

# TM-DTシリーズ 周辺機器制御ガイド

---

概要

デバイス制御プログラムで制御する

デバイス制御プログラムの開発

デバイス制御スクリプトで制御する

デバイス制御スクリプトの開発

## ご注意

- 本書の内容の一部または全部を無断で転載、複写、複製、改ざんすることは固くお断りします。
- 本書の内容については、予告なしに変更することがあります。最新の情報はお問い合わせください。
- 本書の内容については、万全を期して作成いたしましたが、万一ご不審な点や誤り、記載もれなど、お気づきの点がありましたらご連絡ください。
- 運用した結果の影響については、上項にかかわらず責任を負いかねますのでご了承ください。
- 本製品がお客様により不適切に使用されたり、本書の内容に従わずに取り扱われたり、またはエプソンおよびエプソン指定の者以外の第三者により修理・変更されたことなどに起因して生じた損害などにつきましては、責任を負いかねますのでご了承ください。
- エプソン純正品およびエプソン品質認定品以外のオプションまたは消耗品を装着してトラブルが発生した場合には、責任を負いかねますのでご了承ください。

## 商標について

Windows<sup>®</sup>、Internet Explorer<sup>®</sup> は、米国 Microsoft Corporation の米国およびその他の国における商標または登録商標です。

JET-S は JET-STANDARD の略称です。JET-STANDARD は、株式会社カードネットワークの登録商標です。

CAFIS は、株式会社 NTT データの登録商標です。

stera は、三井住友カード株式会社の登録商標です。

銀聯は、CHINA UNIONPAY CO., Ltd. の登録商標です。

楽天 Edy は、楽天 Edy 株式会社の登録商標です。

iD は株式会社 NTT ドコモの登録商標です。

nanaco は、株式会社セブン・カードサービスの登録商標です。

QUICPay は、株式会社ジェーシービーの登録商標です。

Suica は、東日本旅客鉄道株式会社の登録商標です。

WAON は、イオン株式会社の登録商標です。

P i T a P a は、株式会社スルッと K A N S A I の登録商標です。

QR コードは株式会社デンソーウェーブの登録商標です。



その他の製品名および会社名は、各社の商標または登録商標です。

©Seiko Epson Corporation 2017-2023

# 安全のために

## 記号の意味

本書では以下の記号が使われています。それぞれの記号の意味をよく理解してから製品を取り扱ってください。

	ご使用上、必ずお守りいただきたいことを記載しています。この表示を無視して誤った取り扱いをすると、製品の故障や動作不良の原因になる可能性があります。
	補足説明や知っておいていただきたいことを記載しています。

## 使用制限

本製品を航空機・列車・船舶・自動車などの運行に直接関わる装置・防災防犯装置・各種安全装置など機能・精度などにおいて高い信頼性・安全性が必要とされる用途に使用される場合は、これらのシステム全体の信頼性および安全維持のためにフェールセーフ設計や冗長設計の措置を講じるなど、システム全体の安全設計にご配慮いただいた上で弊社製品をご使用いただくようお願いいたします。

本製品は、航空宇宙機器、幹線通信機器、原子力制御機器、医療機器など、きわめて高い信頼性・安全性が必要とされる用途への使用を意図しておりませんので、これらの用途には本製品の適合性をお客様において十分ご確認の上、ご判断ください。

## 本書について

### 本書の目的

TM-DT シリーズのプリンターで周辺機器を制御する方法を解説します。

### 本書の構成

本書は次のように構成されています。

- 第 1 章      [概要](#)
- 第 2 章      [デバイス制御プログラムで制御する](#)
- 第 3 章      [デバイス制御プログラムの開発](#)
- 第 4 章      [デバイス制御スクリプトで制御する](#)
- 第 5 章      [デバイス制御スクリプトの開発](#)

# もくじ

■ 安全のために .....	3
記号の意味 .....	3
■ 使用制限 .....	3
■ 本書について .....	3
本書の目的 .....	3
本書の構成 .....	3
■ もくじ .....	4

---

## 概要 ..... 6

■ 対象プリンターと周辺機器 .....	8
対象プリンター .....	8
サポート周辺機器 .....	8
■ 環境設定 .....	9
Windows の設定 .....	9
EPSON TMNet WebConfig .....	9

---

## デバイス制御プログラムで制御する ..... 10

■ 環境構築 .....	11
■ デバイス登録 .....	12

---

## デバイス制御プログラムの開発 ..... 15

■ 概要 .....	15
デバイス制御プログラムの構成 .....	15
動作シーケンス .....	16
サンプルプログラム .....	17
■ GGateway.dll が提供するメソッド .....	18
Open メソッド .....	18
Receive メソッド .....	19
Send メソッド .....	20
Close メソッド .....	20
■ Element オブジェクト .....	21
コンストラクター .....	21
getName メソッド .....	22
getValue メソッド .....	22
getType メソッド .....	22
getParent メソッド .....	23
haveChild メソッド .....	23
getChildrenNum メソッド .....	23
getChild メソッド .....	24
addChild メソッド .....	24

■ プログラム追加.....	25
■ デバイス登録 .....	27

---

## デバイス制御スクリプトで制御する .....28

■ デバイス登録 .....	29
キー入力デバイス.....	29
シリアル通信デバイス .....	31

---

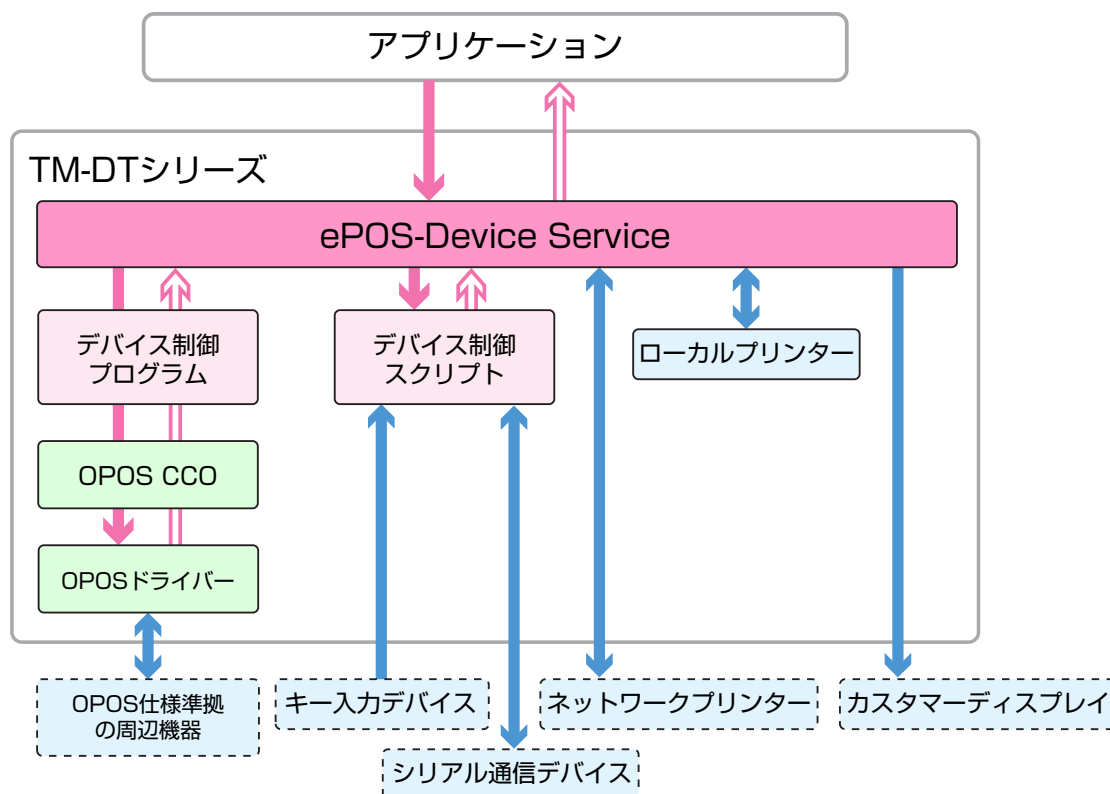
## デバイス制御スクリプトの開発 .....33

■ 概要.....	33
Epson ePOS SDK for JavaScript .....	33
ePOS-Device XML .....	34
デバイス制御スクリプトのオブジェクト .....	34
デバイス制御スクリプトのオブジェクトを使用する機能 .....	34
デバイス制御スクリプトの構成.....	35
■ デバイス制御スクリプト API 一覧.....	37
ClientConnection オブジェクト.....	37
DeviceConnection オブジェクト.....	37
デバイス制御スクリプト名オブジェクト .....	37
■ ClientConnection オブジェクト.....	38
send .....	38
■ DeviceConnection オブジェクト.....	39
send .....	39
■ デバイス制御スクリプト名オブジェクト.....	40
onDeviceData イベント（キー入力デバイス）.....	40
onDeviceData イベント（シリアル通信デバイス）.....	41
任意イベント.....	41
■ スクリプト追加.....	42
■ デバイス登録 .....	44

