SATO Printer API
Reference Guide

Edition 3.0_02
August 2020

CONTENTS

SOFTWARE LICENSE AGREEMENT

PLEASE READ THE FOLLOWING TERMS AND CONDITIONS BEFORE USING THIS PRODUCT. BY INSTALLING THE PRODUCT, YOU THEREBY INDICATE YOUR ACCEPTANCE OF THESE TERMS AND CONDITIONS. SHOULD YOU DISAGREE WITH ANY OF THE TERMS OR CONDITIONS LISTED BELOW, PROMPTLY REMOVE ALL FILES RELATED TO THIS PRODUCT FROM YOUR HOST PLATFORM AND RETURN THE PRODUCT TO SATO CORPORATION.

1. You explicitly agree to accept a non-exclusive license to use the Software identified on the distribution media solely for your own customary business or personal purposes. Your local and national laws govern this Agreement.
2. Henceforth, “Software” shall refer to the digitally encoded, machine-readable data and program. The term “Software Product” includes the Software identified on the distribution media, including any accompanying documentation. The term “Distribution Media” refers to any method by which the Software Product is delivered to the end user, including but not limited to Floppy Disks, CD-ROM, Magnetic Tape and On-Line distribution via the Internet. The Software Product is licensed (not sold) to you, and SATO® either owns or licenses from other vendors who own all copyright, trade secret, patent and other proprietary rights in the Software Product.
3. To protect the proprietary rights of SATO Corporation, you agree to maintain the Software Product and other proprietary information concerning the Software Product in strict confidence and to establish reasonable procedures regulating access to and use of the software.
4. You agree not to duplicate or copy the Software except that you may make one backup copy. You agree that any such copy shall contain the same proprietary notices as those appearing on the original.
5. You shall not sub-license, sell, lease, or otherwise transfer all or any portion of the Software Product separate from the printer(s), without the prior written consent of SATO Corporation.
6. You may not modify or prepare derivative works of the Software Product. You may not transmit the Software Product over a network, by telephone, or electronically using any means; or reverse engineer, decompile or disassemble the Software.
7. You may transfer the Software Product with the printer(s), but only if the recipient agrees to accept the terms and conditions of this Agreement. Your license is automatically terminated if you transfer the Software Product and printer(s).
8. This License remains in force until terminated, and may be terminated by agreement between you and SATO Corporation, or by SATO Corporation, if you fail to comply with the terms of this License if such failure is not corrected within thirty (30) days after notice. When this License is terminated, you shall either return to the place you obtained them from, or destroy, the printer and all copies of the Software and documentation.
9. SATO Corporation warrants that for ninety (90) days after delivery, the Software will perform in accordance with specifications published by SATO Corporation, and that the distribution media will be free from defects in material and workmanship. SATO Corporation does not warrant that the Software is free from all bugs, errors and omissions.
10. Your exclusive remedy and the sole liability of SATO Corporation in connection with the Software is replacement of defective distribution media upon their return to SATO Corporation. SATO Corporation will not be liable for any loss or damage caused by delays in furnishing a Software Product or any other performance under this Agreement. SATO Corporation does not warrant that the functions contained in the Software will meet your requirements or that the operation of the Software will be uninterrupted or error free.
11. Use, duplication or disclosure by the Government is subject to restrictions as set forth in the relevant guidelines of your country, such as the Rights in Technical Data and Computer Software clause at FAR 242.227- 7013, subdivision (b) (3) (ii) or subparagraph (c) (1) (ii), as appropriate. Further use, duplication or disclosure is subject to restrictions applicable to restricted rights software as set forth in FAR 52.227-19 (c) (2) or equivalent rules.

THE PARTIES AGREE THAT ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE AND MERCHANTABILITY ARE EXCLUDED.

SATO Corporation reserves the right to make changes and/or improvements in the Software without notice at any time.

