CITIZEN

CITIZEN XML Device Control Service JavaScript Device Control SDK Programming Manual

For Ver. 1.00

CITIZEN SYSTEMS JAPAN CO., LTD.

Revision History

Date	Version	History
6/1/2018	1.00	Initial version
10/25/2018		- Added CT-S751 to the support printer models. - Added BC-NL3000U to the support peripheral devices.
3/7/2019		- Added CT-S4500 to the support printer models.
6/3/2020		- Added CL-E300EX/303EX, CL-E321EX/331EX to the support printer models.
7/8/2020		- Modified the description of "1.1.System Overview".
3/3/2021		- Added CT-E601 to the support printer models. - Modified the URL of the service. - Changed the XML tag name. Old) SetCursorPostistion, Postistion New) SetCursorPosition, Position - Added how to check the service version.
11/21/2021		- Added CT-S801III/851III to the support printer models. (Page 7) - Added DSP01-LT2/DSP02-LS2 to the support peripheral devices. (Page 7)
12/28/2023		- Added CL-S700III/703III to the support printer models. (Page 7)

Notes

1. Unauthorized reproduction of this document, in part or in whole, is strictly prohibited.
2. The content of this document is subject to change without notice.
3. Every effort has been made to ensure the accuracy of the information in this document. However, if you notice any errors or problems, please contact CITIZEN SYSTEMS JAPAN.
4. CITIZEN SYSTEMS JAPAN shall not be responsible for any consequences resulting from use, notwithstanding item 3 above.
5. If you do not agree to the above, you cannot use this library.

Trademarks

Please note that this manual is subject to change without notice for reasons such as product improvement.

Please request the latest document from CITIZEN SYSTEMS JAPAN before use.

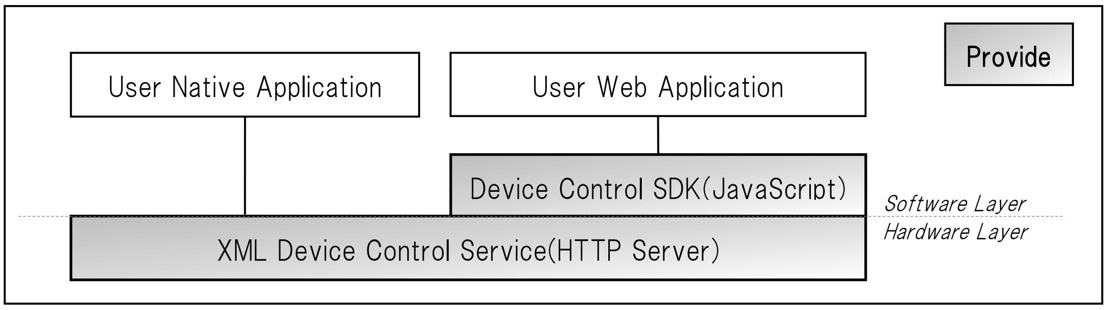
Contents

1. Introduction

This document is a manual for programmers for CITIZEN XML Device Control Service.

1.1. System Overview

CITIZEN XML Device Control Service is provided to control a peripheral device connected to a printer without a device driver in a multi-platform environment which is not operating system dependent.
If you want to control the printing of the printer, please use the CITIZEN XML Print service (JavaScript POS Print SDK / JavaScript Label Print SDK).
Since the control method is HTTP (XML) based, a peripheral device can be controlled easily from a Web service environment. Furthermore, Device Control SDK is provided as a library for XML Device Control to control using JavaScript on the client side. The following shows a conceptual diagram of the provided service.

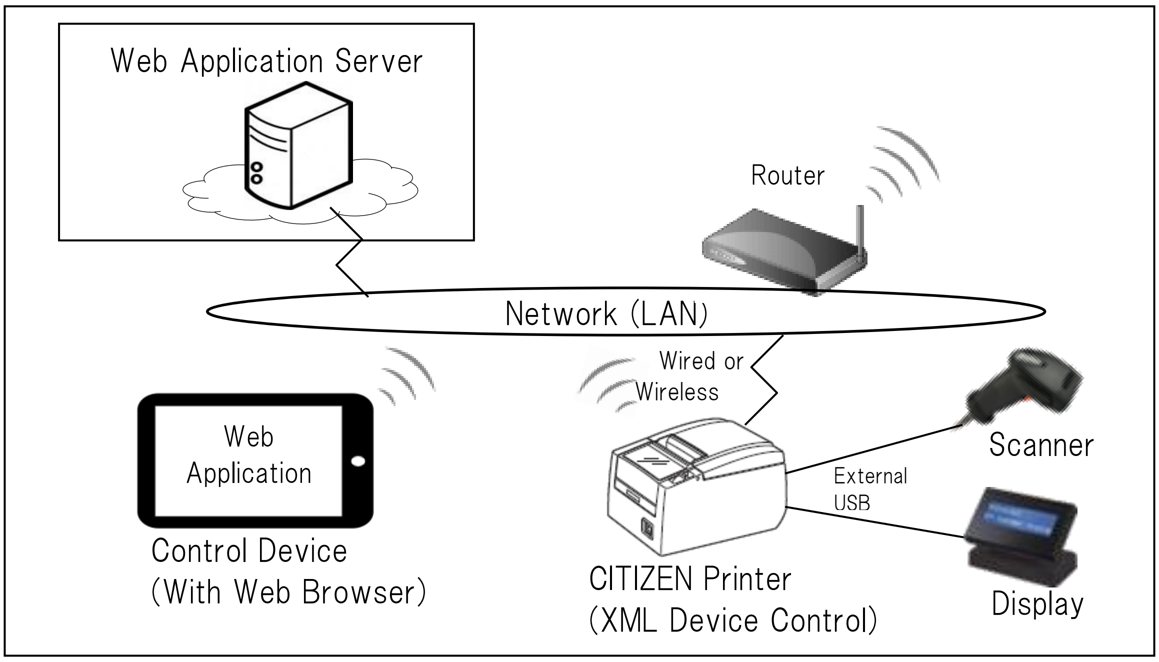


1.2. Scope of This Document

This document is intended to be a reference for developers of applications that use CITIZEN XML Device Control Service compatible peripheral devices.
For the control specifications for HTTP (XML) based control, refer to "2. XML Device Control Messages", "3. Device Control Tags", and "4. XML Device Control Service Settings" in this document.
For the JavaScript SDK specifications for control from a Web service environment, refer to "5. JavaScript Device Control SDK".
For details on checking the operation of this service, refer to "6. Sample Programs".

1.3. System Configuration Example

The following shows a system configuration example for CITIZEN XML Device Control Service.



1.4.Supported Printer Models

The applicable printer models of this service and the corresponding interfaces for those models are as follows.
For details on the functions of each model, refer to each instruction manual.

Applicable Printer Models	Interface
CT-E601/651, CT-S251/751/4500, CL-E300EX/303EX/321EX/331EX	Wired LAN (model number: IF2-EFX2) Wired/Wireless LAN (model number: IF2-WFx5, IF2-WFx6)
CT-S601II/651II/801II/851II/801III/851III, CL-E720/730, CL-S400DT/700III/703III	Wired LAN (model number: IF1-EFX2) Wired/Wireless LAN (model number: IF1-WFx6)

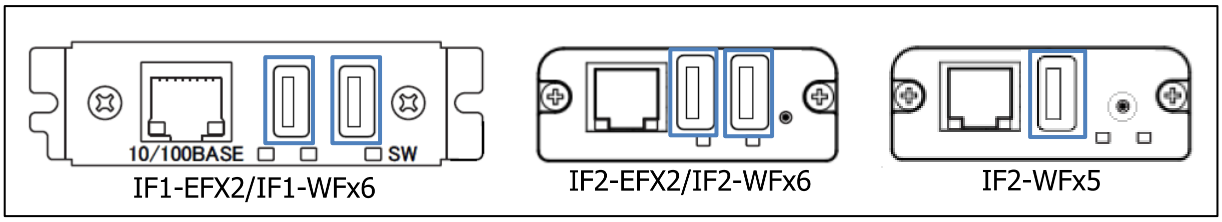
For the version number of this service, refer to "4.1.2. Service Status screen". Usage and functions may differ depending on the version.

