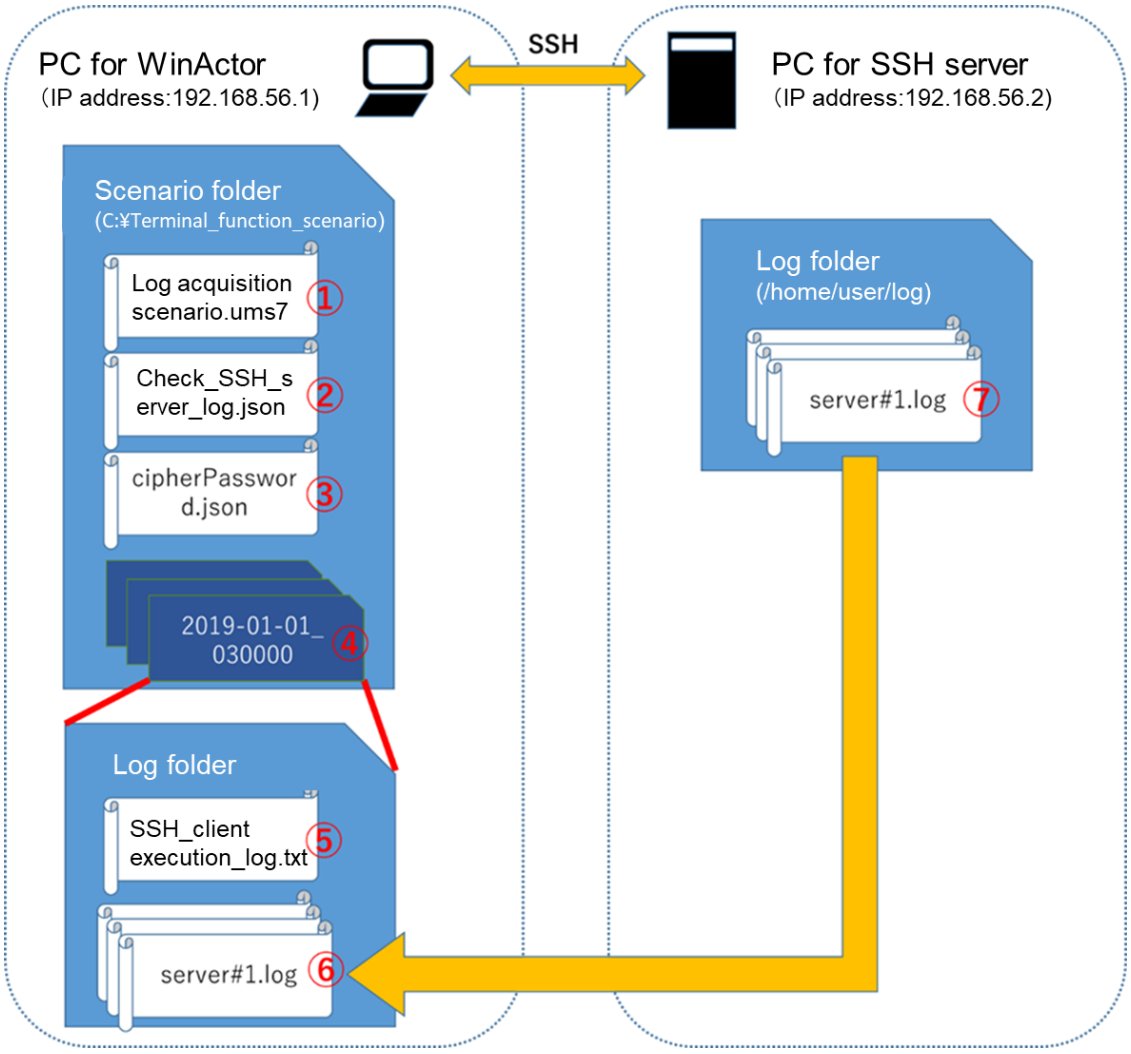


Figure 2-1. Scenario operations

Table 2-2. Description of each file and folder

No. in Figure 2-1	File or folder	Description
①	Log acquisition scenario.ums7	This is a WinActor scenario to be created in this tutorial. Here, the scenario with WinActor Ver.7 is used as an example.
②	Check_SSH_server_log.json	This is a WinActor Note macro to be created in this tutorial.
③	cipherPassword.json	This is a password file for logging in to the SSH server. Create this in advance by referring to the section "Password file generation tool" in the material No.3 in Table 5-1.
④	2019-01-01_030000, etc.	This is a folder for storing logs created by running the scenario of ①. This folder will be created on the computer on which WinActor runs. The date and time when the folder was created becomes the folder name. In the case of the left, it means 3:00:00 on January 1, 2019.
⑤	SSH_client_execution_log.txt	The contents displayed in WinActor Note when running the macro of ② are saved in this file. In the scenario created in this tutorial, the login message to the SSH server and the execution and result of the commands remain as information.
⑥	server#1.log, etc.	This is a file with the extension "log" acquired from the computer that runs the SSH server by running the scenario of ①.
⑦	server#1.log, etc.	This is a file with the extension "log" that exists on the computer that runs the SSH server. The scenario to be created in this tutorial aims to get this file. If the file does not exist on the SSH server, create the file in advance. Text data of less than 10 characters is used as the content of the file.

In the scenario to be created in this tutorial, you will use the nodes and user libraries of WinActor. For details of the nodes and user libraries in the scenario, see the materials No.1 and 2 in Table 5-1.

In addition, general Linux commands are used in the scenario to be created. For details on each command, see Linux-related documents and web sites.

2.4. Scenario overview

The overview of how to create the scenario is as follows.

1. Creating the Creating macro (see Section 2.5)
   A) Create the processing of getting a value that can determine whether the file with the extension "log" exists in the log folder on the SSH server and loading it into the WinActor variable. At that time, create the macro with WinActor Note launched and with the macro recorded.
2. Creating the Creating the WinActor scenario (see Section 2.6)
   A) Add the processing of calling the created macro from the WinActor scenario.
   B) Add the processing of judging whether the file with the extension "log" exists.
   C) Add the processing of receiving the file with the extension "log" when the file with the extension "log" exists.
   D) Add the processing of creating the folder to save logs and moving the log files.

2.5. Creating the WinActor Note macro

For the operation of WinActor Note, see the material No.3 in Table 5-1.

Launch WinActor Note and enable the macro recording.

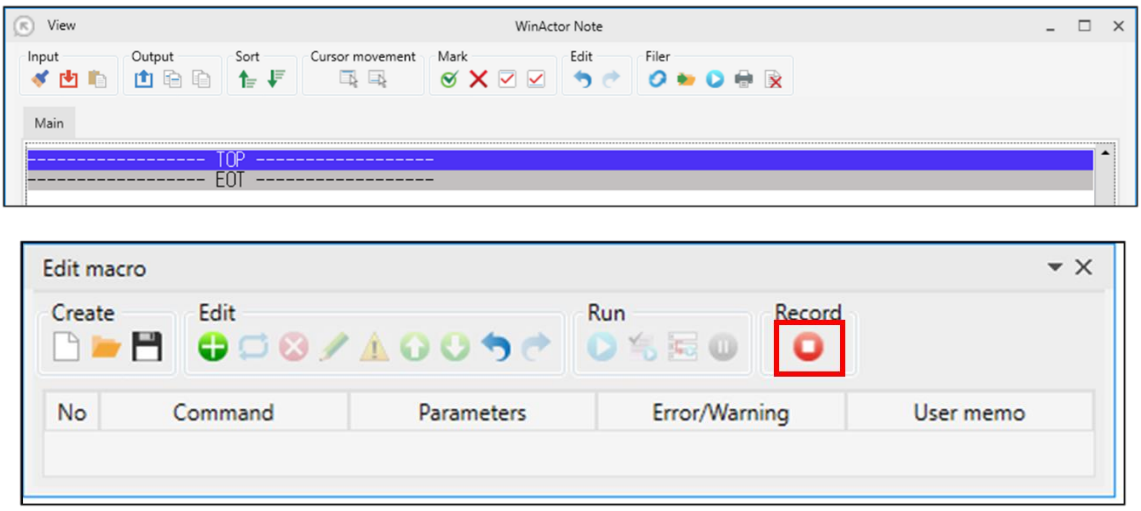


Figure 2-2. WinActor Note at the start of macro creation

2.5.1. Clearing the data on the WinActor Note window

Clear the text data on the WinActor Note window so that the previous execution logs do not remain in WinActor Note.

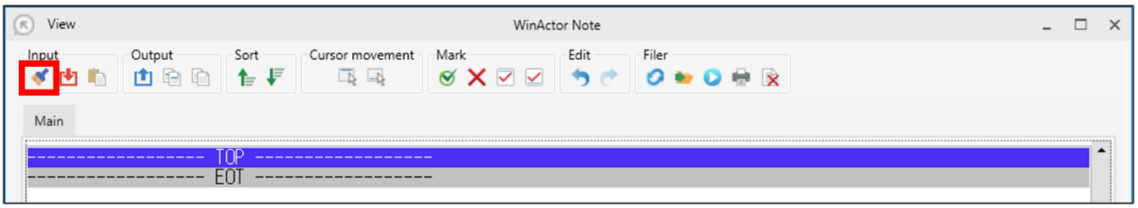


Figure 2-3. Clearing WinActor Note

2.5.2. Processing using "SSH tool"

The processing using "SSH tool" is shown below.

1. Opening the SSH client

   Select "SSH tool" in the "Terminal" menu of WinActor Note.

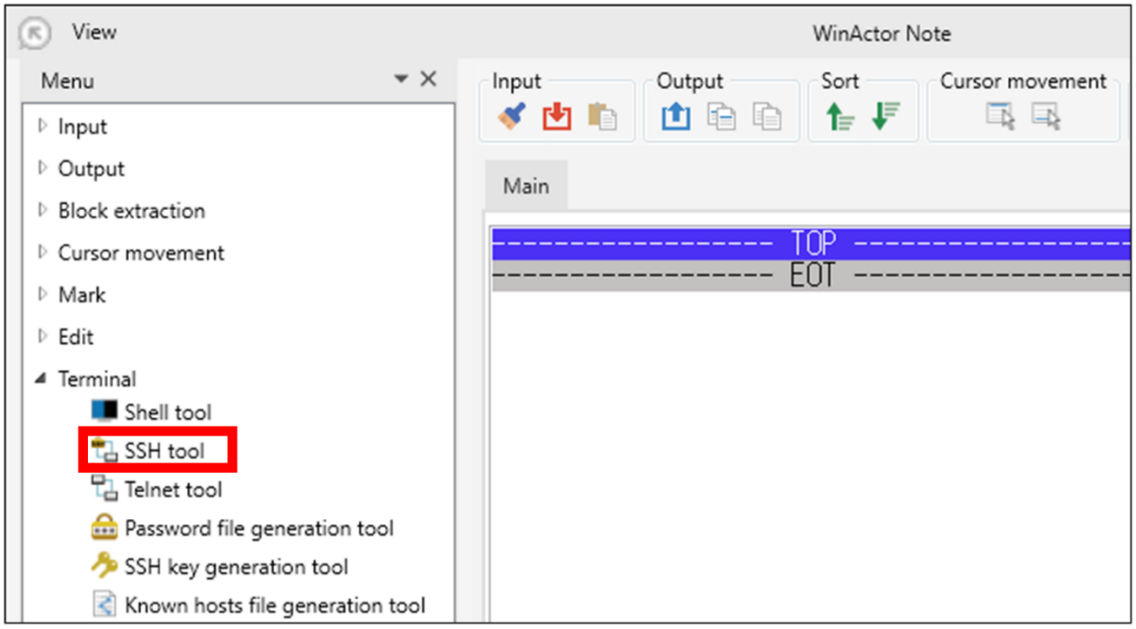


Figure 2-4. Opening "SSH tool"

Set the parameters in the area surrounded by the red frame ① in Figure 2-5, and click the "Open SSH session" button (the red frame ②). The input value of the password file is "C:\Terminal_function_scenario\cipherPassword.json."

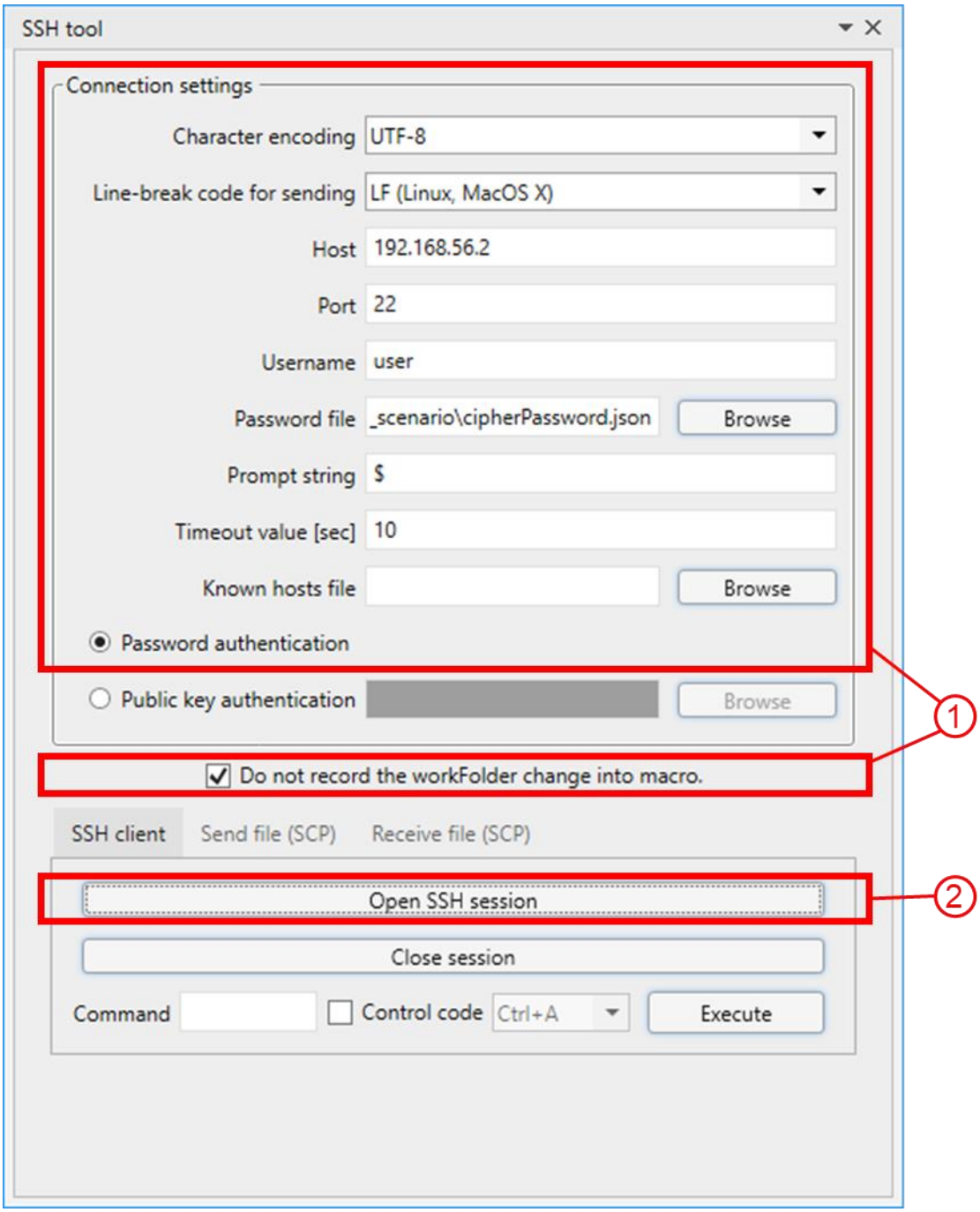


Figure 2-5. Opening the SSH client

2. Executing commands

Enter the commands written under [Command to be entered] in the following A) to D) into ① in Figure 2-6, and click the Execute button (② in Figure 2-6).

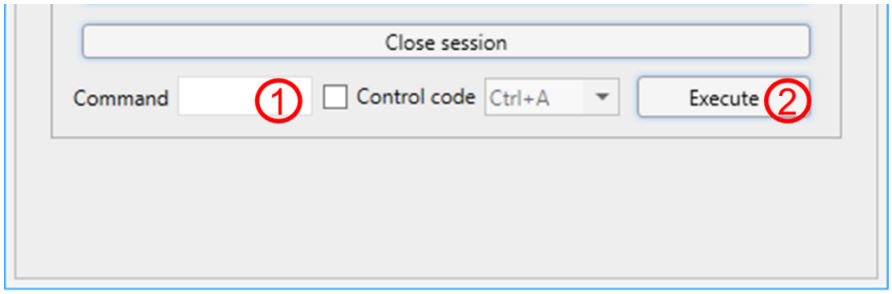


Figure 2-6. Executing commands

A) Move to the log folder

[Command to be entered]

cd log

B) Get the date

[Command to be entered]

date

C) Display the file with the extension "log"

[Command to be entered]

ls *.log

D) Display the result of executing the command C)

[Command to be entered]

echo $?

[Note]

If you execute the commands at an interval from "1. Opening the SSH client," the command execution may fail. In that case, click the "New" button in the "Create" menu of the "Edit macro" pane of WinActor Note and try again from Section 2.5.1.

3. Closing the SSH client.

   Click the "Close session" button in Figure 2-7 to close the SSH client.

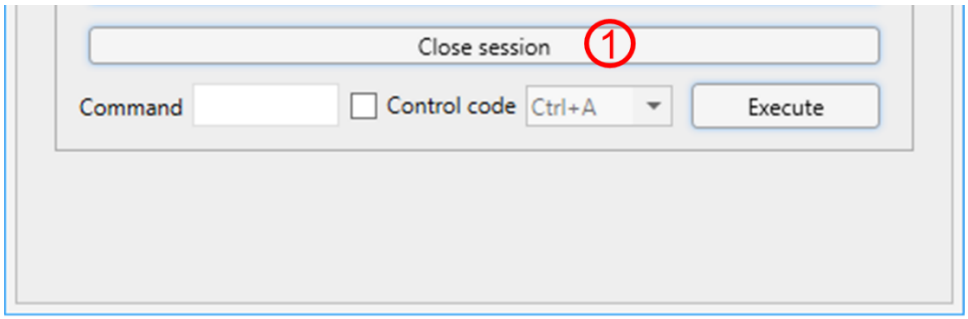


Figure 2-7. Closing the SSH client

When ⑦ in Table 2-2 is server#1.log,server#2.log, the result of executing up to this point should be displayed in WinActor Note as shown in Figure 2-8.

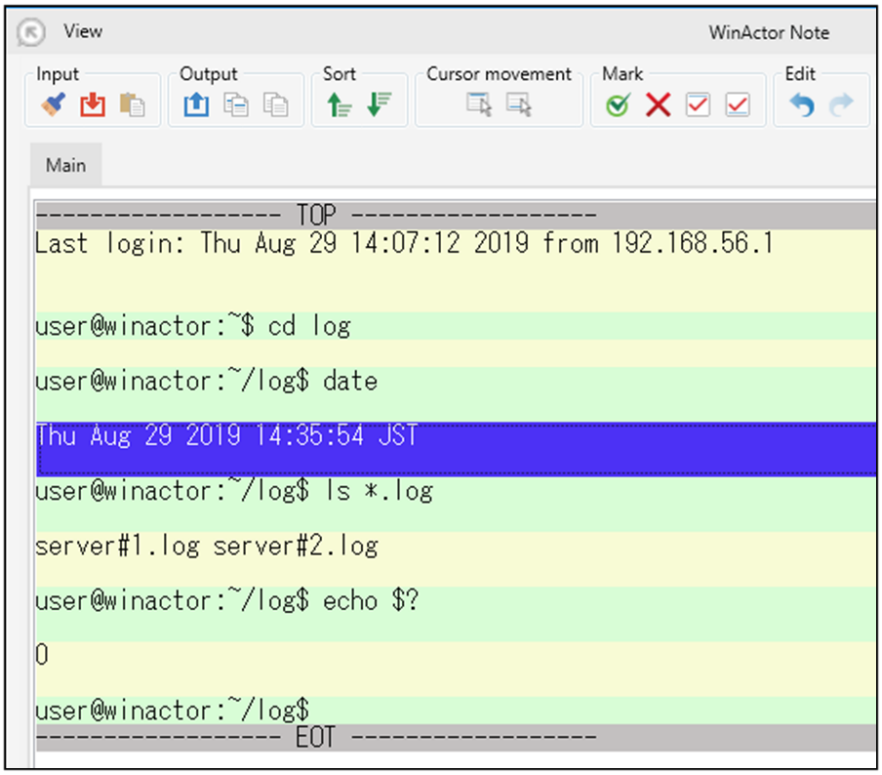


Figure 2-8. Result of executing up to Section 2.5.2

2.5.3. Text processing

Follow the steps below to select the result of the existence of files with the extension "log."

1. Removing unnecessary spaces by using "White space removal tool" in the "Edit" menu
   Select "All blocks" for the target and "Remove line breaks" for the operation (① in Figure 2-9), and click the "Execute" button (② in Figure 2-9).

