DENSO

SE1-HU-P providers

User's Guide

Version 1.0.0

June 26, 2020

NOTE:

[Revision History]

Version	Date	Description
1.0.0	2020-06-26	First edition

[Operation Check Model]

Model	Version	Notes
SE1HUP	1.00	

Contents

1. Introduction

This is a user's guide for providers who use DENSO Wave SE1HUP to read and write RFID tags. Fig. 1-1 shows the overall configuration of this provider and the device. This provider is referred to as the "SE1HUP Provider" in this document.

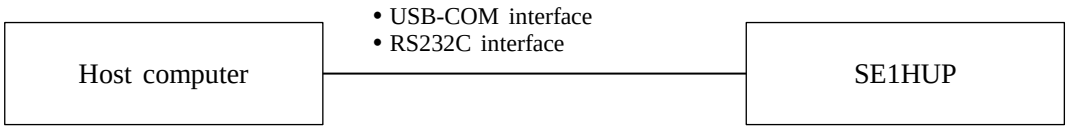


Fig. 1-1 Configuration Diagram

Fig. 1-2 shows the action between this provider and each device.
(※An example. It does not represent everything.)

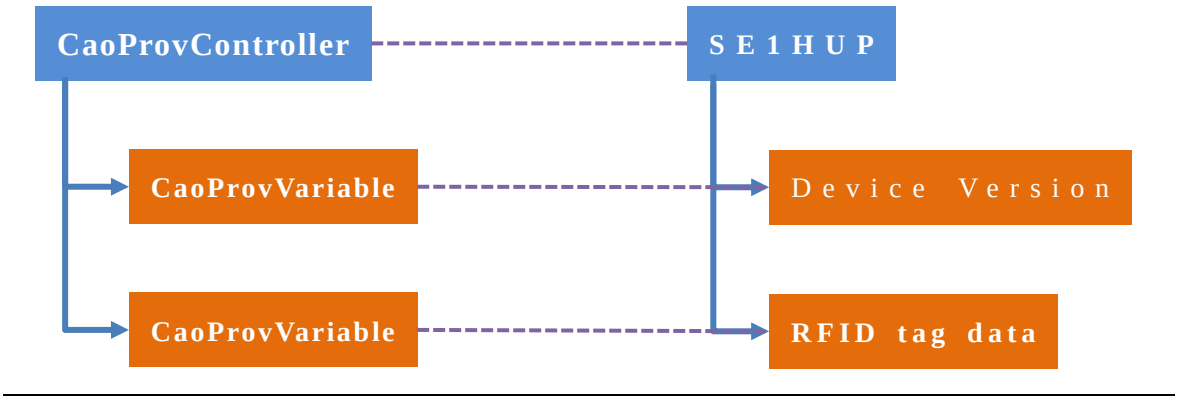


Fig. 1-2 Configuration of providers and their correspondence with SE1HUP

2. Setting Up Your Environment for Application Development

2.1. Connecting SE1HUP to Client-PC

A virtual COM connection with USB connection is required to connect SE1HUP to the client PC. To make virtual COM connections, Active USB−COM driver must be placed on the client-PC from DENSO WAVE's download page. For installation instructions, refer to Active USB-COM driver that came with the product when it was downloaded.

2.2. Setting up a PC development environment

2.2.1. Installing SE1HUP Providers Manually

If you install SE1HUP providers manually, you must register the registry as shown below. To register the registry, start the command prompt with administrator privileges and execute regsvr32 command. When executing the command, either move to the path where the file is located or specify the file path.

Table 2-1 Tabular SE1HUP providers

File name	CaoProvDENSOSE1HUP.dll
ProgID	CaoProv.DENSO.SE1HUP
Registry registration	Regsvr32 CaoProvDENSOSE1HUP.dll
Deletion of Registry Registration	Regsvr32 /u CaoProvDENSOSE1HUP.dll

3. Command Reference

3.1. Method/Property List

Table 3-1 List of methods and properties

Category	Methods/Properties[1]		Function	See Also
CaoWorkspace				
	AddController	M	Connected to controller	P. 9
CaoController				
	VariableNames	M	Get a list of variable names that can be connected	P. 10
	Variables	P	Retrieving Variable Collections Holded by the Controller	P. 11
	AddVariable	M	Adding Variable Objects	P. 11
	Execute	M	Execute Extended Commands	P. 11
	OnMessage	E	Message reception event	P. 19
CaoVariable				
	Value	P	Acquisition/set value	P. 19

3.2. Method properties

3.2.1. CaoWorkspace classes

3.2.1.1. AddController method

In CaoWorkspace, add a controller object. SE1HUP providers connect to the appropriate SE1HUP by referring to the parameters passed when AddController method is executed. The following are the specifics of AddController method:

SYNOPSIS

AddController

```
(
        "<controller name>",            // Controller name (optional)
        "CaoProv.DENSO.SE1HUP",                 // Provider name (fixed)
        "<machine name>",               // Provider execution machine name (unused)
        "<Option>"                      // Option character string (optional)
)
```

Option

[1] M: Indicates methods, P: properties, and E: events, respectively.

The following is an optional specification for Option character string: Option character string is a comma (,) string consisting of the options listed below.