IN NO EVENT WILL SATO CORPORATION BE LIABLE FOR LOST PROFITS, LOST DATA, BUSINESS INTERRUPTIONS OR ANY OTHER DIRECT, INDIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF OR INABILITY TO USE THIS PRODUCT, EVEN IF SATO CORPORATION HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR ANY DAMAGES CAUSED BY ABUSE OR MANIPULATION OF THE SOFTWARE. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, SO THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT, AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. NEITHER PARTY SHALL BE BOUND BY ANY STATEMENT NOR REPRESENTATION NOT CONTAINED IN THIS AGREEMENT. NO CHANGE IN THIS AGREEMENT IS EFFECTIVE UNLESS WRITTEN AND SIGNED BY PROPERLY AUTHORIZED REPRESENTATIVES OF EACH PARTY. BY INSTALLING THIS SOFTWARE PRODUCT, YOU AGREE TO ACCEPT THE TERMS AND-CONDITIONS OF THIS AGREEMENT.

COPYRIGHTS

LIMITATION OF LIABILITY

SATO Corporation and its subsidiaries in Japan, the U.S and other countries make no representations or warranties of any kind regarding this material, including, but not limited to, implied warranties of merchantability and fitness for a particular purpose. SATO Corporation shall not be held responsible for errors contained herein or any omissions from this material or for any damages, whether direct, indirect, incidental or consequential, in connection with the furnishing, distribution, performance or use of this material.

Specifications and contents of this document are subject to change without notice.

TRADEMARKS

SATO is a registered trademark of SATO Holdings Corporation and its subsidiaries in Japan, the U.S and other countries.

Microsoft, Visual Studio, Visual C++, .NET Framework and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

All other trademarks are the property of their respective owners.

SOFTWARE UPDATING DISCLAIMER

While all efforts have been taken to ensure accuracy and currency of the information contained herein, there are instances where the contents of this document may be outdated. In that case, proceed to your local SATO regional website (https://www.sato-global.com/drivers/redirect.html) to check whether an updated document has been made available for your reference.

CONTACT INFORMATION

Access the following site and select the region/country nearest to you.

https://www.sato-global.com/about/locations.html

OVERVIEW

The SATO Printer API was developed to simplify the communication between .NET applications and SATO printers.

Software developers can use this API to send printer commands to printer as well as to receive response from printer if there is any.

Also, SATO Printer API can be used for getting SATO printer list from printer driver, check number of jobs in printer spooler or printer information from printer driver.

SYSTEM REQUIREMENTS

SOFTWARE DEVELOPMENT ENVIRONMENT

- Microsoft Visual Studio 2013 (or higher version)
- Microsoft .NET Framework 4.0 (or higher version)
- Visual C++ Redistributable Package for Visual Studio 2013
    - Available on Microsoft website (http://www.microsoft.com/en-us/download/details.aspx?id=40784). Please download and install the correct platform (vcredist_x86.exe or vcredist_x64.exe)
- Platform Supported: 32bit or 64bit application
- Platform Target: x86 or x64 (Project properties => Build)

CLIENT COMPUTER

Custom application developed with SATO Printer API must be distributed with the DLL files of SATO Printer API. And following components must be installed on the client computer.

- Microsoft .NET Framework 4.0 (or higher version)
- Visual C++ Redistributable Package for Visual Studio 2013

DOWNLOAD

Download the latest version of the SATO Printer API from your local SATO regional website.

https://www.sato-global.com/drivers/redirect.html

SETUP

Execute the SATO Printer API setup package and follow the guidance of the setup wizard. By default, the SATO Printer API files will be installed in below folder.

x86 environment:

C:¥Program Files¥SATO¥SATO Printer API

x64 environment:

C:¥Program Files (x86)¥SATO¥SATO Printer API

Copy SATOPrinterAPI assembly (SATOPrinterAPI.dll) into the project folder. Add reference assembly into project references.

USING THE SATO PRINTER API

PRINTER COMMUNICATION FUNCTIONS

INITIATE THE PRINTER

Initialize the API to use the printer functions.

