WS-POS 技術仕様書 日本語版

Version 1.2

July 15, 2013

日本語訳：**2013** 年 **7** 月 **30** 日版

注：本文書は **ARTS** の承認を得て次世代 **POS** 研究会が翻訳した **WS-POS1.2** の日本語参考訳です。英語版と日本語版で内容が異なる場合は、英語版の内容が優先されます。

Chairman UnifiedPOS Committee:

H Paul Gay	Epson America

UnifiedPOS Committee Members:

Gerald Armentrout	IBM
Kazunori Chihara	Seiko Epson Corporation
Kunio Fukuchi	Fujitsu Frontech Limited
Tadashi Furuhata	Seiko Epson Corporation
Denis Kuniss	Wincor-Nixdorf
Jürgen Moser	Bizerba
Lawrence Owen	Star Micronics Co., Ltd.
Daniel Schwertführer	Bizerba
Brian Spohn	NCR Corporation
Michael Webb	Data Logic, Inc.

Contributors:

Richard Halter	NRF-ARTS

WS-POS Version 1.2 作成に協力した次世代 **POS** 研究会

次世代 POS 研究会 議長:

Toru Yanagisawa	NEC Infrontia Corporation.
Masanori Sambe	Toshiba TEC Corporation
Takao Tamura	Sorimachi Giken Co., Ltd.

次世代 POS 研究会 主メンバー:

Tadashi Furuhata	SEIKO EPSON Corporation
Hideo Nakamura	SEIKO EPSON Corporation
Kunio Fukuchi	Fujitsu Frontech Limited
Toyohiro Yasumoto	Vinculum Japan Corporation
Kiyotaka Abe	Sorimachi Giken Co., Ltd.
Kenichi Nagai	Star Micronics Co., Ltd.
Yuji Mori	Star Micronics Co., Ltd.
Akio Tajima	NCR Japan, Ltd.
Takahiro Akutsu	Hitachi Information & Communication Engineering, Ltd.
Mitsuhiro Igarashi	NEC Infrontia Corporation
Takahide Kubota	Toshiba TEC Corporation
Kazunori Chihara	SEIKO EPSON Corporation

次世代 POS 研究会 貢献者:

Soichi Fujii	Microsoft Co., Ltd.
Hiroshi Ota	Microsoft Co., Ltd.
Mitsuo Nagata	Vinculum Japan Corporation

目次

TABLE OF FIGURES

1. 摘要

1.1 要旨

W3C グロサリーhttp://www.w3.org/TR/ws-gloss/では、Web サービスを以下のように定義している。「a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.」
Web Services for Point of Service(WS-POS)は、これらの W3C 仕様を利用することによって、POS デバイスや端末、サーバを遠隔で相互運用し、典型的な小売業の LAN をよりダイナミックに運用するために必要な技術仕様である。

WS-POS バージョン 1.1 では、すべての UnifiedPOS 1.13 で定義された周辺機器を WSDL によりサポートし、同時に仕様改善と不具合の修正を行った。日本の次世代 POS 研究会(OPOS)での実装経験と提案をもとに更新された。

WS-POS バージョン 1.2 は、旧バージョンをベースとして、ネットワーク経由のアプリケーション連携を強化するために「ハートビート」方式の追加、デバイスからのイベント取得方法に「ロングポーリング」方式の追加、またセッション管理のためのコンシューマ ID の概念が追加された。

このドキュメントは、ネットワーク経由でデバイスの相互運用性を支援するために必要な最小のウェブサービス仕様のプロフィールである。 このプロフィールは、相互運用を促進し、また店舗で Web サービスを実装するために適切な WS 仕様 を選択可能にする。WS-POS 仕様は、Point of Service ソリューションで容易な相互運用可能な実装を実現するために、Web サービス UnifiedPOS XMLPOS や他の重要な技術を使用する。

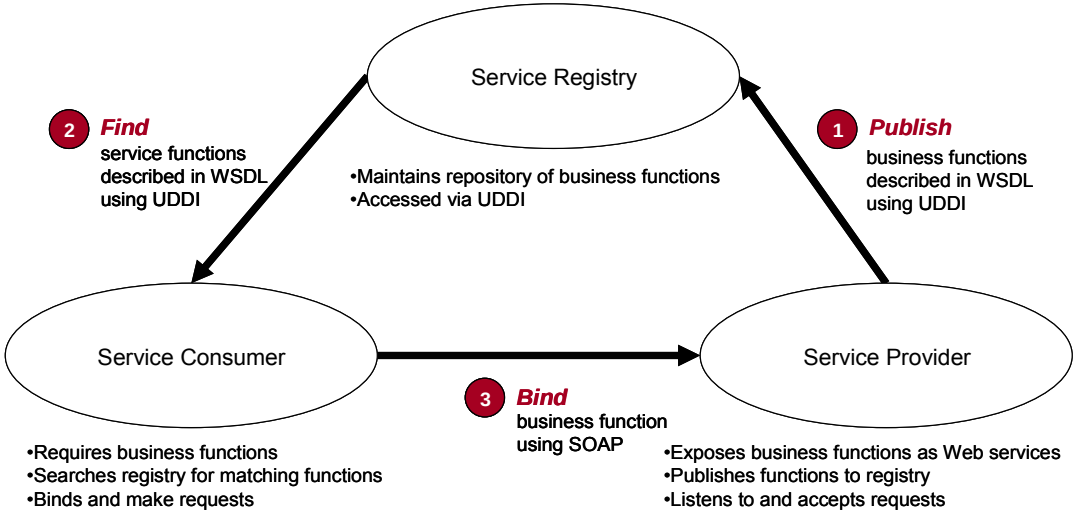


Figure 1: Web サービスアーキテクチャ

図１は、一般的な、というよりは WS-POS で使われている、 Web サービス に対する、典型的な接続モデルを示している。まず、①サービスプロバイダがサービスレジストリ（UDDI）にサービスを登録する。次に、②サービスコンシューマは自分の目的にあったサービスのロケーションを検索する。最後に、③サービスコンシューマはサービスプロバイダに直接バインドする。

1.2　準拠するための要件

本仕様書で定義された相互運用性を守るためには、本仕様書にて要求されているすべての要件を満たす実装を行わなければならない。これらの要件は下記の書式で定義されている。

Id	名称	説明
XY001	要件	これは要件の記述である。

1.3 ARTS WS-POS 標準スタック

次の図は、ARTSのWS-POSスタックを示す。

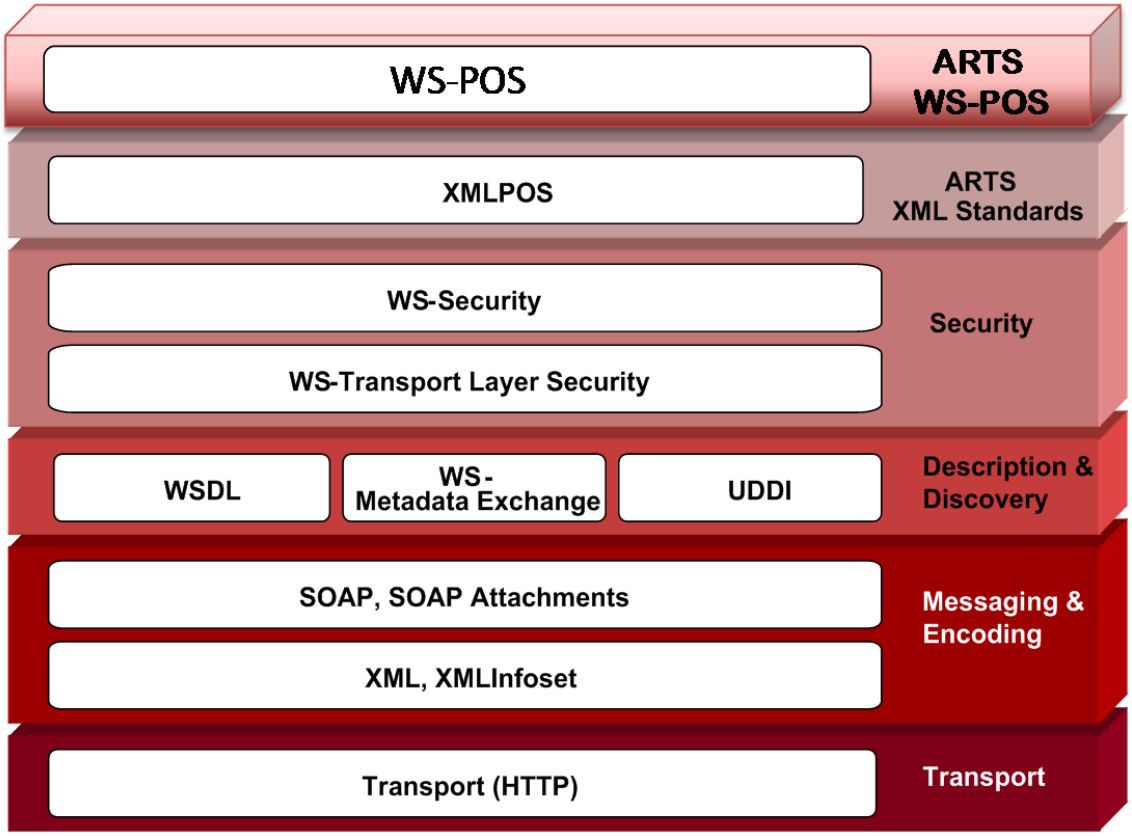


Figure 2: ARTS WS-POS スタック

1.3.1 Web サービスコンポーネント

WS-POS の Web サービス定義は以下を主として取り扱う。

- 基本メッセージ：WS-POS が拠りどころとする、すべてのメッセージに関する記述
- サービス探索：リソースやサービスがどのように配置されるか
- サービスの記述：対象とするサービスやメッセージをどのように記述するか

1.3.2 追加コンポーネント *Version 1.2 で更新*

- セキュリティ：ふさわしいセキュリティレベルを維持するために最適なメカニズムの記述を提供
- XML Payload: The payload of a message. XML ペイロード：メッセージのペイロード
- サービス定義：Universal Data Discovery and Integration (UDDI)を用いて、利用可能なデバイスを簡単にクライアントが見つけられるようにするためには、デバイスサービスの名称と定義を決める必要がある。
- ***(Version 1.2 で追加)*** WS-POS サービスプロバイダと WS-POS サービスコンシューマ間の意図しない切断を取り扱う方法（2.4 章 WS-POS 動作モデルを参照）

1.4 対象としない範囲 *Version 1.2 で更新*

Web サービスの管理、動的なサービスディスカバリ、Web サービスレベルのイベントはこのバージョンでは取り込まれていない。

このバージョンは店舗内環境のデータフローに制限している。将来版は、企業間のデータフローを含めるためにスコープを拡大するかもしれない。これは、UDDI、クライアント、デバイスの相互運用を実装する際、適切なシステム設計ガイダンスの下でセキュリティに対して十分に考慮が必要ということを示している。

本仕様書では以下を対象としない：

- デバイスサービスのリストからデバイスを選択する方法。アプリケーションでの管理が必要。
- セキュリティ証明書をデバイスに設定する方法。またはセキュリティ証明書を認証する方法。
- デバイスまたはクライアントが有効であることを確認する方法。
- Payment Card Industry (PCI) データ要件の記載。メッセージ内の情報が PCI 標準に基づく必要があると認識される。これらの要件は PCI committee によって定義される。

2. WS-POS コンポーネント

本セクションは WS-POS スタックの各コンポーネントの詳細を説明する。

2.1 WS-POS 関連用語の解説

2.1.1 XML POS

ARTS によって標準化された POS デバイスがサポートすべき操作、状態、属性の XML による定義である。

2.1.2 Web サービス

さまざまなプラットフォーム上で動作する異なるソフトウェア同士が相互運用するための標準的な手段を提供するものである(W3C)。実装の多くは、HTTP をトランスポートとして、SOAP とよばれる XML 形式で情報をやり取りするためのプロトコルを用い、WSDL という定義言語で定義される。技術そのものを指す場合と、その技術によって実装されたサービスを指す場合がある。

2.1.3 WS-POS 仕様に準拠した Web サービス

本書では、以降、WS-POS 仕様に準拠した Web サービスのことを、WS-POS サービスと記載する。

2.1.4 WS-POS サービスのプロバイダとコンシューマ

本書では、以降、WS-POS サービスを提供する側を WS-POS サービスプロバイダ、サービスを利用する側を WS-POS サービスコンシューマと記載する。

2.1.5 WS-POS におけるメソッド

WS-POS におけるメソッドは、Web サービスの Operation(操作)として表現される。本書では、Web サービスの操作という文言ではなく、WS-POS メソッドと表現する。

2.1.6 WS-POS におけるプロパティ

WS-POS におけるプロパティは、Getter/Setter メソッドを用いて実現する。たとえば、UnifiedPOS で定義されている DeviceEnabled プロパティは、Getter メソッドとして GetDeviceEnabled メソッド、そして Setter メソッドとして SetDeviceEnabled メソッドで実現する。読み取り専用プロパティには、Setter メソッドがない。本書では、以降、Getter メソッド/Setter メソッドを明示的に記述しない。プロパティを取得/設定する際には、Getter メソッド/Setter メソッドが用いられていると認識すること。

2.1.7 WS-POS におけるイベント

WS-POS におけるイベントは、Unified POS で定義されているデバイスイベントを、WS-POS サービスプロバイダから WS-POS サービスコンシューマに通知することである。

WS-POS は、WS-POS サービスコンシューマが Unified POS で定義されたデバイスイベントを受信するために、Web サービスからクライアントを呼び出す。イベントを双方向通信で受信する POS サービスコンシューマは、受信するためのエンドポイントをサービスに通知しておく必要がある。

2.2 基本メッセージ

2.2.1 説明

基本メッセージは、本仕様書がカバーするすべての範囲において、核心の基礎として使われる技術のセットである。

2.2.2 WS-I 基本プロファイル

WS-POS サービスは、WS-I 基本プロファイル バージョン 1.2 をベースに、WSDL1.1 により記述される。HTTP 上の SOAP 1.1 を、WS-Addressing Core 1.0 と Web Service Description Language (WSDL) 1.1 とともに使っている。WSDL 1.1 の記述は、提供されるサービス（タイプ/インターフェース）、利用方法（bind）、場所（endpoint）を含んでいる。これらは、Web サービスを利用する時の主流となっており、Web サービスの“世界”での運用における“グランドルール”として見なされている。

Id	名称	説明
MB001	基本プロファイル	実装は WS-I 基本プロファイルに準拠しなければならない

2.3 サービスの記述と探索

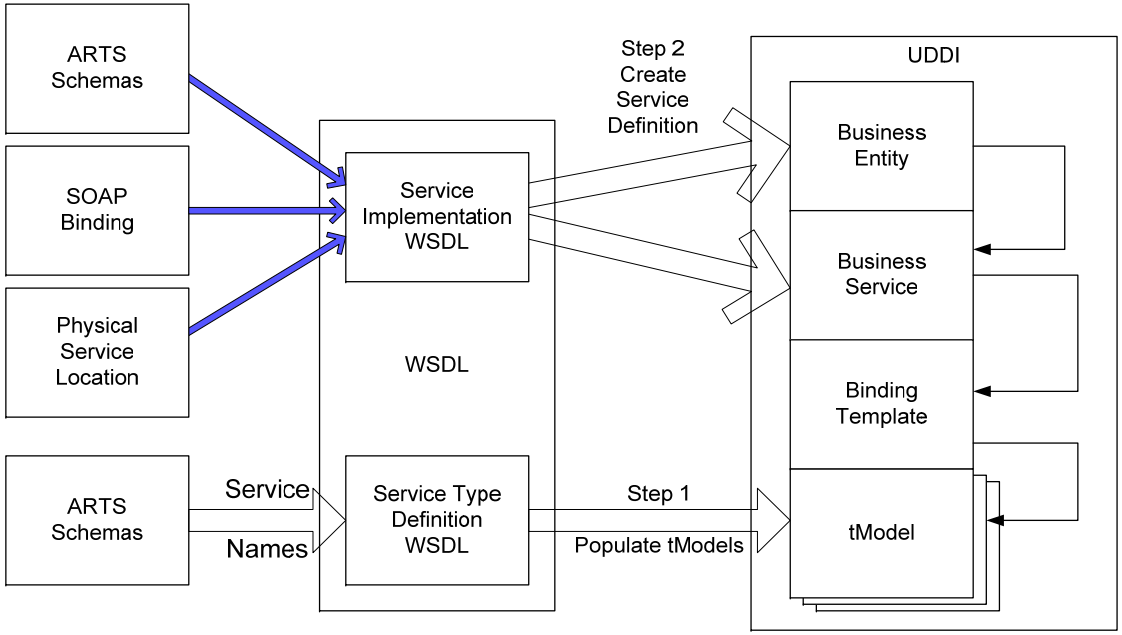


Figure 3: ARTS スキーマ – WSDL と UDDI の関係

UDDI レジストリへの事前設定は２ステップの操作となる。通常サービスプロバイダから提供される適切なサービス情報とともに WSDL から tModel(Technical Model) が、最初に事前設定される。その後、特定のビジネスサービスを実施するために必要な tModel のグループに対するリンクと一緒に、ビジネスサービスが事前登録される。

サービスコンシューマは、その役割に必要なビジネスサービスを、レジストリから見つけるだろう。適切なビジネスサービスが決まれば、サービスコンシューマは、個々の Web サービスに接続するための情報を UDDI レジストリに対して問い合わせることになる。

WS-POS サービスの発見には、UDDI 3.0.2 が使用できる。WS-POS サービスプロバイダは UDDI のサービスレジストリに WS-POS サービスを登録する。

WSDL の記述と、UDDI の登録情報の関係は以下である。

WSDL	UDDI
types message binding	tModel
Service	businessService
Port	bindingTemplate

また、UDDI の businessEntity には、ARTS Device Type(Scanner,　POSPrinter など)を設定する。

2.3.1　サービス記述

2.3.1.1 説明

標準化されたデータは相互運用性を以下の方法で確保する。

1. サービスプロバイダが提供するサービスを、事前に記述すること
2. サービスプロバイダが提供するサービスの機能を呼び出すために必要なメッセージの構造を定義すること
3. サービスプロバイダとサービスコンシューマが同期してメッセージをやり取りが行えていることを確実にすること

良いサービスの記述による利点は、サービスプロバイダとサービスコンシューマ間の相互運用性に準拠した開発や確認を行う時に、効果的な市販ツールを活用できることになることである。

2.3.1.2 メタデータ

WS-IBasic Profile V1.2 で記述される WSDL1.1 は、サービスの記述の基本部分として使われる。

WS-MetaDataExchange は、活用される Web サービスの関連するメタデータを取得するために必要である。

2.3.2 探索

Version 1.2 で更新

2.3.2.1 説明

LAN(もしくは LAN に接続していないデバイスの代理)に接続している周辺デバイスを探索し動作することは WS-POS にとって重要な要求事項の一つである。そのため、既存のサービスや（その後にでてくる新しいサービス）を探索することは、システムソリューションの中でも重要な部分となる。

2.3.2.2 Universal Description, Discovery, and Integration (UDDI)

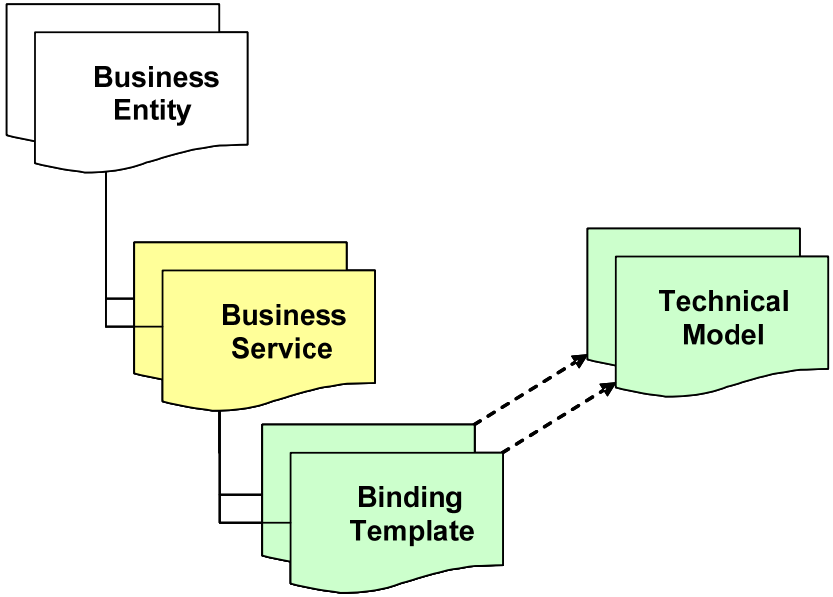


Figure 4: UDDI 概観

WS-POS は Universal Description Discovery and Integration(UDDI) 仕様のバージョン 3.0.2 を使う。UDDI レジストリはサーバ上で動作しており、ネットワーク上で Web サービスを登録するためや、他の Web サービスを探索するための単一の場所を提供する。

Id	名称	説明
DI001	UDDI	実装は UDDI version 3.0.2 に準拠しなければならない

2.3.2.3 UDDI に対するデバイスサービスの登録

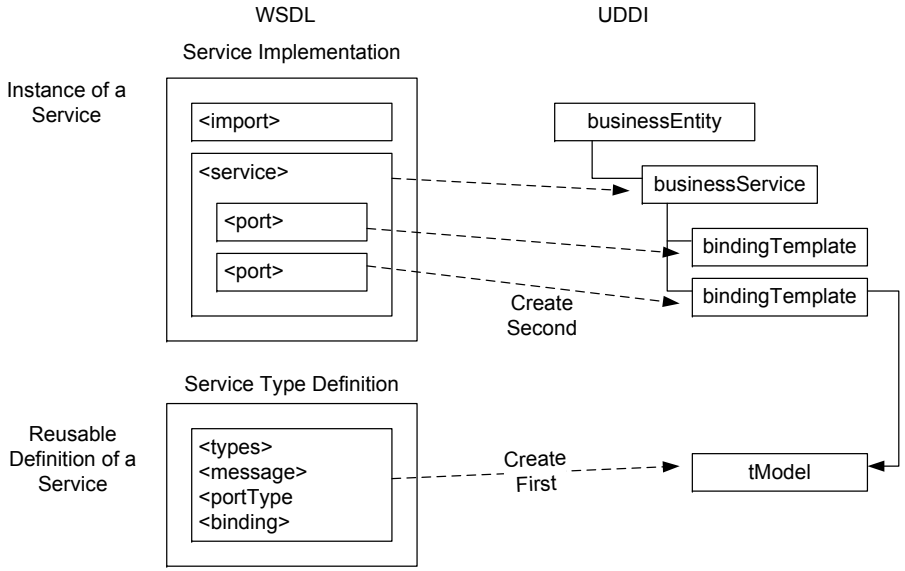


Figure 5: UDDI レジストリに対する WSDL の登録

マッピング方法を用いたデバイスサービスの UDDI への登録は、OASIS “Usigin WSDL in a UDDI Registry” Version 2.0.2 のテクニカルノートに記載されている。

2.3.2.4 UDDI によるサービスの探索

サービスの提供場所を特定するためには、下記の表に示したような、求めているサービスを含むビジネスエンティティ（実体）に対する UDDI を探索することが求められている。

ARTS Device Type (businessEntity)	User Service (businessService)	Administrative Service (businessService)
Scanner	ScanItems	ScannerAdmin

2.3.2.5 UDDI からのデバイスサービスの削除

デバイスサービスは、サービスを止める前に UDDI から登録が削除されているようにすべきである。

2.4 WS-POS 動作モデル *Version 1.2 で追加*

WS-POS では、WS-POS サービスを利用する WS-POS サービスコンシューマと、WS-POS サービスを提供する WS-POS サービスプロバイダの協調的な動作を規定する。

WS-POS Version 1.2 では以下の動作の規定が追加された。

動作	動作追加の背景
イベントポーリング方式	Web ブラウザ上で動作する RIA(Rich Internet Application)は、ブラウザセキュリティによってポートを開けない。このため、RIA では WS-POS バージョン 1.1 で記述されている、WS-POS サービスコンシューマがイベント受信用のエンドポイントをセルフホスティングする仕様は実装できない。
ネットワーク切断監視	WS-POS バージョン 1.1 ではネットワーク通信の切断と復旧に対する考慮が不十分である。ネットワーク通信が切断した場合に、WS-POS サービスコンシューマも WS-POS サービスプロバイダも切断したことを検出できない。この状況では、WS-POS サービスプロバイダは Claim されたデバイスを Release すべきか否か判断できない。
セッション管理	WS-POS バージョン 1.1 では、複数の WS-POS サービスコンシューマが同じ WS-POS サービスプロバイダにアクセスする場合、サービスプロバイダがサービスコンシューマを識別できないため、誤ったメソッド呼出しシーケンスに対して脆弱である。

2.4.1 プロパティ・メソッド・イベントについて

Version 1.2 で追加

WS-POS は、ARTS UnifiedPOS の Device Behavior Model で記述されているプロパティ、メソッド、そしてイベントをネットワークを経由して扱う。プロパティ、メソッド、そしてイベントの詳細については、UnifiedPOS 仕様の Device Behavior Model を参照すること。

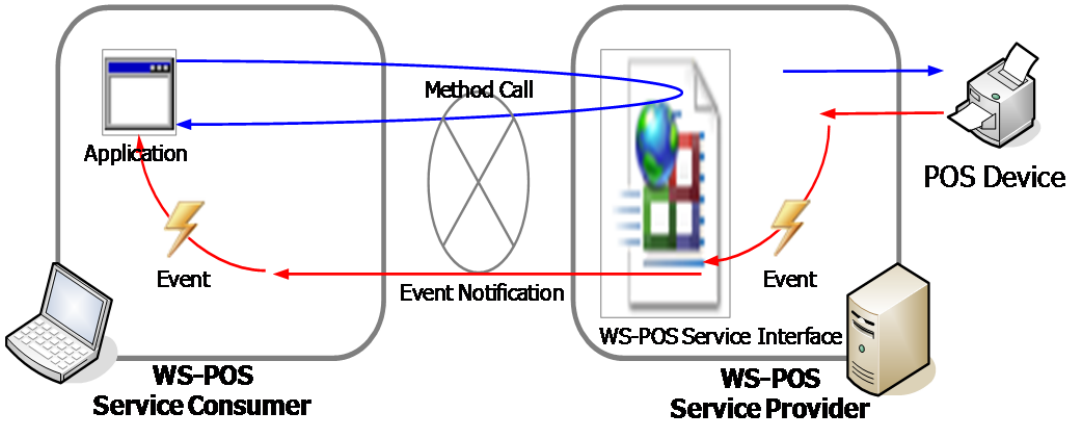
2.4.2 WS-POS 通信モデル

Version 1.2 で追加

WS-POS では、WS-POS サービスコンシューマと WS-POS サービスプロバイダ間での通信について、以下の２つのモデルを扱う。

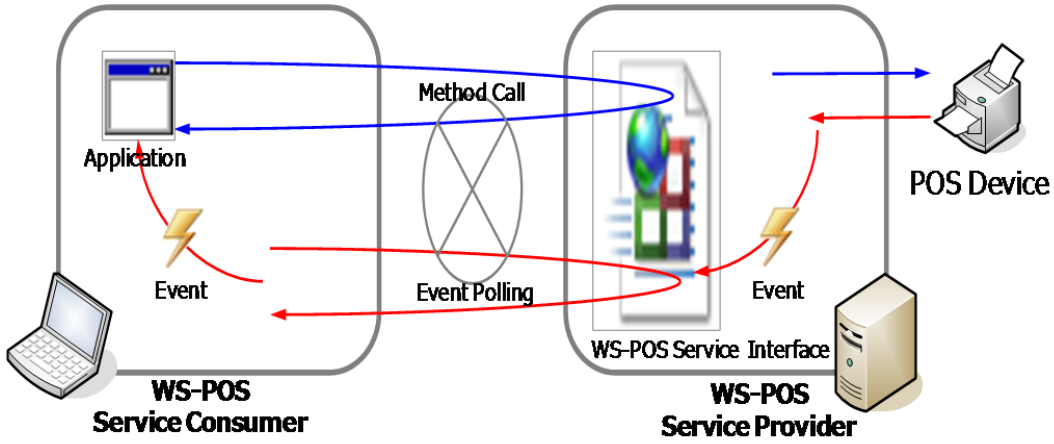
- 双方向通信
- ポーリング *(Version 1.2 で追加)*

双方向通信は、WS-POS サービスコンシューマがイベント受信のためのポートを開き、WS-POS サービスプロバイダからのイベントをポート経由で受信する通信方式である。概念図を下に示す。



この方式のイベント受信の詳細は 2.4.10 を参照すること。

ポーリングは、WS-POS サービスコンシューマが、WS-POS サービスプロバイダに対してイベントをポーリングするためのメソッドを呼び出すことで、イベントを取得する通信方式である。概念図を下に示す。



この方式のイベント受信の詳細は、2.4.11 を参照すること。

2.4.3　セッションマネージャとデバイス制御

Version 1.2 で追加

WS-POS では、WS-POS サービス コンシューマと WS-POS サービスプロバイダ間の通信セッションと、WS-POS サービスプロバイダが提供するデバイス制御の 2 つの機能を扱う。これらの機能は、両方とも API として WS-POS サービスプロバイダに提供されるが、API の目的が異なる。

WS-POS サービスプロバイダのセッションマネージャ機能は、WS-POS サービスコンシューマと WS-POS サービスプロバイダ間の通信を管理する。主な機能は、セッションのオープン、クローズ、そして複数の WS-POS サービスコンシューマが WS-POS サービスプロバイダにアクセスした際の、WS-POS サービスコンシューマの識別である。

WS-POS サービスプロバイダのデバイス制御機能は、WS-POS サービスプロバイダが制御下におく UnifiedPOS デバイスへのアクセスを提供する。(セッションではなく)デバイスの Open、Claim、有効化、Close の他、UnifiedPOS で定義されているデバイスのプロパティ、メソッド、イベントへのアクセスを行う。

セッション管理とデバイス制御の API についての詳細は、2.4.9 章を参照すること。

2.4.4 WS-POS セッションマネージャ概念の導入 *Version 1.2 で追加*

WS-POS サービスプロバイダは、複数の WS-POS サービスコンシューマによるリクエスト送信に応答する必要がある。このため、WS-POS では、個々の WS-POS サービスコンシューマの識別子(コンシューマ ID)を WS-POS サービスコンシューマが発行し、WS-POS サービスコンシューマと WS-POS サービスプロバイダとの間のネットワークセッションを WS-POS サービスプロバイダが管理する。

WS-POS コンシューマは自分自身を識別するためのユニークな識別子をコンシューマ ID として保持している必要がある。WS-POS コンシューマは WS-POS プロバイダに対して openSession メソッドを呼び出す際に、その識別子を使用する。それ以降の WS-POS コンシューマと WS-POS プロバイダの通信はこのコンシューマ ID によって識別されたセッションとして管理される。

各デバイスの WS-POS サービスプロバイダは、１つのセッションマネージャを持つ。セッションマネージャは、WS-POS サービスコンシューマ毎に通信セッションを管理し、そのセッションと１対１で関連付けた UnifiedPOS デバイスのインスタンスを保持する。（下図）

