# RAC provider

## RAC communication

Version 1.1.0

# User's guide

July 17, 2012

[Remarks ]

・A part of this provider uses a part of study results that (foundation)Society for the Promotion of Machine Industry Technical Research Institute executed by assisting the cycle race.

## [ Revision history ]

| Version | Date | Content |
|---|---|---|
| 1.0.0.0 | 2006-02-24 | First edition |
| 1.1.0.0 | 2009-07-03 | Deleted Retry processing (RAC has no retry information). Added Clear command. |
| 1.1.0.1 | 2010-02-12 | Added Error codes. |
| 1.1.0 | 2012-07-17 | Document versioning rules was changed. |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

## [ Hardware ]

| Model | Version | Notes |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

## Contents

1. Introduction ..... 4
2. Outline of provider ..... 5
2.1. Outline ..... 5
2.1.1. RAC command ..... 6
2.1.2. Synchronous/asynchronous mode setting ..... 6
2.1.3. UNICODE conversion ..... 6
2.1.4. ISO conversion and EIA conversion ..... 7
2.1.5. Escape conversion ..... 7
2.2. Method and Property ..... 8
2.2.1. CaoWorkspace::AddController method ..... 8
2.2.1.1. Conn option ..... 9
2.2.1.2. PacketOpt option ..... 9
2.2.1.3. EtherOpt option ..... 10
2.2.2. CaoController::AddCommand method ..... 11
2.2.3. CaoController::AddVariable method ..... 11
2.2.4. CaoController::Execute method ..... 11
2.2.5. CaoController::OnMessage event ..... 12
2.2.6. CaoCommand::Execute method ..... 12
2.2.7. CaoCommand::put_Parameters property ..... 12
2.2.8. CaoCommand::get_Parameters property ..... 12
2.2.9. CaoCommand::get_Result property ..... 12
2.2.10. CaoVariable::put_Value property ..... 13
2.2.11. CaoVariable:get_Value property ..... 13
2.2.12. CaoMessage::Reply method ..... 13
2.3. Variable list ..... 14
2.4. Error code ..... 14
3. Sample program ..... 15
3.1. Sample program ..... 15
3.2. Test program ..... 17
3.2.1. Screen layout ..... 17

# 1. Introduction

This is an user's guide of the RAC provider that is the CAO provider to send and receive RAC (Robot Action Command) by the stream communication.

The RAC provider sends and receives RAC for a remote computer. The available communication methods are RS-232C and socket stream connection (TCP/IP) of Ethernet. When TCP/IP is used, you can choose the server mode or the client mode.

With RAC provider, CAO clients can easily send and receive RAC commands.

This book explains the function of this RAC provider and implemented methods.

# 2. Outline of provider

## 2.1. Outline

RAC provider is a provider that sends and receives RAC (Robot Action Command) by the stream communication. Available communication methods are RS-232C and socket stream connection (TCP/IP) of Ethernet.

In RAC provider, you can set the synchronous/asynchronous mode communication as well. Also, the Unicode converter, the escape character converter and other functions are prepared as communication data processing functions.

The file format of RAC provider is DLL (Dynamic Link Library). Please refer to a Table 2-1 for details.

**Table 2-1 RAC provider**

| | |
|---|---|
| File name | CaoProvRAC.dll |
| ProgID | CaoProv.RAC |
| Registration[1] | regsvr32 CaoProvRAC.dll |
| Deregistration | regsvr32 /u CaoProvRAC.dll |

[1] It is not necessary registration/to blot out by hand power when installing it with ORiN SDK.

### 2.1.1. RAC command

The following tables show the mark rule of the RAC request command and the command response. Items surrounded by square brackets ("[]") are option.