1.5.Supported Peripheral Devices

The models of peripheral devices applicable for control with this service are as follows. For details on the functions of each model, refer to the instruction manual of each peripheral device.

Applicable Display	Product Specification Overview
DSP01-LT/DSP01-LT2	TFT line display
DSP02-LS/DSP02-LS2	STN line display
Applicable Scanner	**Product Specification Overview**
SCN01-Z1D	1D barcode scanner
SCN02-Z2D	2D barcode scanner
BC-NL3000U	2D barcode scanner

For the connection to the applicable peripheral device, first turn off the printer power and then connect to a USB port of the corresponding interface shown in the figure below. Next, turn on the printer power, wait about 30 seconds until the applicable peripheral device becomes ready to use so as to ensure stable operation, and then execute the control start process of the peripheral device.



The following lists prohibited actions that must not be performed with regard to a peripheral device connection.

Prohibited Actions

- Connecting other than a supported peripheral device (USB hub, smartphone, etc.) to a USB port of the interface.
- Inserting and removing the cable connector of the peripheral device into/from a USB port of the interface while the printer power is on.
- Connecting multiple peripheral devices of the same type to a USB port of the interface (e.g. connecting two displays).

If any of the above actions is performed, it may lead to the misoperation and, in the worst case, cause a failure of the printer or connected peripheral device.

The following explains the setting information of each peripheral device for using this service.

1.5.1. Line Display Settings

Normally, you can leave a display set to the factory default state. If you will use a display with settings different from the standard factory default settings, match the settings to the specifications of the display to be used as described in "4.1.1. Service Setting Screen".

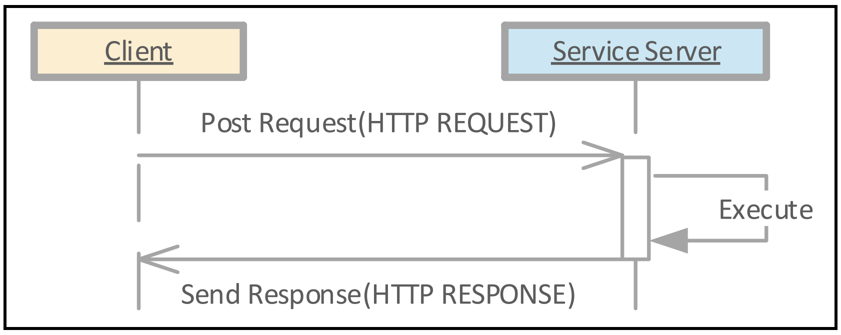
1.5.2. Barcode Scanner Settings

Check that the scanner to be connected is set as follows. For the setting procedure, refer to the instruction manual of the peripheral device to be used.

Item	Value	Description
Interface	USB HID Class	Communication protocol
Keyboard	US Keyboard	Keyboard language
Terminator	Enter	Data suffix

2.XML Device Control Messages

The service user issues a request message as an HTTP request and then receives a response message from the service as an HTTP response as shown in the figure below. Request messages and response messages are defined in the "CitizenXML-Device.xsd" XML schema file.



2.1. Request Messages

Control of the device (peripheral device) is performed in accordance with the request messages issued from clients.

2.1.1. Transmission Method and Message Structure

Send a SOAP message using the following method for a request message.

・ HTTP URL:	http(s)://[IP address of this service]/xmldevice * In the service version 1.0 and earlier, http://[IP address of this service]:8085
・ HTTP method:	POST
・ HTTP header:	Content-Type: text/xml; charset=UTF-8

The structure of a request message is as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
 <s:Body>

  <DeviceRequest xmlns="http://www.citizen.co.jp/DeviceControl/" MajorVersion="1">
   <!-- Request ID -->
   <MessageID>a55f0b87-4bb1-94fb-a92b-dfa2913c4343</MessageID>

    <!-- Line Display Control-->
    <LineDisplay>
      <ClearDisplay/>
      <SetCursorPostistion>
        <Postistion>1,1</Postistion>
       </SetCursorPostistion>
      <DisplayText>
         <Data>Display␍␊Text</Data>
       </DisplayText>
    </LineDisplay>

  </DeviceRequest>

 </s:Body>
</s:Envelope>
```

<DeviceRequest> tag

2.1.2. <DeviceRequest> Tag

Insert the device control tag within the <DeviceRequest> tag in the request message for controlling the device. For details on the device control tags, refer to "3. Device Control Tags" in this document.

2.2. Response Messages

The result of a request can be checked from the response message from this service.

2.2.1. Message Structure

The structure of a response message is as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>

    <DeviceResponse xmlns="http://www.citizen.co.jp/DeviceControl/" MajorVersion="1">
      <MessageID>7338ab39-92f2-416b-ac00-25b21956142e</MessageID>
      <Response ResponseCode="OK">
        <RequestID>a55f0b87-4bb1-94fb-a92b-dfa2913c4343</RequestID>
      </Response>
    </DeviceResponse>

  </s:Body>
</s:Envelope>
```

<DeviceResponse> tag

2.2.2. Acquiring the Request Result

The result of a request can be checked in the information within the <DeviceResponse> tag.

Item	Description
ResponseCode attribute	Stores the process result.
MessageID element	Stores the ID for identifying the response message.
RequestID element	Stores the message ID specified when the message was sent.
BusinessError element	Stores the error information when an error occurred.

Checking for Error Occurrence

Whether or not an error occurred can be checked by checking the value of the ResponseCode attribute of the Response element.

Code	Description
OK	Ended normally.
Rejected	Error occurred.

The following is an example of the Response element when the process ended normally.

```
<Response ResponseCode="OK">
  <RequestID>a55f0b87-4bb1-94fb-a92b-dfa2913c4343</RequestID>
</Response>
```

Checking Error Information

The cause of a result can be checked from the contents of the Code and Description elements in the BusinessError element within the Response element when an error occurs. For details, refer to "2.2.3 Error Codes" in this document.

The following is an example of the Response element when an error occurred.

```
<Response ResponseCode="Rejected">
  <RequestID>a55f0b87-4bb1-94fb-a92b-dfa2913c4343</RequestID>
  <BusinessError Severity="Error">
    <Code>ENotConnect</Code>
    <Description>Failed connection to the device.</Description>
  </BusinessError>
</Response>
```

2.2.3. Error Codes

The error code, error description, and other detailed information are set in the BusinessError element in the Response element within a response message. The error codes used with this service are shown below.

Errors That Occur during Service Processing

Code	Description
RequestInvalid	Request information is invalid
DeviceTimeout	Device timed out

Errors That Occur during Device Processing

Code	Description
ENotConnect	Device connection failed
EConnectNotFound	Unsupported model
EIllegal	Unsupported or invalid parameter
EOffline	Device is offline
ENoExist	No data error
EFailure	Process cannot be executed
ETimeout	Processing timeout
EptrBadFormat	Data format error
EptrTooBig	Data size error

Description is set in the BusinessError element in the Response element. The following shows an example.

```
<BusinessError Severity="Error">
  <Code>ENotConnect</Code>
  <Description> Failed connection to the device.</Description>
</BusinessError>
```

2.3. Acquiring Device Status

To acquire the device status, use the GetDeviceInfo tag to specify a device information acquisition request.

