EPSON
EXCEED YOUR VISION

TM-DT Series
Peripheral Device Control Guide

M00109703
Rev.D

Precautions

- Unauthorized duplication, copying, reproduction, or modification of any part or all of this document is strictly prohibited.
- Contents of this manual are subject to change without prior notice. Contact us directly for the most recent information.
- Every effort is made to ensure that the contents of this manual are without error. Please contact us if any errors or other issues are found.
- The previous statement notwithstanding, we will not be liable for any negative impact as a result of use.
- Epson shall not be liable for any damages caused as a result of using this product incorrectly, failing to comply with the content of this document, or having repair or modifications performed by third parties other than Epson or those specified by Epson.
- Epson shall not be liable for any issues as a result of installing optional parts or consumables that are not genuine Epson parts or parts certified by Epson.

Trademarks

Safety Precautions

Meaning of Symbols

The following symbols are used in this manual. Make sure to understand the meaning of these symbols before using the product.

⚠	Describes usage precautions that must be observed. Incorrect handling due to the disregard of this information may result in product failure or incorrect operation.
✎	Describes additional explanation or other useful information.

Usage Limitations

Please use our products in environments and systems designed with consideration to safety and disaster recovery such as fail-safe configurations and redundant designs, for example, if this product is used in applications in which a high level of reliability and safety in functionality and precision is required such as in aircraft, trains, ships, automobiles and other transportation-related applications or in crime prevention equipment and safety equipment.
This product is not intended for use in applications that require extremely high levels of reliability and safety such as in aerospace equipment, trunk-line communications equipment, nuclear power control equipment, and medical equipment. Consider your usage environment and requirements carefully before using this product in such applications.

About this Manual

Purpose of this Manual

This manual describes the methods for controlling the peripheral devices by a TM-DT series printer.

Organization of this Manual

This manual is organized into the following chapters.

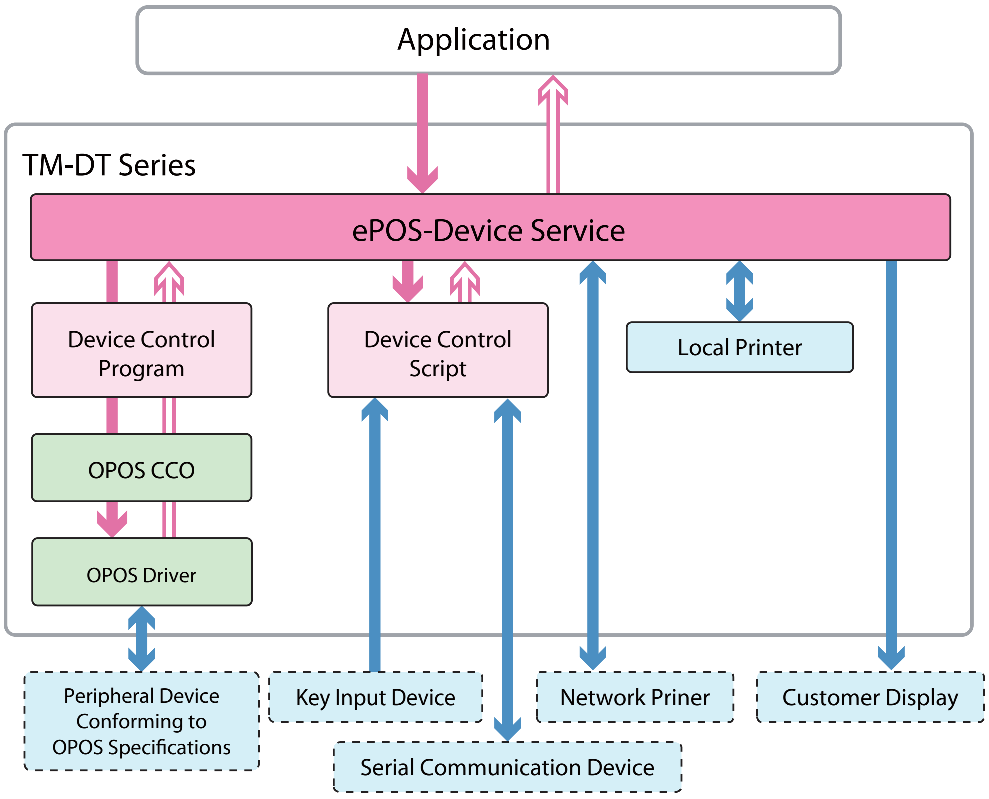
Chapter 1	Overview
Chapter 2	Controlling by a Device Control Program
Chapter 3	Developing a Device Control Program
Chapter 4	Controlling by a Device Control Script
Chapter 5	Developing a Device Control Script

Contents

Overview

In addition to the peripheral devices by Epson, you can control various peripheral devices from the TM-DT series printers (TM-DT series) using the device control programs or device control scripts developed uniquely by Epson.

For details on the development of applications that control the TM-DT series and each peripheral device, refer to the user's manuals of Epson ePOS SDK and ePOS-Device XML.



What is a device control program?

A device control program is an execution file that sends a command from the ePOS-Device Service to the peripheral devices, and returns the execution results.

For TM-T70II-DT, TM-T88V-DT, or TM-H6000IV-DT, the device control program is available with version 4.0 or later TM-DT software.

In the categories described below, peripheral devices having a driver (OPOS driver) that runs in combination with OPOS Common Control Object (OPOS CCO) 1.14.001 can be controlled.

- MSRs
- POS keyboards
- Barcode scanners

For details on the control method, refer to Controlling by a Device Control Program.

In the TM-DT series, APIs are also available for the development of the device control program.
By developing a program conforming to the communication protocol of the peripheral devices, not only the peripheral devices conforming to the OPOS specifications, but any peripheral device can be controlled.
For details on the development method, refer to Developing a Device Control Program.

What is a device control script?

A device control script is a JavaScript file that sends a command from the ePOS-Device Service to the peripheral devices, and returns the execution results.
It is loaded to the TM-DT series, and can control supported key input devices and serial communication devices.
For details on the control method, refer to Controlling by a Device Control Script.

In the TM-DT series, APIs are also available for the development of the device control script.
By developing a script file conforming to the communication protocol of the peripheral devices, any peripheral device can be controlled.
For details on the development method, refer to Developing a Device Control Script.

Supported Printers and Peripheral Devices

Supported Printers

❏ TM-T70II-DT

❏ TM-T70II-DT2

❏ TM-T88V-DT

❏ TM-T88VI-DT2

❏ TM-H6000IV-DT

Controllable Peripheral Devices

The following peripheral devices can be controlled.

Epson products

The following Epson peripheral devices can be controlled.
For details on the products and control methods that can be used, refer to the Technical Reference Guide of the TM-DT series.

❏ Printer

❏ Customer display

Peripheral devices controlled by a device control program

In the categories described below, peripheral devices having a driver that runs in combination with OPOS CCO 1.14.001 can be controlled.

- MSRs
- POS keyboards
- Barcode scanners

Peripheral devices controlled by a device control script

❏ Key input devices

- MSRs (Hitachi-Omron V3TU-FK)
- Keyboards (Standard HID)
- Barcode scanners (Standard HID)

❏ Serial communication devices

- Serial communication devices
- USB devices that can be controlled in a similar manner as a serial communication device

Install a dedicated driver for using a USB device that can be controlled in a similar manner as a serial communication device.
You may not be able to use some drivers depending on their specifications.

Environmental Settings

This section describes the default settings necessary for controlling peripheral devices in the TM-DT series.

Windows Settings

Make the default settings for the Windows OS loaded in the TM-DT series.
Refer to the Technical Reference Guide of the TM-DT series to make the necessary default settings.

EPSON TMNet WebConfig

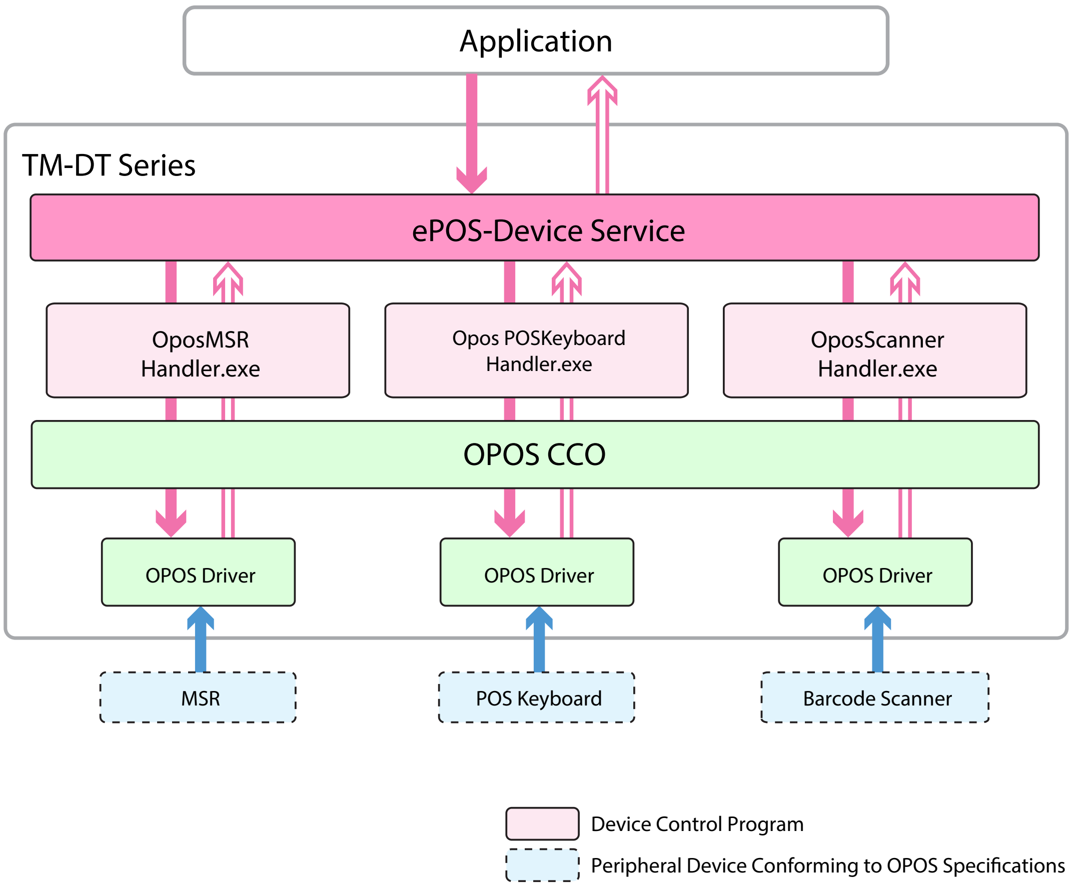
You must register the peripheral devices to be used in the TM-DT series.
You must also set the device control program and device control script in each peripheral device.
Use the TM-DT series EPSON TMNet WebConfig for performing the registration and making settings.