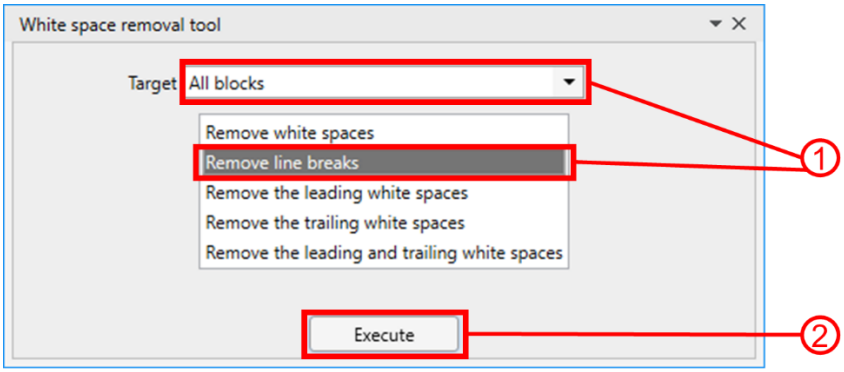


Figure 2-9. Executing "White space removal tool"

2. Selecting the result of executing "echo $?" by using "Block search tool" and "Select the next block"
   ① Click "Block search tool" in the "Block extraction" menu, enter "$?" for the keyword, select "Contain" in the drop-down list and "Forward search (cursor moves down)" for the operation (① in Figure 2-10), and click the "Execute" button (② in Figure 2-10).

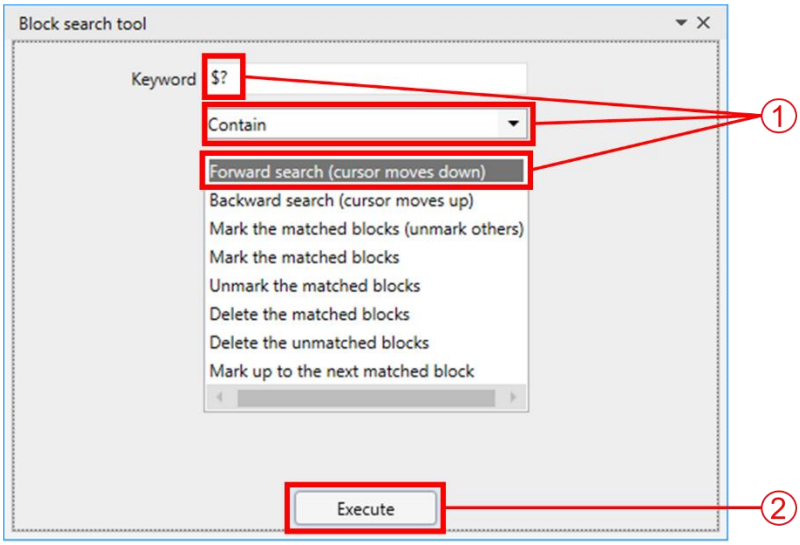


Figure 2-10. Search by "Block search tool"

② Click "Cursor movement tool" in the "Cursor movement" menu, specify "Select the next block" and click the "Execute" button (Figure 2-11).

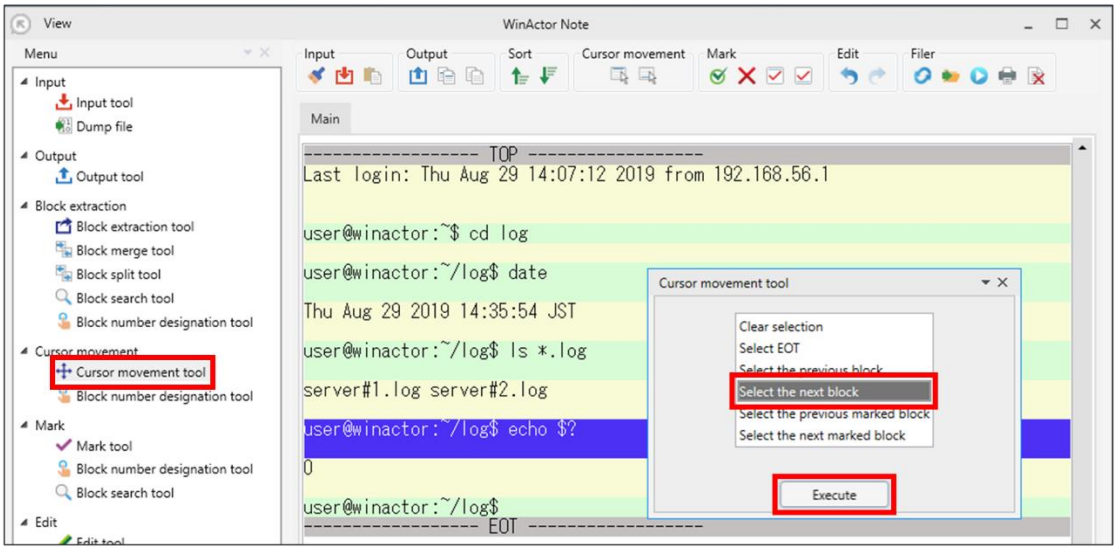


Figure 2-11. Executing "Select the next block"

The result of executing up to this point should be displayed as shown in Figure 2-12.

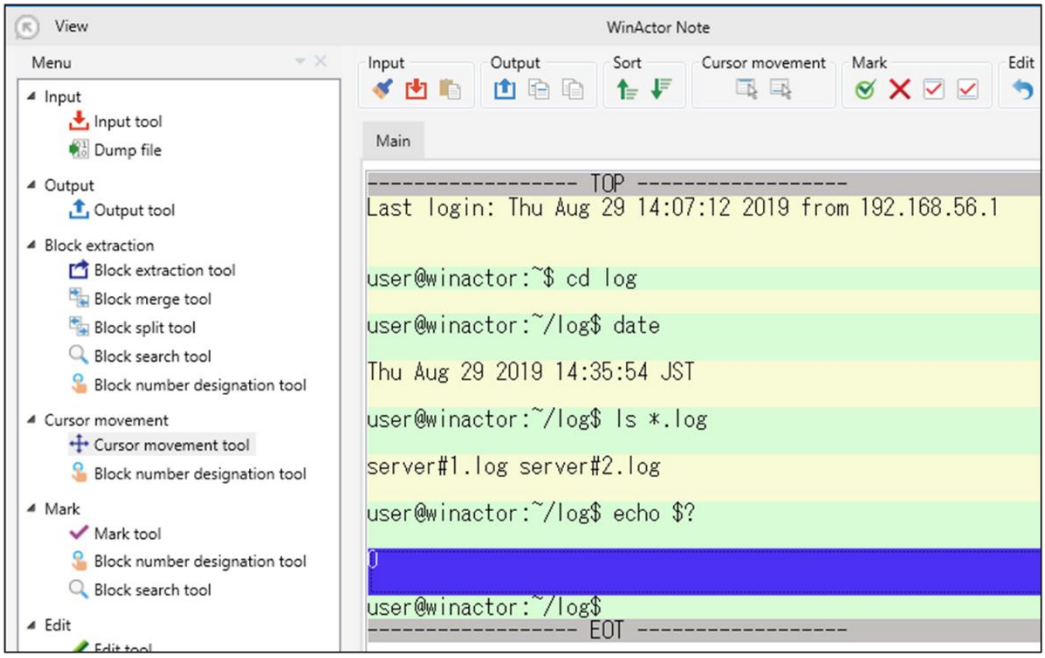


Figure 2-12. Result of executing Section 2.5.3

2.5.4.　Saving the text data on the WinActor Note window to a file

Save the text data on the WinActor Note window as a log.

Click "Output tool" in the "Output" menu, set the values according to Table 2-3 and click the "Execute" button.

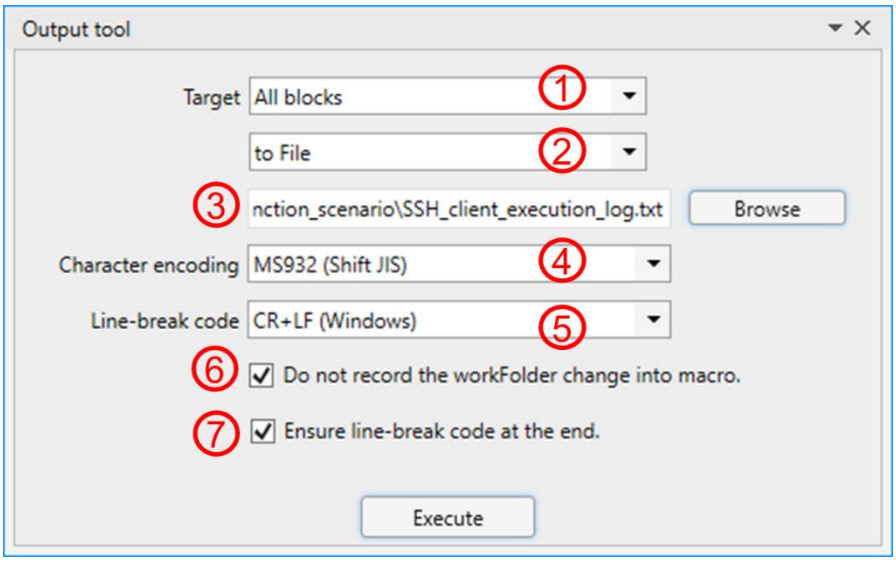


Figure 2-13. Settings in the "Output tool" window

Table 2-3. "Output tool" settings

No. in Figure 2-13	Setting value
①	Select "All blocks"
②	Select "to File"
③	Enter "C:\Terminal_function_scenario\SSH_client_execution_log.txt"
④	Select "MS932(Shift JIS)"
⑤	Select "CR+LF(Windows)"
⑥	Check the box
⑦	Check the box

2.5.5. Saving the macro

In the "Edit macro" pane of WinActor Note, (if the pane is not displayed, open it from the "View" menu), disable the macro recording (① in Figure 2-14), and click the "Save" button (② in Figure 2-14).

Save the macro currently being edited (④ in Figure 2-14) to the folder "C:\Terminal_function_scenario" with the filename "Check_SSH_server_log.json" (③ in Figure 2-14).

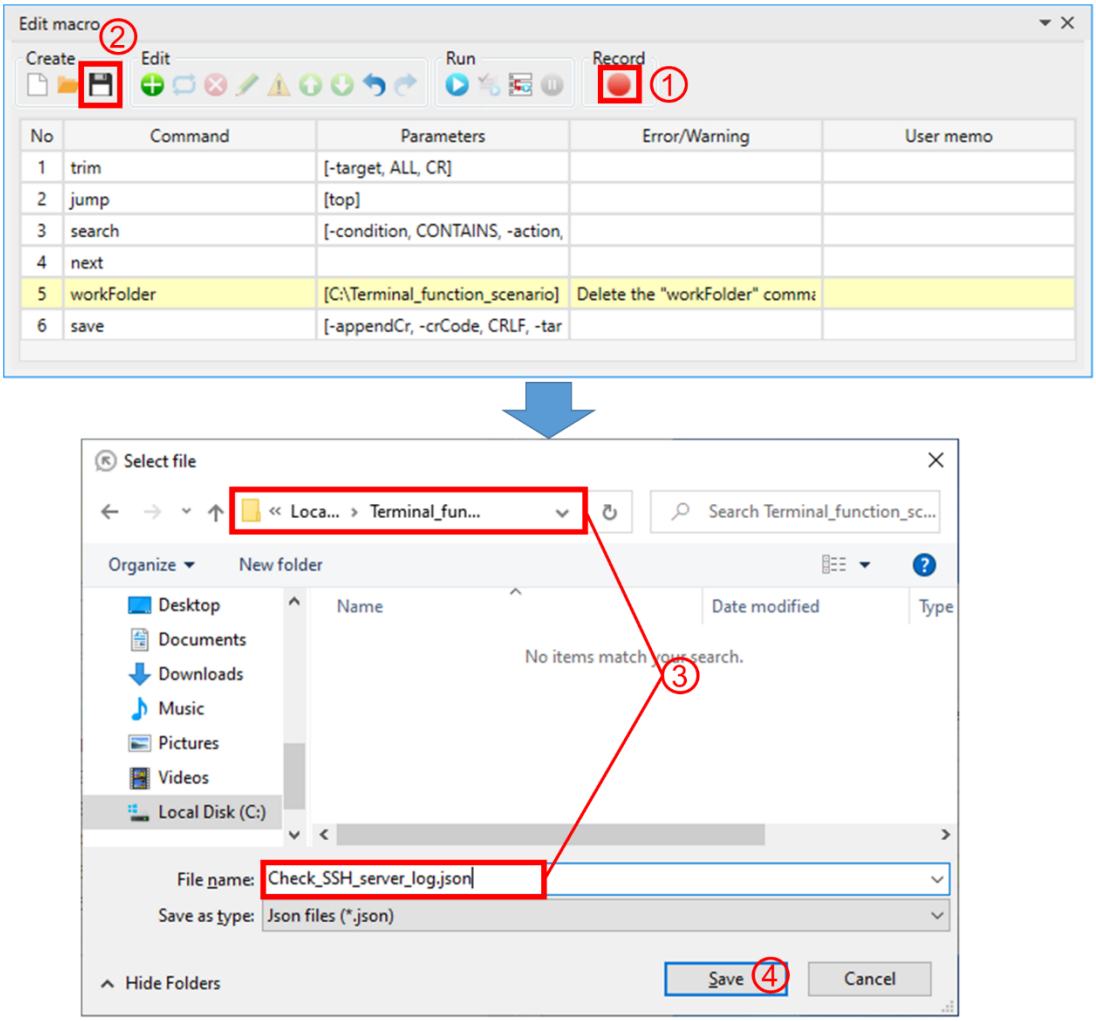


Figure 2-14. Saving the macro

2.6. Creating the WinActor scenario

Save the WinActor scenario to be created this time as a new scenario file.

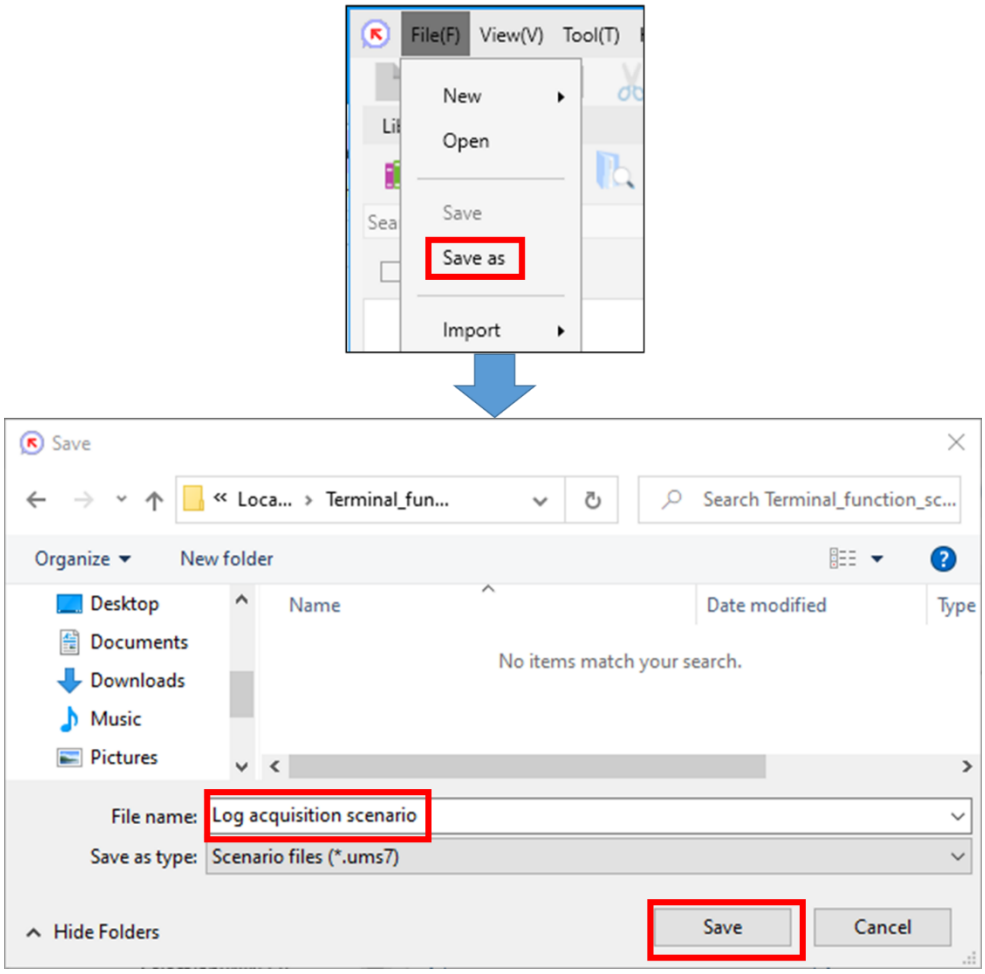


Figure 2-15. Saving the new scenario file

2.6.1. Calling the created macro

Drag and drop the "Note_ReadAndRunMacro" library into the Scenario box in the flowchart area of WinActor.

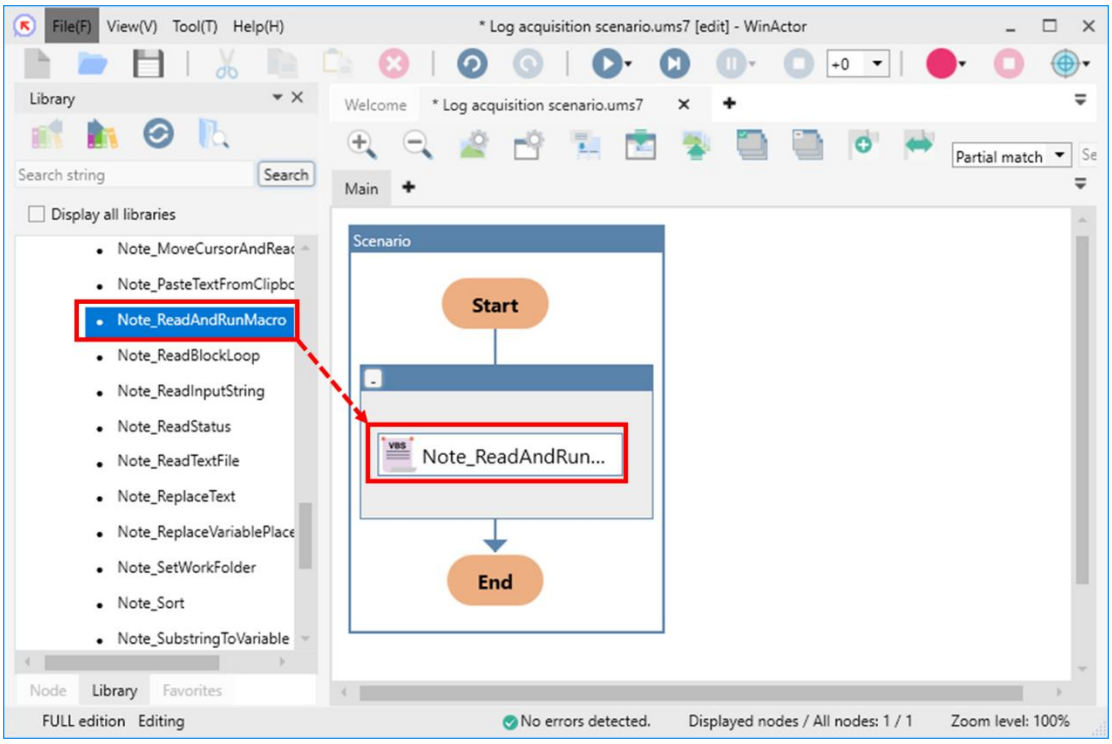


Figure 2-16. Placing the "Note_ReadAndRunMacro" library

Double-click the placed library to open the Property window.

In the macro filename field, enter the same filename (Check_SSH_server_log.json) of the macro created in Section 2.5.5, and click the "Update" button.

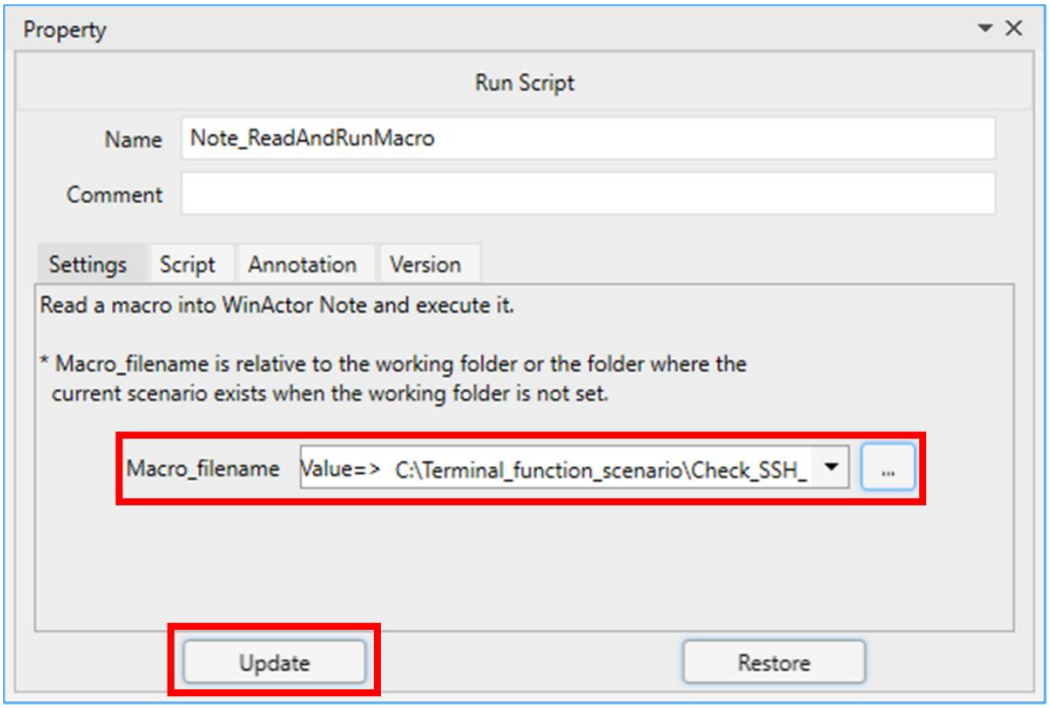


Figure 2-17. Specifying the macro filename

Check whether it works properly with the "Note_ReadAndRunMacro" library alone.