Parameters

Attribute	Meaning	Settable range
Name	Device name	Applicable device name If this was not specified, all registered devices are applicable
Status	Status information flag	Add status information when "true" is specified.

An example of a request message is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
 <s:Body>
  <DeviceRequest xmlns="http://www.citizen.co.jp/DeviceControl/" MajorVersion="1">
   <!-- Device information acquisition request -->
   <GetDeviceInfo Status="true"/>
  </DeviceRequest>
 </s:Body>
</s:Envelope>
```

An example of a response message is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
 <s:Body>
  <DeviceResponse xmlns="http://www.citizen.co.jp/DeviceControl/" MajorVersion="1">
   <MessageID>5e60c4c2-c0a4-44a2-83ee-2984a1c620a0</MessageID>
   <Response ResponseCode="OK">
     <RequestID />
   </Response>
   <GetDeviceInfo>
    <Device Name="LineDisplay" Status="Online" />      } <GetDeviceInfo> tag
    <Device Name="Scanner" Status="Online" />
   </GetDeviceInfo>
  </DeviceResponse>
 </s:Body>
</s:Envelope>
```

A status code is used to store the contents of the device information in the Status attribute of the Device element of the response message. The status codes used with this service are shown below.

2.3.1. Line Display Information (Name="LineDisplay")

Status code	Description
Online	State in which use is possible.
EConnectNotFound	A device not supported by this service is connected.
ENotConnect	A line display is not connected.
EOffline	The device is being used by another service.

2.3.2. Barcode Scanner Information (Name="Scanner")

Status code	Description
Online	State in which use is possible.
EConnectNotFound	A device not supported by this service is connected.
ENotConnect	A barcode scanner is not connected.
EOffline	The device is being used by another service.
StatusScanning	State in which waiting to receive data from the device.

3.Device Control Tags

3.1. Device Control Tag List

The device control tags that can be used with this service are shown below.

Classification	Function	Tag	Description
General	Message ID	<MessageID>	Specify this to enable the sender to identify the message.
Device Control	Line display control	<Display>	Specify this to enable control of a line display.
	Barcode scanner control	<Scanner>	Specify this to enable control of a barcode scanner.

3.1.1. Message ID (MessageID Tag)

Value

Specify the request message ID.

Description

This tag is used to enable the sender to identify the message.
The specified request message ID is added to the <RequestID> tag of the response message. For details on response messages, refer to "2.2. Response Messages" in this document.

Usage example

<MessageID>12345678**</MessageID>**

3.1.2. Line Display Control (LineDisplay Tag)

Value

The following control tags can be inserted in the LineDisplay tag.

Function	Tag	Description
Displaying text string	<DisplayText>	Displays text.
Clearing display	<ClearDisplay>	Clears the displayed text.
Blink display	<BlinkDisplay>	Starts blink display.
Setting display	<SetDisplayMode>	Sets the display mode.
	<SetDisplayConfig>	Configures various settings.
Setting cursor	<SetCursorPostistion>	Specifies the cursor position.
	<MoveCursor>	Moves the cursor.
	<SetCursorType>	Displays the cursor position.
Initializing	<InitializeDisplay>	Initializes the display.
Setting text string display	<SetEncode>	Specifies the encoding of the text strings to be sent.
	<SetInternationalCharacterset>	Specifies the international character.

Description

This tag is used to control a line display. For details, refer to "3.2. Line Display Control Tag Details" in this document.

3.1.3. Barcode Scanner Control (Scanner Tag)

Value

The following control tags can be inserted in the Scanner tag.

Function	Tag	Description
Getting data	<GetScanData>	Gets the data.
Aborting getting of data	<AbortScan>	Aborts the getting of data.

Description

This tag is used to control a barcode scanner. For details, refer to "3.3. Barcode Scanner Control Tag Details" in this document.

3.2. Line Display Control Tag Details

3.2.1. Displaying Text (DisplayText Tag)

Parameters

The meanings and settable values of the parameters are as follows.

Element	Meaning	Settable range
Data	Text data	
Attribute	Text attribute	0: Standard 64: Reverse (Omittable element, 0 when omit)

Description

This tag is used to display text from the current cursor position.
Reverse can be specified for the text attribute.

Usage example

```
<LineDisplay>
  <DisplayText>
    <Data>Hello, World!</Data>
    <Attribute>0</Attribute>
  </DisplayText>
</LineDisplay>
```

3.2.2. Clearing Display (ClearDisplay Tag)

Parameters

The meanings and settable values of the parameters are as follows.

Element	Meaning	Settable range
DisplayArea	Clear area	All: Entire area CursorLine: Cursor line (Omittable element, All when omit)

Description

This tag clears the displayed text.

Usage example

```
<LineDisplay>
  <ClearDisplay>
    <DisplayArea>All</DisplayArea>
  </ClearDisplay>
</LineDisplay>
```

3.2.3. Display Blink (BlinkDisplay Tag)

Parameters

The meanings and settable values of the parameters are as follows.

Element	Meaning	Settable range
IntervalBlink	Blink interval (msec)	From 0

Description

This tag causes the entire display screen to blink.
The blink interval (msec) specifies the interval for on and off. If 0 is specified for the blink interval, blinking is disabled.

Usage example

```
<LineDisplay>
  <BlinkDisplay>
    <IntervalBlink>1000</IntervalBlink>
  </BlinkDisplay>
</LineDisplay>
```

3.2.4. Setting Screen Mode (SetDisplayMode Tag)

Parameters

The meanings and settable values of the parameters are as follows.

Element	Meaning	Settable range
DisplayMode	Display mode	Overwrite: Overwrite mode VerticalScroll: Vertical scroll mode HorizontalScroll: Horizontal scroll mode

Description

This sets the following display modes.

DisplayMode	Overview
Overwrite	Overwrites the text at the cursor position and moves the cursor to the right. (The cursor moves to the bottom left edge for input when it is at the top right edge, and the cursor moves to the top left edge for input when it is at the bottom right edge.)
VerticalScroll	Scrolls the display line of the top edge to the bottom edge by cursor up movement when the cursor is at the top edge (or by left movement when it is at the left edge). Scrolls the display line of the bottom edge to the top edge by cursor down movement when the cursor is at the bottom edge (or by right movement when it is at the right edge).
HorizontalScroll	Scrolls the text leftward in respect to the current cursor line by cursor right movement (or by text input) when the cursor is at the right edge. Scrolls the text rightward in respect to the current cursor line by cursor left movement when the cursor is at the left edge.

Usage example

```
<LineDisplay>
  <SetDisplayMode>
    <DisplayMode>HorizontalScroll</DisplayMode>
  </SetDisplayMode>
</LineDisplay>
```

3.2.5. Display Settings (SetDisplayConfig Tag)

Parameters

The meanings and settable values of the parameters are as follows.

Element	Meaning	Settable range
Brightness	Brightness (%)	0 to 100

Description

This tag changes the brightness of the display screen.
The higher the numerical value, the brighter the brightness becomes. If 0 is specified, the screen turns off (the display content is retained).
After this is set, blinking of the entire display screen is disabled.