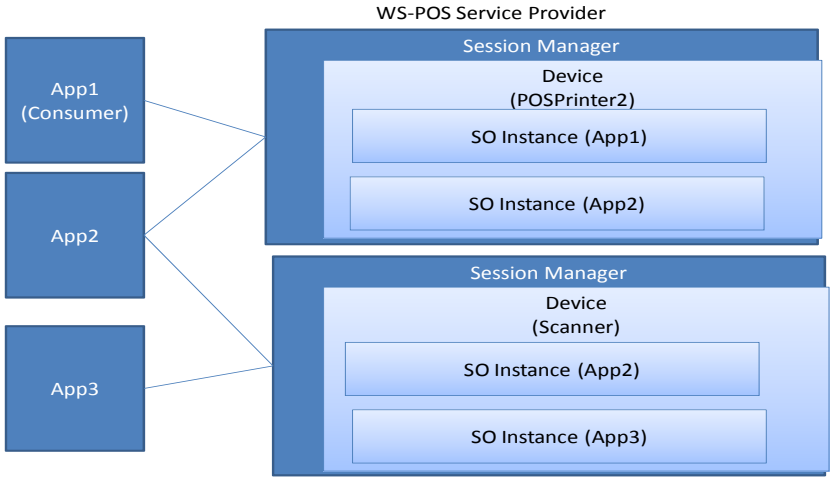


Figure 6: セッションマネージャ層とデバイス制御層

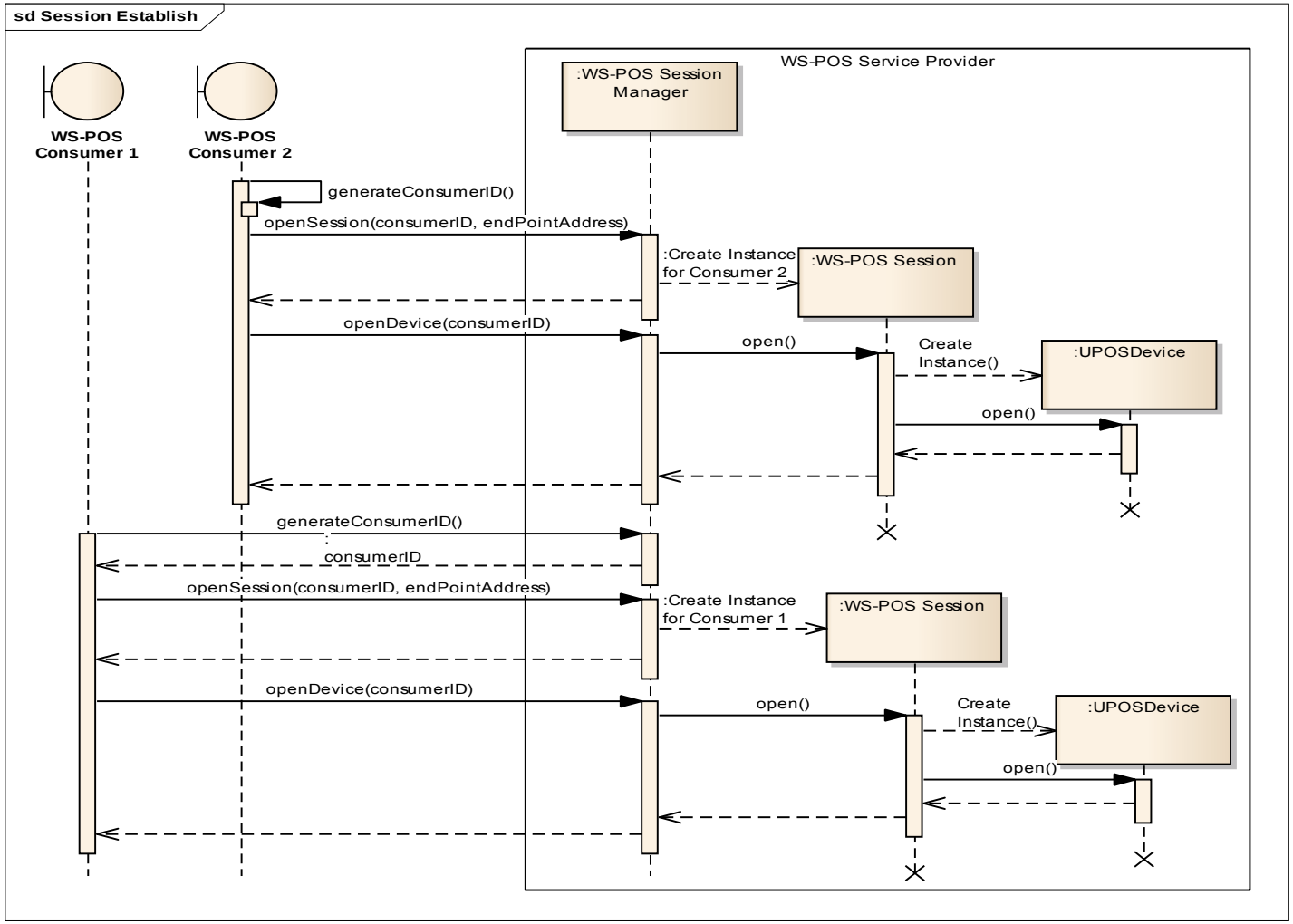
2.4.5 WS-POS セッションの識別 *Version 1.2 で追加*

WS-POS サービスプロバイダは、WS-POS コンシューマから指定される識別子(コンシューマ ID)を用いてセッションを識別する。 コンシューマ ID は WS-POS コンシューマが WS-POS プロバイダへの通信時に生成し指定する、一般的に一意な ID である。(Windows や.NET では GUID、Java では UUID を推奨する)

GUID や UUID を生成できないコンシューマ実行環境のために、WS-POS サービスプロバイダはコンシューマ ID の生成機能を必ず実装する。このリクエストは WS-POS コンシューマがユニークな ID を取得するために一度だけ必要で、そのユニークな ID をコンシューマが保持し、それ以降の WS-POS プロバイダとのセッションに使用することができる。コンシューマ ID はユニークである必要がある。WS-POS プロバイダの **openSession** メソッドにコンシューマ ID を渡す必要があり、その後のプロパティ・メソッド・イベントのファンクションすべてにコンシューマ ID が必要である。

2.4.6 WS-POS セッション確立の典型的なシーケンス *Version 1.2 で追加*

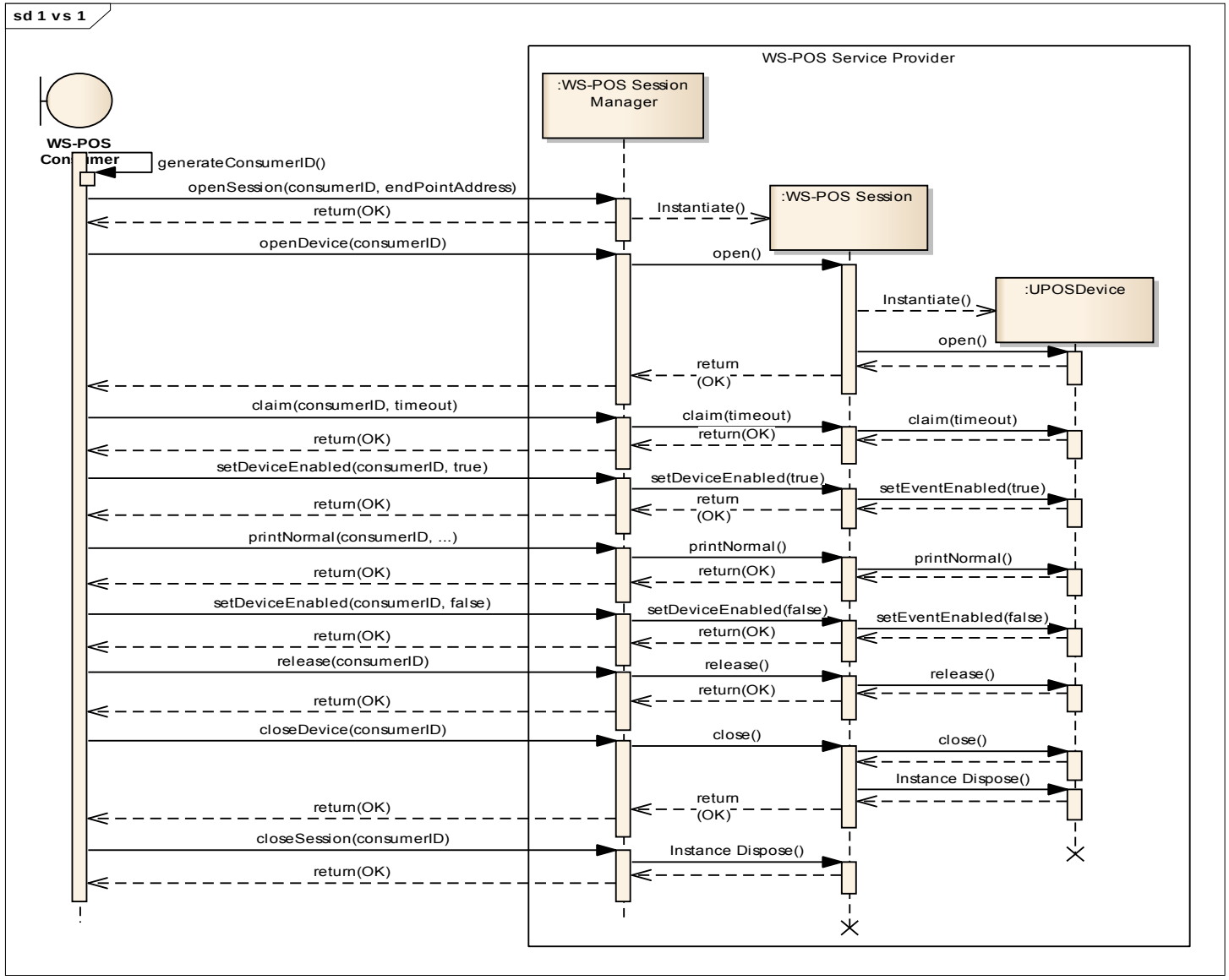
WS-POS におけるセッション確立までの典型的なシーケンスを、下図に示す。下図では、2 つの WS-POS コンシューマに対してそれぞれセッションを確立し UnifiedPOS デバイスをインスタンス化するシーケンスを示している。



Note: The Consumer ID is a unique GUID or UUID

WS-POS Consumer1 は、自らコンシューマ ID を生成せず、WS-POS プロバイダのセッションマネージャに対し、generateConsumerID メソッドを呼び出してコンシューマ ID を生成し使用している。WS-POS Consumer2 は、独自にコンシューマ ID を生成し使用している。

下図は、 POSPrinter の **PrintNormal** 呼び出しシーケンスを示したものである。.



上図では、全体的なシーケンスと共に、コンシューマ ID が WS-POS サービスプロバイダのセッションマネージャで使用されること(UnifiedPOS デバイスで使用されるのではない)を示している。セッションは **openSession** メソッドで始まり、**closeSession** メソッドで終了する。 UnifiedPOS デバイスドライバは **openDevice** メソッドでインスタンス化され、**closeDevice** メソッドで終了する。

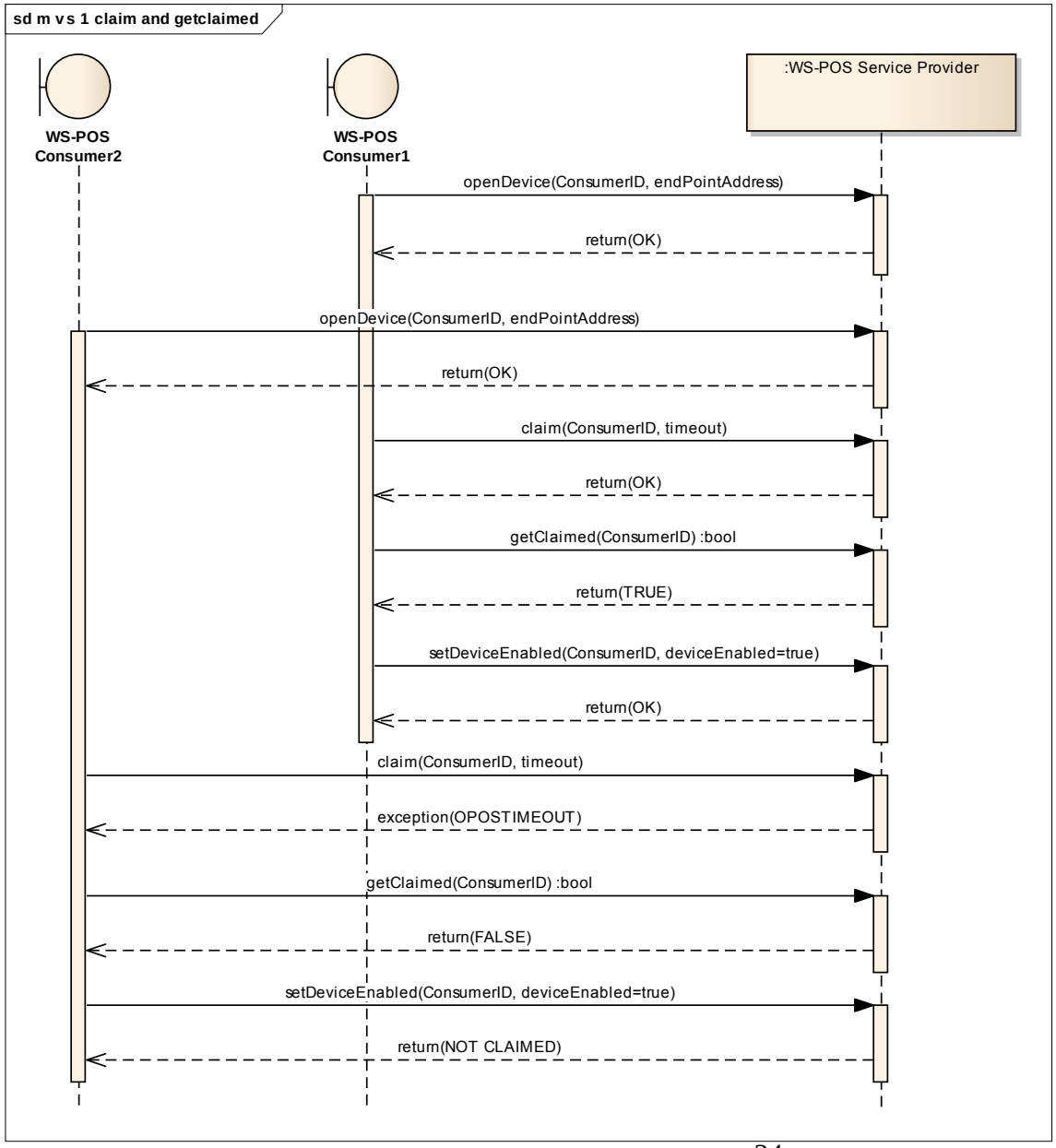
2.4.7 WS-POS サービスメソッドの呼出しとプロパティの使用 *Version 1.2 で追加*

WS-POS サービスコンシューマが WS-POS サービスプロバイダのメソッドを呼び出しまたはプロパティを使用するためには、WS-POS サービスプロバイダのエンドポイント URL を交換しなければならない。WSDL は、WS-POS サービスプロバイダのメソッドおよびプロパティへのアクセスを行うために使用される。
現代のプログラミングにおいては、 .NET の WCF、Java の JAX-WS など、WS-POS サービスのためのプロキシコードを WSDL から生成し、WS-POS サービスコンシューマはプロキシコードを経由して WS-POS サービスプロバイダの機能を使用する。

2.4.8 複数の WS-POS コンシューマによる単一の WS-POS プロバイダに対する Claim *Version 1.2 で更新*

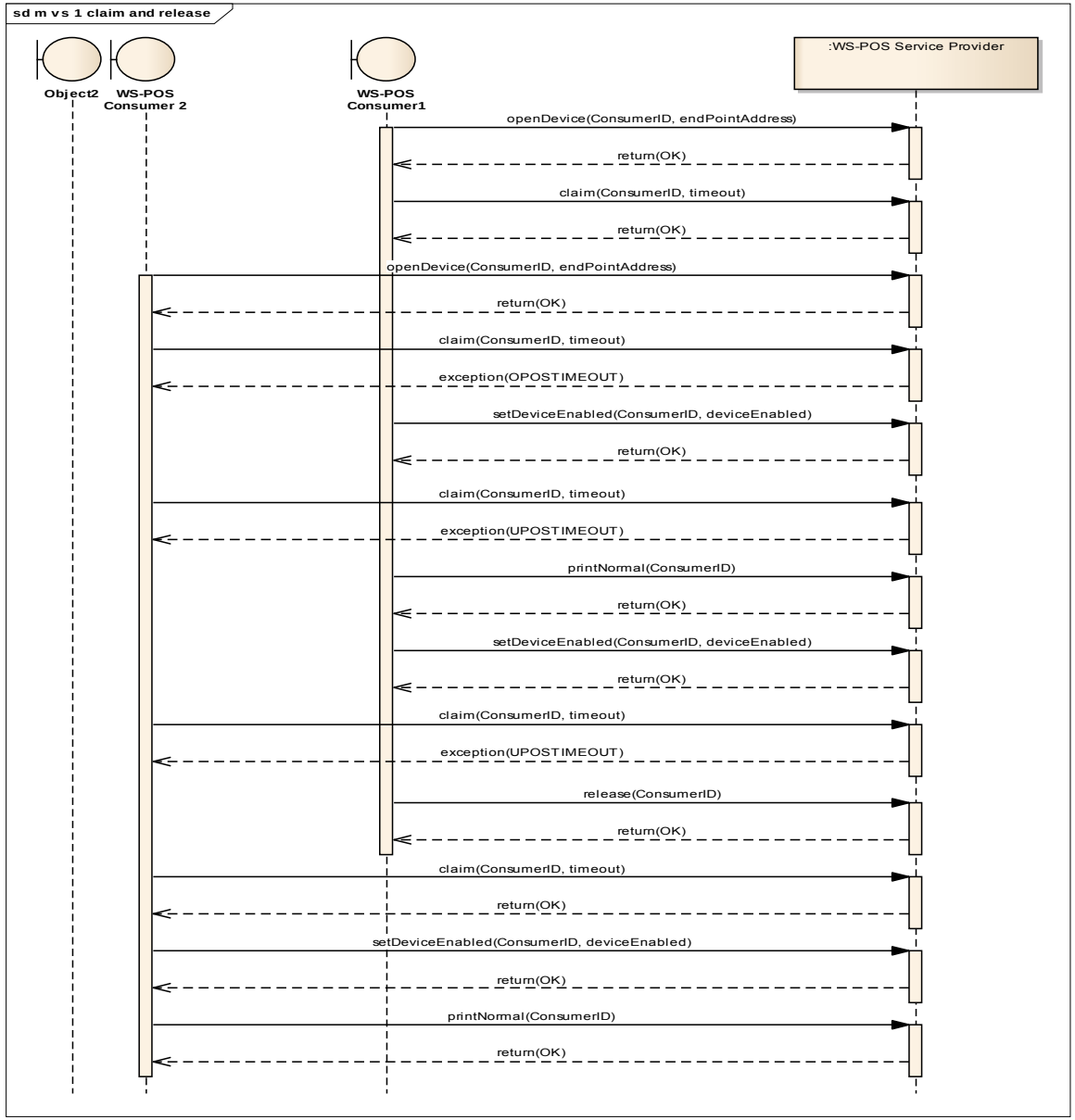
一度 WS-POS サービスコンシューマが **openDevice** メソッドによる初期化に成功したら、それ以降は UnifiedPOS の機能を使用する。WS-POS サービスコンシューマは WS-POS サービスプロバイダの **claim** メソッドに成功しなければならない。その後、WS-POS サービスプロバイダの機能を用いるために、**enable** メソッドを呼び出す必要があるかもしれない。

以下のシーケンス図は、UnifiedPOS デバイスの Claimed 状態を取得する過程を示す。



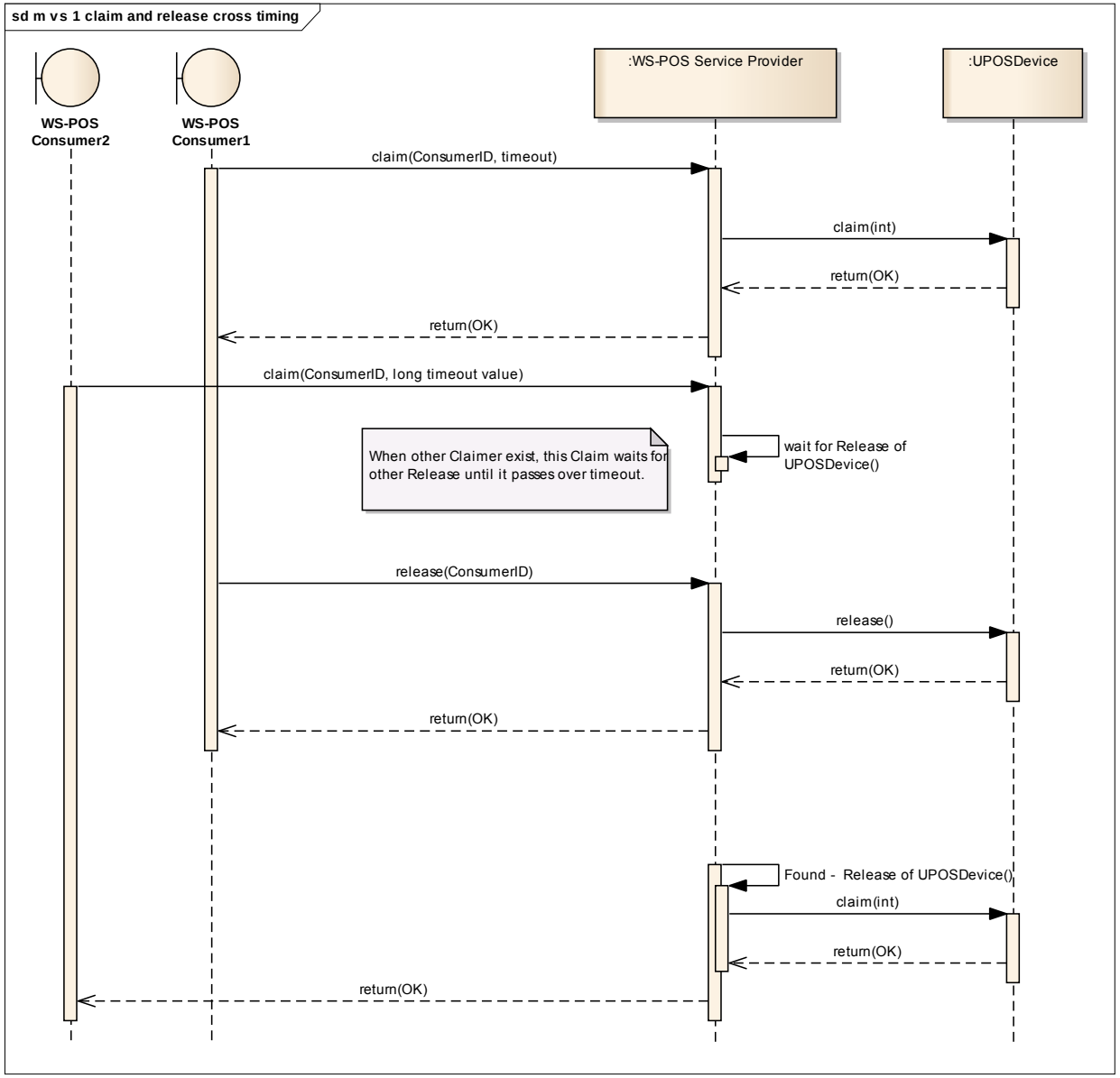
WS-POS サービスプロバイダに複数の WS-POS サービスコンシューマがリクエストを発行する際、**claim** メソッドに成功した（場合によっては **enable** メソッドも呼び出した）WS-POS コンシューマが WS-POS サービスプロバイダの機能を使用できる。

以下のシーケンス図は、WS-POS Consumer1 が **release** メソッドを呼び出すまで、WS-POS Consumer2 は WS-POS プロバイダを使用できないことを示す。



複雑なエンタープライズシステムでは、ある WS-POS サービスプロバイダの機能を多くの WS-POS サービスコンシューマが使用するかもしれない。その場合、調停メカニズムは WS-POS サービスコンシューマが希望する WS-POS サービスプロバイダに適時にアクセスすることを保証する。WS-POS では、複数の WS-POS サービスコンシューマが claim リクエストを発行する問題を解決するため、次の動作を定義する。以下は、複数の WS-POS サービスプロバイダの claim リクエストを扱う方法を示す。

1. WS-POS サービスコンシューマ 1 は WS-POS サービスプロバイダに対し **claim** メソッドを呼び出す。他の WS-POS サービスコンシューマがその WS-POS サービスプロバイダを使用していないため、**claim** メソッドは成功応答する。
2. 次に、WS-POS サービスコンシューマ 2 が同じ WS-POS サービスプロバイダ(WS-POS サービスコンシューマ 1 が使用中)に対して **claim** メソッドを呼び出す。WS-POS サービスプロバイダはそのリクエストをロングポーリング方式で保持し、応答を返さない。WS-POS サービスプロバイダは、WS-POS コンシューマ 2 が **claim** メソッドに渡したタイマ値を過ぎるか、そのタイマ値を過ぎるまでに WS-POS コンシューマ 1 が WS-POS サービスプロバイダの使用を終了するまで待つ。
3. タイマ時間を過ぎた場合、WS-POS サービスプロバイダは WS-POS サービスコンシューマ 2 にタイマ時間切れを返し、時間内に使えなかったことを示す。WS-POS サービスコンシューマ 2 は再度タイマ値を設定して **claim** メソッドを呼び出す。あるいは別の同じ機能を提供する WS-POS サービスプロバイダに対して **claim** メソッドを呼び出すかもしれない。
4. WS-POS サービスコンシューマ 1 が WS-POS サービスプロバイダを使用し、かつ WS-POS サービスコンシューマ 2 が WS-POS サービスプロバイダが使用可能になるのを待っている間に、WS-POS サービスコンシューマ 3 が同じ WS-POS サービスプロバイダに対して **claim** メソッドを発行した場合、WS-POS サービスプロバイダは WS-POS サービスコンシューマ 3 からのリクエストを WS-POS サービスコンシューマ 2 のリクエストの後に処理するようキューイングし、WS-POS サービスコンシューマ 3 からの **claim** メソッドに指定されたタイマ値を使用して WS-POS サービスコンシューマ 3 に対しても応答を返さない。
5. WS-POS サービスプロバイダは FIFO メカニズムを用いて **claim** メソッドの呼び出しキューを管理する。WS-POS サービスコンシューマ 1 が WS-POS サービスプロバイダを使用し終えると **release** メソッドが呼び出される。すると、WS-POS サービスプロバイダは WS-POS サービスコンシューマ 2 に対し claim が成功した旨の応答を返す。WS-POS コンシューマ 3 はキュー順が上昇し claim が成功した応答を待つ。あるいは、タイマ時間切れの通知を受ける。



2.4.9 WS-POS メソッドとデバイスメソッド

Version 1.2 で追加

WS-POS では、セッションマネージャ概念の導入に伴い、セッションマネージャ層にアクセスするためのメソッドとデバイス制御用のメソッドを分離する。セッションマネージャ層のメソッドを「WS-POS メソッド」、デバイス制御用のメソッドを「デバイスメソッド」と呼ぶ。

2.4.9.1 WS-POS メソッド

メソッド	説明	使用条件
generateConsumerID	コンシューマ ID を GUID または UUID として生成する	
openSession	WS-POS セッションを開始する。（デバイスの Open は行わない）	
closeSession	WS-POS セッションを終了する。	openSession
getProviderSessionTimeout	WS-POS プロバイダに設定されているセッションタイムアウト値をコンシューマが取得する。	openSession
keepAlive	WS-POS コンシューマは、セッションを保持するために WS-POS プロバイダのこのメソッドを定期的に呼出しする。	openSession
pollForUPOSEvent	WS-POS プロバイダに対しイベントが発生しているか否かのポーリングをコンシューマが行う。	openSession
setEventResponse	pollForUPOSEvent で取得したイベントに対し、WS-POS コンシューマが応答を設定する。	pollForUPOSEvent
getWSPOSVersion	WS-POS プロバイダが実装している WS-POS のバージョンを取得する。	

2.4.9.2 デバイスメソッド

メソッド	説明	使用条件

openDevice	UnifiedPOS デバイスの Open を行う。	openSession
claim	UnifiedPOS デバイスの Claim を行う。	openDevice
release	UnifiedPOS デバイスの Release を行う。	openDevice
closeDevice	UnifiedPOS デバイスの Close を行う。	openDevice
他の UnifiedPOS メソッド	UnifiedPOS デバイスへのアクセスを行う。 *(UnifiedPOS の仕様書を参照すること)*	openDevice(WS-POS)

2.4.9.3 WS-POS 1.2 で廃止するメソッド

セッション管理の導入に伴い、WS-POS 1.2 では、以下のメソッドを廃止する。

メソッド	説明
open	セッションを確立し、UnifiedPOS デバイスの Open を行う。
close	UnifiedPOS デバイスの Close を行い、セッションを終了する。

これらのメソッドは、セッション管理とデバイス制御の両方の機能を有していた。

WS-POS1.2 では、セッション管理とデバイス制御の機能を分離し、セッションの確立と終了は openSession と closeSession によって行い、デバイスの Open および Close は openDevice および closeDevice によって行うため、open および close メソッドは廃止する。

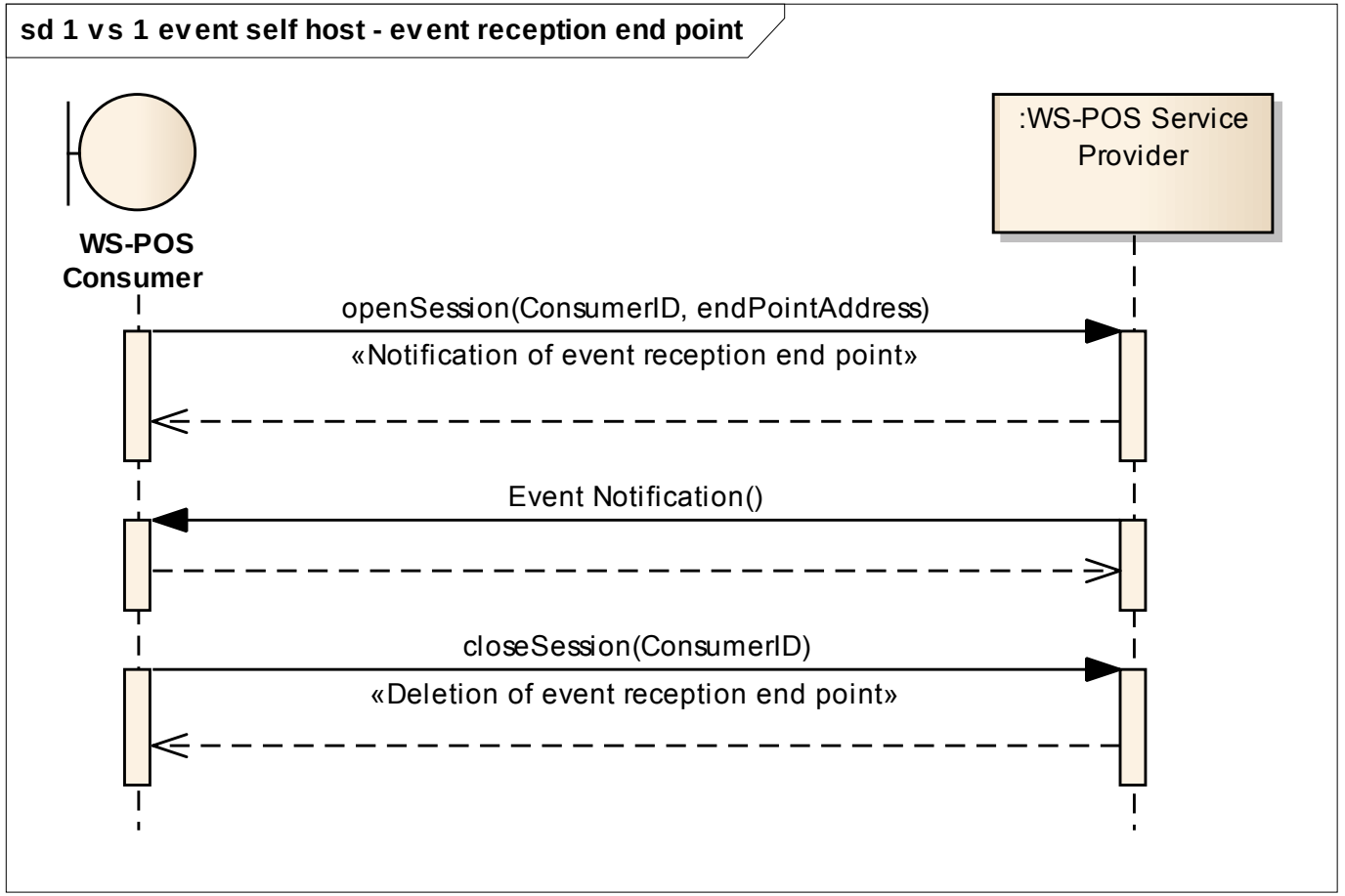
2.4.10 双方向通信を用いた WS-POS イベントハンドリング *Version 1.2 で更新*