Option	Required	Description	Value Range
Conn=	✓	Specifies the number of the COM-port connected to SE1HUP.	COM
Timeout=	--	Specify the time-out period from command sending to data reception from SE1HUP. Default: 2000	1-30000

Usage example

```
// Engine
ORiN2.ManagedCAO.CCaoEngine engine = new ORiN2.ManagedCAO.CCaoEngine();
// Workspace
ORiN2.ManagedCAO.CCaoWorkspace workspace = engine.AddWorkspace("NewWrks", "");
// Controller
ORiN2.ManagedCAO.CCaoController controller =
                                        workspace.AddController("SE1HUP","CaoProv.DENSO.SE1HUP
                                        ","","Conn = COM:1,Timeout = 5000");
```

3.2.1.1.1. Conn Optional

The following is a Conn optional connection parameter string: Here, brackets ("[]") are optional, and the underlined part in the description of each parameter indicates the default value when no options are specified.

RS232C

"Conn=COM:<COM Port>[:BaudRate>[:<Parity>:<DataBits>:<StopBits>[:Flow]]]"

<COM Port> : COM port number '1' -COM1, '2' - COM2, ...

<BaudRate> : Baud rate. 4800, 9600, 19200, 38400, 57600, 115200

<Parity> : Parity 'N'-NONE, 'E'-EVEN, 'O'-ODD

<DataBits> : Number of data bits '7'-7bit, '8'-8bit

<StopBits> : No. of Stop Bits '1'-1bit, '2'-2bit

<Flow> : Flow control. '0'-None, '1'-Xon/Xoff, '2'-Hardware control It can be specified by taking OR.

3.2.2. CaoController classes

3.2.2.1. VariableNames method

Gets a list of variable names that can be connected. The variable name obtained by this method can be used as the first argument of AddVariable method described later.AddVariable method

Usage example

```
// Get variable name list
String[] variableNames = controller.GetVariableNames("");
```

3.2.2.2. Variables Properties

Gets a collection of variables that the controller holds.

Usage example

```
// Variable Collection Retrieval
ORiN2.ManagedCAO.CCaoVariables variables = controller.Variables;

// Variable acquisition
ORiN2.ManagedCAO.CCaoVariable variable = variables[0];
```

3.2.2.3. AddVariable method

Adds a variable object to CaoController. Only the variable names shown in 3.3.1 can be used.

AddVariable is specified as follows.

SYNOPSIS

AddVariable

```
(
        "<variable name>",          // Variable name
        "<Option>"                  // Option character string (optional)
)
```

3.2.2.4. Execute method

Execute CaoController extension. Execute is specified as follows.

SYNOPSIS

Execute

```
(
        "<extension command name>",             // Extended command name
        "<Option string>"           // Option character string (optional)
)
```

The following is a list of extended commands that can be specified in Execute. The usage examples are described in detail in the extended commands.

Command	Description	See Also
SetRFIDReadTriggerWithBinary	Sets RFID read mode to triggered by a binary specification.	P.12
SetRFIDReadTrigerWithChar	Char setting sets RFID read mode to trigger.	P.13
ReadRFIDWithBinary	Reads RFID tags with binary specifications.	P.13
ReadRFIDWithChar	Reads RFID tags by Char specification.	P.14

Command	Description	See Also
SetRFIDWriteTriggerWithBinary	Sets RFID write mode to triggered by binary specification.	P.15
SetRFIDWriteTrigerWithChar	Set RFID write mode to triggered by Char specify.	P.15
WriteRFIDWithBinary	Writes to RFID tags with binary specification.	P.16
WriteRFIDWithChar	Write to RFID with Char specify.	P.17
SetUIDReadTriger	Sets the UID read mode to trigger.	P.17
ReadUID	Load the UID.	P.18
ResetTrigger	Resets the trigger mode settings.	P.18
GetDeviceVersion	Gets the device version.	P.18

3.2.2.4.1. SetRFIDReadTriggerWithBinary Commands

Sets the binary-specified read command for SE1HUP to trigger switch mode. The set command is executed by pressing the trigger switch, and the execution result is notified as a OnMessage event. For details about the events, see SetRFIDReadTriggerWithBinary executions.

Item	Type	Description	
Argument	VT_ARRAY | VT_VARIANT		
	0	VT_UI2	Specifies the starting position of the address to be read. Specification range: 1 to 999 (1k byte tag)[2] 0 to 1999 (2k byte-tag)
	1	VT_UI2	Specifies the length (Byte) of the data to be read. Specified [2]:1 to 400[2]
	2	VT_BSTR	Specifies the UID information as 16 hexadecimal characters. (can be omitted) Range: "0000000000000000" to "FFFFFFFFFFFFFFFF" By default, "0000000000000000" is set.
Return Value	None		

Usage example

```
// Execute SetRFIDReadTriggerWithBinary
Object[] opt = new object[] {0, 4, "0000000000000000"};

Controller.Execute("SetRFIDReadTriggerWithBinary", opt);
```

[2] The range that can be specified depends on the target RFID.

3.2.2.4.2. SetRFIDReadTriggerWithChar Commands

Sets ASCII character string specification read command to trigger switch mode for SE1HUP. The set command is executed by pressing the trigger switch, and the execution result is notified as a OnMessage event. For details about the events, see SetRFIDReadTriggerWithChar .

Item	Type Description		
Argument	VT_ARRAY | VT_VARIANT		
	0	VT_UI2	Specifies the starting position of the address to be read. Specified [2]:1-999 (1k Byte Tags)[2] 0 to 1999 (2k byte-tag)
	1	VT_UI2	Specifies the length (Byte) of the data to be read. Specified [2]:1 to 400[2]
	2	VT_BSTR	Specifies the UID information as a hexadecimal string of 16 characters. (can be omitted) Range: "0000000000000000" to "FFFFFFFFFFFFFFFF" By default, "0000000000000000" is set.
Return Value	None		

Usage example

```
// Execute SetRFIDReadTriggerWithChar
Object[] opt = new object[] {0, 4, "0000000000000000"};

Controller.Execute("SetRFIDReadTriggerWithChar", opt);
```

3.2.2.4.3. ReadRFIDWithBinary Commands

Retrieves the specified data length (Byte) from the specified address position in binary.

