FUjitsu

WS-POS 環境セットアップ・ アプリケーション開発ガイド

富士通アイソテック株式会社

Revision 1.0.0.0

目次

1. 概	要	3
1.1	環境構築に必要なソフトウェア	3
1.2	機器構成について	4
1.3	商標について	4
っ 理	倍ちットマップト恐ウについて	Б
Z.	現セットアッノと設定について・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	9
2.1	インストール	5
2.1	1.1 .NET Framework	. 5
2.1	1.2 OPOS ドライバ	. 6
2.1	1.3 Common Control Object (共通CO)	. 8
2.1	1.4 POS for .NET	. 9
2.1	1.5 WS-POS 環境	12
2.1	1.6 その他の設定	13
2.2	インストール確認	14
2.3	実デバイスを用いた実行のための設定(POS プリンタ)	15
2.4	実デバイスでの実行(コンシューマとプロバイダが同一 PC)	17
2.5	実デバイスでの実行(コンシューマとプロバイダが別 PC)	17
2.6	WS-POS サービスプロバイダを Windows Service で実行	19
2.7	サービスプロバイダの各種設定	20
2.7	7.1 Behavior 名、デバイス名の変更	20
2.7	7.2 サービスプロバイダ URL の変更	20
2.7	7.3 endpoint の変更	21
2.7	7.4 セッションタイムアウトの確認間隔の変更	22
2.7	7.5 通信ポートの最大同時接続数の変更	22
2.7	7.6 使用する.NET Framework バージョンの変更	22
2.8	サービスコンシューマの各種設定	23
2.8	3.1 endpoint の変更	23
2.8	3.2 サービスコンシューマ URL の変更	24
3. WS	-POS アプリケーションの作成	25
3.1	開発環境	25
3.2	開発言語	25
3.3	作成手順	25
3.3	3.1 新規プロジェクトの作成	25
3.3	3.2 WSPOSContract の追加	25
3.3	3.3 System.ServiceModel の追加	25
3.5	3.4 アプリケーション構成ファイルの追加	26
3.3	3.5 アプリケーション 構成ファイルの編集	27
3.3	3.6 アプリケーションソースの編集	28
3.3	3.7 例外処理	30
3.3	3.8 イベント処理-SelfHost 方式	31
3.3	3.9 イベント処理-LongPolling 方式	35
3.3	3.10 KeepAlive 処理	39
3.4	OPOS のメソッド・プロパティについて	41
3.5	WS-POS で使用できないメソッド・プロパティ	41

4.	注意事項	42
5.	改訂履歴	43

1. 概要

本ドキュメントは、FIT 製 OPOSドライバを利用した WS-POS 環境のセットアップ方法とアプリケーションの開発方法について記述します。

本ドキュメントでは、下記の表 1-1 の環境を例として解説しています。環境が異なる場合には、適宜読み替えてください。 また、ダウンロード URL は 2014 年 6 月時点のものです。

WS-POS(Web Services for Point of Service)は、Web ベースの POS システムでデバイス制御を行うものです。 WS-POS の仕様については、下記 URLを参照してください。

https://nrf.com/resources/retail-technology-standards/unifiedpos

1.1 環境構築に必要なソフトウェア

WS-POS 構築については以下のソフトウェアが必要となります。

種類	説明
WS-POS 参照実装	WS-POS サービスの参照実装です。
	以下からダウンロードします。
	https://nrf.com/resources/retail-library/ws-pos-version-12-reference-implementation-c
	日本語のドキュメント等は下記 URL を参照してください。
	http://www.microsoft.com/ja-jp/business/industry/retail/wspos/default.aspx
Microsoft Point of Service	WS-POS が利用します。
for .NET v1.12	以下からダウンロードします。
(POS for .NET 1.12)	http://www.microsoft.com/en-us/download/details.aspx?id=5355
Common Control Object 1.12	上記モジュールが参照する CO です。
	以下からダウンロードします。
	http://monroecs.com/oposccos_current.htm
FP-1100 OPOSドライバ	FP-1100 OPOSドライバー式
	以下からダウンロードします。
	http://jp.fujitsu.com/group/fit/services/printers/downloads/driver/fp1100/#FP-11000POS
.NET Framework	WS-POS の動作には、Microsoft .NET Framework 3.5 以降が必要になります。
	Microsoft のサイトからダウンロードしてください。
	•Microsoft .NET Framework 3.5
	http://www.microsoft.com/ja-jp/download/details.aspx?id=21
	Microsoft .NET Framework 3.5 Service Pack 1
	http://www.microsoft.com/ja-jp/download/details.aspx?id=25150
	•Microsoft .NET Framework 4.0
	http://www.microsoft.com/ja-jp/download/details.aspx?id=17718
	•Microsoft .NET Framework 4.5
	http://www.microsoft.com/ja-jp/download/details.aspx?id=30653

表 1-1 環境構築に必要なソフトウェア

1.2 機器構成について

クライアント PC とサーバーPC の機能配置について、以下に示します。 クライアント PC とサーバーPC は LAN/WAN で通信できる状態としてください。



図1-1 構成図

1.3 商標について

記載されている会社名、製品名は各社の登録商標または商標です。

2. 環境セットアップと設定について

本章では、WS-POS 環境セットアップについて説明します。 ※セットアップは管理者権限で行ってください。

2.1 インストール

以下の手順で、WS-POS 環境に必要なソフトウェアをインストールします。WS-POS サービスプロバイダを構築する PC(サーバー側)にインストールします。

2.1.1 .NET Framework

WS-POSの動作に .NET Framework 3.5以降が必要になります。 Windows7 以降の OS では、有効化(下記参照)することでインストールされ使用可能になります。

手順	内容	詳細
1	.NET Framework 3.5 以降	.NET Framework 3.5以降がインストールされていない場合、
	のダウンロード	以下のURLからダウンロードしてインストールしてください。
		•Microsoft .NET Framework 3.5
		http://www.microsoft.com/ja-jp/download/details.aspx?id=21
		•Microsoft .NET Framework 3.5 Service Pack 1
		http://www.microsoft.com/ja-jp/download/details.aspx?id=25150
		•Microsoft .NET Framework 4.0
		http://www.microsoft.com/ja-jp/download/details.aspx?id=17718
		 Microsoft .NET Framework 4.5
		http://www.microsoft.com/ja-jp/download/details.aspx?id=30653
2	.NET Framework の有効化	コントロールパネル > プログラムと機能 > Windows の機能の有効かまたは
		無効化 を選択して以下の項目にチェックを入れます。
		🔤 Windows の機能 - 🗆 🗙
		WINDOWS の機能の有効化よには無効化
		機能を有効にするには、チェック ポックスをオンにしてください。機能を無効にするには、チェック ポックスをオフ
		にしていたという塗りつかられたデエックパックスは、機能の一部が有効になっていることを改むより。
		■ ■ NET Framework 2.5 (NET 2.0 t) たい 2.0 t をおい
		Windows Communication Foundation #TTP アクティブル Windows Communication Foundation 非 HTTP アクティブル
		INET Framework 4.3 Advanced Services
		ASP.NET 4.5
		」 メッセージ キュー (MSMQ) アクティブ化
		□ □ 名前付きパイプのアクティブ化
		Active Directory ライトウェイト ディルクトリ サービス
		OK キャンセル
		Windows Communication Foundation LITTO 7/1- /-
		・Windows Communication Foundation 非日日アクティノ
		・HTTP アクティフ化
I	A CONTRACT OF	

2.1.2 OPOS ドライバ

ここでは、FP-1100 OPOSドライバを例に説明します。

手順	内容	詳細			
1	OPOS ドライバ	<32bit OSの場合>			
	のインストール	1. 以下のURLからダウンロードします。			
		http://jp.fujitsu.com/group/fit/services/printers/downloads/driver/fp1100/#FP-11000POS			
		2. ダウンロードした実行ファイルをダブルクリックして解凍します。			
		 解凍して作成されたフォルダ内(1つ下の階層のフォルダの場合があります)のSetup.exeを起動します。 			
		4. 以下の画面が表示されます。画面の指示に従いインストールを行ってください。 場 FIT FP POS PRINTER OPOS OCX - セットアップウィザード			
		セットアップウィザードへようこそ			
		セットアップウィザードは、ご使用のコンピュータへ FIT FP POS PRINTER OPOS OCX をインストールします。「次へ」をクリックして、 続行してください。			
		< 戻る(B) 次へ(N) > キャンセル			
		64bit 05の提合\			
		1. 以下のURLからダウンロードします。 http://jp.fujitsu.com/group/fit/services/printers/downloads/driver/fp1100/#FP-11000POS			
		2. ダウンロードした実行ファイルをダブルクリックして解凍します。			
		3. 管理者権限でコマンドプロンプトを起動し、解凍して作成されたフォルダ内のSetup.exeがあるフォ ルダに移動します。			
		 Setup.exeに起動オプション"/32bit"を付けて実行します。 コマンドプロンプトから下記を実行してください。 > Setup.exe /32bit 			
		5. 以下の画面が表示されます。画面の指示に従いインストールを行ってください。 切りたいので、「場 FIT FP POS PRINTER OPOS OCX - セットアップウィザード			
		セットアップウィザードへようこそ			
		セットアップウィザードは、ご使用のコンピュータへ FIT FP POS PRINTER OPOS OCX をインストールします。「次へ」をクリックして、 続行してください。			
		< 戻る(B) 次へ(N) > キャンセル			