# 概要

TM-DT シリーズプリンター（TM-DT シリーズ）は、エプソン製の周辺機器の他に、エプソンが独自に開発したデバイス制御プログラムやデバイス制御スクリプトから、様々な周辺機器を制御できます。

TM-DT シリーズと各種周辺機器を制御するアプリケーションの開発については、Epson ePOS SDK や ePOS-Device XML のユーザーズマニュアルを参照してください。



## デバイス制御プログラムとは

ePOS-Device Service からの命令を周辺機器に送り、実行結果を返す実行ファイルです。



TM-T70II-DT/TM-T88V-DT は、TM-DT ソフトウェア Ver.4.0 以降の製品でデバイス制御プログラムを使用できます。

以下のカテゴリーで、OPOS Common Control Object (OPOS CCO) 1.14.001 と組み合わせて動作するドライバー（OPOS ドライバー）を持つ周辺機器を制御できます。

- 自動つり銭機
- 信用照会端末
- MSR
- POS キーボード
- バーコードスキャナー

制御方法は、[デバイス制御プログラムで制御する](#)を参照してください。

TM-DT シリーズではデバイス制御プログラム開発用の API も用意しています。

周辺機器の通信プロトコルに準じたプログラムを開発することで、OPOS 仕様準拠の周辺機器だけでなく、任意の周辺機器を制御できます。

開発方法は、[デバイス制御プログラムの開発](#)を参照してください。

## デバイス制御スクリプトとは

ePOS-Device Service からの命令を周辺機器に送り、実行結果を返す JavaScript ファイルです。

TM-DT シリーズに搭載され、対応するキー入力デバイスやシリアル通信デバイスを制御できます。

制御方法は、[デバイス制御スクリプトで制御する](#)を参照してください。

TM-DT シリーズではデバイス制御スクリプト開発用の API も用意しています。

周辺機器の通信プロトコルに準じたスクリプトファイルを開発することで、任意の周辺機器を制御できます。

開発方法は、[デバイス制御スクリプトの開発](#)を参照してください。

# 対象プリンターと周辺機器

## 対象プリンター

- TM-T70II-DT
- TM-T70II-DT2
- TM-T88V-DT
- TM-T88VI-DT2

## サポート周辺機器

以下の周辺機器を制御できます。

## エプソン製品

以下のエプソン製周辺機器を制御できます。

使用できる製品や制御方法は、各 TM-DT シリーズの詳細取扱説明書を参照してください。

- プリンター
- カスタマーディスプレイ

## デバイス制御プログラムで制御する周辺機器

以下のカテゴリで、OPOS CCO 1.14.001 と組み合わせて動作するドライバーを持つ周辺機器を制御できます。

- 自動つり銭機
- 信用照会端末
- MSR
- POS キーボード
- バーコードスキャナー

## デバイス制御スクリプトで制御する周辺機器

- キー入力デバイス
  - MSR（日立オムロン V3TU-FK）
  - キーボード（標準 HID）
  - バーコードスキャナー（標準 HID）
- シリアル通信デバイス
  - 自動つり銭機（グローリー RT-200/RAD-200）
  - シリアル通信デバイス
  - シリアル通信デバイス同等の制御が可能な USB デバイス



シリアル通信デバイス同等の制御が可能な USB デバイスを使用する場合、専用のドライバーをインストールする必要があります。  
ドライバーの仕様によっては、使用できない場合があります。



# 環境設定

TM-DT シリーズで周辺機器を制御するために、必要な初期設定について説明します。

## Windows の設定

TM-DT シリーズに搭載している Windows OS の初期設定をします。  
各 TM-DT シリーズの詳細取扱説明書を参照し、必要な初期設定をしてください。

## EPSON TMNet WebConfig

使用する周辺機器は、TM-DT シリーズに登録する必要があります。  
また、周辺機器ごとにデバイス制御プログラムやデバイス制御スクリプトの設定も必要です。  
これらの登録や設定には、TM-DT シリーズの EPSON TMNet WebConfig を使用します。

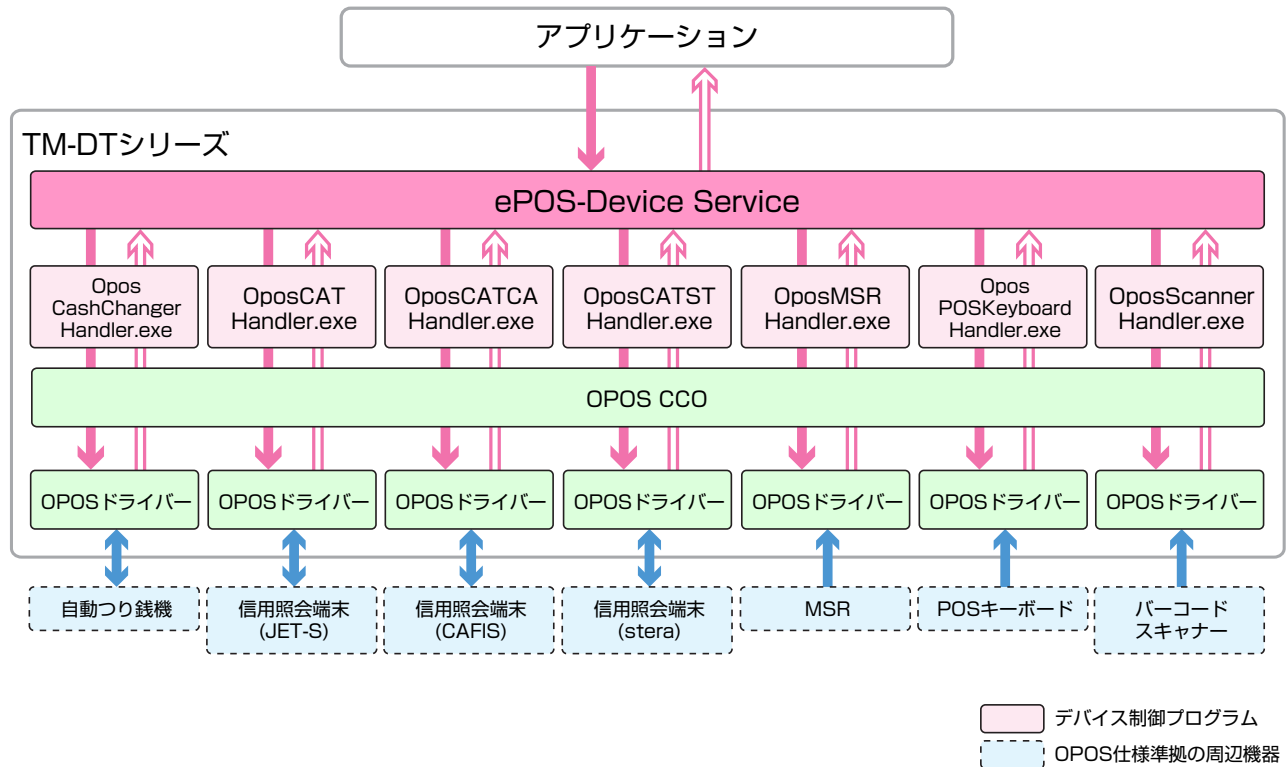
EPSON TMNet WebConfig については、TM-DT シリーズの詳細取扱説明書を参照してください。