Usage example

```
<LineDisplay>
  <SetDisplayConfig>
    <Brightness>40</Brightness>
  </SetDisplayConfig>
</LineDisplay>
```

3.2.6. Setting Cursor (SetCursorPosition Tag)

Parameters

The meanings and settable values of the parameters are as follows.

Element	Meaning	Settable range
Postition	Cursor position	From 1, from 1

Description

This tag is used to set the cursor.

The cursor position is the movement coordinates of the cursor, and is configured with only two ASCII numbers separated by a comma. The configuration is listed in the order of digit position and line position.

Usage example

```
<LineDisplay>
  <SetCursorPosition>
    <Position>1,2</Position>
  </SetCursorPosition>
</LineDisplay>
```

3.2.7. Moving Cursor (MoveCursor Tag)

Parameters

The meanings and settable values of the parameters are as follows.

Element	Meaning	Settable range
Movement	Cursor movement amount	-128 to 127, -128 to 127

Description

This tag is used to move the cursor.
Movement is from the current cursor position. The cursor movement amount is configured with only two ASCII numbers separated by a comma. The configuration is listed in the order of leftward/rightward movement amount (-: leftward, +: rightward) and upward/downward movement amount (-: upward, +: downward).

Usage example

```
<LineDisplay>
  <MoveCursor>
    <Movement>0,1</Movement>
  </MoveCursor>

  <MoveCursor>
    <Movement>-1,0</Movement>
  </MoveCursor>
</LineDisplay>
```

3.2.8. Specifying Cursor Type (SetCursorType Tag)

Parameters

The meanings and settable values of the parameters are as follows.

Element	Meaning	Settable range
CursorType	Cursor type specification	None: Hide cursor UnderLine: Display cursor (Omittable element, UnderLine when omit)

Description

This displays the current cursor position on the display.

Usage example

```
<LineDisplay>
  <SetCursorType>
    <CursorType>UnderLine</CursorType>
  </SetCursorType>
</LineDisplay>
```

3.2.9. Initializing Device (InitializeDisplay Tag)

Parameters

None

Description

Initializes the device.

Usage example

```
<LineDisplay>
   <InitializeDisplay/>
</LineDisplay>
```

3.2.10. Text Display (SetEncoding Tag)

Parameters

The meanings and settable values of the parameters are as follows.

Element	Meaning	Settable range
Encode	Character encoding	SingleByteCharacter: 8-bit Character Japanese: Japanese (katakana) SimplifiedChinese: Simplified Chinese Korean: Korean TraditionalChinese: Traditional Chinese None: No conversion

Description

This sets the character encoding for when sending the text data (element data / UTF-8 format) in the DisplayText tag to the device. The initial value of this setting is "SingleByteCharacter".

Usage example

```
<LineDisplay>
  <SetEncoding>
    <Encode>Japanese</Encode>
  </SetEncoding>
  <DisplayText>
    <Data>ｺﾝﾆﾁﾜ</Data>
  </DisplayText>
</DisplayText>
```

3.2.11. Specifying International Character Set (SetInternationalCharacterset Tag)

Parameters

The meanings and settable values of the parameters are as follows.

Element	**Meaning**	**Settable range**
Characterset	Specify international character set.	0 to 16

Description

This sets the following international character sets.

Characterset	International Character Set	Characterset	International Character Set
0	America	9	Norway
1	France	10	Denmark II
2	Germany	11	Spain II
3	Britain	12	Latin America
4	Denmark I	13	Korea
5	Sweden	14	Croatia
6	Italy	15	China
7	Spain I	16	Vietnam
8	Japan		

Usage example

```
<LineDisplay>
  <SetInternationalCharacterset>
    <Characterset>8</Characterset>
  </SetInternationalCharacterset>
  <SetEncoding>
    <Encode>Japanese</Encode>
  </SetEncoding>
  <DisplayText>
    <Data>Total:□□1,010</Data>
  </DisplayText>
</LineDisplay>
```

3.3.Barcode Scanner Control Tag Details

3.3.1. Getting Scan Data (GetScanData Tag)

Parameters

The meanings and settable values of the parameters are as follows.

Element	Meaning	Settable range
Timeout	Timeout (sec)	0 to 60

Response Messages

The meaning of a response message is as follows.

Element	Meaning	Supplement
Data	Scan data	

Description

This tag is used to acquire data from the device. The acquired data is returned in the Data element of the response message. It waits for the data from the device for the time specified for the timeout. If the data is not received from the device despite waiting the time specified for the timeout, an error (ETimeout) is returned.

Any data that has been received from the device before calling this tag is accumulated. In that case, the accumulated chunk of data is returned when this tag is called. Furthermore, if there is no accumulated data, at the point when new data is received from the device, the received data is immediately returned without waiting for the timeout time.

Usage example

```
<Scanner>
  <GetScanData>
    <Timeout>10</Timeout>
  </GetScanData>
</Scanner>
```

- Response Messages -

```
<DeviceResponse>
   …
   <Scanner>
     <GetScanData>
       <Data>CTJV052985</Data>
     </GetScanData>
   </Scanner>
   …
```

- Response messages -

When there are two or more sets of scan data (the first Data tag is the oldest)

```
<DeviceResponse>
   …
   <Scanner>
     <GetScanData>
       <Data>CTJV052985</Data>      │ OLD
       <Data>0123456789</Data>      │
         …                          │
       <Data>ABCDEFGHIJ</Data>      ↓ NEW
     </GetScanData>
   </Scanner>
   …
```

3.3.2. Aborting Getting of Scan Data (AbortScan Tag)

Parameters

None

Description

This tag aborts the process to wait to get the scan data from the device.

Usage example

```
<Scanner>
  <AbortScan/>
</Scanner>
```

4.XML Device Control Service Settings

This chapter describes how to set CITIZEN XML Device Control Service.

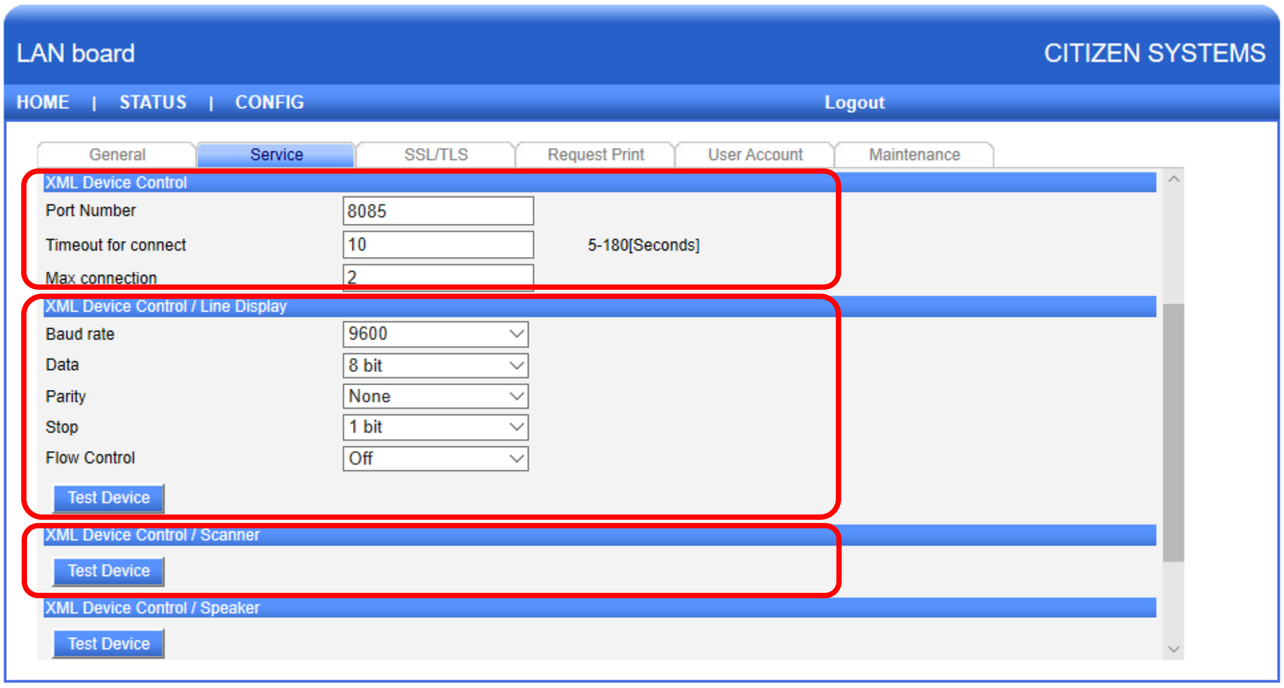
4.1. Web Manager

The settings for the devices can be changed by connecting from a Web browser to each printer. For details on the basic operations, refer to the interface board instruction manual of the printer.

