YASKAWA Cockpit
Add-on Application Software Development Manual

Upon receipt of the product and prior to initial operation, read these instructions thoroughly, and retain for future reference.

Please have the following information available when contacting Yaskawa Customer Support:

- System
- Primary Application
- Software Version (Located on Programming Pendant by selecting: {Main Menu} - {System Info} - {Version})
- Robot Serial Number (Located on robot data plate)
- Robot Sales Order Number (Located on controller data plate)

Part Number: 186633-1CD
Revision: 0
DANGER

- Read this manual carefully and be sure to understand its contents. Any matter, including operation, usage, measures, and an item to use, not described in this manual must be regarded as "prohibited" or "improper".

NOTICE

- YASKAWA may modify this model without notice when necessary due to product improvements, modifications, or changes in specifications.
- If such modification is made, the manual number will also be revised.
- If your copy of the manual is damaged or lost, contact a YASKAWA representative to order a new copy. The representatives are listed on the back cover. Be sure to tell the representative the manual number listed on the front cover.
Notes for Safe Operation

Read this manual carefully before installation, operation, maintenance, or inspection of the YRC1000micro/YRC1000/DX200/DX100/RM100.

In this manual, the Notes for Safe Operation are classified as “DANGER”, “WARNING”, “CAUTION”, or “NOTICE”.

**DANGER**

Indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury. Safety Signs identified by the signal word DANGER should be used sparingly and only for those situations presenting the most serious hazards.

**WARNING**

Indicates a potentially hazardous situation which, if not avoided, will result in death or serious injury. Hazards identified by the signal word WARNING present a lesser degree of risk of injury or death than those identified by the signal word DANGER.

**CAUTION**

Indicates a hazardous situation, which if not avoided, could result in minor or moderate injury. It may also be used without the safety alert symbol as an alternative to “NOTICE”.

**NOTICE**

NOTICE is the preferred signal word to address practices not related to personal injury. The safety alert symbol should not be used with this signal word. As an alternative to “NOTICE”, the word “CAUTION” without the safety alert symbol may be used to indicate a message not related to personal injury.

Even items described as “CAUTION” may result in a serious accident in some situations. At any rate, be sure to follow these important items.

**NOTE**

To ensure safe and efficient operation at all times, be sure to follow all instructions, even if not designated as “DANGER”, “WARNING” and “CAUTION”.
Registered Trademark

In this manual, names of companies, corporations, or products are trademarks, registered trademarks, or brand names for each company or corporation. The indications of (R) and TM are omitted.
Contents

1 Introduction to YASKAWA Cockpit ................................................................. 1-1
  1.1 Features ............................................................................................................. 1-1
  1.2 Specifications of the Data Analysis Add-On Function ...................................... 1-1
  1.3 Specifications of the Data Window Add-On Function ...................................... 1-2

2 Extension Function ............................................................................................ 2-1
  2.1 Specifications of the Data Collection Add-On Function .................................... 2-1
  2.2 Example of Application Software Configuration Using Add-On Functions .......... 2-2

3 Development Environment .................................................................................. 3-1
  3.1 Configuration of the YCP Development Environment ....................................... 3-1
  3.2 YCP Development Task Flow ........................................................................... 3-2
  3.3 Development Environments for DA Apps and DC Apps .................................... 3-3
  3.4 Development Environment for DW Apps ........................................................ 3-3
  3.5 Contents on the DVD ......................................................................................... 3-4
  3.6 Adding Apps ..................................................................................................... 3-5
    3.6.1 DA Apps and DC Apps .................................................................................. 3-5
    3.6.2 DW Apps .................................................................................................... 3-5
      3.6.2.1 Manifest File Creation Procedure .......................................................... 3-5

4 Introduction to the Web API ................................................................................ 4-1
  4.1 Restrictions on Web API Calls ......................................................................... 4-1
  4.2 Types of Web API Calls .................................................................................... 4-2
  4.3 Creating a Project .............................................................................................. 4-3
    4.3.1 Data Analysis Function (DA App) ............................................................... 4-3
    4.3.2 Data Collection Function (DC App) ............................................................ 4-5
    4.3.3 Data Window Function (DW App) ............................................................... 4-7

5 Registering Add-On Apps .................................................................................... 5-1
  5.1 Add/Edit Add-Ons Page .................................................................................... 5-1
1 Introduction to YASKAWA Cockpit

1.1 Features

YASKAWA Cockpit (abbreviated as “YCP” in this manual) is software that collects data from YASKAWA devices (e.g., robot controllers and machine controllers), monitors the condition and operating status of robots, creates backups of data, provides notification of alarms, and performs other various functions.