# デバイス制御プログラムで制御する

TM-DT シリーズに搭載されているデバイス制御プログラムを使って、OPOS 仕様準拠の周辺機器を制御する方法を説明します。

TM-DT シリーズでは、周辺機器の 카테고리ごとにデバイス制御プログラムを用意しています。

OPOS CCO と使用する周辺機器向けの OPOS ドライバーを利用して、OPOS 仕様準拠の周辺機器を制御します。



## 環境構築

OPOS CCO と OPOS ドライバーを TM-DT シリーズにインストールする手順を説明します。

OPOS CCO は、Epson ePOS SDK パッケージで提供する OPOS\_CCOs\_1.14.001.msi を使用します。



評価・運用時に OPOS ログを確認する可能性があります。必要に応じて、リモートデスクトップ（リモートアクセス）の機能を有効にするなどの設定を行ってください。

### 1 TM-DT シリーズにログオンします。

TM-DT シリーズの詳細取扱説明書を参照してください。

### 2 OPOS ドライバーをインストールし、論理デバイス名を確認します。

OPOS ドライバー付属のマニュアルを参照してください。

複数の周辺機器を制御する場合は、制御する周辺機器ごとに OPOS ドライバーをインストールし、論理デバイス名を確認してください。

### 3 OPOS\_CCOs\_1.14.001.msi を実行し、OPOS CCO をインストールします。

OPOS\_CCOs\_1.14.001.msi のインストール画面の指示に従ってください。



OPOS CCO のインストールは、必ず OPOS ドライバーインストール後に実施してください。

手順が逆になった場合、OPOS ドライバーの持つ OPOS CCO に上書きされ、TM-DT シリーズから制御できなくなるおそれがあります。

環境構築後に制御する周辺機器を追加する場合も、OPOS ドライバーインストール後に、再度 OPOS\_CCO を上書きでインストールしてください。

### 4 OPOS ドライバーに付属のテストツールを使用して、周辺機器が正常に動作することを確認します。

OPOS ドライバー付属のマニュアルを参照し、必要に応じて実施してください。

# デバイス登録

使用する周辺機器を TM-DT シリーズに登録する手順を説明します。

- 1 EPSON TMNet WebConfig を起動します。  
TM-DT シリーズの詳細取扱説明書を参照してください。
- 2 「Web サービス設定」 - 「制御プログラム」 - 「デバイス登録」を選択し、デバイス登録画面を開きます。

The screenshot displays the EPSON TMNet WebConfig interface. The left sidebar shows the navigation menu with 'デバイス登録' (Device Registration) selected under the '制御プログラム' (Control Program) section. The main content area is titled 'デバイス登録' and contains a table for '登録するデバイス' (Devices to be registered). The table has two columns: '項目名' (Item Name) and '設定値' (Setting Value). The 'デバイスID' (Device ID) field is empty, and the '制御プログラム' (Control Program) dropdown is set to 'OposCashChangerHandler.exe'. Below this is a table for '登録済みデバイス' (Registered Devices) with columns for 'デバイスID' and '制御プログラム'.

### 3 以下を設定し、[ 追加 ] をクリックします。

制御する周辺機器ごとに設定してください。

デバイス ID には、OPOS ドライバーのインストール時に確認した論理デバイス名を指定します。

周辺機器	設定項目	設定値
自動つり銭機	デバイス ID	論理デバイス名
	制御プログラム	OposCashChangerHandler.exe
信用照会端末 (JET-S)	デバイス ID	論理デバイス名
	制御プログラム	OposCATHandler.exe
信用照会端末 (CAFIS)	デバイス ID	論理デバイス名
	制御プログラム	OposCATCAHandler.exe
信用照会端末 (stera)	デバイス ID	論理デバイス名
	制御プログラム	OposCATSTHandler.exe
MSR	デバイス ID	論理デバイス名
	制御プログラム	OposMSRHandler.exe
POS キーボード	デバイス ID	論理デバイス名
	制御プログラム	OposPOSKeyboardHandler.exe
バーコードスキャナー	デバイス ID	論理デバイス名
	制御プログラム	OposScannerHandler.exe



- OposCATCAHandler.exe を使う場合、論理デバイス名を OPOS インストール時から変更しないこと。
- OposCATCAHandler.exe を使い、信用照会端末の接続方式が有線 / 無線 LAN の場合、以下の場所に端末の IP アドレスおよびポート番号を設定すること。

`HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\EPSON\ePOS\CAT\CAFIS`

`Arch\IpAddress`

`HKEY_LOCAL_MACHINE\SOFTWARE\EPSON\WOW6432Node\ePOS\CAT\CAFIS Arch\Port`

制御する信用照会端末により、利用できる決済メディア種類が異なります。

決済メディア種類	値	JET-S	CAFIS	stera
クレジット	credit	✓	✓	✓
デビット	debit	✓	✓	-
銀聯	unionpay	✓	✓	✓
NFC Pay	nfcpayment	✓	✓	✓
楽天 Edy	edy	✓	✓	✓
iD	id	✓	✓	✓
nanaco	nanaco	✓	✓	✓
QUICpay	quicpay	✓	✓	✓
交通系 IC(Suica)	suica	✓	✓	✓
WAON	waon	✓	✓	✓

決済メディア種類	値	JET-S	CAFIS	stera
ポイント	point	-	✓	-
P i T a P a	pitapa	-	✓	✓
台湾金融カード	fisc	-	✓	-
QRコード決済	qr	-	-	✓
クレジット / デビット	credit_debit	✓	-	-
クレジット / デビット / NFC Pay	multi	✓	-	-
端末共通	common	-	-	✓

## 4 登録が完了すると、"登録済みデバイス" に登録情報が表示されます。

[削除] をクリックすると、該当する登録情報を削除します。

The screenshot shows the TMNet WebConfig interface. The left sidebar has a tree view with categories: 情報 (Information), 設定 (Settings), and Webコンテンツ (Web Content). Under 設定, there is a sub-menu 'デバイス管理' (Device Management) which includes 'デバイス登録' (Device Registration). The main content area is titled 'デバイス登録' and contains a form for adding new devices. The form has two input fields: 'デバイスID' and '制御プログラム' (with a dropdown menu). Below the form is a table titled '登録済みデバイス' (Registered Devices) with columns for 'デバイスID', '制御プログラム', and a '削除' (Delete) button for each row. The table lists several devices: OPOS\_CashChanger, OPOS\_Handler, OPOS\_MSR, OPOS\_POSKeyboard, and OPOS\_Scanner, each with its corresponding handler program.

