Program Guide

N-Server Interface Library (Zr.Library.dll)

User Manual

Web-N Server ver.2.0.6

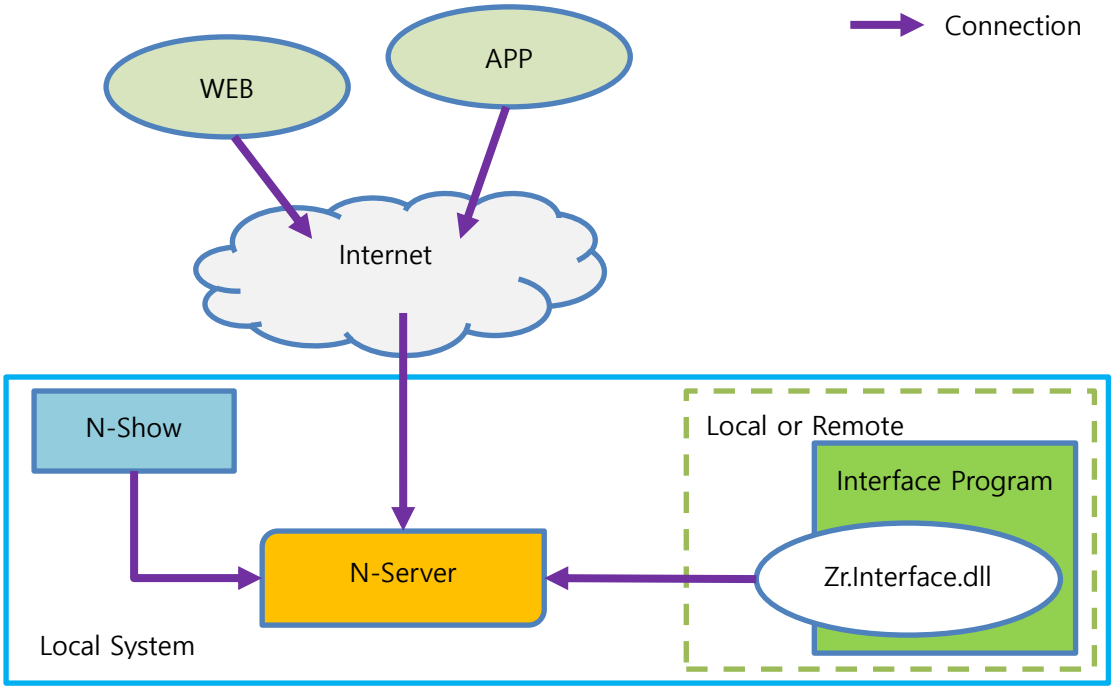
Production Date: 2017. 11. 03

Revision Date:.

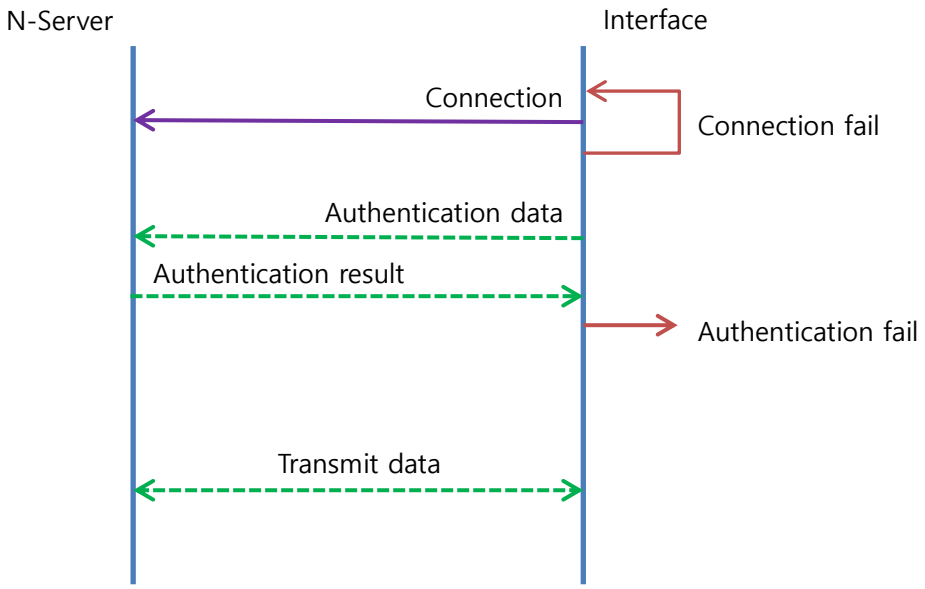
e-mail:Lbhsb@naver.com

Contents

1 System Block diagram



2 Data Flow Chart



2 Library

2.1 Basic Information

```
namespace Zr.Interface
{
    [DefaultMember("Item")]
    public sealed class Module : _ClientBase {

        public Module();
        public Module(string moduleName, byte systemNumber);
        public Module(string moduleName, byte systemNumber, InterfaceSecurityTypes
securityType);
        public Module(string moduleName, byte systemNumber, string ip, int port);
        public Module(string moduleName, byte systemNumber, string ip, int port,
InterfaceSecurityTypes securityType);
     }
}
```

2.2 Properties

Name	Description or content	Remarks
ModuleName	Module name	Read only
SecurityType	Security Type	Read only
AuthenticationKey	Security (authentication) key Applies only when using security type [SymmetricKey / AsymmetricKey]	Read only
SystemNumber	Module number	Read only

2.3 Events

Event Name	Description or content
ErrorCode(byte code, string message)	Error message event
AuthenticationAck(byte key)	Authentication request success event
AuthenticationNock(AlivcResultTypes type)	Authentication request failure event
ReceivedValue(string name, InterfaceFormatTypes type, object value)	Received data event
Connected()	Connection Events
Disconnected(_ClientBase source)	Disconnect event

2.4 Methods

Method names	Description or content
Version GetVersion()	Get library version
void Initialize(string name, string ip, int port)	Basic information insertion
void Start(int delay)	Start server connection
void Stop()	Stop server connection
void RequestAuthentication()	Request authentication from server
void SetSystemNumber(byte number)	Insert module number
void SetSecurityType(InterfaceSecurityTypes type)	Specifying security type
void SetAuthenticationKey(string key)	Specifying the Security (Authentication) Key Used only when Security Type is [SymmetricKey / AsymmetricKey]
ErrorCodes SetTag(uint index, string name, InterfaceTagTypes type)	Register Tag
ErrorCodes SetTag(uint index, string name, string type)	Register Tag
ErrorCodes ReleaseTag(string name)	Release Tag
ErrorCodes ClearTag()	Releasing all tags
ErrorCodes Change(string name, bool value)	Pass the value change to the server (Boolean tag)
ErrorCodes Change(string name, double value)	Pass the value change to the server (double tag)
ErrorCodes Change(string name, string value)	Pass the value change to the server (string tag)

3 Tutorials

3.1 Constructor

Default constructor

```
public class Example
{
    Module _Module;
    public Example ()
    {
        /* contstructor */
        _Module = new Module();
    }
}
```

Specify Module name and System number

```
public class Example
{
    Module _Module;
    public Example ()
    {
        /* contstructor */
        _Module = new Module("Test Module", 1);
    }
}
```

Specify Module name, System number and Security type

```
public class Example
{
    Module _Module;
    public Example ()
    {
        /* contstructor */
        _Module = new Module("Test Module", 1, InterfaceSecurityTypes.Simple);
    }
}
```

Specify Module name, System number, and Server connection information

- The server port number is fixed (19653)..

```
public class Example
{
    Module _Module;
    public Example ()
    {
        /* contstructor */
        _Module = new Module("Test Module", 1, "127.0.0.1", 19653);
    }
}
```

Specify Module name and System number, Server connection information and Security type

- The server port number is fixed (19653)..

```
public class Example
{
    Module _Module;
    public Example ()
    {
        /* contstructor */
        _Module    =    new    Module("Test    Module",    1,    "127.0.0.1",    19653,
InterfaceSecurityTypes.Simple);

    }
}
```

3.2 Initialization

```
public class Example
{
    Module _Module;
    public Example ()
    {
         _Module = new Module();

        /* initialize */
        _Module.Initialize(("Test Module", "127.0.0.1", 19653);

        /* set system number - (0x00 < system number < 0x3F) */
        _Module.SetSystemNumber(1);

        /* set security type(none|simple|symmetric|asymmetric) */
        _Module.SetSecurityType(InterfaceSecurityTypes.Simple);

        /* set security key(len = 8) */
        _Module.SetAuthenticationKey("12345678");
    }
}
```

- Specifies the module name, server IP, and port number.
- Specify the module number
  - Number to identify Interface module in N-Server
  - If not specified, default 1 is specified.
  - Module numbers can be from 1 to 127.
- Specify the security type (if not specified, the default Simple type is specified))
  - InterfaceSecurityTypes.None
  - InterfaceSecurityTypes.Simple
  - InterfaceSecurityTypes.SymmetricKey
  - InterfaceSecurityTypes.AsymmetrickKey
- If the security type is SymmetricKey or AsymmetricKey, apply the security (authentication) key
  - The key length must be 8 characters.

3.3 Tag registration and release

```
public class Example
{
    Module _Module;
    public Example ()
    {
        /* contstructor */
        _Module = new Module("Test Module", 1, "127.0.0.1", 19653);

        /* set tag */
        ErrorCodes error = _Module.SetTag((uint)0, "TAG00001", "Digital");

        /* release tag */
        ErrorCodes error2 = _Module.ReleaseTag("TAG00001");
    }
}
```

- It is possible to register a used tag by specifying tag index, name, and type.
  - ■ When registration is successful, ErrorCode returns [Success], otherwise [Error] code is returned.
- You can remove (disable) the tag with the tag name.
  - ■ When registration is successful, ErrorCode returns [Success], otherwise [Error] code is returned.
- The tag name must be the same as the registered tag name when editing the tag in N-Designer.

3.4 Clear all tags

```
public class Example
{
    Module _Module;
    public Example ()
    {
        /* contstructor */
        _Module = new Module("Test Module", 1, "127.0.0.1", 19653);

        /* set tag */
        ErrorCodes error = _Module.ClearTags();
    }
}
```

- Release all registered tags.

3.5 Start and stop server connections

```
public class Example
{
    Module _Module;
    public Example ()
    {
        /* contstructor */
        _Module = new Module("Test Module", 1, "127.0.0.1", 19653);

        /* start connection */
        _Module.Start(1000);

        /* stop connection */
        _Module.Stop();
    }
}
```

- Start by specifying the server reconnection delay (in ms).

3.6 Sending data to the N-Server.

```
public class Example
{
    Module _Module;
    public Example ()
    {
        /* contstructor */
        _Module = new Module("Test Module", 1, "127.0.0.1", 19653);

        /* Send to server(type boolean) */
        ErrorCodes error1 = _Module.Change("TAG00001", true);

        /* Send to server(type double) */
        ErrorCodes error2 = _Module.Change("TAG00002", 12.5);

        /* Send to server(type string) */
        ErrorCodes error3 = _Module.Change("TAG00003", "test message");
    }
}
```

- Pass the change value by type.
  - ■ If successful, ErrorCode returns [Success]. Otherwise, the corresponding code is returned.

3.7 Processing Received Data from N-Server

```
public class Example
{
    Module _Module;
    public Example ()
    {
        /* contstructor */
        _Module = new Module("Test Module", 1, "127.0.0.1", 19653);

        /* set received value event */
        _Module.ReceivedValue += _Module_ReceivedValue;
    }
    private void _Module_ReceivedValue(string name, InterfaceFormatTypes type, object
value)
    {
        /* casting (InterfaceFormatTypes) */
        switch (type)
        {
            case InterfaceFormatTypes.Float:
                var floatValue = (float)value;
                break;
            case InterfaceFormatTypes.Double:
                var doubleValue = (double)value;
                break;
            case InterfaceFormatTypes.Boolean:
                var boolValue = (bool)value;
                break;

            case InterfaceFormatTypes.String:
                var stringValue = (string)value;
                break;

            case InterfaceFormatTypes.Json:
                /* json format(custom) */
                var jsonValue = (string)value;
                break;
            }
    }
}
```

- Cast the value according to the value type (InterfaceFormatTypes) and process it.