**Request command**

<Top-level command >: [< part name >]: [< part number >]:[< subcommand >]: [< parameter>]

| | | |
|---|---|---|
| < top-level command > | : | Either "START" and "STOP", "PUT" or "GET" |
| < part name > | : | Part of target robot for command |
| < part number > | : | Identification number when there are two or more parts specified by part name |
| < subcommand > | : | Content of concrete operation of top-level command |
| < parameter > | : | Parameter necessary for subcommand execution |

**Command response**

< processing result >[,< data type >,< data string >]

| | | |
|---|---|---|
| < processing result > | : | Integral value of HRESULT type |
| < data type > | : | Data type expressed by VARTYPE type |
| < data string > | : | Character-string data converted by data type |

### 2.1.2. Synchronous/asynchronous mode setting

In RAC provider, you can select synchronous/asynchronous mode communication by specifying Sync option of the AddController method (see 2.2.1).

When synchronous mode communication is selected, the control is returned immediately after the transmission of RAC command. The execution result of the RAC command is acquired by OnMessage event of the controller.

When asynchronous mode communication is selected, the control is not returned from the Command::Execute method until the execution result of the RAC command is returned.[2]

### 2.1.3. UNICODE conversion

When the third bit of <Convert> in PacketOpt option of the AddController method (2.2.1) is TRUE, following UNICODE is converted.

(1) A transmission command is converted from Unicode (WCHAR) to S-JIS (CHAR) and sent.

(2) A received response command is converted from S-JIS (CHAR) into Unicode (WCHAR).

When synchronous communication is selected, UNICODE conversions explained above are always applied.

[2] You can use the asynchronous execution of Command class regardless of the asynchronous mode setting.

### 2.1.4. ISO conversion and EIA conversion

It converts data from ASCII code to the specified code and then sends or receives the data.

When this is used with UNICODE conversion function, data is converted into the code specified from UNICODE and sent and received.

### 2.1.5. Escape conversion

Specify a character-code that is used as an escape character by <Escape> in PacketOpt option of the AddController method (2.2.1). When 0 is specified for the character-code, the escape conversion is not done.

Data is encoded at the transmission and is decoded at the reception.

The conversion rule is as follows.

(1) One-byte alphabetic character (A-Z,a-z,0-9) and a part of signs (*-.@_) are not converted.

(2) One-byte space is converted into "+".

(3) Other characters are converted into "Specified escape character" + "hexadecimal two digits of character-code".

Example) Escape character "%"

Before the conversion : @test=This is sample.

After the conversion : @test%3DThis+is+sample.

## 2.2. Method and Property

### 2.2.1. CaoWorkspace::AddController method

In RAC provider, communication is established by specifying a connection parameter with AddController.

At this time, you can specify communication form, connection parameter, timeout setting, synchronous setting, and an escape character with option character string.

The argument specification of AddController is shown as follows.

```
AddController
(
"<Controller name>"       // Controller name
 "CaoProv.RAC",  // Provider name. Fixed.
"<machine name>"// Execution computer name of provider.
"<option>"       // Option character string
)
```

Table 2-2 lists the option character strings. In the Effective device column, a communication device marked with "-" will be ignored when the option is selected.

**Table 2-2 Option character string of CaoWorkspace::AddController**

| Option | Description | Effective device | | |
|---|---|---|---|---|
| | | Ethernet | | com |
| | | Server | Client | |
| Conn<br>=< connection parameter> | Required. Communication form and connection parameter. (See 2.2.1.1) | ✓ | ✓ | ✓ |
| PacketOpt<br>[=< Packet parameter>] | Set the communication packet.<br>(Default: "4:0:0:0" ) (See 2.2.1.2) | ✓ | ✓ | ✓ |
| EtherOpt<br>[=< Ether parameter >] | Configure Ethernet communication.<br>(Default: "0:10:900") (See 2.2.1.3) | ✓ | ✓ | - |
| MyIP<br>[=< local IP address >] | NIC can be selected by specifying IP address by this option when two or more NIC are used. This option is automatically selected when it is omitted. When IP address not allocated in a local computer is specified, an error is returned. | ✓ | ✓ | - |
| Timeout<br>=< timeout period > | Timeout period when sending and receiving. (millisecond) (Default: 500) | ✓ | ✓ | ✓ |
| Sync | Synchronous setting | - | ✓ | ✓[3] |

[3] Please make the server side an asynchronization by the COM connection. The command from the client cannot be received when setting it synchronously.

| | | | | |
|---|---|---|---|---|
| [=<True/False>] | True: Synchronous (default)<br>False: Asynchronous | Asynch<br>ronous | | |

#### 2.2.1.1. Conn option

The following shows connection parameter character string of Conn option. Items enclosed with square brackets ("[]") are omittable. The underlined parameter within each description will be the default value when the option is not specified.

