# KEYENCE
# LJ-X8000 providers
# User's Guide

**Version 1.0.0**

**July 2, 2020**

NOTE:

## [Revision History]

| Version | Date | Description |
|---|---|---|
| 1.0.0 | 2020-07-02 | First edition |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

## [Operation check model]

| Model | Version | Notes |
|---|---|---|
| LJ-X8000 | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

## Contents

# 1. Introduction

This manual is a user's guide for KEYENCE Corporation that obtains information from its LJ-X8000. Fig. 1-1 shows the overall configuration of this provider and the device. The providers are referred to as LJ-X8000 providers.

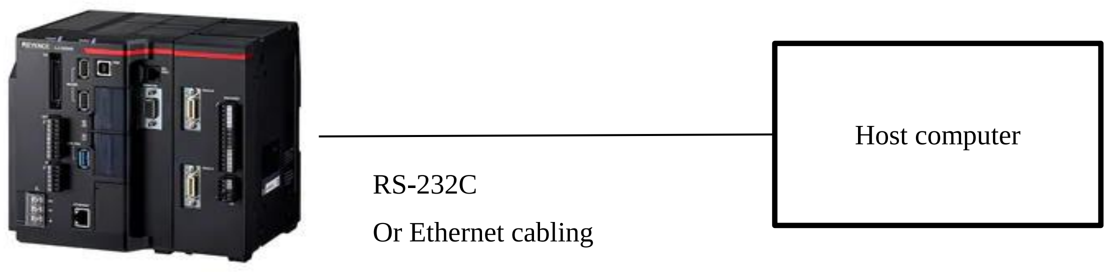

Fig. 1-1 Configuration Diagram

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

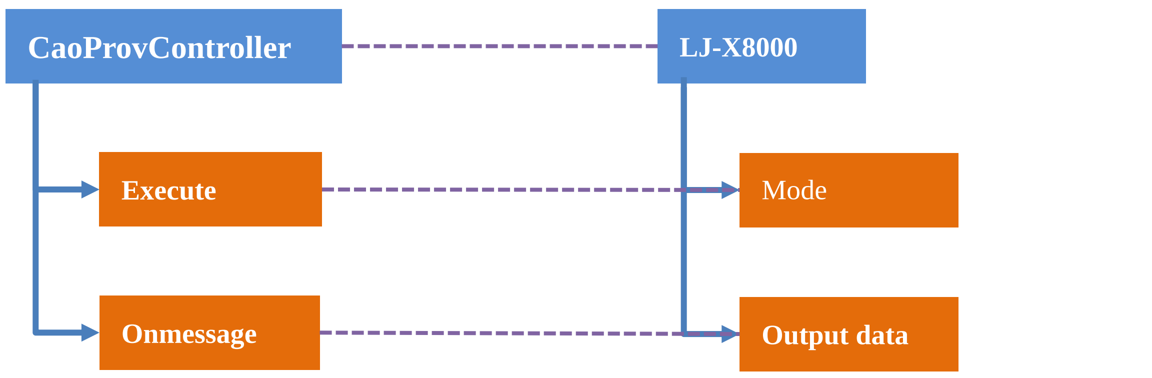

Fig. 1-2 Provider configuration and device information

# 2. Setting Up Your Environment for Application Development

## 2.1. Connecting LJ-X8000 to Client-PC

To connect LJ-X8000 to the client PC, you need to connect a TCP connection using a Ethernet cable or a COM connection using a modular cable.

To receive data from this provider, LJ-X8000 must set delimiters and output settings.

### 2.1.1. LJ-X8000 delimiter setting

This provider supports [CR] or [CR+LF] as the delimiter for sending/receiving data to/from LJ-X8000. If the non-procedural command and output data delimiter settings are set to other than the above in LJ-X8000, change the settings. For the setting procedure, refer to LJ-X8000 series user's manual.

### 2.1.2. How to receive output data

This provider has a function to receive the data set by LJ-X8000.When connecting via COM connection, set the output setting to RS-232C (non-procedural communication). When connecting via Ethernet, set the output setting to Ethernet (non-procedural communication). For the setting procedure, refer to LJ-X8000 series user's manual.