3.8 Error Handling

```
public class Example
{
    Module _Module;
    public Example ()
    {
        /* contstructor */
        _Module = new Module("Test Module", 1, "127.0.0.1", 19653);

        /* set error code event */
        _Module.ErrorCode += _Module_ErrorCode;
    }
    private void _Module_ErrorCode(ErrorCodes code, string message)
    {
    }
}
```

- Receive the corresponding code and message in case of receiving error and library internal error.
  - ■ No tag name when receiving value
  - ■ Module request failed

3.9 Other Actions

```
public class Example
{
    Module _Module;
    public Example ()
    {
        /* contstructor */
        _Module = new Module("Test Module", 1, "127.0.0.1", 19653);

        /* set disconnected & connected event */
        _Module.Disconnected += _Module_Disconnected;
        _Module.Connected += _Module_Connected;
        /* set authentication ack & nock(fail) event */
        _Module.AuthenticationAck += _Module_AuthenticationAck;
        _module_AuthenticationNock += _Module_AuthenticationNock;
    }
    private void _Module_Disconnected (_ClientBase client)
    {
    }
    private void _Module_Connected ()
    {
    }
    private void _Module_AuthenticationAck(byte key)
    {
        /* received authentication ok */
    }
    private void _Module_AuthenticationNock(AliveResultTypes resultType)
    {
        /* retry request authentication */
        _Module.RequestAuthentication();
    }
}
```

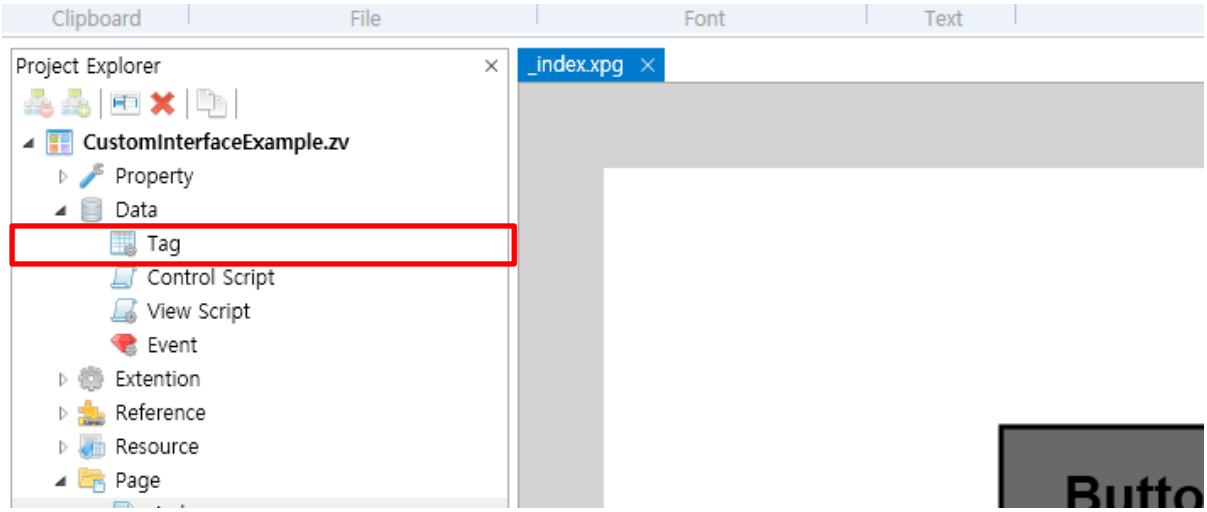
- Handle the server connection and disconnection state in the corresponding event.
  - ■ Module authentication is requested automatically when reconnection is successful.
- Process after the module authentication in the relevant event.
  - ■ Key is the module number registered by the module

4 Example program

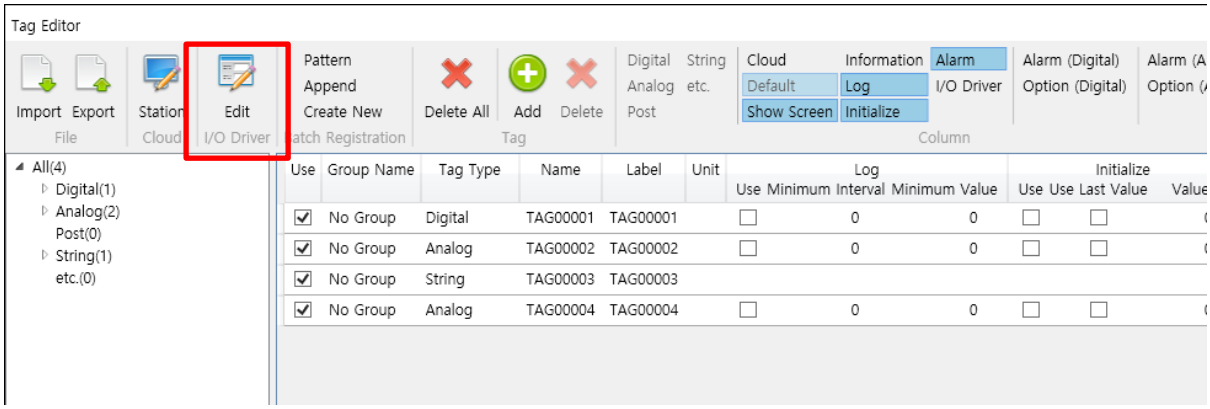
4.1 Registering tags in Web-N Designer

4.1.1 Editing the I/O Driver

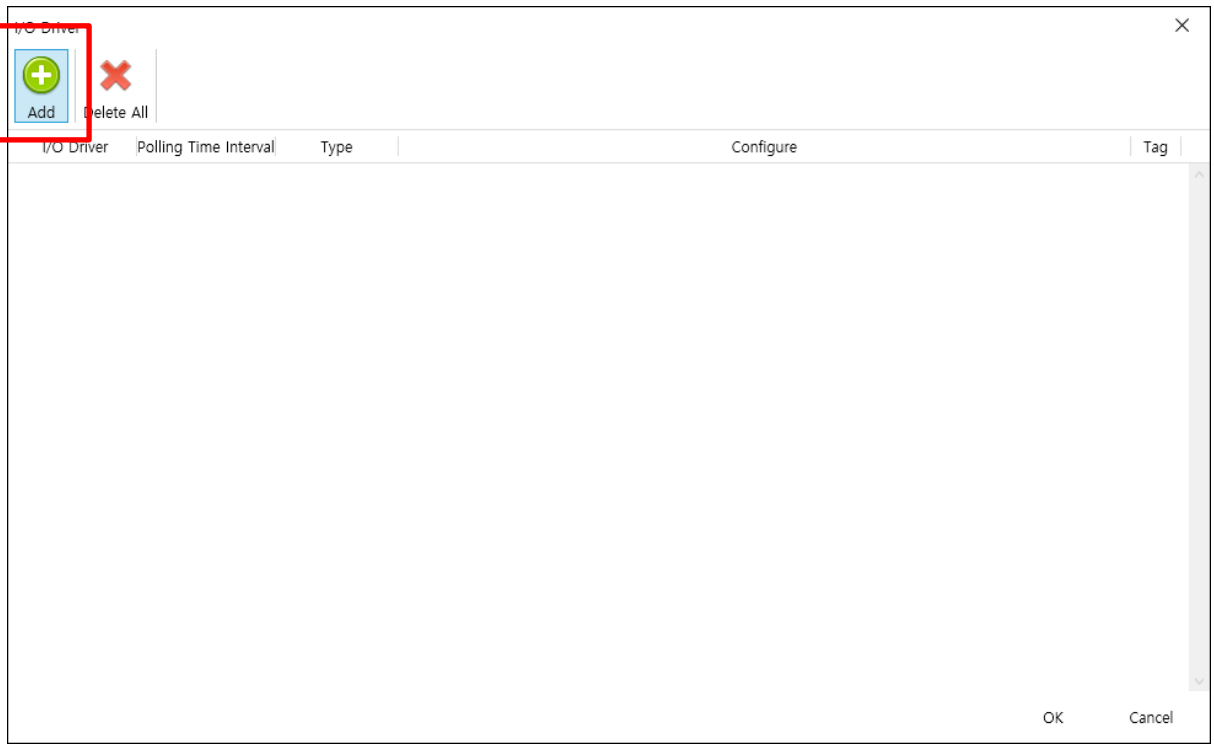
1. Double-click [Data] - [Tag] in the left Project Browser.



2. After registering the required tag, click the [Edit] button of the [IO Driver] in the upper menu item.

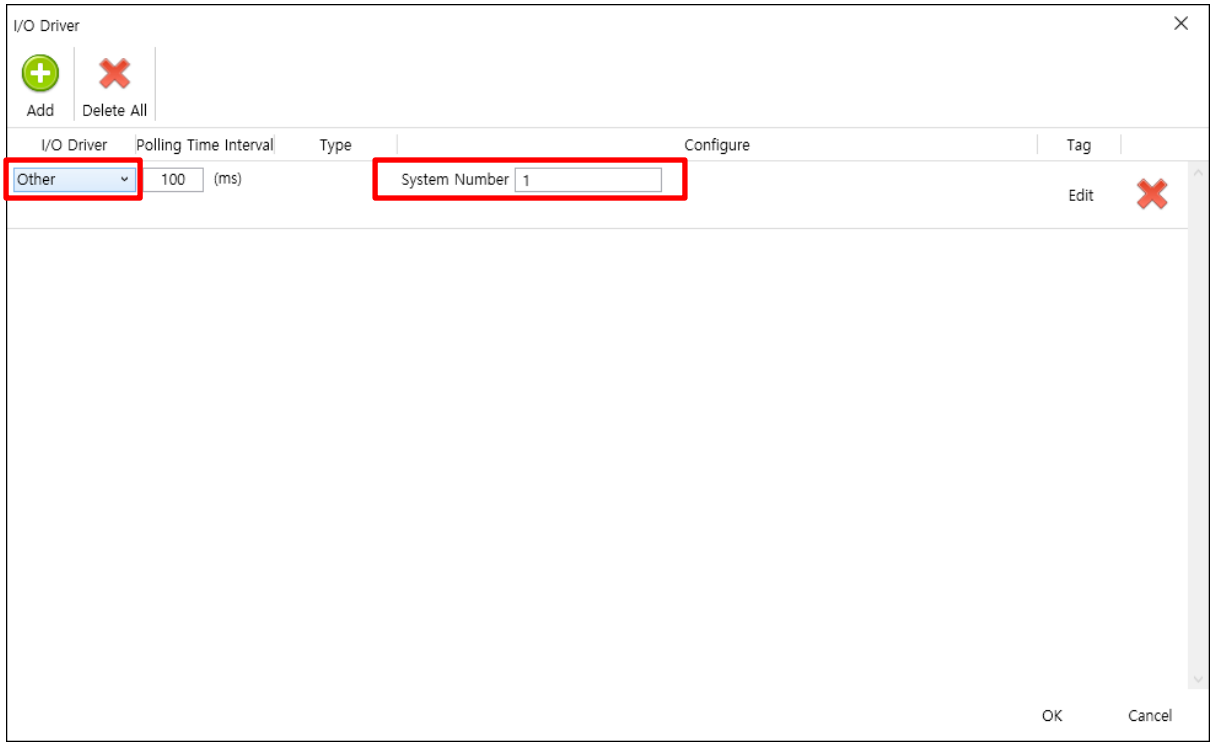


3. Click the top [Add] button to create a new IO driver entry.



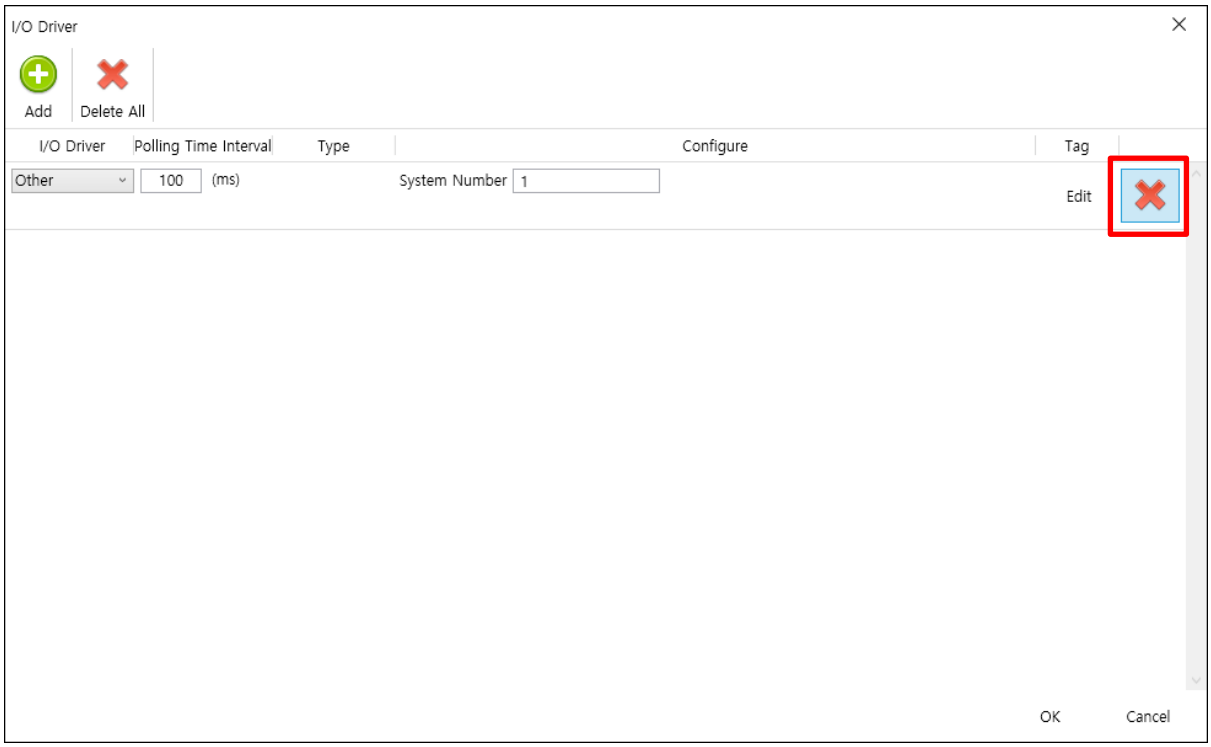
4. Select [Other] for the type of IO driver created, and enter the system number.

- Enter the system (module) number to be used in the interface.
- The system (module) number must be unique for each module.
- System (module) number can be set from 1 to 127.