**RS232C device**

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

| | | |
|---|---|---|
| <COM Port> | : | COM port number. '1'-COM1,'2'-COM2,… |
| <BaudRate> | : | Transmission 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> | : | Number of stop bits. '1'-1bit,'2'-2bit. |
| <Flow> | : | Flow control. '1' -Xon/Xoff, '2'-hardware control.<br>You can specify it by taking OR. |

**EtherNet device**

"Conn=eth:[<IP Address>[:<Port No>]]"

| | | |
|---|---|---|
| <IP Address> | : | IP address of TCP/IP. This parameter value will be ignored when the server mode is specified by Mode option.<br>Example:"127.0.0.1","192.168.0.1" |
| <Port No> | : | TCP/IP connection port number. 5006, 5007,... any number is available |

#### 2.2.1.2. PacketOpt option

The following shows the parameter character string of PacketOpt option. Items enclosed with square brackets ("[]") are omittable. The underlined parameter within each description will be the default value when the option is not specified

"PacketOpt =[<Convert>[:<Header>[:<Term>[:<Escape>]]]]"

| | | |
|---|---|---|
| <Convert> | : | Character string conversion.<br>The lower two-bit sets ISO conversion and EIA conversion. The third-bit sets Unicode conversion.[4]<br>'1' - ISO conversion, '2' - EIA conversion , '4' - Unicode conversion<br>Example 1: Unicode conversion only : 4 |

[4] The Unicode conversion is compulsorily set to turning on at the synchronous mode.

Example 2: ISO conversion only : 1
Example 3: Unicode conversion and the EIA conversion : 6

<Header> : Header specification.
'0' - none '1' - ENQ(0x05)

<Term> : Terminator specification.
'0'-CR(0x0D),'1'-LF(0x0A),'2'-CR+LF(0x0D0A)

<Escape> : Character-code used for escape conversion.
'0' Not escape-converted.
Other than '0' Escape-converted with the specified character code.

PacketOpt option setting has the following restrictions.

- An error occurs if both the ISO conversion and the EIA conversion are turned ON at the same time.
- An error occurs if the escape conversion is set while the Unicode conversion is OFF.

The following table shows available combinations of conversions.

**Table 2-3 Available option values**

| <Convert> | | | | <Escape> |
|---|---|---|---|---|
| ISO conversion | EIA conversion | Unicode conversion | Set value | Set value |
| - | - | - | 0 | 0 |
| ✓ | - | - | 1 | 0 |
| - | ✓ | - | 2 | 0 |
| - | - | ✓ | 4 | any |
| ✓ | - | ✓ | 5 | any |
| - | ✓ | ✓ | 6 | any |

### 2.2.1.3. EtherOpt option

The following shows parameter character strings of EtherOpt option. Items enclosed with square brackets ("[]") are omittable. The underlined parameter within each description will be the default value when the option is not specified. When RS232C is specified for the device, this option is disregarded.

"EtherOpt =[<Mode>[:<ConnMax>[:<ConnTimeout>]]]"

<Mode> : Character string conversion.
'0' - client mode, Other than '0' - Server mode
This option is ignored if "com" is specified in Conn option.

<ConnMax> : Number of maximum clients at server mode. (Default: 10)
This value is disregarded, except for the server mode.

<ConnTimeout> : The maximum communication waiting time at server mode. (Default: 900) This value is disregarded, except for the server mode.
The client is disconnected when there is no communication from the client at the specified time period. The unit of this parameter is second.

### 2.2.2. CaoController::AddCommand method

This method fails in the asynchronous communication or EtherNet (server).

