OAuth-Related Library
Scenario Creation Manual

NTT ADVANCED TECHNOLOGY CORPORATION

Contents

WinActor OAuth-Related Library Scenario Creation Manual

1. Introduction

This manual describes the procedure to create a scenario to call a RESTful API of an external service using WinActor. To create a scenario to call a RESTful API of an external service, use the OAuth-related libraries on WinActor.

When using the OAuth-related libraries, knowledge of OAuth 2.0 authorization and RESTful APIs is required.

The OAuth-related libraries are a general term for libraries that provide an access to APIs with OAuth 2.0 authorization control or authorization information.

For external services assumed in this manual, see "1.1. Operating environment."

1.1. Operating environment

When calling a RESTful API of an external service using the OAuth-related libraries, it is assumed that the targeted external service implements the OAuth 2.0 access authorization control (RFC6749, RFC7009) and RESTful APIs.

The external service that has been successfully operated with OAuth-related libraries is shown in Table 1-1 below. (As of January 25, 2019)

Table 1-1. External service successfully operated with OAuth-related libraries

No.	Name of external service	API version
1	Salesforce.com (*1)	44.0

*1 Salesforce is a registered trademark of Salesforce.com, Inc.

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 ™, ®, and © marks are omitted in this document.

- WinActor is a registered trademark of NTT ADVANCED TECHNOLOGY CORPORATION.
- Microsoft, Windows*1, Microsoft Edge, Excel, and VBScript*2 are trademarks or registered trademarks of Microsoft Corporation in the United States and other countries.

  *1The official name of Windows is Microsoft Windows Operating System.
  *2The official name of VBScript is Microsoft Visual Basic Scripting Edition.
- The names of other companies and products are trademarks or registered trademarks of their respective companies.

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.

1.4. Limitations

There are following limitations when calling a RESTful API of an external service using the OAuth-related libraries.

- Only "Resource Owner Password Credentials" is available as a credential grant by OAuth 2.0.
- The OAuth-related libraries cannot be called from an external service.
- The protocol is limited to HTTPS due to security reasons.
- Available HTTP methods are GET, POST, PATCH, and DELETE only.
- OAuth 2.0 authorization information (such as an obtained access token and the like) cannot be extracted to WinActor variables.
- In the HTTP response headers of calling RESTful APIs, only the status code is available.
  For example, cookies or APIs returning information in the response headers cannot be used.

2. Scenario creation

You can call a RESTful API of an external service by using the OAuth-related libraries.

This chapter describes the procedure to create a basic scenario using the OAuth-related libraries of "OAuth_GetAccessToken," "OAuth_RunRESTfulAPI," and "OAuth_RevokeAccessToken."

[Steps]

2.1. Obtaining credentials

When performing the OAuth 2.0 authorization, it is necessary to obtain credentials from an external service in advance.

The required information for OAuth 2.0 authorization is shown in Table 2-1.

For information on how to obtain each information, check the documentation provided by the external service or contact the provider of the external service.

Table 2-1. List of credentials used for OAuth 2.0 authorization

No.	Information	Description
1	Username	This is a username of an account used to obtain an access token.
2	Password	This is a password for an account used to obtain an access token.
3	Client ID	This is a unique string obtained when registering with an external service, which corresponds to the client_id of RFC6749.
4	Client secret	Similar to the client ID, this is a secret string obtained when registering with an external service, and corresponds to the client_secret of RFC6749.
5	Security token	Use this property depending on external services. Set an empty string if not required.

		If this property is set, OAuth-related libraries use it with a password for authentication. (They are concatenated automatically.)

2.2. Placing the libraries in a scenario and setting the library property items

2.2.1. Placing the "OAuth_GetAccessToken" library and setting its property items

Select the Library tab, and double-click and expand "99_ExternalServiceLinkage" and "01_OAuth."