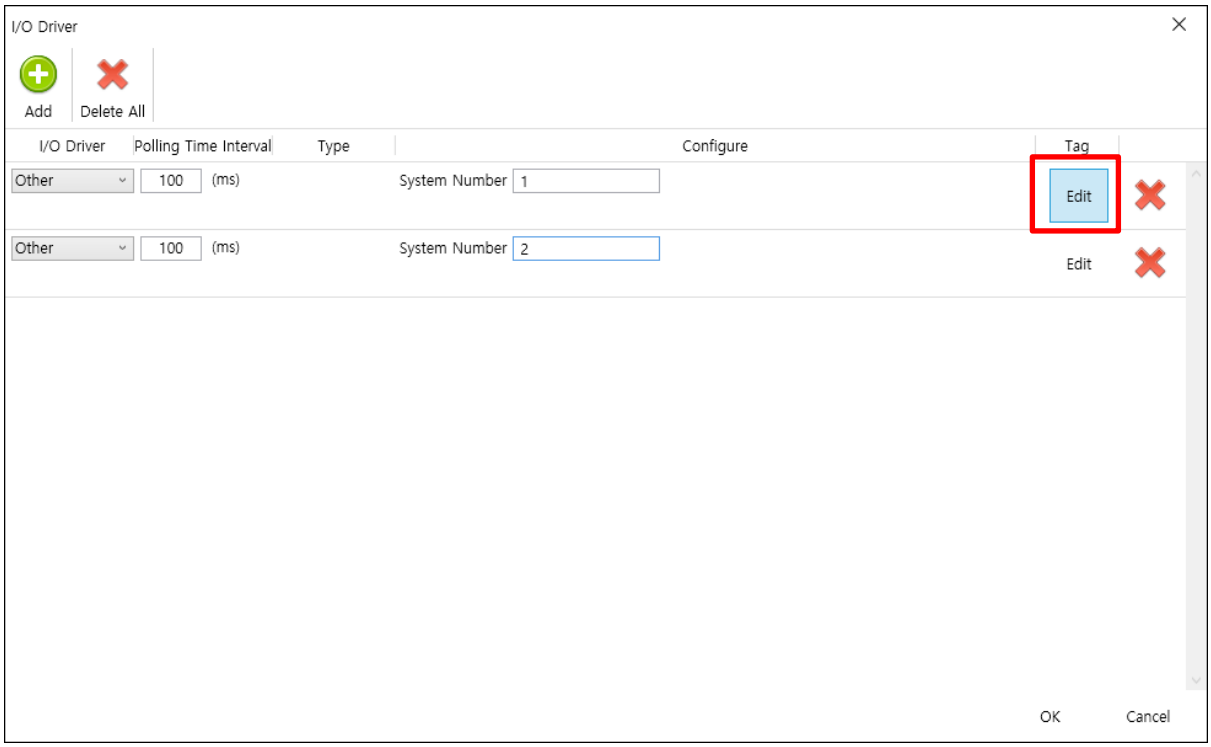
5. If you want to delete the created IO driver item, click the [X] icon to the right of the item.

- Note that there should be no tags bound to the I/O driver that you want to delete.

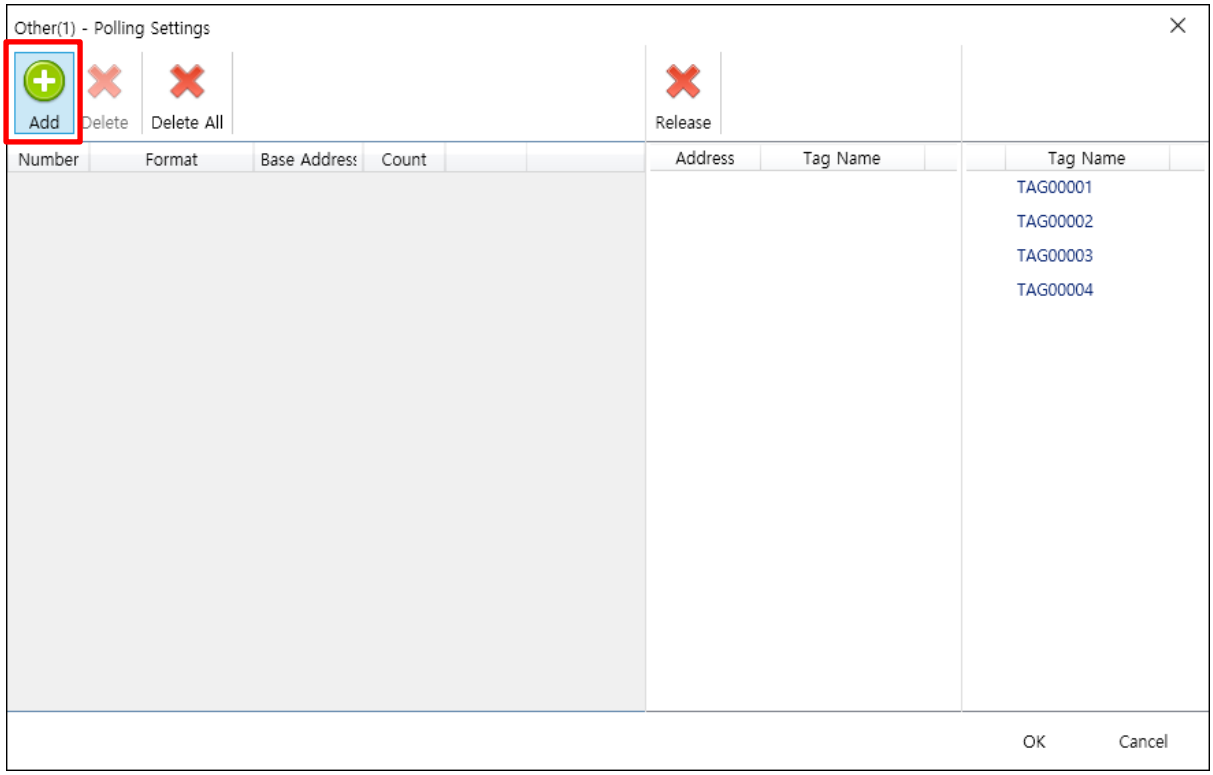


4.1.2 Tag mapping

1. Click the [Edit] button to the right of the item to be registered.

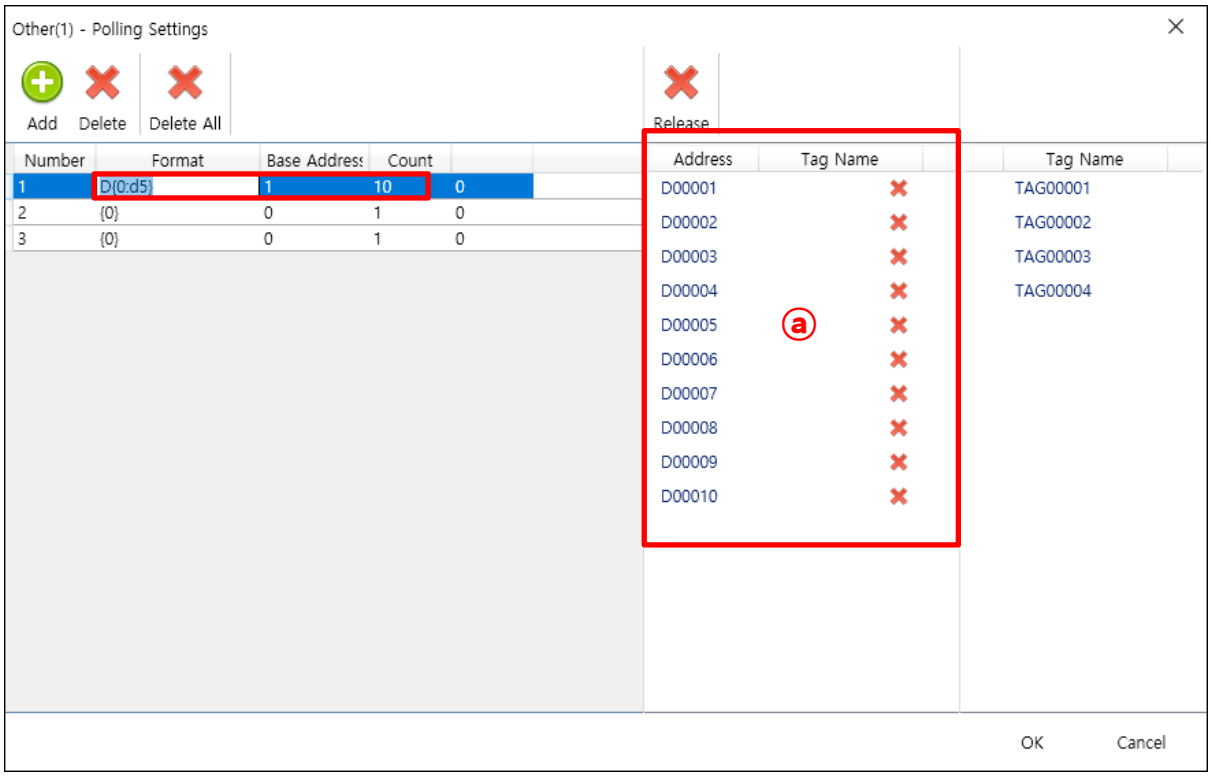


2. Click the top [Add] button to create a new item.



3. Create a tag to associate by entering the tag name format to be used in the interface.

- If you specify the tag format to be used and modify the start address and number, the generated item is automatically printed in the ⓐ area.
- If the format of the tag name is wrong, add a new item..
- For the format, refer to [Appendix # 2].



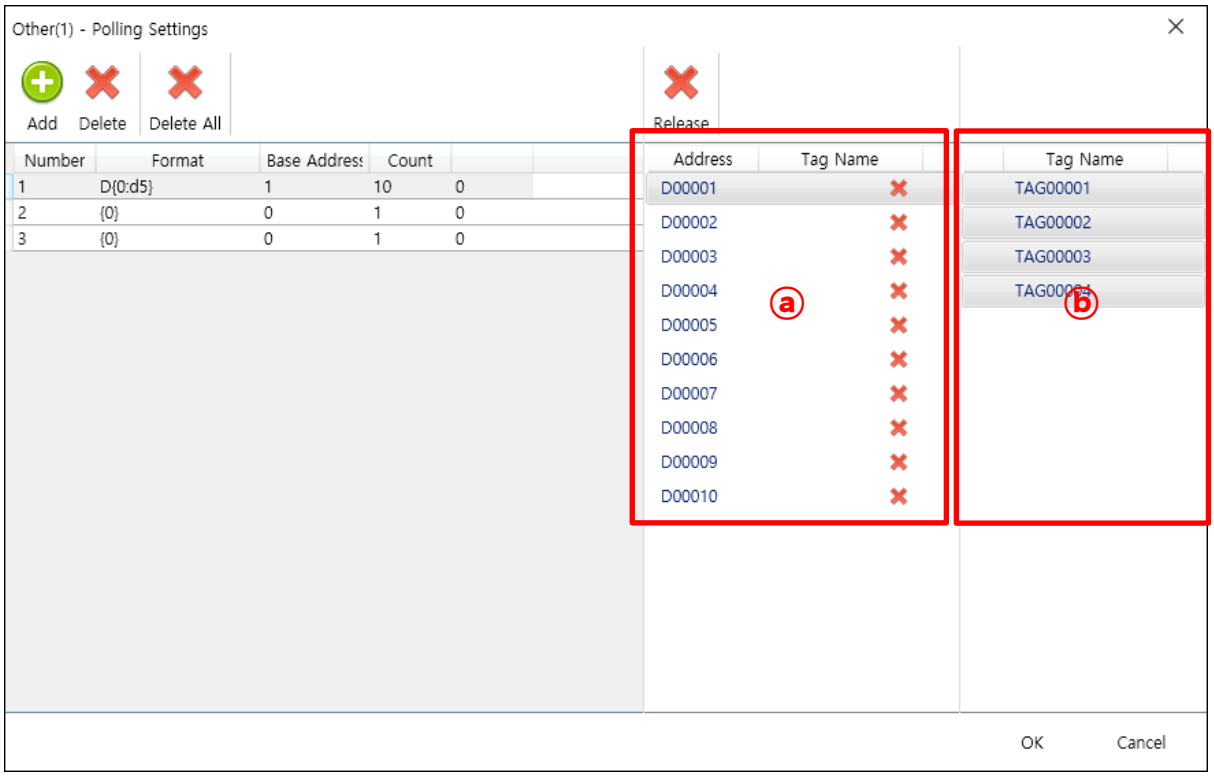
[Caution]