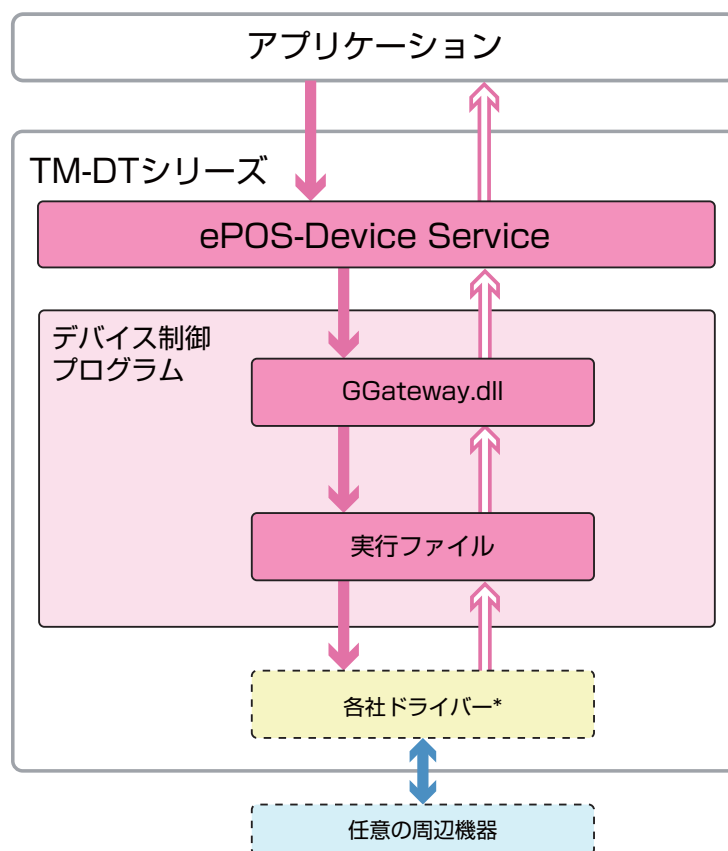
# デバイス制御プログラムの開発

デバイス制御プログラムの開発に必要な情報と、開発したデバイス制御プログラムを TM-DT シリーズに登録する方法を説明します。

## 概要

ePOS SDK から周辺機器との通信開始命令が実行されると、ePOS-Device Service はデバイス制御プログラムを起動します。

起動したデバイス制御プログラムで周辺機器を制御します。



\* 開発するデバイス制御プログラムに合わせ、必要に応じて用意します。

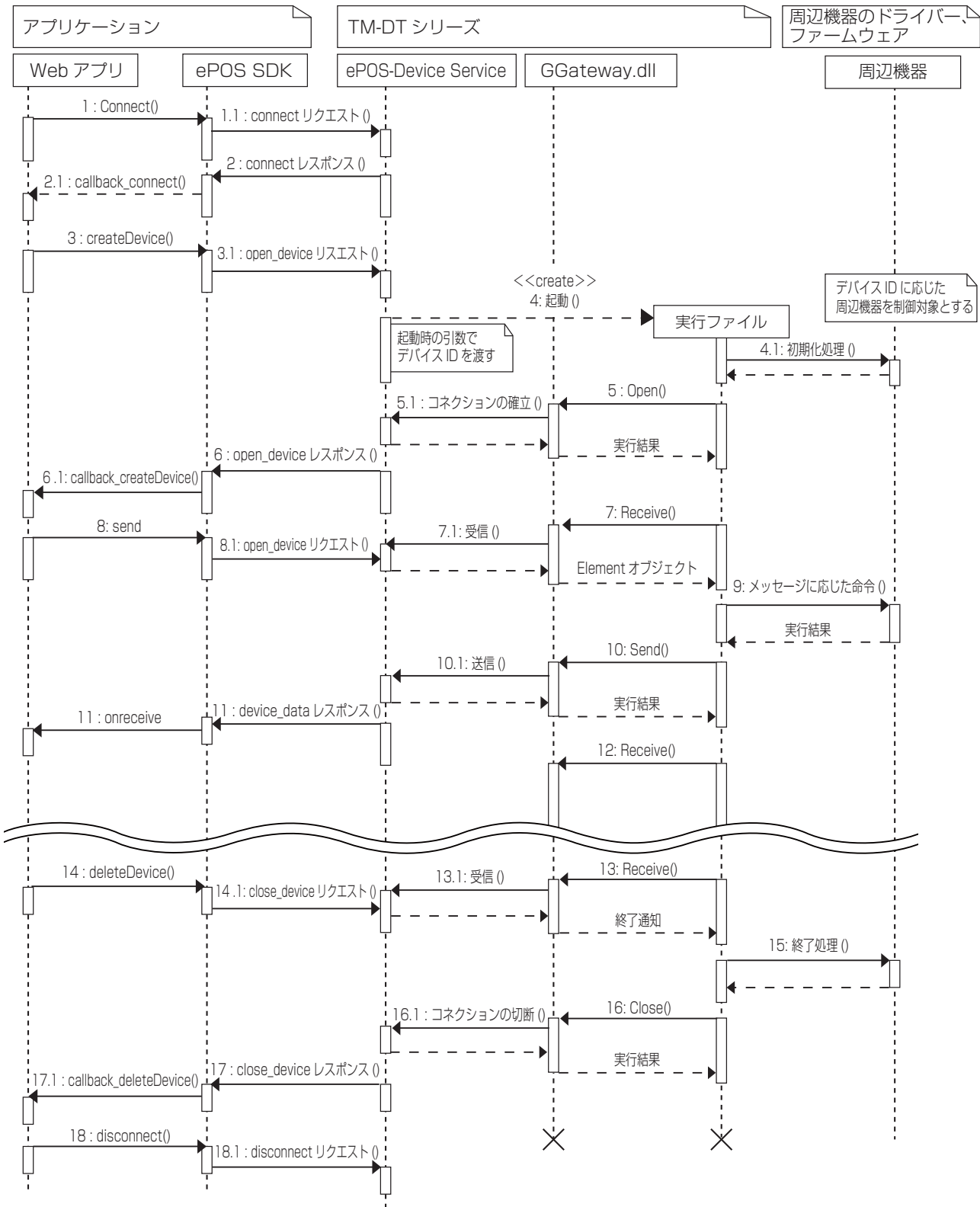
## デバイス制御プログラムの構成

デバイス制御プログラムは、以下の条件で構成してください。

- ❑ .exe 形式の実行ファイルを作成する。
- ❑ 1 つのデバイス制御プログラムに、複数の実行ファイルを含めない。
- ❑ GGateway.dll は、Epson ePOS SDK パッケージで提供するファイルを使用する。

動作シーケンス

ePOS-Device Service とデバイス制御プログラム間の動作シーケンスを以下に示します。  
Epson ePOS SDK を使って開発した Web アプリケーションから制御する場合のシーケンス図です。





サンプルプログラム

デバイス制御プログラム用のサンプルプログラムファイル (DeviceControlProgram\_Sample.zip) には、以下のファイルが含まれています。

ファイル名		説明
ExecuteFiles	GGateway.dll	デバイス制御プログラムが ePOS-Device Service とデータ送受信するための通信ライブラリーです。 開発する実行ファイルと組み合わせて使用します。
	Sample01.exe	実行ファイルのサンプルプログラムです。
	DevCtrlPrgTester.exe	開発したデバイス制御プログラムの動作確認用ツールです。
Sample01Project		Sample01.exe のソースファイル一式です。

# GGateway.dllが提供するメソッド

GGateway.dll は、実行ファイルと ePOS-Device Service がデータを送受信するためのメソッドを提供します。

## Open メソッド

Open 処理を実行します。処理完了後、処理結果を実行ファイルに返します。

### 構文

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

### パラメーター

設定値	説明
Port	ePOS-Device Service との通信ポート番号。 起動引数で渡された値を設定します。
DeviceID	使用する周辺機器のデバイス ID。 起動引数で渡された値を設定します。
XmlLog	受信 / 送信 XML ログを出力する / しないの設定 ( 個人情報に関わるデータを送受信する場合は false を指定してください。 )
SingleThreadApartment	デバイス制御プログラムが所属するアパートメントを設定。 true: SingleThreadApartment false: MultiThreadApartment

### 戻り値

戻り値	内容
GG_SUCCESS(0)	処理成功
GG_SOCKET_ERROR(-1)	ePOS-Device Service との通信でエラー発生
GG_INTERNAL_ERROR(-2)	内部エラー発生
GG_PARAM_ERROR(-3)	パラメーターエラー発生
GG_EPOS_SYS_ERROR(-4)	システムエラー発生
GG_ALREADY_OPENED(-5)	オープン済み

Receive メソッド

ePOS-Device Service からのデータを受信待機します。受信したコマンドを解析し、パラメーターにデータを格納します。処理完了後、処理結果を実行ファイルに返します。

構文

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

パラメーター

設定値	説明
MethodName	受信データの ePOS メソッド名を格納するポインタ
ReceiveData	受信データを格納する Element オブジェクトのポインタ
SequenceNo	受信データのシーケンス番号を格納するポインタ

戻り値

戻り値	内容
GG_SUCCESS(0)	処理成功
GG_SOCKET_ERROR(-1)	ePOS-Device Service との通信でエラー発生
GG_PARAM_ERROR(-3)	パラメーターエラー発生
GG_EPOS_SYS_ERROR(-4)	システムエラー発生
GG_RECV_DISCONNECT(-7)	ePOS-Device Service から切断要求受信
GG_NOT_OPENED(-8)	オープンしていない

## Send メソッド

ePOS-Device Service にデータを送信します。処理完了後、処理結果を実行ファイルに返します。

### 構文

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

### パラメーター

設定値	説明
EventName	ePOS イベント名
SendData	送信データオブジェクトのポインタ
SequenceNo	シーケンス番号

### 戻り値

戻り値	内容
GG_SUCCESS(0)	処理成功
GG_SOCKET_ERROR(-1)	ePOS-Device Service との通信でエラー発生
GG_PARAM_ERROR(-3)	パラメーターエラー発生
GG_EPOS_SYS_ERROR(-4)	システムエラー発生
GG_NOT_OPENED(-8)	オープンしていない