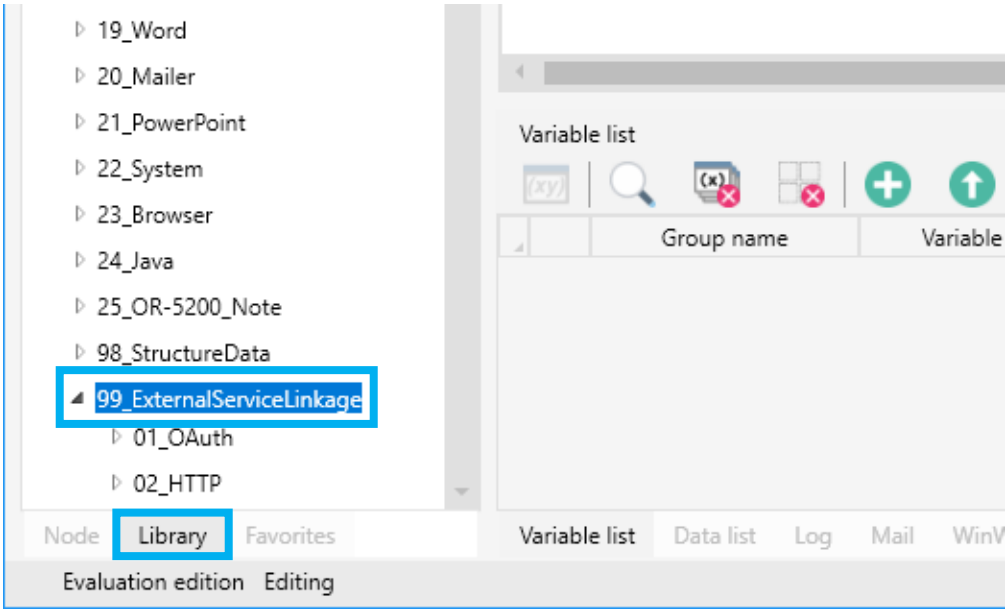

Figure 2-1. Expanding "99_ExternalServiceLinkage" on the Library tab

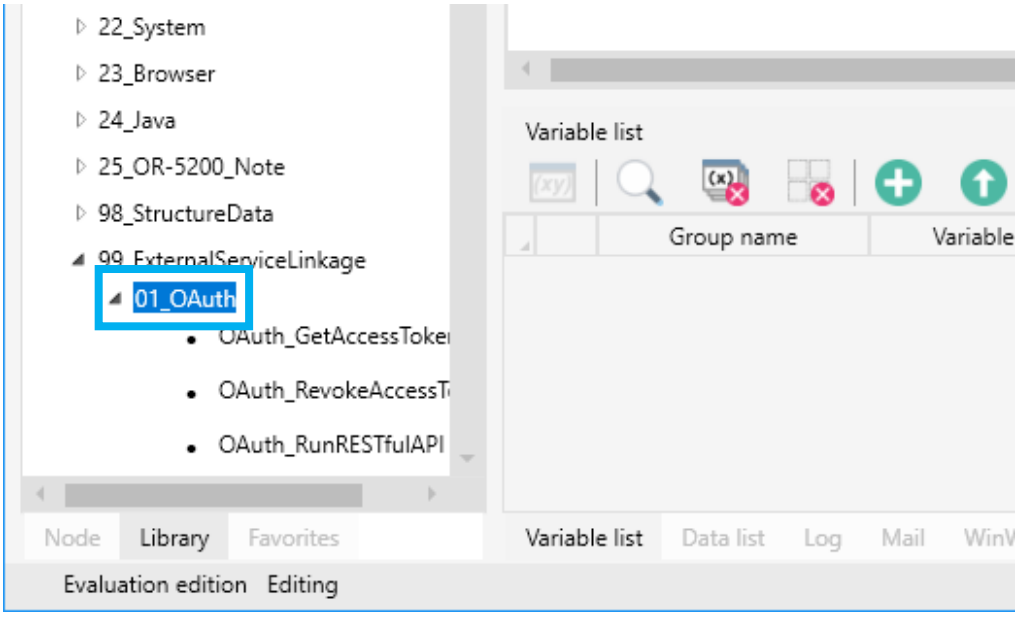

Figure 2-2. Expanding "01_OAuth" on the Library tab

Select "OAuth_GetAccessToken" on the Library tab (① in Figure 2-3), and drag and drop it into the Scenario box (② in Figure 2-3). Then, double-click the "OAuth_GetAccessToken" library placed in the Scenario box to display the Property window (③ in Figure 2-4).

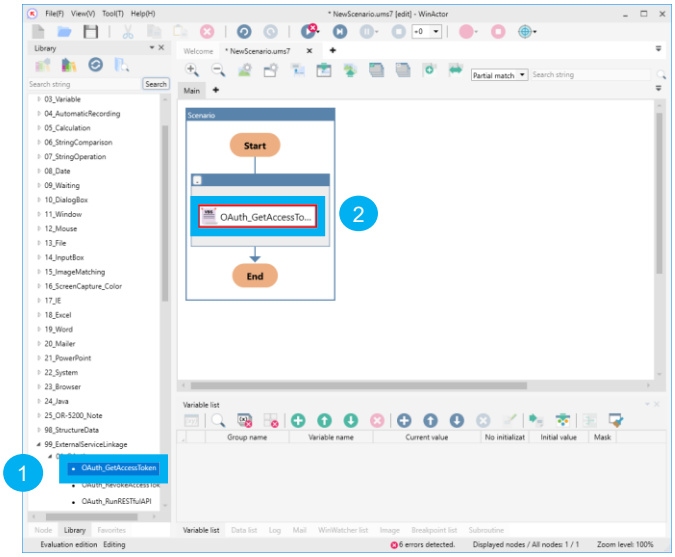

Figure 2-3. Placing the "OAuth_GetAccessToken" library in the Scenario box

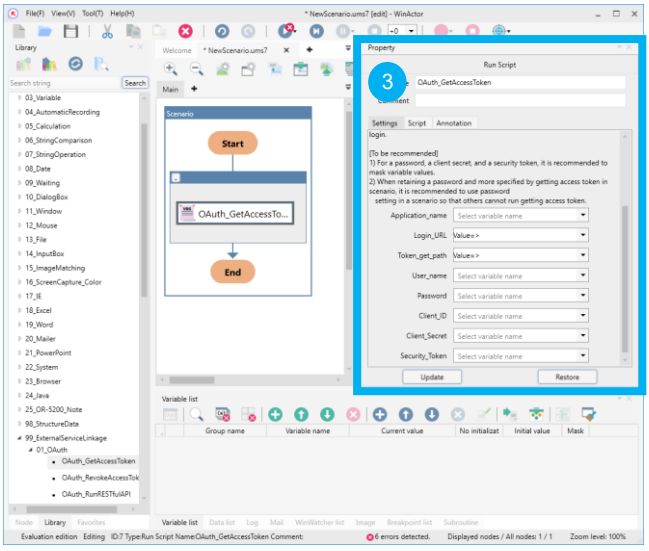

Figure 2-4. "OAuth_GetAccessToken" library property

Set the property items and click the Update button. For the description of each property item, see "3.1. OAuth_GetAccessToken." For Application_name, see descriptions in "3. Library and property list."

2.2.2. Placing the "OAuth_RunRESTfulAPI" library and setting its property items

As with 2.2.1, select "OAuth_RunRESTfulAPI" on the Library tab (① in Figure 2-5), and drag and drop it into the Scenario box (② in Figure 2-5). Then, double-click the "OAuth_RunRESTfulAPI" library placed in the Scenario box to display the Property window (③ in Figure 2-6).