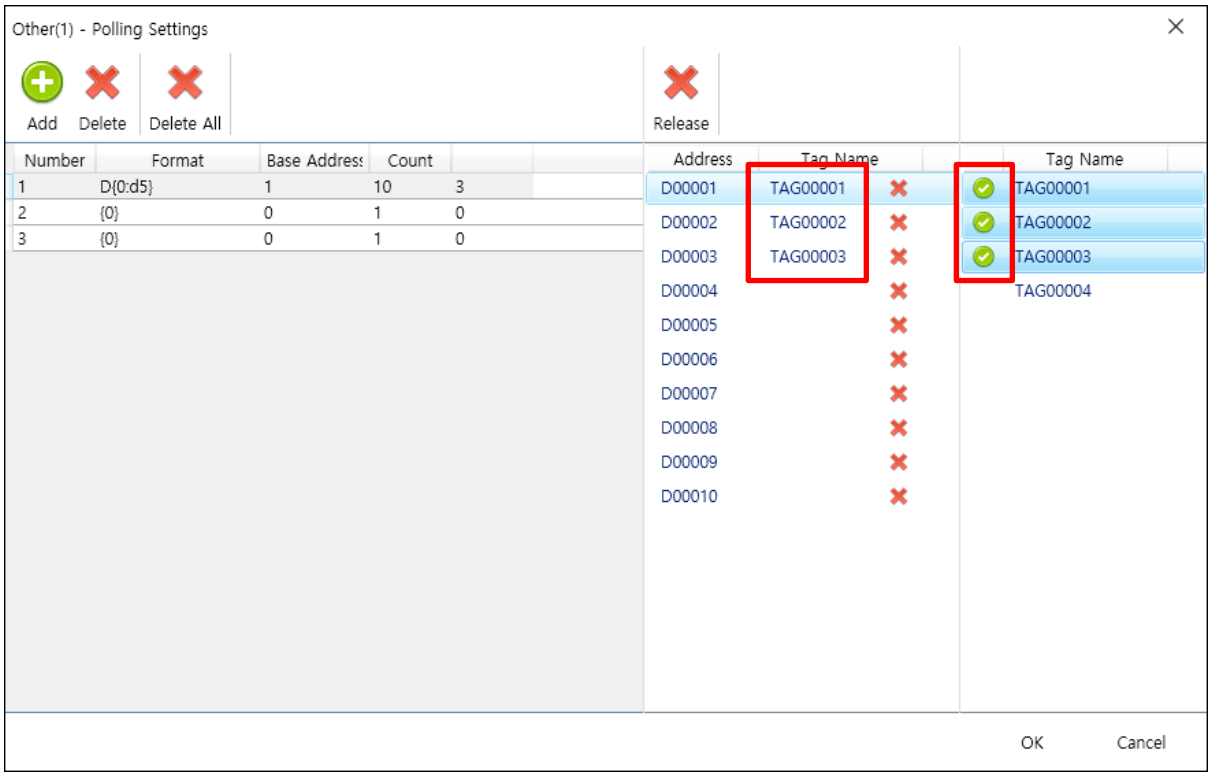
The name in the address field in the ⓐ area (for example, D00001) is set to the name that the user(program developer) uses in the C# code. This [interface address name] can be the same as the [tag name]. I made it different so that [tag name] and [interface address name] are not confused.

4. Drag and drop the tag from ⓑ to ⓐ to match the N-Server and the interface driver.

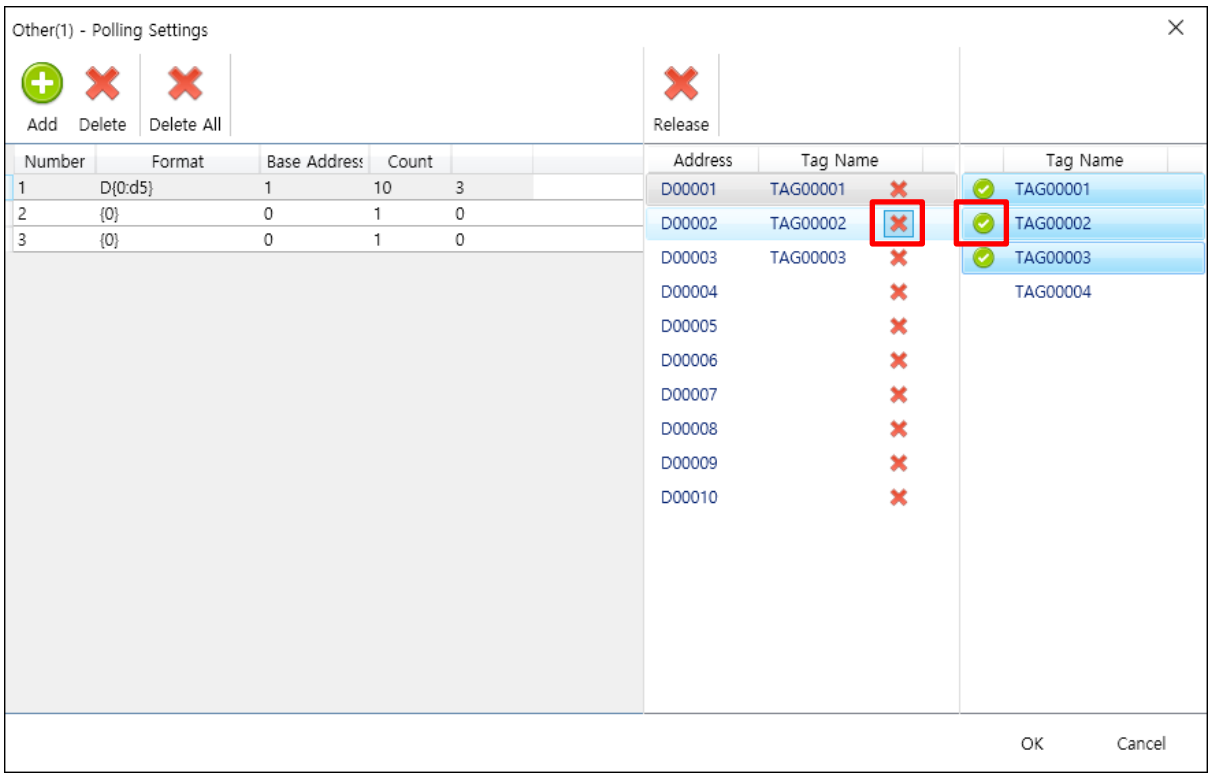
- The [address] field of ⓐ is the tag name to be used by the interface driver.

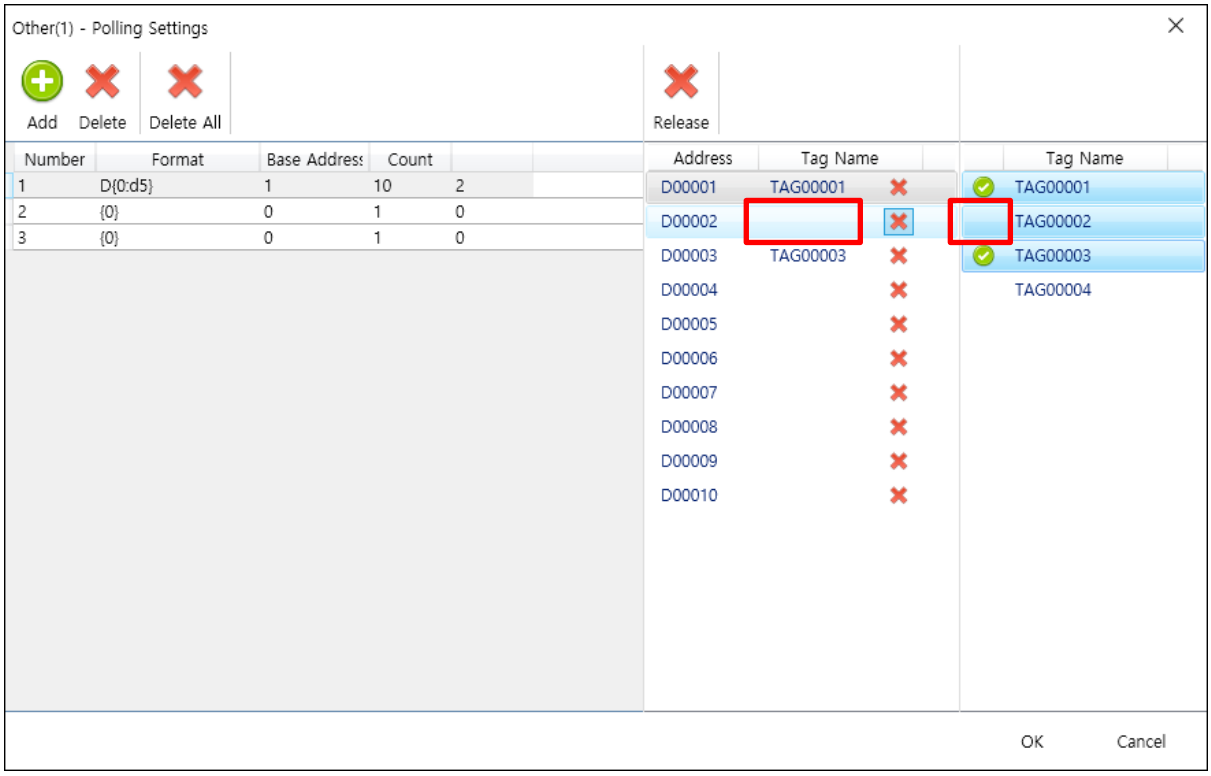
- The tags displayed in the ⓑ area are tags registered in the N-Server.