手順	内容	詳細		
2	OPOS ドライバ	1. 使用するプリンタをPCに接続し、電源を入れます。		
	とプリンタのセッ			
	トアップ	2. OPOSのセットアップを行います。Windows8以降の場合は、アプリー覧画面から、その他のOSの		
		場合は、スタートメニューから「FP Printer OPOS Setup」を起動します。		
		2 インターフェーフ 左躍 切」 「0ビ] たクリック」 古古		
		3. インメーノエースを選択し、[UK]をソリソノしより。		
		C FP-1100		
		OK Exit		
		※ LAN I/Fを選択した場合		
		LAN接続されたプリンタを選択する画面が表示されます。		
		[Printer Detection]をクリックし、検索されたプリンタを選択後、[Printer Setting]をクリックします。		
		Fujitsu Isotec OCX Setup		
		The FP-1100 printer working in LAN is displayed.		
		10.50.134.163 00:80:92:01:14:BF Unsetting		
		Printer Detection Batch Setting Of Printers		
		Printer Setting		
		- Explanation		
		The printer detection begins when the printer detection button is pressed. The device name is set to the specified printer by the printer setting.		
		The unregistered printer is set to the registry by the batch setting of the printer.		
		Exit		
		4. プリンタの設定を任意に変更し、[OK]をクリックします。		
		この画面で表示されている"Device Name"は、2.3、2.4章で使用する"デバイス名"です。		
		Fujitsu Isotec OCX Setup		
		Device Name Port Name IP Address		
		Serial Number (USB) Baud Rat (Serial) Print Columns		
		Peload Format (Serial) Print Level Print Speed		
		8 None 1 ✓ 100% ✓ 220mm/s ✓ □Drawer Status		
		Normal C Invert C Off Mode1 C Mode2 Enable C Disable		
		Construction Point Appartment Cut at Coverclose Conff Con C Mode 1 C Disable C Enable		
		Error Alert		

2.1.3 Common Control Object (共通 CO)

Common Control Object(共通 CO)をインストールします。

手順	内容	詳細
1	Common Control Object のインストール	1. 以下のURLから「1.12.001 CCO Runtime (Wize Install)」をダウンロードします。 http://monroecs.com/oposccos_current.htm
		<text></text>
		※共通COはFP-1100 OPOSより後にインストールしてください。 共通COのインストール後にFP-1100 OPOSドライバをインストールすると、POS for .NET からデバイスが使用できなくなります。これはレジストリのOPOSデバイスのCO設定が、 共通COからFP-1100 OPOSのCOに上書きされるためです。 この場合は、共通CO をアンインストール&再インストールするか、RegSvr32.exe を使用 して、共通CO の各OCX を再度レジストリに登録し直しすると使用できるようになります。

2.1.4 POS for .NET

POS for .NET 1.12をインストールします。 ※インストールの際は、POS for .NET SDKも選択してください。

手順	内容	詳細
1	POS for .NET 1.12 のダウ	1. 以下の URL からダウンロードします。
	ンロードと解凍	http://www.microsoft.com/en-us/download/details.aspx?id=5355
		2. ダウンロードした POS for .NET v1.12.exe をダブルクリックします。
		以下の画面が表示されますので、[はい]をクリックします。
		Microsoft POS for .Net 1.12
		Do you want to unzip Microsoft POS for .Net 1.12 installation files?
		 3 以下の画面が表示されます。"Unzip to folder"に、任意の解凍先フォルダを指
		定して、[Unzip]をクリックします。
		WinZip Self-Extractor - POS for .Net v1.12.exe
		To unzip all files in POS for .Net
		 4. 解凍が完了すると、以下の画面が表示されます。[OK]をクリックします。 WinZip Self-Extractor 170 file(s) unzipped successfully
		5. [Close]をクリックします。 WinZip Self-Extractor - POS for .Net v1.12.exe
		To urzip all files in PCS for .Net Unzip v1.12.exe to the specified folder press the Urzin huftron Run WinZip Unzip to folder: C:\work\POS for .NET Diverwrite files without prompting About Help

手順	内容	「詳細」
2 2	POS for .NET 1.12 の インストール	1. 解凍先フォルダ内の Setup.exe を起動します。
		 以下の画面が表示されます。[Install]をクリックします。 Installation Wizard
		Microsoft POS for .NET
		Cocumentation View Release Notes Statil Install Install Inforcedit POS for NET 1.12
		Exit
		3. 次の画面で[Next]をクリックし、更に次の画面で[Accept]をクリックします。
		4. 以下の画面が表示されます。"Custom"を選択し、[Next]をクリックします。 [▲] Microsoft POS for .NET 1.12 Installation Wizard
		Installation Options Select the type of installation.
		Select the type of installation that best suits your needs and the installation path. Typical Description
		© Complete © Dustom 2 This option allows you to select the features you want to install.
		Space allocation Total size required: 1.34 MB Space available on C: 340.9 GB
		Install to: D#Program Files#Microsoft Point Of Service# Browse
		(1) Cancel
		5. 以下の画面が表示されます。"SDK"にもチェックを入れ、[Next]をクリックしま
		す。 Microsoft POS for .NET 1.12 Installation Wizard
		Custom Installation Select components.
		Select components to install:
		Total size required: 11 93 MB Space available on D: 34.09 GB
		Cancel



2.1.5 WS-POS 環境

WS-POS 参照実装をダウンロードし、配置します。

手順	内容	詳細	詳細	
1	WS-POS 参照実装の	1.	以下からダウンロードします。	
	ダウンロードと配置		https://nrf.com/resources/retail-library/ws-pos-version-12-reference-implementation-c	
		2.	ダウンロードした ARTS WS-POS 1.2 Reference Implementation_0.zip を解凍しま す。	
		3.	解凍したフォルダから、WS-POS 参照実装のフォルダ(WSPOS_Refer_Impl_v3)を、 任意のフォルダにコピーします。(以降コピーしたフォルダを[WS-POS フォルダ]と 記述します。)	
2	WS-POS サービスプロバイダを	1.	コマンドプロンプトを 管理者権限 で起動します。	
	Windows サービスとし てインストール	2.	以下のコマンドを入力し、WS-POS 参照実装のフォルダに移動します。 > cd [WS-POS フォルダ]¥WSPOS_Refer_Impl_v3¥Provider¥_dist¥Release	
		3.	C:¥Windows¥Microsoft.NET¥Framework¥v2.0.50727 配下にある InstallUtil を利用 して WS-POS サービスプロバイダを Windows サービスとしてインストールします。 以下のコマンドを入力します。 > C:¥Windows¥Microsoft.NET¥Framework¥v2.0.50727¥ InstallUtil WcfServiceHostWindowsService35.exe [ENTER]	
		4.	コントロールパネルからサービスを表示し、"WS-POS Service Provider Hosting Container"のサービス名が表示されていることを確認します。	
			ますので、サービスが動作しているとコンソールアプリケーションと衝突してしまう ためです。	

2.1.6 その他の設定

手順	内容	詳細
1	ファイアウォールの設定	参照実装のデフォルトでは、WS-POS サービスプロバイダは8087 ポートにて WS-POS サービスコンシューマからのリクエストを待ちます。
		このため、サービスプロバイダがポートを開いて待ち受け可能なように、オペレーティングシステムやセキュリティソフトウェアにて上記ポートの通信をブロックしている 場合は、設定変更が必要です。