Refer to the Technical Reference Guide of the TM-DT series for EPSON TMNet WebConfig.

Controlling by a Device Control Program

This section describes the method of controlling peripheral devices conforming to the OPOS specifications by using the device control program loaded in the TM-DT series.

In the TM-DT series, a device control program is available for each category of peripheral devices.
Control the peripheral devices conforming to the OPOS specifications by using OPOS CCO and an OPOS driver for the peripheral devices.



Environment Construction

This section describes the procedure of installing OPOS CCO and the OPOS driver in the TM-DT series.
Use OPOS_CCOs_1.14.001.msi provided with the Epson ePOS SDK package as the OPOS CCO.

When evaluating or controlling the peripheral devices, the OPOS log may be checked. So, activate the remote desktop (remote access) function and similar as necessary.

1 Log on to the TM-DT series.

Refer to the Technical Reference Guide of the TM-DT series.

2 Install the OPOS driver and check the logical device name.

Refer to the manual provided with the OPOS driver.
When controlling multiple peripheral devices, install the OPOS driver for each peripheral device to control, and check the logical device name.

3 Run OPOS_CCOs_1.14.001.msi and install OPOS CCO.

Follow the instructions on the installation screen of OPOS_CCOs_1.14.001.msi.

Be sure to install OPOS CCO after installing the OPOS driver.
If the reverse procedure is followed, the OPOS CCO on the OPOS driver will be overwritten, and it may no longer be controlled from the TM-DT series.
Even when you are adding the peripheral devices to be controlled after constructing the environment, first install the OPOS driver, and then install OPOS CCO again by overwriting.

4 Use the test tool provided with the OPOS driver to ensure that the peripheral devices are running normally.

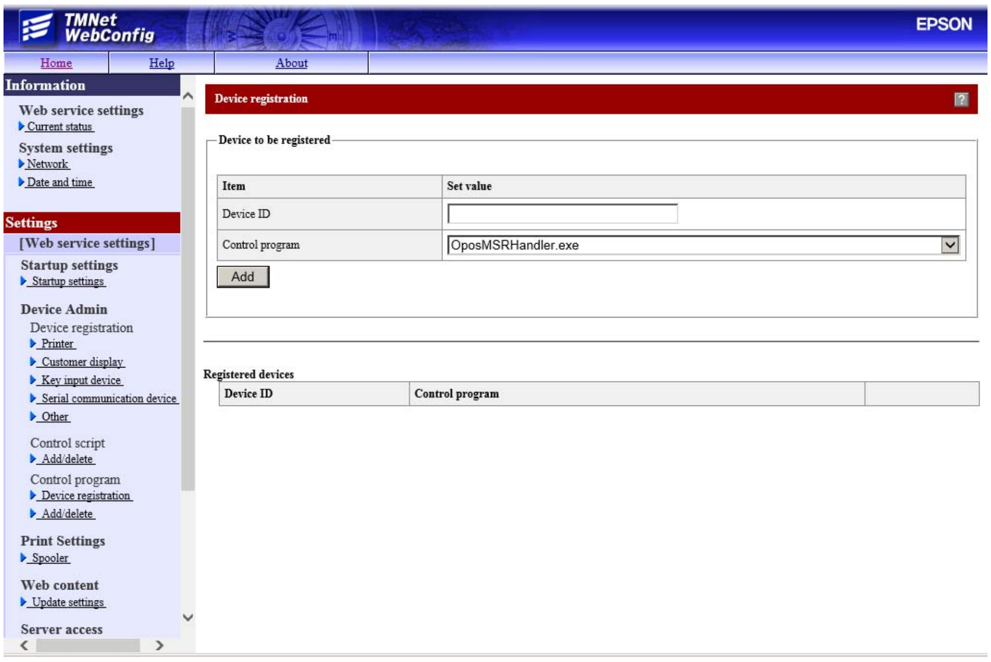
Refer to the manual provided with the OPOS driver, and implement the operations when necessary.

Device Registration

This section describes the procedure of registering the peripheral devices to be used in the TM-DT series.

1 Start EPSON TMNet WebConfig.
Refer to the Technical Reference Guide of the TM-DT series.

2 Select **Web service settings** - **Control program** - **Device registration** to open the Device registration screen.



Device to be registered

Item	Set value
Device ID	
Control program	OposMSRHandler.exe

Registered devices

Device ID	Control program	

3 Configure the following settings and click **Add**.

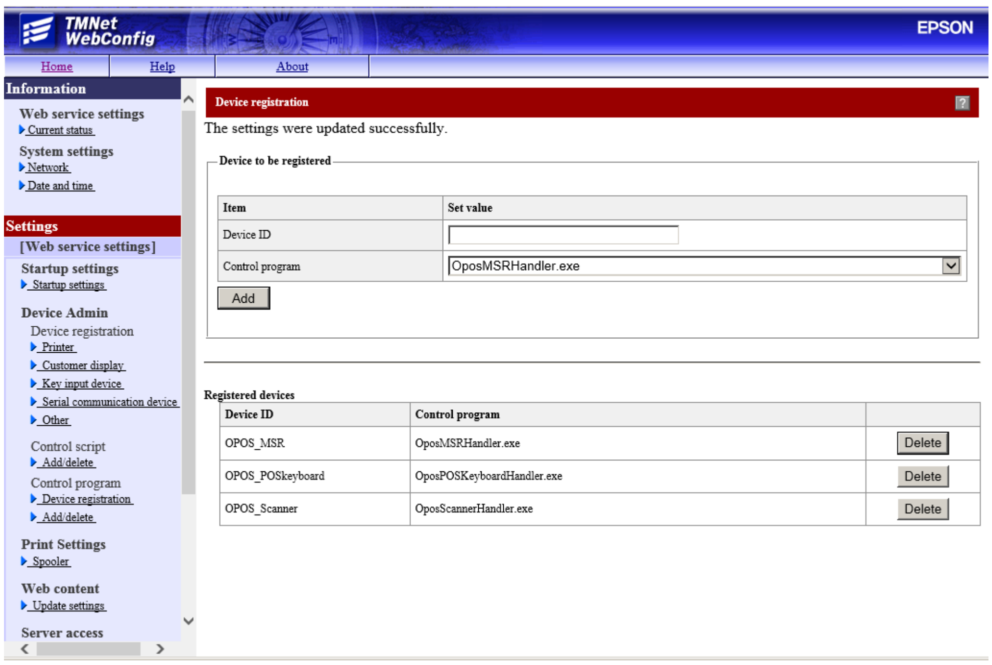
Configure the settings for each peripheral device to be controlled.

In Device ID, specify the logical device name confirmed at the time of installation of the OPOS driver.

Peripheral device	Item to set	Setting value
MSR	Device ID	Logical device name
	Control program	OposMSRHandler.exe
POS keyboard	Device ID	Logical device name
	Control program	OposPOSKeyboardHandler.exe
Barcode scanner	Device ID	Logical device name
	Control program	OposScannerHandler.exe

4 When registration is complete, the registered information is displayed in **Registered devices**.

Click **Delete** to delete the corresponding registered information.



The settings were updated successfully.