WS-POS1.2 より前では、イベントは POS アプリケーションと POS 周辺機器が双方向通信を行うことを前提にしていた。

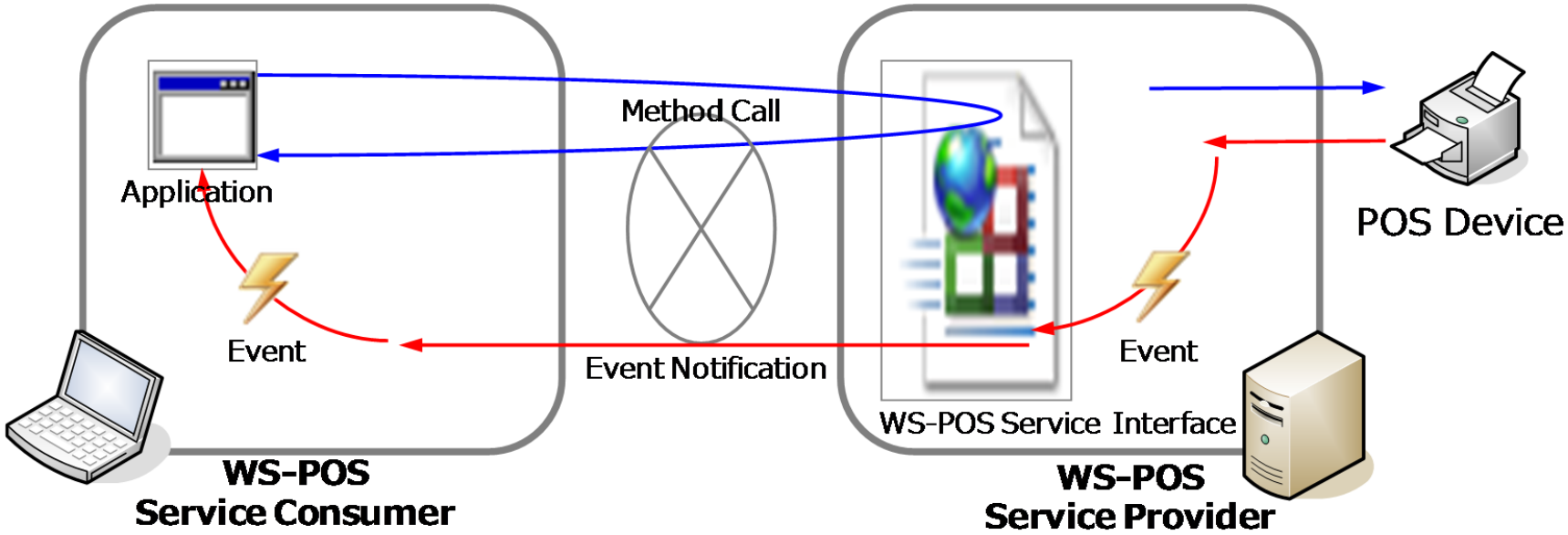
WS-POS サービスプロバイダは、WS-POS サービスコンシューマに対して Unified POS で定義されたデバイスイベントを通知することがある。この通知のために、イベントを受信したい WS-POS サービスコンシューマは、イベント受信のためのエンドポイントを WS-POS サービスプロバイダに通知する必要がある。
WS-POS サービスプロバイダは、WS-POS サービスコンシューマから通知されたイベント受信のためのエンドポイントに対して、Unified POS デバイスイベントを通知する。

WS-POS サービスコンシューマは、イベント受信の必要がなくなった場合、イベント受信エンドポイントを削除するように WS-POS サービスプロバイダに通知する。

手順のシーケンス図を示す。



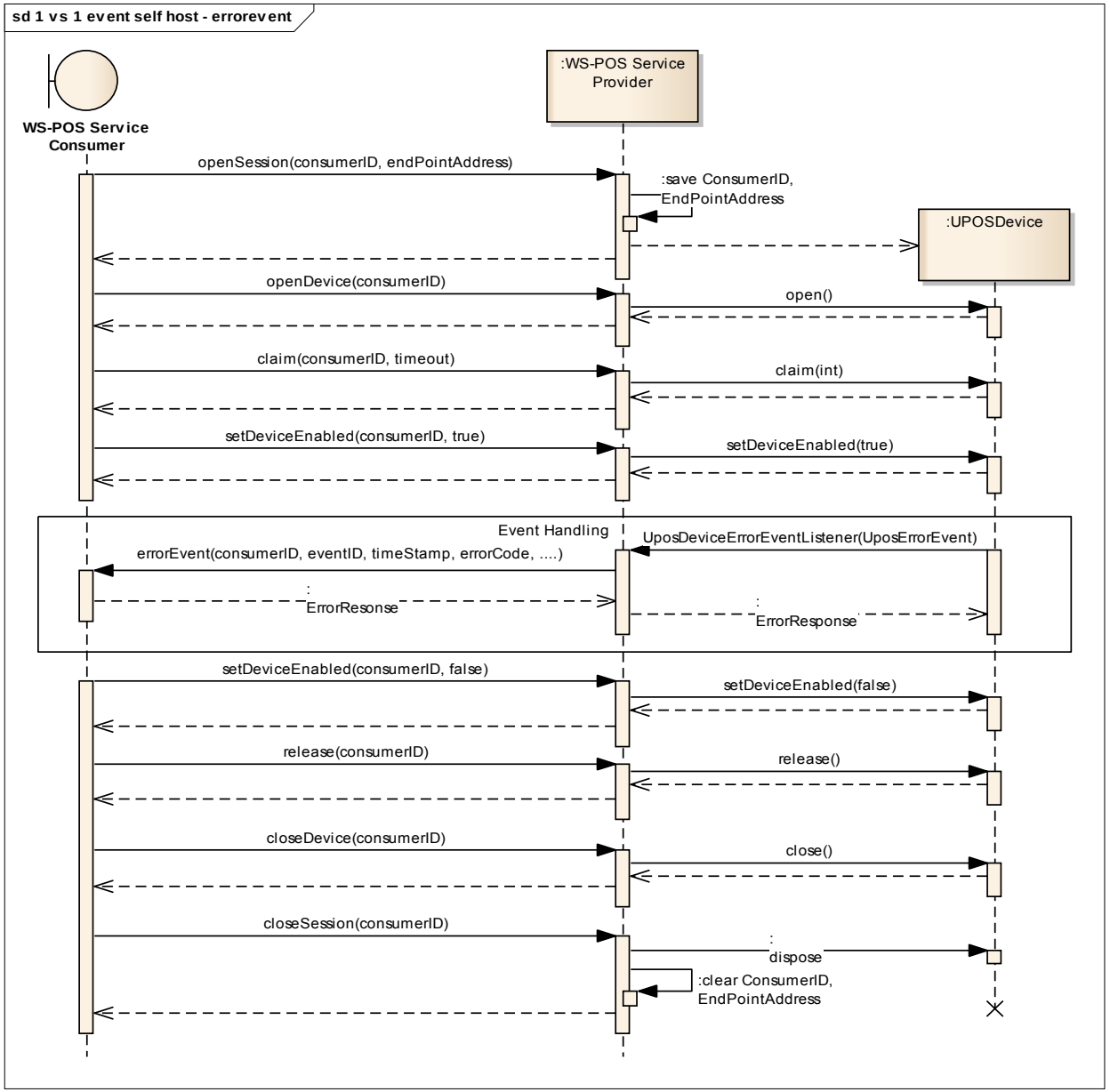
WS-POS 1.2 では新たなメカニズムを追加する。下記に、メソッド呼び出しとイベント通知の概要図を示す。



WS-POS サービスプロバイダがイベント通知を WS-POS サービスコンシューマに対して通知できるようにするための WSDL と XSD スキーマが定義されている。この方式では、WS-POS サービスコンシューマがイベント受信のためのエンドポイントアドレスをサービスコンシューマに登録することを要求する。WS-POS サービスプロバイダが POS デバイスのイベントを受信したら、そのイベント通知を WS-POS サービスコンシューマに送信する。プロキシーコードと設定ファイルが Java もしくは C#開発言語環境に対して利用される。
WS-POS サービスコンシューマはイベント受信のためのエンドポイントアドレスを WS-POS サービスプロバイダに対して、openSession メソッドの引数として渡す。そのエンドポイントアドレスは、WSDL と XSD スキーマで定義されたサービスコンシューマの Web サービスのエントリーポイントでなければならない。単純な実装や特定のプログラミング環境をサポートするために、Java 又は C#で記述されたコントラクトが WS-POS1.2 のサポートファイルとして提供される。イベント受信の為のコントラクトは、”デバイスクラス+イベント”の命名ルールを使う。イベントを扱うための Web サービスメソッドは 2.4.17 章に記述されている。

WS-POS サービスコンシューマが closeSession メソッドを呼び出すときは、WS-POS サービスプロバイダは事前にとうろくされた WS-POS サービスコンシューマのイベント受信のためのエンドポイントアドレスを削除する。

以下のシーケンス図は POSPrinter の ErrorEvent の受け渡しの例である。



2.4.11 ポーリングを用いた WS-POS イベントハンドリング *Version 1.2 で追加*

WS-POS 1.2 では、ポーリング方式がアプリケーションと POS 周辺機器間の通信方法として追加された。

Web アプリケーションにおいては、ブラウザセキュリティやシステムのファイアウォール等のため、アプリケーションがポートを開くことができず、双方向通信が技術的に不可能な場合がある。このシナリオでは、WS-POS サービスプロバイダからのイベント通知が WS-POS サービスコンシューマに届かない。

WS-POS バージョン 1.2 では、イベントハンドリングに“ポーリング”を導入する。ポーリングは、WS-POS サービスコンシューマが WS-POS サービスプロバイダに対して、UnifiedPOS で定義されたイベントを取得する。 ポーリングシーケンスは、WS-POS サービスプロバイダからのイベントが送信されているかどうかを確認するために利用される。

このポーリング技術を使ってタイムリーなイベント処理を実現するため、ポーリング間隔（ポーリングの頻度）は調整可能である。システム構成の最適な状態になるように、ポーリング間隔を調整しなければならない。

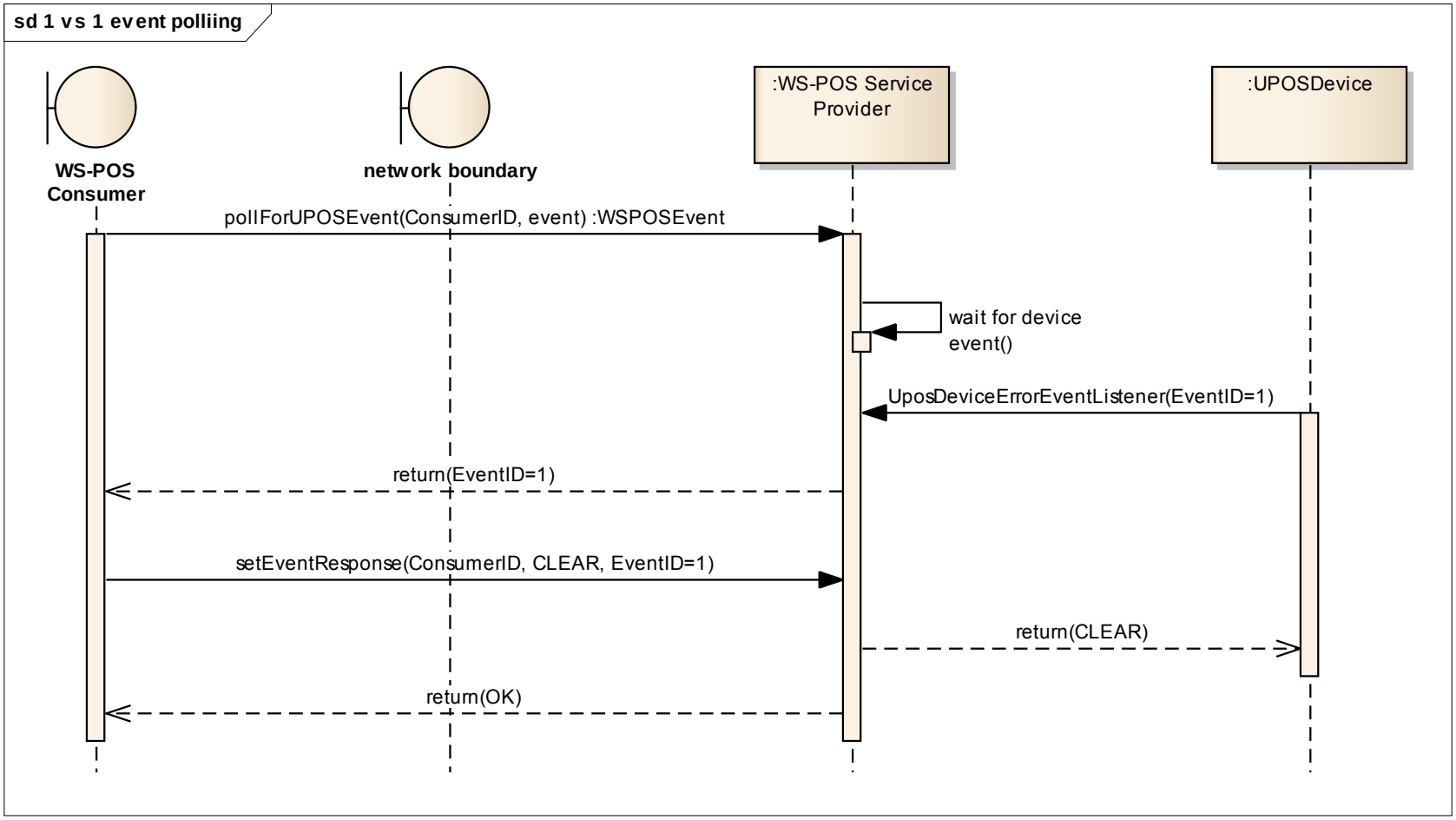
WS-POS サービスコンシューマはイベントをポーリングによって取得したら、ポーリングではない場合と同じようにイベントを処理する。

WS-POS サービスコンシューマがイベントを取得した場合、WS-POS サービスプロバイダに対してイベント応答を作成し送信する。このイベント応答は UnifiedPOS で定義された **ErrorEvent** と **StatusUpdateEvent** へのアクションを含む可能性がある。

WS-POS サービスプロバイダは多くのイベントをキューイングして保持する。WS-POS サービスコンシューマはイベントの処理を速やかに行う必要がある。WS-POS サービスコンシューマからイベント応答を受け取った WS-POS サービスプロバイダは、直ちに別のイベントを WS-POS サービスコンシューマに通知するかもしれない。

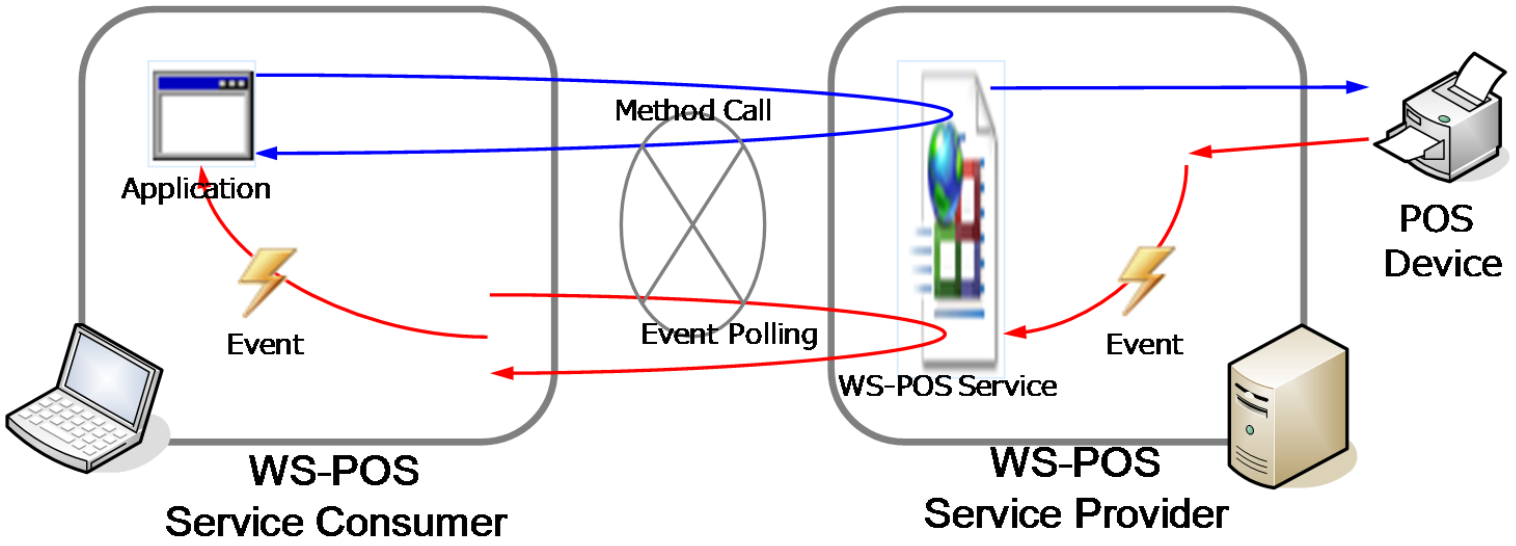
WS-POS サービスコンシューマはイベント取得が不要になった際にはポーリングを中止できる。

ErrorEvent を例に、典型的なシーケンスを図示する。



WS-POS サービスコンシューマは、**ErrorEvent** を受信して、その後、エラーをクリアするよう WS-POS サービスプロバイダに通知している。

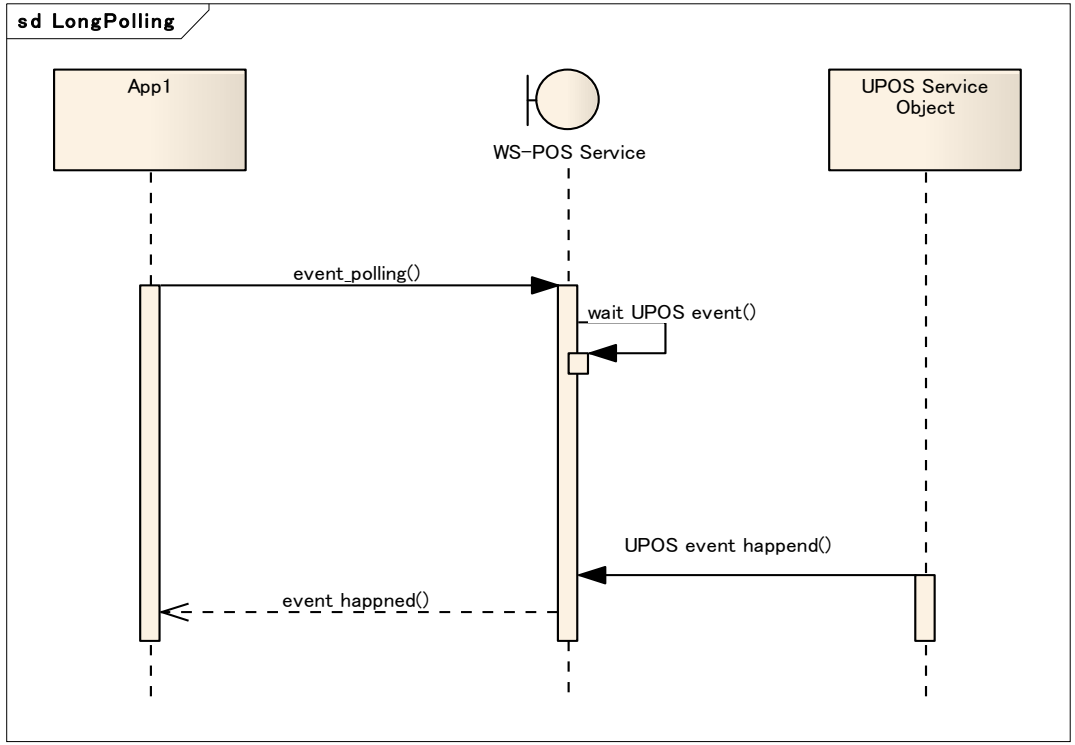
また、下記に、メソッド呼び出しとイベント通知の概要図を示す。



この、イベント受信のための WS-POS サービスコンシューマによるイベントポーリングおよび WS-POS プロバイダによる受取は、プログラミング環境の支援を受け、プロキシコードや設定ファイルによって実現される。

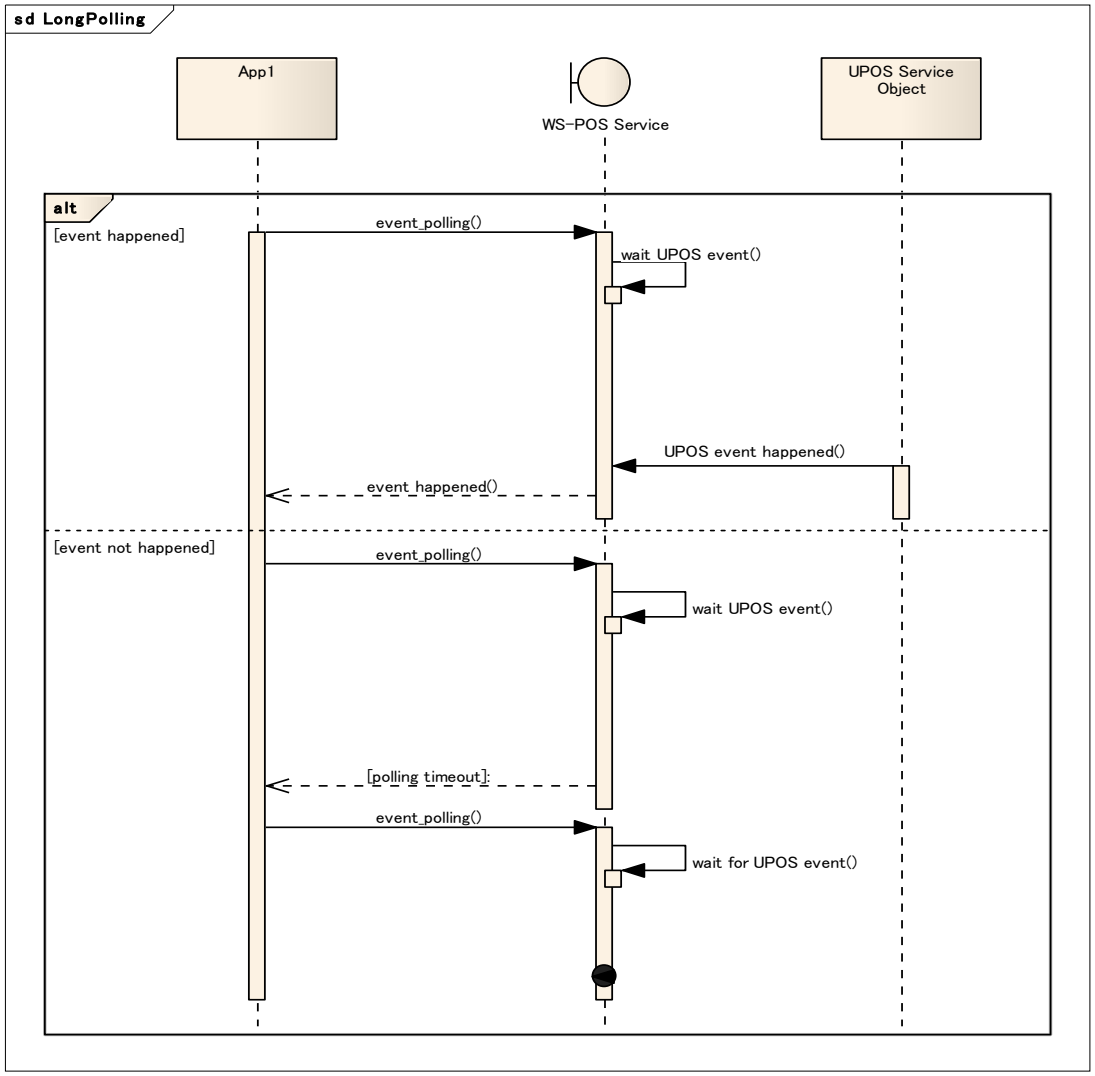
WS-POS サービスコンシューマからサービスプロバイダへのポーリングは、「ロングポーリング」方式*(WS-POS サービスプロバイダはイベントが発生するかタイムアウトが発生するまでリクエストをペンディングする)*である。サービスコンシューマ・サービスプロバイダの処理に起因する、システムの動作に影響を与える負荷を下げる事を考慮して実行する。

一般的なロングポーリングの概念をシーケンス図で以下に説明する。



ロングポーリングでは、ポーリングを行う際、イベントが発生しない場合でも直ちに「イベントが無い」ことを返さず、イベントの発生を待つ。これにより、通常のポーリングに比較して、トラフィック負荷、処理負荷の低減が実現できる。通常、ポーリングは“イベントあり”もしくは“イベントなし”の応答が、サービスプロバイダからサービスコンシューマに対して、すぐに得られることを期待する。

イベントが長時間発生しない場合、ポーリングの待ちがトランスポート層のタイムアウトに達する。WS-POS サービスコンシューマは、ポーリングがタイムアウトしたら、直ちにポーリングを再開する。その動作を下図にて説明する。



2.4.12 WS-POS サービスのネットワーク接続管理の考慮 *Version 1.2 で追加*

WS-POS サービスコンシューマと WS-POS サービスプロバイダ間の通信が断絶してしまう場合が起こりえる。この問題は、WS-POS サービスコンシューマ、WS-POS サービスプロバイダ、またはネットワーク切断のいずれが原因にもなりえる。そのようなエラーの検出と対応、また通信回復の検知と対応が必要となる。このためには、WS-POS サービスコンシューマと WS-POS サービスプロバイダが相互に状況を検知できなければならない。WS-POS バージョン 1.2 では、これらの通信の問題を検知する「**WS-POS Keep Alive**」メカニズムを追加した。

2.4.12.1 WS-POS サービスプロバイダによる WS-POS サービスコンシューマとの通信切断検知

最初に、WS-POS サービスコンシューマは openSession メソッド呼び出しを用いて WS-POS サービスプロバイダとの通信をリクエストする。通常は WS-POS サービスプロバイダから通信成功の応答が返される。

WS-POS サービスプロバイダは WS-POS サービスコンシューマから **open** 後の有効なリクエストを一定期時間ごとに受信することで WS-POS サービスプロバイダの死活監視を行う。

WS-POS コンシューマは、このリクエストを送信する間隔をプログラムで変更可能な「Keep Alive 間隔」として定義する。

WS-POS サービスプロバイダは「プロバイダセッションタイムアウト」として定義された間隔より短い時間で WS-POS コンシューマからのリクエストが送信されることを予期する。

WS-POS サービスプロバイダは WS-POS サービスコンシューマのリクエストをプロバイダセッションタイムアウト間隔より早く受信した場合、セッションタイムアウトのタイマをリセットし、リクエストを処理して返信し、そして WS-POS コンシューマからの次のリクエストを待つ。WS-POS サービスコンシューマもまた Keep Alive 間隔用タイマをリセットする。プロバイダセッションタイムアウトの間隔は、Keep Alive 間隔より長くなければならないことに注意すること。

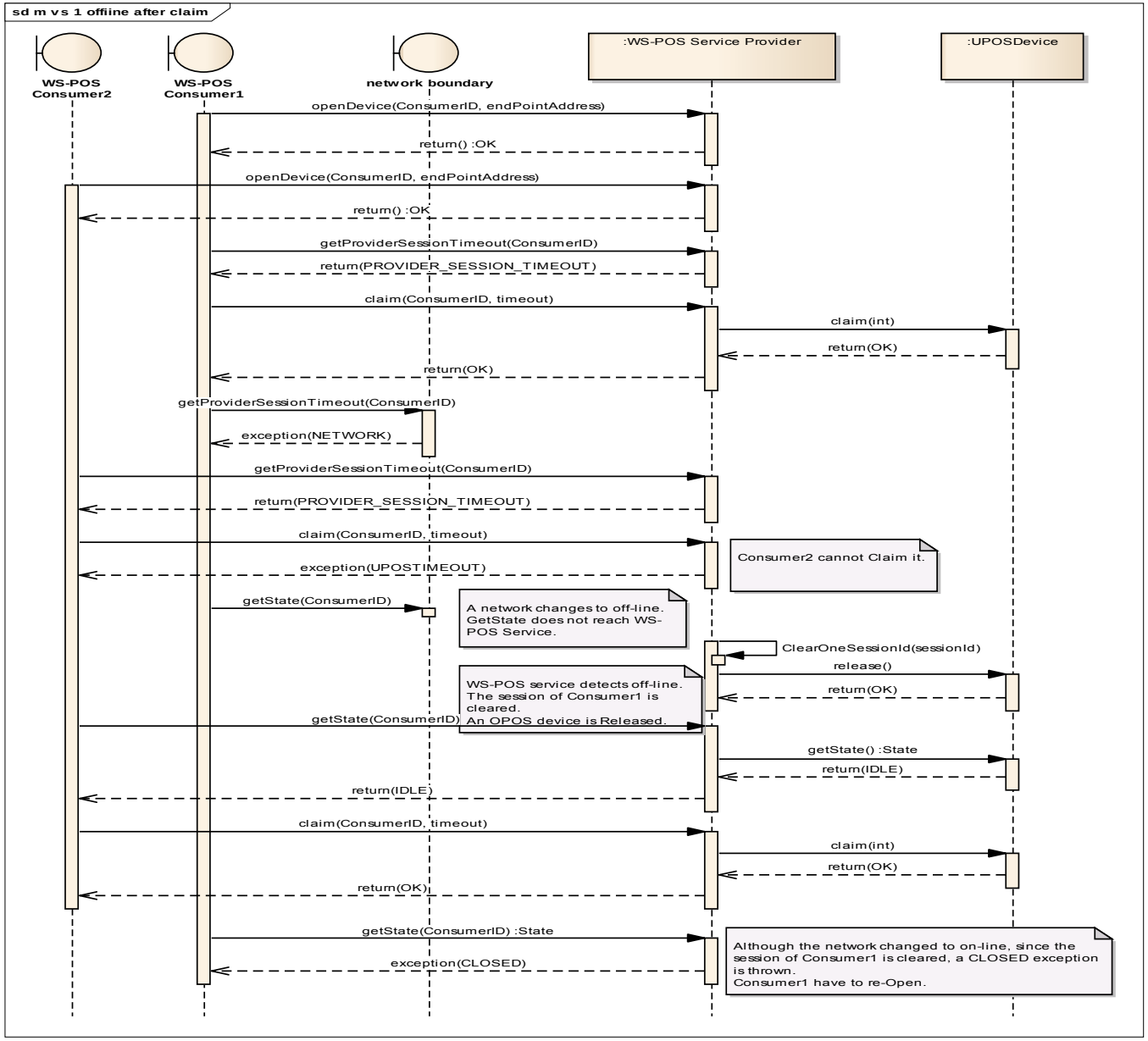
WS-POS サービスプロバイダがプロバイダセッションタイムアウト間隔を超えてリクエストを受信しなかった場合、WS-POS サービスプロバイダは接続が失われたと判断し、WS-POS コンシューマから **closeSession** メソッドを受信した場合と同様の動作を行う。