2.2 インストール確認

インストールが正常に行われたかどうかを確認します。下記手順をサーバーで行ってください。

手順	内容	詳細
1	コンソールアプリケーションによ	[WS-POS フォルダ]¥WSPOS_Refer_Impl_v3¥Provider¥_dist¥
	るサービスプロバイダ起動確認	Release¥WcfServiceHostConsoleExe35.exe
		を起動します。
		成功するとコマントフロンフトが表示され、待ち受け状態となります。
		C:¥WSPOS_Refer_Impl_v3¥Provider¥_dist¥Debug¥WcfServiceHostCon
		WSPOSInitiative.HardTotals.HardTotalsService Ready.
		WSPOSInitiative.Inmaeodammer.Inmaeodammer.etwice.Ready. WSPOSInitiative.ItemDispenser.ItemDispenserService Ready.
		WSPOSInitiative.Lights.LightsService Ready. WSPOSInitiative.Lights.LightsService Ready.
		WSPOSInitiative.MICR.MICRService Ready. WSPOSInitiative.MotionSensor.MotionSensorService Ready.
		WSPOSInitiative.MSR.MSRService Ready. WSPOSInitiative.PINPad.PINPadService Ready.
		WSPOSInitiative.PointCardkW.PointCardkWService Ready. WSPOSInitiative.POSKeyboard.POSKeyboardService Ready.
		WSPUSInitiative.PUSPower.PUSPowerService Keady. WSPOSInitiative.PUSPrinter.PUSPrinterService Ready. WSPOSInitiation.com.torden.pusp.Pusper.com.torden.com.com.text.
		WSPOSInitiative.RelideonerDrspray.NeuloceonerDrspraydervice Ready. WSPOSInitiative.RFIDScanner.RFIDScannerService Ready.
		WSPOSInitiative.Scanner.ScannerService Ready. WSPOSInitiative.Scanner.ScannerService Ready.
		WSPOSInitiative.SmartCardRW.SmartCardRWService Ready. WSPOSInitiative.ToneIndicator.ToneIndicatorService Ready.
		WSPOSInitiative.Common.FileHostService Ready. Press Enter to End.
		- · · · · · · · · · · · · · · · · · · ·
2	サービスコンシューマサンプルの	[WS-POSフォルダ]¥WSPOS_Refer_Impl_v3¥Consumer¥
	実行	Console_POSPrinterConsumerBiDirectionSample¥bin¥ Debug¥
		Console_POSPrinterConsumerBiDirectionSample.exe
		を延期しまり。 ポカナスレッフンパプロンプレがまニさね。 法た 受け状態 したります
		成功するとコマントノロンフトが表示され、付ら受け状態となります。
		C:\WSPOS_Refer_Impl_v3\Consumer\Console_POSPrinterConsumerBi
		Press ENTER to OpenDevice, Claim, Enable.
		Microsoft PosPrinter Simulator
		<u>^</u>
		Cover Open
		Receipt Cover Open
		Power Slip Cover Open Print delay:
		Clear
		Idle, Claimed, Enabled
		Enter キーを押すと、POSPrinter シミュレータが動作します。
		Enter キーを押していくと、終了します。
L		

これらの動作が確認できれば、インストールは正常に完了しています。 これらアプリケーションは動作確認後、終了してください。

2.3 実デバイスを用いた実行のための設定(POS プリンタ)

環境の構築が完了したら、FP-1100向けに固有の設定を行います。

手順	内容	詳細
1	OPOS デバイス名確認	コマンドプロンプトを管理者権限で起動します。
		C:¥Program Files¥Microsoft Point Of Service に移動(64bit OS の場合は、
		C:¥Program Files (x86) ¥Microsoft Point Of Service)し、以下のコマントで、 対象のドライバが出力されるかた確認します
		対象のパンパンロンでもであり、それに応じより。
		> posdm listsos
		以下のような画面が表示されます。FP−1100 用のデバイス名が表示される ことを確認します。
		as 管理者: コマンド プロンプト - ロー× CashDrawer FP1100USECF2
		PosPrinter FP1100LAWFRT_192.168.2.104
		PosPrinter FP1100LMPRT_LEF
		PosPrinter FP1100SER2PRT
		PosPrinter FP1100SERPRT
		PosPrinter FP1100US82PRT
		PosPrinter FP1100USBPR7
		C:YProgram Files (x86)VMicrosoft Point Of Service ²
2	OPOS デバイス登録	以下のコマンドでデバイスを登録します
		> posdm addname FP-1100 /TYPE:PosPrinter /SOName:"FP1100LANPRT_192.168.2.104"
		この例では、"FP-1100"というエイリアスについて、
		デバイス名"FP1100LANPRT_192.168.2. 104"を登録します。
_		
3	登録テハイス確認	以下のコマントでナハイスを登録します
		> posdm listnames
		FP-1100 が登録されていることを確認します。
		PosPrinter FP1100USEPRT
		C:\Program Files (x88)\Microsoft Point Of Service>posdm addname FP-1100 /TYPE:Po sPrinter /SOName: "FP1100LANFRT_182.104"
		C:NProgram Files (x88)VMicrosoft Point Of Service>posdm listnames Type SoName Name Path
		CashDrawer FP1100U8EDR1 DefaultCashDrawer
		CashDrawer FP1100USEDR2 AlternateCashDrawer
		PosPrinter FP1100LAMPRT_192.168.2.104 FP-1100 PosPrinter FP1100LSPRT DefaultPDSPrinter
		C:YProgram Files (x86)YMicrosoft Point Of Service>v

手順	内容	詳細
4	サービスコンシューマ設定ファイ	[WS-POS フォルダ] ¥WSPOS_Refer_Impl_v3¥Consumer¥
	ルの変更	Console_POSPrinterConsumerBiDirectionSample¥bin¥Debug¥
	(クライアント側)	Console_POSPrinterConsumerBiDirectionSample.exe.config
		の以下の部位を修正します。
		◆修止則
		"http://localhost:8087/POSPrinter.svc/POSPrinterSimulator"
		binding="basicHttpBinding"
		contract="UnifiedPOS.POSPrinter.V1_2.POSPrinter"
		name="POSPrinterPort"/>
		\downarrow
		◆修正後
		<pre><endpoint address="</pre"></endpoint></pre>
		"http://localhost:8087/POSPrinter.svc/ <u>POSPrinterFP1100</u> "
		binding= basicHttpBinding
		name="POSPrinterPort"/>
5	サービスプロバイダ設定ファイル	「WS-POS フォルダ]¥WSPOS Refer Impl v3¥Provider¥ dist¥Release¥ フォル
	の変更(サーバー側)	ダにて、下記ファイルの以下に示す部位を修正します。
		•WcfServiceHostConsoleExe35.exe.config
		◆修正前
		<behavior name="POSPrinter_Simulator"></behavior>
		<pre><wsposdevice <="" devicename="Microsoft PosPrinter Simulator" pre=""></wsposdevice></pre>
		providerSessionTimeout="180" eventPollingTimeout="-1"
		eventResponse limeout= -1 />
		↓ ▲ 你正 没
		▼ 修正 仮 〈hebavior name=″POSPrinter FP1100″〉
		<pre><wsposdevice <="" devicename="FP-1100" pre=""></wsposdevice></pre>
		providerSessionTimeout="180" eventPollingTimeout="-1"
		eventResponseTimeout="-1"/>
		◆修止前 () · · · · · · · · · · · · · · · · · ·
		<pre><endpoint address="PUSPrinterSimulator</th"></endpoint></pre>
		binding="basicHttpBinding"
		contract="UnifiedPOS.POSPrinter.V1_2.POSPrinter"
		bindingNamespace=
		"http://www.nrf-arts.org/UnifiedPOS/POSPrinter/"/>
		\downarrow
		◆修正後
		<pre><endpoint <="" address="POSPrinterFP1100" pre=""></endpoint></pre>
		behaviorContiguration= <u>POSPrinter_FP1100</u>
		binding – basichuppinding contract="UnifiedPOS POSPrinter V/1 2 POSPrinter"
		bindingNamespace=
		"http://www.nrf-arts.org/UnifiedPOS/POSPrinter/"/>

2.4 実デバイスでの実行(コンシューマとプロバイダが同一 PC)

コンシューマとプロバイダを同一 PC で動作させ、実デバイスの動作を実行します。

プリンタを PC に接続し、「2.2. インストール確認」の手順を実施すると、プリンタから印刷が行われます。 印刷が行われない場合は、「2.3 実デバイスを用いた実行のための設定(POS プリンタ)」に間違いがないか確認してください。