Click the "Run scenario" button on the toolbar of WinActor and check that the expected text contents are output on WinActor Note. After checking the operation, delete the "C:\Terminal_function_scenario\SSH_client_execution_log.txt" file.

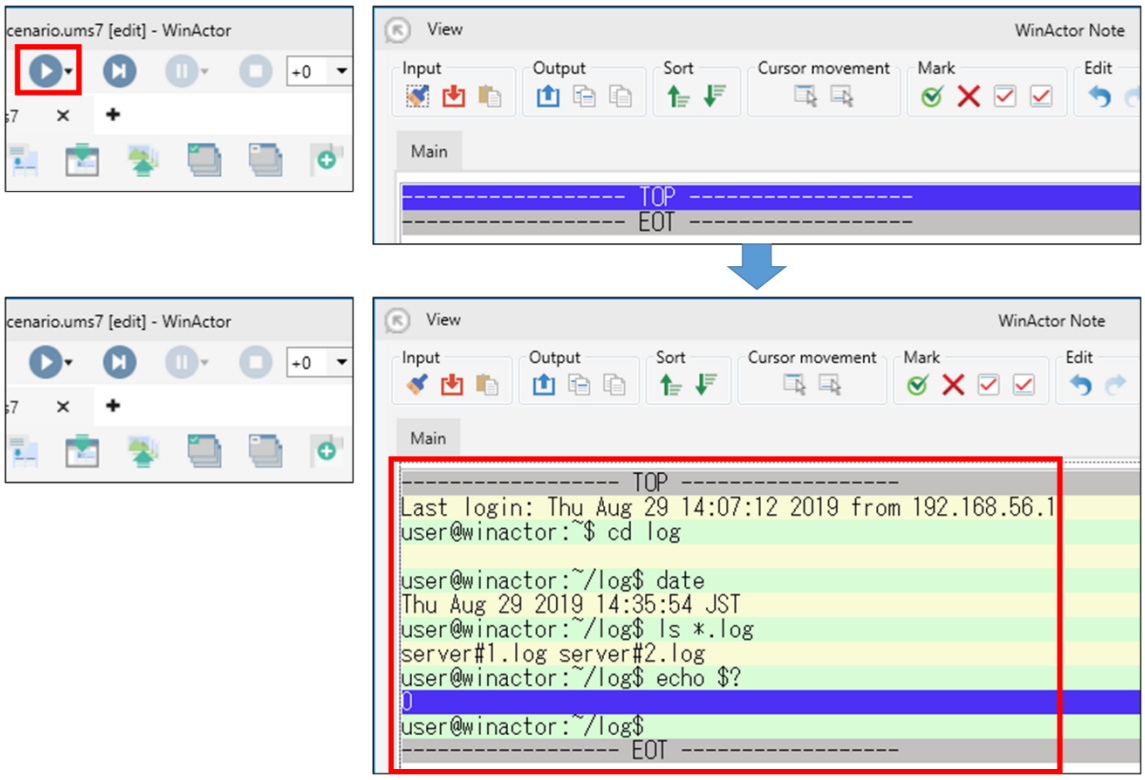


Figure 2-18. Checking the operation of the "Note_ExecuteMacroLoading" library

2.6.2. Adding the processing of judgment

Add the processing of reading the text on WinActor Note and saving it as a variable in WinActor.

Drag and drop "Note_TextToVariable" from the user library.

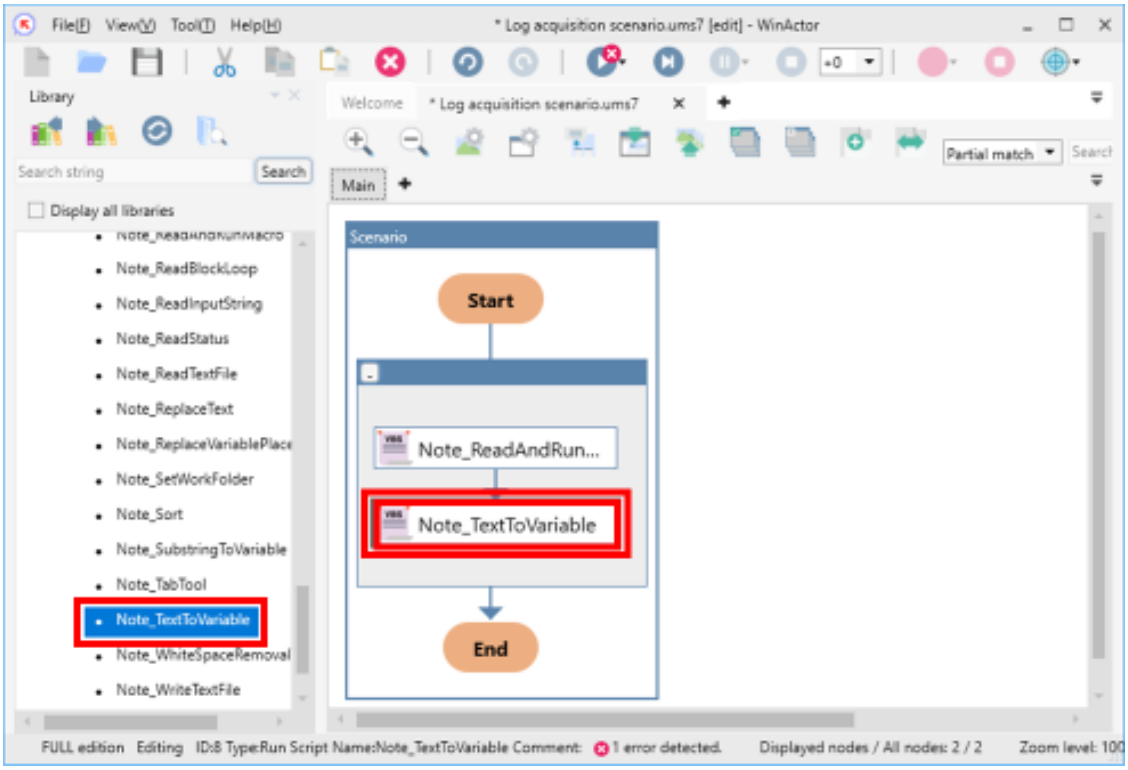


Figure 2-19. Placing the "Note_TextToVariable" library

After placing the library, double-click it to open the Property window.

To select the result of executing "echo $?" after running the "Note_ReadAndRunMacro" library and import the content of the selected result as a variable on WinActor, define the variable name as "echo_execution_result" and click the "Update" button.

A window for confirming whether to register the entered name as a new variable will appear. Click "Yes," and it will be registered as a new variable in the Variable list pane of WinActor.

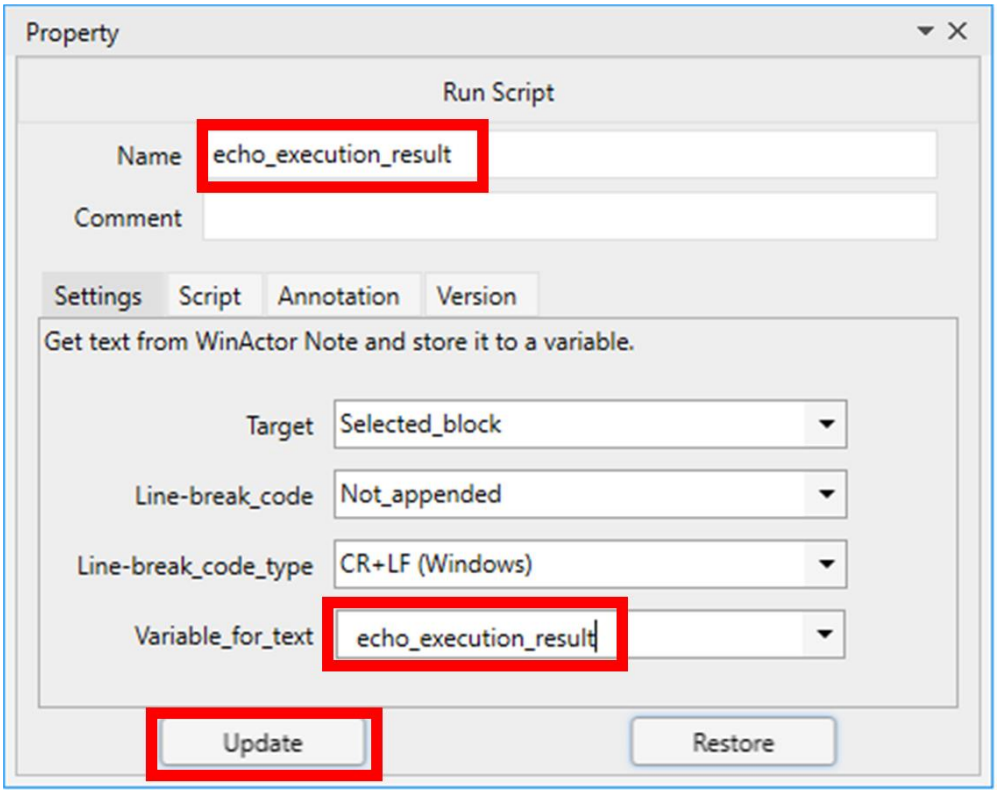


Figure 2-20. Entering the variable name

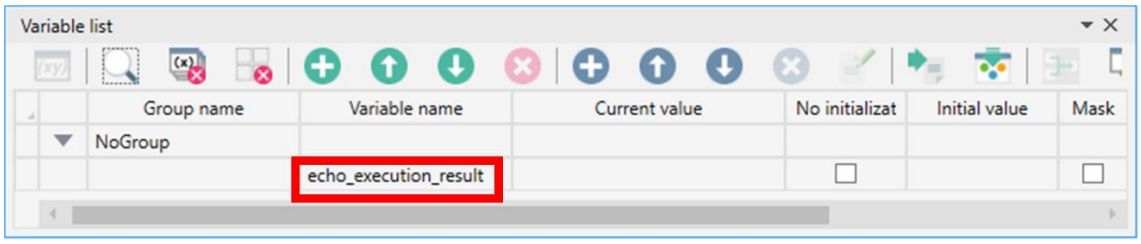


Figure 2-21. Confirming the update of the Variable list pane

Check the operation that the string resulted from executing "echo $?" will be imported into the variable "echo_execution_result."

If the "Run scenario" button is clicked in the current scenario state, the variable value will be initialized after all the processing is completed. To prevent this from happening, place "Waiting Dialog" at the end of the scenario.

Drag and drop "Waiting Dialog" from the Node tab.

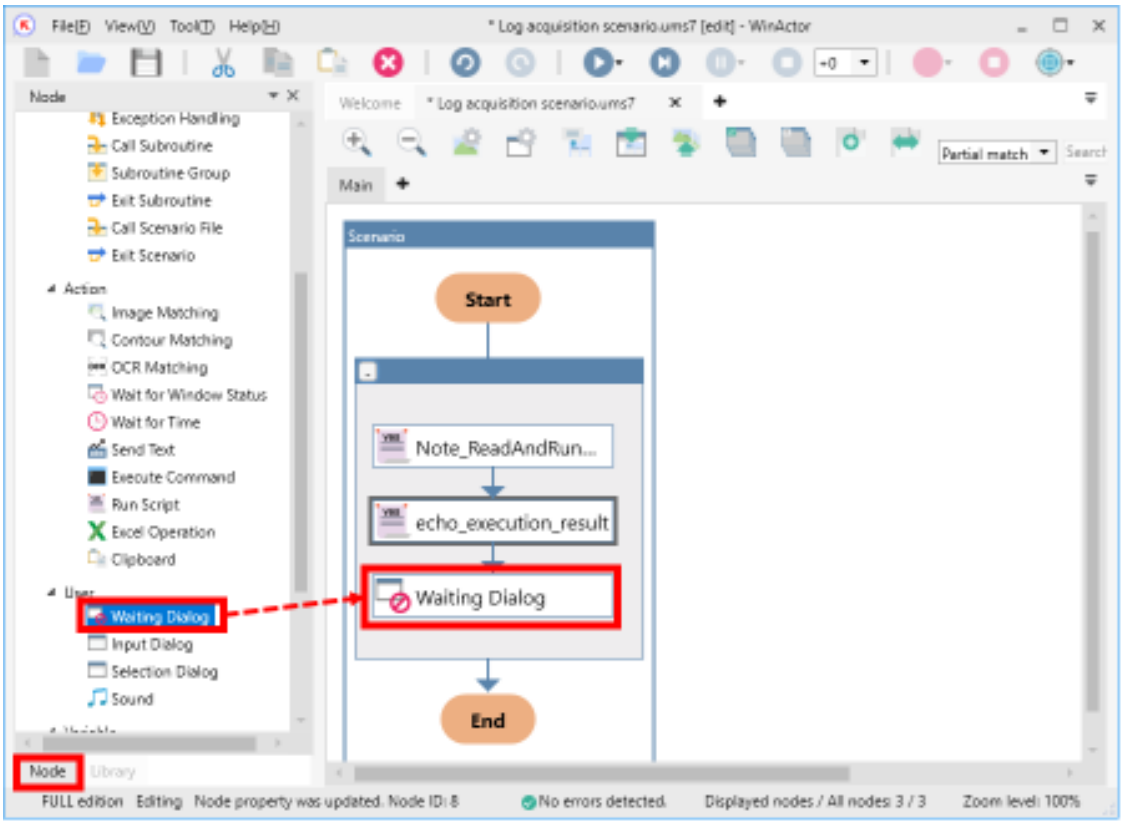


Figure 2-22. Placing "Waiting Dialog"

Run the current scenario to check the operation.

Click the "Run scenario" button on the toolbar of WinActor, and check whether the current value for the "echo_execution_result" in the Variable list pane is updated with the text resulted from executing "echo $?" ("0" in this case) entered.

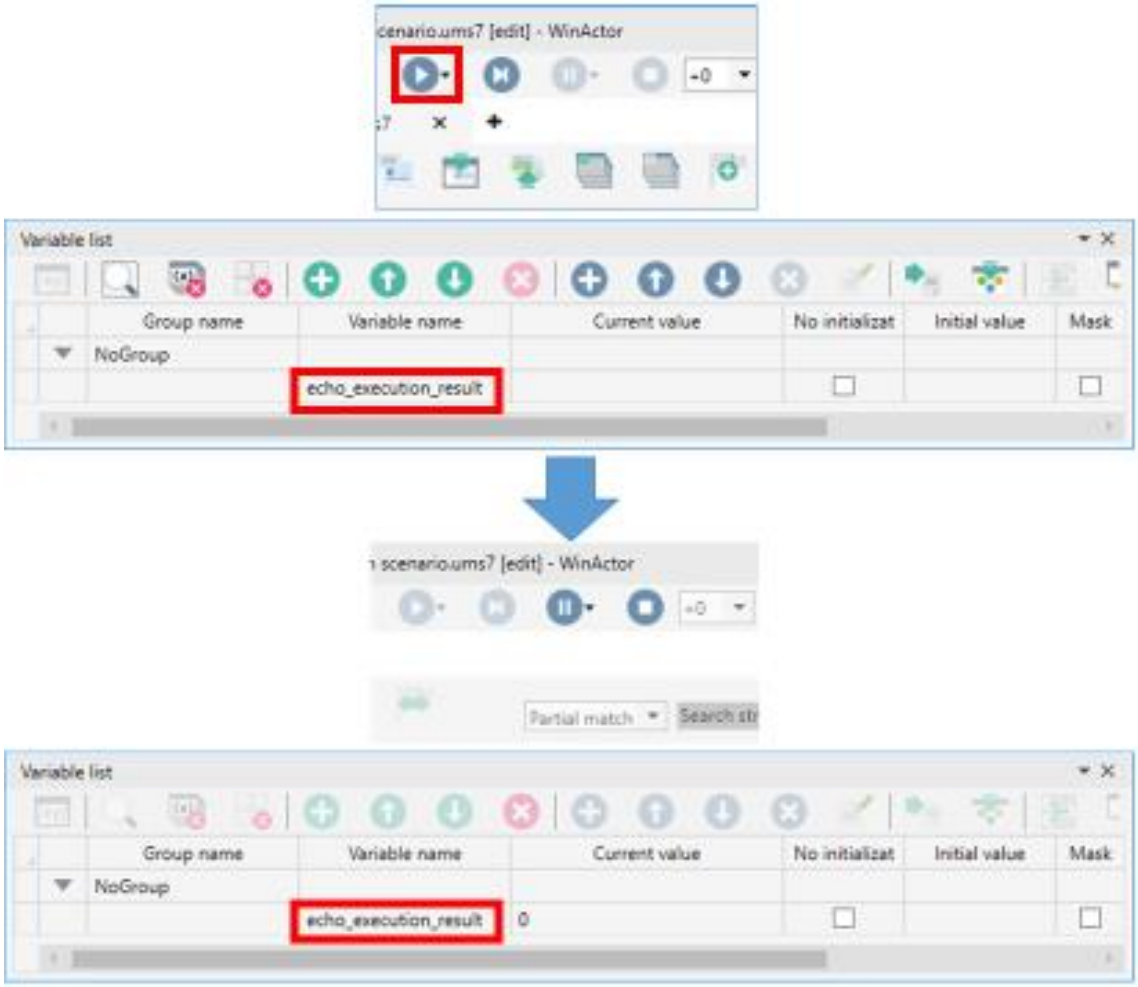


Figure 2-23. Checking the operations of importing the value into the variable

Next, delete "Waiting Dialog" you have added, and add the processing of judgment.

Drag and drop "Decision" from the Node tab.

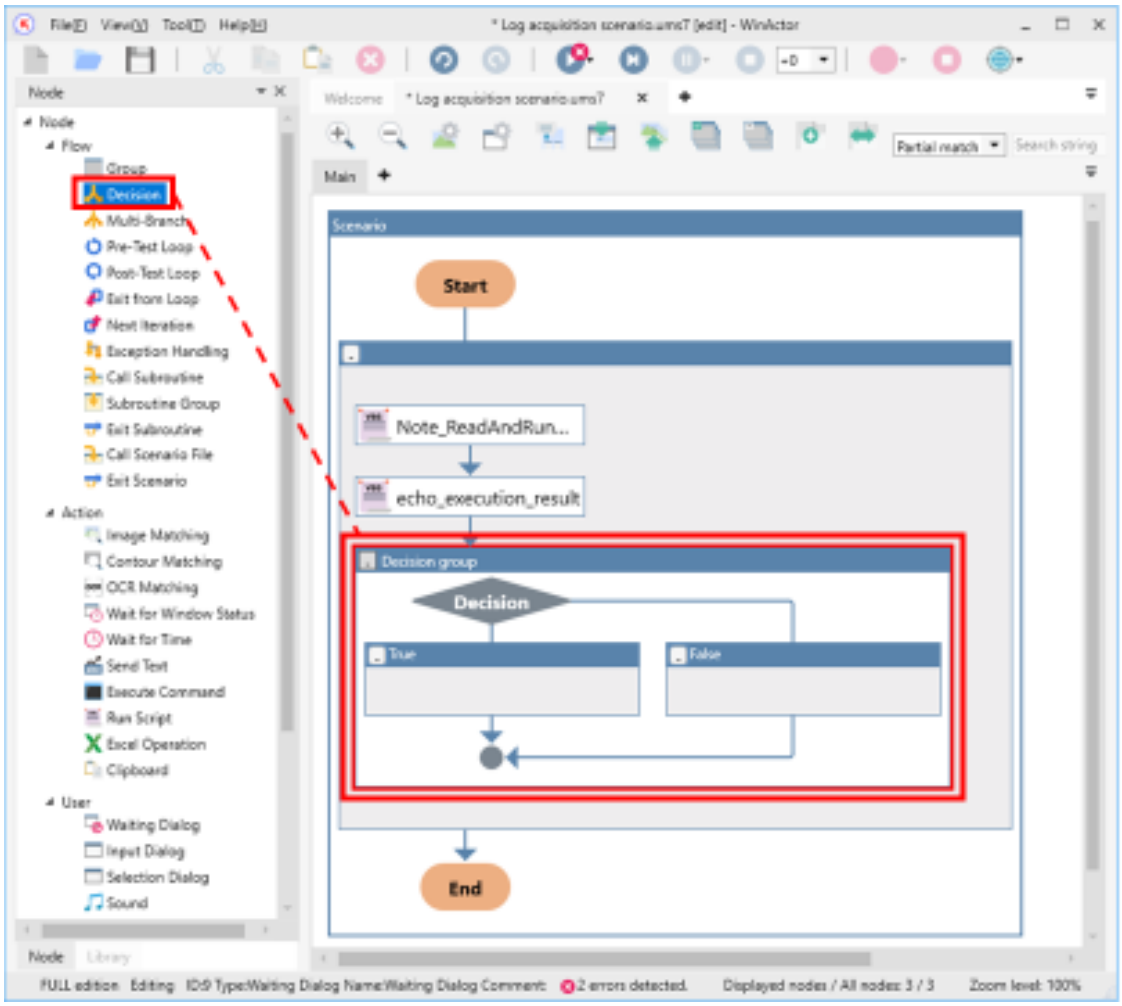


Figure 2-24. Placing the "Decision" node

After placing the node, double-click it to open the Property window.