プロバイダセッションタイムアウトはアプリケーションインストール時やシステム設定時に外部設定できるよう実装する。例として、.NET Framework/WCF 実装では.config ファイルで設定可能に、JAX-WS による実装では WEB-INF\classes 下に properties ファイルを配置し設定可能にする。

WS-POS サービスコンシューマは **ProviderSessionTimeout** プロパティを参照することでプロバイダセッションタイムアウトを取得できる。WS-POS サービスコンシューマは **KeepAliveInterval** を適切な値に設定するためにプロバイダセッションタイムアウト値を使用する。

通信が失われた時、複数の WS-POS サービスコンシューマが一つの WS-POS サービスプロバイダと通信している場合がある。WS-POS サービスプロバイダは通信が失われた場合には各々の WS-POS サービスコンシューマについてエラー処理とリカバリ処理を行う。

下図は、通信が失われた場合に WS-POS サービスプロバイダがどのようにエラー処理とリカバリ処理を行うかを示す。



WS-POS サービスプロバイダがネットワークの切断を検出した場合、WS-POS サービスプロバイダは WS-POS サービスコンシューマから **closeSession** メソッド呼び出しを受信した場合と同様の動作をする。ネットワークが復元した場合、WS-POS サービスコンシューマは WS-POS サービスプロバイダに対して o**penSession** メソッドの呼び出しから再開しなければならない。

2.4.12.2 WS-POS サービスコンシューマによる WS-POS サービスプロバイダとの通信切断検知

最初に、WS-POS サービスコンシューマは openSession メソッド呼び出しを用いて WS-POS サービスプロバイダとの通信をリクエストする。通常は WS-POS サービスプロバイダから通信成功の応答が返される。WS-POS サービスコンシューマは WS-POS Keep Alive メカニズムを用いて WS-POS サービスプロバイダに自身の生存を通知する。

WS-POS サービスコンシューマも、WS-POS サービスプロバイダとの通信が有効な状態にあるか否かを判別できなければならない。これは以下の手順で行う。

- WS-POS サービスコンシューマは Keep Alive 間隔ごとに WS-POS サービスプロバイダに対しリクエストを送信する。
- ネットワークの切断が発生した場合、WS-POS サービスプロバイダからのイベントは WS-POS サービスコンシューマに受け取られない。その代わりに、WS-POS サービスコンシューマは Keep Alive がタイムアウトになり、自身でエラーイベントを作成する。
- WS-POS サービスコンシューマはこのエラーイベントを WS-POS サービスプロバイダとの通信が切断された意味と判断する。WS-POS サービスコンシューマは **openSession** メソッドが呼び出される前の状態に戻る。
- WS-POS サービスコンシューマは新たに **openSession** メソッドをその WS-POS サービスコンシューマに送信するか、他の同等の機能を有する WS-POS サービスプロバイダに対して送信する。

2.4.13 WS-POS サービスにおけるネットワーク接続管理（イベント―双方向通信の場合）

Version 1.2 で追加

双方向通信でのイベント通知において、ネットワーク切断が発生する状況を以下に分類する。

A) WS-POS サービスプロバイダから、WS-POS サービスコンシューマに対してイベント通知が届かない。ネットワーク切断は、WS-POS プロバイダセッションタイムアウト値より前に復旧した。(2.4.13.1 プロバイダ→コンシューマ切断かつセッションタイムアウトせず)

B) WS-POS サービスプロバイダから、WS-POS サービスコンシューマに対してイベント通知が届いたが、WS-POS コンシューマからの応答が WS-POS サービスプロバイダに届かない。ネットワーク切断は、WS-POS プロバイダセッションタイムアウト値より前に復旧した。(2.4.13.2 コンシューマ→プロバイダ切断かつセッションタイムアウトせず)

C) WS-POS サービスプロバイダから、WS-POS サービスコンシューマに対してイベント通知が届かない。ネットワーク切断は、WS-POS プロバイダセッションタイムアウト値を過ぎても復旧しなかった。(2.4.13.3 プロバイダ→コンシューマ切断かつセッションタイムアウト発生)

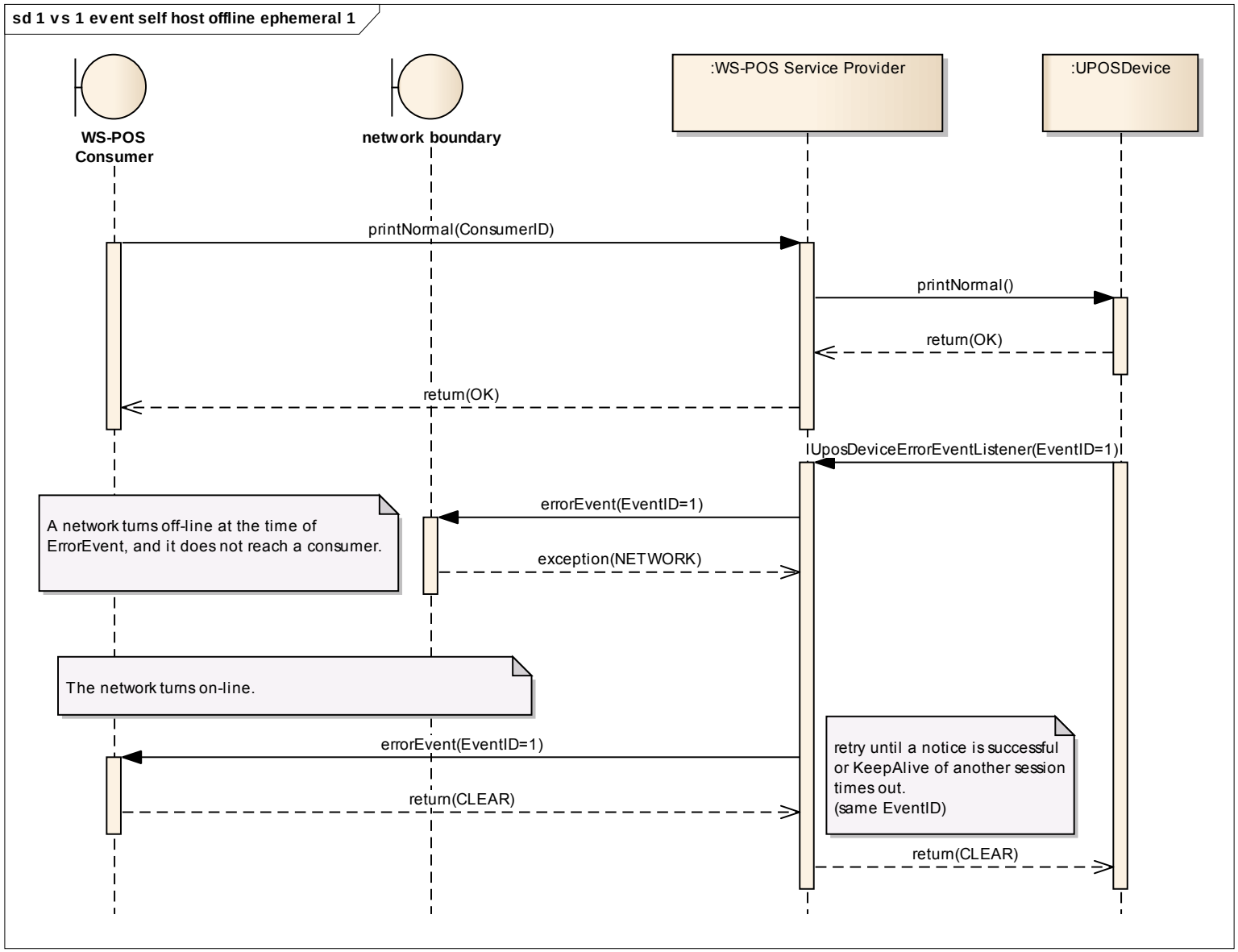
D) WS-POS サービスプロバイダから、WS-POS サービスコンシューマに対してイベント通知が届いたが、WS-POS コンシューマからの応答が WS-POS サービスプロバイダに届かない。ネットワーク切断は、WS-POS プロバイダセッションタイムアウト値を過ぎても復旧しなかった。(2.4.13.4 コンシューマ→プロバイダ切断かつセッションタイムアウト発生)

2.4.13.1 プロバイダ→コンシューマ切断かつセッションタイムアウトせず

シナリオ: WS-POS サービスプロバイダから、WS-POS サービスコンシューマに対してイベント通知が届かない。ネットワーク切断は、WS-POS プロバイダセッションタイムアウト値より前に復旧した。

ErrorEvent 発生時にネットワーク切断状況であった場合、WS-POS サービスプロバイダは通知をリトライする。その場合、同じ EventID が指定される。その動作を下図に示す。

セッションタイムアウトよる切断検出の前に、ネットワークが回復していることに注意すること。

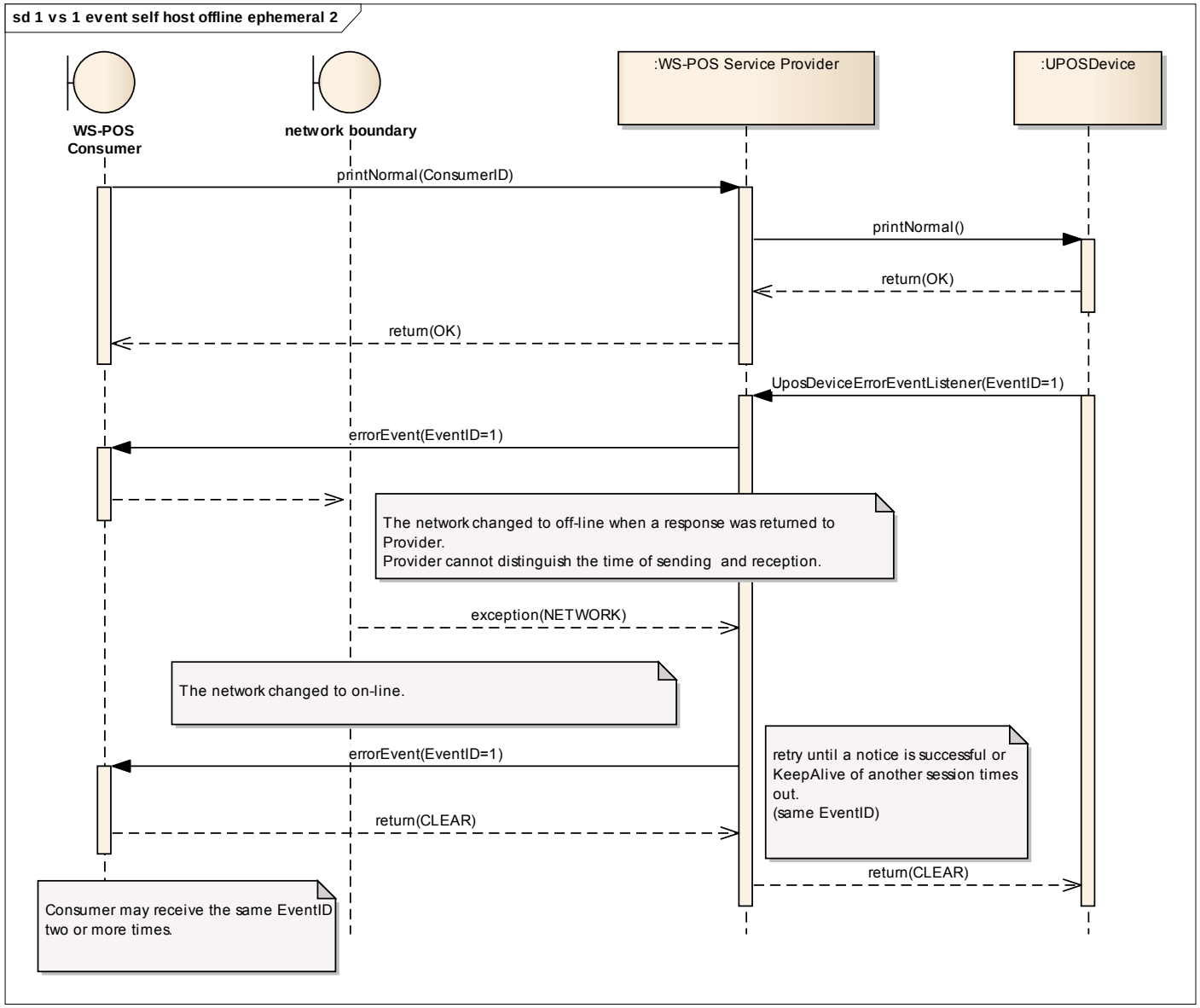


2.4.13.2 コンシューマ→プロバイダ切断かつセッションタイムアウトせず

シナリオ: WS-POS サービスプロバイダから、WS-POS サービスコンシューマに対してイベント通知が届いたが、WS-POS コンシューマからの応答が WS-POS サービスプロバイダに届かない。ネットワーク切断は、WS-POS プロバイダセッションタイムアウト値より前に復旧した。

ErrorEvent 発生時にネットワーク切断状況であった場合、WS-POS サービスプロバイダは通知をリトライする。その場合、同じ EventID が指定される。その動作を下図に示す。

WS-POS サービスプロバイダは、イベントを送信する際にオフラインになったのか、応答を受信する際にオフラインになったのか判断ができないため、リトライを実施する。このため、WS-POS サービスコンシューマは、同じ EventID を持ったイベントが複数回通知されることがある。



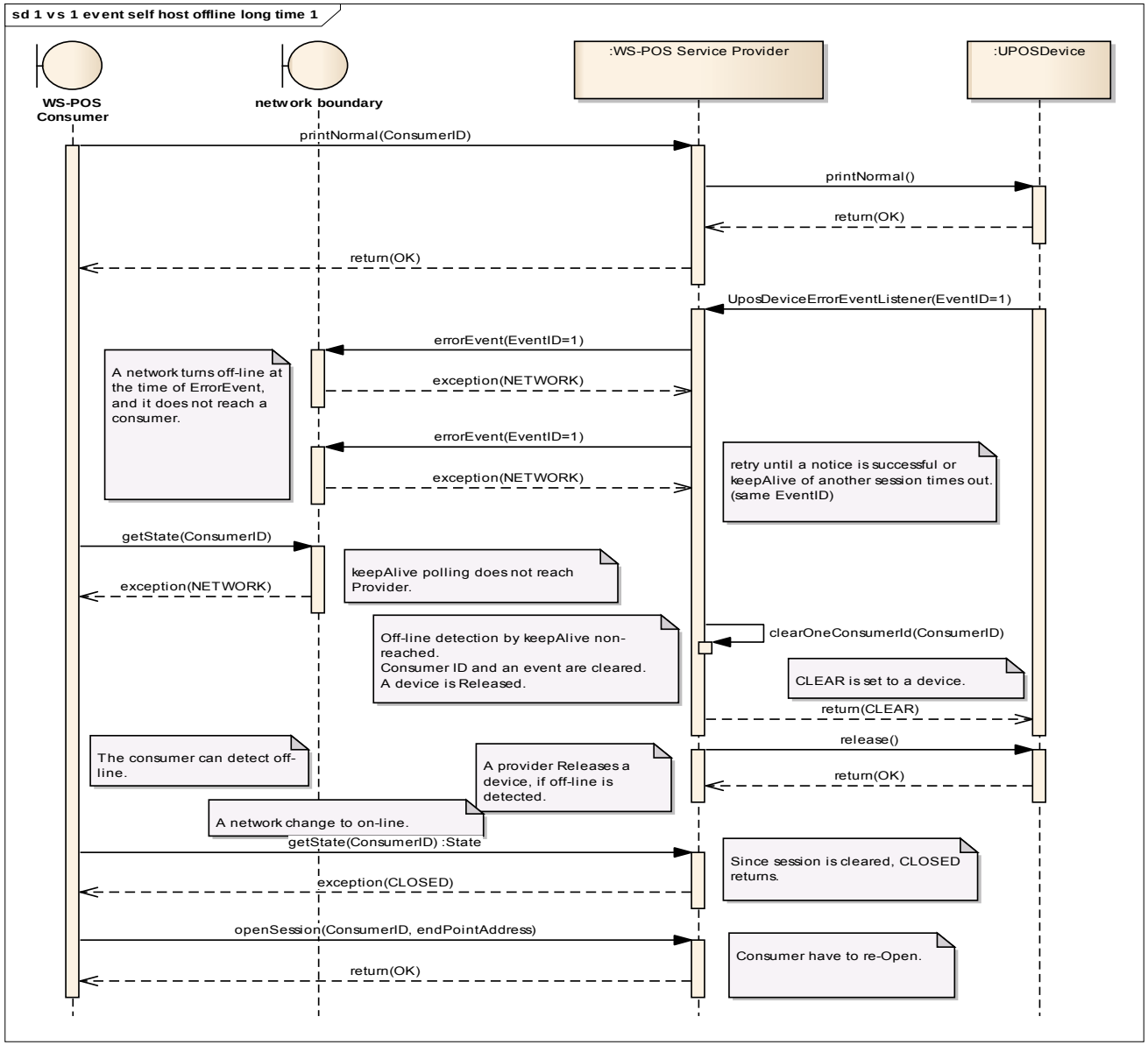
2.4.13.3 プロバイダ→コンシューマ切断かつセッションタイムアウト発生

シナリオ: WS-POS サービスプロバイダから、WS-POS サービスコンシューマに対してイベント通知が届かない。ネットワーク切断は、WS-POS プロバイダセッションタイムアウト値を過ぎても復旧しなかった。

WS-POS プロバイダセッションタイムアウト値を超えてネットワークが回復しなかった場合、WS-POS サービスプロバイダはセッションを破棄し、UnifiedPOS デバイスからのイベント通知に対して返答する。**ErrorEvent** の場合、UnifledPOS デバイスへの返答には **CLEAR** が指定される。

下図にこの動作を示す。このシナリオでは、WS-POS サービスコンシューマは **ErrorEvent** を受信できない。

ネットワークが回復しても、WS-POS サービスプロバイダはセッションを破棄しているため、WS-POS サービスコンシューマは再度 **openSession** から実行しなおす必要がある。



2.4.13.4 コンシューマ→プロバイダ切断かつセッションタイムアウト発生

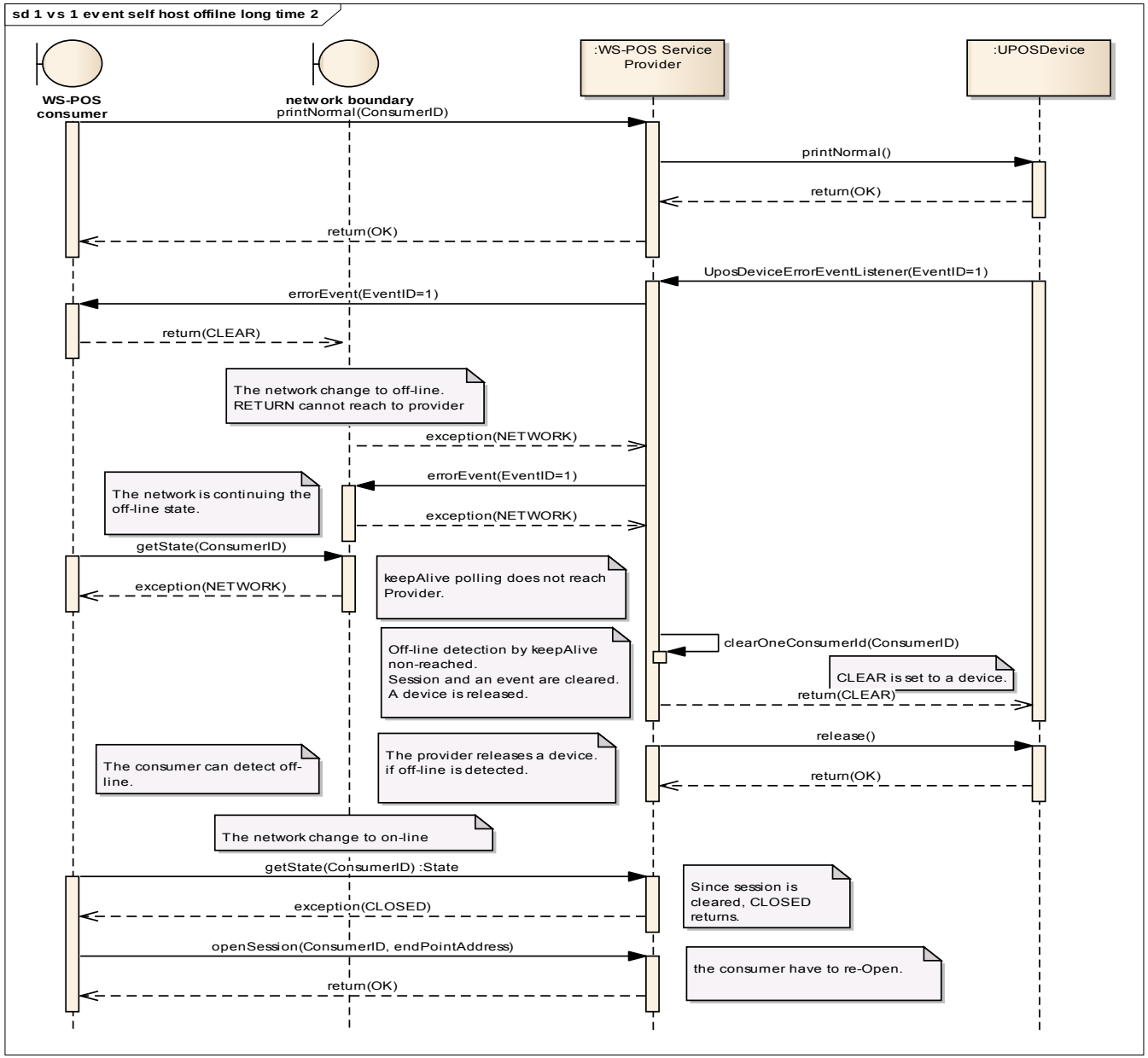
シナリオ: WS-POS サービスプロバイダから、WS-POS サービスコンシューマに対してイベント通知が届いたが、WS-POS コンシューマからの応答が WS-POS サービスプロバイダに届かない。ネットワーク切断は、WS-POS プロバイダセッションタイムアウト値を過ぎても復旧しなかった。

WS-POS プロバイダセッションタイムアウト値を超えてネットワークが回復しなかった場合、WS-POS サービスプロバイダはセッションを破棄し、UnifiedPOS デバイスからのイベント通知に対して返答する。**ErrorEvent** の場合、UnifiedPOS デバイスへの返答には **CLEAR** が指定される。

このシナリオでは、WS-POS サービスコンシューマは **ErrorEvent** を受信できるが、応答を WS-POS サービスプロバイダに届けることができず、WS-POS サービスプロバイダは UnifiledPOS デバイスへの応答に **CLEAR** を指定する。

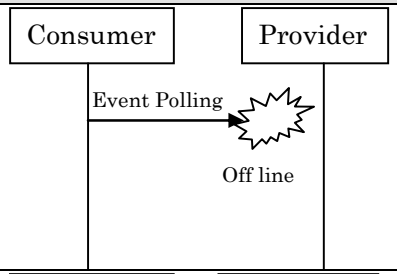
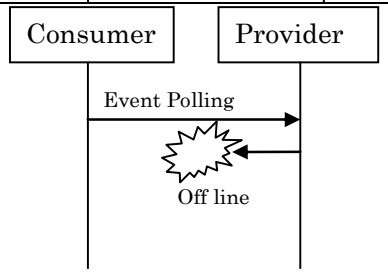
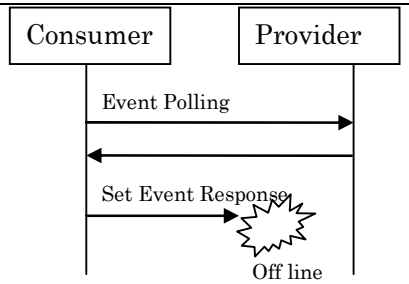
ネットワークが回復しても、WS-POS サービスプロバイダは closeSession メソッドが実行されたとしてセッションを破棄しているため、WS-POS サービスコンシューマは再度 openSession から実行しなおす必要がある。

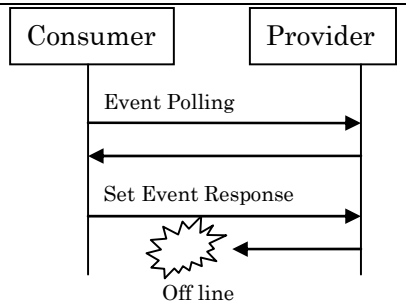
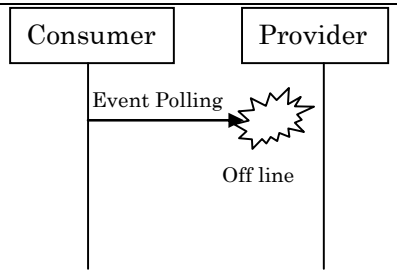
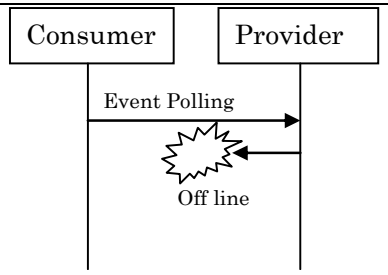
下図にこのシナリオを示す。

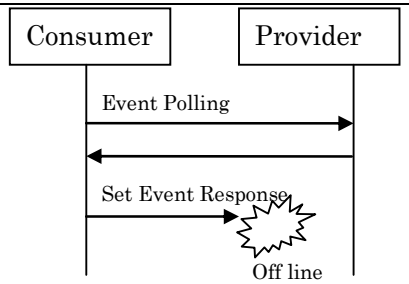
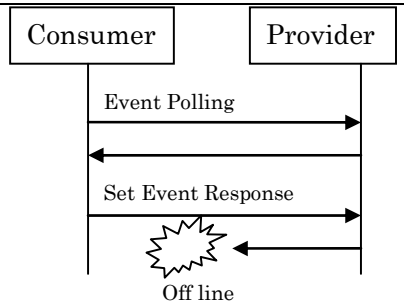


2.4.14 WS-POS サービスにおけるネットワーク接続管理（イベント―ポーリングの場合） *Version 1.2 で追加*

ネットワーク切断管理は WS-POS サービスコンシューマと WS-POS サービスプロバイダの間で整合性を確保する必要がある。ポーリングでのイベント通知において、ネットワーク切断が発生する状況を以下の表に分類し、動作を記述する。

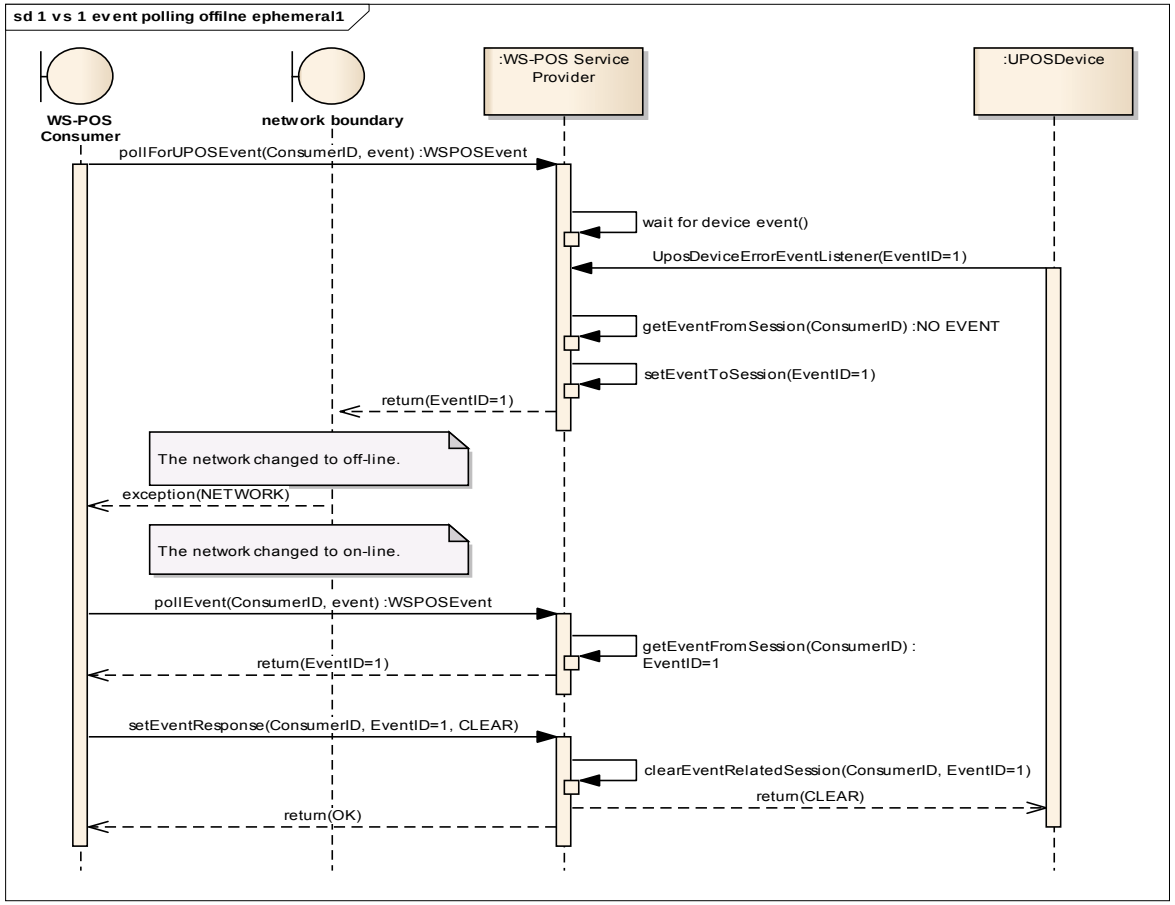
	シナリオ	シーケンス図	動作
A)	WS-POS サービスコンシューマから、WS-POS サービスプロバイダに対して**イベントポーリングが届かない**。 ネットワーク切断は、WS-POS プロバイダセッションタイムアウト値より前に復旧した。		WS-POS サービスコンシューマは、イベントポーリングを再び実行する。 ネットワークが回復すれば、通常のシーケンスと同じである。
B)	WS-POS サービスコンシューマから、WS-POS サービスプロバイダに対してイベントポーリングが届いたが、WS-POS サービスプロバイダからの**イベントポーリング戻り値が WS-POS サービスコンシューマに届かない**。 ネットワーク切断は、WS-POS プロバイダセッションタイムアウト値より前に復旧した。		WS-POS サービスコンシューマは、イベントポーリングを再び実行する。 ネットワークが回復すれば、通常のシーケンスと同じである。 後述「2.4.14.1 イベントポーリング戻り値が届かず、かつセッションタイムアウトせず」を参照すること。
C)	WS-POS サービスコンシューマから、WS-POS サービスプロバイダに対してイベントポーリングは正常に終了したが、**イベント応答設定が届かない**。 ネットワーク切断は、WS-POS プロバイダセッションタイムアウト値より前に復旧した。		WS-POS サービスコンシューマは、イベント応答設定(**setEventResponse**)を再び実行する。 ネットワークが回復すれば、通常のシーケンスと同じである。