2.5 実デバイスでの実行(コンシューマとプロバイダが別 PC)

コンシューマ(クライアント)とプロバイダ(サーバー)をそれぞれ別々の PC で動作させ、実デバイスの動作を実行します。

手順	内容	詳細
1	サービスコンシューマファイルを	サーバーPCのフォルダ、
	クライアントPCにコピー	[WS-POS フォルダ] ¥WSPOS_Refer_Impl_v3¥Consumer¥
		Console_POSPrinterConsumerBiDirectionSample¥bin¥Debug¥
		をクライアントPCにコピーします。
2	サービスコンシューマ設定ファイ	サービスプロバイダ URL を変更します。
	ルのサービスプロバイダ URL の	クライアントPCで、
	変更	[WS-POS フォルダ] ¥WSPOS_Refer_Impl_v3¥Consumer¥
	(クライアント PC 側)	Console_POSPrinterConsumerBiDirectionSample¥bin¥Debug¥
		Console_POSPrinterConsumerBiDirectionSample.exe.config
		の以下の部位を修正します。
		◆修正前
		<pre><endpoint address="</td"></endpoint></pre>
		http://localhost:808//POSPrinter.svc/POSPrinterFPTT00
		pinding – pasicificip pinding
		name="POSPrinterPort"/>
		● 修正後
		<endpoint address="</td"></endpoint>
		"http:// 10.50.13x.xxx :8087/POSPrinter.svc/POSPrinterFP1100"
		binding="basicHttpBinding"
		contract="UnifiedPOS.POSPrinter.V1_2.POSPrinter"
		name="POSPrinterPort"/>
		「10.50.13x.xxx」はサーバーPC の IP アドレスです。

手順	内容	詳細
3	サービスコンシューマ設定ファイ	サービスプロバイダに通知するサービスコンシューマの URL を変更します。
	ルのサービスコンシューマ URL	クライアントPCで、
	の変更	[WS-POS フォルダ] ¥WSPOS_Refer_Impl_v3¥Consumer¥
	(クライアント PC 側)	Console_POSPrinterConsumerBiDirectionSample¥bin¥Debug¥
		Console_POSPrinterConsumerBiDirectionSample.exe.config
		の以下の部位を修正します。
		◆修正前
		<host></host>
		<add baseaddress="http://localhost:8001/POSPrinterEvent"></add>
		↓ ▲ 按 工 送
		▼修止夜
		<pre>\nust/ </pre>
		<pre><add baseaddress="http://10.50.13x.vvv:8001/POSPrinterEvent"></add></pre>
		「10.50.13x.yyy」はクライアント PC の IP アドレスです。
4	実デバイスでの動作確認	プリンタを PC に接続し、「2.2. インストール確認」の手順を実施してください。

2.6 WS-POS サービスプロバイダを Windows Service で実行

ここまでは、サービスプロバイダはコンソールアプリケーションを使用していましたが、「2.1 インストール」の手順 5 で 登録した Windows Service での実行手順を説明します。

手順	内容	詳細
1	サービスプロバイダ設定ファイル の変更(サーバー側)	[WS-POS フォルダ]¥WSPOS_Refer_Impl_v3¥Provider¥_dist¥Release¥ フォル ダの WcfServiceHostWindowsService35.exe.config を 2.2 の手順 5 および 2.3 の手順 5 と同様に変更します。
2	サービスの起動	コントロールパネルからサービスを表示します。 "WS-POS Service Provider Hosting Container"のサービスを選択し、 "サービス起動"をクリックします。 "サービスに起動"をクリックします。 "サービスにはいいいいいいいいいいいいいいいいいいいいいいいいいいいいいいいいいいいい
3	サービスコンシューマの起動	Console_POSPrinterConsumerBiDirectionSample.exe などのコンシューマを 起動します。動作はコンソールアプリケーションのサービスプロバイダと変わ りません。

2.7 サービスプロバイダの各種設定

ここでは、サービスプロバイダの設定可能な項目について説明します。

2.7.1 Behavior 名、デバイス名の変更

以下の部位を変更します。

```
<behavior name="POSPrinter_Simulator">
```

```
<wsposDevice deviceName="Microsoft PosPower Simulator" providerSessionTimeout="180"
eventPollingTimeout="-1" eventResponseTimeout="-1" />
```

</behavior>

Кеу	説明
behaviour name	使用する behavior 名を記載します。
wsposDevice deviceName	使用するデバイス名を記載します。
providerSessionTimeout	セッションのタイムアウト時間を秒単位で指定します。
eventPollingTimeout	ロングポーリングのタイムアウト時間を秒単位で指定しま
	す。
	-1を指定した場合はタイムアウトしません。
eventResponseTimeout	ロングポーリング/自己ホストのイベント処理終了待ちのタイ
	ムアウト時間を秒単位で指定します。
	-1 を指定した場合はタイムアウトしません。

2.7.2 サービスプロバイダ URL の変更

以下の部位を変更します。

</service>

Кеу	説明
add baseAddress	サービスプロバイダ URLを下記形式で指定します。
	http://[使用するプロバイダの IP アドレス:ポート番号]/POSPrinter.svc
	例)
	http://192.168.1.120:8087/POSPrinter.svc

2.7.3 endpointの変更

```
以下の部位を変更します。
```

</service>

Кеу	説明
endpoint address	使用する endpoint 名を記載します。
behaviorConfiguration	使用する behavior 名を記載します。

※同じデバイス(例えば POSPrinter)を複数使用する場合は、<endpoint>の定義をコピーして複数のデバイス定義を記載します。

例)

```
<service name="WSPOSInitiative.POSPrinter.POSPrinterService"</pre>
        behaviorConfiguration="WS-POS_ServiceProvider">
 <host>
   <baseAddresses>
     <add baseAddress="http://localhost:8087/POSPrinter.svc"/>
   </baseAddresses>
 </host>
 <endpoint address="POSPrinter1"</pre>
      behaviorConfiguration="POSPrinter_Behavior1"
      binding="basicHttpBinding" contract="UnifiedPOS.POSPrinter.V1_2.POSPrinter"
      bindingNamespace="http://www.nrf-arts.org/UnifiedPOS/POSPrinter/"/>
 <endpoint address="POSPrinter2"</pre>
      behaviorConfiguration="POSPrinter_Behavior2"
      binding="basicHttpBinding" contract="UnifiedPOS.POSPrinter.V1_2.POSPrinter"
      bindingNamespace="http://www.nrf-arts.org/UnifiedPOS/POSPrinter/"/>
</service>
```

2.7.4 セッションタイムアウトの確認間隔の変更

以下の部位を変更します。

<appSettings>

<add key="POSPrinterService_ProviderSessionTimeoutCheckPeriod" value="1"/>

... </appSettings>

add key の値	Key	説明
POSPrinterService_ProviderSession	value	セッションのタイムアウトを定期的に確認する間隔
TimeoutCheckPeriod		(秒)を指定します。

2.7.5 通信ポートの最大同時接続数の変更

以下の部位を変更します。

```
\langle system.net \rangle
```

```
<connectionManagement>
```

```
<add address = "*" maxconnection = "48" />
```

```
</connectionManagement>
```

```
</system.net>
```

Кеу	説明
maxconnection	サービスプロバイダとサービスコンシューマ間の通信で同時
	に使用する最大ポート数を指定します。
	スレッド毎にポートを開くため、スレッドの数を考慮した値を
	指定してください。例えば、参照実装のサンプルでは、main
	スレッド、Polling 処理スレッド、KeepAlive 処理スレッドの3つ
	が必要になります。

2.7.6 使用する. NET Framework バージョンの変更

以下の部位を変更します。

<startup>

```
<supportedRuntime version="v2.0.50727"/>
```

</startup>

Кеу	説明
supportedRuntime version	使用する Microsoft .NET Framework のバージョンを指定しま
	す。
	 Microsoft .NET Framework 4.0 : "v4.0"
	 Microsoft .NET Framework 3.5 : "v2.0.50727"
	•Microsoft .NET Framework 3.0 : //

2.8 サービスコンシューマの各種設定

ここでは、サービスコンシューマの設定可能な項目について説明します。

2.8.1 endpointの変更

以下の部位を変更します。

<client>

```
<endpoint address="http://localhost:8087/POSPrinter.svc/POSPrinterSimulator"
    binding="basicHttpBinding"
    contract="UnifiedPOS.POSPrinter.V1_2.POSPrinter"
    name="POSPrinterPort"/>
```