## Close メソッド

ePOS-Device Service とのクローズ処理を行います。処理完了後、処理結果を実行ファイルに返します。

### 構文

```
int Close();
```

### 戻り値

戻り値	内容
GG_SUCCESS(0)	処理成功
GG_SOCKET_ERROR(-1)	ePOS-Device Service との通信でエラー発生
GG_NOT_OPENED(-8)	オープンしていない

# Elementオブジェクト

実行ファイルと GGateway.dll 間の Receive メソッドと Send メソッドは、Element オブジェクトを使用します。Element オブジェクトの構造と、情報の取得や設定用のメソッドは以下の通りです。

## Element オブジェクトの構造

m_strName	Element の要素名
m_strValue	Element の要素値
m_strType	Element の Type( 文字列 / 数値 )
m_bArray	Element が配列かどうかの真偽値
m_children	子 Element のポインタ配列
m_parent	親 Element のアドレス ( 最上位の場合は NULL )

## コンストラクター

Element オブジェクトを生成します。4 つの形式で提供します。

### 構文

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

### パラメーター

設定値	説明
Name	Element オブジェクトに設定する要素名
Value	Element オブジェクトに設定する要素の文字列値 / 要素の数値

## getName メソッド

Element オブジェクトから要素名を取得します。

---

### 構文

```
char* getName()
```

---

### 戻り値

Element の要素名

## getValue メソッド

Element オブジェクトから要素値を文字列で取得します。

---

### 構文

```
char* getValue()
```

---

### 戻り値

Element の要素値

## getType メソッド

Element オブジェクトから要素値のタイプを文字列で取得します。

---

### 構文

```
char* getType()
```

---

### 戻り値

Element のタイプ

## getParent メソッド

Element オブジェクトの親エレメントのアドレスを取得します。

### 構文

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

### 戻り値

親 Element のポインタ

## haveChild メソッド

Element オブジェクトの子 Element 存在情報を取得します。

### 構文

```
bool haveChild();
```

### 戻り値

戻り値	内容
true	子 Element が存在する
false	子 Element が存在しない

## getChildrenNum メソッド

Element オブジェクトの子 Element 数を取得します。

### 構文

```
int getChildrenNum ();
```

### 戻り値

子 Element 数

## getChild メソッド

Element オブジェクトの子 Element を取得します。

### 構文

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

### パラメーター

設定値	説明
Num	取得する子 Element の要素番号
Name	取得する子 Element の要素名 ( 同じ名前の子 Element が存在する場合は最初に見つかった子 Element を取得 )

### 戻り値

戻り値	内容
子 Element のポインタ	パラメーターに一致する子 Element のポインタ
NULL	子 Element が存在しない

## addChild メソッド

Element オブジェクトに子 Element を追加します。

### 構文

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

### パラメーター

設定値	説明
Child	Element オブジェクトに追加する子 Element のポインタ

### 戻り値

戻り値	内容
true	処理に成功
false	処理に失敗



# プログラム追加

開発したデバイス制御プログラムを、TM-DT シリーズに追加する手順を説明します。

- 1 EPSON TMNet WebConfig を起動します。  
TM-DT シリーズの詳細取扱説明書を参照してください。
- 2 「Web サービス設定」 - 「制御プログラム」 - 「追加と削除」を選択し、制御プログラム画面を開きます。



- 3 [参照] をクリックし、GGateway.dll を選択します。

#### 4 [+] をクリックして行を追加し、[ 参照 ] から開発した実行ファイルを選択します。



開発したデバイス制御プログラムの構成に合わせ、必要なファイルを選択してください。

The screenshot shows the TMNet WebConfig interface. The left sidebar has a tree view with categories like '情報' (Information), '設定' (Settings), and 'デバイス管理' (Device Management). The '制御プログラム' (Control Program) section is active, showing a table for '登録する制御プログラム' (Registered Control Programs). This table has two rows: '制御プログラムファイル' (Control Program File) with values 'C:\ePOS-Device\GGateway.dll' and 'C:\ePOS-Device\Sample01.exe'. Below this is a table for '登録済み制御プログラム' (Registered Control Programs) listing several handlers with '詳細表示' (Show Details) and '削除' (Delete) buttons.

#### 5 [ 追加 ] をクリックします。

#### 6 追加が完了すると、" 登録済み制御プログラム " に登録情報が表示されます。

[ 詳細情報 ] をクリックすると、デバイス制御プログラムの構成ファイルを参照できます。

[ 削除 ] をクリックすると、該当する登録情報を削除します。

# デバイス登録

開発したデバイス制御プログラムで制御する周辺機器を、TM-DT シリーズに登録する手順を説明します。

- 1 EPSON TMNet WebConfig を起動します。  
TM-DT シリーズの詳細取扱説明書を参照してください。
- 2 「Web サービス設定」 - 「制御プログラム」 - 「デバイス登録」を選択し、デバイス登録画面を開きます。

項目名	設定値
デバイスID	local_other
制御プログラム	Sample01.exe

追加

デバイスID	制御プログラム	
OPOS_CashChanger	OposCashChangerHandler.exe	削除
OPOS_Handler	OposCATHandler.exe	削除
OPOS_MSR	OposMSRHandler.exe	削除
OPOS_POSkeyboard	OposPOSKeyboardHandler.exe	削除
OPOS_Scanner	OposScannerHandler.exe	削除

- 3 以下を設定し、[ 追加 ] をクリックします。  
制御する周辺機器ごとに設定してください。

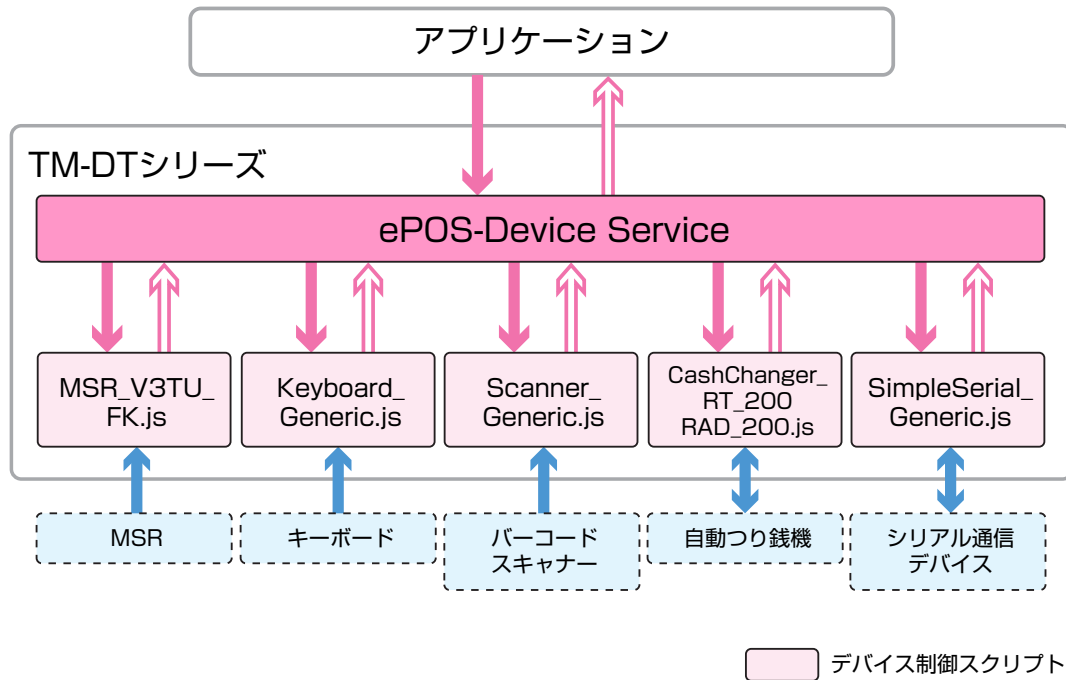
設定項目	設定値
デバイス ID	任意のデバイス ID
制御プログラム	プログラム追加で追加したデバイス制御プログラム