Class constructor syntax

```
Printer()
```

Parameters

Not Applicable

Return Values

Not Applicable

Example (C#)

```
using SATOPrinterAPI;

Printer SATOPrinter = new Printer();
```

(Below example will use SATOPrinter as initialized class variable.)

DISCOVERY PORTS

Gets a list of the port available on the computer. For COM port and LPT port, this returns all available ports based on Windows device management regardless of printer connection. For USB, it will returns connected SATO printer ports.

As for TCP/IP, API will discover SATO printers on the same broadcast segment through available TCP/IP interfaces of PC then return a list of discovered SATO printer information. The information includes printer IP address and MAC address.

Method syntax

COM port:

```
GetCOMList()
```

LPT port:

```
GetLPTList()
```

USB port:

```
GetUSBList()
```

```
TCPIP:
```

```
GetTCPIPList()
```

Parameters

Not Applicable

Return Values

COM port: List of string

LPT port: List of string

USB port: List of USBInfo class

TCPIP : List of TCPIPInfo class

Supported socket server

Yes

Example (C#)

COM port:

```
List<string> COMPorts = SATOPrinter.GetCOMList();
```

LPT port:

```
List<string> LPTPorts = SATOPrinter.GetLPTList();
```

USB port:

```
List<Printer.USBInfo> USBPorts = SATOPrinter.GetUSBList();
```

TCPIP:

```
List<Printer.TCPIPInfo> TCPIPs = SATOPrinter.GetTCPIPList();
```

SET INTERFACE TYPE

Before sending data to the printer, require to specify interface type of the printer connection. Below properties allow setting interface type for COM, USB, LPT or TCP/IP.

Properties syntax

```
.Interface
```

Supported socket server

Yes

Example (C#)

COM port:

```
SATOPrinter.Interface = Printer.InterfaceType.COM;
```

LPT port:

```
SATOPrinter.Interface = Printer.InterfaceType.LPT;
```

USB port:

```
SATOPrinter.Interface = Printer.InterfaceType.USB;
```

TCP/IP port:

```
SATOPrinter.Interface = Printer.InterfaceType.TCPIP;
```

SET INTERFACE PORT

Before sending data to the printer, require to specify the interface port of the printer connection. Below properties allow you to set COM, LPT or USB port or IP address and IP Port of printer you want to communicate.

Properties syntax

COM port:

```
.COMPort
```

LPT port:

```
.LPTPort
```

USB port:

```
.USBPortID
```

TCP/IP port:

```
.TCPIPAddress
```

```
.TCPIPPort
```

Supported socket server

Yes

Example (C#)

COM port:

```
SATOPrinter.COMPort = “COM1”;
```

LPT port:

```
SATOPrinter.LPTPort = “LPT1”;
```

USB port:

```
SATOPrinter.USBPortID =
“¥¥?¥usb#vid_0828&pid_0122#6&1161632e&0&1#{a5dcbf10-6530-11d2-901f-00c04fb
951ed}”;
```

TCP/IP port:

```
SATOPrinter.TCPIPAddress = "10.25.7.204";
```

```
SATOPrinter.TCPIPPort = "1024";
```

SET SERIAL COM PORT SETTINGS

When using serial COM port for connection, it can change baud rate and parameters. Default baud rate is 9600 and parameters 8-N-1.

Properties class syntax

COMSetting

Supported socket server

Yes

Example (C#)

```
SATOPrinter.COMSetting.Baudrate = 115200
SATOPrinter.COMSetting.Parameters = “8-N-1”
```

SET TIMEOUT

This timeout variable applies to connection timeout, send data timeout and receive data timeout. The default timeout is 2500 (2.5 seconds), in milliseconds format.