After changing the name and the branch name of the "Decision" property window, click "Edit," and in the "Conditional expression" property window, select "echo_execution_result" for the left side of the conditional expression and enter 0 on the right side of the conditional expression. As a result of these settings, if the result of executing "echo $?" is 0, that means if the file with the extension "log" exists, it will move to the operation with the branch named "true."

Lastly, click the "Update" button on the property windows of "Conditional expression" and "Decision" to complete the property settings.

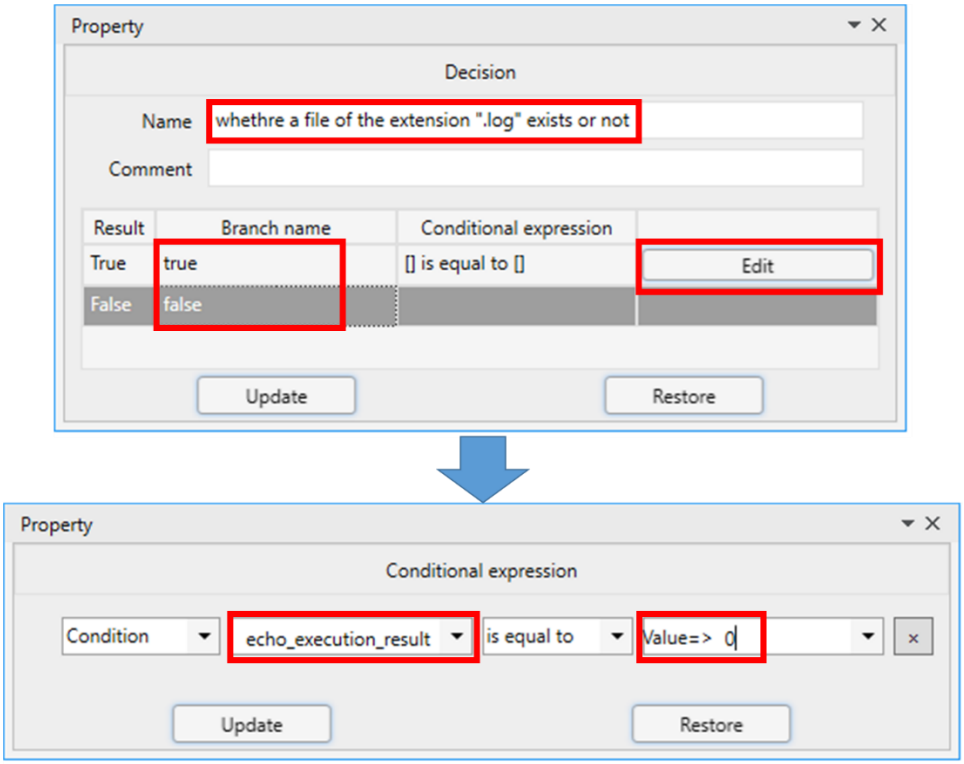


Figure 2-25. Conditional expression settings

2.6.3. Adding the processing of receiving the file

Add the processing of receiving the file with the extension "log" if the file with the extension "log" exists.

Drag and drop "Note_ReceiveFile(SCP)" from the user library.

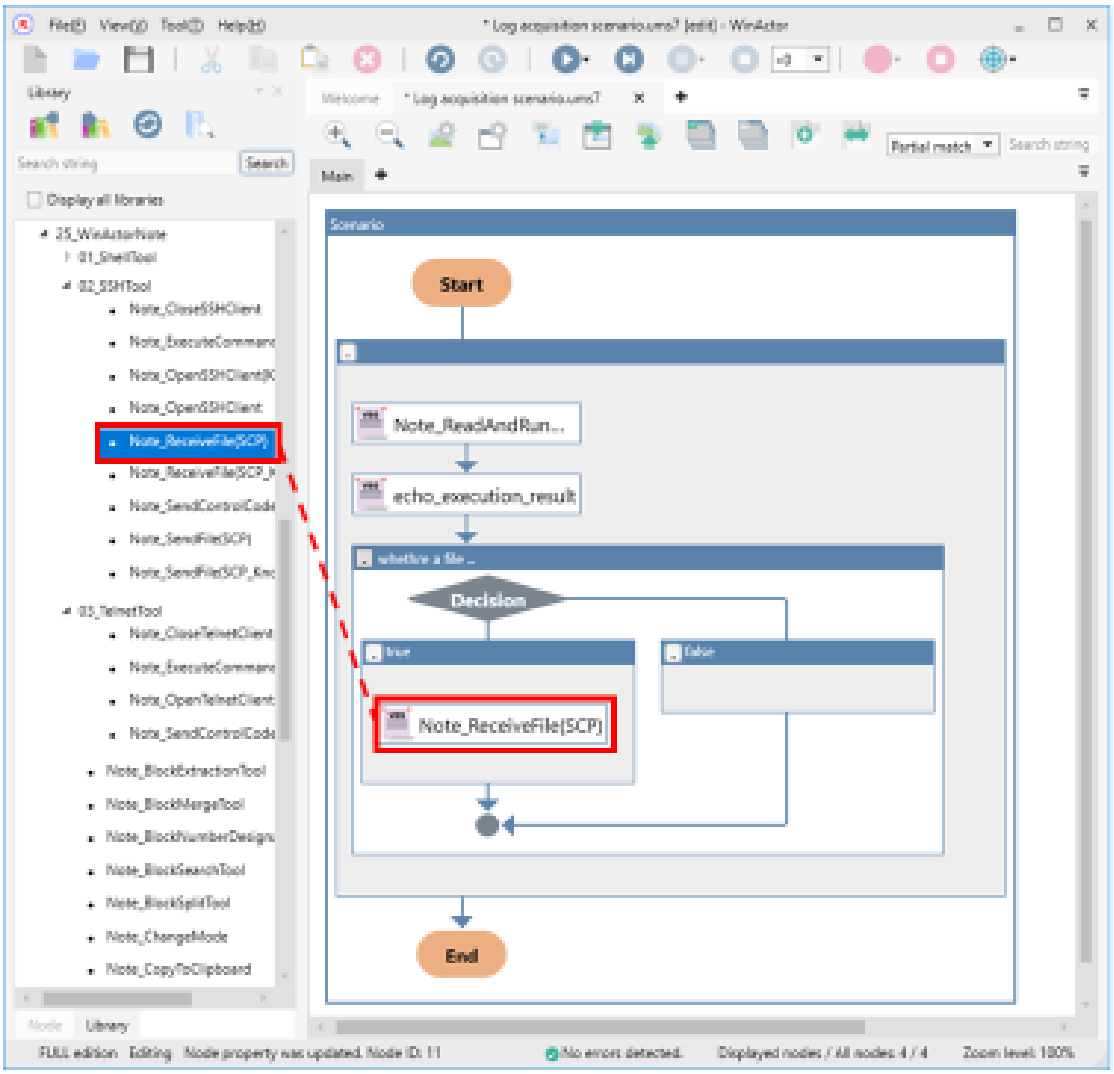


Figure 2-26. Placing the "Note_ReceiveFile(SCP)" library

Double-click the placed library to open the Property window.

For each item, enter the value according to Table 2-4. After entering the values, click the "Update" button. A window for confirming whether to register these values as new variables will appear. Click "Yes," and the values will be registered as new variables in the Variable list pane of WinActor.

Table 2-4. "Note_ReceiveFile(SCP)" library property settings

Item	Setting value	Remarks
Authentication_method	Select "Password_authentication"	-
Host	Enter "Host"	*1
Port	Enter "Port_number"	*1
User_name	Enter "Username"	*1
Password_file	Enter "Password_file"	*1
Private_key_file		Leave "Value=>" as it is
Timeout_value[sec]	Enter "Timeout_value"	*1
Source_file	Enter "Source_file"	*1
Destination_path	Enter "Date"	*1

1　Select "" once and enter the value.

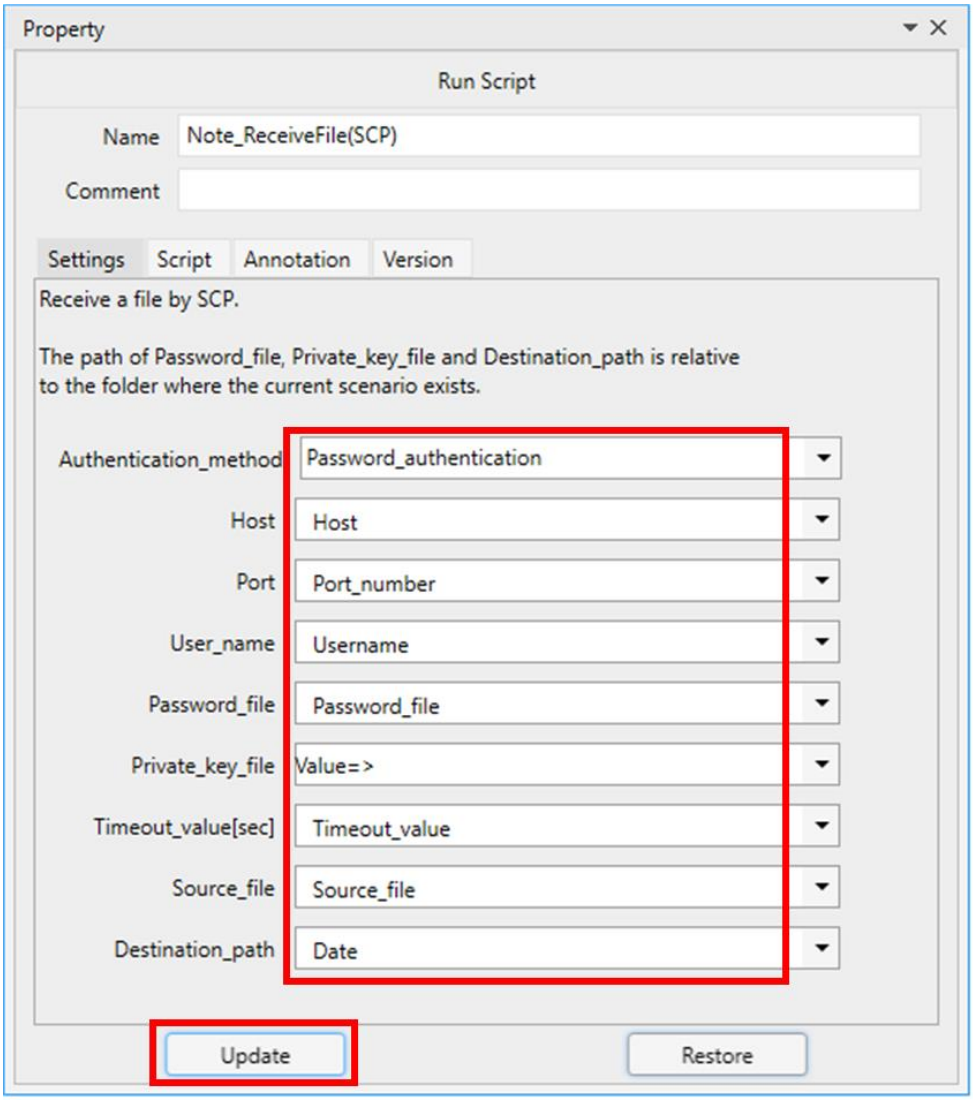


Figure 2-27. After setting properties

Enter the initial value for each variable in the Variable list pane according to Table 2-5.

Table 2-5. Initial value settings in the Variable list pane

Variable name	Initial value	Remarks
Host	192.168.56.2	-
Port_number	22	-
Username	user	-
Password_file	cipherPassword.json	-
Timeout_value	10	Adjust with an appropriate value according to your environment.
Source_file	/home/user/log/*.log	-
Date		Leave the initial value blank. The value will be automatically generated by the processing of Section 2.6.4.

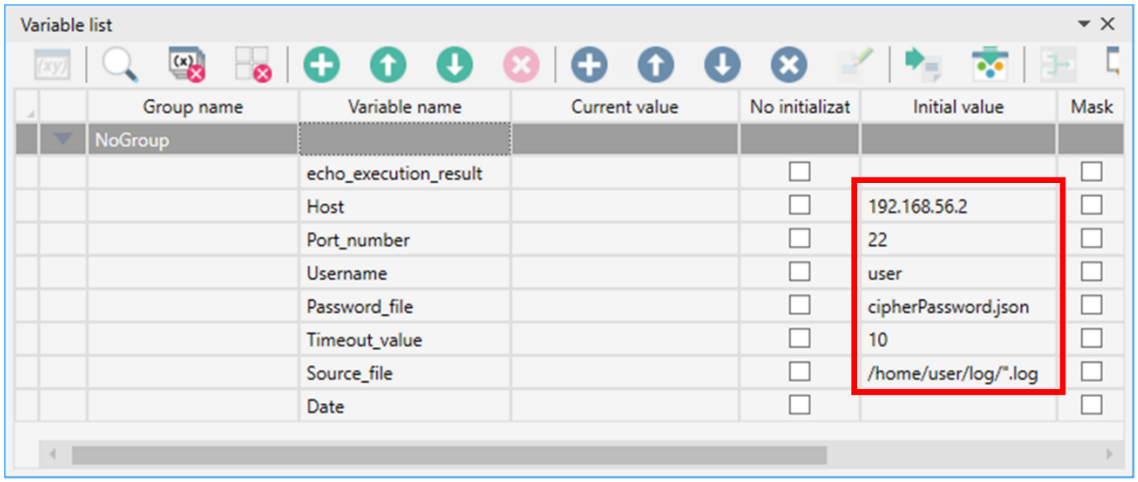


Figure 2-28. After setting the initial values

2.6.4.　Adding the processing of creating the folder for saving logs

①　Creating a subroutine and the processing of calling the subroutine

Add the processing of creating the folder according to the date and time for storing the logs on the SSH server. The log folder creation is processed in a subroutine.

Drag and drop "Call Subroutine" and "Subroutine Group" from the Node tab.

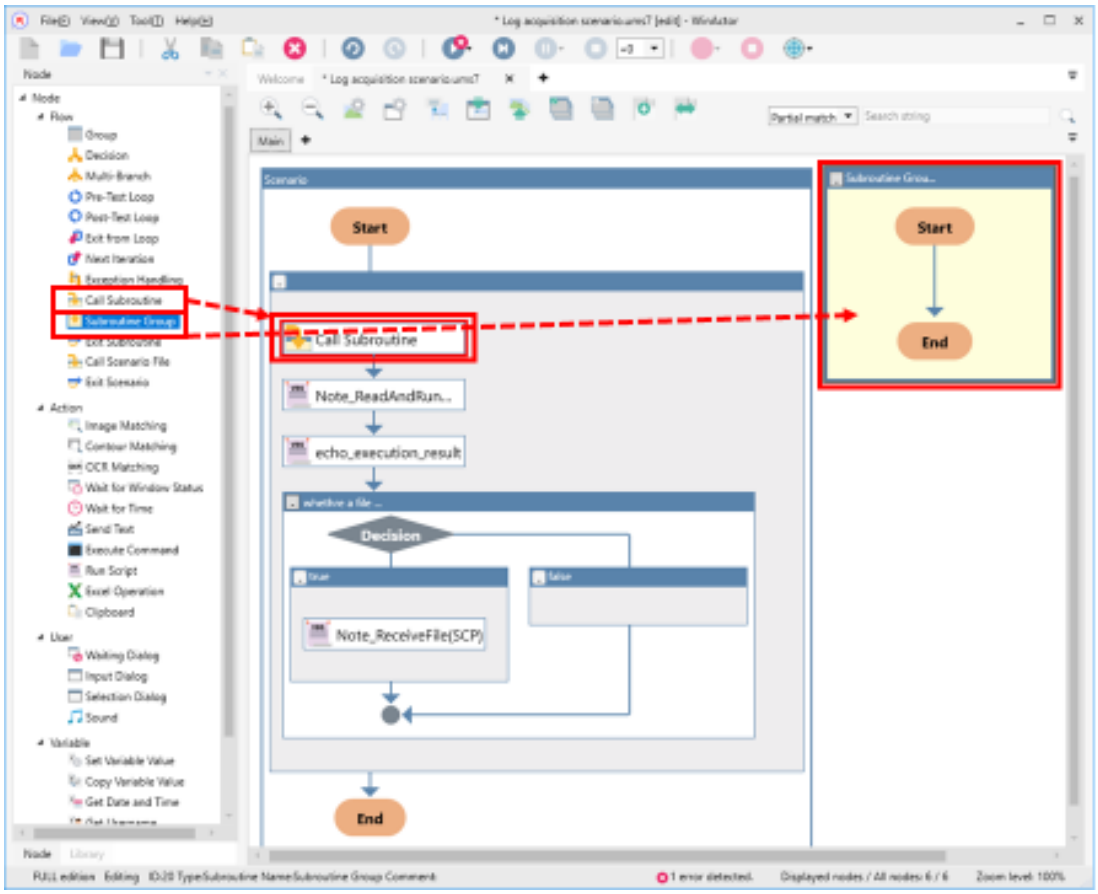
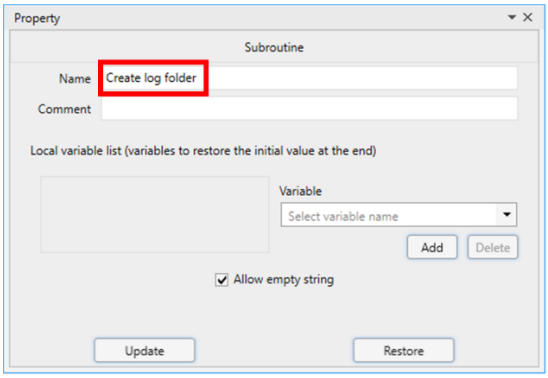


Figure 2-29. Placing the nodes ("Call Subroutine" and "Subroutine Group")

After placing these nodes, open the Property window of each node.

For the "Subroutine Group" node property window, change the name to "Create log folder." For the "Call Subroutine" node property window, change the name to "Call Create log folder" and select "Create log folder" for the subroutine name.



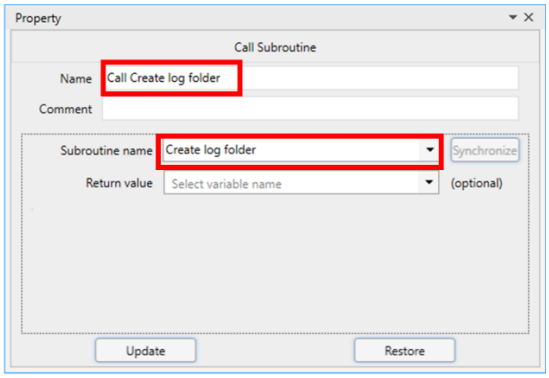


Figure 2-30. "Subroutine Group" and "Call Subroutine" property settings

② Creating the processing in the subroutine

Create the processing in the "Create log folder" subroutine.

The overall steps are as follows.

1. Creating the folder name

   ① Getting the date and time.

   The format of the date and time to be acquired is as follows.

   <Date and time format>

   yyyy-mm-dd HH:mm:ss

   However, it may not be in the above format depending on the regional settings of Windows. In that case, refer to this manual and change the processing contents of the scenario.

   ② Generating the folder name.

   Convert the format of ① into the format "yyyy-mm-dd_HHmmss" of ④ in Table 2-2 by removing ":" and replace the space with an underscore.

2. Creating the folder

   Create the folder with the folder name generated in the step ② of 1 above.

The details of each processing are described below.

1. ①　Getting the date and time

Drag and drop "Get Date and Time" from the Node tab.