Item	Type Description		
Argument	VT_ARRAY | VT_VARIANT		
	0	VT_UI2	Specifies the starting position of the address to be read. Specified [2]:1-999 (1k Byte Tags)[2] 0 to 1999 (2k byte-tag)
	1	VT_UI2	Specifies the length (Byte) of the data to be read. Specified [2]:1 to 400[2]

Item	Type Description		
	2	VT_BOOL	Specifies whether to return by VTARRAY when the length of the acquired data is 1. (can be omitted) TRUE: When the data length is 1, the return type is returned as (VT_UI1 | VTARRAY). FALSE: When the data length is 1, the return type is returned as (VT_UI1). By default, "FALSE" is set.
	3	VT_BSTR	Specifies the UID information as a hexadecimal string of 16 characters. (can be omitted) Range: "0000000000000000" to "FFFFFFFFFFFFFFFF" By default, "0000000000000000" is set.
Return Value	VT_ARRAY | VT_UI		
	N	VT_UI1	Gets the specified address data in binary.

Usage example

```
// Execute ReadRFIDWithBinary
Object[] opt = new object[] {0, 4, "0000000000000000"};

BYTE[] byte = controller.Execute("ReadRFIDWithBinary", opt) as byte[];
```

3.2.2.4.4. ReadRFIDWithChar Commands

Retrieves the specified data length (Byte) from the specified address position as a ASCII text string.

Item	Type Description		
Argument	VT_ARRAY | VT_VARIANT		
	0	VT_UI2	Specifies the starting position of the address to be read. Specified [2]:1-999 (1k Byte Tags)[2] 0 to 1999 (2k byte-tag)
	1	VT_UI2	Specifies the length (Byte) of the data to be read. Specified [2]:1 to 400[2]
	3	VT_BSTR	Specifies the UID information as a hexadecimal string of 16 characters. (can be omitted) Range: "0000000000000000" to "FFFFFFFFFFFFFFFF" By default, "0000000000000000" is set.
Return Value	VT_BSTR		Retrieves the specified address data as a ASCII string.

Usage example

```
//Execute ReadRFIDWithChar
```

```
Object[] opt = new object[] {0, 4, "0000000000000000"};

String strData = controller.Execute("ReadRFIDWithChar", opt) as string;
```

3.2.2.4.5. SetRFIDWriteTriggerWithBinary Commands

Sets the binary-specified write command for SE1HUP to trigger switch mode. The set command is executed by pressing the trigger switch, and the execution result is notified as a OnMessage event. For details about the events, see SetRFIDWriteTriggerWithBinary executions.

Item	Type Description		
Argument	VT_ARRAY | VT_VARIANT		
	0	VT_UI2	Specifies the start position of the address to be written. Specified [2]:1-999 (1k Byte Tags)[2] 0 to 1999 (2k byte-tag)
	1	VT_UI2	Specifies the length (Byte) of the data to be read. Specified [2]:1 to 400[2]
	2	VT_ARRAY | VT_UI1	
		2. N VT_UI1	Specify the data to be written in binary.
	3	VT_BSTR	Specifies the UID information as a hexadecimal string of 16 characters. (can be omitted) Range: "0000000000000000" to "FFFFFFFFFFFFFFFF" By default, "0000000000000000" is set.
Return Value	None		

Usage example

```
// Execute SetRFIDWriteTriggerWithBinary
Byte[] writeData = new byte[] {48, 49, 50, 51}; // Create data to write
Object[] opt = new object[] {0, 4, writeData, "0000000000000000"};

Controller.Execute("SetRFIDWriteTriggerWithBinary", opt);
```

3.2.2.4.6. SetRFIDWriteTriggerWithChar Commands

Sets the binary-specified write command for SE1HUP to trigger switch mode. The set command is executed by pressing the trigger switch, and the execution result is notified as a OnMessage event. For details about the events, see SetRFIDWriteTriggerWithChar executions.

Item	Type Description
Argument	VT_ARRAY | VT_VARIANT

Item	Type Description		
	0	VT_UI2	Specifies the start position of the address to be written. Specified [2]:1-999 (1k Byte Tags)[2] 0 to 1999 (2k byte-tag)
	1	VT_UI2	Specifies the length (Byte) of the data to be read. Specified [2]:1 to 400[2]
	2	VT_BSTR	Specify the data to be written as a ASCII character string.
	3	VT_BSTR	Specifies the UID information as a hexadecimal string of 16 characters. (can be omitted) Range: "0000000000000000" to "FFFFFFFFFFFFFFFF" By default, "0000000000000000" is set.
Return Value	None		

Usage example

```
// Execute SetRFIDWriteTriggerWithChar
String writeData = "ABCD"; // Make data to write
Object[] opt = new object[] {0, 4, writeData, "0000000000000000"};

Controller.Execute("SetRFIDWriteTriggerWithChar", opt);
```

3.2.2.4.7. WriteRFIDWithBinary Commands

Writes data of the specified data length (Byte) from the specified address position in binary.