### 2.1.3. About Trigger Input

When LJ-X8000 receives the trigger input, it starts capturing data from the header and outputs the data. This provider allows triggers to be fired against LJ-X8000 (see 3.2.2.4.1) in order to accept triggering inputs from this provider, the trigger settings must be externally triggered in LJ-X8000 settings and RS-232C or Ethernet must be checked in trigger mode. For trigger settings, refer to LJ-X8000 Series User's Guide.

# 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 | P | Get a list of variable names that can be connected | P.11 |
| | Variables | P | Retrieving Variable Collections Holded by the Controller | P.11 |
| | AddVariable | M | Adding Variable Objects | P.11 |
| | Execute | M | Execute Extended Commands | P.12 |
| | OnMessage | E | Message reception event | P.15 |
| CaoVariable | | | | |
| | Value | P | Get/set value | P.15 |

## 3.2. Method properties

### 3.2.1. CaoWorkspace classes

#### 3.2.1.1. AddController method

In CaoWorkspace, add a controller object. LJ-X8000 providers connect to the controllers in the relevant LJ-X8000 series 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.KEYENCE.LJ-X8000",        // Provider name (fixed)
        "<machine name>",                  // Provider execution machine name (unused)
        "<Option>"                         // Option character string (optional)
)
```

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

Option

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 = <communication parameter> | ✓ | Specify the connection destination information. | ETH or COM |
| Timeout= | -- | Specify the communication timeout in ms. | 1-65535<br>Default value: 2000 |
| Delimiter= | -- | Specifies the delimiter set in LJ-X8000 (see 2.1.1).<br>0:CR<br>1:CR+LF | 0-1<br>Default: 0 |

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("LJ-X8000",
                                        "CaoProv.KEYENCE.LJ-X8000",
                                        "",
                                        "Conn=ETH:192.168.10.10,Timeout=5000,delimiter=1");
```

### 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.

**Connecting with a TCP/IP**

"Conn = ETH: <Destination IP>[:<Destination port>[:<Local IP>[:<local port>]]]]]"

| | | |
|---|---|---|
| <Destination IP> | : | Specifies the destination IP address in the format ***.***.***.***.<br>Be sure to specify this item. |
| <Destination port> | : | Specify the port number to connect to. 8500 |

**Connecting with a 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> | : | Data bit. '7'-7bit, '8'-8bit |
| <StopBits> | : | Stop bit. '1'-1bit, '2'-2bit |

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

##### 3.2.1.1.2. Notes on AddController

The version of the device is acquired from LJ-X8000 during AddController. If the version information retrieval fails, 0x80111000 error code is returned and AddController fails. If the version of the device can be obtained, it is retained in CaoController and is used to acquire it without communicating with the device with the @DEVICE_VERSION variable of CaoVariable class described later, and the @DEVICE_MODEL.

### 3.2.2. CaoController classes

#### 3.2.2.1. VariableNames Properties

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

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 specifi ed 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.

Some commands may succeed depending on the operating mode. However, some commands may not be executed by LJ-X8000. For operation, refer to the explanation of each command.

| Command | Description | See Also |
|---|---|---|
| Trigger | Issue the trigger. | P.12 |
| ClearError | Clears the error information. | P.13 |
| ChangeMode | Switches the operation mode. | P.13 |
| GetMode | Reads the current operation mode status. | P.13 |
| ChangeOperationScreen | Switches the operation screen. | P.14 |
| SendRowCommand | Transfers the parameter data to the device as it is. | P.14 |

#### 3.2.2.4.1. Trigger Commands

Sends a triggering command to LJ-X8000. By triggering, the headers that you set to LJ-X8000 capture only once. In order for LJ-X8000 to receive triggers, you must select an external trigger in the trigger settings of LJ-X8000 body.

| Item | Type Description |
|---|---|
| Argument | Nothing in particular |
| Return Value | Nothing in particular |

Usage example

```
// Issue the trigger.
controller.Execute("Trigger");
```

#### 3.2.2.4.2. ClearError Commands

Send a command to clear the error information of LJ-X8000. This command can be executed regardless of the setting mode or operation mode.

| Item | Type Description |
|---|---|
| Argument | Nothing in particular |
| Return Value | Nothing in particular |

Usage example

```
// Clears the error information.
controller.Execute("ClearError");
```

#### 3.2.2.4.3. ChangeMode Commands

Send a command to switch the operation status of LJ-X8000. Switches between the setting mode and operation mode depending on the value set in the option.