This document describes the setting items of XML Device Control Service.

4.1.1. Service Setting Screen

The settings of the service provided by the printer can be set in the following Service screen.



XML Device Control
Port Number 8085
Timeout for connect 10 5-180[Seconds]
Max connection 2

XML Device Control / Line Display
Baud rate 9600
Data 8 bit
Parity None
Stop 1 bit
Flow Control Off

XML Device Control / Scanner

XML Device Control / Speaker

This screen is not displayed for a printer that is not supported by the XML Device Control Service so use supported printers.

Each setting retains its value even when the power is turned off. When the Factory Default process is performed, each setting is set to its initial value.

XML Device Control

Set the following general settings for the XML Device Control service.

Item	Initial setting	Setting range	Description
Port Number	8085	1025 to 65535	Connection port number
Timeout for connect	10	5 to 180	Timeout time (sec) for waiting for control to start
Max Connections	2	1 to 3	Maximum number of simultaneous connections (normally, use the initial value)

XML Device Control / Line Display

Set the following general settings for displays. The initial values of settings are already the appropriate values for the supported displays so for normal use, do not change them.

Item	Initial setting	Setting range
Baud rate	9600	2400, 4800, 9600, 19200, 38400, 57600, 115200
Data	8 bit	7 bit, 8 bit
Parity	None	None, Odd, Even
Stop	1 bit	1 bit , 2 bit
Flow Control	Off	Hardware, Xon/Xoff, Off

After changing the settings, press the Submit button at the bottom of the screen and then press the Save & Reboot button of the Maintenance menu. The settings will take effect after the board has rebooted.

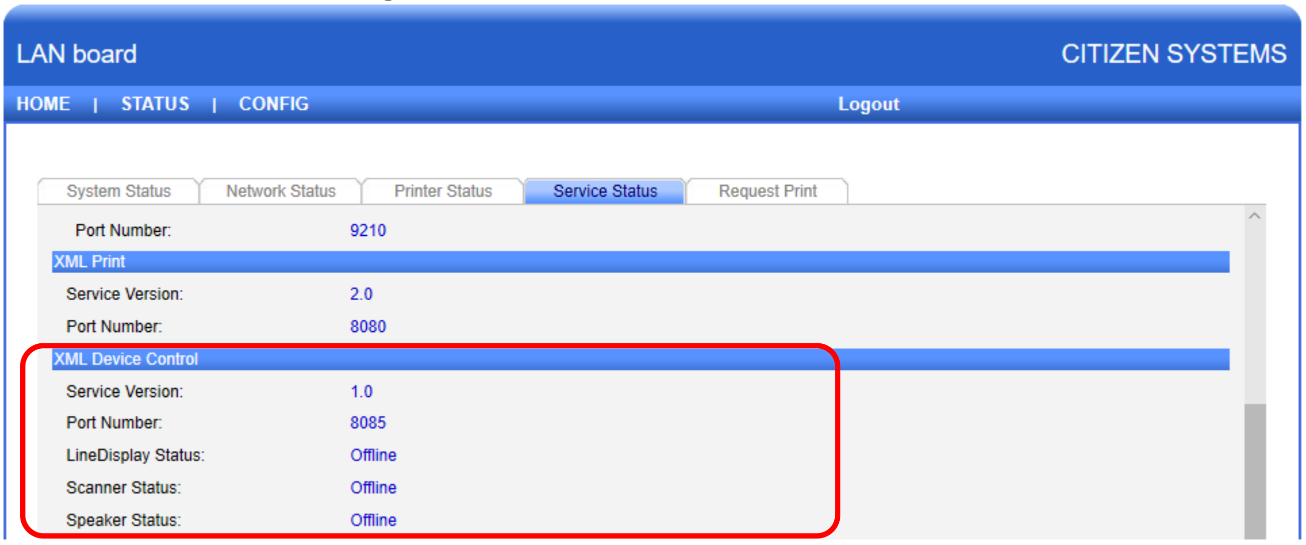
If you press the Test Device button, a test text string will be displayed on the display in accordance with these settings. If the connection with the display cannot be verified, an alert message ("Test failed") will be displayed in the browser.

XML Device Control / Scanner

If you press the Test Device button, the connection with the scanner (USB HID keyboard system) will be verified. If the connection with the scanner cannot be verified, an alert message ("Test failed") will be displayed in the browser.

4.1.2. Service Status Screen

The service version and setting information can be check in the Service Status screen.



Port Number:	9210
XML Print	
Service Version:	2.0
Port Number:	8080
XML Device Control	
Service Version:	1.0
Port Number:	8085
LineDisplay Status:	Offline
Scanner Status:	Offline
Speaker Status:	Offline

5.JavaScript Device Control SDK

Device Control SDK is a library for CITIZEN XML Device Control Service to control a peripheral device using JavaScript on the client side. It enables controlling easily from Web applications using JavaScript.

5.1. Operating Environment

For a Web browser to be supported by this SDK, it needs to support HTML5.

5.2. Programming Guide

5.2.1. Placement of SDK File

Device Control SDK is provided using JavaScript. To use the SDK, place "cxml-device-api.js" on the Web server. If the source code of the provided SDK is changed, correct operation may become no longer possible. Do not change the source code.

5.2.2. Program Configuration

To control a device, write the program in the HTML <script> tag on the Web page placed on the Web server.
The program configuration is as follows.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Device Control SDK Sample</title>
  <script type="text/javascript" src="cxml-device-api.js"></script>
  <script type="text/javascript">
    //Creating Object (Line Display)
    var cxml = new citizen.CXMLDisplay();
    //Setting Response Receive Callback Function
    cxml.OnReceive = function (res) {
      alert(res.ResponseCode);
    };
    //Setting Send Error Callback Function
    cxml.OnError = function (res) {
      alert(res.status);
    };
    //Register device control process
    cxml.MessageID('12345678');
    cxml.ClearDisplay(cxml.AREA_ALL);
    cxml.DisplayText('Hello, World!');
    //Execute sending
    cxml.Send('http://192.168.10.100/xmldevice');
  </script>
</head>

<body>
        ・
        ・
</body>
</html>
```

Integrating SDK

Program text

5.2.3. Creating Object

The control of a device is performed with an object provided for each function. First generate an instance of the object that matches the device to be used.

Line display control object:	citizen.CXMLDisplay
Barcode scanner control object:	citizen.CXMLScanner

5.2.4. Setting Response Receive Callback Function

The control result can be checked in the argument information of the function by setting a callback function in the OnReceive properties of this object.

Item	Description
ResponseCode	Stores the process result.
MessageID	Stores the ID for identifying the response message.
RequestID	Stores the message ID specified when the message was sent.
ErrorCode	Stores the error code when an error occurred.
Description	Stores the explanation when an error occurred.
DeviceInfo	Stores the device status when it is acquired (when using the GetDeviceInfo method).
ScanData	Stores the scan data when it is acquired (when using the StartScanning and GetScanData methods).

An example of setting the response receive callback function is shown below.