Item	Type Description		
Argument	VT_ARRAY | VT_VARIANT		
	0	VT_UI2	Specifies the start position of the address to be written. Specified [2]:1-999 (1k Byte Tags)[2] 0 to 1999 (2k byte-tag)
	1	VT_UI2	Specifies the length (Byte) of the data to be read. Specified [2]:1 to 400[2]
	2	VT_ARRAY | VT_UI1	
		2. N VT_UI1	Specify the data to be written in binary.
	3	VT_BSTR	Specifies the UID information as a hexadecimal string of 16 characters. (can be omitted) Range: "0000000000000000" to "FFFFFFFFFFFFFFFF" By default, "0000000000000000" is set.
Return Value	None		

Usage example

```
// Execute WriteRFIDWithBinary
Byte[] writeData = new byte[] {48, 49, 50, 51}; // Create data to write
Object[] opt = new object[] {0, 4, writeData, "0000000000000000"};

Controller.Execute("WriteRFIDWithBinary", opt);
```

3.2.2.4.8. WriteRFIDWithChar Commands

Writes data from the specified address position to RFID tag as a ASCII character string.

Item	Type Description		
Argument	VT_ARRAY | VT_VARIANT		
	0	VT_UI2	Specifies the start position of the address to be written. Specified [2]:1-999 (1k Byte Tags)[2] 0 to 1999 (2k byte-tag)
	1	VT_UI2	Specifies the length (Byte) of the data to be read. Specified [2]:1 to 400[2]
	2	VT_BSTR	Specify the data to be written as a ASCII character string.
	3	VT_BSTR	Specifies the UID information as a hexadecimal string of 16 characters. (can be omitted) Range: "0000000000000000" to "FFFFFFFFFFFFFFFF" By default, "0000000000000000" is set.
Return Value	None		

Usage example

```
// Execute WriteRFIDWithChar
String writeData = "ABCD"; // Make data to write
Object[] opt = new object[] {0, 4, writeData, "0000000000000000"};

Controller.Execute("WritRFIDWithChar", opt);
```

3.2.2.4.9. SetUIDReadTrigger Commands

Set the Read UID command for SE1HUP to trigger switch mode. The set command is executed by pressing the trigger switch, and the execution result is notified as a OnMessage event. For details about the events, see SetUIDReadTrigger executions.

Item	Type Description
Argument	None
Return Value	None

Usage example

```
// Execute SetUIDReadTrigger
Controller.Execute("SetUIDReadTrigger", opt);
```

3.2.2.4.10. ReadUID Commands

Retrieves the UIDs of RFID tags within a range.

Item	Type	Description
Argument	None	
Return Value	VT_BSTR | ARRAY or VT_BSTR	
	N	Get UIDs for RFID tagging.

Usage example

```
// Execute ReadUID
Object objUID = controller.Execute("ReadUID", opt);
// Check if the acquired data is an array
If(objUID.GetType().IsArray)
{
      Convert to array of string[] strDatas = objUID as string[]; //string
}
Else
{
       Convert to String strData = objUID.ToString(); //string
}
```

3.2.2.4.11. ResetTrigger Commands

When SE1HUP is set to the trigger switch mode, the trigger switch mode is canceled.

Item	Type Description
Argument	None
Return Value	None

Usage example

```
// Execute ResetTrigger
Controller.Execute("SetUIDReadTrigger" ,null) ; // Set trigger switch mode

Controller.Execute("ResetTrigger" ,null) ; // Cancel the set trigger switch mode
```

3.2.2.4.12. GetDeviceVersion Commands

Get the device version of SE1HUP.

Item	Type	Description
Argument	None	
Return Value	VT_BSTR	Gets the device version.

Usage example

```
' Execute GetDeviceVersion
String strDeviceVersion = controller.Execute("GetDeviceVersion", null) as string;
```

3.2.2.5. OnMessage event

Reception in trigger switch mode can be received as a OnMessage event. See 3.4 for the events that can be received.

Usage example

```
// Registers a message event with the controller.
Private void AddOnMessage()
{
      Controller.OnMessage += new OnMesssageEventHandler(OnMessage);
}

// Message reception event body
Private void OnMessage(object obj,OnMessageEventArgs arg)
{
    // Get message content
    Object message = arg.Message.Value;
}
```

3.2.3. CaoVariable classes

3.2.3.1. Value Properties

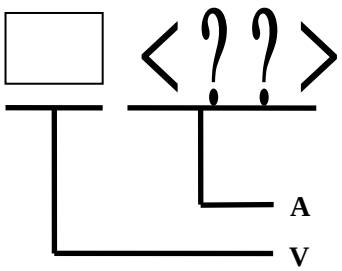
Gets/sets the data from the connected SE1HUP. The behavior depends on the variable name. For details, refer to section 3.3, Variable list.

3.3. Variable list

Defines a list of variables that can be used in each class. Variables refer to objects of CaoVariable classes. An arbitrary string can be added to register multiple variables (useful when changing only options, etc.).

The following format gives an arbitrary string to a variable name:

Multi-variable common specification format


Arbitrary character str
Variable identifi

3.3.1. CaoController class-variable

Variable name	Description	Value		See Also
		Get	Put	
@MAKER_NAME	Obtain the manufacturer's name.	✓	-	P.21
@VERSION	Get the DLL version.	✓	-	P.21
@DEVICE_VERSION	Get the device version of SE1HUP.	✓	✓	P.22
@UID	Retrieves the UID information within the range.	✓	-	P.22
RFID_ADDRESS_BINARY<??>	Sets RFID tag data to be acquired with the binary specification.	✓	✓	P.22
RFID_ADDRESS_CHAR<??>	Gets and sets RFID tag data with the character string specification.	✓	✓	P.23

3.3.1.1. @MAKER_NAME

Obtain the manufacturer's name.

Data Type

Type Description	
VT_BSTR	Obtain the manufacturer's name.

Usage example

```
// Add Variable
ORiN2.ManagedCAO.CCaoVariable var = controller.AddVariable("@MAKER_NAME","");
// Acquisition of Values
String value = var.Value.ToString();
```

3.3.1.2. @VERSION

Gets the DLL version.