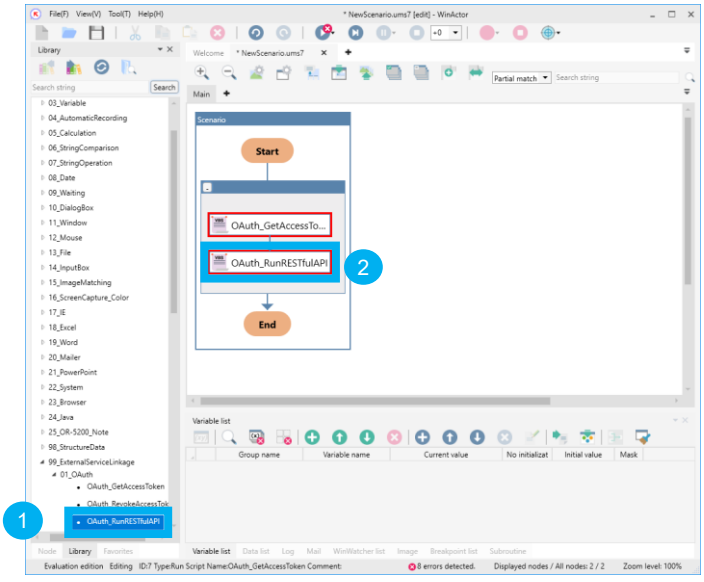

Figure 2-5. Placing the "OAuth_RunRESTfulAPI" library in the Scenario box

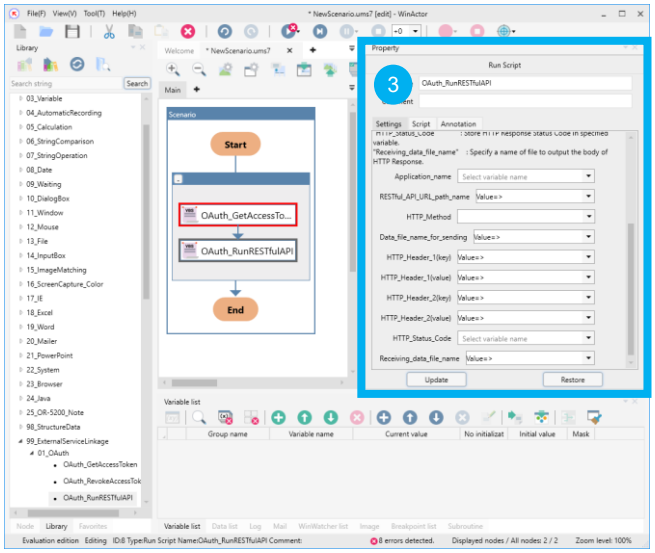

Figure 2-6. "OAuth_RunRESTfulAPI" library property

Set the property items and click the Update button. For the description of each property item, see "3.2. OAuth_RunRESTfulAPI." For Application_name, see descriptions in "3. Library and property list."

2.2.3. Placing the "OAuth_RevokeAccessToken" library and setting its property items

As with 2.2.1 and 2.2.2, select "OAuth_RevokeAccessToken" on the Library tab (① in Figure 2-7), and drag and drop it into the Scenario box (② in Figure 2-7). Then, double-click the "OAuth_RevokeAccessToken" library placed in the Scenario box to display the Property window (③ in Figure 2-8).