```
//Setting Response Receive Callback Function
cxml.OnReceive = function (res) {
  var msg;
  if(res.ResponseCode == 'OK'){
    msg = 'Device control success!□□';
  }
  else{
    msg = 'Device control failure!□□';
    msg += ' Code: ' + res.ErrorCode + '□';
    msg += ' Description: ' + res.Description + '□□';
  }
  msg += ' RequestID: ' + res. RequestID + '□';
  alert(msg);
};
```

Checking for Error Occurrence

Whether or not an error occurred can be checked by checking the value of ResponseCode.

Code	Description
OK	Ended normally.
Rejected	Error occurred.

Checking Error Information

The cause of a result can be checked from the contents of the ErrorCode and Description elements stored when an error occurs.
The error codes used with this service are shown below.

Errors That Occur during Service Processing

Code	Description
RequestInvalid	Request information is invalid
DeviceTimeout	Device timed out

Errors That Occur during Device Processing

ENotConnect	Device connection failed
EConnectNotFound	Unsupported model
EIllegal	Unsupported or invalid parameter
EOffline	Device is offline
ENoExist	No data error
EFailure	Process cannot be executed
ETimeout	Processing timeout
EptrBadFormat	Data format error
EptrTooBig	Data size error

5.2.5. Setting Send Error Callback Function

The error details can be checked in the argument information of the function by setting a callback function in the OnError properties of this object.
The status when an error occurs is stored in status, and the response details are stored in responseText.

An example of setting the send error callback function is shown below.

```
//Setting Send Error Callback Function
cxml.OnError = function (res) {
   var msg = 'Send failure!\n\n';
   msg += ' status: ' + res. status + '\n';
   msg += ' responseText: ' + res. responseText + '\n';
   alert(msg);
};
```

5.2.6. Device Control Process

The device control process can be registered by calling the device control method with this object.
For details on the device control method, refer to "5.4. Details on Device Control Method" in this document.

An example of registering device control is shown below.

```
//Register device control process
//- Set request message ID -
cxml.MessageID('12345678');
//- Device control specification -
cxml.ClearDisplay(cxml.AREA_ALL);
cxml.DisplayText('DisplayText',false);
```

5.2.7. Executing Sending

The control process is started by specifying the URL of XML Display Control and calling the Send function with this object. When the process ends, the set response receive callback function is called and the control result can be acquired. For details on acquiring control results, refer to "5.3.2. Setting Response Receive Callback Function" in this document.

The format of the specified URL is shown below.

http(s)://[IP address of this service]/xmldevice
* In the service version 1.0 and earlier, http://[IP address of this service]:8085

An example of specifying the executing of sending is shown below.

```
//Execute sending (for https)
cxp.Send('https://192.168.10.100/xmldevice');

//Execute sending (for http)
cxp.Send('http://192.168.10.100/xmldevice');

//Execute sending (for http of the service version 1.0 and earlier)
cxml.Send('http://192.168.10.100:8085');
```

5.3. Acquiring Device Status

5.3.1. Device Information Acquisition Method (GetDeviceInfo)

To acquire the device status, use the GetDeviceInfo method to specify a device information acquisition request. The GetDeviceInfo method is implemented in all objects.

Format

GetDeviceInfo (Name, Status)

Parameters

Value	Meaning	Settable range
Name	Device name	Applicable device name If this was not specified, all registered devices are applicable
Status	Status information flag	Add status information when "true" is specified.

5.3.2. Setting Response Receive Callback Function

The device information acquisition result can be checked in the argument information of the function by setting a callback function in the OnReceive properties of this object. An example of acquiring the device status is shown below.

```
//Creating Object
var cxml = new citizen.CXMLDisplay();

//Set response receive callback function (check device information)
cxml.OnReceive = function (res) {
   var msg = 'GetDeviceInfo failure!\n';
   if(res.ResponseCode == 'OK'){
      if((res.DeviceInfo != null) && (0 < res.DeviceInfo.length)) {
         msg = '';
         for (var i=0; i<res.DeviceInfo.length; i++){
            msg += res.DeviceInfo[i].name + ':' +
                        res.DeviceInfo[i].status + '\n';
         }
      }
   }
   alert(msg);
};

//Setting Send Error Callback Function
cxml.OnError = function (res) {
   alert(res.status);
};

//Specify device information acquisition request
cxml.GetDeviceInfo(null, "true");

//Execute sending
cxml.Send('http://192.168.10.100/xmldevice');
```

For the contents of the device information, the device name is stored in name and a status code indicating the device status is stored in status of the elements of the DeviceInfo array. The status codes used with this service are shown below.

Line Display Information (Name="LineDisplay")

Status code	Description
Online	State in which use is possible.
EConnectNotFound	A device not supported by this service is connected.
ENotConnect	A line display is not connected.
EOffline	The device is being used by another service.

Barcode Scanner Information (Name="Scanner")

Status code	Description
Online	State in which use is possible.
EConnectNotFound	A device not supported by this service is connected.
ENotConnect	A barcode scanner is not connected.
EOffline	The device is being used by another service.
StatusScanning	State in which waiting to receive data from the device.

5.4.Device Control Method List

The objects provided with this SDK are shown below.

Function	Object	Description
Line display control	CXMLDisplay	Controls a line display.
Barcode scanner control	CXMLScanner	Controls a barcode scanner.

5.4.1. Line Display Control (CXMLDisplay Object)

The control methods that can be used with the line display control object are shown below.

Function	Method name	Description
Message ID	MessageID	Specify this to enable the sender to identify the message.
Displaying text string	DisplayText	Displays text.
Clearing display	ClearDisplay	Clears the displayed text.
Blink display	BlinkDisplay	Starts blink display.
Setting display	SetDisplayMode	Sets the display mode.
	SetDisplayConfig	Configures various settings.
Setting cursor	SetCursorPosition	Specifies the cursor position.
	MoveCursor	Moves the cursor.
	SetCursorType	Displays the cursor position.
Initializing	InitializeDisplay	Initializes the display.
Setting text string display	SetEncode	Specifies the encoding of the text strings to be sent.
	SetInternationalCharacterset	Specifies the international character.

5.4.2. Barcode Scanner Control (CXMLScanner Object)

The control methods that can be used with the barcode scanner control object are shown below.

Function	Method name	Description
Message ID	MessageID	Specify this to enable the sender to identify the message.
Starting continuous scanning	StartScanning	Starts continuous scanning.
Stopping continuous scanning	StopScanning	Stops continuous scanning.
Getting scan data	GetScanData	Gets the data.
Aborting getting of scan data	AbortScan	Aborts the getting of data.

5.5. Line Display Control Method Details

5.5.1. Message ID (MessageID)

Format

MessageID (ID)

Parameters

The meanings and settable values of the parameters are as follows.

Value	Meaning	Settable range
ID	Request message ID	

Description

This method is used to enable the sender to identify the message.
The specified request message ID is added to the RequestID parameter of the control result. For details on control results, refer to "5.3.2 Setting Response Receive Callback Function" in this document.

Usage example

cxml.**MessageID**('12345678');

5.5.2. Displaying Text (DisplayText)

Format

DisplayText (Data, ReverseFlag)

Parameters

The meanings and settable values of the parameters are as follows.

Value	Meaning	Settable range
Data	Text data	
ReverseFlag	Reverse specification flag	false: Standard true: Reverse When the argument is omitted, it is treated as false.

Description

This method is used to display text from the current cursor position.
Reverse can be specified for the text attribute.

Usage example

cxml.DisplayText("Hello, World!");

5.5.3. Clearing Display (ClearDisplay)

Format

ClearDisplay (DisplayArea)

Parameters

The meanings and settable values of the parameters are as follows.

Value	Meaning	Settable range
DisplayArea	Clear area	AREA_ALL: Entire area AREA_CURSORLINE: Cursor line When the argument is omitted, it is treated as AREA_ALL.

Description

This method clears the displayed text.

Usage example