- 4 登録が完了すると、" 登録済みデバイス " に登録情報が表示されます。  
[ 削除 ] をクリックすると、該当する登録情報を削除します。

# デバイス制御スクリプトで制御する

TM-DT シリーズに搭載されているデバイス制御スクリプトを使って、キー入力デバイスやシリアル通信デバイスを制御する方法を説明します。

TM-DT シリーズでは、周辺機器ごとにデバイス制御スクリプトを用意しています。  
デバイス制御スクリプトに対応した周辺機器を制御できます。



# デバイス登録

使用する周辺機器を TM-DT シリーズに登録する手順を説明します。

## キー入力デバイス

- 1 使用する周辺機器を TM-DT シリーズに接続します。
- 2 EPSON TMNet WebConfig を起動します。  
TM-DT シリーズの詳細取扱説明書を参照してください。
- 3 「Web サービス設定」 - 「デバイス登録」 - 「キー入力デバイス」を選択し、デバイス登録画面を開きます。

TMNet WebConfig EPSON

Home Help About

情報

- Web サービス設定
  - 現在の状態
  - 環境設定
  - ネットワーク
  - 日付と時刻
- 設定
  - [Web サービス設定]
    - 起動設定
      - 起動設定
    - デバイス管理
      - デバイス登録
        - プリンター
        - カスタマーディスプレイ
        - キー入力デバイス
        - シリアル通信デバイス
        - その他
      - 制御スクリプト
        - 追加と削除
      - 制御プログラム
        - デバイス登録
          - 追加と削除
      - 印刷設定
        - スプーラー
      - Web コンテンツ
        - 更新設定
      - サーバーアクセス

キー入力デバイス

登録するキー入力デバイス

項目名	設定値
デバイスID	
デバイス名	: USB Keyboard(vid:04CA pid:004B)
制御スクリプト	Epson_Sample.js

追加

登録済みキー入力デバイス

デバイスID	デバイス名	VID / PID	制御スクリプト
--------	-------	-----------	---------

## 4 使用する周辺機器ごとに以下を設定し、[ 追加 ] をクリックします。

周辺機器	設定項目	設定値
MSR	デバイス ID	任意のデバイス ID
	デバイス名	リストから該当するデバイスを選択します。
	制御スクリプト	MSR_V3TU_FK.js
キーボード	デバイス ID	任意のデバイス ID
	デバイス名	リストから該当するデバイスを選択します。
	制御スクリプト	Keyboard_Generic.js
バーコード スキャナー	デバイス ID	任意のデバイス ID
	デバイス名	リストから該当するデバイスを選択します。
	制御スクリプト	Scanner_Generic.js

## 5 登録が完了すると、" 登録済みデバイス " に登録情報が表示されます。 [ 削除 ] をクリックすると、該当する登録情報を削除します。

The screenshot shows the TMNet WebConfig interface for registering a key input device. The left sidebar has a '設定' (Settings) section with a 'デバイス管理' (Device Management) link. The main content area is titled 'キー入力デバイス' and contains a form to register a new device. The form has fields for 'デバイスID' (Device ID), 'デバイス名' (Device Name), and '制御スクリプト' (Control Script). Below the form is a table of registered devices. The table has columns for 'デバイスID', 'デバイス名', 'VID / PID', and '制御スクリプト'. One device is listed: 'local\_keyboard' with name 'USB Keyboard', VID / PID '04CA / 004B', and script 'Keyboard\_Generic.js'. There are buttons for '追加' (Add), '動作テスト' (Test Operation), and '削除' (Delete).

項目名	設定値
デバイスID	<input type="text"/>
デバイス名	登録可能なデバイスがありません
制御スクリプト	Keyboard_Generic.js

追加

デバイスID	デバイス名	VID / PID	制御スクリプト		
local_keyboard	USB Keyboard	04CA / 004B	Keyboard_Generic.js	動作テスト	削除

## シリアル通信デバイス

- 1 使用する周辺機器を TM-DT シリーズに接続します。
- 2 EPSON TMNet WebConfig を起動します。  
TM-DT シリーズの詳細取扱説明書を参照してください。
- 3 「Web サービス設定」 - 「デバイス登録」 - 「シリアル通信デバイス」を選択し、デバイス登録画面を開きます。

TMNet WebConfig EPSON

Home Help About

情報

Web サービス設定

▶ 現在の状態

▶ 環境設定

▶ ネットワーク

▶ 日付と時刻

設定

【Web サービス設定】

▶ 起動設定

▶ 起動設定

デバイス管理

▶ デバイス登録

▶ プリンター

▶ カスタマーディスプレイ

▶ キー入力デバイス

▶ シリアル通信デバイス

▶ その他

制御スクリプト

▶ 追加と削除

制御プログラム

▶ デバイス登録

▶ 追加と削除

印刷設定

▶ スプーラー

Web コンテンツ

▶ 更新設定

サーバーアクセス

シリアル通信デバイス

登録するシリアル通信デバイス

項目名	設定値
デバイスID	<input type="text"/>
デバイス名	<input type="radio"/> 製品選択 <input type="text" value="登録可能なデバイスがありません"/> <input checked="" type="radio"/> ポート選択 <input type="text" value="物理シリアルポート"/> <a href="#">ポート位置の表示</a>
制御スクリプト	<input type="text" value="CashChanger_RT_200RAD_200.js"/>
通信速度(bps)	<input type="text" value="9600"/>
データビット	<input type="text" value="8"/>
パリティ	<input type="text" value="なし"/>
ストップビット	<input type="text" value="1"/>
フロー制御	<input type="text" value="なし"/>

追加

登録済みシリアル通信デバイス

デバイスID	デバイス名	VID / PID	制御スクリプト

## 4 使用する周辺機器ごとに以下を設定し、[ 追加 ] をクリックします。

周辺機器	設定項目	設定値
自動つり銭機	デバイス ID	任意のデバイス ID
	デバイス名 *1	リストから該当するデバイス、もしくは使用するポートを選択します。
	ポート *2	使用するポートを選択します。
	制御スクリプト	CashChanger_RT_200RAD_200.js
シリアル通信デバイス	デバイス ID	任意のデバイス ID
	デバイス名 *1	リストから該当するデバイス、もしくは使用するポートを選択します。
	ポート *2	使用するポートを選択します。
	制御スクリプト	SimpleSerial_Generic.js

\*1： TM-T70II-DT/TM-T88V-DT の場合

\*2： TM-T70II-DT2/TM-T88VI-DT2 の場合

通信速度 (bps)、データビット、パリティ、ストップビット、フロー制御は、使用する周辺機器に合わせて設定します。

## 5 登録が完了すると、" 登録済みデバイス " に登録情報が表示されます。

[ 削除 ] をクリックすると、該当する登録情報を削除します。

The screenshot shows the TMNet WebConfig interface. On the left is a navigation menu with sections like '情報' (Information), '設定' (Settings), and '印刷設定' (Print Settings). The '設定' section is expanded, showing 'Webサービス設定' (Web Service Settings) and 'デバイス管理' (Device Management). Under 'デバイス管理', 'デバイス登録' (Device Registration) is selected. The main area displays the '登録するシリアル通信デバイス' (Register Serial Communication Device) form. This form includes fields for 'デバイスID', 'デバイス名' (with radio buttons for '製品選択' and 'ポート選択'), '制御スクリプト', '通信速度(bps)', 'データビット', 'パリティ', 'ストップビット', and 'フロー制御'. Below the form is a table titled '登録済みシリアル通信デバイス' (Registered Serial Communication Devices) with columns for 'デバイスID', 'デバイス名', 'VID / PID', and '制御スクリプト'. The table contains one entry: 'local\_serial', '物理シリアルポート', '-', and 'SimpleSerial\_Generic.js'. At the bottom of the table are buttons for '詳細表示' (Show Details), '動作テスト' (Run Test), and '削除' (Delete).