Operation when LJ-X8000 accepts a command differs depending on the mode. For details on the commands implemented by this provider, refer to the applicable command descriptions.

| Item | Type | Description | |
|---|---|---|---|
| Argument | VT_UI1 | Specify the mode to switch. | |
| | | Set value | Mode |
| | | 0 | Setting mode |
| | | 1 | Operation mode |
| Return Value | Nothing in particular | | |

Usage example

```
// Switching to Operation Mode
controller.Execute("ChangeMode",1);
```

#### 3.2.2.4.4. GetMode Commands

Send a command to acquire the present operating status of LJ-X8000. This command can be executed in the setting/operation mode. This command can be executed regardless of the setting mode or operation mode.

| Item | Type | Description | |
|---|---|---|---|
| Argument | Nothing in particular | | |
| Return Value | VT_UI1 | Used to acquire the current operation status. | |
| | | Acquired value | Mode |
| | | 0 | Setting mode |
| | | 1 | Operation mode |

Usage example

```
// Acquires the current operation status mode.
Byte mode = controller.Execute("GetMode") as Byte;
```

### 3.2.2.4.5. ChangeOperationScreen Commands

Send a command to switch LJ-X8000 operation display. Switches the operation screen by specifying the tab type and tab number. This command is executed only in operation mode. Please perform the operation mode after putting it into operation mode.

| Item | Type | Description |
|---|---|---|
| Argument | VT_UI1 | Specify the tab type.<br>Specified value / Type:<br>0 — Tool list tab<br>1 — S** tab |
| | VT_UI1 | The setting range varies depending on the tab type.<br>If the tab type is the Tool List tab, specify 1.<br>Range: 0 to 9 |
| Return Value | Nothing in particular | |

| Specified value | Type |
|---|---|
| 0 | Tool list tab |
| 1 | S** tab |

Usage example

```
// Switching the operation screen to the S00 tab
object[] opt = new object[]{1, 0};
controller.Execute("ChangeOperationScreen",);

// Switch the operation screen to the tool list tab.
object[] opt = new object[]{0, 1};
controller.Execute("ChangeOperationScreen",);
```

### 3.2.2.4.6. SendRawCommand Commands

Sends the data specified by the parameter to LJ-X8000 as it is.

| Item | Type | Description |
|---|---|---|
| Argument | VT_BSTR | Specify the data to send as a character string type. |
| Return Value | VT_BSTR | Returns the response from LJ-X8000 as a character string. |

Usage example

```
// Execute VI-command with SendRawCommand
string opt = "VI"
string retValue = controller.Execute("SendRawCommand",opt) as String;
```

#### 3.2.2.5. OnMessage event

You can receive controller error notifications and status changes as OnMessage events. See 3.4 for the events that can be received.

### 3.2.3. CaoVariable classes

#### 3.2.3.1. Value Properties

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. Note that the variable refers to an object of CaoVariable class.

### 3.3.1. CaoController class variable

| Variable name | Description | Value | | See Also |
|---|---|---|---|---|
| | | Get | Put | |
| @MAKER_NAME | Obtain the manufacturer's name. | ✓ | - | P.15 |
| @VERSION | Get the DLL version. | ✓ | - | P.16 |
| @DEVICE_VERSION | Get the device version. | ✓ | - | P.16 |
| @DEVICE_MODEL | Gets the model number of the device. | ✓ | - | P.16 |