Properties syntax

```
.Timeout
```

Supported socket server

Yes

Example (C#)

```
SATOPrinter.Timeout = 5000; //(5 seconds timeout)
```

SET PERMANENT CONNECTION

The permanent connection flag allows printer connection open at all time. The default value is false. (SATO Printer only allow for one active connection, please use it wisely)

Properties syntax

```
.PermanentConnect
```

Supported socket server

No

Example (C#)

```
SATOPrinter.PermanentConnect = true;
```

CONNECT

This method will start to raise permanent connection to the printer if “PermanentConnect” flag set to true otherwise, it will connect and disconnect immediately. Ensure to set interface port, interface type and PermanentConnect flag before using this method.

Method syntax

```
Connect()
```

Parameters

Not Applicable

Return Values

Not Applicable

Supported socket server

No

Example (C#)

```
SATOPrinter.Connect();
```

SEND SINGLE DATA

Send method will send a single command data to the printer. This method wills automatic raise connection to the printer and sends command. If “PermanentConnect” flag not set or set to false, it will automatic disconnect the connection. Ensure to set interface port and interface type before using this method.

Method syntax

```
Send(byte[] data)
```

Parameters

Byte Array data

Return Values

Not Applicable

Supported socket server

Yes

Example (C#)

```
SATOPrinter.Send(<Byte Array SBPL command>);
```

SEND SINGLE QUERY AND RECEIVE SINGLE REPLY

Query method will send a single command to the printer and return received a single reply from the printer. This method will connect to the printer and send command data, disconnect connection from the printer when a reply data is received. Ensure to set interface port and interface type before use this method. (Note: permanent connection should not use this method)

Method syntax

```
Query(byte[] data)
```

Parameters

Byte Array data

Return Values

Byte Array data

Supported socket server

Yes

Example (C#)

```
Byte[] Printer_Reply = SATOPrinter.Query(<Byte Array SBPL Query command>);
```

RECEIVE DATA FROM PRINTER AS PERMANENT CONNECTION

To process reply data from printer while using the permanent connection, need to use event handler. Below example in C# shows for initializing the event method:-

Example (C#)

Initialize event method handler:

```
SATOPrinter.ByteAvailable += new
EventHandler<Printer.ByteAvailableEventArgs>(ReadMsg);
```

Create an event method:

```
private void ReadMsg(object sender, Printer.ByteAvailableEventArgs e)
{
    byte[] rawData = e.Data;
    string printerReplied = Utils.ByteArrayToString(b);
}
```

DISCONNECTION

This method to disconnect the permanent connection.

Method syntax

```
Disconnect()
```

Parameters

Not Applicable

Return Values

Not Applicable

Supported socket server

No

Example (C#)

```
SATOPrinter.Disconnect();
```

GET PRINTER STATUS

This method to get printer status and return a class values. Ensure to set interface port and interface type before use this method.

Method syntax

```
GetPrinterStatus()
```

Parameters

Not Applicable

Return Values

Printer Status Class

Supported socket server

Yes

Example (C#)

```
Printer.Status pStatus = SATOPrinter.GetPrinterStatus();
```

TEST PRINT

This method to instruct the printer to print a test label. Ensure to set interface port and interface type before use this method.

Method syntax

```
TestPrint()
```

Parameters

Not Applicable

Return Values

Not Applicable

Supported socket server

Yes

Example (C#)

```
SATOPrinter.TestPrint();
```

CLEAR PRINTER BUFFER

This method to empty/clear printer buffer (labels data in printer memory). Ensure to set interface port and interface type before use this method.

Method syntax

```
ClearBuffer()
```

Parameters

Not Applicable

Return Values

Not Applicable

Supported socket server

Yes

Example (C#)

```
SATOPrinter.ClearBuffer();
```

REPRINT LABEL

This method to instruct the printer to reprint the last printed label. Ensure to set interface port and interface type before use this method.

Method syntax

```
Reprint()
```

Parameters

Not Applicable

Return Values

Not Applicable

Supported socket server

Yes

Example (C#)

```
SATOPrinter.Reprint();
```

PRINTER DRIVER FUNCTIONS

INITIATE THE DRIVER

Initialize the API to use the printer driver functions.