Device to be registered

Item	Set value
Device ID	
Control program	OposMSRHandler.exe

Registered devices

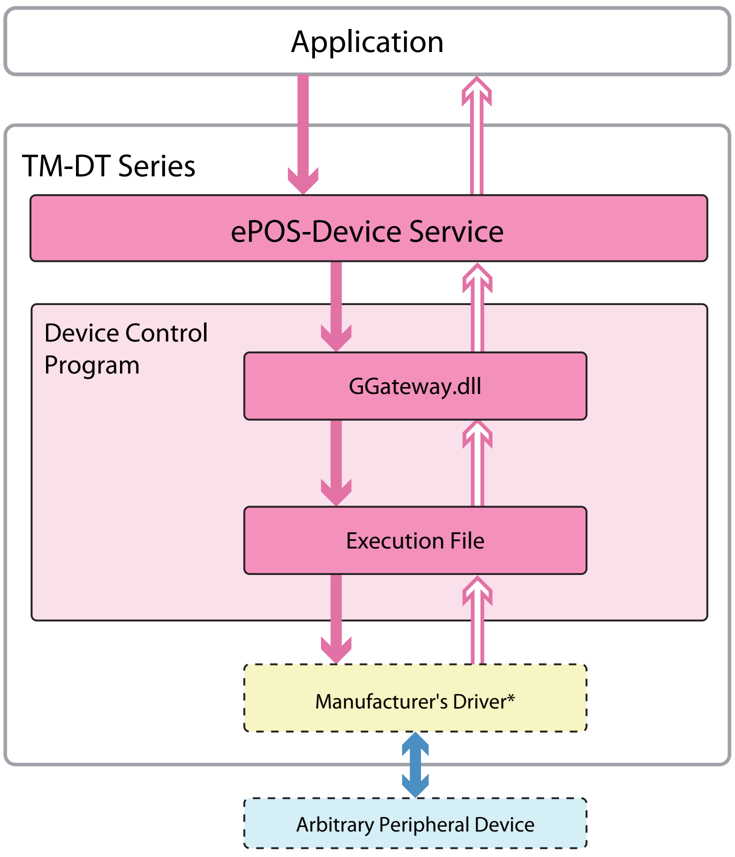
Device ID	Control program	
OPOS_MSR	OposMSRHandler.exe	Delete
OPOS_POSkeyboard	OposPOSKeyboardHandler.exe	Delete
OPOS_Scanner	OposScannerHandler.exe	Delete

Developing a Device Control Program

This section describes the method of registering the information necessary for developing a device control program, and the developed device control program in the TM-DT series.

Overview

When a command for starting communication with the peripheral devices is executed from ePOS SDK, the ePOS-Device Service starts the device control program.
Control the peripheral devices by the device control program that has been started.



* Prepare as required according to the device control program to be developed.

Configuration of a Device Control Program

Configure the device control program under the following conditions:

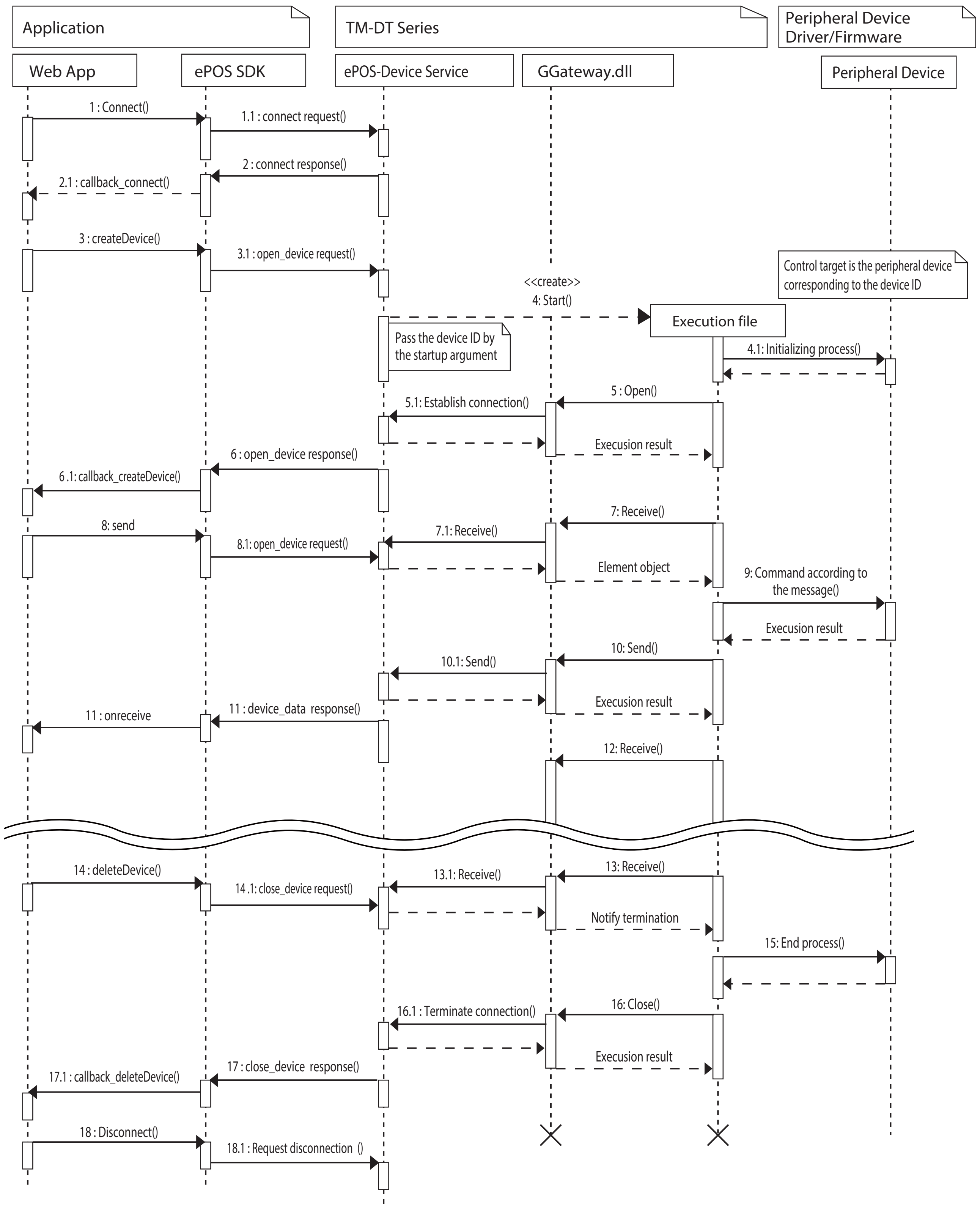
❏ Create an execution file in the .exe format.

❏ Do not include multiple execution files in a single device control program.

❏ Use the file provided with the Epson ePOS SDK package as GGateway.dll.

Operation Sequence

The operation sequence between the ePOS-Device Service and the device control program is shown below. This is a sequence diagram for performing control from the Web application developed by using Epson ePOS SDK.



Application
TM-DT Series
Peripheral Device Driver/Firmware
Web App
ePOS SDK
ePOS-Device Service
GGateway.dll
Peripheral Device
1 : Connect()
1.1 : connect request()
2 : connect response()
2.1 : callback_connect()
3 : createDevice()
3.1 : open_device request()
<<create>>
4: Start()
Control target is the peripheral device corresponding to the device ID
Execution file
Pass the device ID by the startup argument
4.1: Initializing process()
5 : Open()
5.1: Establish connection()
Execusion result
6 : open_device response()
6 .1: callback_createDevice()
7: Receive()
8: send
7.1: Receive()
8.1: open_device request()
Element object
9: Command according to the message()
Execusion result
10: Send()
10.1: Send()
Execusion result
11 : onreceive
11 : device_data response()
12: Receive()
14 : deleteDevice()
13: Receive()
13.1: Receive()
14 .1: close_device request()
Notify termination
15: End process()
16: Close()
16.1 : Terminate connection()
Execusion result
17 : close_device response()
17.1 : callback_deleteDevice()
18 : Disconnect()
18.1 : Request disconnection ()

Sample Program

The following files are included in the sample program file (DeviceControlProgram_Sample.zip) for the device control program.

Filename		Description
ExecuteFiles	GGateway.dll	This is a communication library for enabling the device control program to send and receive data to/from the ePOS-Device Service. It is used in combination with the execution file to be developed.
	Sample01.exe	This is the sample program of the execution file.
	DevCtrlPrgTester.exe	This is a tool for checking the operation of the developed device control program.
Sample01Project		This is a set of source files of Sample01.exe.

Methods Provided by GGateway.dll

GGateway.dll provides methods for the ePOS-Device Service to send and receive data to/from the execution file.

Open Method

This method is used to execute the Open processing. After the completion of the processing, the processing results are returned to the execution file.

Syntax

```
int Open(int Port, char* DeviceID, bool XmlLog, bool
SingleThreadApartment);
```

Parameter

Setting value	Description
Port	This is the port number for communicating with the ePOS-Device Service. Set the value passed by the startup argument.
DeviceID	Device ID of the peripheral device to be used. Set the value passed by the startup argument.
XmlLog	Setting for whether or not to output the reception/ transmission XML log (Specify "false" when sending/ receiving data concerning individual information.)
SingleThreadApartment	Set the apartment to which the device control program belongs. true: SingleThreadApartment false: MultiThreadApartment

Return value

Return value	Contents
GG_SUCCESS(0)	Processing successful.
GG_SOCKET_ERROR(-1)	An error occurred during communication with the ePOS-Device Service.
GG_INTERNAL_ERROR(-2)	An internal error occurred.
GG_PARAM_ERROR(-3)	A parameter error occurred.
GG_EPOS_SYS_ERROR(-4)	A system error occurred.
GG_ALREADY_OPENED(-5)	Already open.

Receive Method

This method is used to await the reception of data from the ePOS-Device Service. Analyze the received command, and save data in the parameter. After the completion of the processing, the processing results are returned to the execution file.

Syntax

```
int Receive(char* MethodName, Element* ReceiveData,
unsigned int* SequenceNo);
```

Parameter

Setting value	Description
MethodName	Pointer in which the ePOS method name of the reception data is saved.
ReceiveData	Pointer of the Element object in which the reception data is saved.
SequenceNo	Pointer in which the sequence number of the reception data is saved.

Return value

Return value	Contents
GG_SUCCESS(0)	Processing successful.
GG_SOCKET_ERROR(-1)	An error occurred during communication with the ePOS-Device Service.
GG_PARAM_ERROR(-3)	A parameter error occurred.
GG_EPOS_SYS_ERROR(-4)	A system error occurred.
GG_RECV_DISCONNECT(-7)	A disconnection request is received from the ePOS-Device Service.
GG_NOT_OPENED(-8)	Not opened.