This method generates a command object to execute RAC command in the synchronous mode.

You can use any name for the command name.

The option character string is not used.

```
AddCommand
(
"< command name >"        // Command name (any name)
"< option >"    // Option character string (unused)
)
```

### 2.2.3. CaoController::AddVariable method

This method fails in the asynchronous mode communication or EtherNet (server).

This method generates a variable object that synchronously executes GET and PUT command of RAC.

The variable name is specified as follows.

"[< part name >]:[< part number >]:[< subcommand>]"

Items enclosed with square brackets ("[]") are omittable and null character string. The option character string is not used.

```
AddVariable
(
"<variable name>"//"[<part name >]:[<part number>]:[<subcommand>]"
"<option>"      // Option character string (unused)
)
```

### 2.2.4. CaoController::Execute method

This method fails in the asynchronous communication or EtherNet (server). There are only two commands available ;"Send" and "Clear". "Clear" only clears the buffer. There is no argument.

"Send" command transmits the RAC command specified in <RAC command> of the argument as-is. A grammatical check of the RAC command is not performed. In this case, you must enter the RAC command with the character string type.

```
Execute
(
 "Clear",                // Buffer clear
 "",            // (omittable)
)
```

```
Execute
(
 "Send",                    // RAC command transmission
"<RAC command>" // [top command]:[Part name]:[Part number]:[Subcommand]:[Parameter]
)
```

### 2.2.5. CaoController::OnMessage event

This event is not generated at synchronous communication.

When the RAC provider receives the data from the communication device, data is received and passed to the client as OnMessage event of the CaoController class. At this time, receive data is stored in the Message::Value property as-is.

Whether the received data is RAC command or a reply command is not confirmed.

### 2.2.6. CaoCommand::Execute method

A RAC command is generated, and executed from the data set to the Parameter property.

The execution result is stored in the Result property.

### 2.2.7. CaoCommand::put_Parameters property

This property sets a RAC command to execute. You can specify the command with the character string, character string array, and VARIANT array and is interpreted as follows by the data type.

**Table 2-4 Interpretive procedure of each data type of RAC command**

| Data type of command | Interpretive procedure |
|---|---|
| Character string (VT_BSTR) | RAC command that has been completed |
| Array (VT_???\|VT_ARRAY) | Each element of RAC command.<br>Only the first to the fifth elements are used and the remaining data is disregarded.<br>If the number of elements are five or less, leave the remaining data empty when a RAC command is created.<br>The data of each element is converted into the character string. |

### 2.2.8. CaoCommand::get_Parameters property

This property obtains a command set to the Parameters property.

When the command is not set, VT_EMPTY is returned.

### 2.2.9. CaoCommand::get_Result property

This property obtains the execution result of CaoCommand::Execute method that has been executed

immediately before.

The execution result of RAC returns the processing result with HRESULT, and puts the response data in the return value. The return value becomes VT_EMPTY when there is no response data.

### 2.2.10. CaoVariable::put_Value property

This property executes a PUT command of RAC. In this timing, a RAC request command is generated from the combination of the variable name and the argument. Values specified by arguments are converted to "<data type>, <value>".

The generated RAC command is shown as follows.

```
"PUT:<variable name>:<argument of put_Value>"
```

The command response of RAC returns the processing result with HRESULT. The response data cannot be acquired.

```
put_Value
(
"<set value>"            //<parameter>
)
```

### 2.2.11. CaoVariable:get_Value property

This property executes a GET command of RAC. RAC request command is generated from the variable name.

The generated RAC command is shown as follows.

```
"GET:< variable name >:"
```

The execution result of RAC returns the processing result with HRESULT, and puts the response data in the return value. The return value becomes VT_EMPTY when there is no response data.

### 2.2.12. CaoMessage::Reply method

The argument <reply message> of this method must be specified by the character string type. This method fails if other data types are used.

The specified character string is transmitted as a reply command to the RAC request command of the Message::Value property.