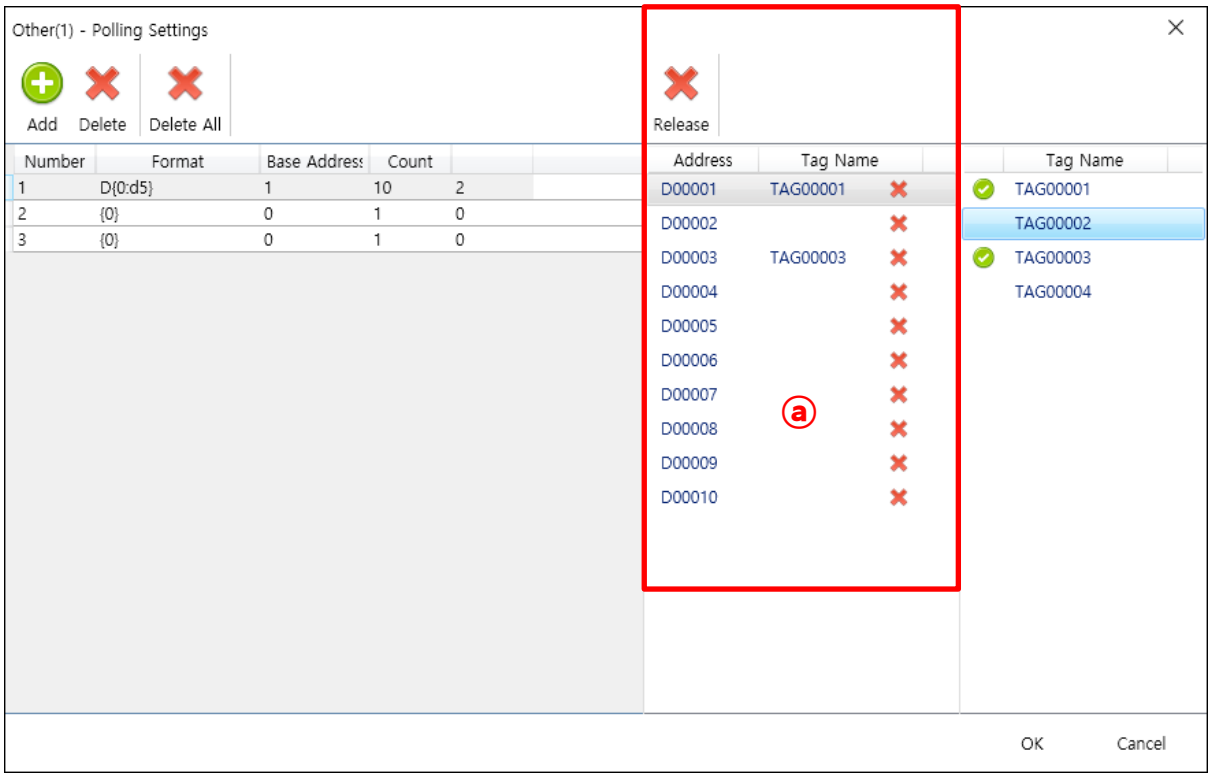


5. If you want to release the binding, you can cancel it by clicking the [X] icon of the item.





6. If you want to release all bindings, click the [Release] button at the top of the ⓐ area.



7. When you have completed all the tasks, click the [OK] button to finish the task.

4.1.3 Update Information

1. Publish the project and deliver the changed information to N-Server.

4.2 Interface practice (using example program)

4.2.1 N-Server connect

1. Run the sample program.

2. Specify the module name and system (module) number.

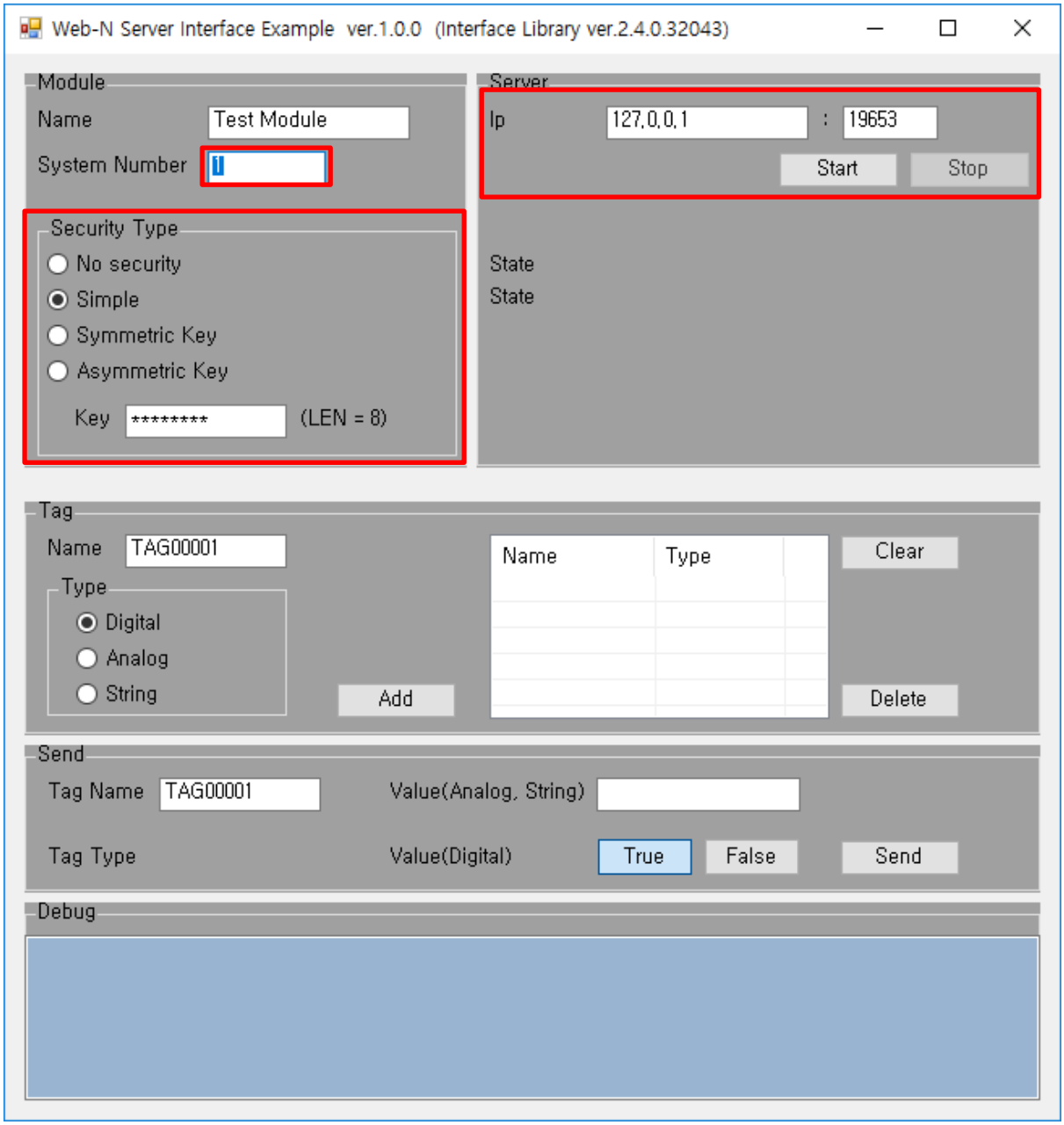
- The system (module) number must be the same as the system (module) number of the IO driver registered in N-Designer.

3. If you want to use secure communication, specify the corresponding type in [Security Type].

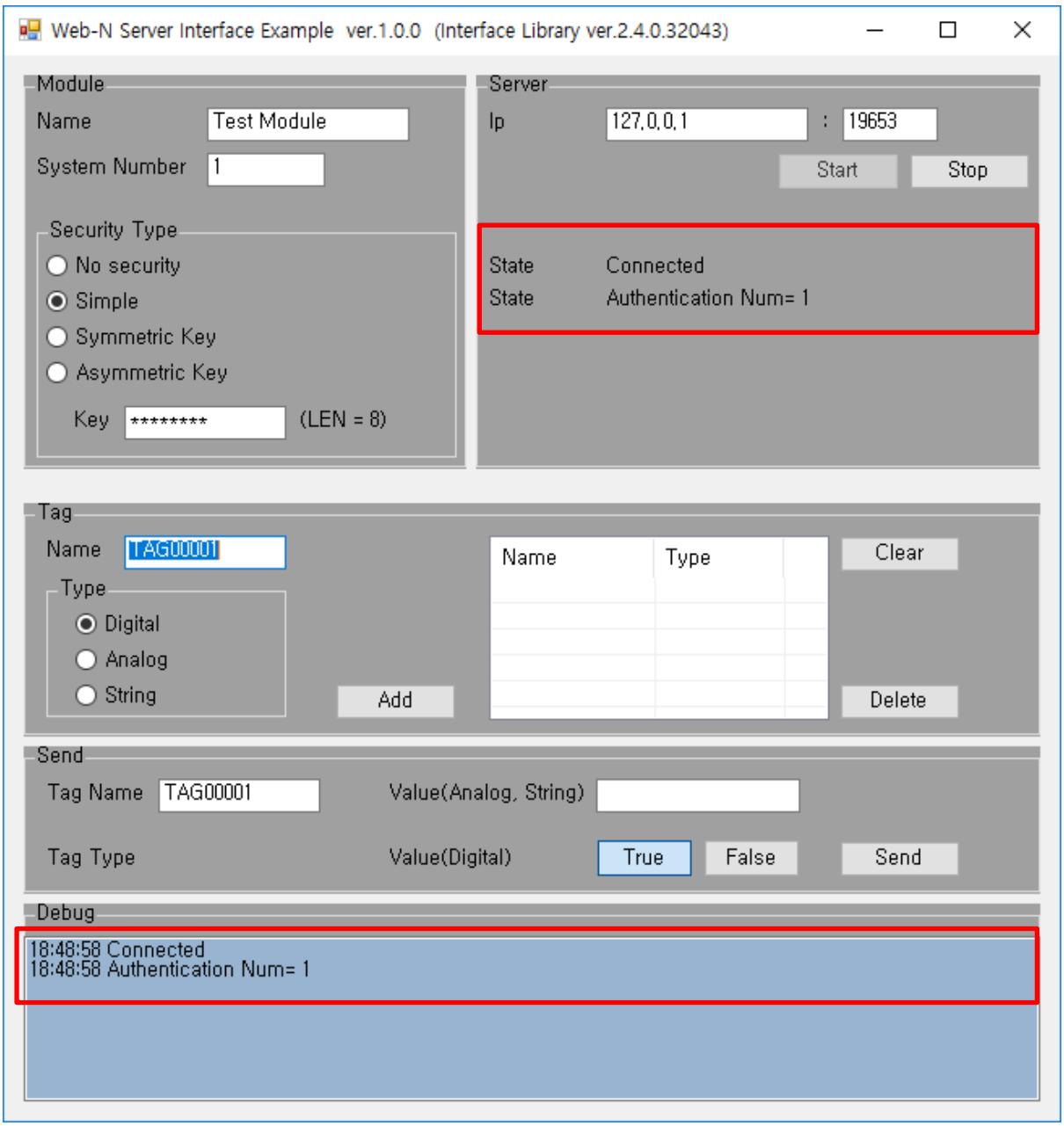
- To use security, you also need to set it on the N-Server side. Refer to [Appendix # 1] for detailed setting method.
- The length of the key [Symmetric Key / Asymmetric Key] must be 8 characters and must be a combination of alphabetic and numeric characters.

4. Enter the IP address and port of the N-Server and click the [Start] button.

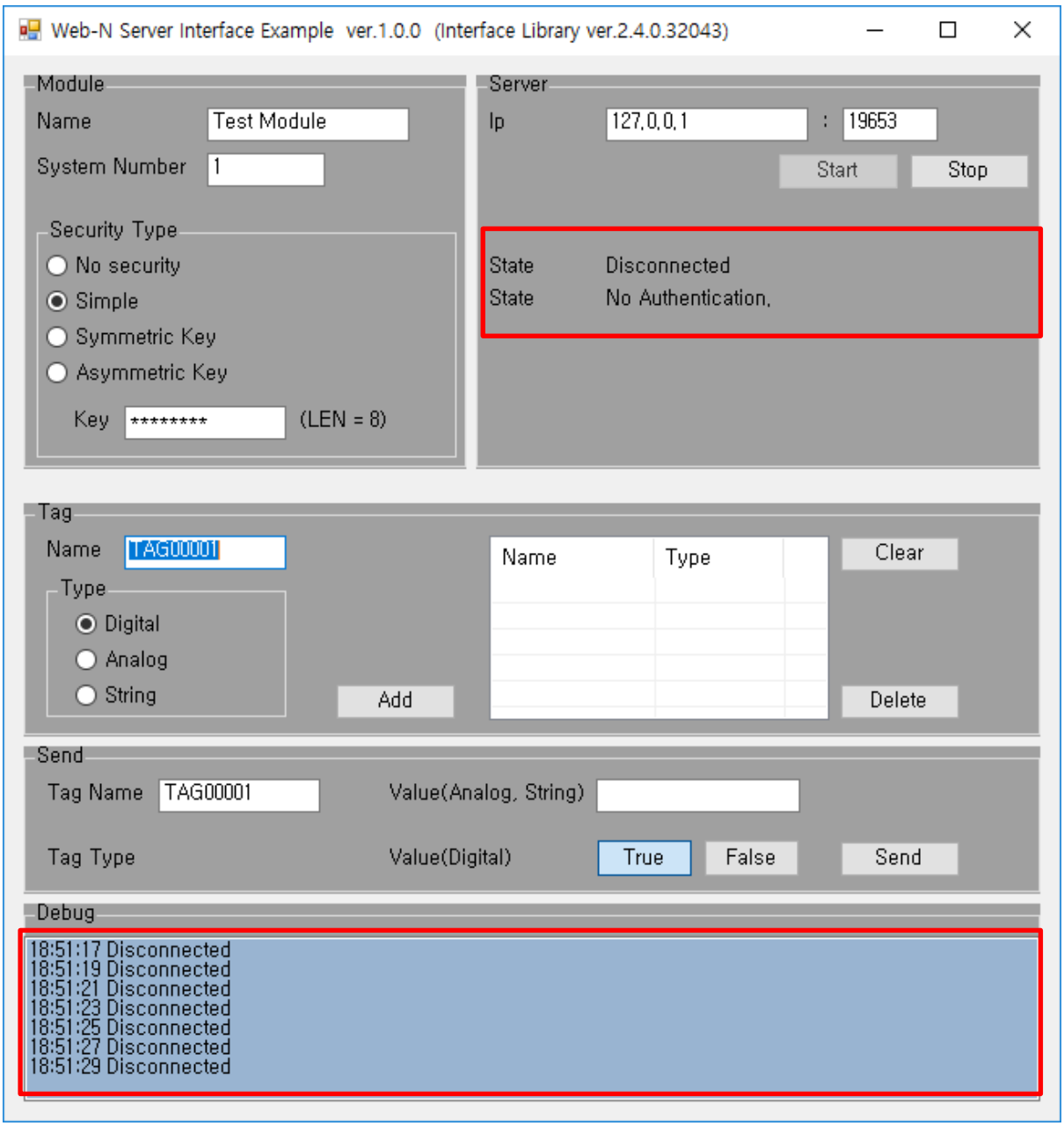
- The server port is fixed (19653).


Web-N Server Interface Example ver.1.0.0 (Interface Library ver.2.4.0.32043)
Module
Name
Test Module
System Number
1
Server
Ip
127.0.0.1
:
19653
Start
Stop
Security Type
No security
Simple
Symmetric Key
Asymmetric Key
Key
****
(LEN = 8)
State
State
Tag
Name
TAG00001
Type
Digital
Analog
String
Add
Name
Type
Clear
Delete
Send
Tag Name
TAG00001
Value(Analog, String)
Tag Type
Value(Digital)
True
False
Send
Debug

- Server connection succeeded



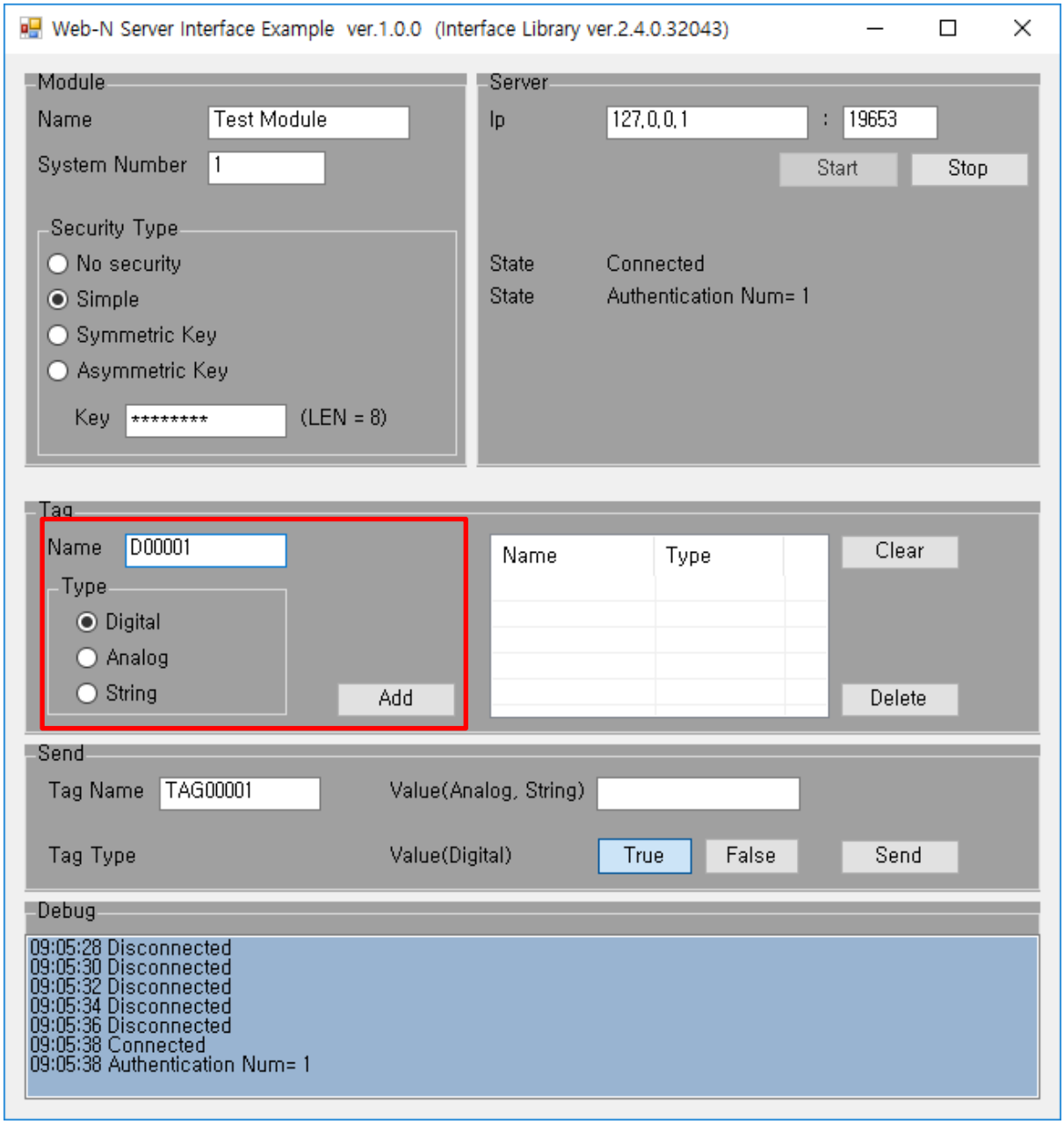
- Server connection failure and disconnect