- Collects various types of robot data required for analysis and stores that data in the database.
- Using the web API provided by YCP, data analysis and data window add-on application software (called “Add-on Apps” in this manual) can be created that utilize the collected data.
- Created Add-on Apps can be added to and executed in YCP.

![Fig. 1-1: Add-On Apps That Can Be Created by the Customer](image)

1.2 Specifications of the Data Analysis Add-On Function

Data analysis Add-on Apps created by the customer can be added to and used in YCP.

<table>
<thead>
<tr>
<th>Item</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of software that can be added</td>
<td>Executable files</td>
</tr>
<tr>
<td></td>
<td>Example: .EXE</td>
</tr>
<tr>
<td>Interface with the database</td>
<td>Web API</td>
</tr>
<tr>
<td>Reference program (Sample program)</td>
<td>Programming language: C#, C++, Java, Python, etc.</td>
</tr>
</tbody>
</table>
1.3 Specifications of the Data Window Add-On Function

Data window Add-on Apps created by the customer can be added to and used in YCP.

<table>
<thead>
<tr>
<th>Item</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of software that can be added</td>
<td>The following files contained in a ZIP file are added to YCP. Example: .JSP (JavaServer Pages files), .CSS (stylesheets), .JS (script)</td>
</tr>
<tr>
<td>Interface with the database</td>
<td>Web API</td>
</tr>
<tr>
<td>Reference program</td>
<td>Programming language: JSP</td>
</tr>
</tbody>
</table>

Fig. 1-2: Configuration of the Data Analysis Add-On Function

Fig. 1-3: Configuration of the Data Window Add-On Function
2 Extension Function

2.1 Specifications of the Data Collection Add-On Function

Add-on Apps that collect data from non-YASKAWA devices can be added to and used in YCP.

Table 2-1: Specifications of the Data Collection Add-On Function

<table>
<thead>
<tr>
<th>Item</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of software that can be added</td>
<td>Executable files</td>
</tr>
<tr>
<td></td>
<td>Example: .EXE</td>
</tr>
<tr>
<td>Interface with the database</td>
<td>Web API</td>
</tr>
<tr>
<td>Reference program (Sample program)</td>
<td>Programming language: C#, C++, Java, Python, etc.</td>
</tr>
</tbody>
</table>

Fig. 2-1: Configuration of the Data Collection Add-On Function

(Device other than robot controller: e.g., PLC or sensor)
2.2 Example of Application Software Configuration Using Add-On Functions

There are many different kinds of application software configurations depending on the purpose of the system. The following diagram shows a basic example of a software configuration.

- DC App: Data collection application software
- DA App: Data analysis application software
- DW App: Data window application software

![Diagram of software configuration](image)
3 Development Environment

3.1 Configuration of the YCP Development Environment

Fig. 3-1: Configuration of the YCP Development Environment

Table 3-1: Elements of the YCP Development Environment

<table>
<thead>
<tr>
<th>No.</th>
<th>Device</th>
<th>Specifications</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Windows computer</td>
<td>CPU: Core i7, 4Core</td>
<td>Prepared by customer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Operating system: Windows 10 Professional 64-bit</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Text editor</td>
<td>For creating the source code of the application software.</td>
<td>Prepared by customer</td>
</tr>
<tr>
<td>3</td>
<td>Web browser</td>
<td>For viewing add-ons.</td>
<td>Prepared by customer</td>
</tr>
<tr>
<td>4</td>
<td>Compiler, linker, libraries, etc.</td>
<td>For creating the executable application software.</td>
<td>Prepared by customer</td>
</tr>
<tr>
<td>5</td>
<td>Ethernet cable</td>
<td>For connecting the device and computer to provide communications for the developed application software.</td>
<td>Prepared by customer</td>
</tr>
<tr>
<td>6</td>
<td>Ethernet hub</td>
<td>For connecting the computer and controllers.</td>
<td>Prepared by customer</td>
</tr>
<tr>
<td>7</td>
<td>Installation DVD</td>
<td>For installing YCP on the Windows computer.</td>
<td>Provided by the manufacturer</td>
</tr>
</tbody>
</table>
## 3.2 YCP Development Task Flow