Data Type

Type Description	
VT_BSTR	The DLL version is "*. *. * Get in the " format.

Usage example

```
// Add Variable
ORiN2.ManagedCAO.CCaoVariable var = controller.AddVariable("@VERSION","");
// Acquisition of Values
String value = var.Value.ToString();
```

3.3.1.3. @DEVICE_VERSION

Gets the DLL version.

Data Type

Type Description	
VT_BSTR	The device version of SE1HUP is "*. ** Obtain in the" format

Usage example

```
// Add Variable
ORiN2.ManagedCAO.CCaoVariable var = controller.AddVariable("@DEVICE_VERSION","");
// Acquisition of Values
String value = var.Value.ToString();
```

3.3.1.4. @UID

Gets the DLL version.

Data Type

Type Description	
VT_BSTR | ARRAY or VT_BSTR	
N	Get UIDs for RFID tagging.

Usage example

```
// Add Variable
ORiN2.ManagedCAO.CCaoVariable var;
Var = controller.AddVariable("@VERSION","");
// Acquisition of Values
Object value = var.Value;
```

3.3.1.5. RFID_ADDRESS_BINARY<??>

Gets/sets the data of RFID tag by binary specification. Enter an arbitrary character string after RFID_ADDRESS_BINARY to specify the variable name.

Option

Option	Required	Description	Range-of-Value [22]
StartPos=	✓	Specifies the start position of the address.	1 to 999 (1k byte-tag) 0 to 1999 (2k byte-tag)
Elem=	✓	Specifies the number of elements to read from the address start position.	1~400

UID=	--	Specifies the UID information as a hexadecimal string of 16 characters. (can be omitted) When "0000000000000000" is specified, data is acquired from any tag.	Hexadecimal string of 16 digits Default value: "0000000000000000"

Data Type

Type Description		
VT_ARRAY | VT_UI		
N	VT_UI1	Gets and sets the specified address data as a binary array.

Usage example

```
// Add Variable
ORiN2.ManagedCAO.CCaoVariable var;
Var = controller.AddVariable("RFID_ADDRESS_BINARY","StartPos=0,Elem=2");
// Acquisition of Values
Byte[] value = var.Value as byte[];
// Insert the letter A in the first address.
Byte[0] = 'A';
// Value setting
Var Value = byte;
```

3.3.1.6. RFID_ADDRESS_CHAR<??>

Gets/sets the data of RFID tag by binary specification. Enter an arbitrary character string after RFID_ADDRESS_BINARY to specify the variable name.

Option

Option	Required	Description	Range-of-Value [2]
StartPos=	✓	Specifies the start position of the address.	1 to 999 (1k byte-tag) 0 to 1999 (2k byte-tag)
Elem=	✓	Specifies the number of elements to read from the address start position.	1~400
UID=	--	Specifies the UID information as a hexadecimal string of 16 characters. (can be omitted) When "0000000000000000" is specified, data is acquired from any tag.	Hexadecimal string of 16 digits Default value: "0000000000000000"

Data Type

Type Description	
VT_BSTR	Acquisitions and sets the specified address data as a ASCII character string.

5.5

Usage example

```
// Add Variable
ORiN2.ManagedCAO.CCaoVariable var;
Var = controller.AddVariable("RFID_ADDRESS_CHAR","StartPos=0,Elem=2");
// Acquisition of Values
String data = var.Value.ToString();
// Set AB character string
Data = "AB";
// Setting Values
Var.Value = data;
```

3.4. Event list

The following is a list of outcomes during trigger switch mode that can be received by OnMessage.

Message number	Description	See Also
0	Results of executing the command set in SetRFIDReadTriggerWithBinary	P.24
1	Results of executing the command set in SetRFIDReadTriggerWithChar	P.24
2	Results of executing the command set in SetRFIDWriteTriggerWithBinary	P.25
3	Results of executing the command set in SetRFIDWriteTriggerWithChar	P.25
4	Results of executing the command set in SetUIDReadTrigger	P.25

If the analysis result of the received data from SE1HUP is an error, an error is stored in the message content and the event is notified. Please refer to 5.5 for the details of the error.

3.4.1. SetRFIDReadTriggerWithBinary executions

After executing SetRFIDReadTriggerWithBinary command, it notifies the result when the switch is pressed on SE1HUP.

Data Type

Message Description		
VT_ARRAY | VT_UI1		
N	VT_UI1	Received RFID tags are acquired in binary.

3.4.2. SetRFIDReadTriggerWithChar executions

After executing SetRFIDReadTriggerWithChar command, it notifies the result when the switch is pressed on SE1HUP.

Data Type

Message Description

VT_BSTR	Retrieves the received RFID tag data as a ASCII character string.

3.4.3. SetRFIDWriteTriggerWithBinary executions

After executing SetRFIDWriteTriggerWithBinary command, it notifies the result when the switch is pressed on SE1HUP.

Data Type

Message Description
None

3.4.4. SetRFIDWriteTriggerWithChar executions

After executing SetRFIDWriteTriggerWithChar command, it notifies the result when the switch is pressed on SE1HUP.

Data Type

Message Description
None

3.4.5. SetUIDReadTrigger executions

After executing SetUIDReadTrigger command, it notifies the result when the switch is pressed on SE1HUP.

Data Type

Message Description		
VT_ARRAY | VT_BSTR or VT_BSTR		
N	VT_BSTR	Retrieves the received UID data.