4.2.2 Tag Registration and Release

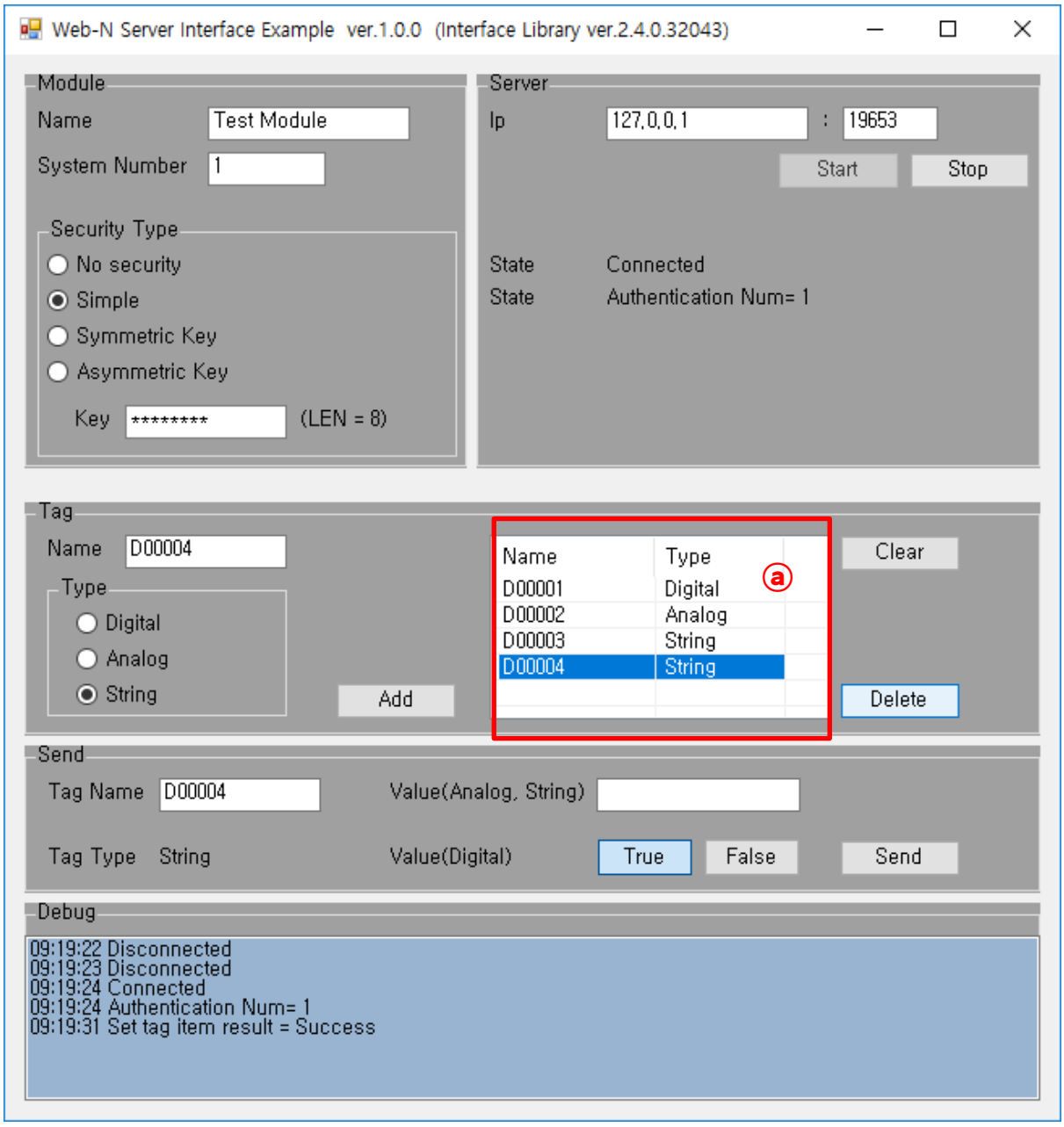
1. Specify the tag name and type and click the [Add] button to register.

- The tag name must match the tag name registered in N-Designer's IO driver.



2. Select the tag you want to delete from ⓐ and click the [Delete] button.

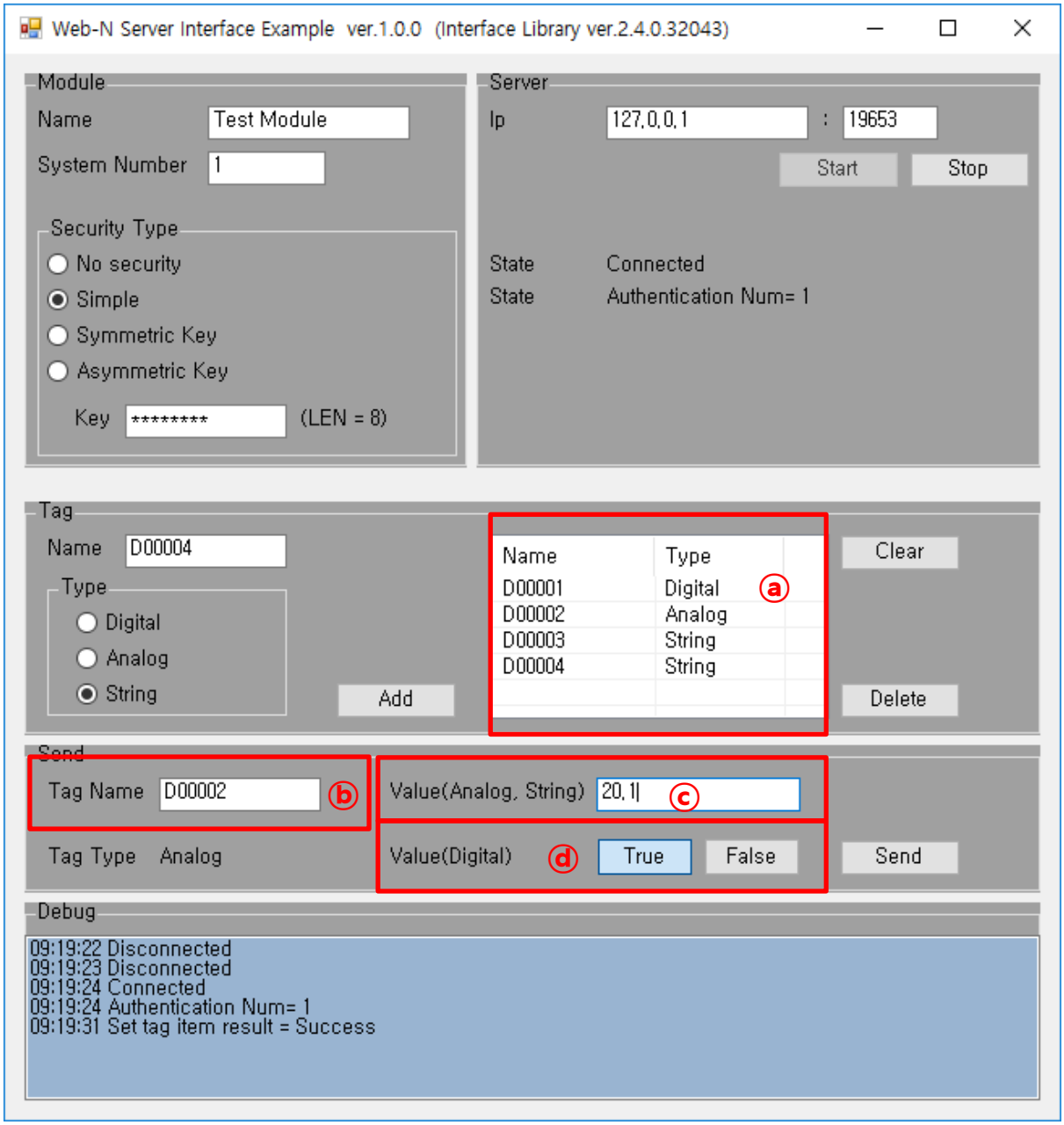
- To clear all, click the [Clear] button.