</client>

Кеу	説明
endpoint address	接続するサービスプロバイダの URL と endpoint 名を下記形
	式で指定します。
	http://[使用するプロバイダの IP アドレス:ポート番号]/
	POSPrinter.svc/[endpoint 名]
	例)
	http://192.168.1.120:8087/POSPrinter.svc/POSPrinterFP1100
name	この接続の名称を指定します。

※同じデバイス(例えば POSPrinter)を複数使用する場合は、<endpoint>の定義をコピーして複数のデ バイス定義を記載します。

例)

```
<client>
        <endpoint address="http://192.168.1.120:8087/POSPrinter.svc/POSPrinter1"
            binding="basicHttpBinding"
            contract="UnifiedPOS.POSPrinter.V1_2.POSPrinter"
            name="POSPrinterPort1"/>
            <endpoint address="http:// 192.168.1.120:8087/POSPrinter.svc/POSPrinter2"
            binding="basicHttpBinding"
            contract="UnifiedPOS.POSPrinter.V1_2.POSPrinter.svc/POSPrinter2"
            binding="basicHttpBinding"
            contract="UnifiedPOS.POSPrinter.V1_2.POSPrinter.svc/POSPrinter2"
            binding="basicHttpBinding"
            contract="UnifiedPOS.POSPrinter.V1_2.POSPrinter"
            name="POSPrinterPort2"/>
```

</client>

2.8.2 サービスコンシューマ URL の変更

サービスコンシューマがサービスプロバイダからイベントを受信するため、サービスプロバイダへサービス コンシューマの URL を下記設定により通知します。(Open メソッドで通知されます) 以下の部位を変更します。

```
<host>
<baseAddresses>
<add baseAddress="http://localhost:8001/POSPrinterEvent"/>
</baseAddresses>
</host>
```

Кеу	説明
add baseAddress	サービスコンシューマの URL を下記形式で指定します。 http://[コンシューマの IP アドレス:ポート番号]/POSPrinterEvent.svc 例)
	http://192.168.1.101:8001/POSPrinterEvent.svc

3. WS-POS アプリケーションの作成

3.1 開発環境

Visual Studio 2010 または、Visual Studio 2012 を用いて開発を行います。 ここでは、Visual Studio 2012 を例に説明します。

3.2 開発言語

WS-POS サービスを Windows サービスで提供する場合を説明します。 下記の開発言語が使用可能です。 •Microsoft Visual Basic .NET

•Microsoft Visual C#

3.3 作成手順

3.3.1 新規プロジェクトの作成

手順	Visual Basic .NET	Visual C#
1	Visual Basic の Windows アプリケーション で作成します。	C#の Windows アプリケーションで作成します。
ツ旺方	のプロジークレスナル出す他です	

※既存のプロジェクトでも作成可能です。

※コンソールアプリケーションや Windows フォームアプリケーションでも作成可能です。

3.3.2 WSPOSContractの追加

手順	Visual Basic .NET	Visual C#
1	参照実装に含まれている、	
	"WSPOSContract.dll を、プロジェクトフォル:	ダにコピーします。
2	ソリューションエクスプローラからプロジェクトを	右クリックし、[参照の追加]をクリックします。
3	[参照]タブを選択し、	
	"WSPOSContract.dll"を選択後、[OK]をクリ	ックします。
4	ソリューションエクスプローラに	
	"WSPOSContract.dll"が追加されていること	を確認してください。

3.3.3 System.ServiceModelの追加

手順	Visual Basic .NET	Visual C#
1	ソリューションエクスプローラからプロジェクトを	右クリックし、[参照の追加]をクリックします。
2	[.NET]タブを選択し、	
	"System.ServiceModel"を選択後、[OK]をク	リックします。
3	ソリューションエクスプローラに	
	"System.ServiceModel"が追加されているこ	とを確認してください。

手順	Visual Basic .NET	Visual C#
1	ソリューションエクスプローラの「すべてのフ	ソリューションエクスプローラからプロジェクトを
	ァイルを表示」をクリックします。	選択し、メニューバーの[ブロジェクト]-[新しい項
		目の追加]を選択します。
2	ソリューションエクスプローラに	[Visual C#アイテム]-[全般]から、[アプリケーシ
	"App.config"が追加されていることを確認し	ョン構成ファイル]を選択し、[追加]をクリックしま
	てください。	す。
	※"App.config"が追加されていない場合	
	は、4 以降を実施してください。	
3	"App.config"を右クリックし、[プロジェクトに	ソリューションエクスプローラに
	含める]をクリックします。	"App.config"が追加されていることを確認して
		ください。
	※4 以降の実施は不要です。	
4	ソリューションエクスフローラからフロジェクト	
	を選択し、メニューバーの[プロジェクト]-[新し	
	い項目の追加]を選択します。	
5	[共通項目]-[全般]から、[アプリケーション構	
	成ファイル]を選択し、[追加]をクリックしま	
	す。	
6	ソリューションエクスプローラに	
	"App.config"が追加されていることを確認し	
	てください。	

3.3.4 アプリケーション構成ファイルの追加

3.3.5 アプリケーション構成ファイルの編集

アプリケーション構成ファイルに下記の太字部分を記述します。 (Visual Basic .NET, Visual C#共通です)

```
<?xml version="1.0"?>
<configuration>
```

```
<system.serviceModel>
<client>
<endpoint address="http://192.168.1.210:8087/POSPrinter.svc/POSPrinter1"
binding="basicHttpBinding"
contract="UnifiedPOS.POSPrinter.V1_2.POSPrinter"
name="POSPrinterPort1"/>
</client>
<services>
</system.serviceModel>
```

```
<system.net>
<connectionManagement>
<add address = "*" maxconnection = "48" />
</connectionManagement>
</system.net>
```

</configuration>

Кеу	説明
endpoint address	使用するサービスプロバイダの URL を指定します。
binding	サービスプロバイダで設定している binding
	"basicHttpBinding"を指定します。
contract	使用するデバイスカテゴリーのコントラクトを指定します。
name	この接続情報を識別するための名称を指定します。
	サービスプロバイダと接続する際に、接続情報を取得するた
	めに使用します。
maxconnection	サービスプロバイダとサービスコンシューマ間の通信で同時
	に使用する最大ポート数を指定します。
	スレッド毎にポートを開くため、スレッドの数を考慮した値を
	指定してください。例えば、参照実装のサンプルでは、main
	スレッド、Polling 処理スレッド、KeepAlive 処理スレッドの3つ
	が必要になります。

3.3.6 アプリケーションソースの編集

アプリケーションのソースに、サービスプロバイダとの接続処理を記述します。 サービスプロバイダとのセッション開始~終了の間に、メソッド・プロパティの処理を記述します。

<Visual Basic .NET>

◆記述ファイル [アプリケーションで指定したファイル名].vb

◆ConsumerID を生成 ConsumerID は、デバイスを識別するために使用します。 Dim consumerID As String = Guid.NewGuid().ToString() ※上記例は、GUID を生成して ID として使用しています。

◆メッセージ送信用チャネル生成

Dim device As UnifiedPOS.POSPrinter.V1_2.POSPrinter ' アプリケーション構成ファイルで設定した endpoint の contract と name を指定する Dim factory As New ChannelFactory (Of UnifiedPOS.POSPrinter.V1_2.POSPrinter)("POSPrinterPort1") ' チャネルを生成 device = factory.CreateChannel()

◆セッションの開始 サービスプロバイダとのセッションを開始します。 device.OpenSession(consumerID, Nothing)

※OpenSession後、各メソッドの呼び出しが可能になります。

◆メソッド・プロパティの使用 printNormal メソッドを使用する場合、下記のように記述します。

device.OpenDevice(consumerID)

- device.Claim(consumerID, 10000)
- device.SetDeviceEnabled(consumerID, true)
- device.PrintNormal(consumerID,

UnifiedPOS.POSPrinter.V1_2.PrinterStation.Receipt, "印刷するテキスト")

```
<sup>・</sup>エスケープシーケンスを使用し Bold 印刷
```

device.PrintNormal(consumerID,

UnifiedPOS.POSPrinter.V1_2.PrinterStation.Receipt, "\u001b|bC 太字印刷") device.SetDeviceEnabled(consumerID, False) device.Release(consumerID)

device.CloseDevice(consumerID)