D)	T WS-POS サービスコンシューマから、WS-POS サービスプロバイダに対してイベントポーリングは正常に終了し、イベント応答設定も届いたが、**イベント応答設定に対する戻りが WS-POS サービスコンシューマに届かない**。 ネットワーク切断は、WS-POS プロバイダセッションタイムアウト値より前に復旧した。		WS-POS サービスコンシューマは、イベント応答設定(**setEventResponse**)を再び実行する。ネットワークが回復した場合、イベント応答設定の戻りとして ILLEGAL が返される。それ以降は、通常のシーケンスと同じである。 後述「2.4.14.2 イベント応答設定の戻り値が届かず、かつセッションタイムアウトせず」を参照すること。
E)	WS-POS サービスコンシューマから、WS-POS サービスプロバイダに対して**イベントポーリングが届かない**。 ネットワーク切断は、**WS-POS プロバイダセッションタイムアウト値を過ぎても復旧しなかった**。		WS-POS サービスコンシューマは、イベントポーリングを再び実行するが、プロバイダセッションタイムアウト値を過ぎても回復しない。 WS-POS サービスコンシューマは、WS-POS サービスプロバイダとのセッションが失われたと判断する。 WS-POS サービスプロバイダは、セッションを破棄し、UnifiedPOS デバイスを Release する。 ネットワークが回復した場合、WS-POS サービスコンシューマは **openSession** を再度実行する。以降は通常のシーケンスと同じである。
F)	WS-POS サービスコンシューマから、WS-POS サービスプロバイダに対してイベントポーリングが届いたが、WS-POS サービスプロバイダからの**イベントポーリング戻り値が WS-POS サービスコンシューマに届かない**。 ネットワーク切断は、**WS-POS プロバイダセッションタイムアウト値を過ぎても復旧しなかった**。		WS-POS サービスコンシューマは、イベントポーリングを再び実行するが、プロバイダセッションタイムアウト値を過ぎても回復しない。 WS-POS サービスコンシューマは、WS-POS サービスプロバイダとのセッションが失われたと判断する。 WS-POS サービスプロバイダは、セッションをクリアし、UnifiedPOS デバイスを Release する。 ネットワークが回復した場合、WS-POS サービスコンシューマは openSession を再度実行する。以降は通常のシーケンスと同じである。

			後述 「2.4.14.3 イベントポーリングが届かず、かつセッションタイムアウト発生」を参照すること。
G)	WS-POS サービスコンシューマから、WS-POS サービスプロバイダに対してイベントポーリングは正常に終了したが、**イベント応答設定が届かない**。 ネットワーク切断は、**WS-POS プロバイダセッションタイムアウト値を過ぎても復旧しなかった**。		WS-POS サービスコンシューマは、イベント応答設定(**setEventResponse**)を再び実行するが、プロバイダセッションタイムアウト値を過ぎても回復しない。 WS-POS サービスコンシューマは、WS-POS サービスプロバイダとのセッションが失われたと判断する。 WS-POS サービスプロバイダは、セッションを破棄し、UnifiedPOS デバイスを Release する。 ネットワークが回復した場合、WS-POS サービスコンシューマは **openSession** を再度実行する。以降は通常のシーケンスと同じである。
H)	WS-POS サービスコンシューマから、WS-POS サービスプロバイダに対してイベントポーリングは正常に終了し、イベント応答設定も届いたが、**イベント応答設定に対する戻りが WS-POS サービスコンシューマに届かない**。 ネットワーク切断は、**WS-POS プロバイダセッションタイムアウト値を過ぎても復旧しなかった**。		WS-POS サービスコンシューマは、イベント応答設定(**setEventResponse**)を再び実行するが、プロバイダセッションタイムアウト値を過ぎても回復しない。 WS-POS サービスコンシューマは、WS-POS サービスプロバイダとのセッションが失われたと判断する。 WS-POS サービスプロバイダは、セッションをクリアし、UnifiedPOS デバイスを Release する。 ネットワークが回復した場合、WS-POS サービスコンシューマは **openSession** を再度実行する。以降は通常のシーケンスと同じである。

2.4.14.1 イベントポーリング戻り値が届かず、かつセッションタイムアウトせず

このシナリオでは、WS-POS サービスコンシューマは、WS-POS サービスプロバイダに対するイベントポーリングが成功しない場合、再度イベントポーリングを行う。



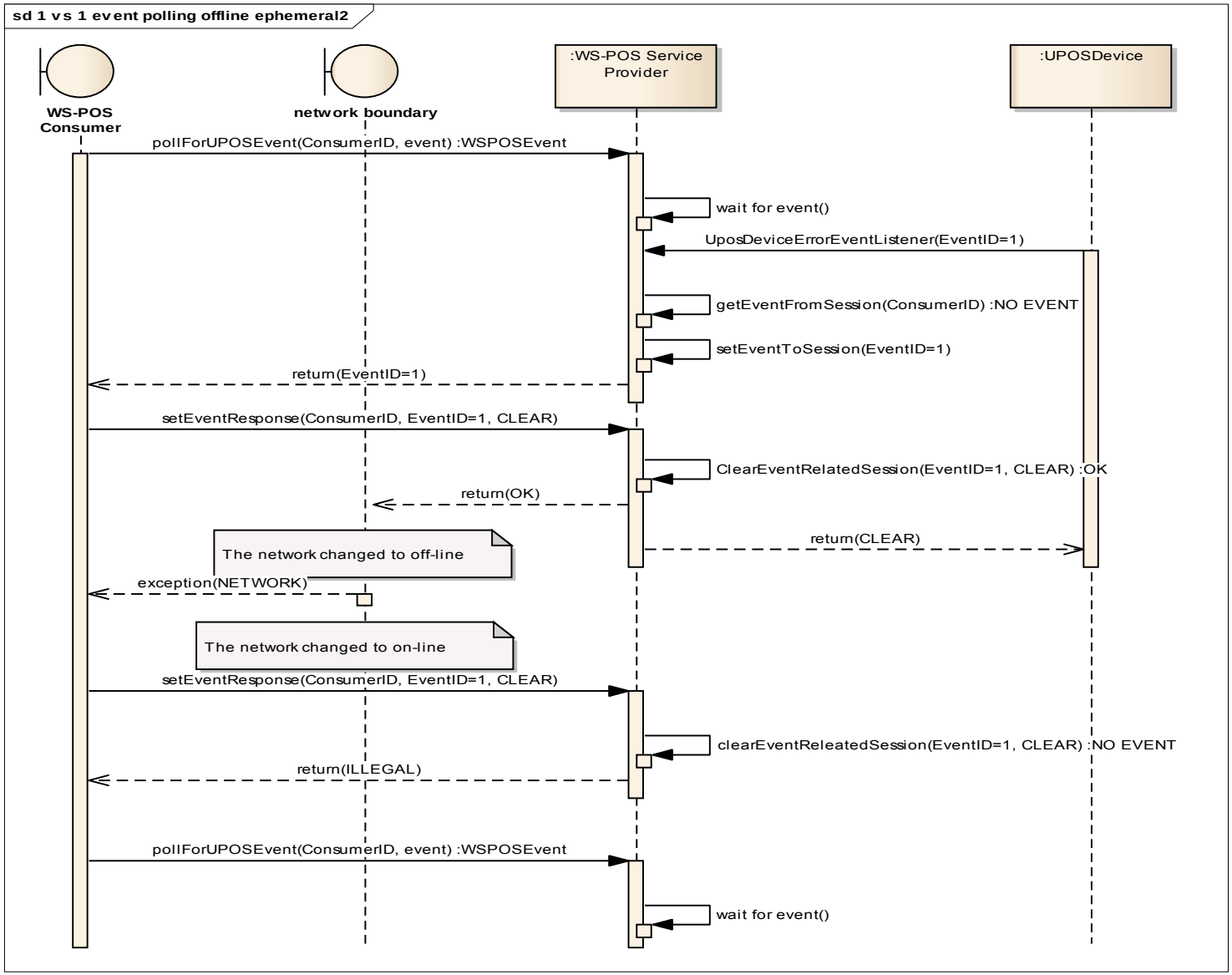
WS-POS プロバイダは、イベント応答(**setEventResponse**)が成功するまで、POS デバイスからのイベントを保持するため、WS-POS コンシューマはネットワークが回復した後の再ポーリングでイベントを取得できる。

2.4.14.2 イベント応答設定の戻り値が届かず、かつセッションタイムアウトせず

このシナリオは、イベント応答設定(**setEventResponse**)の戻り時にネットワークが切断される。WS-POS サービスコンシューマは、WS-POS サービスプロバイダに対してイベント応答設定をリトライする。このとき、ネットワーク回復後の最初のイベント応答に対して、イベントが存在しないことを示す **ILLEGAL** が返される。

WS-POS サービスコンシューマではネットワーク例外が発生するが、その前に WS-POS サービスプロバイダはイベント応答に対して処理を行い UnifiedPOS デバイスのイベントハンドラを終了させている。そのため、ネットワーク回復後のイベント応答では、応答すべきイベントが存在せず、ILLEGAL を返している。

このシナリオを下図で示す。

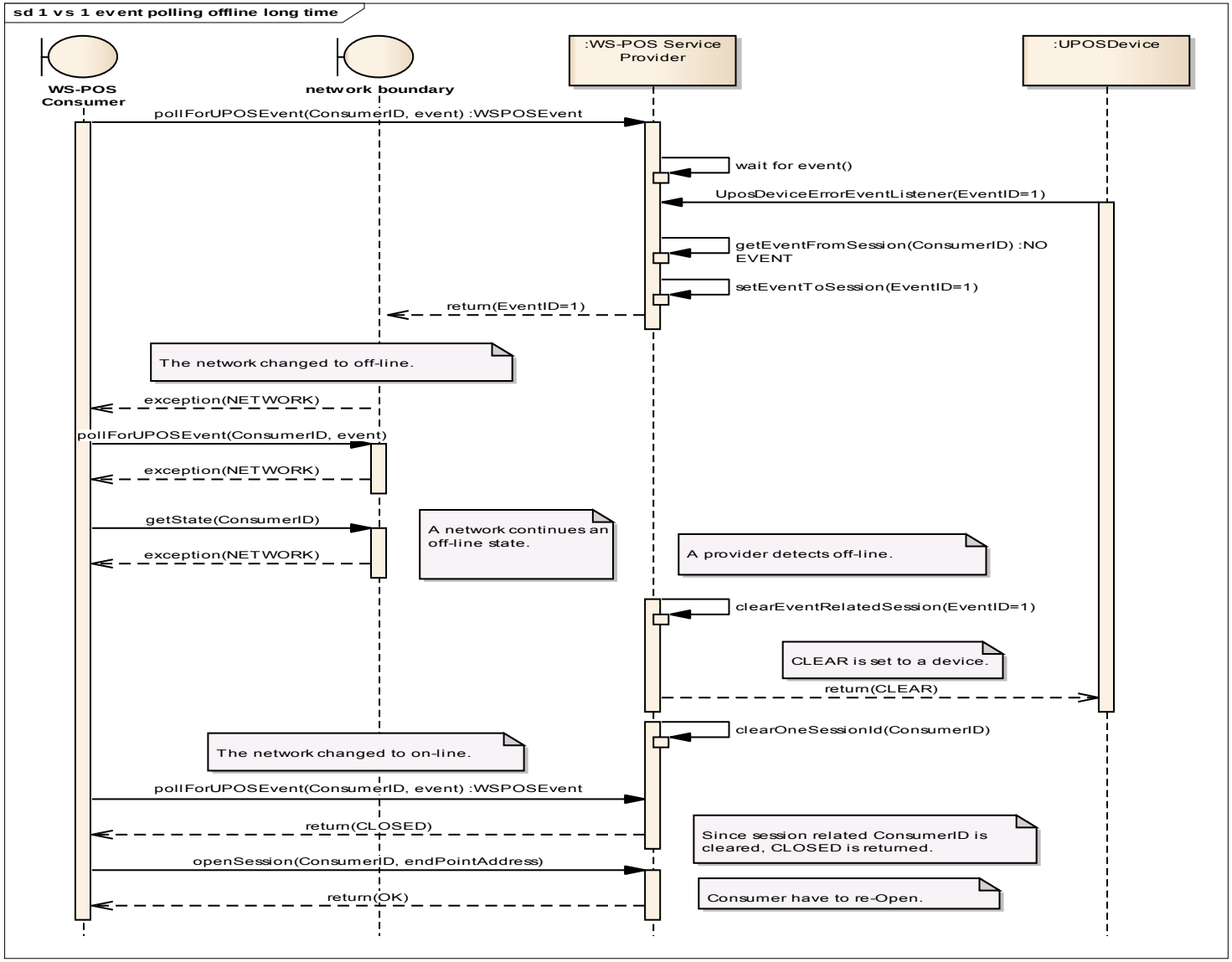


2.4.14.3 イベントポーリングが届かず、かつセッションタイムアウト発生

このシナリオは、プロバイダセッションタイムアウト値を超えて、ネットワークの切断が回復しない場合である。WS-POS サービスプロバイダはセッションと共にイベントを破棄する。**ErrorEvent** の場合、UnifiedPOS デバイスへの返答には CLEAR が指定される。

この場合、WS-POS サービスプロバイダはセッションをクリアしているため、ネットワークが回復しても、WS-POS サービスコンシューマは再度 OpenSession から実行する必要がある。

このシナリオを下図で示す。



2.4.15 WS-POS メソッドリファレンス(UnifiedPOS UML 形式) *Version 1.2 で追加*

以下にセッションレベルのメソッドを記載する。

2.4.15.1 generateConsumerID Method

Syntax **generateConsumerID(out consumerID: *string*):**
void { raises-exception }

パラメータ	説明
consumerID	WS-POS サービスプロバイダ内でコンシューマを一意に識別するための ID

Remarks ユニークであることが“保障”された *consumerID* を生成。GUID や UUID のアルゴリズムを使うことを提案

Errors 例外が発生する可能性がある。

値	意味
E_FAILURE	*consumerID* の生成に失敗した

2.4.15.2 openSession Method

Syntax **openSession(consumerID: string, consumerEventEndPoint: string) : void {raises-exception}**

パラメータ	説明
consumerID	WS-POS サービスプロバイダ内でコンシューマを一意に識別するための ID
consumerEventEndPoint	WS-POS サービスプロバイダが、WS-POS サービスコンシューマにイベントを通知するためのサービスエンドポイント。*consumerEventEndPoint* が null の場合は、WS-POS サービスプロバイダはイベント通知を行わず、WS-POS サービスコンシューマが **pollForUPOSEvent** メソッドを呼ぶことで UnifiedPOS イベントを取得する。

Remarks WS-POS サービスコンシューマと WS-POS サービスプロバイダのセッションを確立する。

Errors 例外が発生する可能性がある。

値	意味
E_ILLEGAL	同一の *consumerID* で確立されているセッションが既に存在する。

2.4.15.3 closeSession Method

Syntax **closeSession(consumerID: *string*) :**
void {raises-exception}

パラメータ	説明
consumerID	WS-POS サービスプロバイダ内でコンシューマを一意に識別するための ID

Remarks WS-POS サービスコンシューマと WS-POS サービスプロバイダのセッションを破棄する。

Errors 例外が発生する可能性がある。

値	意味
E_ILLEGAL	**openSession** メソッドが呼び出されていない。または、*consumerID* が不正である。(指定された *consumerID* で確立されているセッションが無い。)

2.4.15.4 getProviderSessionTimeout Method

Syntax **getProviderSessionTimeout(consumerID: *string*, out timeout : *int32*): void {raises-exception}**

パラメータ	説明
consumerID	WS-POS サービスプロバイダ内でコンシューマを一意に識別するための ID
timeout	WS-POS サービスプロバイダが管理する、WS-POS セッションのタイムアウト値

Remarks WS-POS サービスプロバイダが管理する、WS-POS セッションのタイムアウト値を秒で返す。

Errors 例外が発生する可能性がある。

値	意味
E_ILLEGAL	**openSession** メソッドが呼び出されていない。または、*consumerID* が不正である。(指定された *consumerID* で確立されているセッションが無い。)

2.4.15.5 keepAlive Method

Syntax **keepAlive (consumerID: *string*) :**
void {raises-exception}

パラメータ	説明
consumerID	WS-POS サービスプロバイダ内でコンシューマを一意に識別するための ID

Remarks WS-POS サービスコンシューマと WS-POS サービスプロバイダ間のセッションを保持するために、WS-POS サービスコンシューマが **keepAlive** メソッドを定期的に呼び出す必要がある。WS-POS サービスコンシューマは、**getProviderSessionTimeout** メソッドで取得したタイムアウト値より短い間隔で、**keepAlive** メソッドを呼び出す必要がある。

KeepAlive メソッド呼出しが、**getProviderSessionTimeout** メソッドで取得したタイムアウト値を超えて WS-POS サービスプロバイダに受信されなかった場合は、WS-POS サービスプロバイダは該当セッションを破棄する。その場合、再接続には **openSession** メソッドの呼び出しが必要である。

Errors 例外が発生する可能性がある。

値	意味
E_ILLEGAL	openSession メソッドが呼び出されていない。または、*consumerID* が不正である。(指定された *consumerID* で確立されているセッションが無い。)

2.4.15.6 pollForUPOSEvent Method

Syntax **pollForUPOSEvent (consumerID: *string*, out WSPOSEvent: *string*):**
void {raises-exception}

パラメータ	説明
consumerID	WS-POS サービスプロバイダ内でコンシューマを一意に識別するための ID
WSPOSEvent	UnifiedPOS で定義されたイベント

Remarks WS-POS サービスコンシューマが、WS-POS サービスプロバイダに対して UnifiedPOS イベントの有無をポーリングするために呼び出す。 UnifiedPOS イベントが発生していたらそのイベントを返す。WS-POS サービスコンシューマが、WS-POS サービスプロバイダの **openSession** メソッドを呼びだした際、*consumerEventEndPoint* に null を指定すると **pollForUPOSEvent** メソッドが呼出し可能である。

pollForUPOSEvent メソッドは、**openSession** メソッドを呼び出した後ならば呼び出すことが可能である。しかし、**openDevice** メソッドを呼び出すまでは、イベントが発生しないため、**pollForUPOSEvent** メソッドを呼び出しても必ずタイムアウトする。

Errors 例外が発生する可能性がある。例外による *ErrorCode* として取りうる値：

値	意味
E_ILLEGAL	openSession メソッドが呼び出されていない。または、*consumerID* が不正である。(指定された *consumerID* で確立されているセッションが無い。). **pollForUPOSEvent** メソッドが呼び出されたが、その前の **openSession** メソッドの *consumerEventEndPoint* パラメータに NULL が設定されてない。
E_TIMEOUT	WS-POS サービスプロバイダ内で設定されているイベントポーリングタイムアウト値を超えてイベントの発生を待ったが、イベントは発生していない。

2.4.15.7 setEventResponse Method

Syntax **setEventResponse (consumerID: *string*, WSPOSEventResponse: *string*) : void {raises-exception}**

パラメータ	説明
consumerID	WS-POS サービスプロバイダ内でコンシューマを一意に識別するための ID
WSPOSEventResponse	**pollForUPOSEvent** メソッドで取得したイベントに対する応答

Remarks WS-POS サービスコンシューマが、**pollForUPOSEvent** メソッドでイベントを取得した場合は、イベント処理の完了を WS-POS サービスプロバイダに通知するために、WS-POS セッションのタイムアウトまでに、本メソッドを呼び出す必要がある。「2.4.16 WSPOSEventResponse について」を参照すること。.

本メソッドを呼び出さない場合は、WS-POS サービスプロバイダは WS-POS セッションのタイムアウトまでイベント処理を停止する。

Errors 例外が発生する可能性がある。

値	意味
E_ILLEGAL	openSession メソッドが呼び出されていない。または、*consumerID* が不正である。(指定された *consumerID* で確立されているセッションが無い。)

2.4.15.8 getWSPOSVersion Method

Syntax **getWSPOSVersion (out WSPOSVersion: *string*) :**
void {raises-exception}

パラメータ	説明
WSPOSVersion	WS-POS サービスプロバイダがサポートしているバージョン.

Remarks WS-POS サービスプロバイダが実装している WS-POS のバージョンを返す。

下記に WS-POS 1.2 デバイスレベルのメソッドを記載する。

2.4.15.9 openDevice Method

Syntax **openDevice(consumerID: *string*) :**
void {raises-exception}

パラメータ	説明
consumerID	WS-POS サービスプロバイダ内でコンシューマを一意に識別するための ID

Remarks WS-POS サービスコンシューマが、WS-POS サービスプロバイダに対して UnifiedPOS デバイスをオープンするために呼び出す。WS-POS サービスプロバイダは、**openDevice** メソッドを受けて、デバイスクラスに対応する UnifiedPOS ドライバのインスタンスを生成し、そのドライバに対してオープンする。

Errors 例外が発生する可能性がある。例外による *ErrorCode* として取りうる値：

値	意味
E_ILLEGAL	指定された *consumerID* で確立されているセッションが無い。または、デバイスがすでに open されている。 既に UnifiedPOS コントロールがオープンされている。
E_NOEXIST	サービスプロバイダが使用する論理名が、サービスプロバイダが実行されているノード内で UnifiedPOS ドライバとして登録されていない。
E_NOSERVICE	Unified POS サービスとの接続を確立できない。

2.4.15.10 closeDevice Method

Syntax **closeDevice(consumerID: *string*) :**
void {raises-exception}

パラメータ	説明
consumerID	WS-POS サービスプロバイダ内でコンシューマを一意に識別するための ID

Remarks WS-POS サービスコンシューマが、WS-POS サービスプロバイダに対して UnifiedPOS デバイスをクローズするために呼び出す。WS-POS サービスプロバイダは、**closeDevice** メソッドを受けて、そのドライバに対してクローズし、ドライバのインスタンスを破棄する。

Errors 例外が発生する可能性がある。例外による *ErrorCode* として取りうる値：

値	意味
E_ILLEGAL	指定された *consumerID* で確立されているセッションが無い。
E_CLOSED	既にクローズされている。

2.4.16 WSPOSEvent と WSPOSEventResponse *Version 1.2 で追加*

2.4.16.1 WSPOSEvent

PollForUPOSEvent メソッドで取得する WSPOSEvent は、実際にはデバイスクラス＋イベントの型に置き換えられる。

たとえば、POSPrinter の PollEvent メソッドは、POSPrinterEvent 型を返し、Scanner の **PollForUPOSEvent** は ScannerEvent 型を返す。

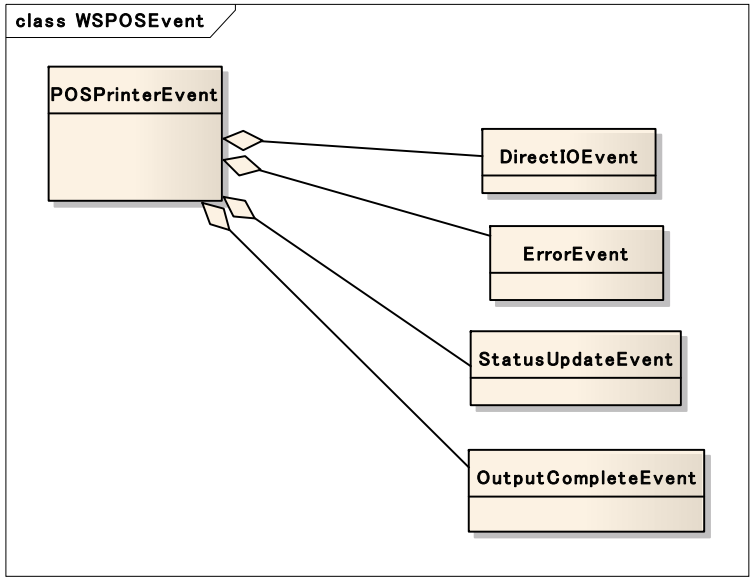
WSPOSEvent を構成する項目を下表に示す。

Unified**POS Event**	**説明**
DataEvent	UnifiedPOS で定義されている **DataEvent**。WS-POS サービスプロバイダにおいて、**DataEvent** が発生した時に有意な情報が設定され、WS-POS サービスコンシューマに渡される。他のイベントが発生した時は NULL が設定される。
StatusUpdateEvent	UnifiledPOS で定義されている **StatusUpdateEvent**。WS-POS サービスプロバイダにおいて、**StatusUpdateEvent** が発生した時に有意な情報が設定され、WS-POS サービスコンシューマに渡される。他のイベントが発生した時は NULL が設定される。
DirectIOEvent	UnifiedPOS で定義されている **DirectIOEvent**。WS-POS サービスプロバイダにおいて **DirectIOEvent** が発生した時に有意な情報が設定され、WS-POS サービスコンシューマに渡される。他のイベントが発生した時は NULL が設定される。
OutputCompleteEvent	UnifiedPOS で定義されている **OutputCompleteEvent**。WS-POS サービスプロバイダにおいて **OutputCompleteEvent** が発生した時に有意な情報が設定され、WS-POS サービスコンシューマに渡される。他のイベントが発生した時は NULL が設定される。
ErrorEvent	UnifiedPOS で定義されている **ErrorEvent**。WS-POS サービスプロバイダにおいて **ErrorEvent** が発生した時に有意な情報が設定され、WS-POS サービスコンシューマに渡される。他のイベントが発生した時は NULL が設定される。

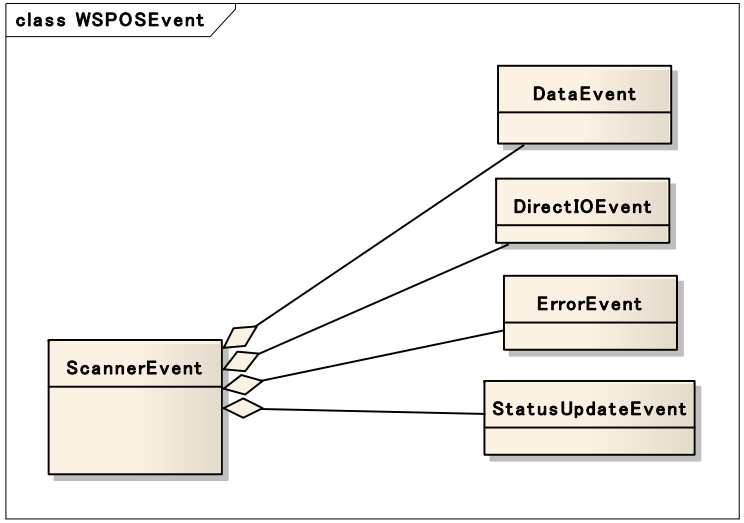
*注意: UnifiedPOS 1.14 で "**TransitionEvent**" イベントが電子バリューリーダライタデバイスクラスに対して追加されている。このイベントも上記と同じように扱われる。*

また、WSPOSEvent は、UnifiedPOS のデバイスクラスが通知しうるイベント型を集約する。

たとえば、出力デバイスである POSPrinter の POSPrinterEvent は、**StatusUpdateEvent**, **DirectIOEvent**, **OutputCompleteEvent**, **ErrorEvent** の 4 種類を集約する。



たとえば、入力デバイスである Scanner の ScannerEvent は、**DataEvent**、**StatusUpdateEvent**, **DirectIOEvent**, **ErrorEvent** の 4 種類を集約する。



DataEvent, **StatusUpdateEvent**, **DirctIOEvent**, **OutputCompleteEvent**, **ErrorEvent** については、UnifiedPOS の仕様書を参照すること。

*注意: UnifiedPOS 1.14 で "**TransitionEvent**" イベントが電子バリューリーダライタデバイスクラスに対して追加されている。このイベントも上記と同じように扱われる。*

2.4.16.2 WSPOSEventResponse

setEventResponse メソッドに WS-POS サービスコンシューマが渡す WSPOSEventResponse は実際にはデバイスクラス＋EventResponse 型に置き換えられる。

たとえば、POSPrinter デバイスの **setEventResponse** メソッドには WSPOSEventResponse は ScannerEventResponse クラスを渡し、Scanner デバイスの **setEventResponse** には WSPOSEventResponse は ScannerEventResponse を渡す。

WSPOSEventResponse を構成する項目を下表に示す。

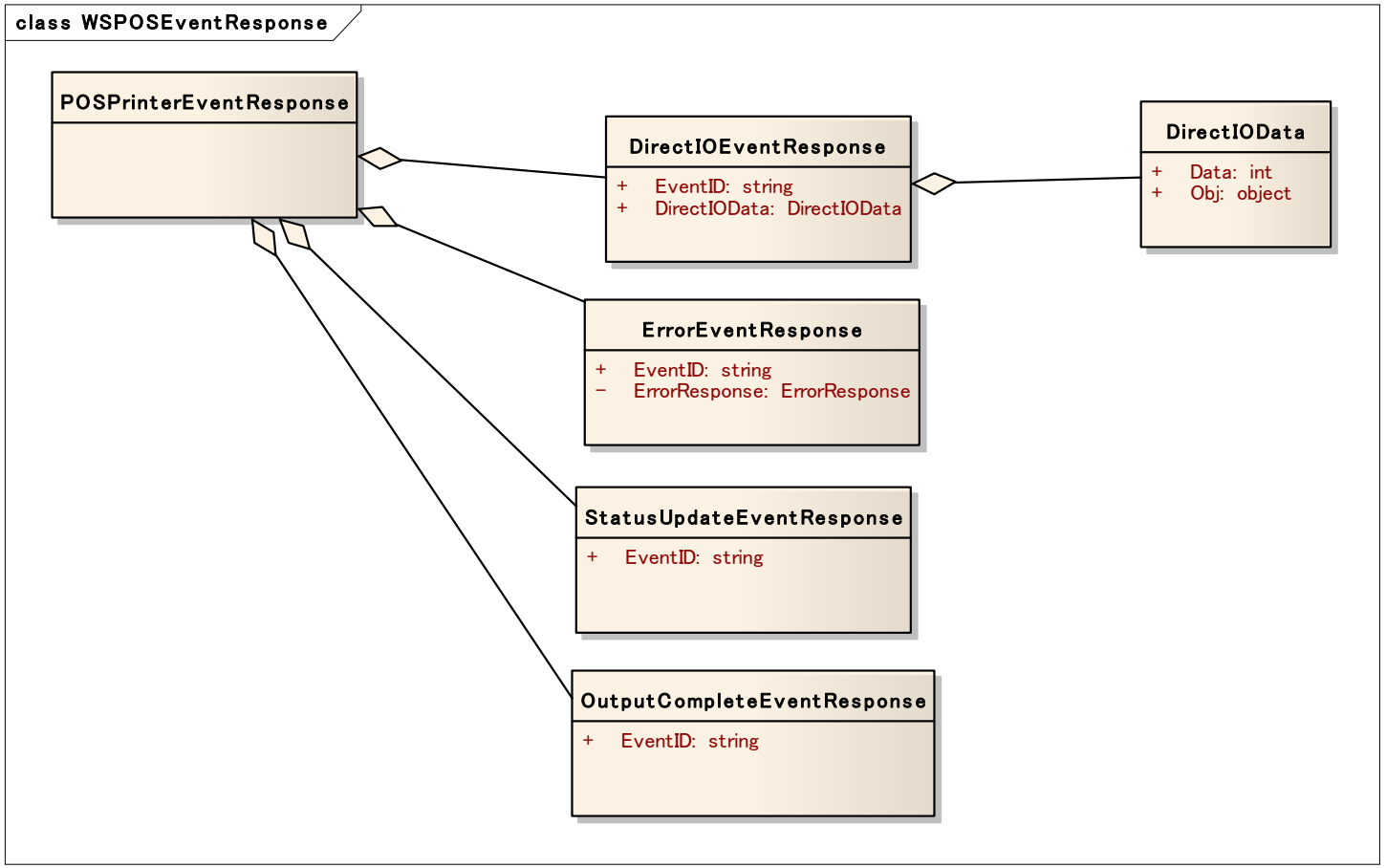
型	説明	対応する Event
DataEventResponse	WS-POS サービスコンシューマは、**pollForUPOSEvent** メソッドで取得した WSPOSEvent に含まれる DataEvent が NULL でなかった場合に、WSPOSEventResponse の DataEventResonse に有意な値を設定し、WS-POS サービスプロバイダに通知する。 DataEventResponse は、EventID 項目を持っている。WS-POS サービスコンシューマは、**pollForUPOSEvent** で取得した **DataEvent** の EventID 項目の値を、DataEventResponse の EventID に設定しなければならない。	**DataEvent**
StatusUpdateEventRespose	WS-POS サービスコンシューマは、**pollForUPOSEvent** メソッドで取得した WSPOSEvent に含まれる **StatusUpdateEvent** が NULL でなかった場合に、WSPOSEventResponse の StatusUpdateEventResonse に有意な値を設定し、WS-POS サービスプロバイダに通知する。 StatusUpdateEventResponse は、EventID 項目を持っている。WS-POS サービスコンシューマは、**pollForUPOSEvent** で取得した **StatusUpdateEvent** の EventID 項目の値を、StatusUpdateEventResponse の EventID に設定しなければならない。	**StatusUpdateEvent**
DirectIOEventResponse	WS-POS サービスコンシューマは、**pollForUPOSEvent** メソッドで取得した WSPOSEvent に含まれる **DirectIOEvent** が NULL でなかった場合に、WSPOSEventResponse の DirectIOEventResonse に有意な値を設定	**DirectIOEvent**