<table>
<thead>
<tr>
<th>Step</th>
<th>Task Details</th>
<th>Reference</th>
</tr>
</thead>
</table>
| 1. Prepare the development environment. | Install the following items on the computer (program development environment).  
• YCP  
• Sample database  
• Sample source code  
• Visual Studio for application development | — |
| 2. Develop the Add-on App using the sample source code. | 1. Replace the default data and sample database created by the installer, and check the behavior of the sample from the web browser.  
2. Restore the sample database to the original database, and develop the application software to collect, analyze, or view the data. | chapter 4.3 “Creating a Project” |
| 3. Load (add) the App on computer for YCP and verify it with the actual devices. | 1. Connect the actual devices and collect actual data.  
2. Upload the DW Add-on App to YCP and check the GUI. | chapter 5 “Registering Add-On Apps” |
| 4. Adjust the schedule for the resources, such as the CPU load. | 1. Check the resources with the YCP resource monitor to determine if data is being lost due to processing delays between application software or if errors are occurring.  
2. Adjust the execution schedule and other functions based on the content checked with the YCP resource monitor. | — |
3.3 Development Environments for DA Apps and DC Apps

The following table gives the development environments for DA Apps and DC Apps.

Table 3-2: Development Environments for DA Apps and DC Apps

<table>
<thead>
<tr>
<th>Item</th>
<th>Development Environment</th>
</tr>
</thead>
</table>
| DA App    | The following functions can be implemented:  
• Issue HTTP request and analyze HTTP response  
• Serialize/deserialize JSON  
Development tool: Visual C++/C#, Eclipse (Java), etc. |
| DC App    | The following functions can be implemented:  
• Issue HTTP request and analyze HTTP response  
• Serialize/deserialize JSON  
• Capable of communicating with the controller for data collection (supports the physical layer and protocol).  
Development tool: Visual C++/C#, Eclipse (Java), etc. |

3.4 Development Environment for DW Apps

The following table gives the development environment for DW Apps.

Table 3-3: Development Environment for DW Apps

<table>
<thead>
<tr>
<th>Item</th>
<th>Development Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DW App</td>
<td>Debugging environment contained in the web browser, such as Internet Explorer and Google Chrome, or the Eclipse+ JSDT plug-in.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Web browser</th>
<th>Debugging environment</th>
</tr>
</thead>
</table>
| Internet Explorer | F12 developer’s tool  
Start with [F12] |
| Google Chrome   | Chrome DevTools  
Start with [Ctrl] + [Shift] + [I] |

**NOTE**

- DW Apps are executed in the web browser. For this reason, the source code and other parts of the DW App can be easily viewed and accessed if you use the functions in the web browser. Therefore, do not include any processing that will be problematic if leaked in the DW App, such as proprietary information or algorithms.
3.5 Contents on the DVD

The DVD contains the following data and documents.

Table 3-4: Contents on the DVD

<table>
<thead>
<tr>
<th>Folder</th>
<th>Item</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root</td>
<td>YCP installer</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>YCP Add-On Application Software Programmer's Manual</td>
<td>This manual describes programming procedures.</td>
</tr>
<tr>
<td></td>
<td>YCP Add-On Application Software Reference Manual</td>
<td>This manual describes the specifications of the web API.</td>
</tr>
<tr>
<td>Manifest file tool</td>
<td>Manifest File Tool</td>
<td>—</td>
</tr>
<tr>
<td>Samples</td>
<td>Sample source code</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>Sample database</td>
<td>—</td>
</tr>
</tbody>
</table>
3.6 Adding Apps

3.6.1 DA Apps and DC Apps

DA Apps and DC Apps can be developed in the customer-selected programming language.

3.6.2 DW Apps

DW Apps are registered from the YASKAWA Cockpit GUI. When registering an App, select the factory, line, cell, and controller to associate with the App, and then register the DW App project file.

When registering the DW App project file, compress the DW App into a ZIP file without using any folders. The DW App must also contain a manifest file (manifest.txt).

The following section gives the procedure for creating the manifest file.

3.6.2.1 Manifest File Creation Procedure

1. On the command prompt, navigate to the location where “manifestTool.jar” has been stored, and then execute the following “java -jar manifestTool.jar” command. The Manifest File Tool will start.