◆セッションの終了 サービスプロバイダとのセッションを終了します。 device.CloseSession(consumerID) <Visual C#>

- ◆記述ファイル:
 - [アプリケーションで指定したファイル名].cs

◆ConsumerID を生成

ConsumerID は、デバイスを識別するために使用します。 private string consumerID = Guid.NewGuid().ToString(); ※上記例は、GUID を生成して ID として使用しています。

◆メッセージ送信用チャネル生成

private UnifiedPOS.POSPrinter.V1_2.POSPrinter device = null; // アプリケーション構成ファイルで設定した endpoint の contractとname を指定 ChannelFactory< UnifiedPOS.POSPrinter.V1_2.POSPrinter > factory = new ChannelFactory< UnifiedPOS.POSPrinter.V1_2.POSPrinter >("POSPrinterPort1"); // チャネルを生成 device = factory.CreateChannel();

◆セッションの開始 サービスプロバイダとのセッションを開始します。 device.OpenSession(consumerID, null);

※OpenSession後、各メソッドの呼び出しが可能になります。

◆メソッド・プロパティの使用

printNormal メソッドを使用する場合、下記のように記述します。

device.OpenDevice(consumerID);

device.Claim(consumerID, 10000);

device.SetDeviceEnabled(consumerID, true);

device.PrintNormal(consumerID,

UnifiedPOS.POSPrinter.V1_2.PrinterStation.Receipt, " 印刷するテキスト");

// エスケープシーケンスを使用し Bold 印刷

device.PrintNormal(consumerID,

UnifiedPOS.POSPrinter.V1_2.PrinterStation.Receipt, "\u001b|bC太字印刷"); device.SetDeviceEnabled(consumerID, false); device.Release(consumerID);

device.CloseDevice(consumerID);

◆セッションの終了 サービスプロバイダとのセッションを終了します。 device.CloseSession(consumerID);

3.3.7 例外処理

Catch した例外から UposException 型の詳細情報を取得することで、エラーの原因を判断できます。

<Visual Basic .NET>

```
try

'デバイスオープン

device. OpenDevice (consumer ID)

Catch ex As FaultException (Of UnifiedPOS. POSPrinter. V1_2. UposException)

'Catch した例外から UposException 型の詳細情報を取得

Dim ue As UnifiedPOS.POSPrinter.V1_2.UposException = ex.Detail

'ErrorCode を取得

Dim code As UnifiedPOS.POSPrinter.V1_2.ErrorCode = ue.ErrorCode

'ErrorCodeExtended を取得

Dim errorCodeEx As Integer = ue.ErrorCodeExtended

End Try
```

<Visual C#> try { // デパイスオープン device.OpenDevice(consumerID); } catch (FaultException<UnifiedPOS.POSPrinter.V1_2.UposException> ex) { // Catch した例外から UposException 型の詳細情報を取得 UnifiedPOS.POSPrinter.V1_2.UposException ue = ex.Detail; // ErrorCode を取得 UnifiedPOS.POSPrinter.V1_2.ErrorCode code = ue.ErrorCode; // ErrorCodeExtended を取得 int errorCodeEx = ue.ErrorCodeExtended; }

}

3.3.8 イベント処理-SelfHost 方式

サービスコンシューマ(アプリケーション)に用意したイベント受信サービスクラスに、サービスプロバイダ からイベントを通知してもらう方式です。

<Visual Basic .NET>

手順	イベント処理-SelfHost 方式 Visual Basic.NET	
1	Visual Studio でイベント取得用サ	ソリューションエクスプローラからプロジェクトを右クリックし、[追
	ービスクラスを作成	加]-[新しい項目]をクリックします。
2	VB のクラスを追加	ファイル名"POSPrinterEventService.vb"でクラスを追加しま
		す。
3	追加したファイル"POSPrinterEvent	Service.vb"を編集します。以下を参考にしてください。
	◆POSPrinterEventService.vb	-
	<pre></pre> <pre></pre> <pre></pre> <pre> </pre> <pre> <pre> <pre> <pre> <pre> <pre> <pre> <pre> <pre> <pre> <pre> <pre> <pre> <pre> <pre> <pre> <pre> <pre> <pre> <pre> <pre> <pre> <pre> <pre> <pre> <!--</th--><th>e :="http://www.nrf-arts.org/UnifiedPOS/POSPrinterEvents/ ",</th></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre>	e :="http://www.nrf-arts.org/UnifiedPOS/POSPrinterEvents/ ",
	InstanceContextMode:=Ins	stanceContextMode.Single)> _
	public class POSPrinterEvent	Service リのイベントクラスを継承する
	Implements UnifiedPOS.POS	SPrinterEvents.V1_2.POSPrinterEvent
	'ErrorEvent 発生時に呼ばれる Public Function ErrorEvent(メソット
	ConsumerID As String, So	urce As String, EventID As Integer, TimeStamp As Date,
	ErrorCode As UnifiedPOS	.POSPrinterEvents.V1_2.ErrorCode,
	ErrorLocus As UnifiedPOS	B.POSPrinterEvents.V1_2.ErrorLocus,
	ErrorResponse As Unified	POS.POSPrinterEvents.V1_2.ErrorResponse) _
	As UnifiedPOS.POSPrinte UnifiedPOS.POSPr	rEvents.v1_2.ErrorResponse Implements interEvents.V1_2.POSPrinterEvent.ErrorEvent
	Return UnifiedPOS.POSP	interEvents.V1_2.ErrorResponse.Clear
	End Function	
	'OutputComplateEvent 発生	時に呼ばれるメソッド
	Public Sub OutputComplete	Event(
	TimeStamp As Date, Outp	utid As Integer)
	Implements	
	UnifiedPOS.POSPrinto	erEvents.V1_2.POSPrinterEvent.OutputCompleteEvent
	' StatusUpdateEvent 発生時に	こ呼ばれるメソッド apt/
	ConsumerID As String, So	urce As String, EventID As Integer, TimeStamp As Date,
	Status As Integer) _	
	End Sub	
	' DirectIOEvent 発生時に呼ば Public Function DirectIOEve	れるメソッド unt(
	ConsumerID As String, So	urce As String, EventID As Integer, TimeStamp As Date,
	EventNumber As Integer,	Data As Integer, Obj As Object) _
	UnifiedPOS.POSPINte	PrinterEvents.V1_2.POSPrinterEvent.DirectIOEvent
	Return New UnifiedPOS.P	OSPrinterEvents.V1_2.DirectIOData()
	End Function	
	End Class	
	End Namespace	

手順	イベント処理-SelfHost 方式 Visual Basic.NET
4	アプリケーション構成ファイル(App.config)を編集します。
	◆App.config <configuration> <system.servicemodel> <services> <!-- サービス名を指定(サービス名は[サービスクラスのnamespace]+[クラス名])--> <service name="WSPOSEventService.POSPrinterEventService
behaviorConfiguration=" wsposclient.posprintereventservicebehavior"=""> <host> <baseaddresses> <!--baseAddresses--> <!--baseAddresslcイベント取得用URLを指定--> <add baseaddress="http://192.168.1.101:8001/POSPrinterEvent"></add> </baseaddresses> </host> </service></services></system.servicemodel></configuration>
	1ハント取得用サービスのEndpoint情報を指定 <endpoint <br="" address="">binding="basicHttpBinding" contract="UnifiedPOS.POSPrinterEvents.V1_2.POSPrinterEvent" bindingNamespace=http://www.nrf-arts.org/UnifiedPOS/POSPrinterEvents/ /> </endpoint>
5	 アプリケーションソースにイベント取得用のサービスホストを生成する処理を記述します。 Dim posPrinterEventServiceHost As ServiceHost 'イベント取得用のサービスの開始 posPrinterEventServiceHost = New ServiceHost(GetType(WSPOSEventService.POSPrinterEventService)) ' OpenSession の引数で Endpoint を渡す device.OpenSession(consumerID, "http://192.168.1.101:8001/POSPrinterEvent")
	'メソッドの呼び出しなどの処理 'イベント取得用サービス終了 posPrinterEventServiceHost.Close()