	し、WS-POS サービスプロバイダに通知する。 DirectIOEventResponse は、EventID 項目と DirectIOData 項目を持っている。WS-POS サービスコンシューマは、**pollForUPOSEvent** で取得した **DirectIOEvent** の EventID 項目の値を、DirectIOEventResponse の EventID に設定しなければならない。WS-POS サービスコンシューマは、DirectIOEventResponse の DirectIOData に、任意のデータを設定できる。	
OutputCompleteEventResponse	WS-POS サービスコンシューマは、**pollForUPOSEvent** メソッドで取得した **OutputCompleteEvent** が NULL でなかった場合に、WSPOSEventResponse の OutputCompleteEventResponse に有意な値を設定し、WS-POS サービスプロバイダに通知する。 OutputCompleteEventResponse は EventID 項目を持っている。WS-POS サービスコンシューマは **pollForUPOSEvent** メソッドで取得した **OutputCompleteEvent** の EventID の値を OutputCompleteEventResponse の EventID に設定しなければならない。	**OutputCompleteEvent**
ErrorEventResponse	WS-POS サービスコンシューマは **pollForUPOSEvent** メソッドで取得した WSPOSEvent に含まれる **ErrorEvent** が NULL でなかった場合に、WSPOSEventReponse の ErrorEventResponse に有意な値を設定し、WS-POS サービスプロバイダに通知する。 ErrorEventResponse は EventID 項目と ErrorResponse 項目を持っている。WS-POS サービスコンシューマは **pollForUPOSEvent** メソッドで取得した **ErrorEvent** の EventID 項目の値を ErrorEventResponse の EventID に設定しなければならない。WS-POS サービスコンシューマは ErrorEventResponse の ErrorResponse 項目に、 “Clear”, “ContinueInput”, “Retry” のいずれかを設定しなければならない。	**ErrorEvent**

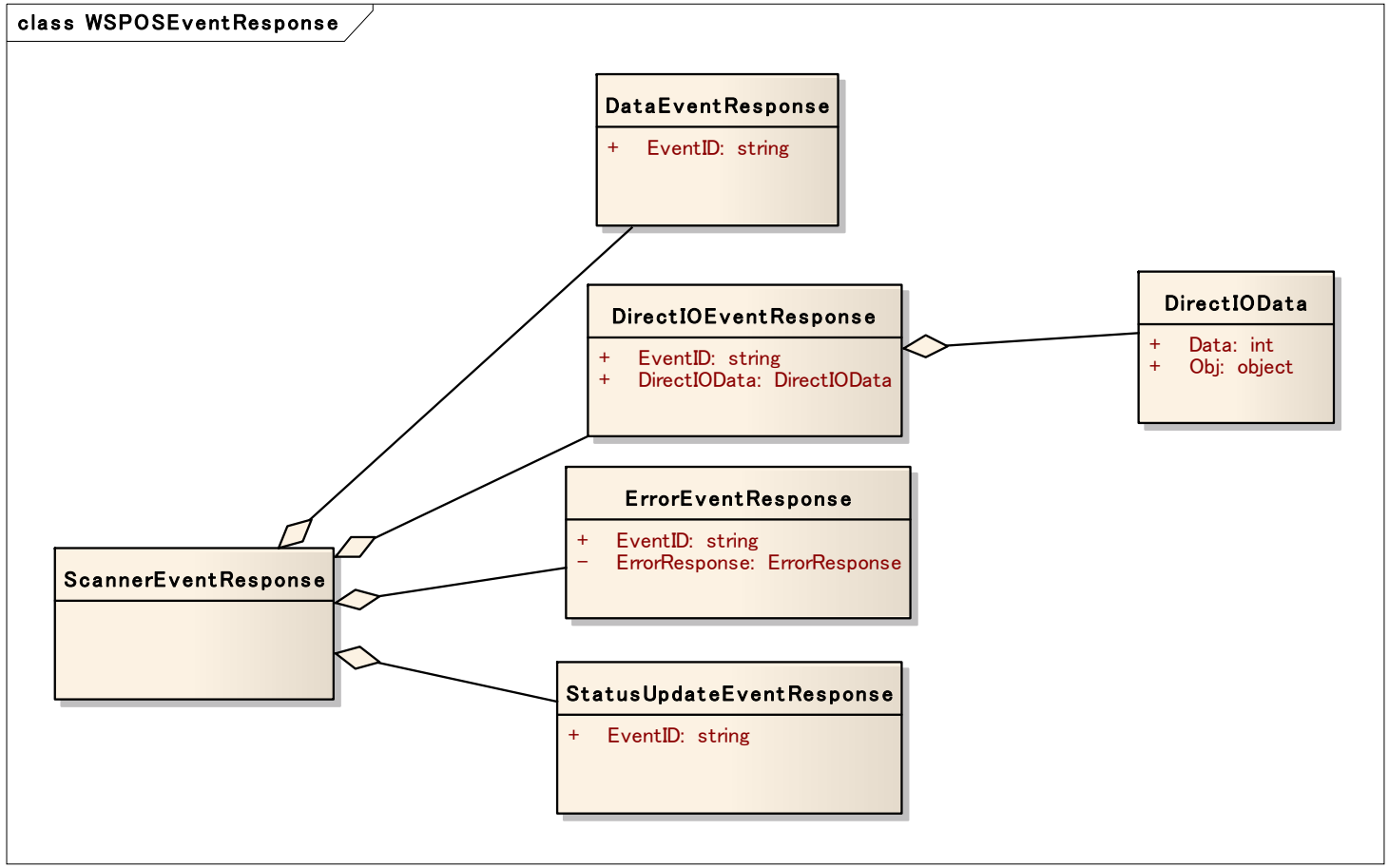
WSPOSEventResponse は、WSPOSEvent を構成する **DataEvent**, **StatusUpdateEvent**, **DirectIOEvent**, **OutputCompleteEvent**, **ErrorEvent** の 5 種類のイベントに対応した、DataEventResponse, StatusUpdateEventResponse, DirectIOEventResponse, OutputCompleteEventResponse, ErrorEventResponse のうち、デバイスクラスが受信しうる EventResponse 型を集約する。

*注意: UnifiedPOS 1.14 で "**TransitionEvent**" イベントが電子バリューリーダライタデバイスクラスに対して追加されている。このイベントも上記と同じように扱われる。*

たとえば、出力デバイスである POSPrinter の POSPrinterEventResponse は、StatusUpdateEventResponse, DirectIOEventResponse, OutputCompleteEventResponse, ErrorEventResponse の 4 種類を集約する。



入力デバイスである Scanner の ScannerEventResponse は、DataEventResponse、StatusUpdateEventResponse, DirectIOEventResponse, ErrorEventResopnse の 4 種類を集約する。



2.4.17 WS-POS 双方向通信 イベントリファレンス *Version 1.2 で追加*

本章では WS-POS 双方向通信のイベントのメソッドリファレンスを記す。

これらのメソッドは WS-POS サービスコンシューマが公開し、WS-POS サービスプロバイダが呼び出す。
このため、このリファレンスのパラメータの **inout** 宣言は、WS-POS サービスコンシューマから WS-POS サービスプロバイダへの情報の受け渡しもあることを意味する。(DirectIOEvent の data, obj, ErrorEvent の errorResponse)
下記のイベントメソッドは、デバイスクラスによっては WS-POS サービスプロバイダから呼び出されないものがある。たとえば入力デバイスである Scanner は outputCompleteEvent を呼び出さず、出力デバイスである POSPrinter は DataEvent を呼び出さない。

ポーリング方式の場合は、これらのメソッドは WS-POS サービスプロバイダから呼び出されず、WS-POS サービスコンシューマは PollForUPOSEvent メソッドを用いてイベントを取得し、SetEventResponse メソッドを用いてイベントに応答する。

2.4.17.1 dataEvent Method

Syntax **dataEvent (consumerID: *string*, source: *string*, eventID: *int32*, timeStamp: *DateTime*, status: *int32*): void**

パラメータ	説明
consumerID	WS-POS サービスプロバイダ内でコンシューマを一意に識別するための ID
source	UPOS デバイスクラス
eventID	UPOS デバイスが通知した、イベントの ID
timeStamp	UPOS デバイスが通知した、タイムスタンプ情報
status	UPOS デバイスで通知した、Status 情報 UPOS 仕様書:The input status with its value dependent upon the device category; it may describe the type or qualities of the input data.

Remarks 双方向通信方式でセッションを確立した後、WS-POS サービスコンシューマが WS-POS サービスプロバイダ

を経由してオープンした UPOS デバイスで DataEvent が発生した際、WS-POS サービスプロバイダは WS-POS サービスコンシューマのこのメソッドを呼び出す。イベントハンドリングについては UPOS の仕様に従うこと。

2.4.17.2 directIOEvent Method

Syntax **dataEvent (consumerID: *string*, source: *string*, eventID: *int32*, timeStamp: *DateTime*, eventNumber: *int32*, inout data: *int32*, inout obj: *object*) : void**

パラメータ	説明
consumerID	WS-POS サービスプロバイダ内でコンシューマを一意に識別するための ID
source	UPOS デバイスクラス
eventID	UPOS デバイスが通知した、イベントの ID
timeStamp	UPOS デバイスが通知した、タイムスタンプ情報
eventNumber	UPOS デバイスが通知した、EventNumber 情報。 UPOS 仕様書:Event number whose specific values are assigned by the UnifiedPOS Service.
data	UPOS デバイスが通知した、数値のデータ。 UPOS 仕様書:Additional numeric data. Specific values vary by the EventNumber and the UnifiedPOS Service. This attribute is settable.
obj	UPOS デバイスが通知した、オブジェクト情報 UPOS 仕様書:Additional data whose usage varies by the EventNumber and the UnifiedPOS Service. This attribute is settable.

Remarks 双方向通信方式でセッションを確立した後、WS-POS サービスコンシューマが WS-POS サービスプロバイダを経由してオープンした UPOS デバイスで DirectIOEvent が発生した際、WS-POS サービスプロバイダは WS-

POS サービスコンシューマのこのメソッドを呼び出す。イベントハンドリングについては UPOS の仕様に従うこと。

2.4.17.3 errorEvent Method

Syntax **errorEvent (consumerID: *string*, source: *string*, eventID: *int32*, timeStamp: *DateTime*, errorCode: *int32*, errorCodeExtended: *int32*, errorLocus: *int32*, inout errorResponse: *int32*) : void**

パラメータ	説明
consumerID	WS-POS サービスプロバイダ内でコンシューマを一意に識別するための ID
source	UPOS デバイスクラス
eventID	UPOS デバイスが通知した、イベントの ID
timeStamp	UPOS デバイスが通知した、タイムスタンプ情報
errorCode	UPOS デバイスが通知した、ErrorCode。 UPOS 仕様書:Error Code causing the error event.
errorCodeExtended	UPOS デバイスが通知した、ErrorCodeExtended。 UPOS 仕様書:Extended Error Code causing the error event. These values are device category specific.
errorLocus	UPOS デバイスが通知した、ErrorLocus。 UPOS 仕様書:Location of the error.
errorResponse	UPOS デバイスに指示する ErrorResponse。 UPOS 仕様書:Error response, whose default value may be overridden by the application (i.e., this property is settable).

Remarks 双方向通信方式でセッションを確立した後、WS-POS サービスコンシューマが WS-POS サービスプロバイダを経由してオープンした UPOS デバイスで ErrorEvent が発生した際、WS-POS サービスプロバイダは WS-POS サービスコンシューマのこのメソッドを呼び出す。イベントハンドリングについては UPOS の仕様に従うこと。

2.4.17.4 outputCompleteEvent Method

Syntax **outputCompleteEvent (consumerID: *string*, source: *string*, eventID: *int32*, timeStamp: *DateTime*, outputID: *int32*) : void**

パラメータ	説明
consumerID	WS-POS サービスプロバイダ内でコンシューマを一意に識別するための ID
source	UPOS デバイスクラス
eventID	UPOS デバイスが通知した、イベントの ID
timeStamp	UPOS デバイスが通知した、タイムスタンプ情報
outputID	UPOS デバイスが通知した、OutputID。 UPOS仕様書: The ID number of the asynchronous output request that is complete.

Remarks 双方向通信方式でセッションを確立した後、WS-POS サービスコンシューマが WS-POS サービスプロバイダを経由してオープンした UPOS デバイスで OutputCompleteEvent が発生した際、WS-POS サービスプロバイダは WS-POS サービスコンシューマのこのメソッドを呼び出す。イベントハンドリングについては UPOS の仕様に従うこと。

2.4.17.5 statusUpdateEvent Method

Syntax **statusUpdateEvent (consumerID: *string*, source: *string*, eventID: *int32*, timeStamp: *DateTime*,**

status: *int32*) : void

パラメータ	説明
consumerID	WS-POS サービスプロバイダ内でコンシューマを一意に識別するための ID
source	UPOS デバイスクラス
eventID	UPOS デバイスが通知した、イベントの ID
timeStamp	UPOS デバイスが通知した、タイムスタンプ情報
status	UPOS デバイスが通知した、status 情報。 UPOS仕様書: Device category-specific status, describing the type of status change..

Remarks 双方向通信方式でセッションを確立した後、WS-POS サービスコンシューマが WS-POS サービスプロバイダを経由してオープンした UPOS デバイスで StatusUpdateEvent が発生した際、WS-POS サービスプロバイダは WS-POS サービスコンシューマのこのメソッドを呼び出す。イベントハンドリングについては UPOS の仕様に従うこと。

2.4.18 XMLPOS への変更 *Version 1.2 で追加*

ARTS UnifiedPOS 1.14 にて定義されている XML POS スキーマを WS-POS 1.2 に対応させるには、以下の変更が必要になる。

- 共通メソッドの追加と削除
- ConsumerID パラメータの追加

上記のいずれも、WS-POS 1.2 において、セッション管理機能を追加したことで必要となる変更である。

2.4.18.1 共通メソッドの追加と削除

UnifiedPOS 1.14 XMLPOS の各 WSDL と xsd について、以下のメソッドを追加する。

メソッド	説明
generateConsumerID	コンシューマ ID を GUID または UUID として生成する。
openSession	WS-POS セッションを開始する。（デバイスの Open は行わない）
closeSession	WS-POS セッションを終了する。
getProviderSessionTimeout	WS-POS プロバイダに設定されているセッションタイムアウト値をコンシューマが取得する。
keepAlive	WS-POS コンシューマは、セッションを保持するために WS-POS プロバイダのこのメソッドを定期的に呼出しする。
pollForUPOSEvent	WS-POS プロバイダに対しイベントが発生しているか否かのポーリングをコンシューマが行う。
setEventResponse	PollForUPOSEvent で取得したイベントに対し、WS-POS コンシューマが応答を設定する。
getWSPOSVersion	WS-POS プロバイダが実装している WS-POS のバージョンを取得する。
openDevice	UnifiedPOS デバイスの Open を行う。
closeDevice	UnifiedPOS デバイスの Close を行う。

UnifiedPOS 1.13 XMLPOS の各 wsdl と xsd について、以下のメソッドを削除する。

メソッド	説明
open	openSession, openDevice に機能を分割したため、この UnifiedPOS メソッドは削除する。
close	cloesSession, closeDevice に機能を分割したため、この UnifiedPOS メソッドは削除する。

2.4.18.2 ConsumerID パラメータの追加

WS-POS バージョン 1.2 では、サービスプロバイダがサービスコンシューマを識別するために、サービスプロバイダにとって一意な ConsumerID を使用する。UUID または GUID 生成方法は、RFC4122 “The Internet Society”, http://www.ietf.org/rfc/rfc4122.txt を参照すること。ConsumerID は統計的に一意であるため、WS-POS サービスコンシューマは一度 ConsumerID を生成すれば、それ以降はずっとその ConsumerID を使うことができる。WS-POS サービスコンシューマが一意な ConsumerID を作成できない場合に備えて、WS-POS バージョン 1.2 では、WS-POS サービスプロバイダは ConsumerID を生成し WS-POS サービスコンシューマに返す機能 を要件として備えている。

ConsumerID パラメータは、すべてのメソッド（プロパティの Getter, Setter を含む）に対して、最初のパラメータとして挿入する。UnifiedPOS1.14 に記載されている XMLPOS の xsd スキーマをすべて変更する必要がある。

以下に例を示す。

UnifiedPOS 1.14 XML POS GetDeviceEnabled(パラメータなし)記述→ WS-POS 1.2 対応 XML POS 赤字が追加部分

```
<xs:element name="GetDeviceEnabled">
  <xs:complexType>
    <xs:sequence />
  </xs:complexType>
</xs:element>
<xs:element name="GetDeviceEnabledResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element
        minOccurs="0" name="GetDeviceEnabledResult"
       type="xs:boolean" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

->

```
<xs:element name="GetDeviceEnabled">
  <xs:complexType>
     <xs:element
        minOccurs="0"  name="ConsumerID" nillable="true"
        type="xs:string" />
   </xs:complexType>
</xs:element>
<xs:element name="GetDeviceEnabledResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element
        minOccurs="0" name="GetDeviceEnabledResult"
       type="xs:boolean" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

UnifiedPOS 1.14 XML POS SetDeviceEnabled(パラメータ有り)記述→WS-POS 1.2 対応 XML POS 赤字が追加部分

```
<xs:element name="SetDeviceEnabled">
 <xs:complexType>
  <xs:sequence>
   <xs:element minOccurs="0"
   name="DeviceEnabled" type="xs:boolean" />
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="SetDeviceEnabledResponse">
 <xs:complexType>
  <xs:sequence />
 </xs:complexType>
</xs:element>
```

->

```
<xs:element name="SetDeviceEnabled">
 <xs:complexType>
  <xs:sequence>
  <xs:element
      minOccurs="0"  name="ConsumerID" nillable="true"
      type="xs:string" />
   <xs:element minOccurs="0"
   name="DeviceEnabled" type="xs:boolean" />
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name="SetDeviceEnabledResponse">
 <xs:complexType>
  <xs:sequence />
 </xs:complexType>
</xs:element>
```

上記の修正をすべての UnifiedPOS1.14 対応周辺機器の xsd について実施する。

2.4.19 メソッド呼び出しのパラメータとして渡すファイルパス *Version 1.2 で追加*

UnifiedPOS には、POSPrinter の SetBitmap など、ファイルのパス名をパラメータとするメソッドが存在する。WS-POS では、通常 WS-POS サービスプロバイダと WS-POS サービスコンシューマは異なるデバイス上で実行されるため、このパラメータは不適切である。UnifiedPOS の API のセマンティクスを破壊せずに、WS-POS のコンテキストでこれらのパラメータに対応するために、WS-POS ではすべてのパス名を示すパラメータの指定方法を以下のように定義する。:

- パス名として WS-POS サービスコンシューマが WS-POS サービスプロバイダへ与えるパラメータは、URI(Uniform Resource Indicator)でなければならない。
- WS-POS サービスプロバイダは最低でも http:スキーマをサポートしなければならない。
- WS-POS サービスプロバイダは、http:スキーマに加えて、https:、file:および data:の 3 つのオプションのスキーマをサポートすることが推奨される。
- WS-POS サービスプロバイダに対し、file: URI を与える場合、その URI は、WS-POS サービスプロバイダが実行されているデバイス上のファイルを示す。

- WS-POS サービスプロバイダが file: URI を受け付ける場合、その URI の形式は WS-POS サービスプロバイダの個々の実装に依存し、WS-POS の規定外である。

典型的な動作は以下となる。

1. WS-POS サービスコンシューマは、WS-POS サービスプロバイダに受け付けさせたいファイルを Skydrive などのファイル共有サービス上へ事前にコピーし公開する。
   注意: WS-POS サービスプロバイダはこれらの URI へのアクセスに対し、認証をサポートしない。
2. WS-POS サービスコンシューマは、WS-POS サービスプロバイダへの要求パラメータとして上記の URI を指定する。
3. WS-POS サービスプロバイダは、指定された URI からリソースを取得する。
4. WS-POS サービスプロバイダは、後続のリクエストに対する応答速度向上のために、取得したリソースをローカルデバイス上にキャッシュする（実装によるオプションの動作）。
5. WS-POS サービスプロバイダは、下層のデバイスサービスに対し、取得したリソースを与える。このとき、下層のデバイスサービスがディスク上のファイルリソースを要求する場合、適切な方法でテンポラリファイルを作成して、該当ファイルのパス名を与える。このテンポラリファイルは、下層のデバイスサービスのメソッド呼び出しからの復帰時に削除する。
6. WS-POS サービスプロバイダは、メソッドの結果を WS-POS サービスコンシューマへ返す。

2.4.19.1 セキュリティに関する考慮

以下に、重要なセキュリティに関する考慮の例を挙げる:

- WS-POS サービスプロバイダは、与えられた URI が示すファイルの内容の妥当性について常に必要となる検証を行う必要がある。
- file: であれば、WS-POS サービスコンシューマに対して公開範囲のディレクトリ／ファイルであること
- http: であれば、指定されたホストが WS-POS サービスプロバイダが持つホワイトリストに掲載されていること
- data: であれば、データ長が適切な範囲であること

2.5 ステータス、状態モデル、および例外

次に示すように、共通で使う列挙体、イベント、プロパティの中には、ステータス、エラーコード、状態モデルが設定されているものがある。

2.5.1 StatusUpdateEvent

StatusUpdateEvent は特定のクラス固有の状態やステータス変数が変更されたときに発生するイベントである。

2.5.2 ControlState

ControlState は現在の状態を格納する列挙体。可能な値は以下のものである。

Closed
Idle
Busy
Error

2.5.3 例外

すべての WS-POS サービスのメソッドの呼び出しは、失敗時に **UposException** をスローする可能性がある。
UposException は、ARTS の XMLPOS スキーマによって、名前空間 http://www.nrf-arts.org/UnifiedPOS/以下の各デバイスで定義されている。

2.5.4 Public プロパティ

名前	説明
ErrorCode	**ErrorCode** はエラー例外の原因を表すエラーコード。列挙体で定義されている。
ErrorCodeExtended	**ErrorCodeExtended** はエラー例外の原因を表す拡張エラーコード。これには、サービス固有の値が格納されることがある。

2.6デバイス共有モデル

WS-POS のデバイス共有モデルは、複数のアプリケーション(WS-POS サービスコンシューマ)が部分的または全面的に共有できるデバイスのほか、同時には 1 つの WS-POS サービスコンシューマのみが排他的に使用するデバイスをサポートする。すべての WS-POS サービスプロバイダは、1 つ以上のアプリケーションにオープンされる。しかし、WS-POS サービスプロバイダで実行できる動作の中には、デバイスへのアクセス権を獲得するアプリケーションを 1 つに制限するものがある。

2.6.1 排他使用デバイス

もっとも一般的なデバイスタイプは「排他使用デバイス」で、一例が POS プリンタである。物理的特性、動作上の特性により、このデバイスは同時には 1 つのアプリケーション(WS-POS サービスコンシューマ)でしか利用できない。WS-POS サービスコンシューマは、大半のメソッド、プロパティ、イベントが有効になる前に、**Claim** メソッドを呼び出して排他アクセスできるようにしなければならない。排他アクセス権を獲得するまえに、メソッドを呼び出したり、プロパティを設定したりすると、エラーが発生する。
密接に関連する 2 つの WS-POS サービスコンシューマが共有方式で排他使用デバイスを使用したい場合、一方の WS-POS サービスコンシューマは一部の時間だけデバイスを排他アクセスして、その後、他方の WS-POS サービスコンシューマも使用できるように、排他アクセス権を解放するという方法もある。
Claim メソッドが再び呼び出されると、設定可能なデバイスの特性は、**Release** メソッド時の状態に復元される。復元される例として、ラインディスプレイの輝度、磁気ストライプリーダの読み取りトラック、POS プリンタの行あたりの文字数などがある。POSPrinter のセンサプロパティのような **State** 特性は復元されない。その代わり、現在の状態に設定される。

2.6.2 Sharable Devices

いくつかのデバイスは共有可能デバイスである。一例がキーロックである。共有可能デバイスの場合、複数のアプリケーション(WS-POS サービスコンシューマ)でそのメソッドを呼び出して、プロパティにアクセスすることが可能となる。また、デバイスをオープンしたすべてのアプリケーションにイベントが通知される。ただし、共有可能デバイスを排他アクセスした WS-POS サービスコンシューマにより、一部のメソッドやプロパティのアクセスを制限することや、その WS-POS サービスコンシューマだけにイベントを通知させることもできる。

2.7 イベントメッセージ

イベントは、デバイスの様々な動作や変化、あるいはデバイスの着脱をアプリケーション(WS-POS サービスコンシューマ)に通知する。イベントには以下 5 つがある。

イベント	説明
DataEvent	入力データがデバイスクラス固有のプロパティに格納されている。
ErrorEvent	イベント駆動入力中または非同期出力中にエラーが発生した。
StatusUpdateEvent	デバイスの状態変化を報告する。
OutputCompleteEvent	非同期出力が正常に完了した。
DirectIOEvent	このイベントは、サービスプロバイダによって定義され、本仕様でカバーできないものを提供する。

WS-POS サービスプロバイダはイベントが発生するとイベントをキューイング（Queue）する。キューイングされたイベントは通知可能な状態になるとアプリケーション(WS-POS サービスコンシューマ)に通知（Deliver）される。イベントの引き渡しを遅らせる要因には以下のものがある。

- WS-POS サービスコンシューマが FreezeEvents プロパティを true に設定している。
- イベントタイプが DataEvent または入力 ErrorEvent で、DataEventEnabled プロパティが false である。

イベントの通知に関する用語は、本項「イベント」以外では下記用語の使い分けは行っていないが、参考のため掲載している。

備考: 本書では、下記のイベントに関する用語が使用されている。

Queue	サービスがWS-POSサービスコンシューマに対しイベントの通知（Fire）の必要性を決定したとき、サービスが内部のイベントキューにイベントを詰め込む。
Deliver	イベントキューが空でなく、キューの先頭のイベントについてのすべての条件が揃ったとき、このイベントはキューから外され、アプリケーションに対するイベント通知要求が実行される。
Fire	キューイング（Queue）と通知（Deliver）の組み合わせと言える。時にこの用語は大まかに使用され、これらのステップの片方だけしか意味しない場合もある。前後関係からこれらを識別しなければならない。

イベントのキューの管理に関する規定は以下のとおりである

- WS-POS サービスプロバイダは、デバイスがイネーブルである間は新しいイベントをキューイング（Queue）するのみである。
- WS-POS サービスプロバイダは、WS-POS サービスコンシューマが Release メソッド（排他使用デバイス）または **closeDevice** メソッド（すべてのデバイス）を呼び出すまで、キューイング（Queue）されたイベントの通知（Deliver）を行う可能性がある。また、前記メソッド実行時には残存しているイベントはすべて削除される。
- 入力デバイスでは、**ClearInput** メソッドはデータと入力エラーイベントをクリアする。イベントハンドラ内で、WS-POS サービスコンシューマは、プロパティにアクセスしたり、メソッドの呼び出しを行える。しかし、WS-POS サービスコンシューマは **release** メソッドや **closeDevice** メソッドをイベントハンドラ内から呼び出してはならない。**release** メソッドはイベントハンドリング（イベントを通知（Deliver）させるスレッドを含んでいる）を閉鎖してしまい、**closeDevice** は戻る前にイベントハンドリングを閉鎖しなければならないためである。

2.8 入力モデル

WS-POS の入力モデルは、イベント駆動入力をサポートする。イベント駆動入力では、入力データは **DeviceEnabled** が true に設定された後に受信できるようになる。受信されたデータは、条件が満たされている場合に、アプリケーション(WS-POS サービスコンシューマ)に通知される **DataEvent** イベントとしてキューイングされる。データ受信時に **AutoDisable** プロパティが true の場合、WS-POS サービスプロバイダは **DeviceEnabled** を false に設定して自動的にデバイスをディセーブルにする。これはサービスが更に入力をキューイングするのを妨げ、可能な場合は物理的にデバイスをディセーブルにする。

WS-POS サービスコンシューマは、デバイスからの入力を受信する準備ができたら **DataEventEnabled** プロパティを true に設定する。入力を受信すると、WS-POS サービスプロバイダは、**DataEvent** イベントをキューイングし、通知する。（入力がすでにキューイングされている場合は、**DataEvent** イベントが通知される。）このイベントは、入力ステータス情報を含むことがある。WS-POS サービスプロバイダは、イベントを通知する直前に、入力データと、必要に応じてその他の情報をデバイス固有のプロパティに格納する。

イベントを通知する直前に、WS-POS サービスプロバイダは内部的に **DataEventEnabled** プロパティを false に設定することによって、それ以上のデータイベントを不可能にする。これにより WS-POS サービスコンシューマが現在の入力と関連のプロパティを処理している間、それ以降の入力データはサービスによってキューイングされする。WS-POS サービスコンシューマは、現在の入力の処理を終了し、次のデータを受け付ける準備ができたとき、**DataEventEnabled** を true に設定してイベントの通知を再開するようにする。

入力デバイスが排他使用デバイスの場合、WS-POS サービスコンシューマは、デバイスが入力の読み取りを開始する前に、そのデバイスの排他アクセス権の獲得とイネーブルの両方を行わなければならない。

共有可能な入力デバイスの場合、WS-POS サービスコンシューマは、デバイスが入力の読み取りを開始する前に、そのデバイスをオープンして、イネーブルにしなければならない。WS-POS サービスコンシューマは、WS-POS サービスプロバイダが **DataEvent** イベントを使用してデータを送る前に、**Claim** メソッドを呼び出してデバイスの排他アクセスを要求しなければならない。イベント駆動入力を受信したとき、WS-POS サービスコンシューマがデバイスを排他アクセスしていない場合、デバイスの排他アクセス権を獲得する（かつ **DataEventEnabled** プロパティが true になる）まで、入力はキュ

ーイングされる。これによって、複数の WS-POS サービスコンシューマがデバイスを順序よく共有でき、WS-POS サービスコンシューマ間で入力フォーカスを効果的に受け渡すことができる。