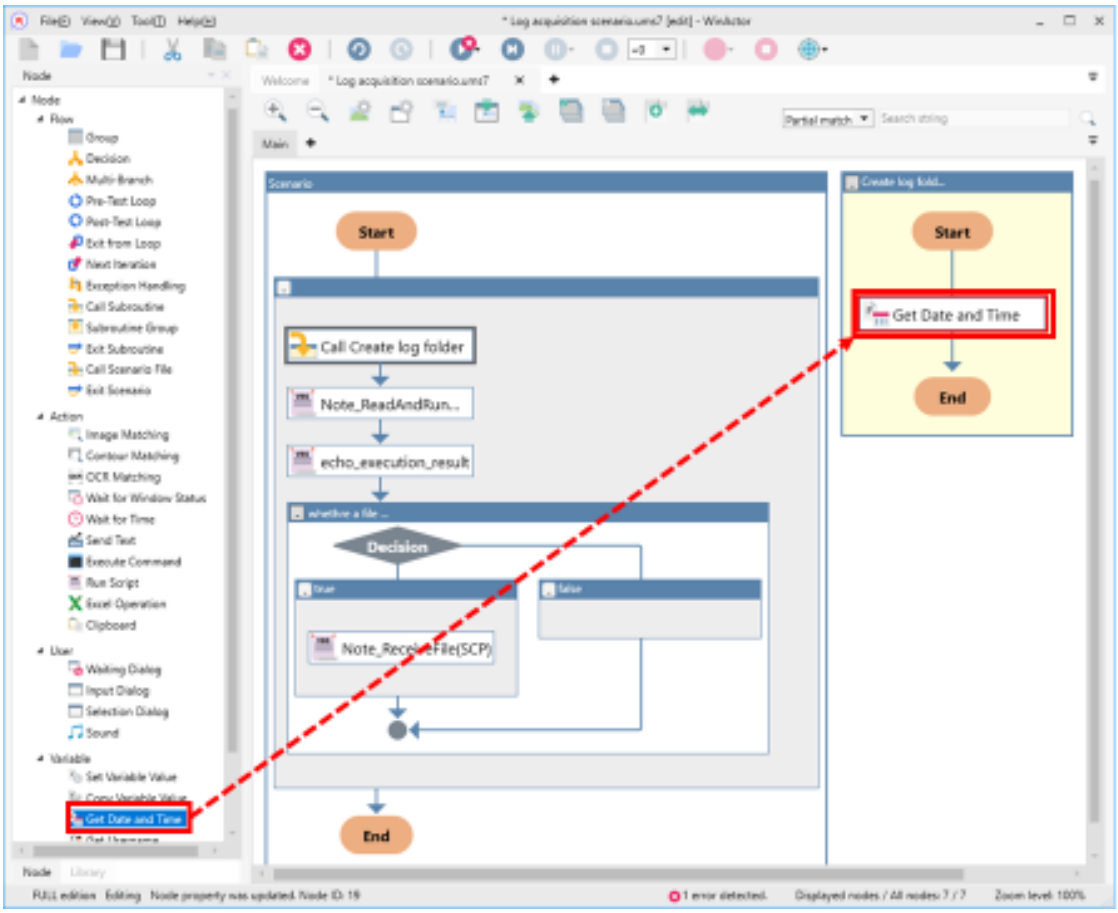


Figure 2-31. Placing the "Get Date and Time" node

After placing the node, double-click it to open the Property window.

Select "Date" for the output variable, "yyyy-mm-dd" for the date format, "OS default" for the time zone, and click the "Update" button.

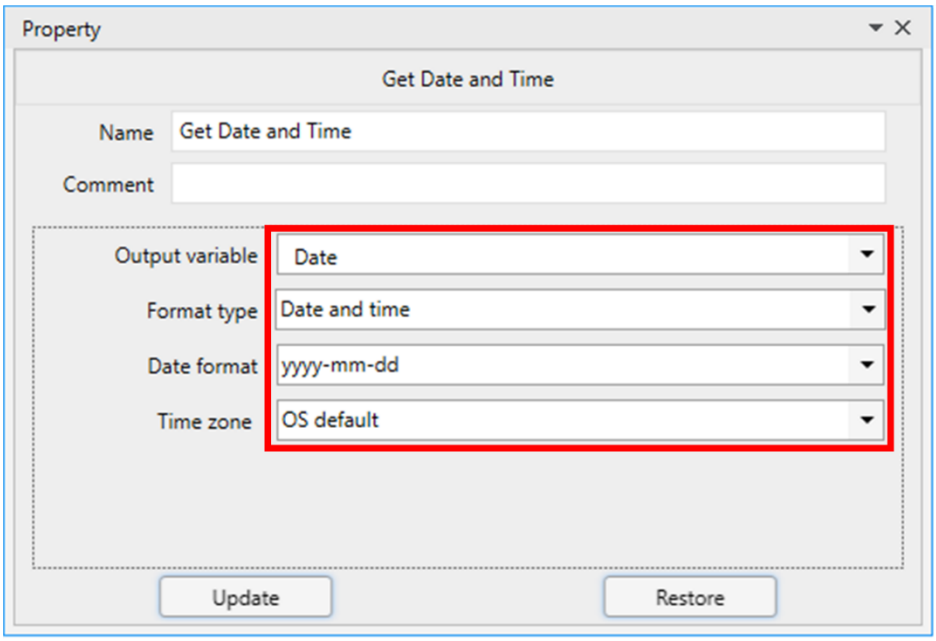


Figure 2-32. "Get Date and Time" node property settings

1. ②　Generating the folder name

First, add two "Conversion_ReplaceString" libraries in a row. For Converted_string, select "Date" for both libraries. For Before_replacement and After_replacement, select ":" and blank for the first "Conversion_ReplaceString" library, and a single space and "_" for the second "Conversion_ReplaceString" library.

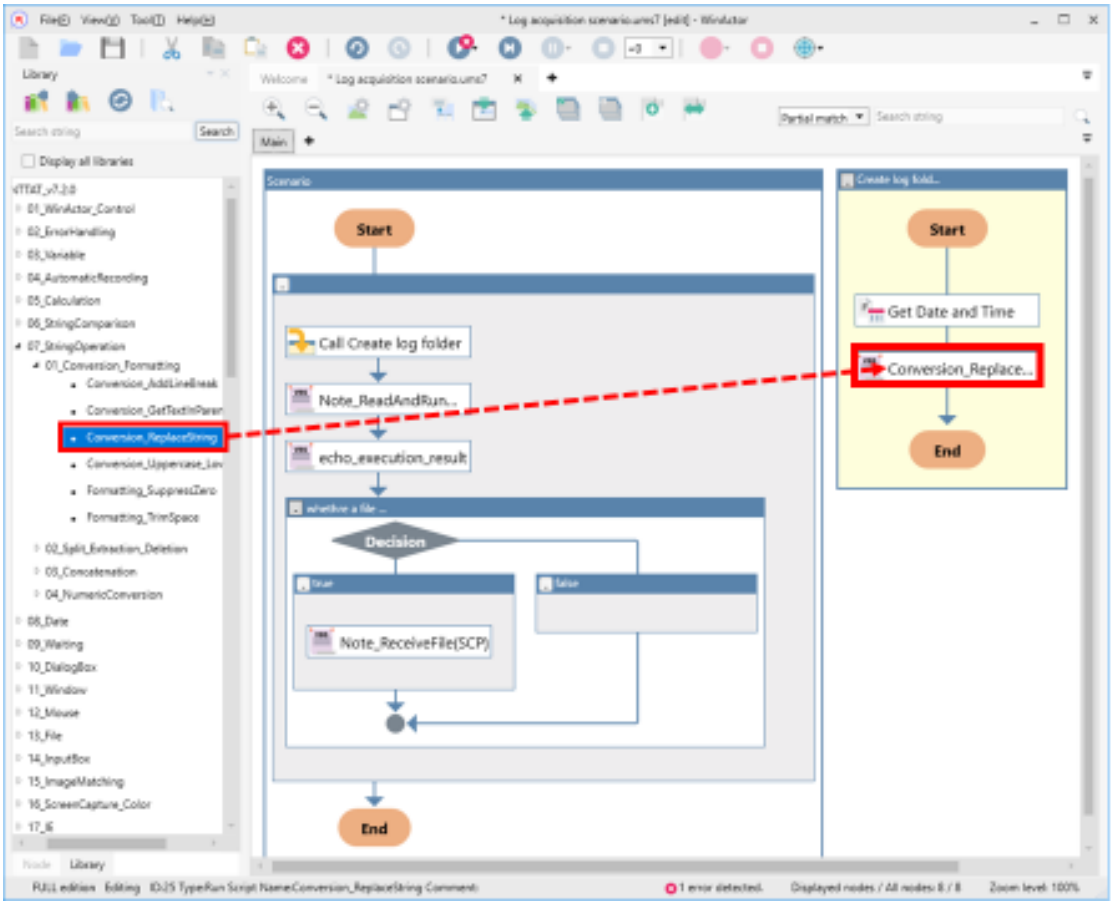


Figure 2-33. Placing the first "Conversion_ReplaceString" library

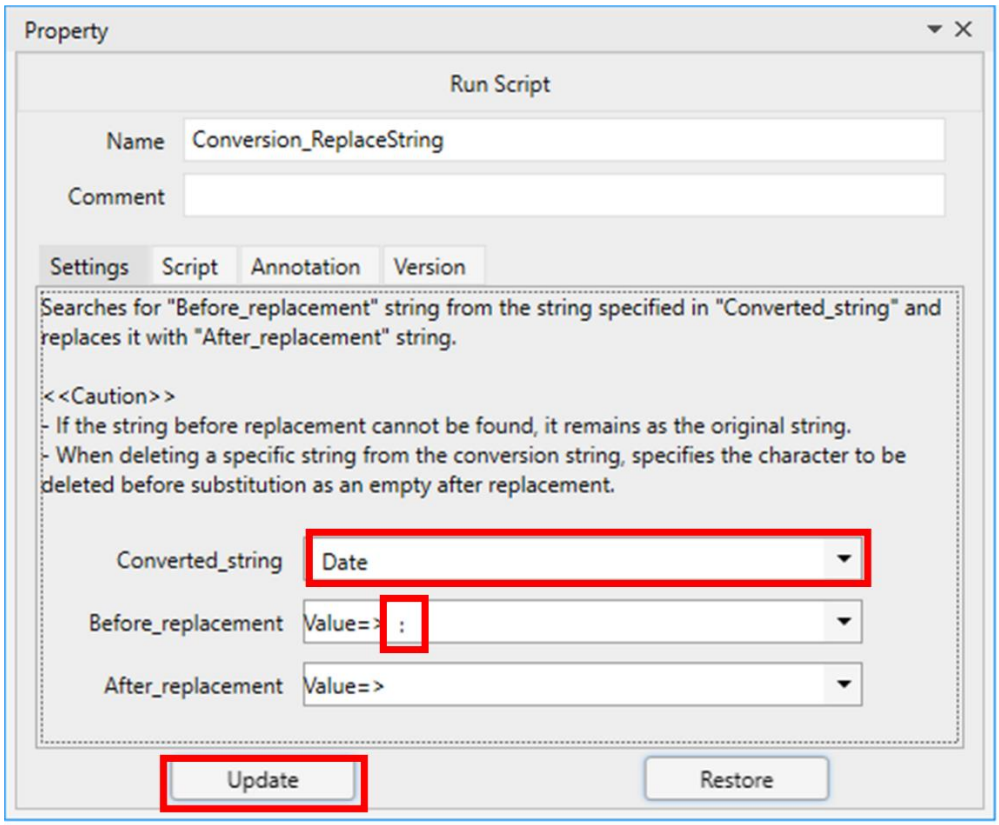


Figure 2-34. The first "Conversion_ReplaceString" library property settings (":" and blank)

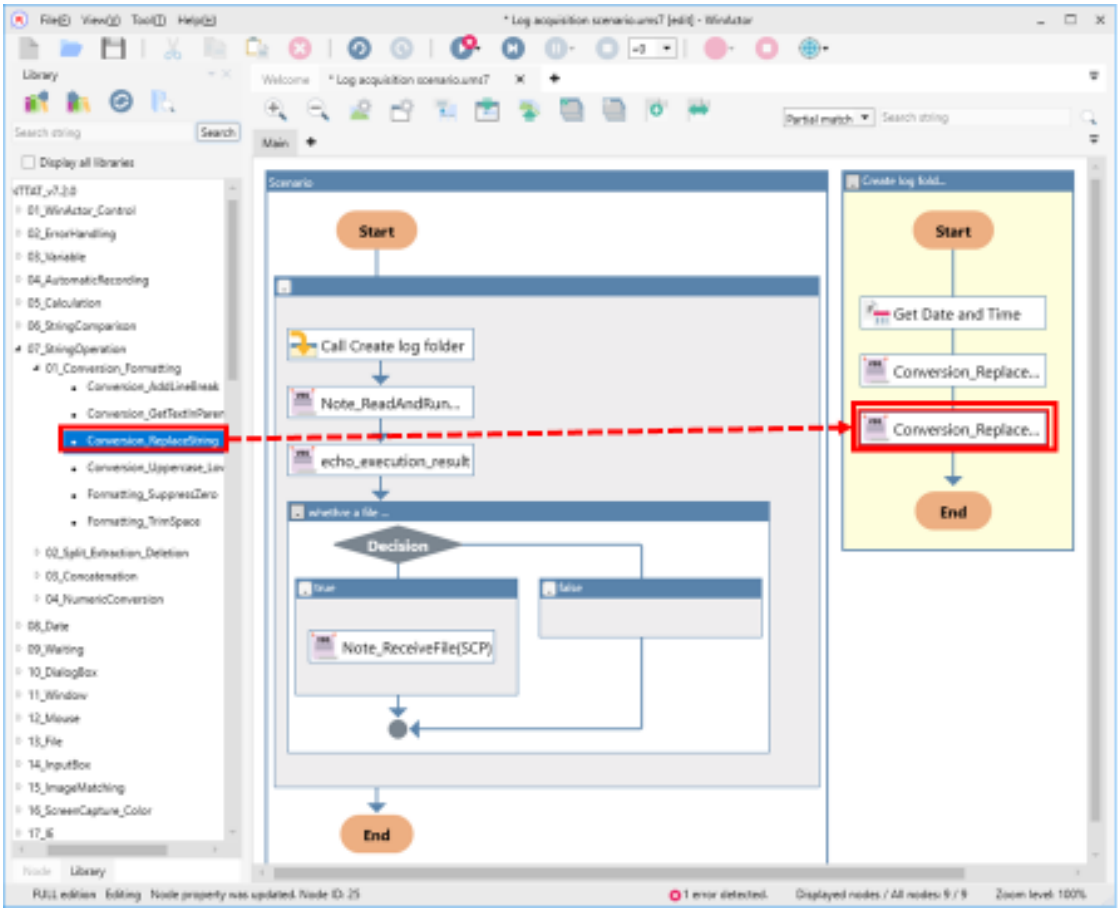


Figure 2-35. Placing the second "Conversion_ReplaceString" library

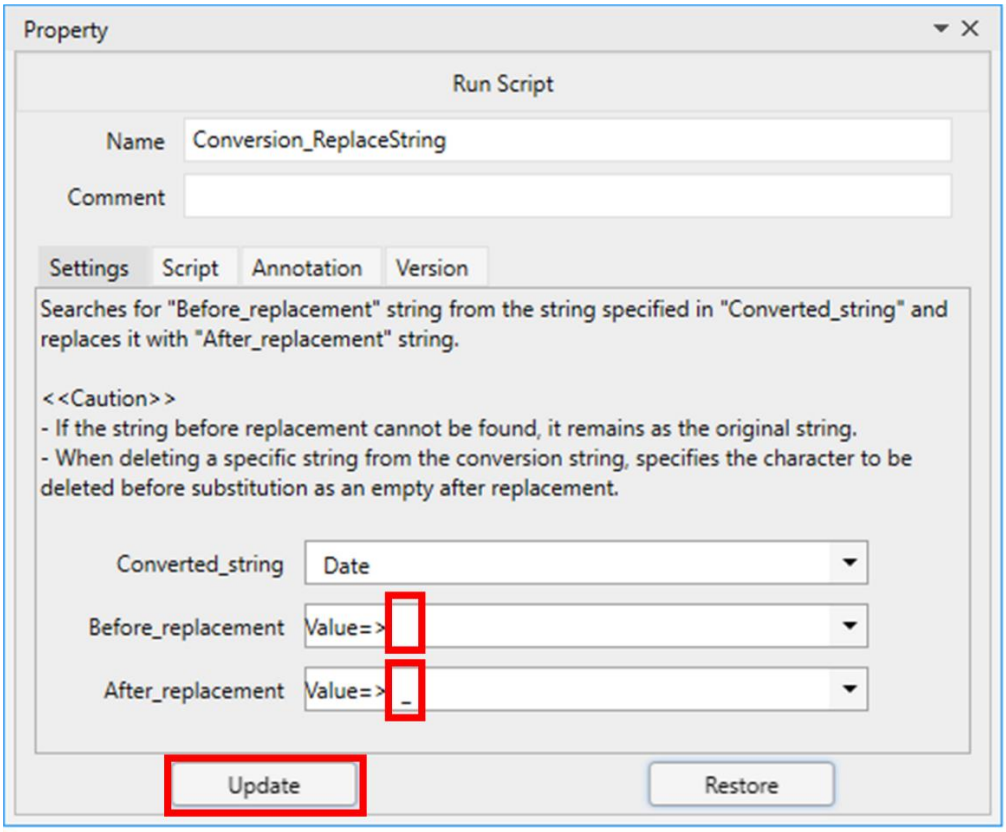


Figure 2-36. The second "Conversion_ReplaceString" library property settings (single space and "_")

Next, drag and drop the "FileName_GetScenarioFilePath" library.

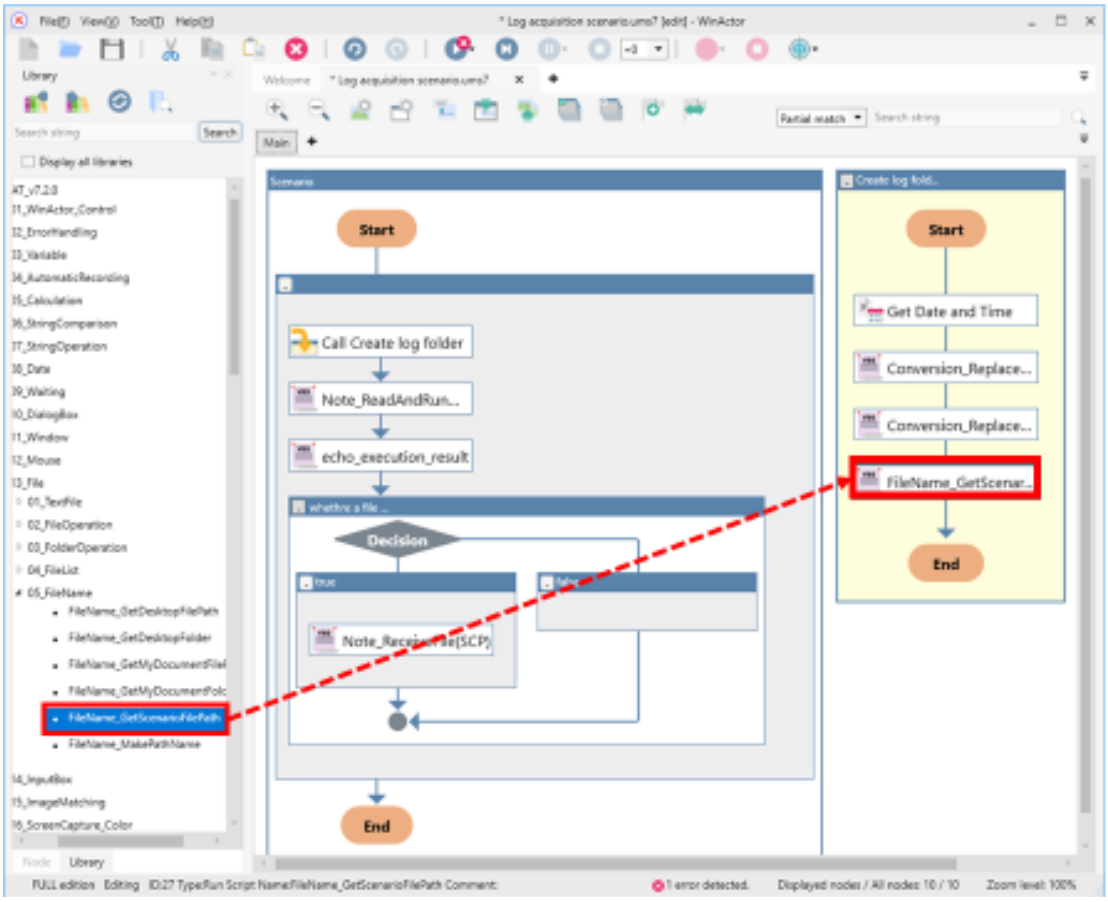


Figure 2-37. Placing the "FileName_GetScenarioFilePath" library

Double-click the placed library to open the Property window.

Select "Date" for the filename, enter "Result_folder" for the file path generation result, and click the "Update" button.

A window for confirming whether to register the entered value as a new variable will appear. Click "Yes," and it will be registered as a new variable in the Variable list pane of WinActor.