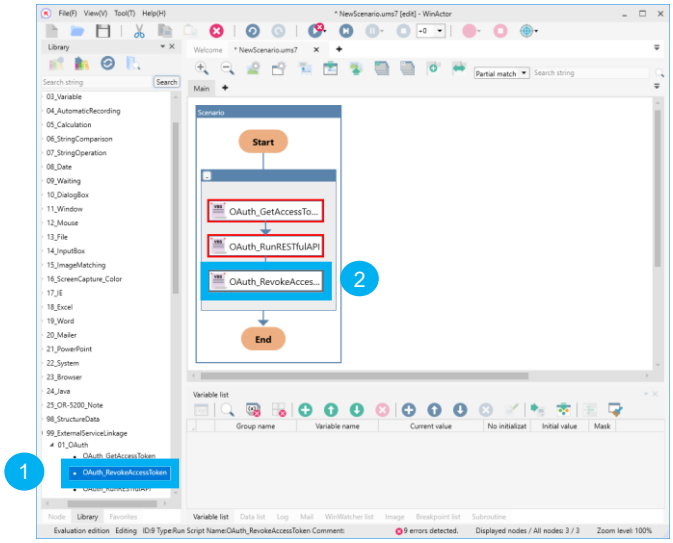

Figure 2-7. Placing the "OAuth_RevokeAccessToken" library in the Scenario box

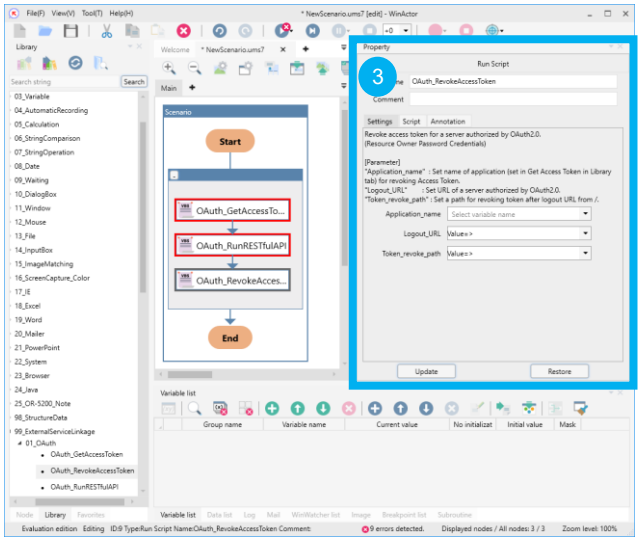

Figure 2-8. "OAuth_RevokeAccessToken" library property

Set the property items and click the Update button. For the description of each property item, see "3.3. OAuth_RevokeAccessToken." For Application_name, see descriptions in "3. Library and property list."

3. Library and property list

This chapter describes the OAuth-related libraries and their property items.

In each OAuth-related library, there is a common property item "Application_name".

Application_name is a unique string of one or more characters used to identify a connection destination. By using the same Application_name, the same credentials and authorization information can be shared by multiple API calls.

The following is an example of calling a RESTful API using two accounts A and B.

Example:

1) Get an access token for account A by setting "a" as the Application_name.
2) Get an access token for account B by setting "b" as the Application_name.
3) Call a RESTful API by setting "a" as the Application_name.
   * Call the API as account A.
4) Call a RESTful API by setting "b" as the Application_name.
   * Call the API as account B.
5) Revoke the access token by setting "a" as the Application_name.
   * The information of account A is revoked.
6) Revoke the access token by setting "b" as the Application_name.
   * The information of account B is revoked.

It is recommended that the account and Application_name have a one-to-one correspondence as described above.

The credentials (account information) and authorization information linked to the Application_name should be explicitly revoked using the "OAuth_RevokeAccessToken" library in a scenario.

Considering a runtime error such as a network error during an API call, it’s better to put the "OAuth_RevokeAccessToken" library after exception handling. It ensures the credentials (account information) and authorization information are always revoked.

3.1. OAuth_GetAccessToken

The "OAuth_GetAccessToken" library is to obtain an access token from the OAuth 2.0 authorization server.

The authorization will be performed based on the credentials specified by the user, and the authorization information will be returned. If it is successful, the credentials and authorization information (access token, resource server URL*) will be linked with the Application_name.

* Since the access token and the resource server URL are managed in association with the Application_name, there is no need for the user to take care of them.