4. Sample Programming with SE1HUP Providers

With SE1HUP providers, you can connect SE1HUP to the client computer as follows:

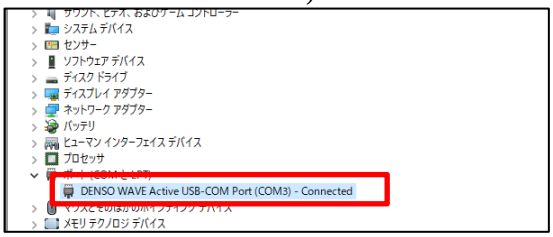
- Creating a CaoEngine
- Creating a CaoWorkspace
- Creating a CaoController

After connecting to SE1HUP, you can retrieve CaoController information or set up data retrieval for Execute tags by using CaoVariabe method or by creating a RFID object.

4.1. Sample programming to set UID information to start address of tag data

In this example, set the UID information of the first read RFID tag as the start address. Table 4-1 describes the requirements of the sample program, and Fig. 4-1 describes the flow of the sample program.

Table 4-1 Sample program requirements

Requirements	Description
Host	COM connection
	Host: DENSO WAVE Active USB-COM Port (Active USB-COM) ※Open Device Manager and specify the COM port number to which Active USB-COM is assigned. (The red frame in the figure below becomes Active USB-COM.) 
Process Description	Gets the UIDs of RFID tags.
	Gets the first 16 characters of a RFID tag.
	Set the UID information from the start address.
	Acquires and confirms 16 characters of data from the start address.

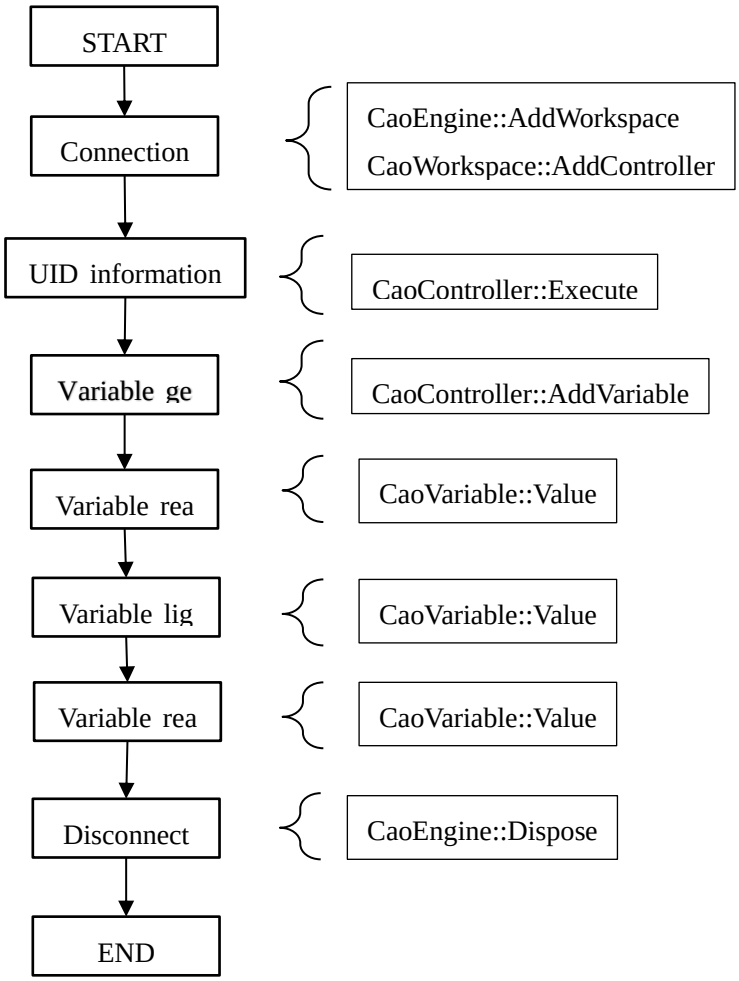


Fig. 4-1 Process flow

Specific codes are given in the following sections.

4.1.1. Sample program

The following is an overview of the sample program.

Sample	SE1HUPSample.cs

```
// Object
Private ORiN2.ManagedCAO.CCaoEngine m_caoEngine = null;
Private ORiN2.ManagedCAO.CCaoWorkspace m_caoWorkspace = null;
Private ORiN2.ManagedCAO.CCaoController m_caoController = null;
Private ORiN2.ManagedCAO.CCaoVariable m_rfidTagData = null;

Public void Main()
{
    // Connection
    This.Connect();
    Try
    {
        // Read UID information
        String uid = ReadUID();
```

```
            // When the UID information is acquired, the processing is executed.
            If (!string.IsNullOrEmpty(uid))
            {
                // Create an option specifying the data size of UID information from the start address.
                String option = "StartPos=0,elem="+uid.Length.ToString()+",UID="+ uid;
                // Create a variable with UID information
                This.m_rfidTagData = this.m_caoController.AddVariable("RFID_ADDRESS_CHAR", option);
                 // Acquires the data before data is written.
                 String beforData = this.m_rfidTagData.Value.ToString();
                 // Write data
                 This.m_rfidTagData.Value = uid;
                 // Get data after writing
                 String afterData = this.m_rfidTagData.Value.ToString();

                 String DispString = "UID Info = " + uid + Environment.NewLine + "Before Writing = " +
beforData + Environment.NewLine + "After Writing = " + afterData;
                 // Show Results in the Console
                 Console.WriteLine(DispString);
             }
        }
        Catch(Exception ex)
        {
                 // Displays error details and results.
              Console.WriteLine(ex.Message.ToString());
        }
        Finally
        {
             // Disconnect
             This.Disconnect();
        }
}

// Connection method
Private void Connect()
{
    // Generate CaoEngine object
    This.m_caoEngine = new ORiN2.ManagedCAO.CCaoEngine();
    // Generate CaoWorkspace object
    This.m_caoWorkspace = this.m_caoEngine.AddWorkspace("NewWrks", "");
    // Generate CaoController object
    This.m_caoController = this.m_caoWorkspace.AddController("SE1HUP","CaoProv.DENSO.SE1HUP",
"","Conn=COM:3,Timeout=2000");
    // Registers a OnMessage event
    This.m_caoController.OnMessage += new OnMesssageEventHandler(OnMessage);
}

// Disconnect method
Private void Disconnect()
{
    This.m_caoEngine.Dispose();
    This.m_caoEngine = null;
}

// Get UID information
Private string ReadUID()
```

```
{
    Return this.m_caoController.Execute("ReadUID", null).ToString();
}
```