2. Enter the appropriate information for Addon-Title, Addon-Version, and Addon-Vendor.

- **AddOn-Title**
  - Enter the name of the Add-on App.

- **AddOn-Version**
  - Enter the version number of the Add-on App.

- **AddOn-Vendor**
  - Enter the vendor of the Add-on App.

Click “Clear” to clear the information that was entered.
3. Development Environment
3.6 Adding Apps

3. Click “Create File”.

4. Manifest.txt will be created in the same folder as the Manifest File Tool. The content entered for Addon-Title, Addon-Version, and Addon-Vendor are displayed in the following locations on the Add/Edit Add-Ons window.
4 Introduction to the Web API

4.1 Restrictions on Web API Calls

Web API calls conform to HTTP.

Table 4-1: Calling the Web API

<table>
<thead>
<tr>
<th>Item</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port number</td>
<td>The default values are listed below.</td>
</tr>
<tr>
<td></td>
<td>• HTTP: 8080</td>
</tr>
<tr>
<td></td>
<td>• HTTPS: 8443</td>
</tr>
<tr>
<td>Methods</td>
<td>GET, POST, PUT, DELETE</td>
</tr>
<tr>
<td>Request header</td>
<td>The following header is required.</td>
</tr>
<tr>
<td></td>
<td>• Content-Type: application/json</td>
</tr>
<tr>
<td>Character encoding</td>
<td>UTF-8</td>
</tr>
<tr>
<td>HTTP version</td>
<td>HTTP1.1</td>
</tr>
</tbody>
</table>

The following is an example of a web API call (request).

```
POST/DDMSWeb/api/saveCollectedData
Content-Type: application/json

{"title":"Hello World!", "body":"This is my first post!"}
```
4.2 Types of Web API Calls

Over 100 types of web API calls have been prepared, including calls for accessing data in the database. For details, refer to the following manual.

YASKAWA Cockpit Add-on Application Software Web API Reference Manual (HW1485817)

Table 4-2: Examples of Web API Calls

<table>
<thead>
<tr>
<th>Web API (HTTP) Call</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET /apps/{appname}/validate</td>
<td>Validates the application software license. It returns the validation result and expiration date of the specified App.</td>
</tr>
<tr>
<td>GET /factories</td>
<td>Retrieves the list of factories. It returns all factory keys registered to this YCP. If no factories are registered, a blank array is returned (“[]”).</td>
</tr>
<tr>
<td>GET /factories/{factory-key}/lines</td>
<td>Retrieves the list of lines. It returns all line keys registered to the factory defined by factory-key. If no lines are registered, a blank array is returned (“[]”).</td>
</tr>
<tr>
<td>GET /factories/{factory-key}/lines/{line-key}/cells</td>
<td>Retrieves the list of cells. It returns all cell keys registered to the factory defined by factory-key and line-key. If no cells are registered, a blank array is returned (“[]”).</td>
</tr>
<tr>
<td>GET /factories/{factory-key}/lines/{line-key}/cells/{cell-key}/controllers/{controller-key}/attributegroups</td>
<td>Retrieves the attribute group keys for the controller attributes. If there are no groups, a blank array is returned (“[]”).</td>
</tr>
<tr>
<td>GET /factories/{factory-key}/lines/{line-key}/cells/{cell-key}/controllers/{controller-key}/data</td>
<td>Retrieves raw data related to the controller. Raw data is returned in JSON format. A sample response is contained in &quot;Other Documents&quot; (/apidoc/sample-getdata.html). If there is no data, a blank array is returned (“[]”).</td>
</tr>
<tr>
<td>POST /factories/{factory-key}/lines/{line-key}/cells/{cell-key}/controllers/{controller-key}/data</td>
<td>Stores collected data (raw data) to YCP System Core. Collected data is passed in the body of a request and stored in the database.</td>
</tr>
<tr>
<td>GET /factories/{factory-key}/lines/{line-key}/cells/{cell-key}/controllers/{controller-key}/data</td>
<td>Retrieves raw data related to the controller. Raw data is returned in JSON format. A sample response is contained in &quot;Other Documents&quot; (/apidoc/sample-getdata.html). If there is no data, a blank array is returned (“[]”).</td>
</tr>
<tr>
<td>DELETE /factories/{factory-key}/lines/{line-key}/cells/{cell-key}/controllers/{controllerkey}/data/items/{item-key}</td>
<td>Deletes an item setting for collected data of the controller.</td>
</tr>
</tbody>
</table>
4.3 Creating a Project

4.3.1 Data Analysis Function (DA App)

The following diagram shows the configuration of a DA App.