Class constructor syntax

```
Driver()
```

Parameters

Not Applicable

Return Values

Not Applicable

Example (C#)

```
Driver SATODriver = new Driver();
```

(Below example will use SATODriver as initialized class variable.)

GET LIST OF PRINTER DRIVER

Gets list of printer driver on the computer. This will reply SATO printer driver only.

Method syntax

```
GetDriverList()
```

Parameters

Not Applicable

Return Values

List of Driver Info

Supported socket server

Yes

Example (C#)

```
List<Driver.Info> Drivers = SATODriver.GetDriverList();
```

GET LIST OF DRIVER PORT NAME

Gets list of printer driver port name on the computer.

Method syntax

```
GetPortNames()
```

Parameters

Not Applicable

Return Values

List of string

Supported socket server

Yes

Example (C#)

```
List<string> Ports = SATODriver.GetPortNames();
```

GET DRIVER INFO

Gets driver information by the driver name

Method syntax

```
GetDriverInfo(string DriverName)
```

Parameters

String of Driver Name

Return Values

Driver Info class

Supported socket server

Yes

Example (C#)

```
Driver.Info dInfo = SATODriver.GetDriverInfo("SATO CL412e");
```

SET DRIVER INFO

Set/Update driver information

Method syntax

`SetDriverInfo(DriverInfo driver)`

Parameters

Driver Info class

Return Values

Not Applicable

Supported socket server

Yes

Example (C#)

```
Driver.Info dInfo = SATODriver.GetDriverInfo("SATO CL412e");
dInfo.Bidirectional = false;
SATODriver.SetDriverInfo(dInfo);
```

GET NUMBER OF JOBS IN SPOOLER FROM PRINTER DRIVER

Gets number of jobs in printer driver spooler by the driver name.

Method syntax

`GetSpoolerPrintJobsNumber(string DriverName)`

Parameters

String of Driver Name

Return Values

Integer

Supported socket server

Yes

Example (C#)

```
Int NoOfPrintJob = SATODriver.GetSpoolerPrintJobsNumber("SATO CL412e");
```

CLEAR SPOOLER PRINT JOBS

Empty/Clear print jobs in printer driver spooler by the driver name.

Method syntax

```
ClearSpoolerPrintJobs(string DriverName)
```

Parameters

String of Driver Name

Return Values

Not Applicable

Supported socket server

Yes

Example (C#)

```
SATODriver.ClearSpoolerPrintJobs("SATO CL412e");
```

GET PORT INFO FROM DRIVER

Get port information by the driver name.

Method syntax

```
GetPortInfoByDriverName(string DriverName)
```

Parameters

String of Driver Name

Return Values

Port Info Class

Supported socket server

Yes

Example (C#)

```
Driver.PortInfo pInfos = SATODriver.GetPortInfoByDriverName("SATO CL412e");
```

GET PORT INFO FROM PORT NAME

Get port information by the port name.

Method syntax

```
GetPortInfoByName(string PortName)
```

Parameters

String of Port Name

Return Values

Port Info Class

Supported socket server

Yes

Example (C#)

```
Driver.PortInfo pInfos = SATODriver.GetPortInfoByName("COM1");
```

GET DRIVER VERSION

Get printer driver company and printer driver version by driver name.

Method syntax

```
GetVersion(string DriverName)
```

Parameters

String of Driver Name

Return Values

String of driver version

Supported socket server

Yes

Format

Driver Company|Driver Version

(example: SEAGULL SCIENTIFIC, INC.|7.4.2.0

SATO CORPORATION|7.1.02.10208)

Example (C#)

```
string driverVersion = SATODriver.GetVersion("SATO CL412e");
```

SEND BYTE ARRAY RAW DATA THROUGH DRIVER

Send a raw byte array data to the printer driver.