イベント駆動入力によりデータの入力処理をしているときに、WS-POS サービスプロバイダがエラーを見つけると、WS-POS サービスプロバイダは自身の **State** プロパティを Error に変更し、WS-POS サービスコンシューマにエラー状態を警告するために、1～2 個の **ErrorEvent** イベントをキューイングする。**DataEventEnabled** プロパティが true になるまで、このイベントは通知されないため、WS-POS サービスコンシューマは順序よく処理することができる。エラーイベントは、次のエラー位置を示すデータを付帯して通知される。

InputData:

InputData は 1 つ以上の **DataEvent** イベントがキューイングされている間にエラーが発生した場合に、キューイングされる。このイベントは、すべての **DataEvent** イベントより先にキューイングされる。このイベントにより、WS-POS サービスコンシューマは入力を直ちにクリアするか、オプションでユーザにエラーの警報を出して、バッファリングされた入力済のデータを処理することができる。

後者のケースはスキャナで有効である。ユーザに直ちにエラー警報を出すことができるので、エラーが解除されるまで後続のアイテムはスキャンされない。エラー復旧を実行する前に、以前スキャンしたアイテムを正常に処理できる。

Input:

Input はエラーが発生し、利用できるデータがないときに通知される。（一般的な実装では、このイベントをイベントキューの末尾にセットする。）エラーが発生したときに入力データの一部をすでにキューイングしている場合、エラー位置が **InputData** の **ErrorEvent** イベントがまずキューイング、通知され、このエラーイベントはすべての **DataEvent** イベントが通知された後に通知される。（**“InputData”**イベントが通知され、WS-POS サービスコンシューマのイベントハンドラが”Clear”応答した場合、本**“Input”**イベントは通知されない。）

WS-POSサービスプロバイダは、以下のいずれかが発生した場合に、Error状態を終了する。

- WS-POS サービスコンシューマが **Input** の ErrorEvent イベントを終了する場合。
- WS-POS サービスコンシューマが **InputData** の **ErrorEvent** イベントを **Clear** の ErrorResponse で終了する場合。
- WS-POS サービスコンシューマが **ClearInput** メソッドを呼び出した場合。

WS-POS サービスプロバイダによっては、イベント駆動入力を開始するためにメソッドを呼び出す必要がある。WS-POS サービスプロバイダがデバイスから入力データを受け取った後、一般的には、入力を再初期化するためにメソッドを再度呼び出すまでは、以降の入力データを受け付けない。例として、MICR とシグニチャキャプチャデバイスがある。このタイプのイベント駆動入力は「非同期入力」と呼ばれることもある。

DataCount プロパティは WS-POS サービスプロバイダでキューイングされた **DataEvent** イベントの数を得るために参照される。

WS-POS サービスプロバイダがキューイングしたすべての入力データは、**ClearInput** メソッドを呼び出せば削除できる。**ClearInput** メソッドは、共有可能デバイスでは **openDevice** 後に、排他使用デバイスでは **claim** 後に呼び出せる。

一般的なイベント駆動入力モデルは、入力データを直接返すメソッドやプロパティを含むデバイスクラスの定義を特に定めていない。デバイスクラスの中には、より直感的または柔軟に操作できるようなメソッドやプロパティを定義しているものがある。例として、キーロックデバイスがある。このタイプの入力は「同期入力」と呼ばれることもある。

2.9 出力モデル

WS-POS の出力は、同期と非同期の 2 つの出力タイプからなる。各デバイスクラスは、一方または両方のタイプをサポートすることも、両方ともサポートしないこともある。

2.9.1 同期出力

デバイス出力を迅速に実行したい場合にこの出力タイプが有効である。そのメリットは単純さにある。

アプリケーション（WS-POS サービスコンシューマ）はクラス固有のメソッドを呼び出して出力を実行する。WS-POS サービスプロバイダは出力が完了するまで制御を返さない。

2.9.2 非同期出力

デバイス出力が低速のハードウェアで、アプリケーション（WS-POS サービスコンシューマ）が対話的処理を必要とする場合に、この出力タイプが有効である。デバイスが出力を実行しているときに、WS-POS サービスコンシューマが別の処理を実行できることから、確実な応答性があることがメリットである。

さらなるプロセスの詳細は以下の通りである：

- WS-POS サービスコンシューマはクラス固有のメソッドを呼び出して出力を開始する。サービスは、プログラムメモリ内のリクエストをバッファリングして、物理デバイスがそれを受け取り処理できるようになると、すぐに物理デバイスに送付し、このリクエストの識別子を **OutputId** プロパティに設定して、できる限り迅速に戻る。デバイスがリクエストを正常に終了すると、WS-POS サービスプロバイダは、**OutputCompleteEvent** イベントを通知する。このイベントのパラメータは完了したリクエストの **OutputId** である。

- 非同期リクエストを実行しているときにエラーが発生すると、**ErrorEvent** イベントが通知される。WS-POS サービスコンシューマのイベントハンドラは、未処理の出力を再試行するか、またはそれをクリアできる。**ErrorEvent** イベントの処理中は、サービスは Error 状態である。（備考：エラーを引き起こす原因が除去されていない場合は、サービスは直ちに Error 状態に戻り、別の **ErrorEvent** イベントを通知する。）

- 非同期出力は FIFO ベースで実行される。バッファリングされた出力はすべて（非同期出力をすべて含む）、**ClearOutput** メソッドを呼び出すことによって削除できる。**OutputCompleteEvent** イベントは、クリアされた出力に対しては通知されない。このメソッドは（可能ならば）処理中の出力も停止する。

2.10 デバイス電源通知モデル

アプリケーション（WS-POS サービスコンシューマ）では、使用しているデバイスの電源状態を知る必要がある場合がある。この状態は、**PowerState** 列挙体によって管理される。

注: *このモデルは、PCやPOS端末本体の電源状況（「バッテリ・オン」や「バッテリ・ロー」など）を通知することを目的にしていない。これらの情報の通知は、現在POSPowerデバイスによって管理されている。*

2.10.1 モデル

WS-POS デバイスの電源を以下の 4 つの状態に分類する。

- **Online** デバイスは*電源オン状態*で*かつレディ状態*である。これは「使用可能」状態を示す。
- **Off** デバイスは*電源オフ状態*または*本体に未接続状態*である。これは「使用不可能」状態を示す。
- **Offline** デバイスは*電源オン状態*だが、*ノットレディ状態*か*応答不可状態*である。ボタンを押して*オンライン状態*にする必要があるかもしれない。または、本体からの要求に応答できない状態かもしれない。これは「使用不可能」状態を示す。
- **OffOffline** デバイスは*電源オフ状態*か*ノットレディ状態*かを判断できない。これは「使用不可能」状態を示す。

電源通知は、デバイスがオープン、排他アクセス権の獲得（排他デバイスの場合）を経てイネーブル状態となっている場合にのみ機能する。

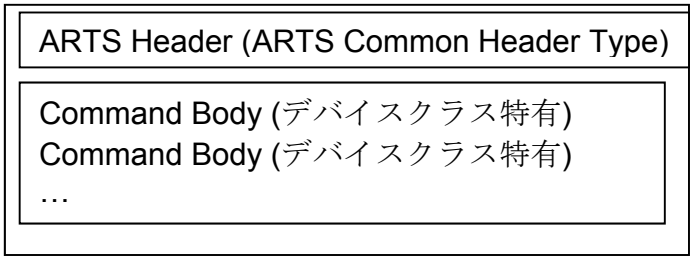
備考：イネーブル／ディセーブル vs. 電源状態

これら 2 つの状態は、全く別のもので通常は関連がない。WS-POS では「イネーブル」／「ディセーブル」は論理的な状態であるのに対し、電源状態は物理的な状態であると定義している。デバイスによっては、論理的には「イネーブル」であっても、物理的には「オフライン」かもしれない。または、論理的には「ディセーブル」であっても、物理的には「オンライン」かもしれない。

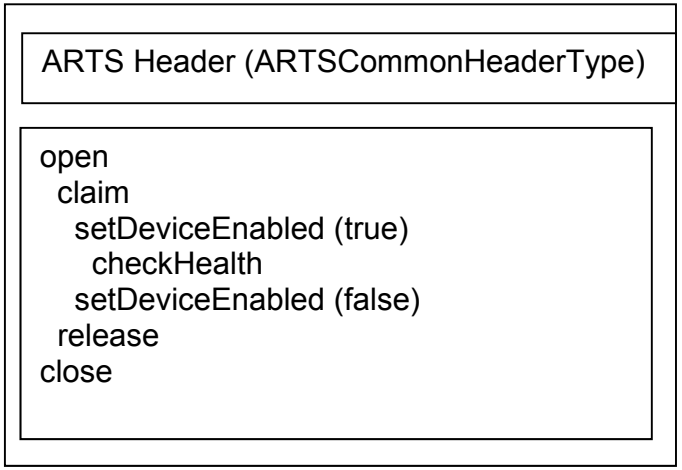
物理的な電源状態にかかわらず、WS-POS は、デバイスがイネーブル状態の場合のみ状態を通知する。（サービスは、一般的にイネーブル状態の間しかデバイスと通信できないので、この制限が必要である。）デバイスが「オフライン」の場合でも、サービスは、デバイスを「イネーブル」にしようとして失敗することがある。しかし、一度「イネーブル」になると、サービスは電源状態が変わってもデバイスを「ディセーブル」にはしない。

2.11 ARTS XMLPOS コマンド群

ARTS XMLPOS では、メソッドおよびプロパティに対する一連の処理を一括のメッセージで送信するためのコマンド群が定義されている。
これは、以下の構成を取る。



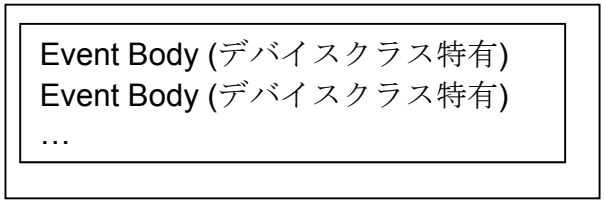
たとえば、open, claim, enabled, checkHealth, disable, release, close を一括で処理する場合は、



という一連のコマンドを一度に送信することができる。
これにより、WS-POS サービスコンシューマと WS-POS サービスプロバイダ間のメッセージ粒度を適切に調整可能である。

2.12 ARTS XMLPOS イベント群

ARTS XMLPOS では、複数種類のイベントを一括してサービスコンシューマに通知するためのイベント群が定義されている。
これは、以下の構成を取る。



2.13 ARTS XMLPOS スキーマ

WS-POS では、デバイスクラスの定義に ARTS XMLPOS スキーマを使用する。ARTS XMLPOS スキーマは xsd ファイルで提供される。

スキーマファイル名と、デバイスクラスの対応は、以下のようになっている。

スキーマ	説明
<Device Class Name> <Version>.xsd	デバイスクラスのプロパティ・メソッドを定義しているスキーマ
<Device Class Name> CommandSet <Version>.xsd	デバイスクラスの、Command Set を定義しているスキーマ
<Device Class Name> Event <Version>.xsd	デバイスクラスのイベントを定義しているスキーマ
<Device Class Name> EventSet <Version>.xsd	デバイスクラスの Event Set を定義しているスキーマ

注: *バージョンは、V をプレフィックスしてメジャーバージョン、マイナーバージョン、フィックスバージョンをピリオドで連結した文字列となる。(例: V1.13.1)*
バージョンに関しては ARTS SOA Best Practicies Technical Report を参照すること。

2.14 WS-POS WSDL

ARTS XMLPOS スキーマに準拠して、WS-POS サービスの WSDL を定義しているものが、以下の WSDL ファイル群である。

以下の WSDL は、ARTS XMLPOS スキーマと互換性があるが、WSDL にはバインディング情報も記述されているため、実際に使用する際には、バインディング情報を、使用する WS-POS サービスプロバイダに合わせて変更する必要がある。この WSDL ファイルに含まれる WSDL のバインディング情報はプレースホルダである。

WSDL	説明
<Device Class Name> <Version>.wsdl	デバイスクラスのプロパティ・メソッドを記述している WSDL
<Device Class Name> CommandSet <Version>.wsdl	デバイスクラスの、Command Set を記述している WSDL
<Device Class Name> Event <Version>.wsdl	デバイスクラスのイベントを記述している WSDL
<Device Class Name> EventSet <Version>.wsdl	デバイスクラスの Event Set を記述している WSDL

注: *バージョンは、V をプレフィックスしてメジャーバージョン、マイナーバージョン、フィックスバージョンをピリオドで連結した文字列となる。(例: V1.13.1)*
バージョンに関しては ARTS SOA Best Practicies Technical Report を参照すること。

2.14.1 Web Service Description Language (WSDL)

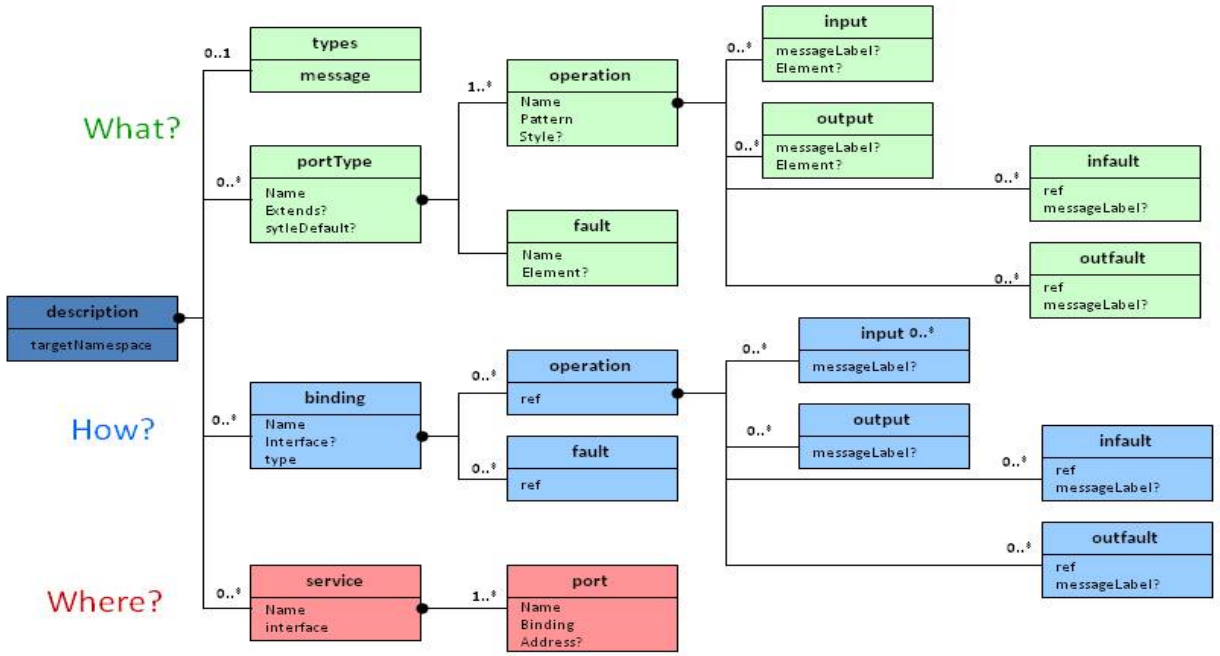


Figure 7: WSDL 概観

WS-POS は、WS-I basic profile version 1.2 に記述されている、WSDL 1.1 をサービス記述言語として採用する。

Id	名称	説明
SD001	WSDL	デバイスが提供する Web サービスは、本仕様書の “サービス記述”の章にしめされた WSDL に準拠しなければならない
SD002	WSA-Action	準拠するためには、Web サービスの受信を行う場合 WS-Addressing Action ヘッダを含まないメッセージは破棄しなければならない

2.14.2 WWS-POS WSDL 相互運用性と等価コード変換を確実にするための開発プラットフォームへの要求

ハードウェアとソフトウェアの相互運用性を確保するため、WS-POS WSDL は Java コードにより生成された場合と、.NET コードにより生成された場合で同じでなければならない。

WCF コントラクトが Windows .NET コードが WS-POS WSDL を生成する処理を行う時に使われる。さらに、JAX-WS コントラクトは Java コードが WS-POS WSDL を生成する処理を行う時に使われる。これらの２つの WS-POS WSDL は完全に一致していなければならない。これは、WS-POS 相互運用性が WSDL を生成する開発環境に依存せずに確保できることを確実にする唯一の方法である。

以下の図は、これらの関係と等価な WS-POS WSDL 生成の重要性を示す。

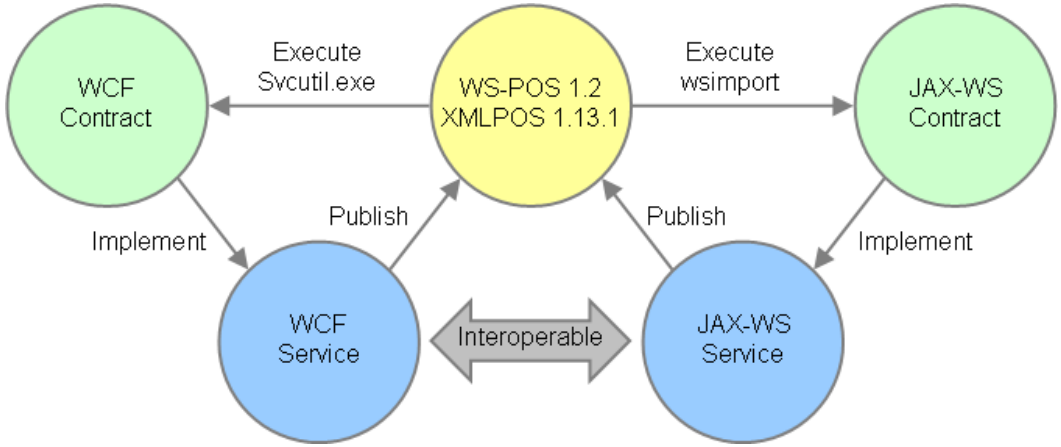


Figure 8: 相互運用性とコード変換に対する環境の等価性

WS-POS 標準は、ARTS が定義した WSDL を用いて対応付けされた UnifiedPOS クラスを定義するため、既存の UnifiedPOS 標準 XMLPOS メッセージを利用する。これにより、周辺機器に対して、UnifiedPOS のプロパティ、メソッド、イベントと 整合性のとれた WSDL の対応付けが確実に行える。さらに、UnifiedPOS デバイスの引数、戻り値、例外のリストが、WSDL により、それぞれリクエストメッセージ、レスポンスメッセージ、失敗メッセージに対応づけされる。

最後に、UnifiedPOS データ型は、等価な XML スキーマデータ型に対応付けされる。

Figure 9 はこれらの対応付けを示す。

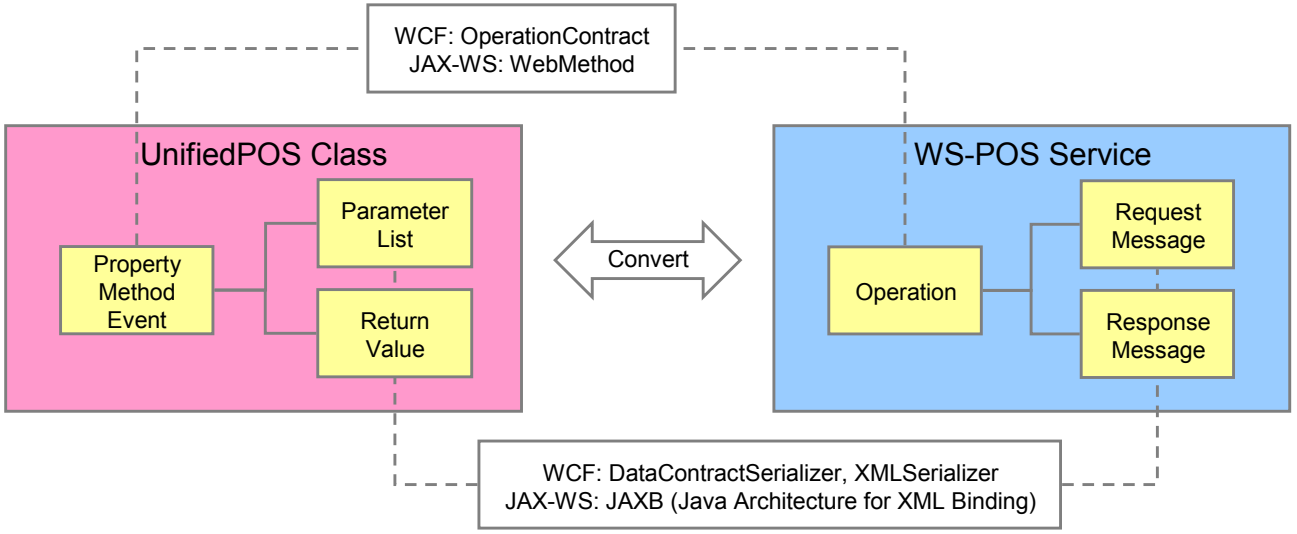


Figure 9: UnifiedPOS と WSDL の対応付け

2.14.3 ARTS が提供する WSDL 文書

それぞれのデバイスのタイプに対して、ARTS は WSDL の２つのカテゴリを提供する：１つ目はデバイス側に実装するサービスを記述し、もう一つは、そのデバイスのサービスを利用するアプリケーション側に実装するサービスである。提供される WSDL はそれぞれ、操作に必要な１つか２つのポートの型、操作に必要なすべてのメッセージ、およびそれぞれのポート型に関連する SOAP バインディング方法を含んでいるだろう。

ARTS WSDL ファイル群は WS-POS1.2 のダウンロードパッケージに含まれる。最新の WS-POS 文書と WSDL の zip/tar パッケージは WWW.NRF-ARTS.ORG の WS-POS セクションから取得できる。

2.14.4 実装者により生成されるべき WSDL 文書

WS-POS を実装しようとする、それぞれのデバイスもしくはクライアントは、関連する ARTS WSDL を取り込んだ新しい WSDL 文書を生成する必要がある。また、実装者は ARTS WSDL の一つもしくは複数のバインディングを参照するエンドポイントアドレスとポートをもったサービスを作成する。

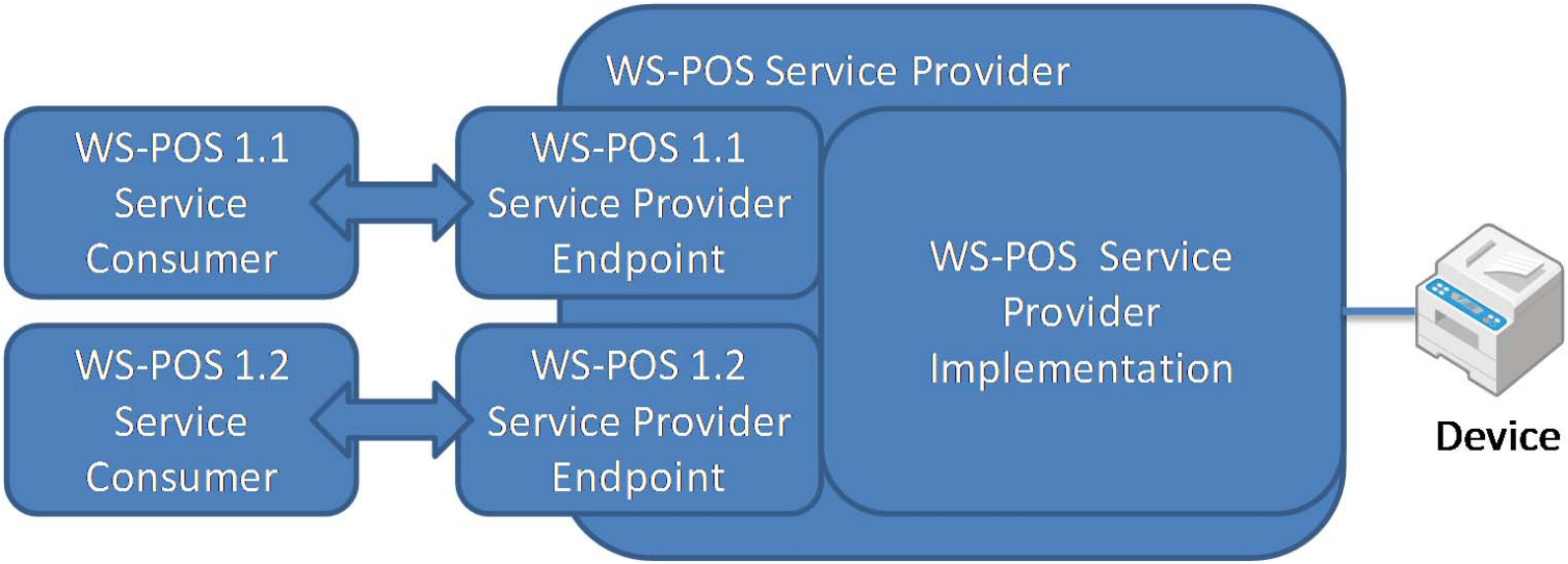
Id	名称	説明
SD003	ARTS-Binding	実装されるサービスは ARTS WSDL によって提供される（そのサービスに）対応したバインディングを参照しなければならない

2.15 後方互換性 *Version 1.2 で追加*

2.15.1 WSDL Version 1.1 と Version 1.2 の使用

WS-POS バージョン 1.1 と WS-POS バージョン 1.2 の WSDL は同一ではない。WS-POS 標準において機能的な違いがある。このことから、WS-POS バージョン 1.1 と WS-POS バージョン 1.2 のコンシューマ・プロバイダ間での相互運用はできない。機能が追加された WS-POS バージョン 1.2 へのマイグレーションを強く推奨する。

To be clear, WS-POS サービスコンシューマと WS-POS サービスプロバイダのバージョンは同じである必要がある。



2.16 セキュリティ

2.16.1 セキュリティについて

セキュリティは ARTS 分散周辺機器システム全体の重要な焦点である。今日のコンシューマシステムはセキュリティへの考慮が必要な情報、たとえば注文時のクレジットカード番号や明細などを送受信しており、安全なメッセージ交換は必須である。デバイスの認証、認証の保証、データ盗聴や改ざんからの安全性が必要である。

2.16.2 トランスポート層セキュリティ(TLS)

ポイントツーポイントメッセージでは、Transport Layer Security (TLS) 1.2 を使用できる。また、メッセージレベルのセキュリティでは WS-POS は WS-Security core, version 1.1 を使用できる。

Id	名称	説明
SE001	TLS	実装は TLS version 1.2 に準拠しなければならない。

2.16.3 UDDI によるデバイス認証

UDDI サーバはデバイスのキーが受理可能か否かを判断できなければならない。これには、プリロードされたテーブルや外部の認証機構等を用いる。

デバイスは UDDI による通信のための署名を提供する必要がある。UDDI サーバはデバイス使用を許可するためにこれらの署名を使用しなければならない。

Id	名称	説明
SE002	DeviceAuthUDDI	UDDI は、すべての認証されていないメッセージを受理してはいけない
SE003	DeviceAuthUDDIFail	UDDI は、認証されていないデバイスからのメッセージをすべて受理してはいけない

2.16.4 UDDI によるクライアント認証

UDDI サーバはクライアントのキーが受理可能か否かを判断できなければならない。これには、プリロードされたテーブルや外部の認証機構等を用いる。

クライアントは UDDI による通信のための署名を提供する必要がある。UDDI サーバはクライアント使用を許可するためにこれらの署名を使用しなければならない。

Id	名称	説明
SE004	ClientAuthUDDI	UDDI は、クライアントからの認証されていないメッセージをすべて受理してはいけない
SE005	ClientAuthUDDIFail	UDDI は認証されていないクライアントからのメッセージをすべて受理してはいけない

2.16.5 デバイスによるクライアント認証

デバイスと通信する場合、クライアントはすべてのメッセージに署名する必要がある。最初のメッセージ(OpenSession)は、デバイスがイベントを送るエンドポイントを含む。 デバイスは後で通信するために、クライアント用に Name-Value ペア(エンドポイントと署名)を格納する必要がある。 クライアントからデバイスにメッセージを送信する際、デバイスは署名と<from>WS-Addressing フィールドからエンドポイントを使用できる。デバイスをオープンおよびクローズする際、デバイスは<from> WS-Addressing フィールドが SOAP ボディのエンドポイントと一致するか否かを検証する。

Id	名称	説明
SE006	ClientAuthDevice	デバイスは、クライアントからの認証されてないメッセージをすべて受理してはいけない
SE007	FromSIGMatch	デバイスは、ヘッダの<from>要素が、シグネチャの EndpointReference と一致していないすべてのメッセージを受理していはいけない
SE008	FromEPRMatch	デバイスは、SOAP 本文の<in>と SOAP ヘッダの<from>が一致していない open と close メッセージを受理してはいけない

注意: クライアントは自分自身のエンドポイントのみ参照できる; クライアントが他のエンドポイントを参照する方法は考慮されていない。これは WebService のイベント登録におけるすべての形式で既知のエンドポイントの登録の制限である。

2.17 XML ペイロード

WSDL で記載されているように、WS-POS における SOAP メッセージの XML ペイロードは例外やクライアントへのコールバックメソッドを含め、ほぼすべて XMLPOS を使用して指定されている。

3. 概要フロー

3.1 WS-POS の概要フロー

このセクションは WS-POS スタックを使用するうえで規定された方法を説明する。

3.1.1 デバイスサービスの初期化

1. デバイスのセットアップ— アドレス、ケイパビリティ、名称、UDDI ロケーション。
2. UDDI への登録—自己認証、アドレス、ケイパビリティ、名称
    a. デバイスプライベートキーによるメッセージへの署名
    b. UDDI による署名の有効化

3.1.2 デバイス シャットダウン

1. デバイスは UDDI から自身を検知する。
2. デバイスの電源を切断する。

3.1.3 クライアントの開始

1. クライアントのエンティティをセットアップ—アドレス、ケイパビリティ、名称、UDDI ロケーション

3.1.4 クライアントがデバイスと通信する

1. UDDI に対してデバイスサービスのリストを問い合わせる—認証
    a. メッセージはクライアントのプライベートキーによって署名
2. UDDI はデバイスのリストをクライアントに返す—クライアントは接続先デバイスを選択
3. クライアントはデバイスサービスをオープンし、イベント用の登録を行う
    a. メッセージはクライアントのプライベートキーによって署名
4. デバイスは署名を検証する
5. デバイスはエンドポイントと署名のペアを今後の検証のために保存する。

6. クライアント― claim, enable
    a. 両方のメッセージはクライアントのプライベートキーによって署名
7. デバイスの使用
    a. デバイスサービスはイベントをクライアントに送信(デバイスのプライベートキーで署名)
8. クライアントの設定変更として、エンドポイント/証明テーブルに設定を保存する。
    a. デバイス宛てのすべてのメッセージはクライアントによって署名
9. クライアントはデバイスを disable し、release する
    a. すべてのメッセージはクライアントプライベートキーで署名
10. クライアントはデバイスをクローズする
    a. メッセージはクライアントのプライベートキーで署名
    b. デバイスはクライアントのエンドポイントと署名を検証

3.2 シンプル ユースケース

3.2.1 クライアントサービスはスキャナデバイスサービスを取得し使用する

Point-of-Serviceソリューションのクライアントサービスは店員が商品をスキャン可能にする。このサービスは取引を処理するために商品を識別するためにスキャナでバスサービスを取得し使用する。クライアントサービスは取引が終了した後スキャナデバイスサービスをrelaseおよびクローズする。