#### 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 as string;
```

### 3.3.1.2. @VERSION

Gets the DLL version.

Data Type

| Type | Description |
|---|---|
| VT_BSTR | Get the DLL version.<br>*.*.* |

Usage example

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

### 3.3.1.3. @DEVICE_VERSION

Gets the device version. Getting the device version retains the data acquired during AddController in CaoController and gets the value.

Data Type

| Type | Description |
|---|---|
| VT_BSTR | Gets the device version as a 14-character string. |

Usage example

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

### 3.3.1.4. @DEVICE_MODEL

Gets the model number of the device. When the model number of the device is acquired, the data acquired at the time of AddController is retained in CaoController, and the value is acquired.

Data Type

| Type | Description |
|---|---|
| VT_BSTR | Model number of the device |

Usage example

```
// Add Variable
ORiN2.ManagedCAO.CCaoVariable var = controller.AddVariable("@DEVICE_MODEL","");
// Acquisition of Values
string value = var.Value as string;
```

## 3.4. Event list

You can configure the output data settings in LJ-X8000 and receive the output data as a OnMessage event when it is received by the provider.

The content of each message can be obtained by Value property.

### 3.4.1. Output data

Returns the output data from LJ-X8000 as a character string.

Data Type

| Message number | Message Description | |
|---|---|---|
| 1 | VT_BSTR | Outputs from LJ-X8000 controllers |

### 3.4.2. Disconnect message

If an error is detected when confirming the reception of data from LJ-X8000, it is judged to be disconnected and a message is issued. If this message is issued, reconnect it.

Also, when the COM is connected, a message cannot be issued because an error cannot be detected. If you want to check whether the line is broken during COM communication, use GetMode command of CaoController::Execute command to judge that the line is broken when the data cannot be received. In this case, reconnect.

Data Type

| Message number | Message Description | |
|---|---|---|
| 99 | VT_I4 | Error code number |

# 4. Sample Programming with LJ-X8000 Providers

With LJ-X8000 providers, you can connect your client PC to LJ-X8000 controllers as follows:

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

After you connect to LJ-X8000, you can receive output data from CaoController using Execute method or from the.

## 4.1. Sample programming to receive outgoing data from LJ-X8000

This section explains sample programming in which data set to be output by LJ-X8000 is received as an example. 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 | Connect with a TCP/IP |
| | The destination IP address is 192.168.10.10. |
| | The destination port number is 8500. |
| Process Description | Receiving Outputs from LJ-X8000 |
| | Separate the received data with commas. |

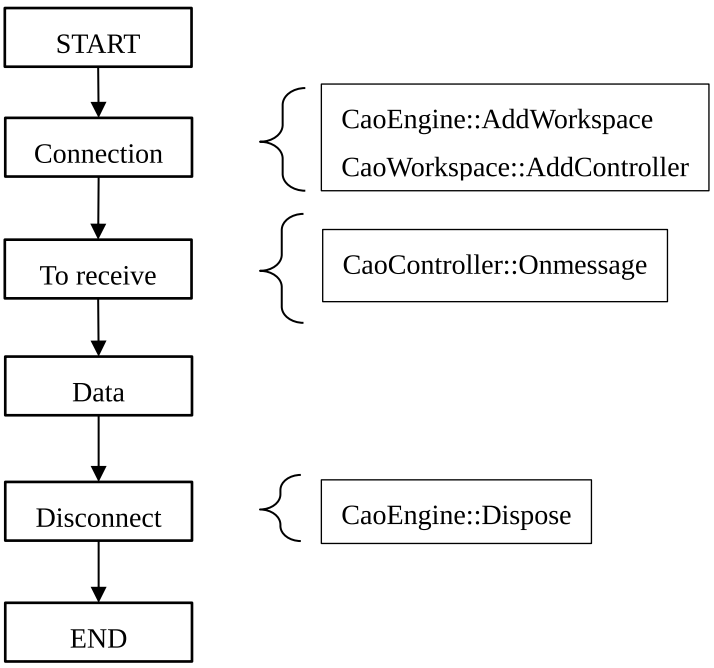

Fig. 4-1 Flow of output data reception

Specific codes are given in the following sections.

### 4.1.1. Sample program

The following is an overview of the sample program.

**Sample** **SampleApp.cs**