Method syntax

```
SendRawData(string DriverName, byte[] Data)
```

Parameters

String of Driver Name

Byte Array of data

Return Values

Bool of send result

Supported socket server

Yes

Example (C#)

```
bool sentResult = SATODriver.SendRawData("SATO CL4NX 203dpi",<Byte array of SPBL Command>);
```

UTILITIES FUNCTIONS

CONVERT GRAPHIC TO SBPL

This static method to convert image file into SBPL graphic commands (ESC+G)

Method syntax

```
ConvertGraphicToSBPL(string GraphicFilePath)
```

Parameters

GraphicFilePath as string

Return Values

SBPL commands as string

Supported socket server

Yes

Example (C#)

```
string graphicCommands = Utils.ConvertGraphicToSBPL("C:/Pictures/my.png");
```

Or

```
string graphicCommands = Utils.ConvertGraphicToSBPL(new
Uri("https://abc.com/my.png"));
```

COMMAND DATA REPLACE

This static method to uses predesigned template file with variable data to replaced desired value. Mainly use for PRN template file and it created by Nice Label Software.

Method syntax

```
CommandDataReplace(string CommandFilePath, Dictionary<string,string>
VariablesValue)
```

Parameters

CommandFilePath as string

VariablesValue list as Dictionary

Return Values

Replaced variables command as string

Supported socket server

Yes

Example (C#)

```
Dictionary<string,string> variables = new Dictionary<string,string>();
```

```
variables.Add(“@Var1@”,”SATO Printer”);
variables.Add(“@Var2@”,”$100.00”);
variables.Add(“@Barcode1@”,”123456789”);SATO
string PRNCommands = Utils.CommandDataReplace (“C:/data/my.prn”, variables);
```

Or

```
string PRNCommands = Utils.CommandDataReplace (new
Uri(“https://abc.com/my.prn”), variables);
```

CONVERT STRING TO BYTE ARRAY

This static method to convert string data to byte array data

Method syntax

```
StringToByteArray(string Data)
```

Parameters

String data

Return Values

Byte array data

Supported socket server

No

Example (C#)

```
byte[] byteData = Utils.StringToByteArray(<string data>);
```

CONVERT BYTE ARRAY TO STRING

This static method to convert byte array data string data

Method syntax

```
ByteArrayToString(byte[] Data)
```

Parameters

Byte array data

Return Values

String data

Supported socket server

No

Example (C#)

```
string strData = Utils.ByteArrayToString(<Byte array data>);
```

WEB SOCKET FUNCTIONS

INITIATE THE SOCKET SERVER

Initialize the web socket server. (By enable web socket server, it allow web application accessing SATO Printer API functions like SATO printer TCP/IP discovery, driver settings change etc.) It defaults 8055 port for the socket server and you may change it if the port is been occupied in the PC.

Class constructor syntax

```
SocketServer()
```

Parameters

Port as Integer (default 8055)

Web Socket URL for client to connect

Non-Secure: ws://localhost:8055/SATOPrinterAPI

Secure: wss://localhost:8055/SATOPrinterAPI (required to enable secure socket server)

Example (C#)

```
using SATOPrinterAPI;

SocketServer SATOServer = new SocketServer(); //Use default 8055 port
```

Or

```
SocketServer SATOServer = new SocketServer(8081); //Change default port to 8081
```

ENABLE SECURE SOCKET SERVER

If the Web application using SSL secure channel like HTTPS, it better to enable secure socket server for the data communication. Follow the instruction link below to export the localhost certificate for secure socket server. (https://www.sslsupportdesk.com/export-ssl-certificate-private-key-pfx-using-mmc-windows/)

Properties syntax

```
.Certificate
```

Parameters

Not Applicable

Return Values

Not Applicable

Example (C#)

```
SATOServer.Certificate = new
System.Security.Cryptography.X509Certificates.X509Certificate2
(@"C:/cert/localhost_cert.pfx", "0310"); //(certificate file path and
password)
```

START SOCKET SERVER

Starting socket server it may allow client to connect and consume API functions.

For secure socket server, please ensure to define a certificate details before start the socket server.