```
cxml.ClearDisplay(cxml.AREA_ALL);
```

5.5.4. Display Blink (BlinkDisplay)

Format

BlinkDisplay (IntervalBlink)

Parameters

The meanings and settable values of the parameters are as follows.

Value	Meaning	Settable range
IntervalBlink	Blink interval (msec)	From 0

Description

This method causes the entire display screen to blink.
The blink interval (msec) specifies the interval for on and off. If 0 is specified for the blink interval, blinking is disabled.

Usage example

```
cxml.BlinkDisplay(1000)
```

5.5.5. Setting Screen Mode (SetDisplayMode)

Format

SetDisplayMode (DisplayMode)

Parameters

The meanings and settable values of the parameters are as follows.

Value	Meaning	Settable range
DisplayMode	Display mode	OVERWRITE: Overwrite mode VERTICALSCROLL: Vertical scroll mode HORIZONTALSCROLL: Horizontal scroll mode

Description

This method sets the following display modes.

DisplayMode	Overview
Overwrite	Overwrites the text at the cursor position and moves the cursor to the right. (The cursor moves to the bottom left edge for input when it is at the top right edge, and the cursor moves to the top left edge for input when it is at the bottom right edge.)
VerticalScroll	Scrolls the display line of the top edge to the bottom edge by cursor up movement when the cursor is at the top edge (or by left movement when it is at the left edge). Scrolls the display line of the bottom edge to the top edge by cursor down movement when the cursor is at the bottom edge (or by right movement when it is at the right edge).
HorizontalScroll	Scrolls the text leftward in respect to the current cursor line by cursor right movement (or by text input) when the cursor is at the right edge. Scrolls the text rightward in respect to the current cursor line by cursor left movement when the cursor is at the left edge.

Usage example

```
cxml.SetDisplayMode(cxml.MODE_VERTICALSCROLL)
```

5.5.6. Display Settings (SetDisplayConfig)

Format

SetDisplayConfig (Brightness)

Parameters

The meanings and settable values of the parameters are as follows.

Value	Meaning	Settable range
Brightness	Brightness (%)	0 to 100

Description

This method changes the brightness of the display screen.
The higher the numerical value, the brighter the brightness becomes. If 0 is specified, the screen turns off (the display content is retained).
After this is set, blinking of the entire display screen is disabled.

Usage example

```
cxml.SetDisplayConfig(40);
```

5.5.7. Setting Cursor (SetCursorPosition)

Format

SetCursorPosition (x, y)

Parameters

The meanings and settable values of the parameters are as follows.

Value	Meaning	Settable range
x	Digit position	From 1
y	Line position	From 1

Description

This method is used to set the cursor position.
The cursor position is the movement coordinates of the cursor, and specifies the digit position and line position.

Usage example

```
cxml.SetCursorPosition(1, 2);
```

5.5.8. Moving Cursor (MoveCursor)

Format

MoveCursor (dx, dy)

Parameters

The meanings and settable values of the parameters are as follows.

Value	Meaning	Settable range
dx	Rightward/leftward movement amount	-128 to 127
dy	Upward/downward movement amount	-128 to 127

Description

This method is used to move the cursor.
Movement is from the current cursor position. Specify the leftward/rightward movement amount (-: leftward, +: rightward) and upward/downward movement amount (-: upward, +: downward) for the cursor movement amount.

Usage example

```
cxml.MoveCursor(2, 0);
```

5.5.9. Specifying Cursor Type (SetCursorType)

Format

SetCursorType (CursorType)

Parameters

The meanings and settable values of the parameters are as follows.

Element	Meaning	Settable range
CursorType	Cursor type specification	TYPE_NONE: Hide cursor TYPE_UNDERLINE: Display cursor (Omittable element, TYPE_UNDERLINE when omit)

Description

This displays the current cursor position on the display.

Usage example

```
cxml.SetCursorType(cxml.TYPE_UNDERLINE);
```

5.5.10. Initializing Device (InitializeDisplay)

Format

InitializeDisplay ()

Parameters

None

Description

Initializes the device.

Usage example

```
cxml.InitializeDisplay();
```

5.5.11. Specifying Character Encoding (SetEncoding)

Format

SetEncoding (Encode)

Parameters

The meanings and settable values of the parameters are as follows.

Value	Meaning	Settable range
Encode	Character encoding	SingleByteCharacter: 8-bit Character Japanese: Japanese (katakana) SimplifiedChinese: Simplified Chinese Korean: Korean TraditionalChinese: Traditional Chinese None: No conversion

Description

This sets the character encoding for when sending the text data (parameter data / UTF-8 format) in the DisplayText method to the device. The initial value of this setting is "SingleByteCharacter".

Usage example

```
cxml.SetEncoding("Japanese");
cxml.DisplayText("ｺﾝﾆﾁﾜ");
```

5.5.12. Specifying International Character Set (SetInternationalCharacterset)

Format

SetInternationalCharacterset (Characterset)

Parameters

The meanings and settable values of the parameters are as follows.

Value	Meaning	Settable range
Characterset	Specify international character set.	0 to 16

Description

This sets the following international character sets.

Characterset	International Character Set	Characterset	International Character Set
0	America	9	Norway
1	France	10	Denmark II
2	Germany	11	Spain II
3	Britain	12	Latin America
4	Denmark I	13	Korea
5	Sweden	14	Croatia
6	Italy	15	China
7	Spain I	16	Vietnam
8	Japan		

Usage example

```
cxml.SetInternationalCharacterset(8);
cxml.DisplayText("Total:□□1,010");
```

5.6. Barcode Scanner Control Method Details

5.6.1. Message ID (MessageID)

Format

MessageID (ID)

Parameters

The meanings and settable values of the parameters are as follows.

Value	Meaning	Settable range
ID	Request message ID	

Description

This method is used to enable the sender to identify the message.
The specified request message ID is added to the RequestID parameter of the control result. For details on control results, refer to "5.3.2 Setting Response Receive Callback Function" in this document.

Usage example

cxml.**MessageID**('12345678');

5.6.2. Starting Continuous Scanning (StartScanning)

Format

StartScanning (url, callbackResult, callbackStatus, scanOnceTimeout)

Parameters

The meanings and settable values of the parameters are as follows.

Value	Meaning	Settable range
url	URL of XML Display Control	
callbackResult	Callback function to receive start result	
callbackStatus	Callback function to receive status change	
scanOnceTimeout	Timeout of one scan	0 to 60 (Omittable, 10 when omit)

Description

This method is used to continuously acquire scan data from the device.
It waits for the data from the device until the StopScannig method is executed.

If the callback function for receiving the start result was specified, one of the following text strings indicating the result is passed to the first parameter.

Text String	Description
OK	Start process was successful.
ENotConnect	A device is not connected.
StatusScanning	Cannot start because scanning is in progress.
ENotResponse	Connection failed.

If the callback function for receiving the status change was specified, one of the following text strings indicating the result is passed to the first parameter.

Text String	Description
Online	Online
Offfile	Offline
Stoped	Stopped

The scan data can be acquired in the argument information of the function by setting a callback function in the OnReceive properties of this object. The scan data is stored in the ScanData array.

Usage example