Send Method

This method is used to send data to the ePOS-Device Service. After the completion of the processing, the processing results are returned to the execution file.

Syntax

```
int Send(string EventName, Element* SendData, unsigned int
SequenceNo);
```

Parameter

Setting value	Description
EventName	ePOS event name
SendData	Pointer of the transmission data object
SequenceNo	Sequence number

Return value

Return value	Contents
GG_SUCCESS(0)	Processing successful.
GG_SOCKET_ERROR(-1)	An error occurred during communication with the ePOS-Device Service.
GG_PARAM_ERROR(-3)	A parameter error occurred.
GG_EPOS_SYS_ERROR(-4)	A system error occurred.
GG_NOT_OPENED(-8)	Not opened.

Close Method

This method is used to perform the close processing with the ePOS-Device Service. After the completion of the processing, the processing results are returned to the execution file.

Syntax

```
int Close();
```

Return value

Return value	Contents
GG_SUCCESS(0)	Processing successful.
GG_SOCKET_ERROR(-1)	An error occurred during communication with the ePOS-Device Service.
GG_NOT_OPENED(-8)	Not opened.

Element Object

The Element object is used by the Receive method and Send method between the execution file and GGateway.dll. The structure of the Element object, and the acquisition of information and method for making settings are as described below.

Structure of element object

m_strName	Element name of Element
m_strValue	Element value of Element
m_strType	Type of Element (String/numeric value)
m_bArray	Truth value of whether or not the Element is an array
m_children	Pointer array of child Element
m_parent	Address of parent Element (NULL in the case of the highest order)

Constructor

Used to generate the Element object. It is provided in four formats.

Syntax

```
Element();
Element(char* Name);
Element(char* Name, char* Value);
Element(char* Name, long Value);
```

Parameter

Setting value	Description
Name	Element name set in the Element object.
Value	String value of the element / numeric value of the element set in the Element object.

getName Method

This method is used to acquire the element name from the Element object.

Syntax

```
char* getName()
```

Return value

Element name of Element

getValue Method

This method is used to acquire the element value from the Element object by a string.

Syntax

```
char* getValue()
```

Return value

Element value of Element

getType Method

This method is used to acquire the type of the element value from the Element object by a string.

Syntax

```
char* getType()
```

Return value

Type of Element

getParent Method

This method is used to acquire the address of the parent element of the Element object.

Syntax

```
Element* getParent();
```

Return value

Pointer of the parent Element

haveChild Method

This method is used to acquire the child Element existence information of the Element object.

Syntax

```
bool haveChild();
```

Return value

Return value	Contents
true	A child Element exists.
false	A child Element does not exist.

getChildrenNum Method

This method is used to acquire the number of child Elements of the Element object.

Syntax

```
int getChildrenNum ();
```

Return value

Number of child Elements

getChild Method

This method is used to acquire the child Element of the Element object.

Syntax

```
Element* getChild(int Num);
Element* getChild(char* Name);
```

Parameter

Setting value	Description
Num	Element number of the child Element to be acquired.
Name	Element name of the child Element to be acquired (If a child Element with the same name already exists, the child Element found first i acquired.)

Return value

Return value	Contents
Pointer of child Element	Pointer of child Element matching the parameter
NULL	A child Element does not exist.

addChild Method

This method is used to add a child Element to the Element object.

Syntax

```
bool addChild(Element* Child);
```

Parameter

Setting value	Description
Child	Pointer of the child Element to be added to the Ele-ment object

Return value

Return value	Contents
true	Process succeeded.
false	Process failed.

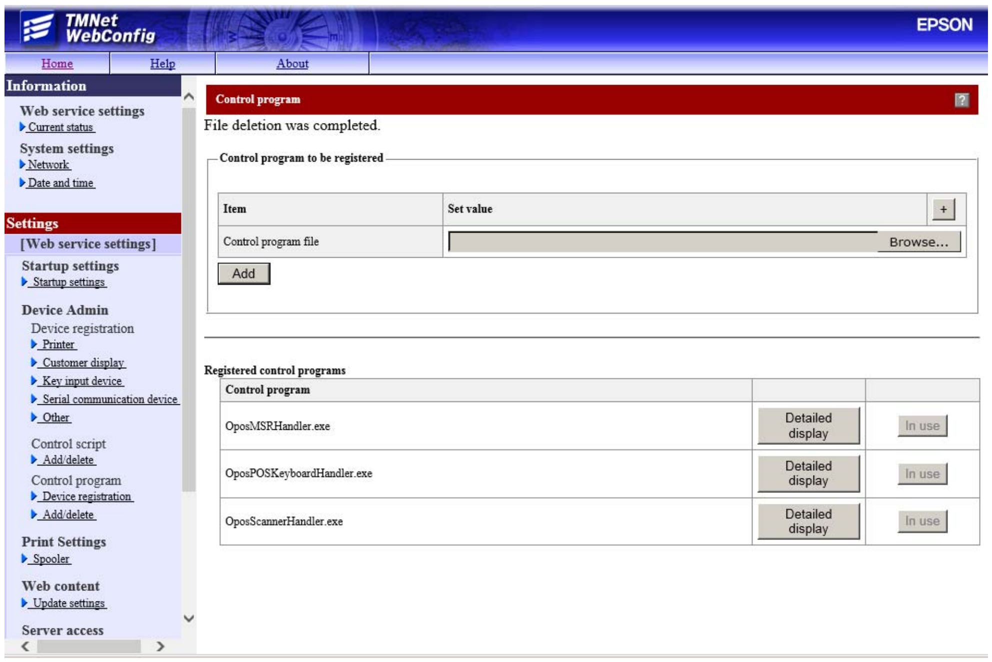
Adding a Program

This section describes the procedure of adding the developed device control program to the TM-DT series.

1 Start EPSON TMNet WebConfig.

Refer to the Technical Reference Guide of the TM-DT series.

2 Select **Web service settings** - **Control program** - **Add/delete** to open the Control program screen.



Control program

File deletion was completed.

Control program to be registered

Item	Set value
Control program file	

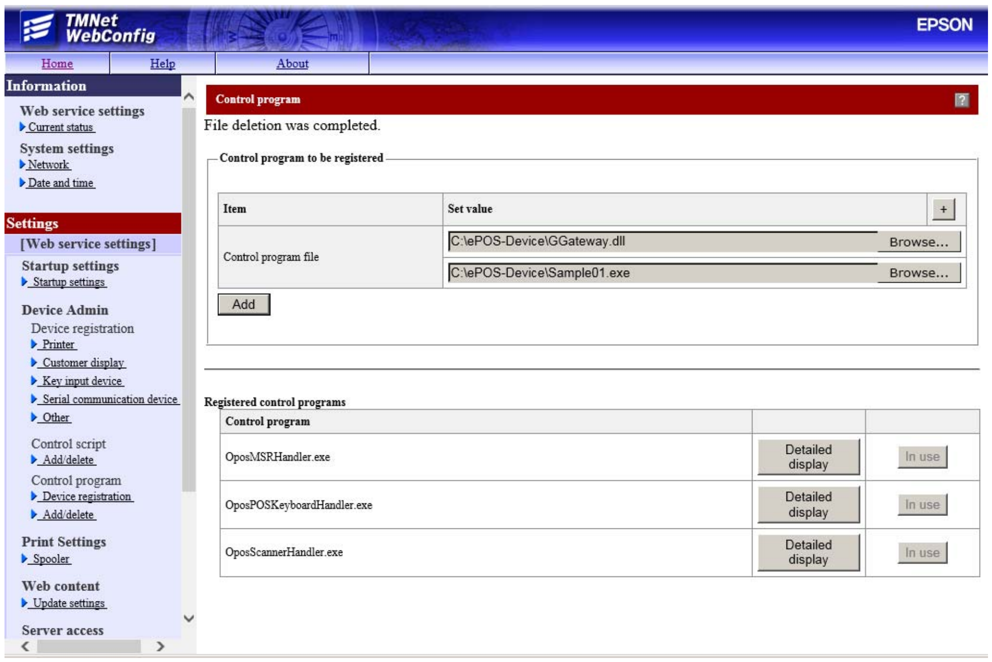
Registered control programs

Control program		
OposMSRHandler.exe	Detailed display	In use
OposPOSKeyboardHandler.exe	Detailed display	In use
OposScannerHandler.exe	Detailed display	In use

3 Click **Browse** and select GGateway.dll.

4 Click **+** to add a row, and click **Browse** to select the developed execution file.

> Select the required file according to the configuration of the developed device control program.



Control program

File deletion was completed.

Control program to be registered

Item	Set value
Control program file	C:\ePOS-Device\GGateway.dll
	C:\ePOS-Device\Sample01.exe

Registered control programs

Control program		
OposMSRHandler.exe	Detailed display	In use
OposPOSKeyboardHandler.exe	Detailed display	In use
OposScannerHandler.exe	Detailed display	In use

5 Click **Add**.

6 When addition is complete, the registered information is displayed in **Registered control programs**.

Click **Detailed display** to view the configuration file of the device control program.
Click **Delete** to delete the corresponding registered information.