*Fig. 4-1: Configuration of a DA App*

![Diagram of DA App configuration]

A DA App takes data collected from controllers, mechanisms, and devices in the database and analysis results, attributes, and files of other DA Apps as input, and it stores the results of executing various types of processing, such as data analysis and data processing, to the database. The App is implemented as a separate executable file from YASKAWA Cockpit.

The executable file can be implemented at the following types of application software:

- Java (.jar, .war)
- C++, C# (.exe)
- Script (.rb, .php, etc.)

Scripts require the execution environment for each type of file. For example, .rb files require Ruby to be installed on the server that will execute the file, and .php files require PHP to be installed.
4 Introduction to the Web API
4.3 Creating a Project

The following diagram shows the configuration of DA App functions.

*Device configuration: line, cell, controller, mechanism, device

The web API provided by the YASKAWA Cockpit web server is used to store data to and read data from the database. Therefore, the DA App can be executed on the computer that runs YASKAWA Cockpit, and it can also be executed on a separate device (e.g., computer) connected to the network. Data processing, such as aggregating and analyzing data, can be performed using data obtained through the web API. A DA App can also have its own unique user interface.

As additional functions, the App has a scheduler that can manage the time of each function, a thread manager that can execute functions in threads, and a function to output execution results to a log.

The App also has a function to register OPC UA reference data when retrieving data stored to the database from an OPC UA server.
4.3.2 Data Collection Function (DC App)

The following diagram shows the configuration of a DC App.

A DC App collects data related to controllers, mechanisms, and devices, and transmits the collected data to the database. The App is implemented as a separate executable file from YASKAWA Cockpit.

The executable file can be implemented at the following types of application software:

- Java (.jar, .war)
- C++, C# (.exe)
- Script (.js, .rb, .php, etc.)
To collect data from controllers, mechanisms, and devices, the DC App must be compatible with the communications medium and protocol for each controller, mechanism, and device. The web API provided by the YASKAWA Cockpit web server is used to store data to and read data from the database. Therefore, the DC App can be executed on the computer that runs YASKAWA Cockpit, and it can also be executed on a separate device (e.g., computer) connected to the network. A DC App can also have its own unique user interface.

As additional functions, the App has a scheduler that can manage the time of each function, a thread manager that can execute functions in threads, and a function to output execution results to a log.

The App also has a function to register OPC UA reference data when retrieving data stored to the database from an OPC UA server.

*Device configuration: line, cell, controller, mechanism, device*
4.3.3 Data Window Function (DW App)

The following diagram shows the configuration of a DW App.

Fig. 4-5: Configuration of a DW App

A DW App can be associated with factories, lines, cells, or controllers to visualize data (collected data, analysis results, attributes, and files) stored in the database.

When a DW App is registered in YASKAWA Cockpit, it is stored to the file system. To use a DW App, access the YASKAWA Cockpit GUI from a web browser, and click the menu of the DW App from the page on the level at which the DW App has been associated. When the DW App is started, the App stored in storage is loaded, rendered, and executed in the web browser.

Therefore, DW Apps can use the following features and documents supported as standard by the web browser.

- HTML documents (.html and .jsp)
- Images (.jpg, .png, .gif, etc.)
- Stylesheets (.css)
- JavaScript (.js)
The following diagram shows the configuration of DW App functions.

*Device configuration: line, cell, controller, mechanism, device

You can create a GUI that accesses and modifies data (collected data, analysis results, attributes, and files) held in YASKAWA Cockpit by calling the web API using JavaScript.
5 Registering Add-On Apps

5.1 Add/Edit Add-Ons Page

Fig. 5-1: Add/Edit Add-Ons Page

1 "Add"
Use this button to display the following page on which you can add an add-on.

Fig. 5-2: Add-On Page

• UI Name
Specify the name of the add-on.

• Description
Specify a comment about the add-on.

• Zip File
This can be used to upload the add-on module.

• Register UI to
Select the target that will display the add-on.

2 "Edit"
Use this button to edit the content registered for the applicable add-on.

3 "Delete"
Use this button to delete the applicable add-on.