Table 3-1. Library and property items

Library name	Property item	Description
OAuth_GetAccessToken	Application_name	Specify a string to identify a connection destination with a variable.
	Login_URL	Specify a host name of the OAuth 2.0 authorization server with a value or a variable. (*1 and *2)
	Token_get_path	Specify a path name to obtain an access token from OAuth 2.0 authorization server with a value or a variable. (*2 and *3)
	User_name	Specify credentials obtained in "2.1. Obtaining credentials" with a variable.
	Password	
	Client_ID	
	Client_Secret	
	Security_Token	

*1 Specify a URL for HTTPS protocol

*2 An example of specifying Login_URL and Token_get_path is shown below.

Example:

URL for getting an access token : "https://access.example.com/oauth2/token"

Login_URL : "https://access.example.com"

Token_get_path : "/oauth2/token"

*3 Specify a path name that starts with "/" character.

3.2. OAuth_RunRESTfulAPI

The "OAuth_RunRESTfulAPI" library is to call a RESTful API that requires the OAuth 2.0 authorization.

Based on the authorization information (access token, resource server URL*) obtained by the "OAuth_GetAccessToken" library, a RESTful API is called for a specified URL, and an HTTP status code and a response body are returned. If the access token has expired at the

time of access, the access token will be updated at the same time.

The "OAuth_RunRESTfulAPI" library receives a response from the resource server. The user should explicitly check the result of the API call in a scenario.

Table 3-2. Library and property items

Library name	Property item	Description
OAuth_RunRESTfulAPI	Application_name	Specify a string to identify a connection destination with a variable. (*1)
	RESTful_API_URL_path_name	Specify a path name of the RESTful API with a value or a variable. (*2 and *3)
	HTTP_Method	Select an HTTP method for the request. PATCH, GET, POST, and DELETE are available.
	Data_file_name_for_sending	Specify a file path to be sent as a request body with a value or a variable. (*4) If omitted, a request with an empty body will be sent.
	HTTP_Header_1(key) HTTP_Header_1(value) HTTP_Header_2(key) HTTP_Header_2(value)	Specify a name and contents of a request header with a value or a variable. (*5 and *6)
	HTTP_Status_Code	Specify a variable to store the status code in the received response.
	Receiving_data_file_name	Specify a file path to output the received response body with a value or a variable. (*4) If omitted or the response

		body is empty, an output to a file will not be performed.

*1 Specify the same Application_name as the one specified in the "OAuth_GetAccessToken" library placed previously in the scenario.

*2 Specify a path name that starts with "/" character.

*3 An example of specifying RESTful_API_URL_path_name is shown below. The resource server URL is the one obtained from the "OAuth_GetAccessToken" library.

Example:

URL for RESTful API execution : "https://resource.example.com/services/data"
Resource server URL : "https://resource.example.com"
RESTful_API_URL_path_name : "/services/data"

*4 If a relative path is specified, it will be the relative path from the folder where WinActor is installed (the folder where WinActor7.exe is located).

*5 The HTTP_Header can be specified up to two pairs of HTTP_Header_1 and HTTP _Header_2. If HTTP_Header is unnecessary, set an empty string.

*6 Authorization header that specifies the OAuth 2.0 authorization information will be automatically set.

3.3. OAuth_RevokeAccessToken

The "OAuth_RevokeAccessToken" library is to request the OAuth 2.0 authorization server to revoke the access token.

This library will go to the initial state before obtaining an access token regardless of the result of the revocation.

Table 3-3. Library and property items

Library name	Property item	Description
OAuth_RevokeAccessToken	Application_name	Specify a string to identify a connection destination with a variable.
	Logout_URL	Specify a host name of the OAuth 2.0 authorization server with a value or a variable. (*1 and *2)

	Token_revoke_path	Specify a path name to make a request to the OAuth 2.0 authorization server for revoking an access token with a value or a variable. (*2 and *3)

*1 Specify an empty string or a URL for HTTPS protocol.
If an empty string is specified, a URL of resource server will be used.