Method syntax

```
Start()
```

Parameters

Not Applicable

Return Values

Not Applicable

Example (C#)

```
SATOServer.Start();
```

STOP SOCKET SERVER

Stop the socket server by calling this method

Method syntax

```
Stop()
```

Parameters

Not Applicable

Return Values

Not Applicable

Example (C#)

```
SATOServer.Stop();
```

EXAMPLE OF CALLING API FUNCTION IN WEB SOCKET CLIENT

It work similar from the API called, JSON input data consist of properties, method and parameters. It supported string input or byte array input. By using string input, those byte array data required to encode as base64 string. If API return value is Class form or List form, it will be JSON, other than that it will return as string (Byte array return will be encoded as base64 string). Please refer below example to understand further.

Example 1:

Printer API function: Get Printer TCP/IP List

JSON input:

{
"Method":"Printer.GetTCPIPList"
}

Output:

[{"Name":"CL4NX 203dpi","MacAddress":"00:19:98:12:30:DD","IPAddress":"10.65.2.178","Details":"SATO LAN Interface"},
{"Name":"CL4NX 203dpi","MacAddress":"78:A5:04:93:D6:57","IPAddress":"10.65.2.75","Details":"SATO LAN Interface"}]

Example 2:

Printer API function: Query commands to printer

JSON input: ([ENQ] control char in base64 encoded string is [BQ==])

{
"Interface":"TCPIP",
"TCPIPAddress":"10.65.2.75",
"TCPIPPort":"9100",
"Method":"Printer.Query",
"Parameters":{
"Data":"BQ=="}
}

Output:

AAAAHAUCICBBMDAwMDAwICAgICAgICAgICAgICAgIAM=

Example 3:

Printer API function: Instruct printer to print a test print label

JSON input:

{

"Interface":"TCPIP",

"TCPIPAddress":"10.65.2.75",

"TCPIPPort":"9100",

"Method":"Printer.TestPrint"

}

Output:

{"Result":"Executed"}

Example 4:

Driver API function: Get list of printer driver on the computer

JSON input:

{

“Method":"Driver.GetDriverList"

}

Output:

[{"PrinterModel":"SATO PW208NX","DriverName":"SATO PW208NX","PortName":"COM1:","Online":false,"Default":false,"Bidirectional":true},{"PrinterModel":"SATO PW208","DriverName":"SATO PW208","PortName":"FILE:","Online":true,"Default":false,"Bidirectional":true}]

Example 5:

Driver API function: Get number of print jobs in the printer driver spooler by the driver name.

JSON input:

{

"Method":"Driver.GetSpoolerPrintJobsNumber",

"Parameters":{

"DriverName":"SATO PW208"}

}

Output:

0

Example 6:

Utils API function: Commands data replace from PRN file

JSON input:

{

"Method":"Utils.CommandDataReplace",

"Parameters":{

"CommandFilePath":"file:///C:/data/my.prn",

"VariablesValue":{

"@var1@":"SATO Printer",

"@var2@":"$100.00",

"@barcode1@":"123456789"}}

}

Output:

GyMjAcK8BgQAAA ……..

CLASSES

ENUM FOR PRINTER.INTERFACETYPE

```
TCPIP = 0, USB = 1, COM = 2, LPT = 3
```

CLASS FOR PRINTER.USBINFO

```
string Name
string PortID
```

CLASS FOR PRINTER.TCPIPINFO

```
string Name
string MACAddress
string IPAddress
string Details
```

CLASS FOR PRINTER.STATUS

```
int Buffer
string Code
string Description
bool IsOnline
bool IsError
string JobName
string JobID
string Raw
string State
```

CLASS FOR DRIVER.INFO

```
string PrinterModel
string DriverName
string PortName
bool Online
bool Default
bool Bidirectional
```

CLASS FOR DRIVER.PORTINFO

```
string IPAddress
string Port
string Name
Printer.InterfaceType Interface
```

SATO