# デバイス制御スクリプトの開発

デバイス制御スクリプトの開発に必要な情報と、開発したデバイス制御スクリプトを TM-DT シリーズに登録する方法を説明します。

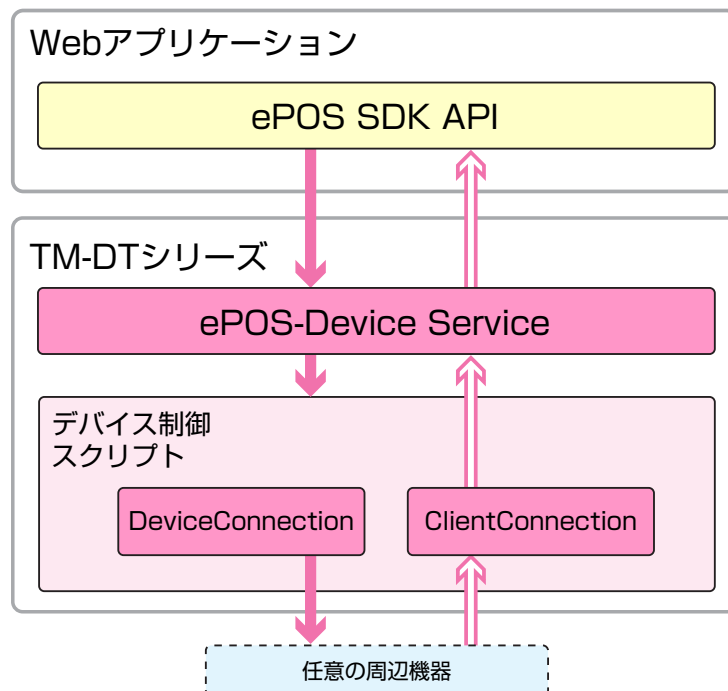
開発したデバイス制御スクリプトを使った周辺機器制御は、Epson ePOS SDK for JavaScript と ePOS-Device XML で対応しています。

## 概要

### Epson ePOS SDK for JavaScript

ePOS SDK API の createDevice メソッドが実行されると、ePOS-Device Service はデバイス制御スクリプトのオブジェクトを生成します。

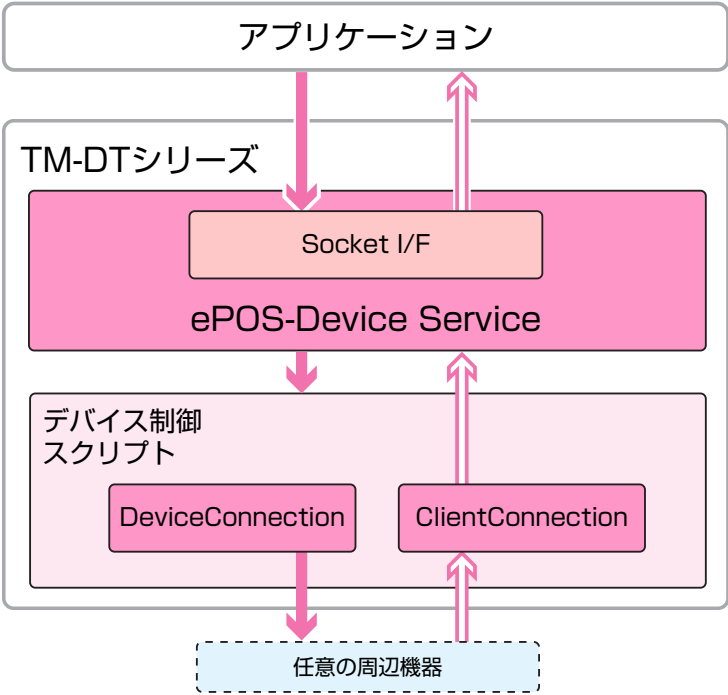
生成したオブジェクトで周辺機器を制御します。



## ePOS-Device XML

open\_device メッセージが送信されると、ePOS- Device Service はデバイス制御スクリプトのオブジェクトを生成します。

生成したオブジェクトで周辺機器を制御します。



## デバイス制御スクリプトのオブジェクト

デバイス制御スクリプトには、ePOS-Device Service から以下のオブジェクトが渡されます。これらのオブジェクトを使用して、デバイス制御スクリプトはアプリケーションおよび周辺機器と通信します。

オブジェクト	説明
ClientConnection	アプリケーション側のデバイスオブジェクトにデータを送信するためのオブジェクト
DeviceConnection	周辺機器とデータを送受信するためのオブジェクト

## デバイス制御スクリプトのオブジェクトを使用する機能

デバイス制御スクリプト用 API を利用して以下の機能を使用できます。

- ❑ アプリケーション側デバイスオブジェクトの任意のイベント呼び出し
- ❑ 周辺機器へのデータ送信
- ❑ 周辺機器で発生したデータの受信

## デバイス制御スクリプトの構成

デバイス制御スクリプトは、JavaScript で開発します。以下の条件でコーディングしてください。

- ❑ デバイス制御に必要なコードは、1 つのファイルにまとめて記述する。  
(EPSON TMNet WebConfig を使用する場合、デバイス設定は 1 ファイルしか登録できません。)
- ❑ ファイル名の最初のドット "." までの名称と、コンストラクターの名称を同じにする。  
例) ファイル名 : Keyboard\_Generic.ver1.0.js -> コンストラクターの名称 : Keyboard\_Generic
- ❑ コンストラクターの外部参照のために、exports 宣言する。  
例) exports.Keyboard\_Generic = Keyboard\_Generic;
- ❑ コンストラクターに 2 つの引数を持たせる。
- ❑ デバイス制御スクリプトには、以下のプロパティーが必要。コンストラクターで適切な名称を設定する。
  - DEVICE\_TYPE プロパティー (オブジェクト種類 : String)

設定値	説明
type_cash_changer	自動つり銭機を使用する場合に指定します。
type_keyboard	キーボードデバイスを使用する場合に指定します。
type_msr	MSR を使用する場合に指定します。
type_scanner	バーコードスキャナーを使用する場合に指定します。
type_simple_serial	単純なシリアル通信する場合に指定します。

- DEVICE\_GROUP プロパティー (オブジェクト種類 : String)

設定値	説明
group_hid	HID ドライバーで動作可能なキー入力デバイスを使用する場合に指定します。
group_serial	シリアル通信デバイスを使用する場合に指定します。
group_other	その他の周辺機器を使用する場合に指定します。

- ❑ 周辺機器で発生したデータを受け取るため onDeviceData メソッドを用意する。  
詳細は、[デバイス制御スクリプト名オブジェクト](#)を参照してください。
- ❑ アプリケーション側で動作するデバイスオブジェクトのメソッドに対応するメソッドを用意する。  
詳細は、[任意イベント](#)を参照してください。

## デバイス制御スクリプトの構成例

```
// exports 宣言
exports.Keyboard_Generic = Keyboard_Generic;

// 2つの引数を持ち、ファイル名と同名
function Keyboard_Generic(clientConn, deviceConn){
  // DEVICE_TYPE プロパティを持つ
  this.DEVICE_TYPE = 'type_keyboard';
  // DEVICE_GROUP プロパティを持つ
  this.DEVICE_GROUP = 'group_hid';
  this.clientConn = clientConn;
  this.deviceConn = deviceConn;
  .....
  .....
}

Keyboard_Generic.prototype = {
  // onDeviceData メソッドを持つ
  onDeviceData : function(event, keycode, ascii){...},
  // デバイスオブジェクト対応したメソッドがある
  setprefix : function(data){...}
}
```

# デバイス制御スクリプト API 一覧

デバイス制御スクリプト API には、以下のオブジェクトが用意されています。

- ❑ [ClientConnection](#) オブジェクト
- ❑ [DeviceConnection](#) オブジェクト
- ❑ [デバイス制御スクリプト名オブジェクト](#)

## ClientConnection オブジェクト

デバイス制御スクリプトのコンストラクターの第 1 パラメーターに渡されるオブジェクトです。

API		説明
送信	<a href="#">send</a>	Web ブラウザーで動作するデバイスオブジェクトにデータを送信

## DeviceConnection オブジェクト

デバイス制御スクリプトのコンストラクターの第 2 パラメーターに渡されるオブジェクトです。

API		説明
送信	<a href="#">send</a>	シリアル通信デバイスにデータを送信

## デバイス制御スクリプト名オブジェクト

周辺機器からのデータ受信用のオブジェクトです。

API		説明
結果受信	<a href="#">onDeviceData</a> イベント (キー入力デバイス)	キー入力デバイスからのデータ受信イベント
	<a href="#">onDeviceData</a> イベント (シリアル通信デバイス)	シリアル通信デバイスからのデータ受信イベント
	<a href="#">任意イベント</a>	Web ブラウザーで動作するデバイスオブジェクトの API 実行結果受信イベント