*2 An example of specifying Logout_URL and Token_revoke_path is shown below.

Example:

URL for revoking an access token : "https://access.example.com/oauth2/revoke"
Logout_URL : "https://access.example.com"
Token_revoke_path : "/oauth2/revoke"

*3 Specify a path name that starts with "/" character.

4. Special notes

This chapter describes special notes when calling a RESTful API of an external service using the OAuth-related libraries.

4.1. Protection of credentials

・For User_name, Password, Client_ID, Client_Secret, and Security_Token, which are the properties of "OAuth_GetAccessToken" library, it is recommended to use a masking function for variable values so that no one else can see the variable values. A variable value can be masked by checking the "Mask" column in the Variable list.

* For details on masking variable values, see "WinActor Operation Manual."

・When retaining a password and the like specified in the "OAuth_GetAccessToken" library into a scenario, it is recommended to use the scenario password to prevent others from running the scenario.

* For details on the scenario password, see "WinActor Operation Manual."

4.2. Analyzing runtime errors

This section describes how to analyze runtime errors. If an error occurs when a scenario is running, the OAuth-related libraries output the message "Failed to run Library (spv_Web)." in the log area. The detailed information of the error can be collected by combining the "Exception Handling" node and the "Debugging_CollectSPVErrorInformation" library. The procedure is as follows:

1. Drag and drop the "Exception Handling" node into the Scenario box.
2. Drag and drop the "OAuth_GetAccessToken" library into the Normal flow of the "Exception Handling" node.
3. Drag and drop the following libraries and node in order into the Exceptional flow of the "Exception Handling" node.
    1) "Debugging_CollectSPVErrorInformation" library
    2) "Clipboard" node
    3) "TextFile_WriteTextToFile" library

* Since the "Debugging_CollectSPVErrorInformation" library outputs information to the clipboard, place the "Clipboard" node to copy the information to a variable (variable name example: error_log) and then place the "TextFile_WriteTextToFile" library to write

the information to the specified file.

Figure 4-1 shows the diagram of libraries and nodes arranged in the Scenario box.

* The "OAuth_GetAccessToken" library is used as an example in this section. For the "OAuth_RunRESTfulAPI" library and "OAuth_RevokeAccessToken" library, error information can be collected by the same procedure.

* For details on the "Exception Handling" node, see "WinActor Operation Manual."
* For details on the "Debugging_CollectSPVErrorInformation" library, see "WinActor User Library Sample Manual."