```
// Object
private ORiN2.ManagedCAO.CCaoEngine m_caoEngine = null;
private ORiN2.ManagedCAO.CCaoWorkspace m_caoWorkspace = null;
private ORiN2.ManagedCAO.CCaoController m_caoController = null;

public void Main()
{
    // Connection
    this.Connect();
    // Adds the action to take when a OnMessage is received on the controllers.
    m_caoController.OnMessage += new
ORiN2.ManagedCAO.OnMesssageEventHandler(OnMessageEvent);
    // Wait 10 seconds to accept OnMessage events
    thread.Sleep(10000);

    // 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("LJX8000",
        "CaoProv.KEYENCE.LJ-X8000",
        "",
        "Conn=ETH:127.0.0.1:8500, Timeout=1000");
}

// Disconnect method
private void Disconnect()
{
    this.m_caoEngine.Dispose();
    this.m_caoEngine = null;
}

///<summary>
//// OnMessage reception event
///</summary>
private void OnMessageEvent(object sender, ORiN2.ManagedCAO.OnMessageEventArgs e)
{
    // Get message content as string type
    string messageData = e.Message.Value as string;
    // Divide the message content by comma
    string[] messageArray = messageData.Split(',');
}
```

#### 4.1.1.1. Connection

To connect to LJ-X8000, 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_varD0100 = null;
// Variables for CaoVariable
private ORiN2.ManagedCAO.CCaoVariable m_varD1100 = 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 LJ-X8000 providers, specify the device to communicate with in Conn options. The following is a code example:

```
        // Generate CaoController object
        this.m_caoController = this.m_caoWorkspace.AddController("LJX8000",
                    "CaoProv.KEYENCE.LJ-X8000",
                    "",
                    "Conn=ETH:127.0.0.1:8500, Timeout=1000");
```

### 4.1.1.2. To receive data

(1) LJ-X8000 providers issue OnMessage events when they receive output data from LJ-X8000 series. To receive OnMessage events, set OnMessage event handler of CaoController object to the operation when the event is received. The following is a code example:

```
       // Adds the action to take when a Onmessage is received on the controllers.
        m_caoController.OnMessage += new
ORiN2.ManagedCAO.OnMesssageEventHandler(OnMessageEvent);
```

(2) When OnMessage event is issued, the event registered above is executed, and the retrieved data can be retrieved by Value property of OnMessage. Here is an example code:

```
///<summary>
//// OnMessage reception event
///</summary>
private void OnMessageEvent(object sender, ORiN2.ManagedCAO.OnMessageEventArgs e)
{
      // Get message content as string type
      string messageData = e.Message.Value as string;
      // Divide the message content by comma
      string[] messageArray = messageData.Split(',');
}
```

### 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. LJ-X8000 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.

## 5.1. List of Error Codes for CaoWorkspace::AddController

Table 5-1 AddController Error Codes

| Error Number | Description |
|---|---|
| 0x80110000 | Communication check failed on the specified Conn. Check whether the specified Conn option is the IP address, port number, or COM port set on LJ-X8000 series. |
| 0x80111001 | Conn option was specified incorrectly. Check that Conn option was specified correctly. |
| 0x80111002 | The specification of Timeout option is incorrect. Check the specification of Timeout option. |
| 0x80111003 | The specification of Delimiter option is incorrect. Check the specification of Delimiter option. |

## 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. |
| 0x80121002 | The option specification method is incorrect. Check the options of the command to be executed. |
| 0x80121005 | An unexpected data was received. Review the communication environment, etc. |
| 0x801200** | The error code "**" was returned from the device.<br>** = Error code from device for execution command<br>※For error codes from devices, refer to the list of control communication commands in section 9 of LJ-X8000 Series User's Guide. |

# Appendix A.

## Communication protocol command correspondence table

| CaoControllre::AddController method | LJ-X8000 non-procedural communication command |
|---|---|
| Communication confirmation | VI |
| CaoControllre::Execute method | LJ-X8000 non-procedural communication command |
| Trigger | T1 |
| ClearError | CE |
| ChangeMode (when operating mode is specified) | R0 |
| ChangeMode (when setting mode is specified) | S0 |
| GetMode | RM |
| ChangeOperationScreen | VW |