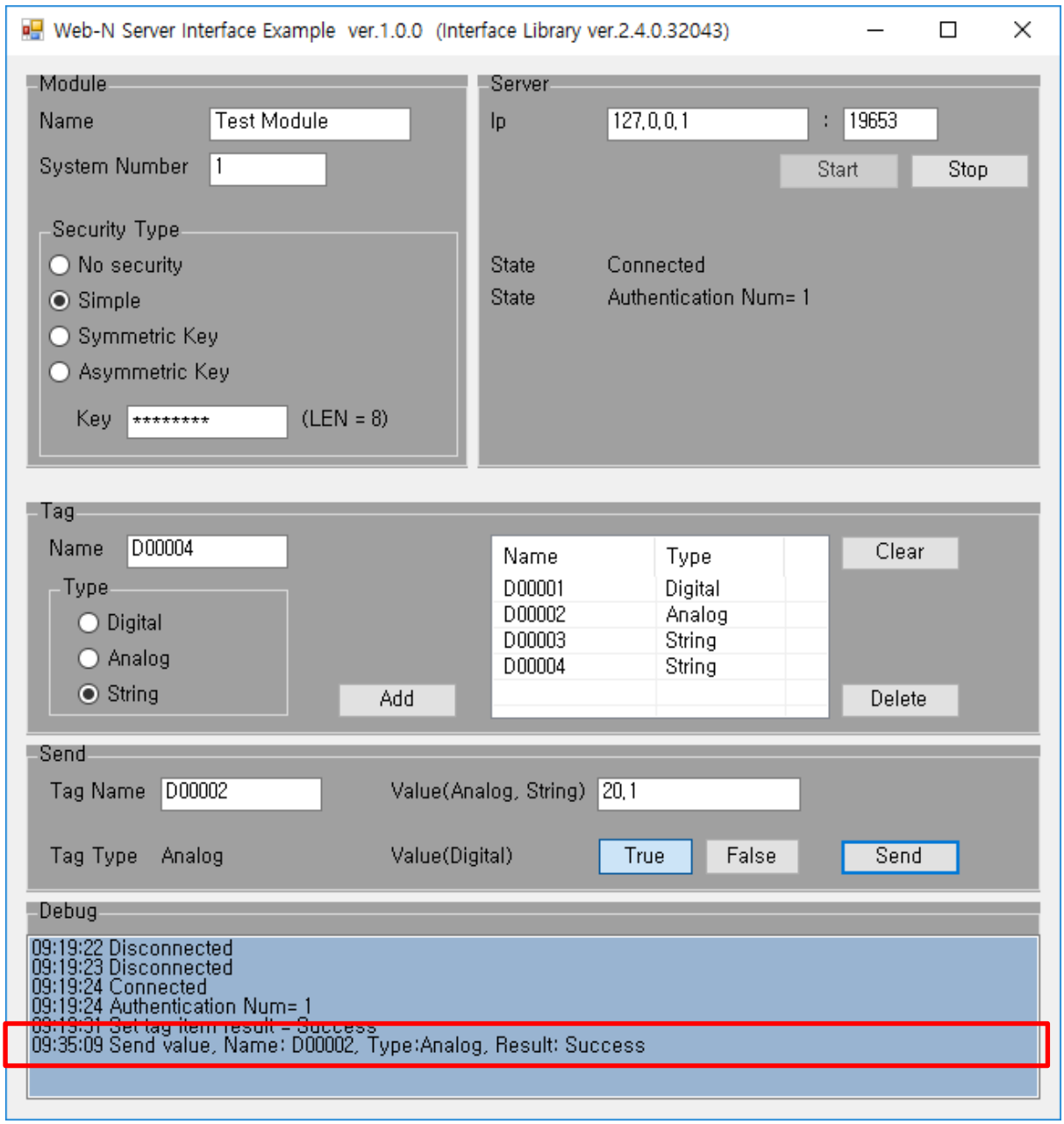
4.2.3 Sending Value to an N-Server

1. Select the tag in ListView (ⓐ), or enter the name of the tag to send the value in the textbox (ⓑ).

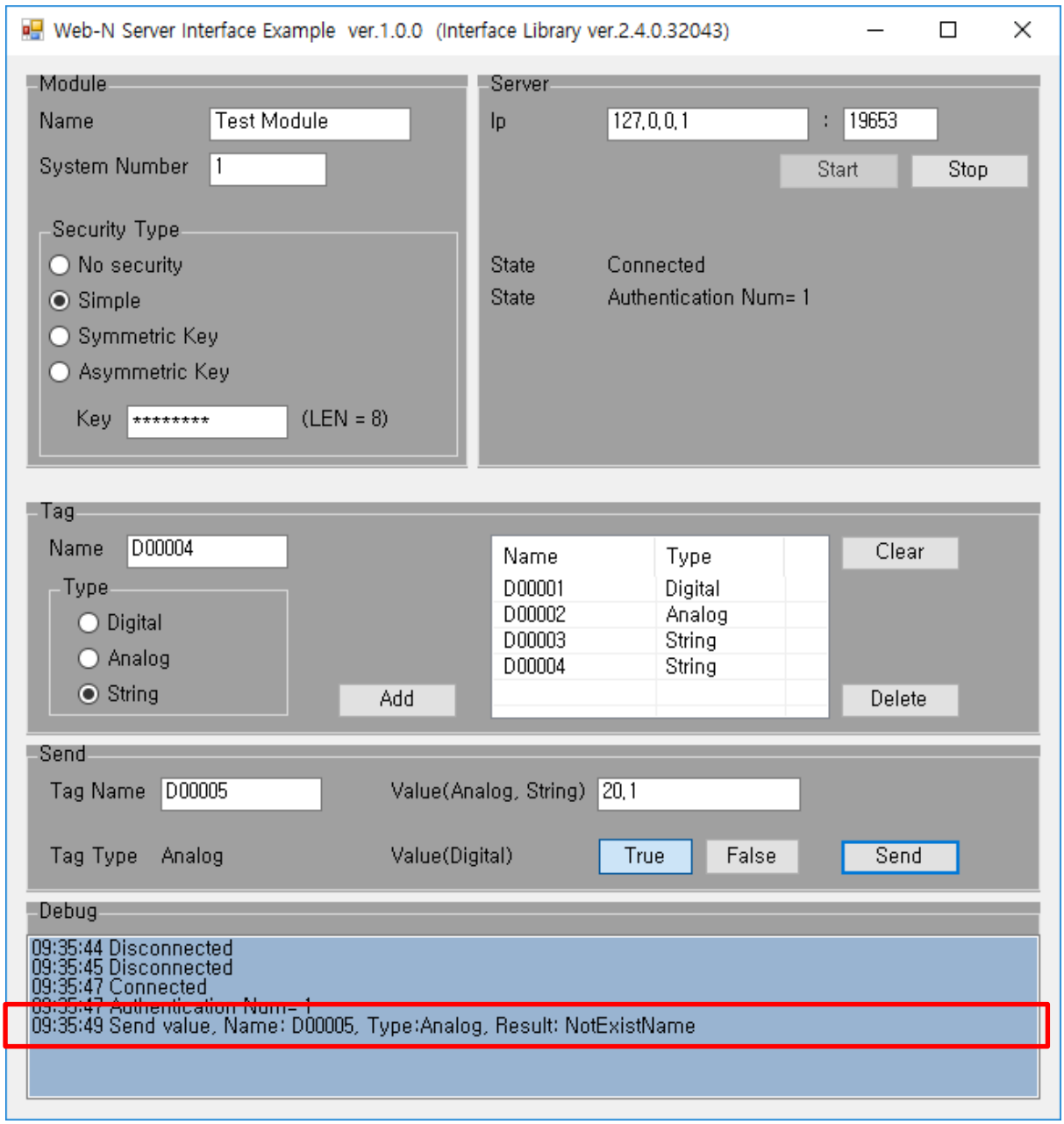
2. Input analog and string type in text box (ⓒ), digital type in two buttons (ⓓ), select the state you want to input and click [Send] button.



- Transmission successful

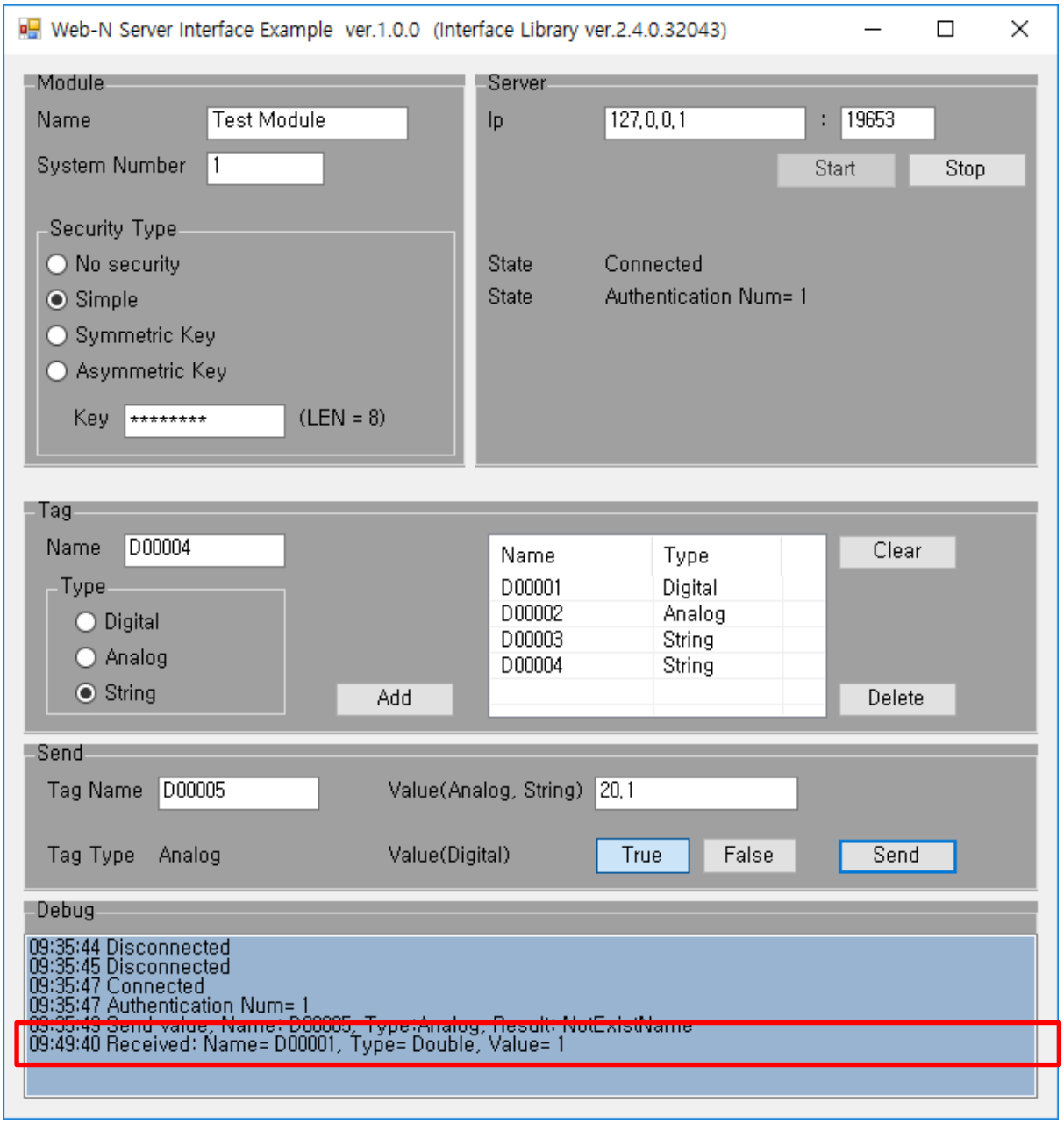


- Transmission failed (no tag name, no registration)

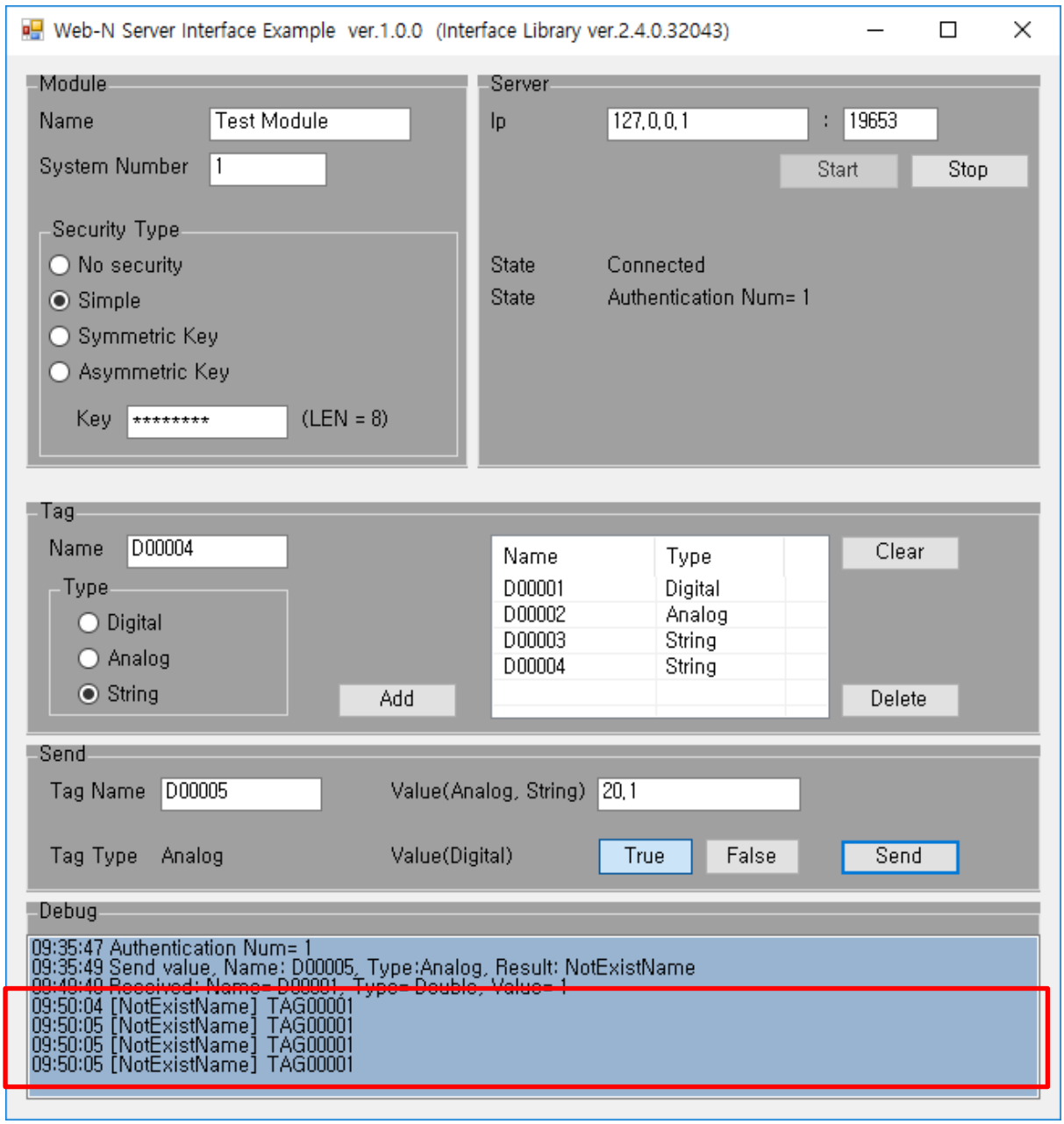


4.2.4 Receiving a value

- Data received successfully



- Error when receiving data (no tag name)

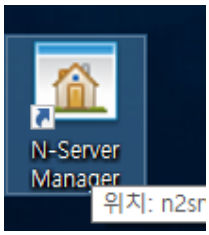


[Appendix #1]

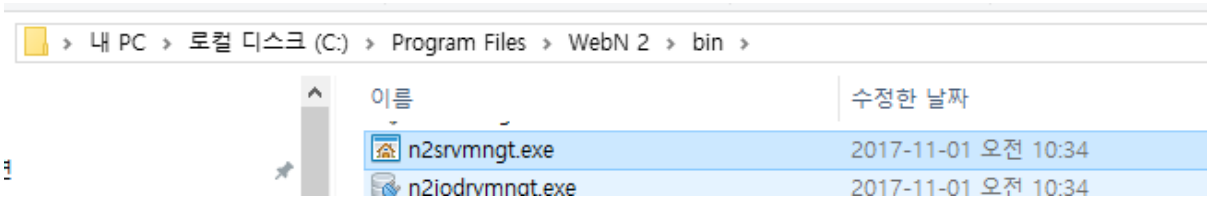
- Security settings
  - ■ Support from Web-N 2.0.6 or later