The reply command written in the argument <reply message> is replied as-is. A grammatical check of the RAC command is not performed.

```
Reply
(
"<reply message>"        // Response data
```

)

## 2.3. Variable list

There is no RAC provider-original variable.

## 2.4. Error code

RAC provider has no original error code. For ORiN2 common errors, please refer to the chapter of the error code in "ORiN2 Programming guide".

# 3. Sample program

## 3.1. Sample program

The following sample shows how to establish a communication between "Server.exe" which runs in the server mode and "Client.exe" which runs in the client mode with TCP/IP communication.

Server IP address : 10.7.9.250

Execution command: GET:sample:1:Val:

**List 3-1 SampleServer.frm**

```
Private eng As CaoEngine
Private WithEvents ctrl As CaoController

Private Sub Form_Load()

 Set eng = New CaoEngine

'Communication starts in the server mode.
 Set ctrl = eng.Workspaces(0).AddController("Sample", _
 "CaoProv.RAC", _
 "", _
 "Conn=eth::5006,Mode=1")

End Sub

'Receive an event
Private Sub ctrl_OnMessage(ppCaoMess As CAOLib.ICaoMessage)

'Reply the received contents as-is.
 ppCaoMess.Reply text1.text

End Sub
```

**List 3-2 SampleClient.frm**

```
Private eng As CaoEngine
Private ctrl As CaoController
Private cmd As CaoCommand

Private Sub Form_Load()

 Set eng = New CaoEngine

'Start a communication in the client mode.
 Set ctrl = eng.Workspaces(0).AddController("Sample", _
 "CaoProv.RAC", _
 "", _
 "Conn=eth:10.7.9.250:5006")

 Set cmd = ctrl.AddCommand("sample")
End Sub

Private Sub Command1_Click()
On Error GoTo ErrProc

'Execute RAC command
 cmd.Parameters = "GET:sample:1::10"
 cmd.execute(0)

 Exit Sub
ErrProc:
 MsgBox Err.Description
End Sub
```

## 3.2. Test program

RACTester is prepared as a test client only for the RAC provider of CAO. This client is designed to handle only CAO interface implemented in RAC provider.

### 3.2.1. Screen layout

The following figure shows the screen layout of RACTester.