手順	イベント処理-SelfHost 方式 Visual C#	
1	Visual Studio でイベント取得用サ	ソリューションエクスプローラからプロジェクトを右クリックし、[追
	ービスクラスを作成	加]-[新しい項目]をクリックします。
2	C#のクラスを追加	ファイル名"POSPrinterEventService.cs"でクラスを追加しま
		<u>,</u>
3	追加したファイル"POSPrinterEvent	Service.cs"を編集します。以下を参考にしてください。
	POSPrinterEventService.cs	
	namespace WSPOSEventServic	
	POSPrinterEvents/".Instance	e = nttp://www.nn-ans.org/onnedPOS/
	// イベントを取得したいカテ:	ゴリのイベントクラスを継承する
	public class POSPrinterEve	entService: UnifiedPOS.POSPrinter.V1_2.POSPrinterEvent
	{	
	// ErrorEvent 発生時に呼は	れるメソッド DOSDrinterEvente V4 2 ErrerDeenenee ErrerEvent/
	string ConsumerI) string Source int EventID DateTime TimeStamp
	UnifiedPOS.POSP	rinterEvents.V1_2.ErrorCode ErrorCode,
	int ErrorCodeExte	nded,
	UnifiedPOS.POSPrinterEvents.V1_2.ErrorLocus ErrorLocus,	
	UnifiearOS.POSPrinterEvents.v1_2.ErrorKesponse ErrorKesponse) {	
	return UnifiedPOS.POSPrinterEvents.V1_2.ErrorResponse.Clear;	
	}	
	// OutputCompeteEvent 発生時に呼ばれるメソッド	
	public virtual void OutputCompleteEvent(
	string ConsumerII	D, string Source, int EventID, DateTime TimeStamp, int OutputID)
	3	
	// StatusUpdateEvent 発生	生時に呼ばれるメソッド
	public virtual void Status	UpdateEvent(
	string ConsumerII	D, string Source, int EventID, DateTime TimeStamp, int Status)
	}	
	// DirectIOEvent 発生時に	呼ばれるメソッド NDOOD-interaction VALIOD Discretion Discretion Function
	public virtual UnifiedPOs string ConsumerII	S.POSPrinterEvents.v1_2.DirectiOData DirectiOEvent(
	EventNumber, int	: Data, object Obj)
	{	
	return new UnifiedPOS	.POSPrinterEvents.V1_2.DirectIOData()
	{ Data = D	$a_1a_1, O_0 = O_0 $
	}	
	}	

<Visual C#>

手順	イベント処理-SelfHost 方式 Visual C#
4	アプリケーション構成ファイル(App.config)を編集します。
4	<pre>>フリケーション構成ファイル(App.config)を編集します。 </pre> ◆App.config <configuration> <system.servicemodel> <services> <l サービス名を指定(サービス名は[サービスクラスのnamespace]+[クラス名])=""> <service name="WSPOSEventService.POSPrinterEventService behaviorConfiguration=" wsposclient.posprintereventservicebehavior"=""> <host> <pre> </pre> <pre> </pre></host></service></l></services></system.servicemodel></configuration>
5	アプリケーションソースにイベント取得用のサービスホストを生成する処理を記述します。
	private ServiceHost posPrinterEventServiceHost; posPrinterEventServiceHost = new ServiceHost(typeof(WSPOSEventService.POSPrinterEventService));
	// イベント取得用のサービスの開始 posPrinterEventServiceHost.Open();
	// OpenSession の引数で Endpoint を渡す device.OpenSession(consumerID, "http://192.168.1.101:8001/POSPrinterEvent");
	Ⅱ メソッドの呼び出しなどの処理
	//イベント取得用サービス終了 posPrinterEventServiceHost.Close();

3.3.9 イベント処理-LongPolling 方式

サービスコンシューマ(アプリケーション)からサービスプロバイダに、イベント取得リクエストを送信し、そのレスポンスとしてイベントを通知してもらう方式です。

<Visual Basic .NET>

手順	イベント処理-LongPolling 方式 Visual Basic.NET
1	LongPolling用のスレッドを動作させ、サービスプロバイダにイベント取得リクエストを送る処理を追加します。
	Dim longPollingThread As Threading.Thread
	'セッションオープン
	device.OpenSession(consumerID, Nothing)
	longPollingThread = New Threading.Thread(AddressOf POSPrinterEvent LongPolling)
	longPollingThread.IsBackground = True
	longPollingThread.Start()
2	POSPrinterEvent_LongPollingメソッドにイベント取得時の処理を記載します。以下を参考にしてください。
	Private Sub POSPrinterEvent_LongPolling() While Not consumerID Is Nothing
	Dim deviceEvent As UnifiedPOS.POSPrinter.V1 2.POSPrinterEvent
	' イベントポーリングを実行(イベント取得リクエスト発行)
	deviceEvent = device.PollForUPOSEvent(consumerID)
	Continue While
	End If
	Dim res As New UnifiedPOS.POSPrinter.V1_2.POSPrinterEventResponse
	' 受信したイベントごとの処理を実行
	If Not deviceEvent.ErrorEvent Is Nothing Then
	「ErrorEvent 取得時の処理 Dim ev As UnifiedPOS.POSPrinter.V1 2.ErrorEvent = deviceEvent.ErrorEvent
	res.ErrorEventResponse = new UnifiedPOS.POSPrinter.V1_2.ErrorEventResponse() res.ErrorEventResponse.EventID = ev.EventID
	Elself Not deviceEvent.OutputCompleteEvent Is Nothing Then
	'OutputCompleteEvent 取得時の処理
	UnifiedPOS.POSPrinter.V1_2.OutputCompleteEventResponse()
	res.OutputCompleteEventResponse.EventID = ev.EventID
	Elself Not deviceEvent.StatusUpdateEvent Is Nothing Then
	'StatusUpdateEvent 取得時の処理
	Dim ev As UnifiedPOS.POSPrinter.V1_2.StatusUpdateEvent = deviceEvent.StatusUpdateEvent
	res.StatusUpdateEventResponse = new
	res.StatusUpdateEventResponse.EventID = ev.EventID
	Elself Not deviceEvent.DirectIOEvent Is Nothing Then
	'DirectIOEvent 取得時の処理 Dim av As UnifiedBOS BOSBrinter V1. 2 DirectIOEvent – deviceEvent DirectIOEvent
	res.DirectIOEventResponse = new UnifiedPOS.POSPrinter.V1_2.DirectIOEventResponse() res.DirectIOEventResponse.EventID = ev.EventID
	End If

手順	イベント処理-LongPolling 方式 Visual Basic.NET
2	' イベントへの応答 (WS-POSサービスプロバイダ への応答)
続き	While (True)
196	Try
	device.SetEventResponse(consumerID, res)
	' 応答が正常に完了したら、次のイベント取得のためPollForUPOSEventに戻る
	Exit While
	Catch ex As limeoutException
	イベントへの応答がオフラインまたはタイムアウトの場合、イベント応答を再試行
	Continue While
	Catch as As EquitException (Of UnifiedPOS POSPrinter V1 2 UnesException)
	If ex Detail ErrorCode = UnifiedPOS POSPower V1_2 ErrorCode Illegal Then
	- IIIFGAL が帰ってきた場合は SatEventResponse は時に受信されていたので
	Fit While
	Fise
	Throw
	End If
	End Try
	End While
	Catch ex As TimeoutException
	イベントポーリングがオフラインまたはタイムアウトの場合、PollForUPOSEventを再試行
	Continue While
	End Try
	End While
	End Sub