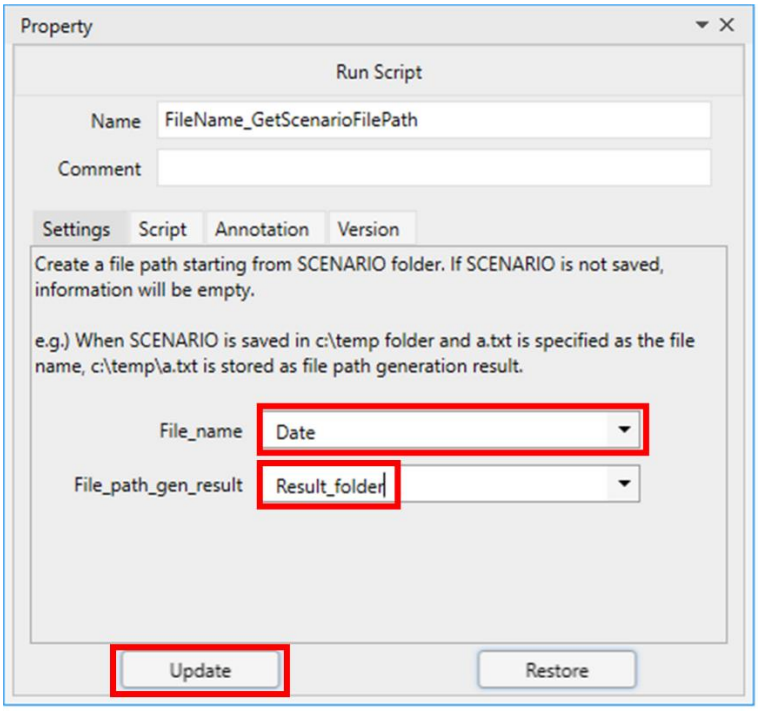


Figure 2-38. "FileName_GetScenarioFilePath" library property settings

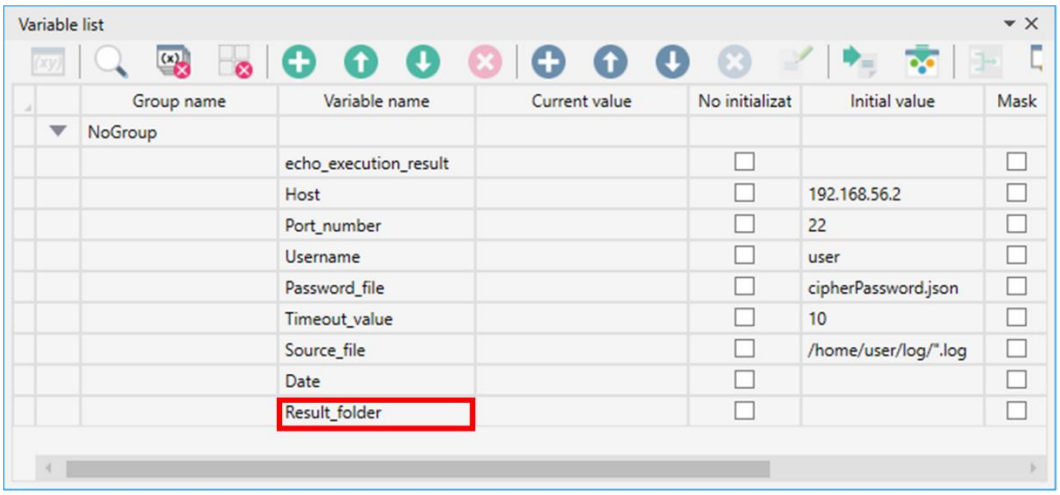


Figure 2-39. Confirming the update of the Variable list pane

2. Creating the folder

Create the folder based on the folder name (variable "Result_folder") to save the log file created so far.

Drag and drop "FolderOperation_CreateFolder" from the user library.

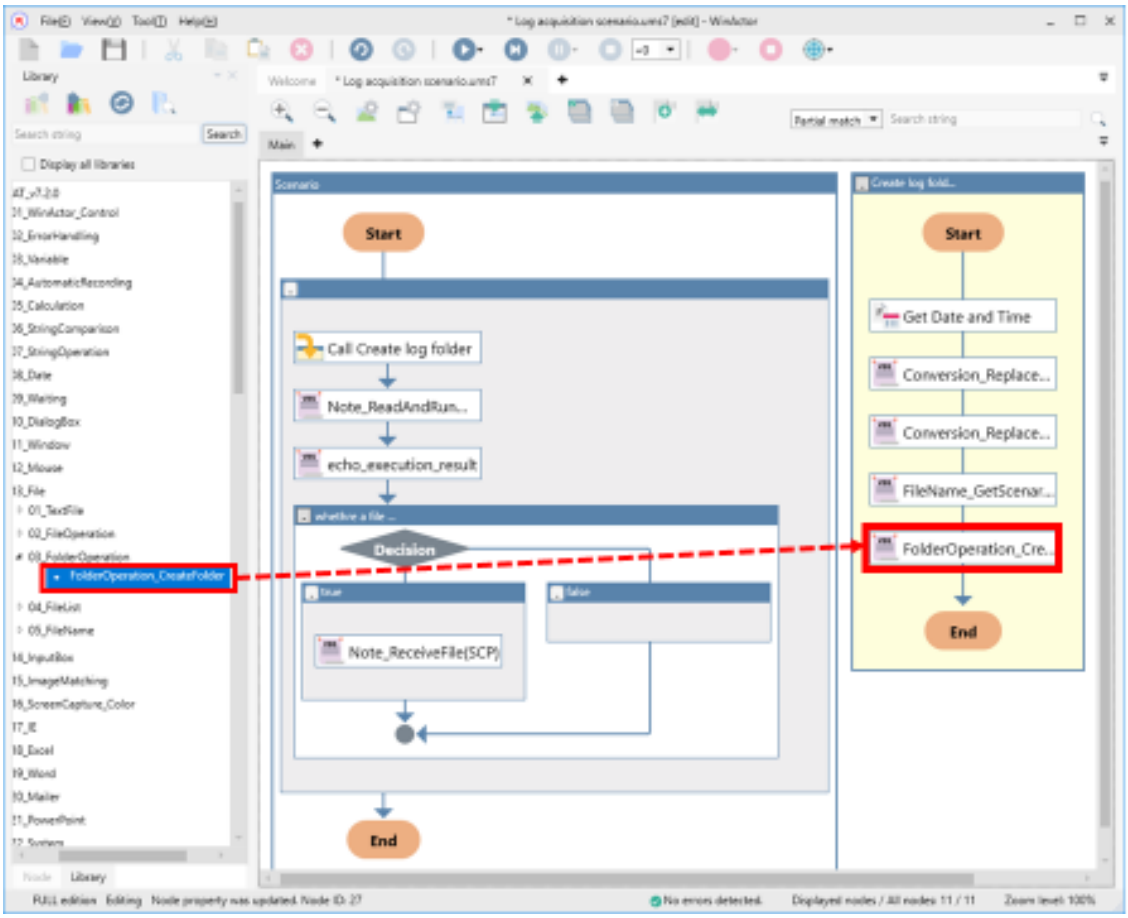


Figure 2-40. Placing the "FolderOperation_CreateFolder" library

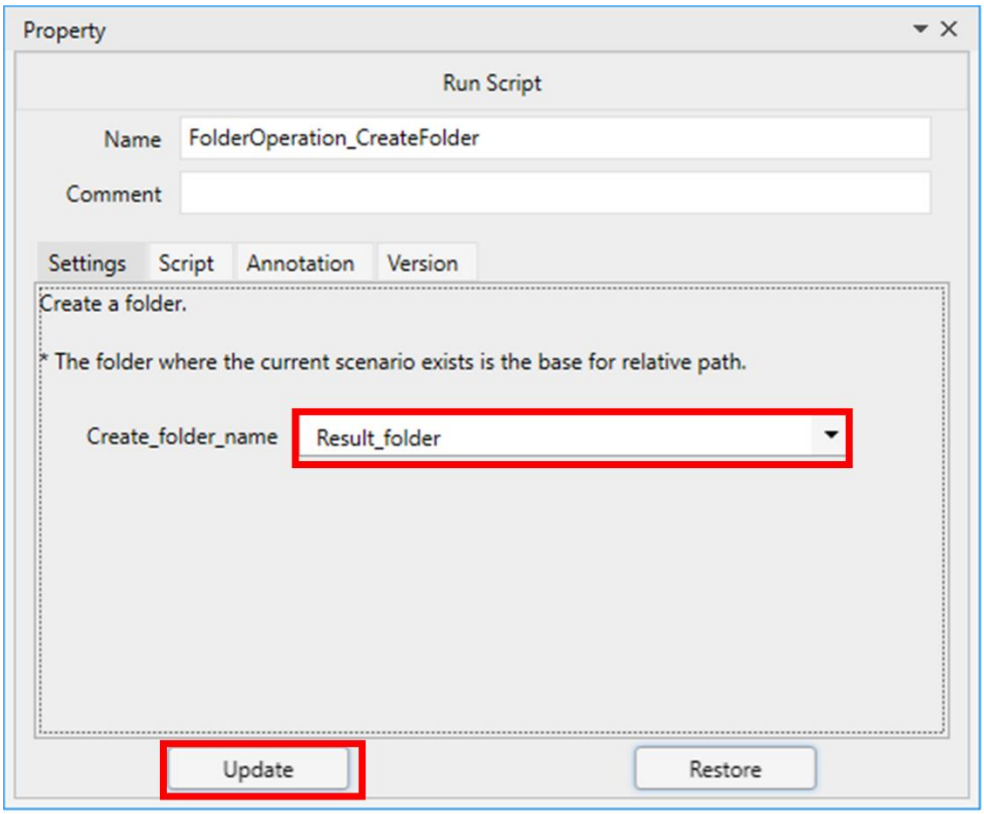


Figure 2-41. "FolderOperation_CreateFolder" library property setting

To check the operations, perform "Partial run" for the current scenario.

After clicking and selecting "Create log folder" in the flowchart area, right-click it and select "Partial run."

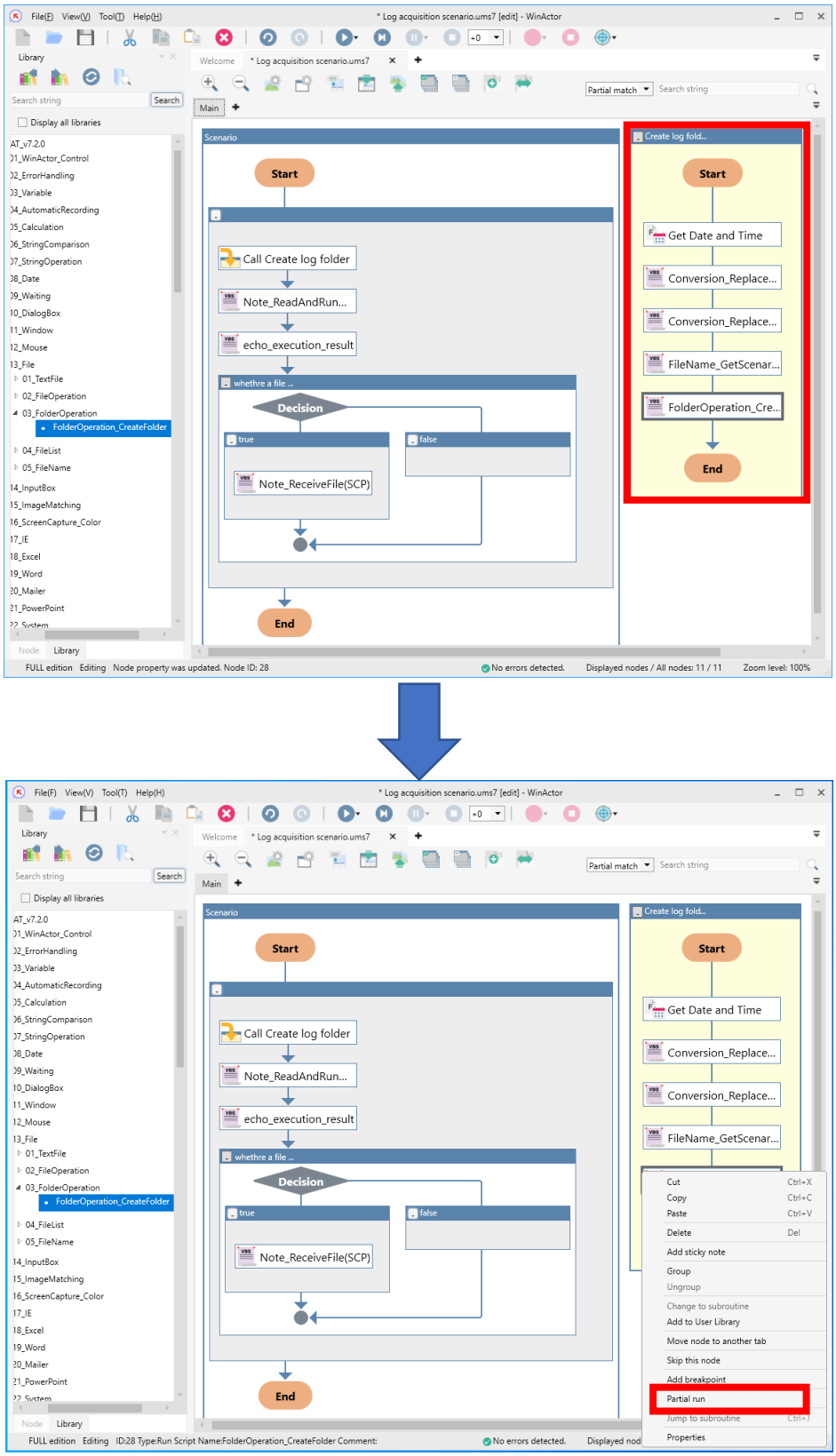


Figure 2-42. Performing "Partial run" for "Create log folder"

After performing "Partial run," open C:\Terminal_function_scenario in Explorer and confirm that the empty folder is created in the format of ④ in Table 2-2. Delete the created folder after confirmation.

2.6.5.　Adding the processing of moving the log file

①　Creating a subroutine and the processing of calling the subroutine

Add the processing of moving the file (SSH_client_execution_log.txt) where the display contents of WinActor Note are saved to the log folder.

Drag and drop "Call Subroutine" and "Subroutine Group" from the Node tab.

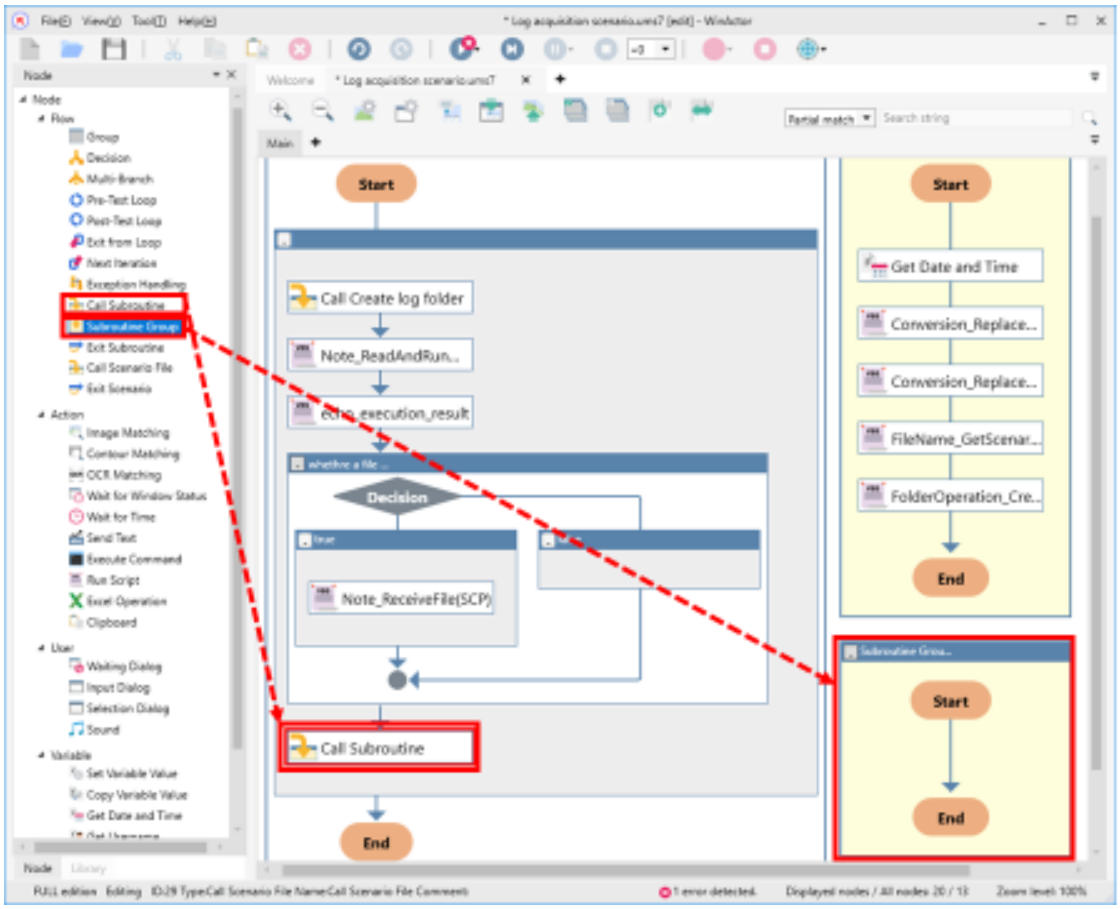


Figure 2-43. Placing the nodes ("Call Subroutine" and "Subroutine Group")

After placing these nodes, open the Property window of each node.

For the "Subroutine Group" node property window, change the name to "Move log file." For the "Call Subroutine" node property window, change the name to "Call Move log file" and select "Move log file" for the subroutine name.

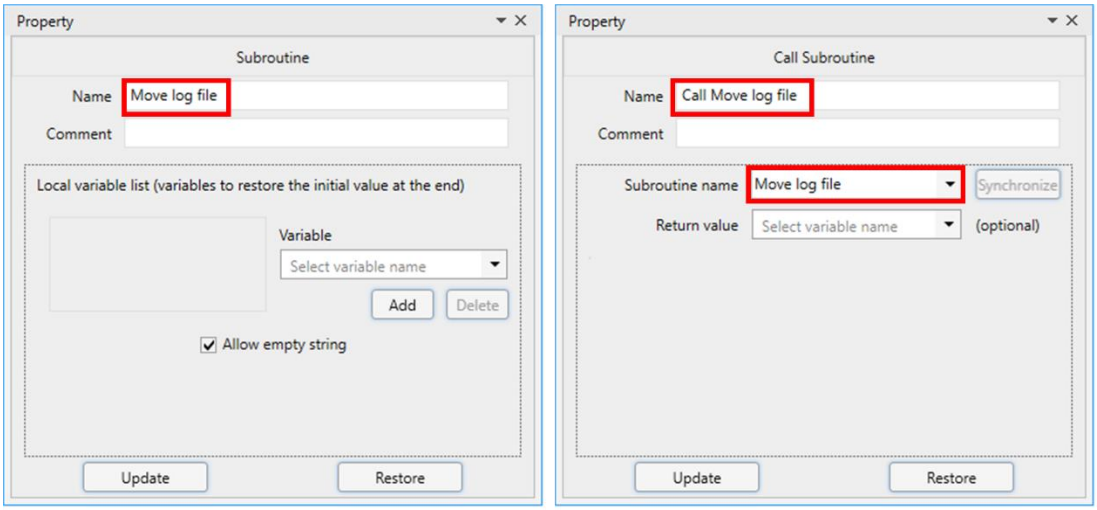


Figure 2-44. "Subroutine Group" and "Call Subroutine" property settings

② Creating the processing in the subroutine

Create the processing in the "Move log file" subroutine.

1. Adding the "FileName_GetScenarioFilePath" library

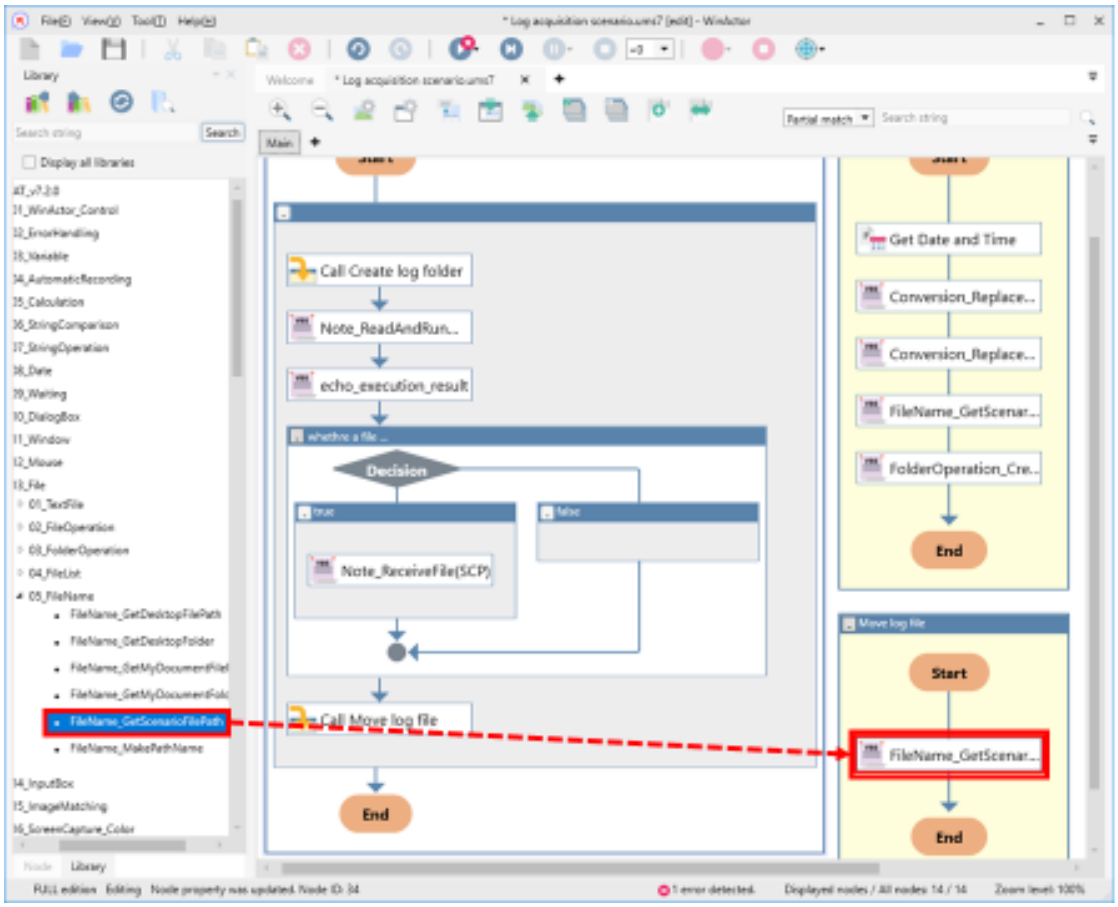


Figure 2-45. Placing the "FileName_GetScenarioFilePath" library

Double-click the placed library to open the Property window.