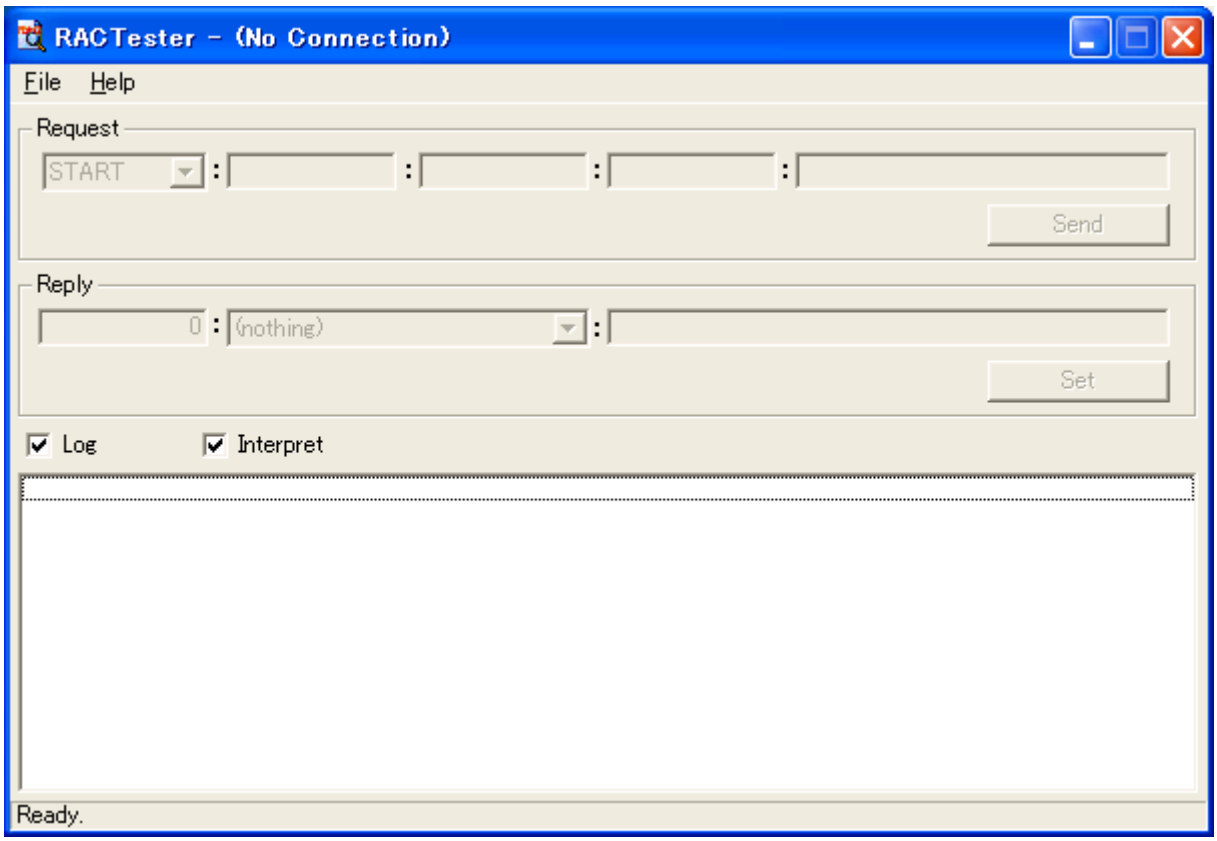


**Figure3-1 RACTester screen**

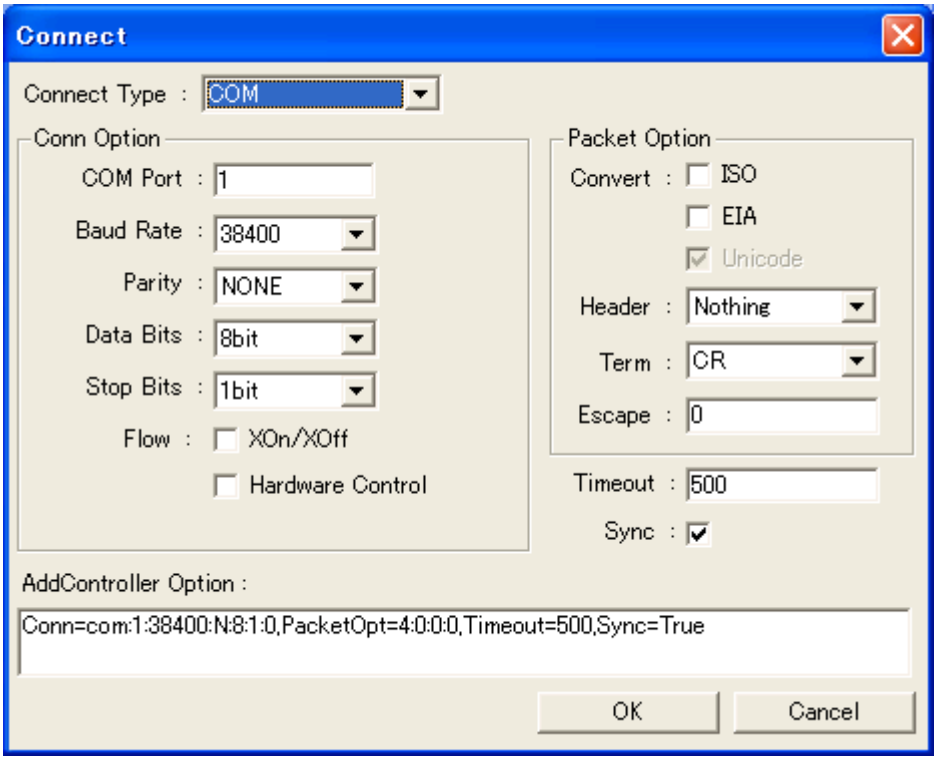


**Figure3-2 "Connect" Screen**