<Visual C#>

手順	イベント処理-LongPolling 方式 Visual C#		
1	LongPolling用のスレッドを動作させ、サービスプロバイダにイベント取得リクエストを送る処理を追加します。		
	private Thread longPollingThread;		
	// セッションオープン device OpenSession(consumerID_null):		
	// LongPolling用スレッド作成・開始		
	longPollingThread = new Thread(new ThreadStart(POSPrinterEvent_LongPolling)); longPollingThread IsBackground = true;		
	longPollingThread.Start();		
2	POSPrinterEvent_LongPollingメソッドにイベント取得時の処理を記載します。以下を参考にしてください。		
	void POSPrinterEvent_LongPolling()		
	while (consumerID != null)		
	try		
	UnifiedPOS.POSPrinter.V1_2.POSPrinterEvent deviceEvent;		
	// イベントホーリングを実行 deviceEvent = device.PollForUPOSEvent(consumerID);		
	if (deviceEvent == null)		
	t continue; // イベントがなかった場合、再度ポーリング		
	}		
	UnifiedPOS.POSPrinter.V1_2.POSPrinterEventResponse res = new UnifiedPOS.POSPrinter.V1_2.POSPrinterEventResponse();		
	// 受信したイベントごとの処理を実行 if (deviceEvent.ErrorEvent != null)		
	// ErrorEvent 取得時の処理 UnifiedPOS.POSPrinter.V1_2.ErrorEvent ev = deviceEvent.ErrorEvent; res.ErrorEventResponse = new UnifiedPOS.POSPrinter.V1_2.ErrorEventResponse(); res.ErrorEventResponse.EventID = ev.EventID;		
	} else if (deviceEvent.OutputCompleteEvent != null) {		
	、 // OutputCompleteEvent 取得時の処理 UnifiedPOS.POSPrinter.V1_2.OutputCompleteEvent ev = deviceEvent OutputCompleteEvent:		
	res.OutputCompleteEventResponse = new UnifiedPOS.POSPrinter.V1_2.OutputCompleteEventResponse(); res.OutputCompleteEventResponse.EventID = ev.EventID;		
	} else if (deviceEvent.StatusUpdateEvent != null)		
	1 // StatusUpdateEvent 取得時の処理 UnifiedPOS.POSPrinter.V1_2.StatusUpdateEvent ev = deviceEvent.StatusUpdateEvent:		
	res.StatusUpdateEventResponse = new UnifiedPOS.POSPrinter.V1_2.StatusUpdateEventResponse();		
	res.StatusUpdateEventResponse.EventID = ev.EventID; }		
	else if (deviceEvent.DirectIOEvent != null) {		
	// DirectIOEvent 取得時の処理 UnifiedPOS.POSPrinter.V1_2.DirectIOEvent ev = deviceEvent.DirectIOEvent; res.DirectIOEventResponse = new UnifiedPOS.POSPrinter.V1_2.DirectIOEventResponse();		
	res.DirectIOEventResponse.EventID = ev.EventID; }		

```
手順
       イベント処理-LongPolling 方式
                                  Visual C#
             // イベントへの応答(WS-POSサービスプロバイダへの応答)
2
             while (true)
続き
            {
              try
               {
                device.SetEventResponse(consumerID, res);
                // 応答が正常に完了したら、次のイベントを取得するためPollForUPOSEventに戻る
                break;
               }
               catch (TimeoutException)
               {
                // イベントの返信がオフラインで帰ってきた場合、PollEventではなく、再度SetEventResponseを行う
                continue;
               }
               catch (FaultException<UnifiedPOS.POSPrinter.V1_2.UposException> ex)
               {
                if (ex.Detail.ErrorCode == UnifiedPOS.POSPrinter.V1_2.ErrorCode.Illegal)
                {
                  // ILLEGALが帰ってきた場合は、SetEventResponseは既に受信されていたので、
                   // 再度PollForUPOSEvent に戻る
                  break;
                }
                else
                {
                  throw;
                }
              }
            }
          }
           catch (TimeoutException)
           {
            // イベントポーリングがオフラインまたはタイムアウトの場合、PollForUPOSEventを再試行
             continue;
          }
        }
```

3.3.10 KeepAlive 処理

OpenSession()後、アプリケーションとデバイス間で通信が無い時間が、サービスプロバイダ側で設定されているタイムアウト時間を超えると、デバイスとのコネクションが切断されます。KeepAlive はこれを防止するための仕組みです。

<Visual Basic .NET>

手順	KeepAlive 処理 Visual Basic.NET
1	タイマーを使用し、KeepAlive処理を行う処理を記載します。以下を参考にしてください。
	Dim timer As System.Windows.Forms.Timer timer = New Timer()
	AddHandler timer.Tick, New EventHandler(AddressOf timer_Tick)
	 'アプリケーション開始時に Keep Alive 用のタイマーを開始 Dim provider Session Timeout As Integer = device. GetProvider Session Timeout(consumerID) 'Service Provider で設定されているタイムアウト時間を取得 'ここでは、サービスプロバイダから取得したタイムアウト時間の半分を Keep Alive の時間とします timer.Interval = ((provider Session Timeout / 2) * 1000) 'タイマー開始 timer.Start()
	Private Sub timer_Tick(sender As Object, e As EventArgs) 'WS-POS KeepAlive If Not consumerID Is Nothing Then Try device.KeepAlive(consumerID) Catch ex As TimeoutException 'タイムアウト処理 timer.Stop() 'KeepAlive がタイムアウトになった場合(オフライン) 'チャネル情報、ConsumerIDをnullにします consumerID = Nothing device = Nothing 'LongPollingでイベント取得時は、LongPollingスレッドを停止させます 'この後、再度Openし直してください End Try End If
	End Sub

<Visual C#>

手順	KeepAlive 処理 Visual C#
1	タイマーを使用し、KeepAlive処理を行う処理を記載します。以下を参考にしてください。
	private System.Windows.Forms.Timer timer;
	timer.Tick += new System.EventHandler(timer_Tick);
	<pre>// アプリケーション開始時に KeepAlive 用のタイマーを開始する int providerSessionTimeout = device.GetProviderSessionTimeout(consumerID); // Service Provider で設定されているタイムアウト時間を取得する // ここでは、サービスプロバイダから取得したタイムアウト時間の半分を KeepAlive の時間とします timer.Interval = providerSessionTimeout / 2 * 1000; // タイマーを開始 timer.Start();</pre>
	<pre>private void timer_Tick(object sender, EventArgs e) { // WS-POS KeepAlive if (consumerID != null) { try { device.KeepAlive(consumerID); } catch (TimeoutException) { // タイムアウト処理 timer.Stop(); // KeepAlive がタイムアウトになった場合(オフライン) // チャネル情報、ConsumerIDをnullにします consumerID = null; device = null; // LongPollingでイベント取得時は、LongPollingスレッドを停止させます // この後、再度Openし直してください } } } }</pre>

3.4 OPOS のメソッド・プロパティについて

生成したチャネルのメンバーとして、OPOSと同様のメソッド・プロパティが使用できます。 メソッド・プロパティについては、OPOSのAPG(アプリケーションプログラマーズガイド)を参照してください。

*FP-1100 OPOS の場合は下記の APG を参照してください。 「FP-1100 POSPrinter, CashDrawer シリアル・USB・LAN インターフェース対応 OPOS-OCXドライバ アプリケーションプログラマーズガイド」 (FP1100_OPOS_APG_ja.pdf)

3.5 WS-POS で使用できないメソッド・プロパティ

下記の OPOS のプロパティ・メソッドは WS-POS で使用できません。 コンシューマおよびプロバイダが停止する場合がありますので、使用しないでください。

- ◆メソッド
 - CompareFirmwareVersion
 - ResetStatistics
 - PrintMemoryBitmap
 - DrawRuledLine
- ◆プロパティ
 - BinaryConversion
 - OpenResult
 - ResultCode
 - ResultCodeExtended
 - CapRecRuledLine
 - CapSlpRuledLine

4. 注意事項

PCを休止またはスリープ状態にする場合 コンシューマ実行中に PCを休止またはスリープ状態にすることは推奨しません。 通信がない時間が続いた場合、タイムアウトとなり、プロバイダとのセッションが切断する可能性があ ります。その場合、再度アプリケーションを起動してください。復旧しない場合はサーバー側でサービ スプロバイダの再起動を行ってください。

➢ SendTimeout 値の変更について

WCF サービスのバインディングプロパティの SendTimeout の既定値は"00:01:00"です。コンシュー マ側でバッファサイズが大きいデータを送信し、1分間処理が返ってこない場合、タイムアウトとなり、 アプリケーション内で例外が発生する可能性があります。その場合、サービスコンシューマ側で以下 のように SendTimeout 値を変更してください。

サービスコンシューマ設定ファイルに以下の太字部分を追加します。

<client>

 $<\!\!\mathsf{endpoint\ address}='' \mathsf{http://192.168.1.120:8087/POSPrinter.svc/POSPrinter1''$

binding="basicHttpBinding"

bindingConfiguration="longTimeBinding"

 $contract = ``Unified POS.POSPrinter.V1_2.POSPrinter''$

```
name="POSPrinterPort1"/>
```

</client>

<bindings>

```
<basicHttpBinding>
```

<binding name="longTimeoutBinding" SendTimeout="00:01:00"/> ····※

</basicHttpBinding >

```
</bindings >
```

※ご使用の環境により、適切なタイムアウト値を設定してください。

➢ WS-POS 参照実装の 64bit 環境での使用について

現在提供されている WS-POS 参照実装のプロバイダは 64bit 環境で動作しません。 ^{"WcfServiceHostWindowsService35.exe"が「Any CPU」でビルドされているため、サービスとして 動作させると 64bit 動作となり、32bit である CCO にアクセスできないためです。64bit 環境で動作さ せるには 32bit(x86)でビルドする必要があります。}

5. 改訂履歴

Revesion	更新日
1.0.0.0	初版(2014.09.12)

FUJITSU