```
//Setting response receive callback function (getting scan data)
cxml.OnReceive = function (res) {
   if(res.ResponseCode == 'OK'){
      if((res.ScanData!= null) && (0 < res.ScanData.length)) {
         msg = '';
         for (var i=0; i< res.ScanData.length; i++){
            msg += res.ScanData[i] + '□n';
         }
         alert(msg);
      }
   }
};

//Callback function for start result
var callbackResult = function (result) {
   if (result != 'OK') {
      alert('Start error: ' + result);
   }
};

//Callback function for status change
var callbackStatus = function (status) {
      alert('Scan status: ' + status);
};

//Starting continuous scanning
cxml.StartScanning('http://192.168.10.100:8085', callbackResult, callbackStatus);
```

5.6.3. Stopping Continuous Scanning (StopScanning)

Format

StopScanning ()

Parameters

None

Description

This method stops the continuous scanning processing that was started with the StartScanning method.

Usage example

```
cxml.StopScanning();
```

5.6.4. Getting Scan Data (GetScanData)

Format

GetScanData (Timeout)

Parameters

The meanings and settable values of the parameters are as follows.

Value	Meaning	Settable range
Timeout	Timeout (sec)	0 to 60

Description

This method is used to acquire scan data only once from the device.
It waits for the data from the device for the time specified for the timeout.
The scan data can be acquired in the argument information of the function by setting a callback function in the OnReceive properties of this object. The scan data is stored in the ScanData array.

Usage example

```
//Setting response receive callback function (getting scan data)
cxml.OnReceive = function (res) {
   if(res.ResponseCode == 'OK'){
      if((res.ScanData!= null) && (0 < res.ScanData.length)) {
         msg = '';
         for (var i=0; i< res.ScanData.length; i++){
            msg += res.ScanData[i] + '□n';
         }
         alert(msg);
      }
   }
};

//Getting scan data
cxml.GetScanData(10);
```

5.6.5. Aborting Getting of Scan Data (AbortScan)

Format

AbortScan ()

Parameters

None

Description

This method aborts the process to wait to get the scan data from the device.

Usage example

```
   cxml.AbortScan();
```

5.7. SDK Settings/Other Functions

The following are provided for SDK settings and other functions. They can be used even if a control device cannot be used.

Function name	Description
GetVersionCode	Acquires the SDK version number.
GetVersionName	Acquires the SDK version text string.

5.7.1. Acquiring SDK Version Number (GetVersionCode)

Format

GetVersionCode ()

Return value

Version number: Number

Parameters

None

Description

Acquires the SDK version number as a numerical value (when Ver.1.23: 123).

Usage example

```
var vno = cxml.GetVersionCode();
alert("SDK Version:" + vno/100);
```

5.7.2. Acquiring SDK Version Text String (GetVersionName)

Format

GetVersionName ()

Return value

Version text string: String

Parameters

None

Description

Acquires the SDK version as a text string.

Usage example

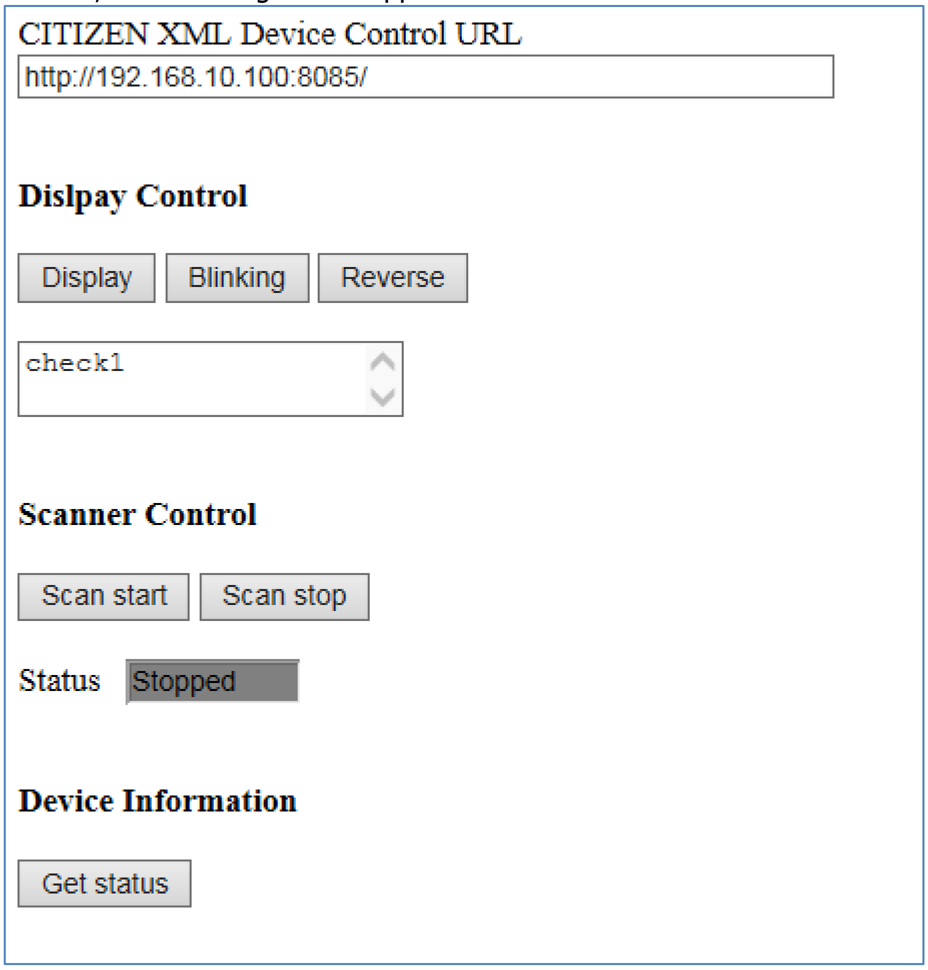
```
var vname = cxml.GetVersionName();
alert("SDK Version:" + vname);
```

6.Sample Programs

The following shows how to use the sample programs of JavaScript Device Control SDK.

6.1.JavaScript Device Control SDK Sample

Start the Web browser and access the URL for the location where the sample is placed. When the sample program is run, the following screen appears.



CITIZEN XML Device Control URL
http://192.168.10.100:8085/

check1

Status Stopped

Set the URL of the device to which to send request messages in the URL field at the top of the screen. When a button in the center of the screen is pressed, the corresponding sample program is run.

The following describes each sample program.

Peripheral Device	Item	Description
Display Control	Display	Displays entered text on the display.
	Blinking	Displays entered text blinking on the display.
	Reverse	Displays entered text in reverse on the display.
Scan Control	Scan start	Starts scanning.
	Scan stop	Stops scanning.
	Status	Stopped : Stopped status Online : Online status Offline : Offline status
Device Information	Get status	Acquires the device status.

CITIZEN XML Device Control Service

JavaScript Device Control SDK Programming Manual

December 28, 2023 For Ver.1.00

CITIZEN SYSTEMS JAPAN CO., LTD.

https://csj.citizen.co.jp/