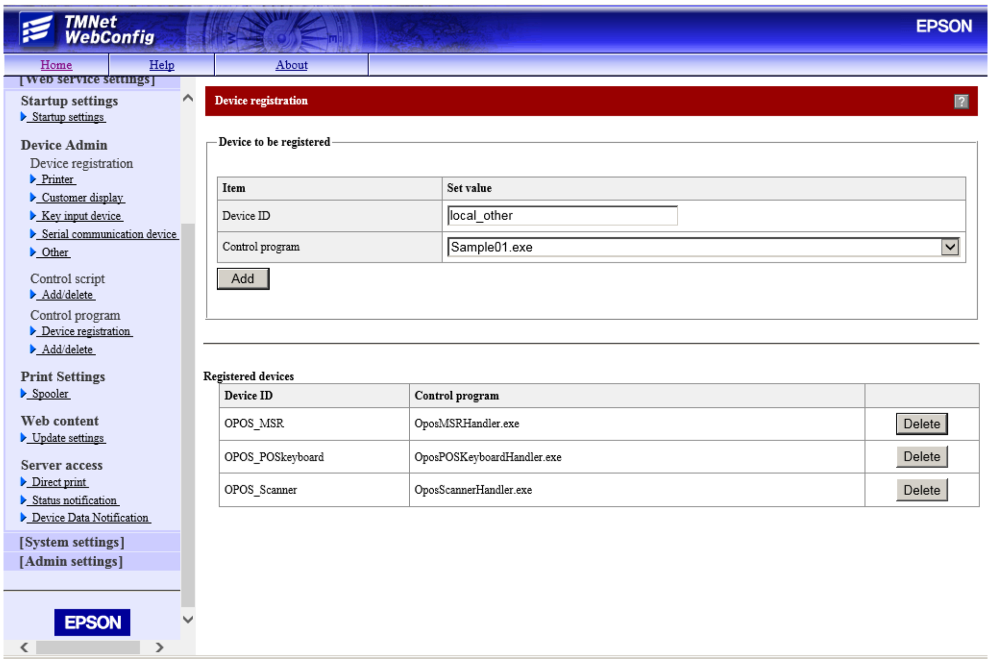
Device Registration

This section describes the procedure of registering the peripheral devices to be controlled by the developed device control program in the TM-DT series.

1 Start EPSON TMNet WebConfig.

Refer to the Technical Reference Guide of the TM-DT series.

2 Select **Web service settings** - **Control program** - **Device registration** to open the Device registration screen.



Device registration

Device to be registered

Item	Set value
Device ID	local_other
Control program	Sample01.exe

Add

Registered devices

Device ID	Control program	
OPOS_MSR	OposMSRHandler.exe	Delete
OPOS_POSkeyboard	OposPOSKeyboardHandler.exe	Delete
OPOS_Scanner	OposScannerHandler.exe	Delete

3 Configure the following settings and click **Add**.

Configure the settings for each peripheral device to be controlled.

Item to set	Setting value
Device ID	Optional device ID
Control program	Device control program added in Adding a Program

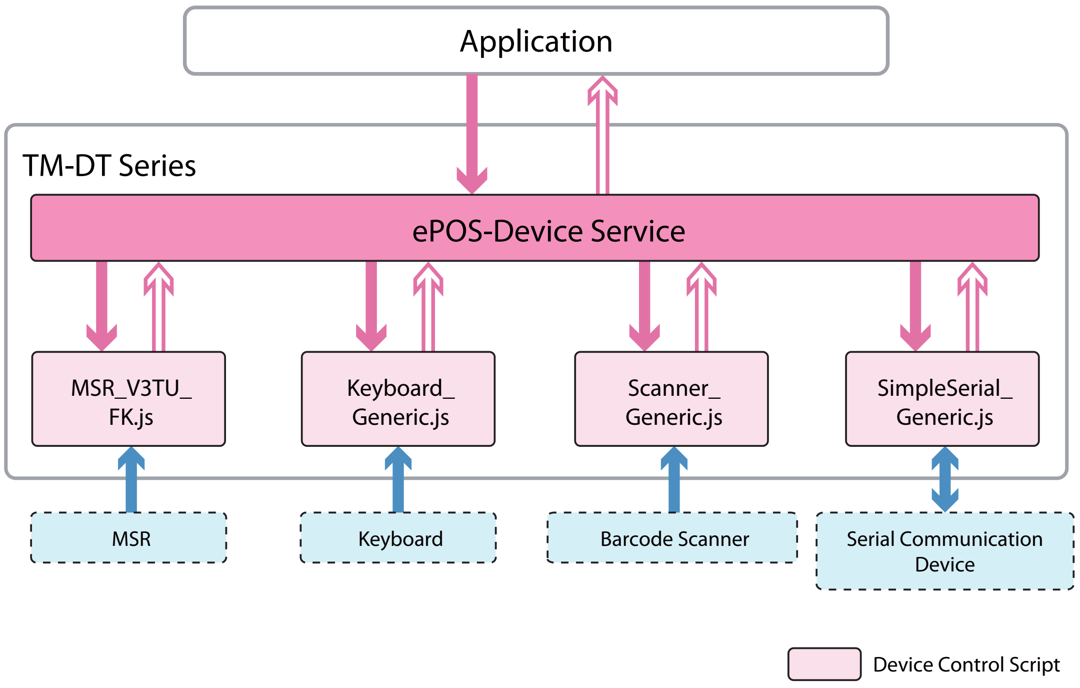
4 When registration is complete, the registered information is displayed in **Registered devices**.

Click **Delete** to delete the corresponding registered information.

Controlling by a Device Control Script

This section describes the method of controlling the key input devices and serial communication devices by using the device control script loaded in the TM-DT series.

In the TM-DT series, a device control script is available for each peripheral device.
Peripheral devices corresponding to the device control script can be controlled.



Device Registration

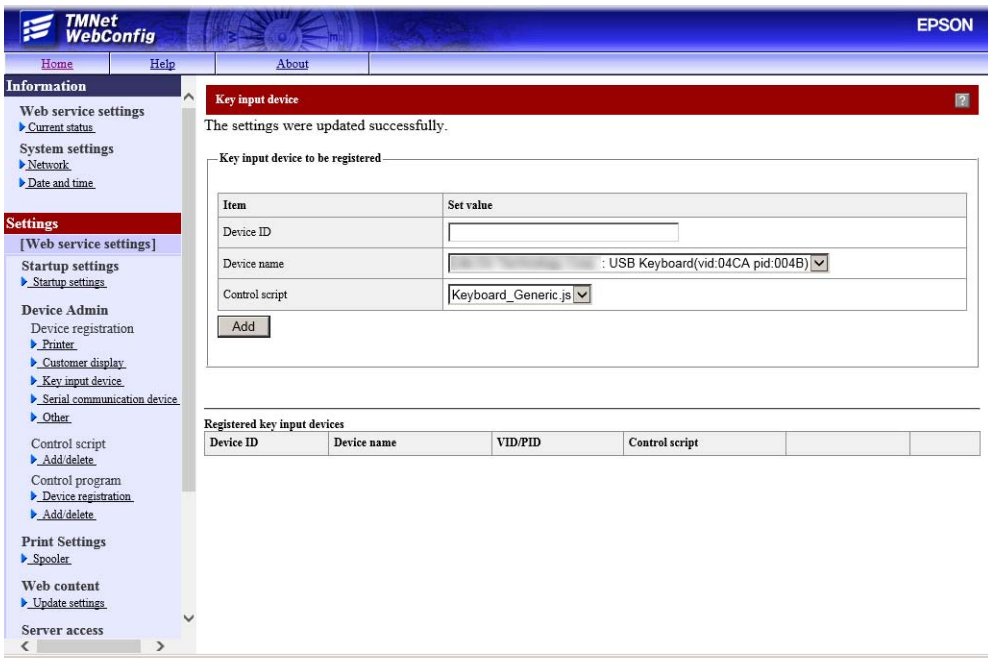
This section describes the procedure of registering the peripheral devices to be used in the TM-DT series.

Key Input Device

1 Connect the peripheral devices to be used to the TM-DT series.

2 Start EPSON TMNet WebConfig.
Refer to the Technical Reference Guide of the TM-DT series.

3 Select **Web service settings** - **Device registration** - **Key input device** to open the Device registration screen.



Key input device

The settings were updated successfully.

Key input device to be registered

Item	Set value
Device ID	
Device name	[redacted] : USB Keyboard(vid:04CA pid:004B)
Control script	Keyboard_Generic.js

Registered key input devices

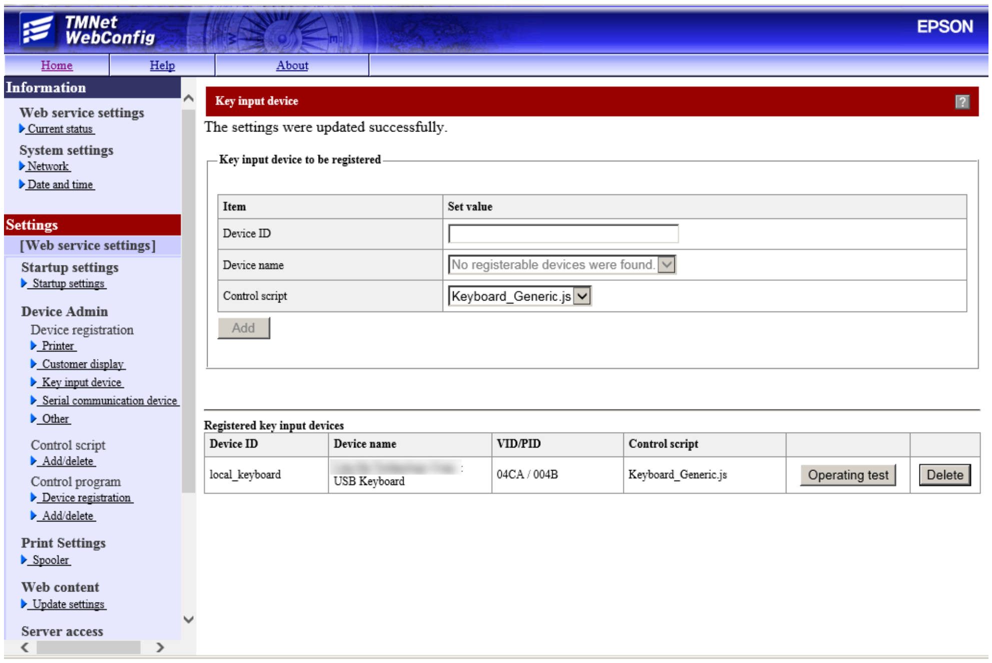
Device ID	Device name	VID/PID	Control script		

4 Configure the following settings for each peripheral device to be used, and click **Add**.

Peripheral device	Item to set	Setting value
MSR	Device ID	Optional device ID
	Device name	Select the appropriate device from the list.
	Control script	MSR_V3TU_FK.js
Keyboard	Device ID	Optional device ID
	Device name	Select the appropriate device from the list.
	Control script	Keyboard_Generic.js
Barcode Scanner	Device ID	Optional device ID
	Device name	Select the appropriate device from the list.
	Control script	Scanner_Generic.js

5 When registration is complete, the registered information is displayed in **Registered key input devices**.

Click **Delete** to delete the corresponding registered information.