# ClientConnection オブジェクト

## send

Web ブラウザーで動作するデバイスオブジェクトにデータを送信します。

### 構文

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

### パラメーター

event

設定値	説明
String	デバイスオブジェクトのイベント名を指定

data

設定値	説明
Object	デバイスオブジェクトのイベントに渡すデータを指定

### サンプルプログラム

デバイスオブジェクトの onkeypress イベントを呼び出し、onkeypress イベントの data パラメーターを使用して data.keycode から 49、data.ascii から '1' を取得する。

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

# DeviceConnection オブジェクト

## send

シリアル通信デバイスにデータを送信します。

### 構文

```
send(data);
```

### パラメーター

data

設定値	説明
Buffer	周辺機器に送信するデータを指定

### 補足説明

HID ドライバーで動作可能な入力デバイスには送信できません。

# デバイス制御スクリプト名オブジェクト

## onDeviceData イベント(キー入力デバイス)

HID ドライバーで動作可能なキー入力デバイスから検出されたデータを受信するイベントです。  
キー入力デバイス用のデバイス制御スクリプトの場合、本形式のイベントを記述してください。

### 構文

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

### パラメーター

event

キー操作の方向を受け取ります。

値	説明
1	キーダウン
2	キーアップ

keycode

値	説明
Number	キーコード

ascii

値	説明
String	操作されたキーに対応する文字

### 補足説明

- ❑ keycode で受け取る値は、Epson ePOS SDK for JavaScript または ePOS-Device XML のユーザーズ マニュアルを参照してください。
- ❑ ascii で受け取ったキーコードに対応する文字が無い場合、undefined が入ります。



## onDeviceData イベント(シリアル通信デバイス)

シリアル通信デバイスから検出されたデータを受信するイベントです。  
シリアル通信デバイス用のデバイス制御スクリプトの場合、本形式のイベントを記述してください。

### 構文

```
onDeviceData(data);
```

### パラメーター

data

値	説明
Buffer	シリアル通信デバイスから受信したデータ

## 任意イベント

Web ブラウザーで動作するデバイスオブジェクトの API 実行結果を受信するイベントです。

### 構文

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

### パラメーター

data

値	説明
Object	デバイスオブジェクトの callEvent メソッドで パラメーター指定したオブジェクト

### 補足説明

callEvent メソッドについては、Epson ePOS SDK for JavaScript または ePOS-Device XML のユーザーズマニュアルを参照してください。

# スクリプト追加

開発したデバイス制御スクリプトを、TM-DT シリーズに追加する手順を説明します。

- 1 EPSON TMNet WebConfig を起動します。  
TM-DT シリーズの詳細取扱説明書を参照してください。
- 2 「Web サービス設定」 - 「制御スクリプト」 - 「追加と削除」を選択し、制御スクリプト画面を開きます。

The screenshot displays the 'Control Scripts' management interface. On the left, a sidebar lists various configuration options, with 'Control Scripts' selected. The main content area is divided into two sections. The top section, 'Register Control Scripts', provides a form to add new scripts, including fields for 'Item Name' and 'Script Value', and a 'Reference...' button. The bottom section, 'Registered Control Scripts', lists existing scripts organized by device type. For example, under 'Keyboard Device', there are entries for 'type\_keyboard' (Keyboard\_Generic.js, status: '使用中'), 'type\_msr' (MSR\_V3TU\_FK.js, status: '削除'), and 'type\_scanner' (Scanner\_Generic.js, status: '削除'). Similar entries are shown for 'Serial Communication Device' and 'Other'.

- 3 [ 参照 ] から追加するデバイス制御スクリプトを選択し、[ 追加 ] をクリックします。

- 4 追加が完了すると、" 登録済み制御スクリプト " に登録情報が表示されます。  
DEVICE\_TYPE プロパティと DEVICE\_GROUP プロパティで指定した設定値に応じて、自動で分類されます。  
[ 削除 ] をクリックすると、該当する登録情報を削除します。

TMNet  
WebConfig

EPSON

Home

Help

About

情報

Webサービス設定

▶現在の状態

環境設定

▶ネットワーク

▶日付と時刻

設定

[Webサービス設定]

起動設定

▶起動設定

デバイス管理

デバイス登録

▶プリンター

▶カスタマーディスプレイ

▶キー入力デバイス

▶シリアル通信デバイス

▶その他

制御スクリプト

▶追加と削除

制御プログラム

▶デバイス登録

▶追加と削除

印刷設定

▶スプーラー

Webコンテンツ

▶更新設定

サーバーアクセス

制御スクリプト

スクリプトファイルを登録しました。

登録する制御スクリプト

項目名	設定値
制御スクリプト	<input type="text"/> 参照...

追加

登録済み制御スクリプト

キー入力デバイス

デバイスタイプ	制御スクリプト	
type_keyboard	Keyboard_Generic.js	使用中
type_keyboard	Epson_Sample.js	削除
type_msr	MSR_V3TU_FK.js	削除
type_scanner	Scanner_Generic.js	削除

シリアル通信デバイス

デバイスタイプ	制御スクリプト	
type_cash_changer	CashChanger_RT_200RAD_200.js	削除
type_simple_serial	SimpleSerial_Generic.js	使用中

# デバイス登録

開発したデバイス制御スクリプトで制御する周辺機器を、TM-DT シリーズに登録する手順を説明します。

- 1 使用する周辺機器を TM-DT シリーズに接続します。
- 2 EPSON TMNet WebConfig を起動します。  
TM-DT シリーズの詳細取扱説明書を参照してください。
- 3 「Web サービス設定」 - 「デバイス登録」 から、使用する周辺機器のデバイス登録画面を開きます。

The screenshot shows the EPSON TMNet WebConfig interface. The left sidebar has a menu with 'Web サービス設定' (Web Service Settings) expanded, showing '起動設定' (Startup Settings) and 'デバイス管理' (Device Management). Under 'デバイス管理', 'デバイス登録' (Device Registration) is selected. The main content area is titled 'キー入力デバイス' (Keyboard Input Device). It contains a form for adding a new device with the following fields:

- 項目名 (Item Name): 設定値 (Setting Value)
- デバイスID (Device ID): [Empty text box]
- デバイス名 (Device Name): [Dropdown menu showing 'USB Keyboard(vid.04CA pid.004B)']
- 制御スクリプト (Control Script): [Dropdown menu showing 'Epson\_Sample.js']

Below the form is a button labeled '追加' (Add). At the bottom, there is a table titled '登録済みキー入力デバイス' (Registered Keyboard Input Devices) with columns for 'デバイスID', 'デバイス名', 'VID / PID', and '制御スクリプト'.

- 4 以下を設定し、[ 追加 ] をクリックします。  
制御する周辺機器ごとに設定してください。

設定項目	設定値
デバイス ID	任意のデバイス ID
デバイス名	リストから該当するデバイス、もしくは使用するポートを選択します。
制御スクリプト	スクリプト追加で追加したデバイス制御スクリプト

5 登録が完了すると、" 登録済みデバイス " に登録情報が表示されます。  
[ 削除 ] をクリックすると、該当する登録情報を削除します。

TMNet  
WebConfig

EPSON

HomeHelpAbout

情報

Webサービス設定

現在の状態

環境設定

ネットワーク

日付と時刻

設定

Webサービス設定

起動設定

起動設定

デバイス管理

デバイス登録

プリンター

カスタマーディスプレイ

キー入力デバイス

シリアル通信デバイス

その他

制御スクリプト

追加と削除

制御プログラム

デバイス登録

追加と削除

印刷設定

スプーラー

Webコンテンツ

更新設定

サーバーアクセス

キー入力デバイス

設定は正常に更新されました。

登録するキー入力デバイス

項目名	設定値
デバイスID	
デバイス名	登録可能なデバイスがありません
制御スクリプト	Epson_Sample.js

追加

登録済みキー入力デバイス

デバイスID	デバイス名	VID / PID	制御スクリプト		
Epson_sample	USB Keyboard	04CA / 004B	Epson_Sample.js	動作テスト	削除