4.1.1.1. Connection

To connect to SE1HUP, perform the following steps:

(1) Prepare a variable to hold the object. The objects required to connect to the controller are CaoEngine object, CaoWorkspace object, and CaoController object. CaoWorkpace object does not need to have a variable to obtain CaoController object from CaoWorkspaces. You will also need a CaoVariable for accessing the variable. The following is a code example for C#.

```
// Variables for CaoEngine
Private ORiN2.ManagedCAO.CCaoEngine m_caoEngine = null;
// Variables for CaoWorkspace
Private ORiN2.ManagedCAO.CCaoWorkspace m_caoWorkspace = null;
// Variables for CaoController
Private ORiN2.ManagedCAO.CCaoController m_caoController = null;
// Variables for CaoVariable
Private ORiN2.ManagedCAO.CCaoVariable m_rfidTagData = null;
```

(2) Creates a CaoEngine object. CaoEngine object is generated using the New keyword.

```
// Generate CaoEngine object
This.m_caoEngine = new ORiN2.ManagedCAO.CCaoEngine();
```

(3) Gets or generates a CaoWorkspace object. When you create a CaoEngine object, it defaults to one CaoWorkspaces object and one object. The following is a sample code/default CaoWorkspace for creating a new CaoWorkspace.

```
// Generate CaoWorkspace object
This.m_caoWorkspace = this.m_caoEngine.AddWorkspace("NewWrks", "");
```

(4) Create a CaoController object. To generate a CaoController object, set the provider name to use and the parameters to use. For SE1HUP providers, specify Conn in COM. The following is a code example:

```
// Generate CaoController object
This.m_caoController = this.m_caoWorkspace.AddController("SE1HUP", "CaoProv.DENSO.SE1HUP", "",
                                                         "Conn=COM:3,Timeout=2000");
```

4.1.1.2. Write UID information from the start address

To write to the start address after acquiring the UID information, follow the procedure below.

(1) Use CaoController::Execute method to retrieve UID information. The data obtained by Execute is retrieved with object type, and is converted to string type using ToString().

```
// Execute Execute
This.m_caoController.Execute("ReadUID", null).ToString();
```

(2) Generates RFID_ADDRESS_CHAR <??> variable using the acquired UID information. Specify "0" for StartPos option to specify the start address, specify the number of characters of the acquired UID information for Elem option, and specify the UID string for the UID option.

```
// Create an option specifying the data size of UID information from the start address.
String option = "StartPos=0,elem="+uid.Length.ToString()+",UID="+ uid;
// Generates a Variable with UID-information
This.m_rfidTagData = this.m_caoController.AddVariable("RFID_ADDRESS_CHAR",
```

(3) Uses the generated Variable variable to write UID information to the specified address location. Use Value property to retrieve the pre-write data once. Then set the UID information and retrieve the data to verify that it was last written successfully.

```
// Acquires the data before data is written.
String beforData = this.m_rfidTagData.Value.ToString();
// Write data
This.m_rfidTagData.Value = uid;
// Get data after writing
String afterData = this.m_rfidTagData.Value.ToString();
```

4.1.1.3. Disconnect

To disconnect from the controller, you can erase the generated objects and delete the objects that you want to erase from the collection class that manages the objects. However, you do not need to explicitly remove it if you use ORiN.ManagedCAO. The following is a code example:

```
    // Remove all objects from CaoEngine
    This.m_caoEngine.Dispose();
    // Clear CaoEngine
    This.m_caoEngine = null;
```

5. SE1HUP Provider Error Codes

This provider has its own error code for each method, property, and event.

For information about common ORiN2 errors, see the Error Codes section of ORiN2 Programming Guide (Link).C:¥ORiN2¥CAO¥Doc¥ORiN2_ProgrammersGuide_en.pdf

5.1. List of Error Codes for CaoWorkspace::AddController

Table 5-1 AddController Error Codes

Error Number	Description
0x80111101	There is no Conn optional specification.
0x80111102	Conn option is specified incorrectly.
0x80111103	Conn option has been set to a value that is out of scope.
0x80111202	Timeout option is specified incorrectly.
0x80111203	Timeout option has been set to a value that is out of scope.

5.2. List of Error Codes for CaoController;;Execute

Table 5-2 Error Codes in CaoController::Execute

Error Number	Description
0x80121001	There are not enough arguments for the specified option. Check the optional arguments of the command to be executed.
0x80121005	An unexpected data was received. Review the communication environment, etc.
0x80121006	Execute was executed during trigger mode. Press the trigger switch on SE1HUP to execute the command or execute RisetTrigger command.
0x80121102	The option specification method of the first argument is incorrect. Check the options of the first argument of the command to be executed.
0x80121103	The option of the first argument is set to a value outside the valid range. Check the options of the first argument of the command to be executed.
0x80121202	The option specification method of the second argument is incorrect. Check the option of the second argument of the command to be executed.
0x80121203	The option of the second argument is set to a value outside the valid range. Check the option of the second argument of the command to be executed.
0x80121302	The option specification method of the third argument is incorrect. Check the option of the third argument of the command to be executed.
0x80121303	The option of the third argument is set to a value outside the valid range. Check the option of the third argument of the command to be executed.
0x80120092	A RFID tagging error was returned from SE1HUP.