- N-Server settings

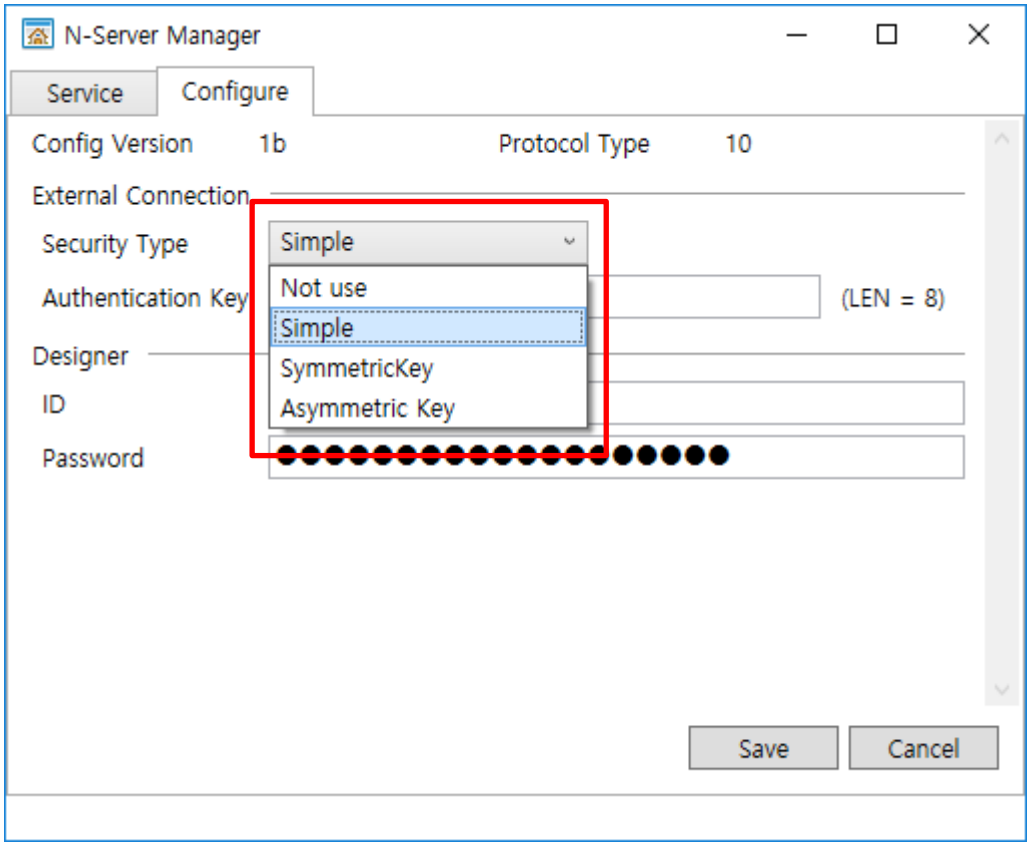
1. Run [N-Server Manager] on the desktop.



- ■ If there is no corresponding icon on the desktop, run the program from the following location.

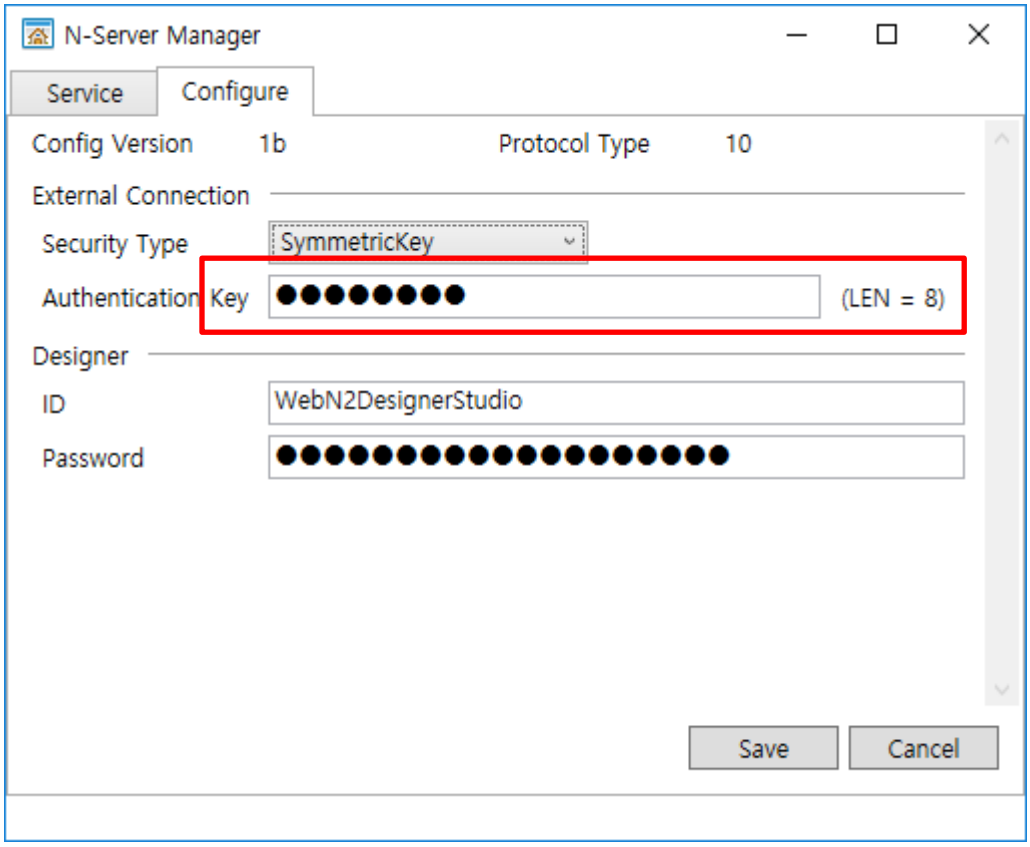


2. On the [Settings] tab, specify [External connection] - [Security type] as the type to be changed.



3. Enter the authentication key when using the [Symmetric Key / Asymmetric Key] method.

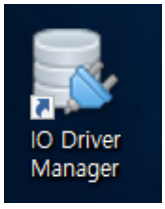
■ Key length is 8 characters and can be used in combination of alphanumeric characters.



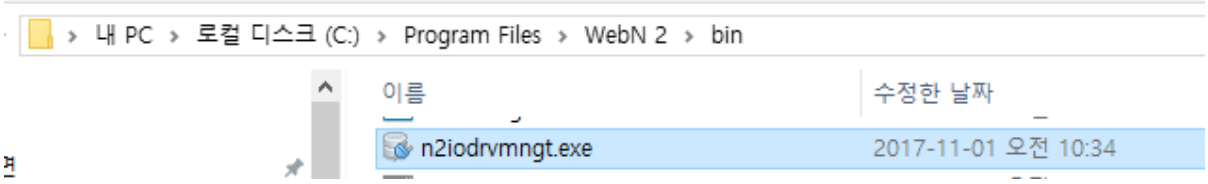
※ **The security type and authentication key must match with the interface or IO driver..**

- I/O Driver Settings

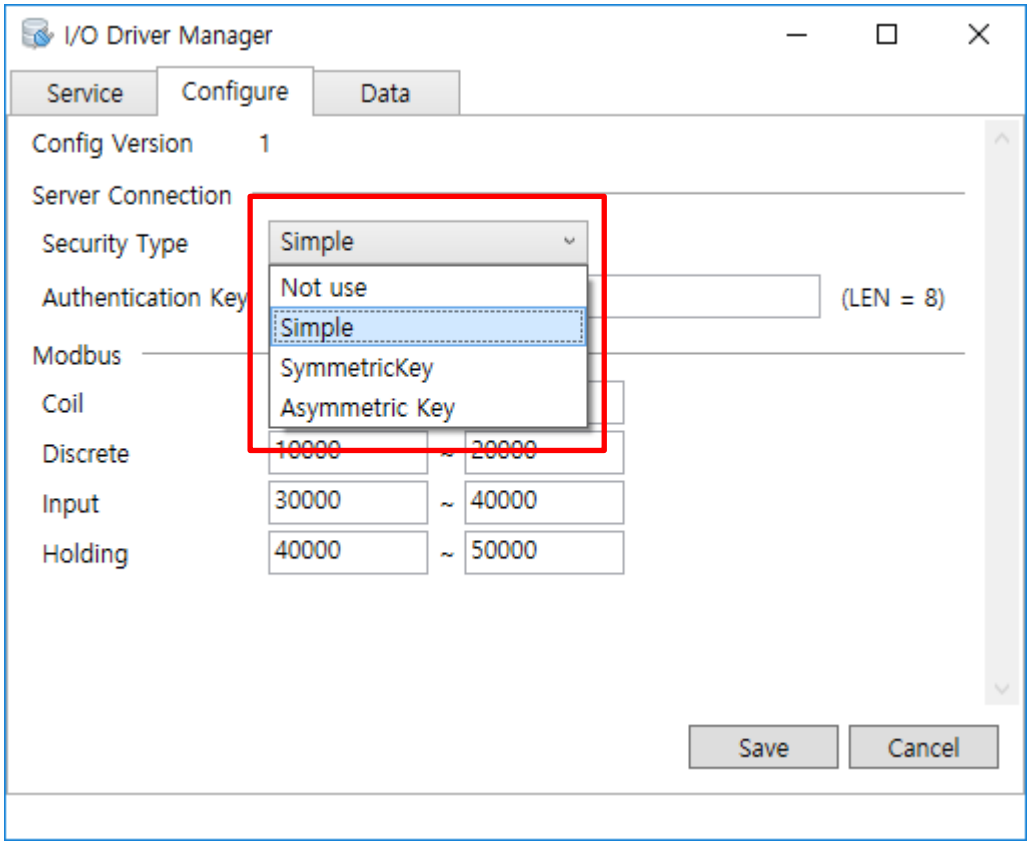
4. Run [IO Driver Manager] on the desktop.



■ If you do not have this icon on your desktop, run the program in the path directory below..

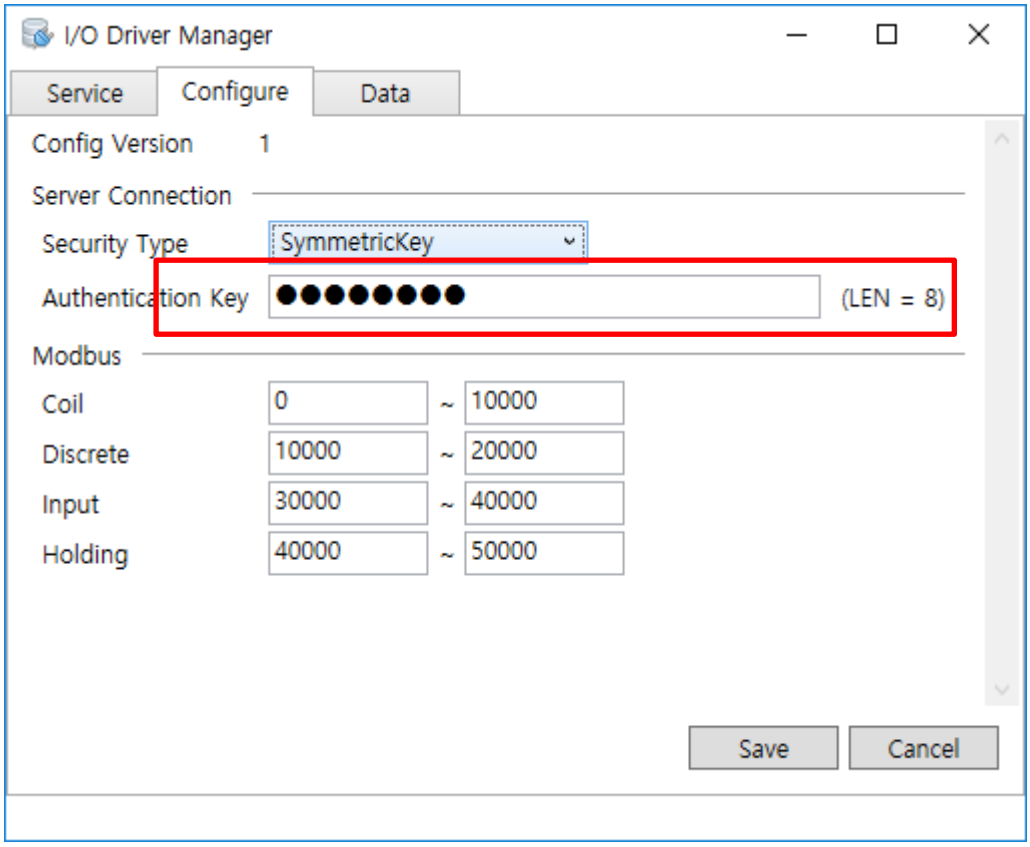


5. On the [Settings] tab, specify [Server Connection] - [Security Type] as the type to be changed.



6. Enter the authentication key when using the [Symmetric Key / Asymmetric Key] method.

■ Key length is 8 characters and can be used in combination of alphanumeric characters.



※ The security type and authentication key must match the N-Server setting.

[Appendix #2]

- IO driver map data tag name format specification method

Description or content	Format	Ex.
Prefix is TAG, number increment	TAG{0}	TAG0,TAG1,TAG2...
Prefix is TAG, numeric increment, fixed 5 digits	TAG{0:d5}	TAG00000,TAG00001
If the prefix is TAG, a numeric increment (hexadecimal representation)	TAG{0:X}	TAG0,TAG,1TAGA,TAGB
Prefix is TAG, numeric increment (hex representation), 5 digit fixed	TAG{0:X5}	TAG00000,TAG0000A