Enter "SSH_client_execution_log.txt" for the filename and "SSH_log_filename_absolute_path" for the file path generation result, and click the "Update" button.

A window for confirming whether to register the entered value as a new variable will appear. Click "Yes," and it will be registered as a new variable in the Variable list pane of WinActor.

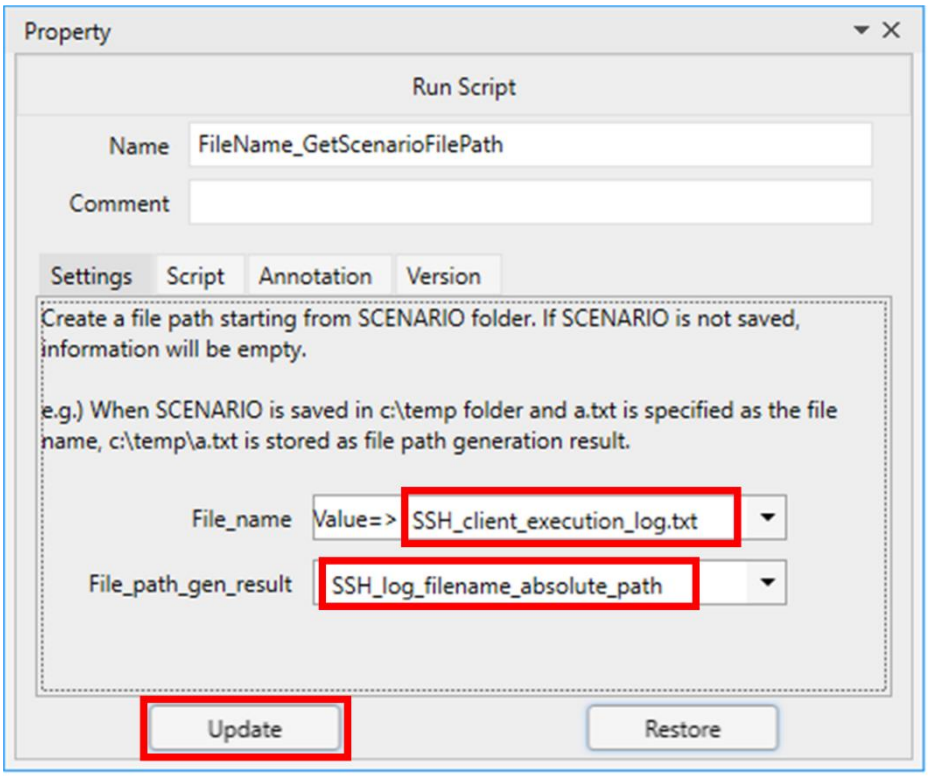


Figure 2-46. "FileName_GetScenarioFilePath" library property settings

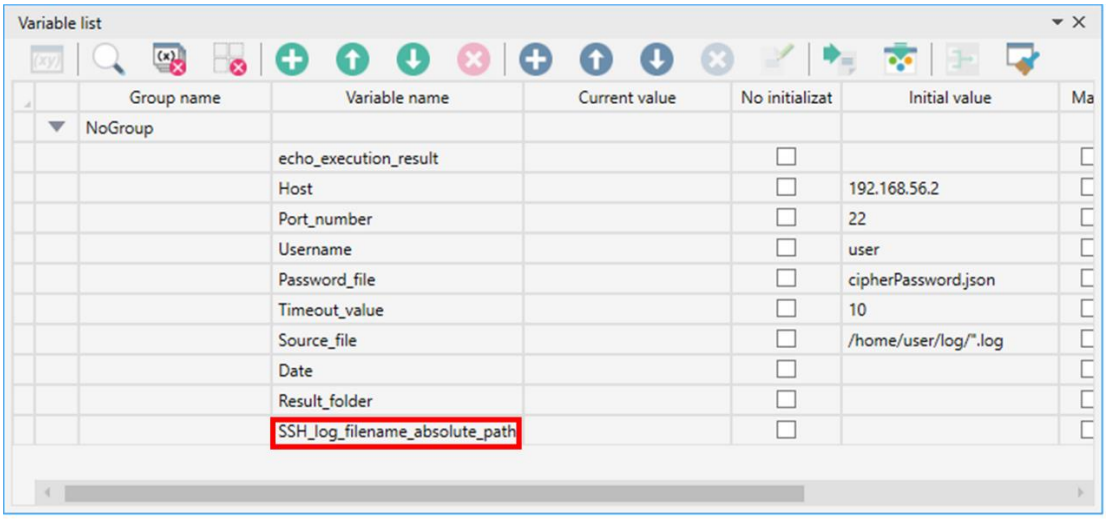


Figure 2-47. Confirming the update of the Variable list pane

2. Adding the "FileOperation_MoveFile" library

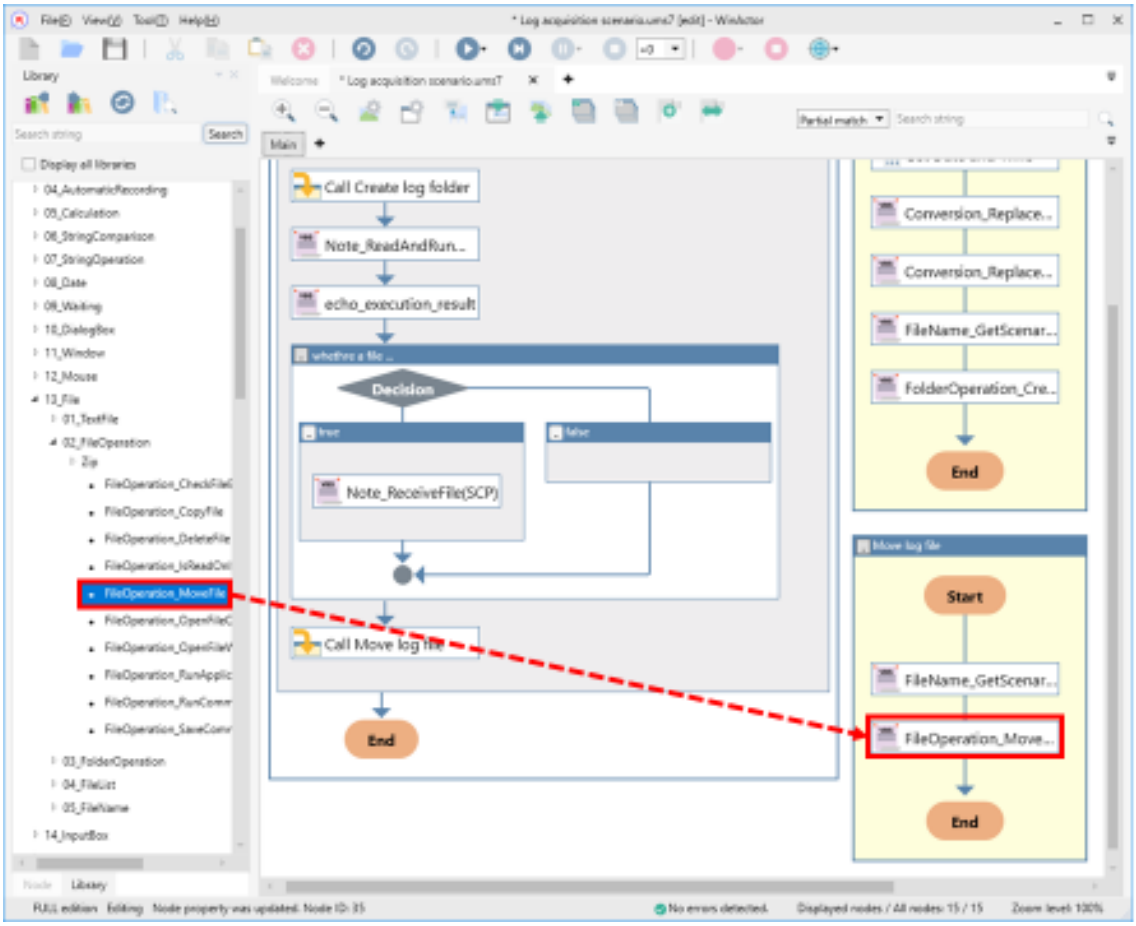


Figure 2-48. Placing the "FileOperation_MoveFile" library

Double-click the placed library to open the Property window.

Select "SSH_log_filename_absolute_path" for Move_from, enter "Result_folder" for Move_to, and click the "Update" button.

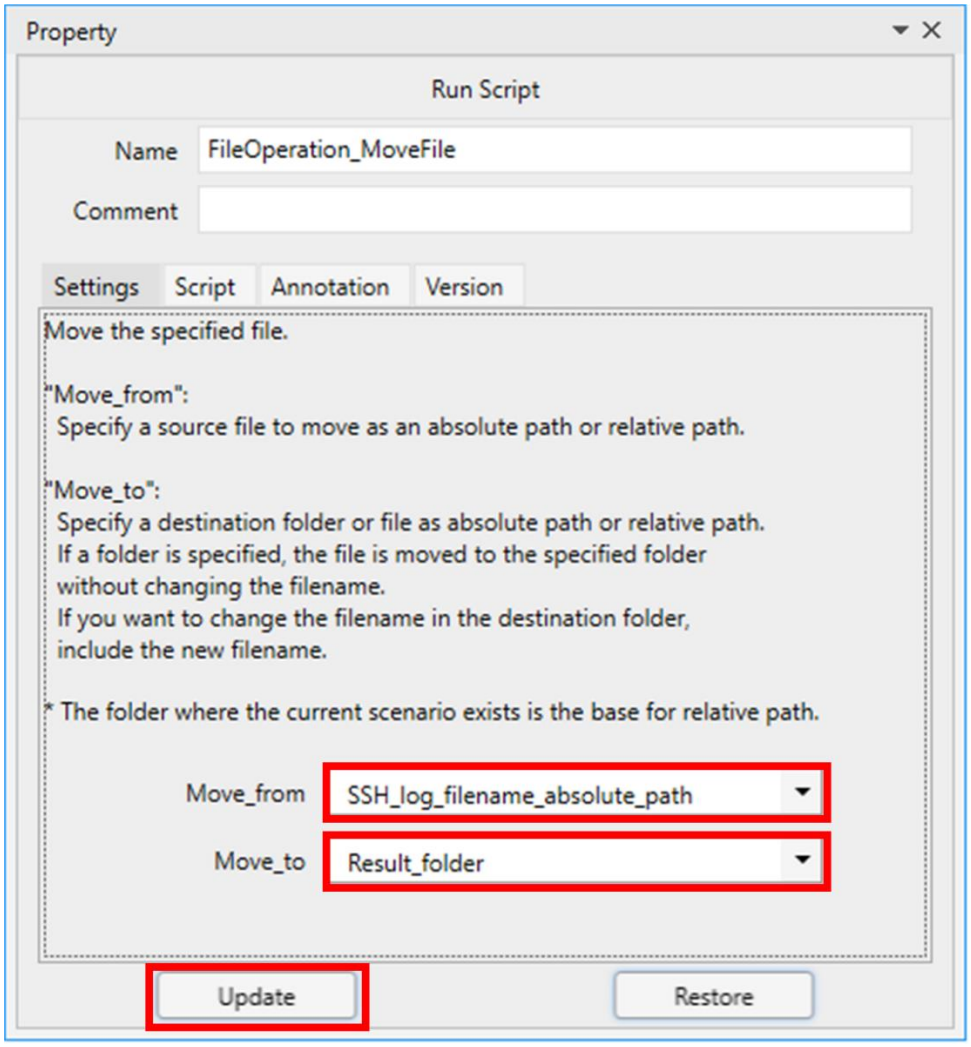


Figure 2-49. "FileOperation_MoveFile" library property settings

Save the scenario modified up to this point.

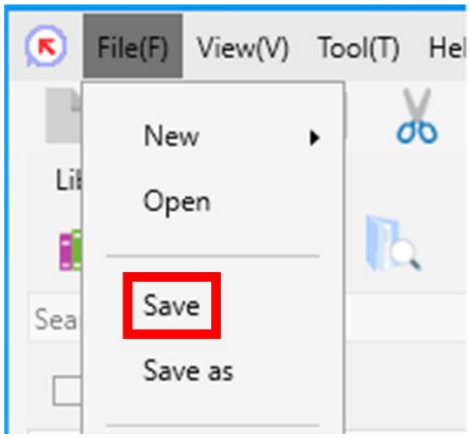


Figure 2-50. Saving the scenario

2.7. Checking the operations

Check the final operations of the scenario.

Click the "Run scenario" button on the toolbar of WinActor.

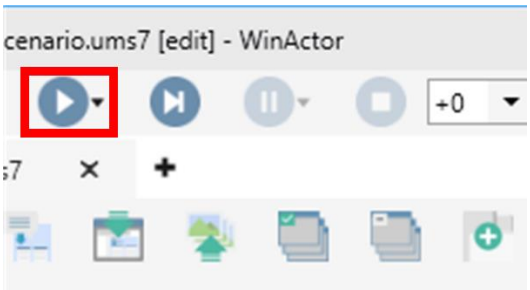


Figure 2-51. Running the scenario

After running the scenario, the folder at the date and time when the scenario was run will be created in the "C:\Terminal_function_scenario" folder, and three files of "server#1.log," "server#2.log," and "SSH_client_execution_log.txt" will be acquired.

Figure 2-52 shows the example of the acquired "SSH_client_execution_log.txt."

```
Last login: Thu Aug 29 14:07:12 2019 from 192.168.56.1
[user@demoserver ~]$ cd log

[user@demoserver log]$ date
Thursday, August 29 2019 14:35:54 JST
[user@demoserver log]$ ls *.log
server#1.log  server#2.log
[user@demoserver log]$ echo $?
0
[user@demoserver log]$
```

Figure 2-52. Example of the acquired execution log

3. Library and property list

3.1. Shell tool

This section describes the libraries for using PowerShell and Command Prompt and the procedure to create a basic scenario using those libraries. The property of each library related to "Shell tool" is described in 3.1.1 to 3.1.4.

[Procedure to execute PowerShell]

1. Placing the libraries in the Scenario box

   Place the following libraries in order in the Scenario box.

   3.1.1　Note_OpenPowerShell

   3.1.3　Note_ExecuteCommand

   3.1.4　Note_ClosePowerShellOrCommandPrompt

   The scenario after placing these libraries is as shown in Figure 3-1.

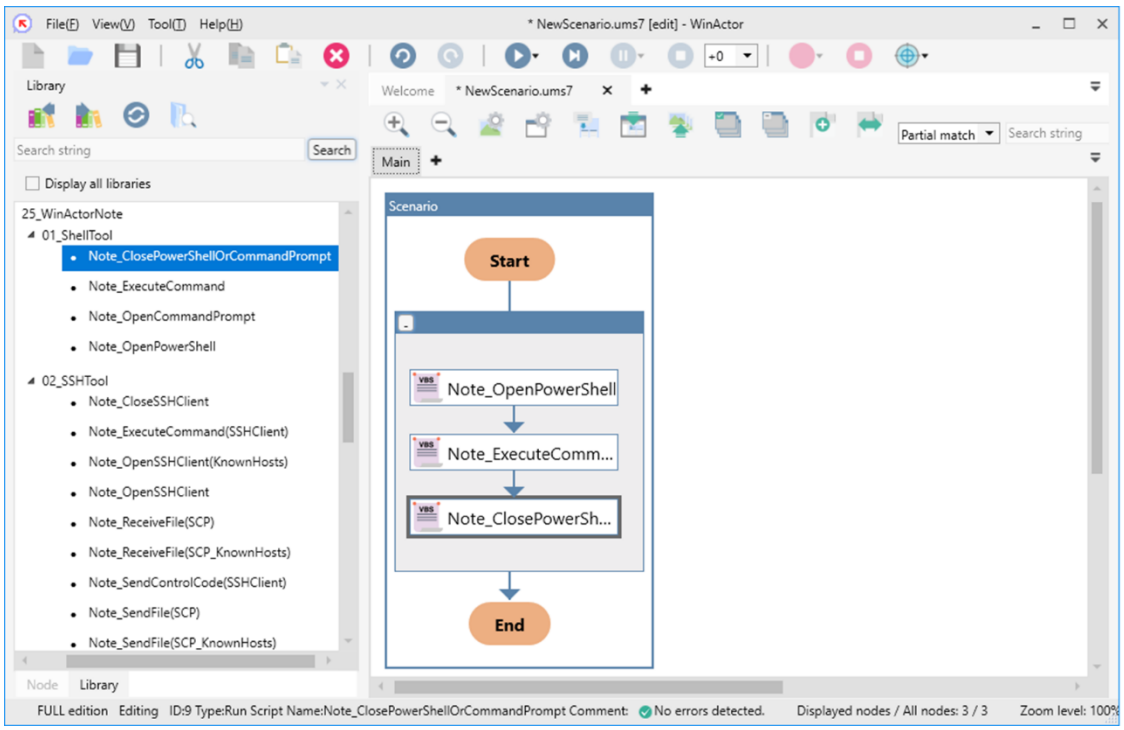


Figure 3-1. Example of the basic scenario using PowerShell

2. Setting the library properties

   Set the properties according to the information in the subsections below.

   3.1.1 Note_OpenPowerShell

   3.1.3 Note_ExecuteCommand

   3.1.4 Note_ClosePowerShellOrCommandPrompt

[Procedure to execute Command Prompt]

1. Placing the libraries in the Scenario box

   Place the following libraries in order in the Scenario box.

   3.1.2 Note_OpenCommandPrompt

   3.1.3 Note_ExecuteCommand

   3.1.4 Note_ClosePowerShellOrCommandPrompt

   The scenario after placing these libraries is as shown in Figure 3-2.

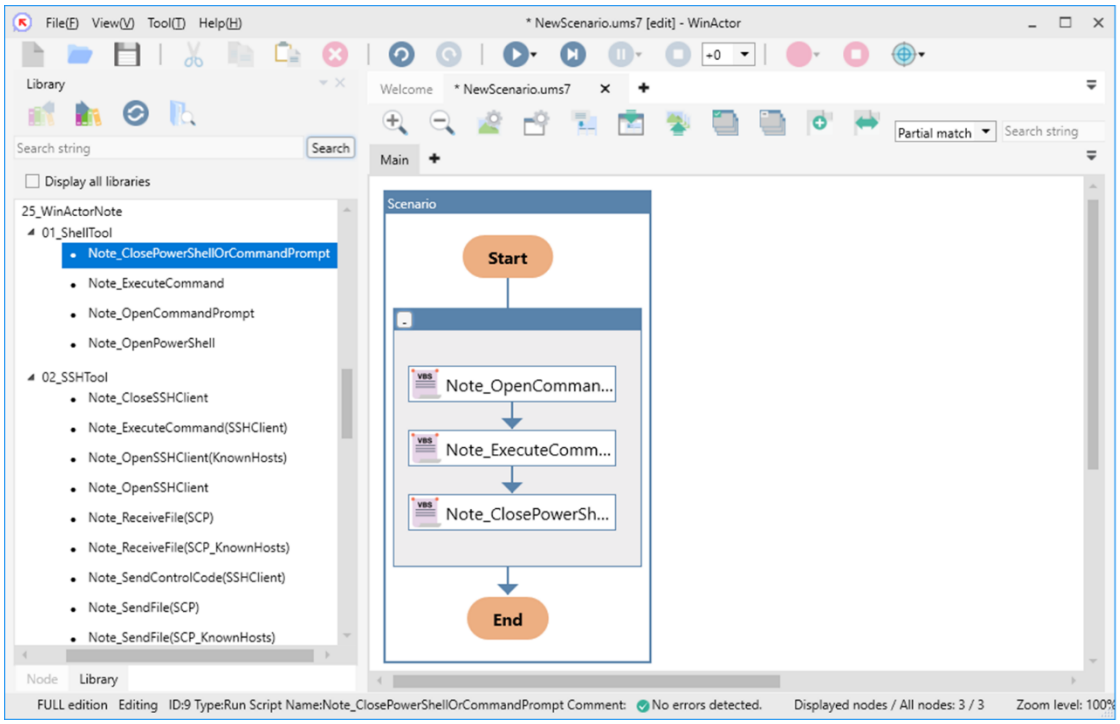


Figure 3-2. Example of the basic scenario using Command Prompt

2. Setting the library properties

   Set the properties according to the information in the subsections below.