Key input device

The settings were updated successfully.

Key input device to be registered

Item	Set value
Device ID	
Device name	No registerable devices were found.
Control script	Keyboard_Generic.js

Add

Registered key input devices

Device ID	Device name	VID/PID	Control script		
local_keyboard	[redacted] USB Keyboard	04CA / 004B	Keyboard_Generic.js	Operating test	Delete

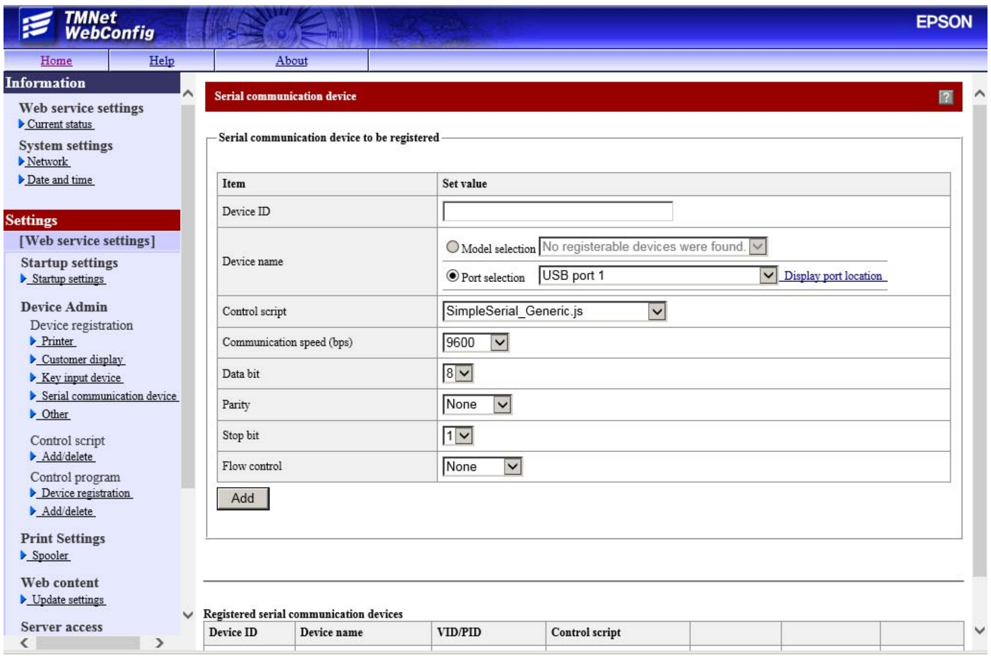
Serial Communication Device

1 Connect the peripheral devices to be used to the TM-DT series.

2 Start EPSON TMNet WebConfig.

Refer to the Technical Reference Guide of the TM-DT series.

3 Select **Web service settings** - **Device registration** - **Serial communication device** to open the Device registration screen.



Serial communication device

Serial communication device to be registered

Item	Set value
Device ID	
Device name	○ Model selection: No registerable devices were found. / ● Port selection: USB port 1 — Display port location
Control script	SimpleSerial_Generic.js
Communication speed (bps)	9600
Data bit	8
Parity	None
Stop bit	1
Flow control	None

Registered serial communication devices

Device ID	Device name	VID/PID	Control script			

4 Configure the following settings for each peripheral device to be used, and click **Add**.

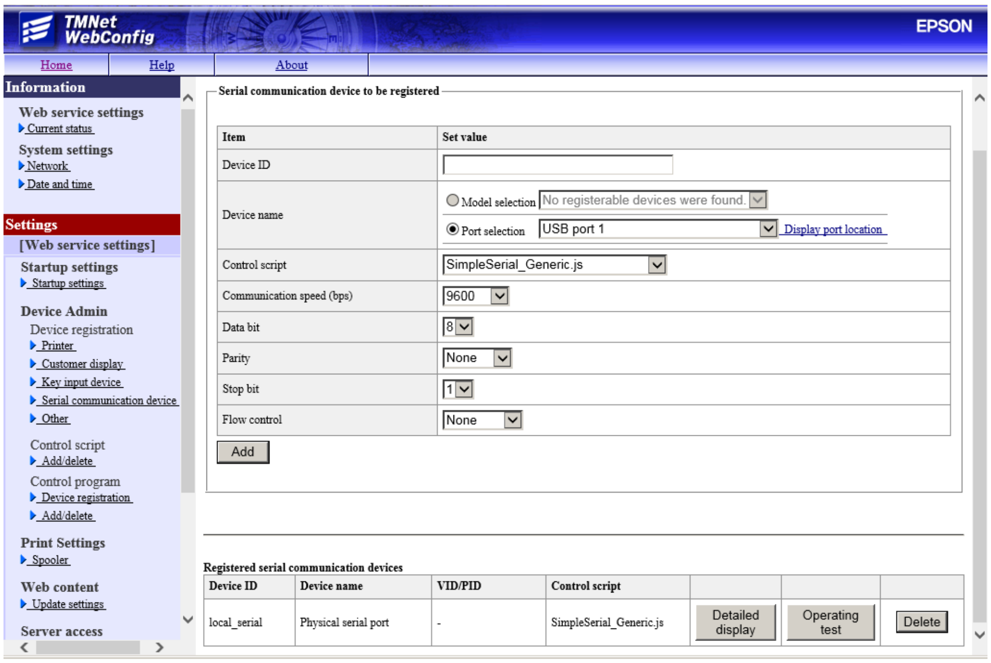
Peripheral device	Item to set	Setting value
Serial communication device	Device ID	Optional device ID
	Device name [*1]	Select the appropriate device from the list, or select the port to be used.
	Port [*2]	Select the port to be used.
	Control script	SimpleSerial_Generic.js

*1: Displayed only for TM-T70II-DT/TM-T88V-DT/TM-H6000IV-DT.
*2: Displayed only for TM-T70II-DT2/TM-T88VI-DT2.

Set the **Communication speed (bps)**, **Data bit**, **Parity**, **Stop bit**, and **Flow control** according to the peripheral device to be used.

5 When registration is complete, the registered information is displayed in **Registered serial communication devices**.

Click **Delete** to delete the corresponding registered information.



Serial communication device to be registered

Item	Set value
Device ID	
Device name	Model selection: No registerable devices were found. [✓ Port selection: USB port 1] Display port location
Control script	SimpleSerial_Generic.js
Communication speed (bps)	9600
Data bit	8
Parity	None
Stop bit	1
Flow control	None

Registered serial communication devices

Device ID	Device name	VID/PID	Control script			
local_serial	Physical serial port	-	SimpleSerial_Generic.js	Detailed display	Operating test	Delete

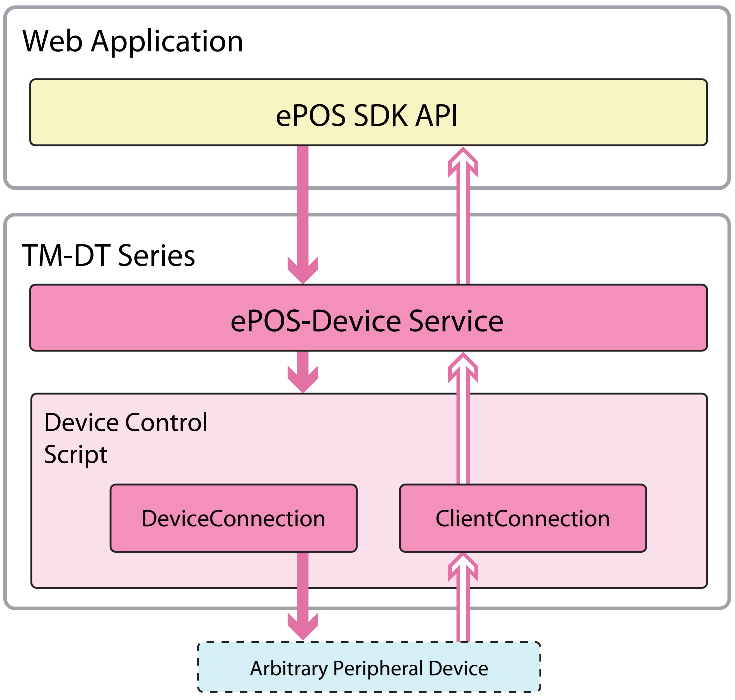
Developing a Device Control Script

This section describes the information necessary for developing a device control script, and the method of registering the developed device control script in the TM-DT series.
The peripheral device control using the developed device control script is supported by Epson ePOS SDK for JavaScript and ePOS-Device XML.