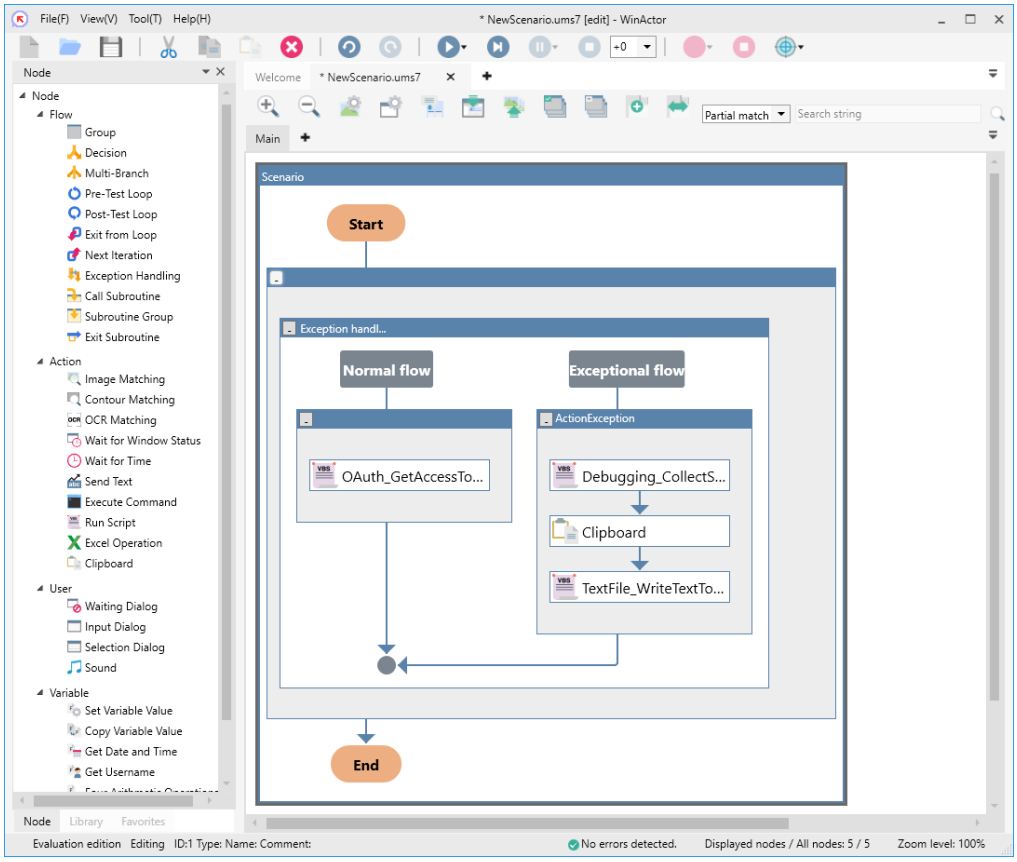

Figure 4-1. How to collect runtime error information

Table 4-1 shows examples of error information obtained by the "Debugging_CollectSPVErrorInformation" library.

Table 4-1. Examples of SPV error information

Error information	Error details and check points
Unintended response. (code:*xxx*) <*snip*> "*xxx*" is a number	Indicates that an external service detected an error. Make sure that the User_name, Password, Client_ID, and Client_Secret are entered and the password is correct.
Please set token acquisition path with starting / <*snip*>	Indicates that the Token_get_path does not start with "/." Check the Token_get_path.
Please set REST URL path with starting / <*snip*>	Indicates that the RESTful_API_URL_path_name does not start with "/." Check the RESTful_API_URL_path_name.
Please set token revocation path with starting / <*snip*>	Indicates that the Token_revoke_path does not start with "/." Check the Token_revoke_path.
Failed to read file. (*filename-for-sending*) <*snip*>	Indicates that the Data_file_name_for_sending could not be read. Check that the path of the Data_file_name_for_sending is correct and the file exists.
Failed to write file. (*receiving-filename*) <*snip*>	Indicates that the Receiving_data_file_name could not be written. Check that the path of the Receiving_data_file_name is correct.
unmatch argument designation between key and value. (key:, value:*xxx*) <*snip*> *xxx* is a value	Indicates that the key value of an HTTP header is empty. Set a correct key.

unmatch argument designation between key and value. (key:*xxx*, value:) <*snip*> *xxx* is a key	Indicates that a value of an HTTP header is empty. Set a correct value.
failed to communicate. <*snip*>	Indicates that the communication has failed. Make sure that the specified URL is correct, and also check the network connection settings. When a proxy is used, make sure that the proxy server is configured correctly.
Application has not been login yet. <*snip*>	Indicates that the access token has not been obtained. Before this error occurs, run the "OAuth_GetAccessToken" library and confirm that it is successful.

4.3. Use of a proxy

If the OAuth-related libraries need to go through a proxy server when accessing an external service, make sure that the proxy server of WinActor is configured correctly.

For how to set the proxy server of WinActor, see "WinActor Installation Manual" or "WinActor Operation Manual."

WinActor®

OAuth-Related Library Scenario Creation Manual

NTT ADVANCED TECHNOLOGY CORPORATION

WA7-G-20250603