   3.1.2 Note_OpenCommandPrompt

   3.1.3 Note_ExecuteCommand

   3.1.4 Note_ClosePowerShellOrCommandPrompt

3.1.1. Note_OpenPowerShell

This library is to open PowerShell.

Table 3-1. "Note_OpenPowerShell" library property setting

No.	Item	Description
①	Encoding	Specify a character encoding for importing into PowerShell and exporting to WinActor Note.

3.1.2. Note_OpenCommandPrompt

This library is to open Command Prompt.

Table 3-2. "Note_OpenCommandPrompt" library property setting

No.	Item	Description
①	Encoding	Specify a character encoding for importing into Command Prompt and exporting to WinActor Note.

3.1.3. Note_ExecuteCommand

This library is to execute a command on PowerShell or Command Prompt.

Table 3-3. "Note_ExecuteCommand" library property setting

No.	Item	Description
①	Command	Specify a command you want to execute on PowerShell or Command Prompt. Only text can be entered. Control characters cannot be sent.

3.1.4. Note_ClosePowerShellOrCommandPrompt

This library is to close a session of PowerShell or Command Prompt.

3.2. SSH tool

This section describes the libraries for using the SSH client function and the file transfer function (SCP) provided in "SSH tool" and the procedure to create a basic scenario using those libraries. The setting methods to connect to a common server with the SSH client file transfer function (SCP) are described in 3.2.1. The property of each library related to "SSH tool" is described in 3.2.2 to 3.2.9.

[Procedure to execute the SSH client function]

1. Placing the libraries in the Scenario box

   Place the following libraries in order in the Scenario box.

   3.2.2 Note_OpenSSHClient

   3.2.4 Note_ExecuteCommand(SSHClient)

   3.2.5 Note_CloseSSHClient

   The scenario after placing these libraries is as shown in Figure 3-3.

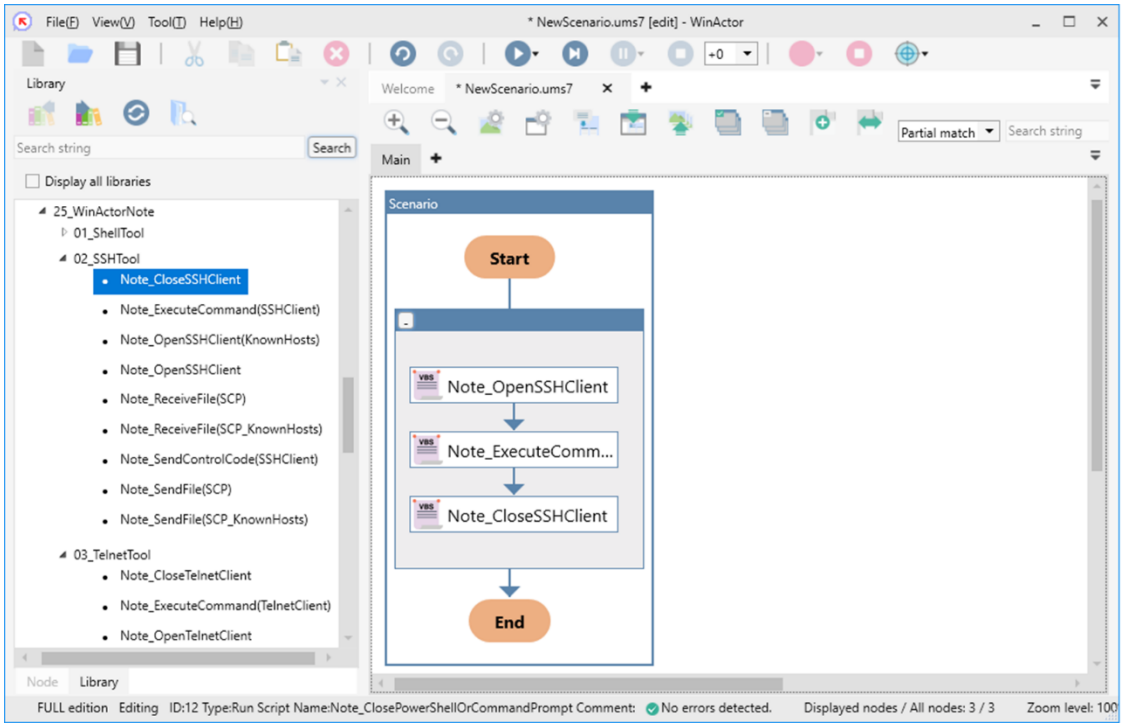


Figure 3-3. Example of the basic scenario using SSH client

2. Setting the library properties

   Set the properties according to the information in the subsections below.

3.2.2　Note_OpenSSHClient
3.2.4　Note_ExecuteCommand(SSHClient)
3.2.5　Note_CloseSSHClient

[Procedure to execute the function to send a file (SCP)]

1. Placing the library in the Scenario box
   Place the following library in the Scenario box.
   3.2.6　Note_SendFile(SCP)

2. Setting the library property
   Set the property according to the information in the subsection below.
   3.2.6　Note_SendFile(SCP)

[Procedure to execute the function to receive a file (SCP)]

1. Placing the library in the Scenario box
   Place the following library in the Scenario box.
   3.2.8　Note_ReceiveFile(SCP)

2. Setting the library property
   Set the property according to the information in the subsection below.
   3.2.8　Note_ReceiveFile(SCP)

3.2.1. Connection settings

This subsection describes the setting methods to connect to a common server with the SSH client file transfer function (SCP).

In "SSH tool," two authentication methods, Password authentication and Public key authentication, are available to connect to an SSH server. The following describes the setting items required to connect to an SSH server for each of the above two authentication methods.

[Password authentication]

1. Generate a password file that contains the password information for connecting to an SSH server in advance. For details on how to generate a password file, see the section of "Password file generation tool" in the material No.3 in Table 5-1.
2. Set the values according to Table 3-4. The setting values described here follow the example in Table 2-1. Environment used in this tutorial. Set the values according to your environment.

Table 3-4. Settings for password authentication

Item	Description	Setting values in the example in this document
Authentication_method	Select the authentication method.	Password authentication
Character_encoding	Specify a character encoding for importing into the server and exporting to WinActor Note.	UTF-8
Line-break_code_for_sending	Specify a line-break code for sending to the server.	LF(Linux, MacOS X)
Host	Specify an IPv4 address of the SSH server to be connected.	192.168.56.2
Port	Specify a port number of the SSH server to be connected.	22
User_name	Specify a login name when logging in to the SSH server.	user
Password_file	Specify a password file	secret\sshLogin.json

	that contains the password required to log in to the SSH server. Specify a relative path from the folder where the scenario file is located.	* This is an example of the case where the folder containing the scenario file has the secret folder and the password file (sshLogin.json) is located in that folder.
Private_key_file	It is not necessary to set when using password authentication.	
Prompt_string	Specify a string containing the end of the prompt that will be displayed when the login process is completed. If you want to specify more than one, enter with comma-separated values. (Example) Enter "$,#" for specifying "$" and "#."	"$ " " is not required. There is a space after $.
Timeout_value[sec]	Specify a maximum wait time in each step of the login process to the SSH server in seconds. Adjust with an appropriate value according to your environment.	10 * This setting value is just an example. Note that it may not operate depending on the environment.
Known_hosts_file	Specify a known hosts file for the hosts whose connections have been confirmed with the "Known hosts file generation tool." Specify a relative path from the folder where the scenario file is located.	[Only when setting a known hosts file] destination_hosts * This is an example of the case where the known hosts file (destination_hosts) is located directly under the folder where the scenario

		file is located.

[Public key authentication]

1. Generate a private key. For details on how to generate a private key, see the section of "SSH key generation tool" in the material No.3 in Table 5-1.
2. If you set a passphrase when generating a private key, generate a password file. For details on how to generate a password file, see the section of "Password file generation tool" in the material No.3 in Table 5-1.
3. Set the values according to Table 3-5. The setting values described here follow the example in Table 2-1. Environment used in this tutorial. Set the values according to your environment.

Table 3-5. Settings for public key authentication

Item	Description	Setting values in the example in this document
Authentication_method	Select the authentication method.	Public key authentication
Character_encoding	Specify a character encoding for importing into the server and exporting to WinActor Note.	UTF-8
Line-break_code_for_sending	Specify a line-break code for sending to the server.	LF(Linux, MacOS X)
Host	Specify an IPv4 address of the SSH server to be connected.	192.168.56.2
Port	Specify a port number of the SSH server to be connected.	22
User_name	Specify a login name when logging in to the SSH server.	user
Password_file	Specify a password file for the private key with a passphrase. Specify a	[Only if a passphrase for the private key is set] secret\sshLoginPassphras

	relative path from the folder where the scenario file is located. It is not necessary to set if you do not set a passphrase for the private key.	e.json * This is an example of the case where the folder containing the scenario file has the secret folder, and the password file (sshLoginPassphrase.json) is located in that folder.
Private_key_file	Specify a private key file. Specify a relative path from the folder where the scenario file is located.	secret\sshLoginKey * This is an example of the case where the folder containing the scenario file has the secret folder, and the private key file (sshLoginKey) is located in that folder.
Prompt_string	Specify a string containing the end of the prompt that will be displayed when the login process is completed. If you want to specify more than one, enter with comma-separated values. (Example) Enter "$,#" for specifying "$" and "#."	"$ " " is not required. There is a space after $.
Timeout_value[sec]	Specify a maximum wait time in each step of the login process to the SSH server in seconds. Adjust with an appropriate value according to your environment.	10 * This setting value is just an example. Note that it may not operate depending on the environment.
Known_hosts_file	Specify a known hosts file for the hosts whose	[Only when setting a known hosts file]

	connections have been confirmed with the "Known hosts file generation tool." Specify a relative path from the folder where the scenario file is located.	destination_hosts * This is an example of the case where the known hosts file (destination_hosts) is located directly under the folder where the scenario file is located.

3.2.2. Note_OpenSSHClient

This library is to open an SSH client.

Table 3-6. "Note_OpenSSHClient" library property settings

No.	Item	Description
①	Authentication_method	See 3.2.1.
②	Character_encoding	
③	Line-break_code_for_sending	
④	Host	
⑤	Port	
⑥	User_name	
⑦	Password_file	
⑧	Private_key_file	
⑨	Prompt_string	
⑩	Timeout_value[sec]	

3.2.3. Note_OpenSSHClient(KnownHosts)

This library is to open an SSH client by specifying a known hosts file.

For details on how to generate a known hosts file, see the section of "Known hosts file generation tool" in the material No.3 in Table 5-1.

Table 3-7. "Note_OpenSSHClient(KnownHosts)" library property settings

No.	Item	Description
①	Authentication_method	See 3.2.1.
②	Character_encoding	
③	Line-break_code_for_sending	
④	Host	
⑤	Port	
⑥	User_name	
⑦	Password_file	
⑧	Private_key_file	
⑨	Prompt_string	
⑩	Timeout_value[sec]	
⑪	Known_hosts_file	

3.2.4. Note_ExecuteCommand(SSHClient)

This library is to execute a command on an SSH client.

Table 3-8. "Note_ExecuteCommand(SSHClient)" library property settings

No.	Item	Description
①	Command	Specify a command you want to execute on the SSH client. Only text can be entered. Control characters cannot be sent.
②	Prompt_string	Specify a string containing the end of the message indicating that the command processing has been completed. It will usually be the same as the "Prompt_string" in 3.2.1.
③	Timeout_value[sec]	Specify a maximum wait time until the string of ② is displayed.

3.2.5. Note_CloseSSHClient

This library is to close a session of SSH client.

3.2.6. Note_SendFile(SCP)

This library is to send a file with SCP.

Table 3-9. "Note_SendFile(SCP)" library property settings

No.	Item	Description
①	Authentication_method	See 3.2.1.
②	Host	
③	Port	
④	User_name	
⑤	Password_file	
⑥	Private_key_file	
⑦	Timeout_value[sec]	
⑧	Source_file	Specify a file to be sent with a relative path from the folder where the scenario file is located. Only a single file can be specified. Multiple files or a folder cannot be specified.
⑨	Destination_path	Enter a storage destination path for the source file ⑧ on the SSH server.

3.2.7. Note_SendFile(SCP_KnownHosts)

This library is to send a file with SCP by specifying a known hosts file.

For details on how to generate a known hosts file, see the section of "Known hosts file generation tool" in the material No.3 in Table 5-1.

Table 3-10. "Note_SendFile(SCP_KnownHosts)" library property settings

No.	Item	Description
①	Authentication_method	See 3.2.1.
②	Host	
③	Port	
④	User_name	
⑤	Password_file	
⑥	Private_key_file	
⑦	Timeout_value[sec]	
⑧	Known_hosts_file	
⑨	Source_file	Specify a file to be sent with a relative path from the folder where the scenario file is located. Only a single file can be specified. Multiple files or a folder cannot be specified.
⑩	Destination_path	Enter a storage destination path for the source file ⑨ on the SSH server.

3.2.8. Note_ReceiveFile(SCP)

This library is to receive a file with SCP.

Table 3-11. "Note_ReceiveFile(SCP)" library property settings

No.	Item	Description
①	Authentication_method	See 3.2.1.
②	Host	
③	Port	
④	User_name	
⑤	Password_file	
⑥	Private_key_file	
⑦	Timeout_value[sec]	
⑧	Source_file	Specify a file to be received on the SSH server. A folder cannot be specified. In the　Environment used in this tutorial, wildcards (* and ?) can be used for a file.
⑨	Destination_path	Enter a storage destination path for the source file ⑧ on the SSH server.

3.2.9. Note_ReceiveFile(SCP_KnownHosts)

This library is to receive a file with SCP by specifying a known hosts file.

For details on how to generate a known hosts file, see the section of "Known hosts file generation tool" in the material No.3 in Table 5-1.

Table 3-12. "Note_ReceiveFile(SCP_KnownHosts)" library property settings

No.	Item	Description
①	Authentication_method	See 3.2.1.
②	Host	
③	Port	
④	User_name	
⑤	Password_file	
⑥	Private_key_file	
⑦	Timeout_value[sec]	
⑧	Known_hosts_file	
⑨	Source_file	Specify a file to be received on the SSH server. A folder cannot be specified. In the Environment used in this tutorial, wildcards (* and ?) can be used for a file.
⑩	Destination_path	Enter a storage destination path for the source file ⑨ on the SSH server.

3.2.10. Note_SendControlCode(SSHClient)

This library is to send a control code on an SSH client.

Table 3-13. "Note_SendControlCode(SSHClient)" library property settings

No.	Item	Description
①	Control_code	Specify a control code to be sent by an SSH client. Ctrl+A to Ctrl+Z can be specified.
②	Prompt_string	Specify a string containing the end of the message indicating that the command processing has been completed. It will usually be the same as the "Prompt_string" in 3.2.1.
③	Timeout_value[sec]	Specify a maximum wait time until the string of ② is displayed.

3.3. Telnet tool

This section describes the libraries for using the Telnet client function and the procedure to create a basic scenario using those libraries. The property of each library related to "Telnet " is described in 3.3.1 to 3.3.3.

[Procedure to execute the Telnet client function]

1. Placing the libraries in the Scenario box

   Place the following libraries in order in the Scenario box.

   3.3.1 Note_OpenTelnetClient

   3.3.2 Note_ExecuteCommand(TelnetClient)

   3.3.3 Note_CloseTelnetClient

   The scenario after placing these libraries is as shown in Figure 3-4.

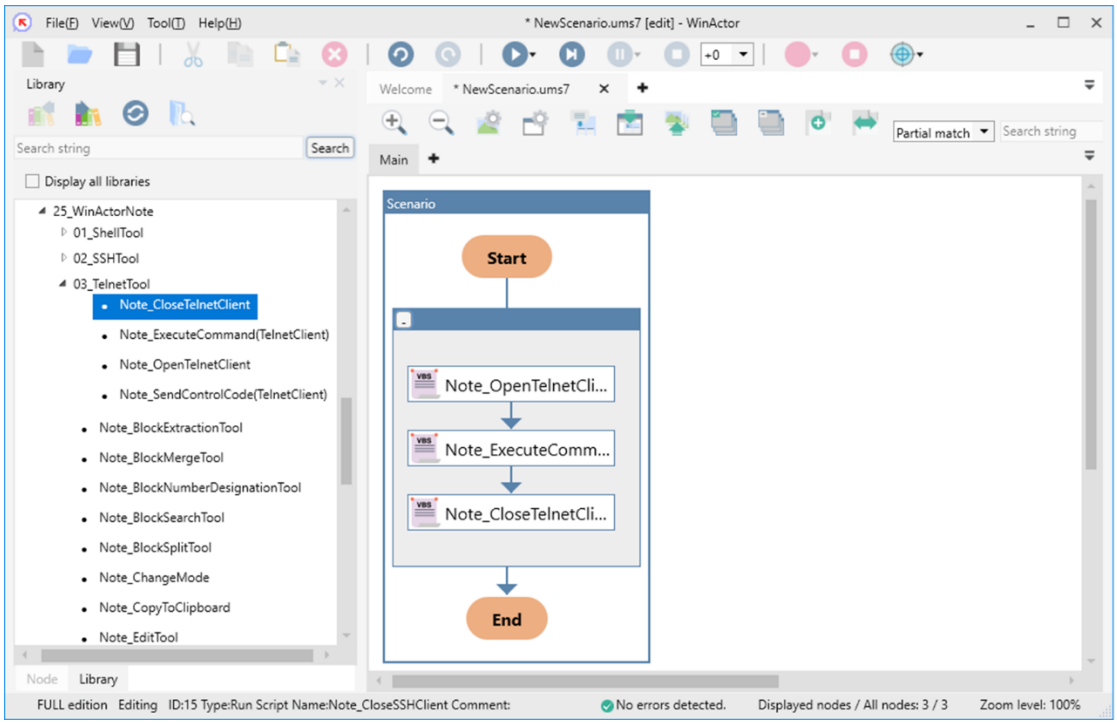


Figure 3-4. Example of the basic scenario using the Telnet client function

2. Setting the library properties

Set the properties according to the information in the subsections below.

3.3.1 Note_OpenTelnetClient

3.3.2 Note_ExecuteCommand(TelnetClient)

3.3.3 Note_CloseTelnetClient

3.3.1. Note_OpenTelnetClient

This library is to open a Telnet client.

Table 3-14. "Note_OpenTelnetClient" library property settings

No.	Item	Description
①	Character_encoding	Specify a character encoding for importing into the server and exporting to WinActor Note.
②	Host	Specify an IPv4 address of the Telnet server to be connected.
③	Port	Specify a port number of the Telnet server to be connected.
④	User_name	Specify a login name when logging in to the Telnet server.
⑤	Password_file	Specify a password file that contains the password required to log in to the Telnet server. Specify a relative path from the folder where the scenario file is located.
⑥	Login_prompt_string	Specify a string containing the end of the message that the Telnet server prompts for a username.
⑦	Password_prompt_string	Specify a string containing the end of the message that the Telnet server prompts for a password.
⑧	Command_prompt_string	Specify a string containing the end of the prompt that will be displayed when the login process is completed. If you want to specify more than one, enter with comma-separated values. (Example) Enter "$,#" for specifying "$" and "#."
⑨	Timeout_value[sec]	Specify a maximum wait time in each step of the login process to the Telnet server in seconds.

		Adjust with an appropriate value according to your environment.

*Line-break code for sending is fixed to CR+LF.

3.3.2. Note_ExecuteCommand(TelnetClient)

This library is to execute a command on a Telnet client.

Table 3-15. "Note_ExecuteCommand(TelnetClient)" library property settings

No.	Item	Description
①	Command	Specify a command you want to execute on a Telnet client. Only text can be entered. Control characters cannot be sent.
②	Prompt_string	Specify a string containing the end of the message indicating that the command processing has been completed. It will usually be the same as the "Command_prompt_string" in 3.3.1.
③	Timeout_value[sec]	Specify a maximum wait time until the string of ② is displayed.

3.3.3. Note_CloseTelnetClient

This library is to close a session of Telnet client.

3.3.4. Note_SendControlCode(TelnetClient)

This library is to send a control code on a Telnet client.

Table 3-16. "Note_SendControlCode(TelnetClient)" library property settings

No.	Item	Description
①	Control_code	Specify a control code to be sent by a Telnet client. Ctrl+A to Ctrl+Z can be specified.
②	Prompt_string	Specify a string containing the end of the message indicating that the command processing has been completed. It will usually be the same as

		"Command_prompt_string" in 3.3.1.
③	Timeout_value[sec]	Specify a maximum wait time until the string of ② is displayed.

4. Docking window

For descriptions of the docking window, see "WinActor Note Operation Manual."

5. Reference materials

Table 5-1 shows the materials referenced in this manual.

Table 5-1. Reference materials

No.	Material name
1	WinActor Operation Manual
2	WinActor User Library Sample Manual
3	WinActor Note Operation Manual
4	WinActor Note Text Processing Scenario Creation Manual

WinActor Note
Terminal Function Scenario
Creation Manual

NTT ADVANCED TECHNOLOGY CORPORATION

Copyright © 2013-2025 NTT, Inc. & NTT ADVANCED TECHNOLOGY CORPORATION

This document is protected under copyright law. It is forbidden to duplicate or copy any part or all of this document without prior consent.

The contents of this document are subject to change without notice.

WA7-P-20250603