Overview

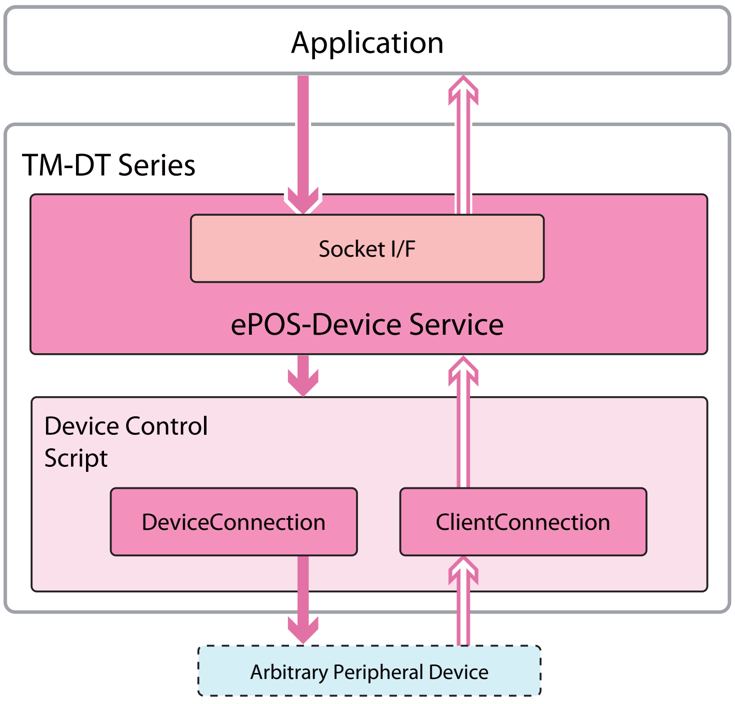
Epson ePOS SDK for JavaScript

When the createDevice method of ePOS SDK API is executed, the ePOS-Device Service generates the objects of the device control script.
Control the peripheral devices by the generated objects.



ePOS-Device XML

When the open_device message is sent, the ePOS-Device Service generates the objects of the device control script.
Control the peripheral devices by the generated objects.



Device Control Script Objects

In the device control script, the following objects are passed from the ePOS-Device Service. The device control script communicates with applications and peripheral devices by using these objects.

Object	Description
ClientConnection	Object used to send data to device objects in the applications
DeviceConnection	Object used to send and receive data to/from peripheral devices

Functions that Use Device Control Script Objects

The available functions that use the device control script API are as follows.

❏ Calling of user-defined events for device objects in applications

❏ Sending of data to peripheral devices

❏ Receiving of data generated in peripheral devices

Configuration of a Device Control Script

A device control script is developed in JavaScript. Perform coding under the following conditions:

❏ The codes necessary for device control shall be compiled in one file.
(This is because the device setting can be registered only in one file when using EPSON TMNet WebConfig.)

❏ The part of the file name before the first dot shall be same as the name of the constructor.
Ex.) Filename: Keyboard_Generic.ver1.0.js -> Name of constructor: Keyboard_Generic

❏ Declare exports for constructor external references.
Ex.) exports.Keyboard_Generic = Keyboard_Generic;

❏ The constructor shall have two arguments.

❏ The device control script must have the following properties. Constructors must be configured with appropriate names.

- DEVICE_TYPE property (Type of object: String)

Setting value	Description
type_keyboard	Specifies the use of keyboard devices.
type_msr	Specifies the use of MSR.
type_scanner	Specifies the use of barcode scanners.
type_simple_serial	Specifies the use of simple serial communication.

- DEVICE_GROUP property (Type of object: String)

Setting value	Description
group_hid	Specifies the use of key input devices operable via HID drivers.
group_serial	Specifies the use of serial communication devices.
group_other	Specifies the use of other peripheral devices.

❏ The onDeviceData method must be created to receive data generated by peripheral devices.
Refer to Device Control Scripts Naming Object for more information.

❏ Methods corresponding to methods for device objects that function in applications must be created.
Refer to User-defined Event for more information.

Example device control script structure

```
// Declares exports
exports.Keyboard_Generic = Keyboard_Generic;

// Defines a function with the same name as the filename with two arguments
function Keyboard_Generic(clientConn, deviceConn){
// Defines the DEVICE_TYPE properties
    this.DEVICE_TYPE = 'type_keyboard';
// Defines the DEVICE_GROUP properties
    this.DEVICE_GROUP = 'group_hid';
    this.clientConn = clientConn;
    this.deviceConn = deviceConn;
......
......
}

Keyboard_Generic.prototype = {
// Defines the onDeviceData method
    onDeviceData : function(event, keycode, ascii){...},
// Defines a method corresponding to a device object
    setprefix : function(data){...}
}
```

List of Device Control Script APIs

Device control script APIs are preconfigured with the following objects.

❏ ClientConnection Object

❏ DeviceConnection Object

❏ Device Control Scripts Naming Object

ClientConnection Object

This object is passed to the first parameter from the device control script constructor.

API		Description
Send	send	This object sends data to device objects operating in the Web browser.

DeviceConnection Object

This object is passed to the second parameter from the device control script constructor.

API		Description
Send	send	This object sends data to serial communication devices.

Device Control Scripts Naming Object

This object receives data from peripheral devices.

API		Description
Receive results	onDeviceData Event (key input device)	Event that receives data from key input devices
	onDeviceData Event (serial communication device)	Event that receives data from serial communication devices
	User-defined Event	Event that receives results of API execution regarding device objects operating in the Web browser

ClientConnection Object

send

This object sends data to device objects operating in the Web browser.

Syntax

```
send(event, data);
```

Parameter

event

Setting value	Description
String	Specifies the name of the device object event

data

Setting value	Description
Object	Specifies the data to be passed to the device object event.

Sample program

This sample program calls the device object onkeypress event and uses the onkeypress event data parameter to receive a value of 49 from data.keycode and a value of 1 from data.ascii.

```
data = {'keycode' : 49, 'ascii' : '1'};
clientConn.send('onkeypress', data)
```

DeviceConnection Object

send

This object sends data to serial communication devices.

Syntax

```
send(data);
```

Parameter

data

Setting value	Description
Buffer	Specifies data to send to peripheral devices.

Supplemental information

Data cannot be sent to input devices operable via HID drivers.

Device Control Scripts Naming Object

onDeviceData Event (key input device)

This event receives detected data from key input devices operable via HID drivers.
Create this event when using device control scripts for key input devices.

Syntax

```
onDeviceData(event, keycode, ascii);
```

Parameter

event

Receives key operation state information.

Value	Description
1	Key down
2	Key up

keycode

Value	Description
Number	Keycode

ascii

Value	Description
String	Text corresponding to the operated key

Supplemental information

❏ For details on values received by keycode, refer to the User's Manual of Epson ePOS SDK for JavaScript or ePOS-Device XML.

❏ A value of "undefined" is input when there is no text corresponding to the keycode received by the ascii parameter.

onDeviceData Event (serial communication device)

This event receives detected data from serial communication devices.
Create this event when using device control scripts for serial communication devices.

Syntax

```
onDeviceData(data);
```

Parameter

data

Value	Description
Buffer	Data received from serial communication devices

User-defined Event

This event receives results of API execution regarding device objects operating in the Web browser.

Syntax

```
Name specified with callEvent(data);
```

Parameter

data

Value	Description
Object	Object with parameter specified by the callEvent method of device object.

Supplemental information

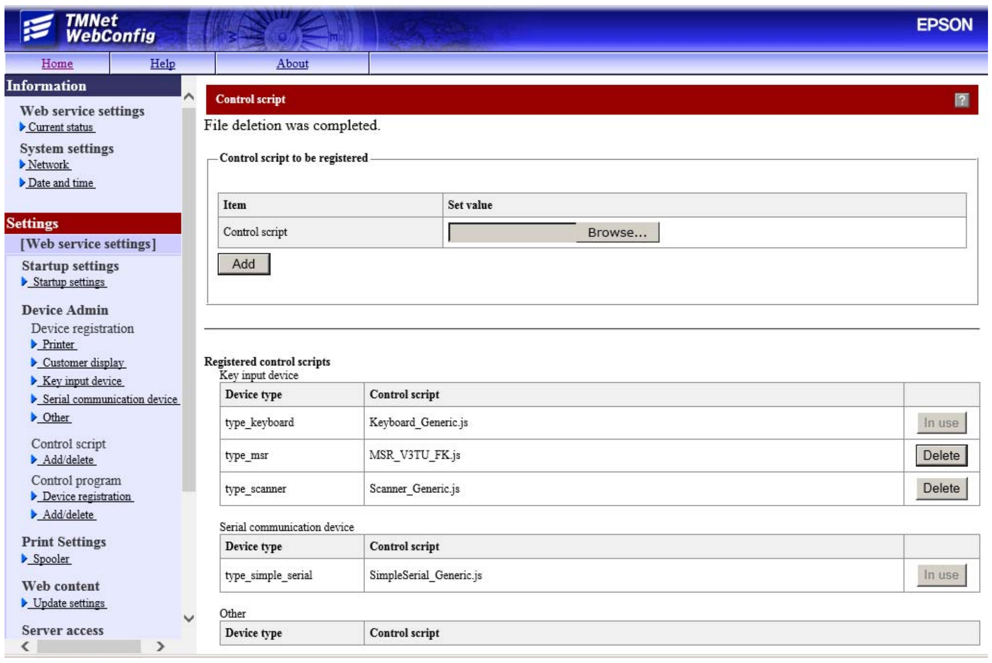
For details on the callEvent method, refer to the User's Manual of Epson ePOS SDK for JavaScript or ePOS-Device XML.