<table>
<tr><th>Error Number</th><th>Description</th></tr>
<tr><td></td><td>※RFID tag handling error includes the following: Check if RFID tag exists in the valid area of SE1HUP.
<table>
<tr><th>RFID tag error items</th></tr>
<tr><td>No response from tag</td></tr>
<tr><td>Error receiving from tag</td></tr>
<tr><td>Error during tag transmission</td></tr>
<tr><td>Receive error response from tag</td></tr>
<tr><td>Tags with the same UID exist in the area</td></tr>
</table></td></tr>
<tr><td>0x80120093</td><td>A read out-of-range error was returned from SE1HUP.
※The data to be read when a character string is specified may be outside the range of 0x20 to 0x7E. Please confirm by binary specification.</td></tr>
<tr><td>0x80120280</td><td>A value outside the setting range was set in the command parameter of SE1HUP.</td></tr>
</table>

5.3. List of Error Codes for CaoController::AddVariable

Table 5-3 Error Codes in Controller::AddVariable

Error Number	Description
0x80131001	The number of specified options is insufficient. Check the optional arguments of the variable to be added.
0x80131102	The method of specifying the first option is incorrect. Check the options for the variable you want to add.
0x80131103	A value outside the valid range is set for the first option. Check the first option of the command to be executed.
0x80131108	The mandatory first option is not specified.
0x80131202	The method of specifying the second option is incorrect. Check the option of the second argument of the command to be executed.
0x80131203	A value outside the valid range is set for the second option. Check the option of the second argument of the command to be executed.
0x80131208	A mandatory second option is not specified.
0x80131302	The specification method of the third option is incorrect. Check the option of the third argument of the command to be executed.
0x80131303	A value outside the valid range is set for the third option. Check the third option of the command to be executed.

5.4. List of Error Codes for CaoVariable::Value Properties

Table 5-4 CaoVariable::Value Property Error Codes Table

<table>
<tr><th>Error Number</th><th>Description</th></tr>
<tr><td>0x80141005</td><td>An unexpected data was received. Review the communication environment, etc.</td></tr>
<tr><td>0x80141006</td><td>Value property was used during trigger mode. Press the trigger switch on SE1HUP to execute the command or execute RisetTrigger command.</td></tr>
<tr><td>0x80141303</td><td>AddVariable setting for Elem does not match RFID_ADDRESS_BINARY <??> or RFID_ADDRESS_CHAR <??> setting.</td></tr>
<tr><td>0x80140092</td><td>A RFID tagging error was returned from SE1HUP.
※RFID tag handling error includes the following: Check if RFID tag exists in the valid area of SE1HUP.<table><tr><th>RFID tag error items</th></tr><tr><td>No response from tag</td></tr><tr><td>Error receiving from tag</td></tr><tr><td>Error during tag transmission</td></tr><tr><td>Receive error response from tag</td></tr><tr><td>Tags with the same UID exist in the area</td></tr></table></td></tr>
<tr><td>0x80140093</td><td>A read out-of-range error was returned from SE1HUP.
※The data to be read when a character string is specified may be outside the range of 0x20 to 0x7E. Please confirm by binary specification.</td></tr>
<tr><td>0x80140280</td><td>A value outside the setting range was set in the command parameter of SE1HUP.</td></tr>
</table>

5.5. List of Error Codes for CaoController::OnMessage Events

<table>
<tr><th>Error Number</th><th>Description</th></tr>
<tr><td>0x80151007</td><td>An unexpected data was received. Review the communication environment, etc.</td></tr>
<tr><td>0x80150092</td><td>A RFID tagging error was returned from SE1HUP.
※RFID tag handling error includes the following: Check if RFID tag exists in the valid area of SE1HUP.<table><tr><th>RFID tag error items</th></tr><tr><td>No response from tag</td></tr><tr><td>Error receiving from tag</td></tr><tr><td>Error during tag transmission</td></tr><tr><td>Receive error response from tag</td></tr><tr><td>Tags with the same UID exist in the area</td></tr></table></td></tr>
<tr><td>0x80150093</td><td>A read out-of-range error was returned from SE1HUP.</td></tr>
</table>

Error Number	Description
	※The data to be read when a character string is specified may be outside the range of 0x20 to 0x7E. Please confirm by binary specification.

Appendix A. Communication protocol command correspondence table

CaoControllre::Execute method	SE1HUP communication protocol command
SetRFIDReadTriggerWithBinary SetRFIDReadTriggerWithChar ReadRFIDWithBinary ReadRFIDWithChar	RFR0
SetRFIDWriteTriggerWithBinary SetRFIDWriteTriggerWithChar WriteRFIDWithBinary WriteRFIDWithChar	RFW0
SetUIDReadTrigger ReadUID	RFI0
ResetTrigger	RFS0
GetDeviceVersion	VER0
CaoVariable::Value Properties	SE1HUP Communication protocol
@DEVICE_VERSION::get	VER0
@UID::get	RFI0
RFID_ADDRESS_BINARY::get RFID_ADDRESS_CHAR::get	RFR0
RFID_ADDRESS_BINARY::set RFID_ADDRESS_CHAR::set	RFW0