Adding a Script

This section describes the procedure of adding the developed device control script to the TM-DT series.

1 Start EPSON TMNet WebConfig.
Refer to the Technical Reference Guide of the TM-DT series.

2 Select **Web service settings** - **Control script** - **Add/delete** to open the Control script screen.



Control script

File deletion was completed.

Control script to be registered

Item	Set value
Control script	Browse...

Add

Registered control scripts

Key input device

Device type	Control script	
type_keyboard	Keyboard_Generic.js	In use
type_msr	MSR_V3TU_FK.js	Delete
type_scanner	Scanner_Generic.js	Delete

Serial communication device

Device type	Control script	
type_simple_serial	SimpleSerial_Generic.js	In use

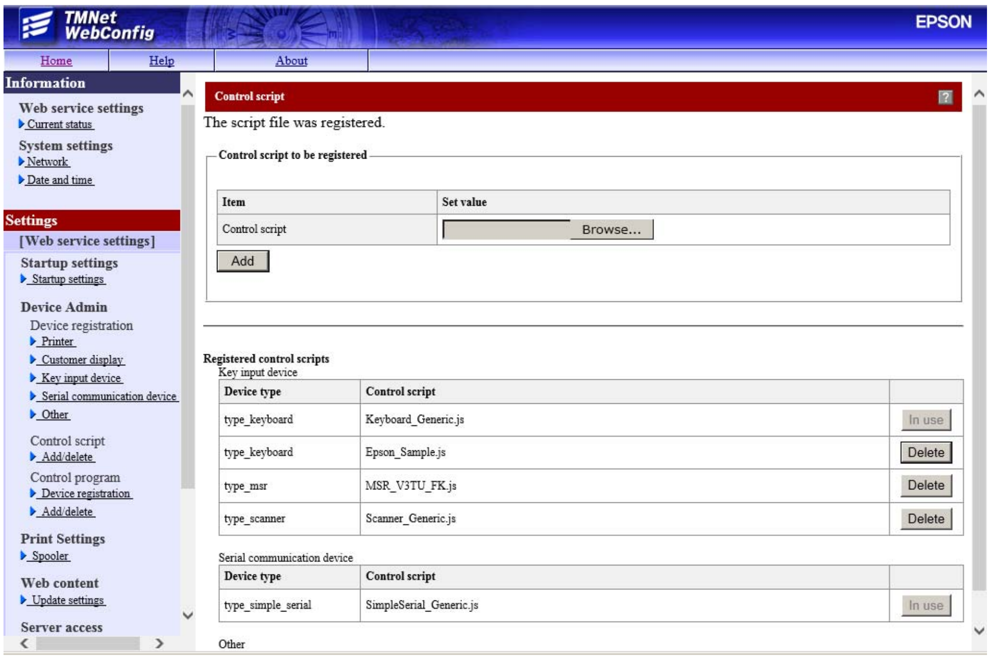
Other

Device type	Control script

3 Click **Browse** to select the device control script to be added, and click **Add**.

4 When addition is complete, the registered information is displayed in **Registered control scripts**.

The scripts are automatically classified according to the setting value specified in the DEVICE_TYPE property and the DEVICE_GROUP property.
Click **Delete** to delete the corresponding registered information.



Control script

The script file was registered.

Control script to be registered

Item	Set value
Control script	

Registered control scripts

Key input device

Device type	Control script	
type_keyboard	Keyboard_Generic.js	In use
type_keyboard	Epson_Sample.js	Delete
type_msr	MSR_V3TU_FK.js	Delete
type_scanner	Scanner_Generic.js	Delete

Serial communication device

Device type	Control script	
type_simple_serial	SimpleSerial_Generic.js	In use

Other

Device Registration

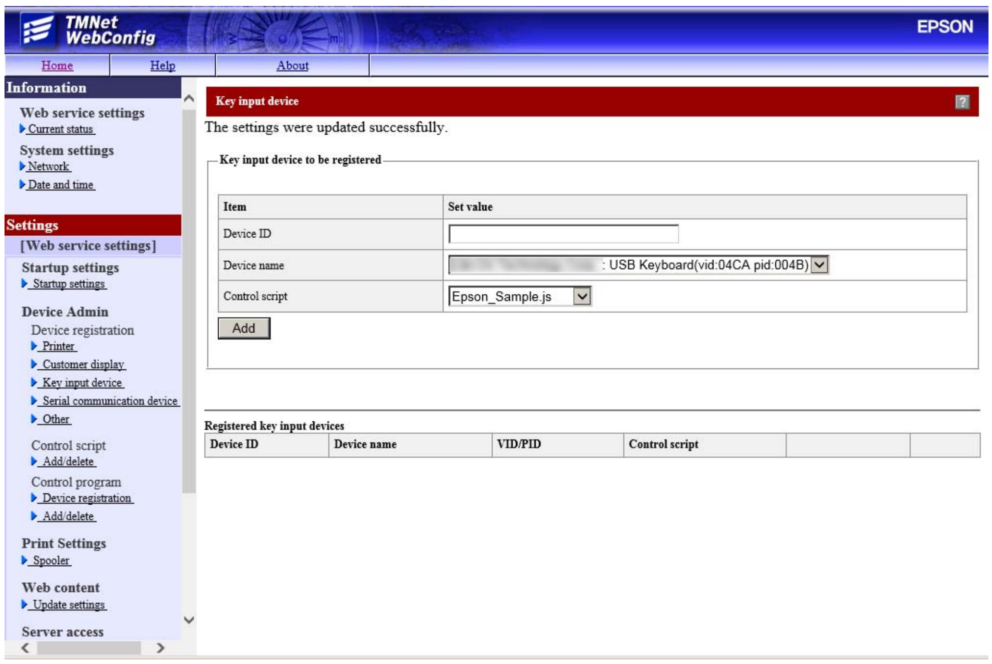
This section describes the procedure of registering the peripheral devices to be controlled by the developed device control script in the TM-DT series.

1 Connect the peripheral devices to be used to the TM-DT series.

2 Start EPSON TMNet WebConfig.

Refer to the Technical Reference Guide of the TM-DT series.

3 From **Web service settings** - **Device registration**, open the Device registration screen of the peripheral device to be used.



The settings were updated successfully.

Key input device to be registered

Item	Set value
Device ID	
Device name	[redacted] : USB Keyboard(vid:04CA pid:004B)
Control script	Epson_Sample.js

Registered key input devices

Device ID	Device name	VID/PID	Control script		

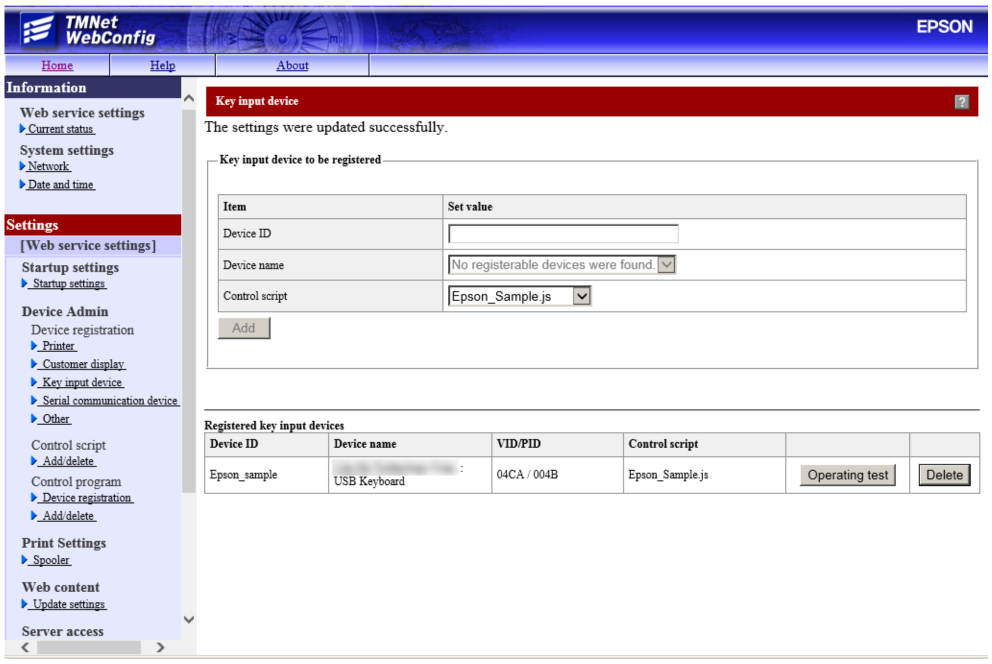
4 Configure the following settings and click **Add**.

Configure the settings for each peripheral device to be controlled.

Item to set	Setting value
Device ID	Optional device ID
Device name	Select the appropriate device from the list, or select the port to be used.
Control script	Device control script added in Adding a Script

5 When registration is complete, the registered information is displayed in **Registered key input devices**.

Click **Delete** to delete the corresponding registered information.



Key input device

The settings were updated successfully.

Key input device to be registered

Item	Set value
Device ID	
Device name	No registerable devices were found.
Control script	Epson_Sample.js

Add

Registered key input devices

Device ID	Device name	VID/PID	Control script		
Epson_sample	[redacted] : USB Keyboard	04CA / 004B	Epson_Sample.js	Operating test	Delete