3.2.1.1 SOA Main Flow Description

1. UDDI のセットアップ ―スキャナデバイスサービスとクライアントサービスの認証パラメータ
2. スキャナデバイスの電源を入れる
3. スキャナデバイスの初期化
    - スキャナのセットアップ - アドレス、ケイパビリティ、名称、UDDI ロケーション
    - UDDI への登録 ―自己認証、WSDL 実装
        i. メッセージはスキャナのプライベートキーで署名
        ii. UDDI は署名を有効化
4. クライアントサービスがロードされ実行される.
    - クライアントサービスのセットアップ―アドレス、ケイパビリティ、名称、UDDI ロケーション
    - UDDI へスキャナデバイスのリストを問い合わせる―自己認証
        i. メッセージはクライアントのプライベートキーで署名
5. UDDI はデバイスのリストを返す- クライアントサービスは接続先のスキャナを選択

6. クライアントサービスはスキャナデバイスサービスをオープンし、イベントのための登録を行う。

- ```
  <?xml version="1.0" encoding="http://schemas.xmlsoap.org/soap/envelope/" standalone="no"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:ns3="http://www. www.nrf-arts.org/UnifiedPOS/Scanner/">
    <soapenv:Header>
        <wsa:To>http://scanner.WS-POS-example.nrf-arts.org:8081/services/Scanner-WS</wsa:To>
        <wsa:MessageID>urn:uuid:C5C856C8175724AD9E1195687902183</wsa:MessageID>
        <wsa:Action>http://www.nrf-arts.org/UnifiedPOS/Scanner/Open</wsa:Action>
    </soapenv:Header>
    <soapenv:Body>
        <ns3:Open>
            <ns3:EndpointAddress>http://192.168.1.1:8082/axis2/services/Client-WS</ns3:EndpointAddress>
        </ns3:Open>
    </soapenv:Body>
  </soapenv:Envelope>
  ```

  - メッセージはクライアントのプライベートキーで署名

7. スキャナは署名を検証する
   スキャナはエンドポイントと署名のペアを格納する。
8. クライアントサービス―スキャナデバイスの claim, enable (両メッセージはクライアントのプライベートキーで署名)

- ```
  <?xml version="1.0" encoding="http://schemas.xmlsoap.org/soap/envelope/" standalone="no"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:ns3="http://www.nrf-arts.org/UnifiedPOS/Scanner/">
     <soapenv:Header>
         <wsa:To>http://scanner.WS-POS-example.nrf-arts.org:8081/services/Scanner-WS</wsa:To>
         <wsa:MessageID>urn:uuid:C5C856C8175724AD9E1195687902183</wsa:MessageID>
         <wsa:Action>http://www.nrf-arts.org/UnifiedPOS/Scanner/Claim</wsa:Action>
     </soapenv:Header>
     <soapenv:Body>
         <ns3:Claim>
             <ns3:Timeout>1000</ns3:Timeout>
         </ns3:Claim>
     </soapenv:Body>
  </soapenv:Envelope>
  ```

- ```
  <?xml version="1.0" encoding="http://schemas.xmlsoap.org/soap/envelope/" standalone="no"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:ns3="http://www.nrf-arts.org/UnifiedPOS/Scanner/">
     <soapenv:Header>
          <wsa:To>http://scanner.WS-POS-example.nrf-arts.org:8081/services/Scanner-WS</wsa:To>
          <wsa:MessageID>urn:uuid:C5C856C8175724AD9E1195687902183</wsa:MessageID>
          <wsa:Action>http://www.nrf-arts.org/UnifiedPOS/Scanner/SetDeviceEnabled</wsa:Action>
     </soapenv:Header>
     <soapenv:Body>
          <ns3:SetDeviceEnabled>
               <ns3:DeviceEnabled>true</ns3:DeviceEnabled>
          </ns3:SetDeviceEnabled>
     </soapenv:Body>
  </soapenv:Envelope>
  ```

9. クライアントによる設定変更は、クライアントごとに格納される。
10. 店員が商品をスキャンする
11. スキャナデバイスは WS-POS イベントをクライアントに送信する(クライアントコールバックメソッドを XMLPOS を用いて呼び出す) (スキャナのプライベートキーで署名)

    ```
    <?xml version="1.0" encoding="http://schemas.xmlsoap.org/soap/envelope/" standalone="no"?>
    <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:ns3="http://www.nrf-arts.org/UnifiedPOS/ScannerEvents/">
        <soapenv:Header>
            <wsa:To>http://client.WS-POS-example.nrf-arts.org/services/Client-WS</wsa:To>
            <wsa:ReplyTo>
            <wsa:Address>http://www.w3.org/2005/08/addressing/none</wsa:Address>
            </wsa:ReplyTo>
            <wsa:MessageID>urn:uuid:548181C361F2A1EE5C1195691857095</wsa:MessageID>
            <wsa:Action>http://www.nrf-arts.org/UnifiedPOS/ScannerEvents/DataEvent</wsa:Action>
        </soapenv:Header>
        <soapenv:Body>
            <ns3:DataEvent>
                <ns3:Source>http://scanner.WS-POS-example.nrf-arts.org/services/Scanner-WS</ns3:Source>
                <ns3:EventID>1234</ns3:EventID>
                <ns3:TimeStamp>2011-01-09T12:34:56</ns3:TimeStamp>
                <ns3:Status>0</ns3:Status>
            </ns3:DataEvent>
        </soapenv:Body>
    </soapenv:Envelope>
    ```

    クライアントサービスはイベントを受信する。
    ```

12. クライアントサービスはスキャナサービスを disable、release、close する。

```
<?xml version="1.0" encoding="http://schemas.xmlsoap.org/soap/envelope/" standalone="no"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:ns3="http://www.nrf-arts.org/UnifiedPOS/Scanner/">
 <soapenv:Header>
 <wsa:To>http://scanner.WS-POS-example.nrf-arts.org/services/Scanner-WS</wsa:To>
 <wsa:MessageID>urn:uuid:C5C856C8175724AD9E1195687902183</wsa:MessageID>
     <wsa:Action>http://www.nrf-arts.org/UnifiedPOS/Scanner/SetDeviceEnabled</wsa:Action>
 </soapenv:Header>
 <soapenv:Body>
      <ns3:SetDeviceEnabled>
          <ns3:DeviceEnabled>false</ns3:DeviceEnabled>
      </ns3:SetDeviceEnabled>
 </soapenv:Body>
</soapenv:Envelope>
<?xml version="1.0" encoding="http://schemas.xmlsoap.org/soap/envelope/" standalone="no"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:ns3="http://www.nrf-arts.org/UnifiedPOS/Scanner/">
<soapenv:Header>
 <wsa:To>http://scanner.WS-POS-example.nrf-arts.org/services/Scanner-WS</wsa:To>

      <wsa:MessageID>urn:uuid:C5C856C8175724AD9E1195687902183</wsa:MessageID>
      <wsa:Action>http://www.nrf-arts.org/UnifiedPOS/Scanner/Release</wsa:Action>
  </soapenv:Header>
  <soapenv:Body>
       <ns3:Release />
  </soapenv:Body>
</soapenv:Envelope>
```

13. クライアントサービスはスキャナをクローズし、イベントの登録を解除する(クライアントのプライベートキーで署名)

```
<?xml version="1.0" encoding="http://schemas.xmlsoap.org/soap/envelope/" standalone="no"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:ns3="http://www.nrf-arts.org/UnifiedPOS/Scanner/">
    <soapenv:Header>
         <wsa:To>http://scanner.WS-POS-example.nrf-arts.org/services/Scanner-WS</wsa:To>
         <wsa:MessageID>urn:uuid:C5C856C8175724AD9E1195687902183</wsa:MessageID>
          <wsa:Action>http://www.nrf-arts.org/UnifiedPOS/Scanner/Close</wsa:Action>
   </soapenv:Header>
   <soapenv:Body>
        <ns3:Close>
      <ns3:EndpointAddress>http://192.168.1.1:8082/axis2/services/Client-WS</ns3:EndpointAddress>
        </ns3:Close>
   </soapenv:Body>
 </soapenv:Envelope>
```

14. スキャナデバイスサービスは UDDI に対してデバイスの削除を通知。

15. スキャナデバイスの電源を切断

3.3 ユースケースカタログ

本ドキュメントでは、WS-POS 標準や、その導入により考えられる、新たなビジネスシーンをユースケースとして具体化し、カタログとして紹介するものである。

各ユースケースは、次世代 POS 研究会のビジネスシナリオ分科会参加メンバー企業各社が作成した。

3.4 スコープ

Contributor: セイコーエプソン株式会社

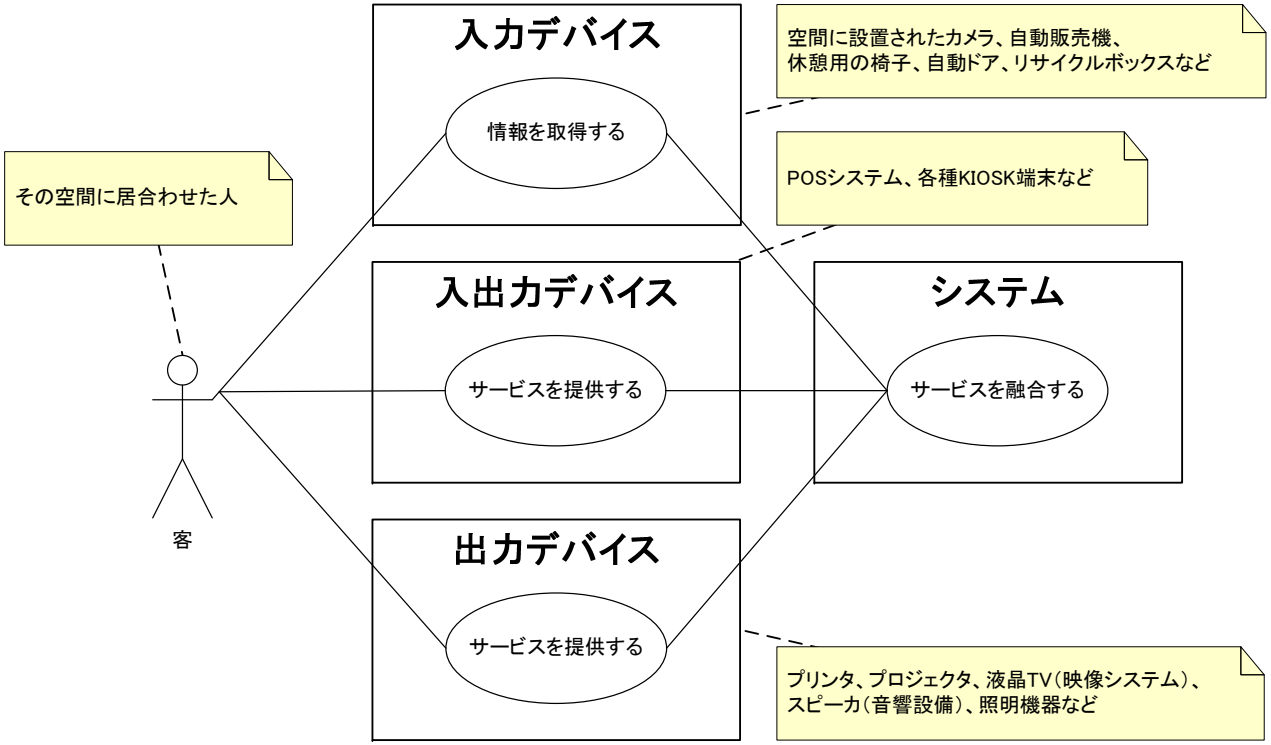
ここでは、WS-POS を基盤とした新しいビジネスシナリオ全般をカバーする、ハイレベルなユースケースを紹介する。以降に続くユースケースは、ここに示す基本計の具象化、もしくは組み合わせである。

あらゆるデバイス・アプリケーションがサービス化した空間協力会社

ショッピングモールの各店舗のみならず、休憩所や共有スペースなど、あらゆる空間に配置された入力デバイス、出力デバイスおよび入出力デバイスがサービス化されている環境で、様々なデバイスを組み合わせて新しいサービスを提供するアプリケーション・プログラムを用意することで、様々なアイデアが顧客への新しいサービスを提案できる空間を生み出す。

エンティティの定義:

お客様	その空間に居合わせた人
入力デバイス	空間に設置されたカメラ、自動販売機、休憩用の椅子、自動ドア、リサイクルボックスなど
出力デバイス	プリンタ、プロジェクタ、液晶 TV（映像システム）、スピーカ（音響設備）、照明機器など
入出力デバイス	POS システム、各種 KIOSK 端末など
アプリーケーション・サービス	上記入出力デバイスを組み合わせてサービスを提供するアプリケーション・プログラム



High-Level Use Case

Use Case –		
Stakeholders		**Concerns**
お客様		お客様は、その空間において、快適な時間を過ごしたい。また、きめの細かい、より良いサービスを受けることで満足感を味わいたい。
店舗経営者		店舗経営者は、お客様に付加価値の高いサービスを提供することで、集客力を上げ、売上を伸ばしたい。
デバイスベンダー		デバイスベンダーは、デバイスの機能を簡単に使える手段を提供することで、本来想定していた機能を超えて、デバイスの付加価値を高める可能性を提供することで、売り上げを伸ばしたい。
ソフトウェアベンダー		ソフトウェアベンダーは、様々なデバイスの機能を自由に組み合わせて、新たな価値を提供することで、色々なアイデアを形にして新たな提案を店舗経営者に提供することで売り上げを伸ばしたい。
Actors	**Type**	**Description**
お客様		買い物、または暇つぶしでその場に訪れた人。
入力デバイス		監視カメラなどの入力デバイスのみならず、自動ドアなどに取りつけられたセンサーなどの情報も入力デバイスとして想定できる。また、自動販売機での販売情報がリアルタイムで管理できる場合は入力デバイスとして利用できる。
出力デバイス		KIOSK 端末に設置されたプリンタ等の出力デバイスのみならず、プロジェクタや液晶 TV 等の映像出力デバイスや、スピーカを含む音響出力デバイス、さらにはソフトウェアにより制御可能な照明機器や空調機器なども出力デバイスとして利用できる。
入出力デバイス		POS 端末や KIOSK 端末など入力デバイスと出力デバイスの両方を持つものは、それぞれ入力デバイス、出力デバイスとして利用できる。
アプリケーション・プログラム		各入力デバイス、出力デバイス、入出力デバイスがサービスとして利用可能であれば、入力デバイスからの情報を元に出力デバイスを制御する事で、アイデア次第で様々なサービスを提供可能となる。

主要な正常シナリオ**:**

ラベル	シナリオ
1.	**お客様**は、**入力デバイス**から情報を入力する（意識して入力操作をするとは限らない）
1.1.	**入力デバイス**は、**アプリケーション・プログラム**に入力情報を配信する
1.2.	**アプリケーション・プログラム**は、入力情報を分析して出力情報を準備する
1.3	**アプリケーション・プログラム**は、出力情報を**出力デバイス**に送信する
1.4	**出力デバイス**は、出力情報を**お客様**に配信する
1.5	**お客様**は、**出力デバイス**から出力情報を受取る（意識して出力情報を受取るとは限らない）

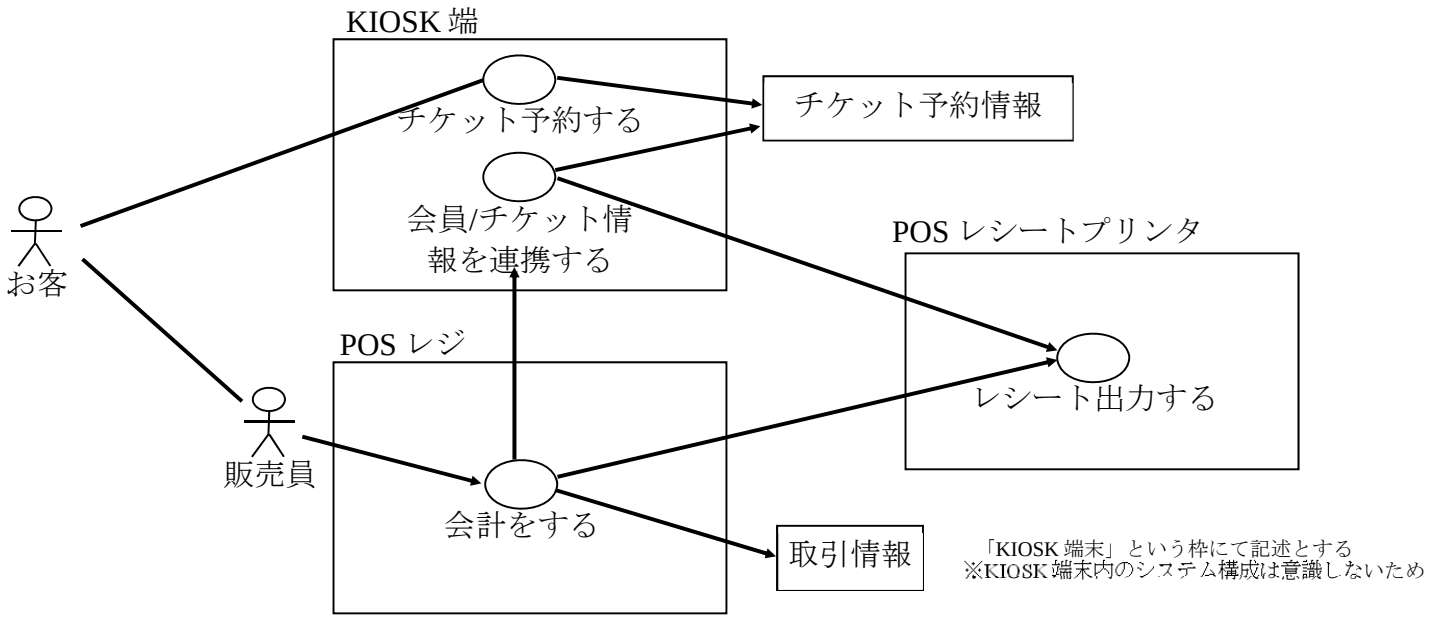
3.5 サブスコープ ：店内 KIOSK と POS（デバイス）連動

Contributor: ヴィンキュラムジャパン株式会社

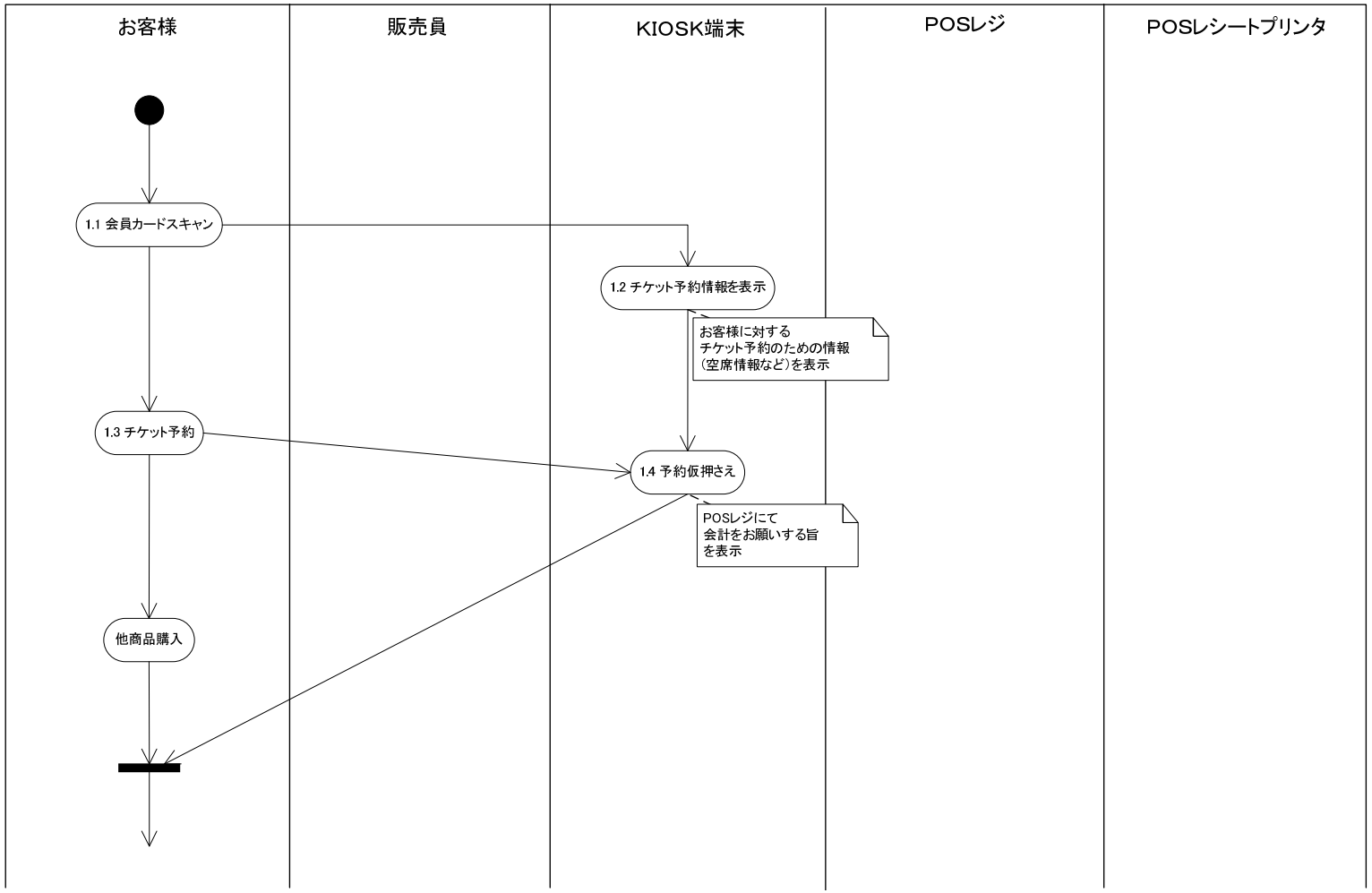
お客様が、店内 KIOSK で、各種情報（映画館や飛行機の座席予約情報など）を見てチケット予約した場合、その内容を LAN 上にある POS プリンターで出力。

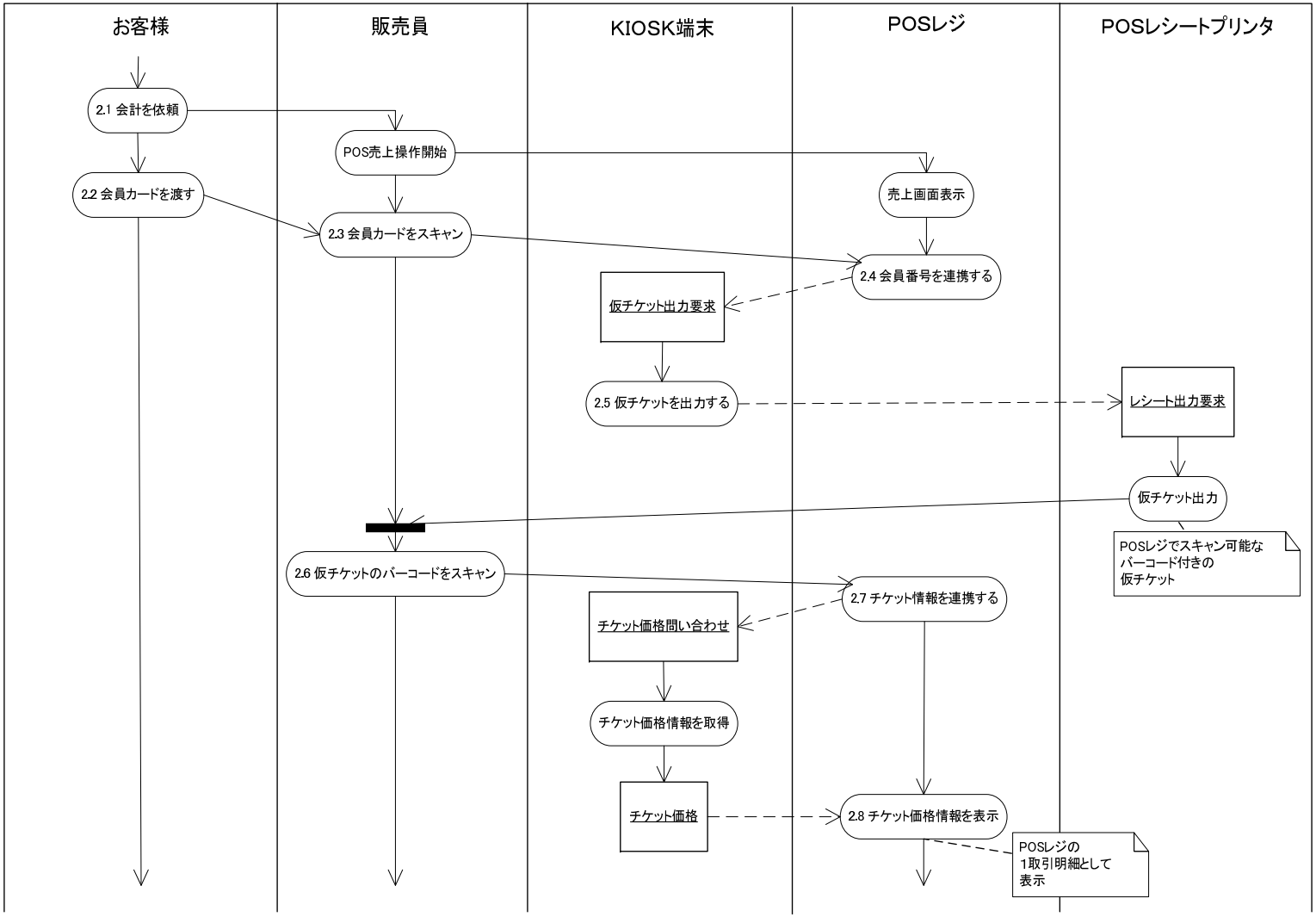
エンティティの定義:

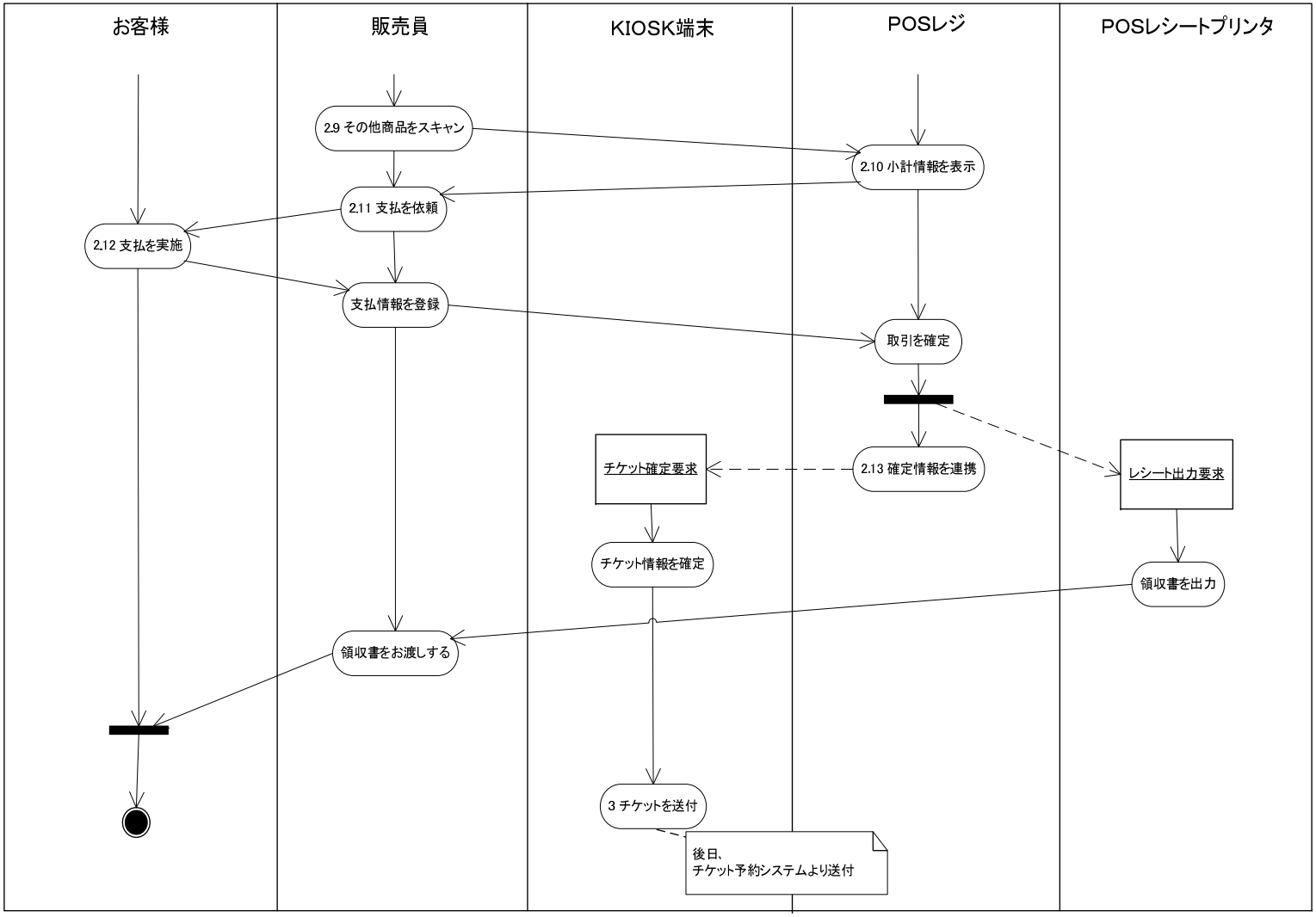
- お客　お客は、チケット予約や商品購入を行う人である。
- KIOSK 端末　KIOSK 端末は、各種情報の表示やチケット予約を管理する端末である。
- POS レジ　POS レジは、店内商品の会計を行うための端末である。
- 販売員　販売員は、POS レジの操作を行う人である。
- POS レシートプリンタ　POS レシートプリンタは、取引領収書やチケット予約レシートを出力するための端末である。
- チケット予約情報　チケット予約情報は、KIOSK 端末より管理される座席予約情報などの管理情報である。
- 取引情報　取引情報は、POS レジより管理される会計に対する実績情報である。



Activity Diagram







3.6 サブスコープ：販売員支援端末と POS デバイスとの連携

Contributor: 東芝テック株式会社

アパレル小売店において、お客様に対して、販売員が以下のサービスを提供することを支援するシステム。

- 商品在庫情報や商品に関する詳細情報提供
- 新作の入荷情報やアパレルのお勧めコーディネート情報、売れ筋情報などの商品関連情報提供
- お客様の商品購入に対する、POS で会計する前の前捌き機能

支援システムは、

- 商品情報の検索/参照機能や購入処理機能を提供するバックエンドサービス（Web サービスアプリケーション等で構築）
- 各販売員が所持し、該当商品の商品コードの読み取り（バーコードや RFID を使用）やバックエンドサービスと連携してお客様に情報を提供する為の、販売員支援端末
- 商品情報やレシートをプリントするためのプリンタを持つ POS 端末からなる。

エンティティの定義:

- 販売員支援端末

販売員支援端末とは、店員が保持し、お客様に対して商品の紹介を行ったり、お客様からの問い合わせで在庫の検索を行ったりするための、店員用店舗内情報端末（ＰＤＡやタブレットＰＣ）であり、販売員支援システムのクライアントとして使用する端末であり、商品コードを読み取るためのバーコードリーダやＲＦＩＤリーダデバイスを有する。

- レシートプリンタ

ＰＯＳターミナルに接続されるプリンタであり、通常は会計時にレシートや領収書等を印字する。

- POS ターミナル:

  販売員がお客様に対して商品を売り上げる場面で使用する端末である。販売員がお客様から預かった商品を本端末に登録すると、販売価格の合計が表示されるので、お客様はその価格分支払う。一般的な POS ターミナルは価格情報や値引き情報を扱うため、商品情報、および価格情報を扱うサーバと通信を行う。また、多くの POS 端末はレシートプリンタ、バーコードスキャナ、価格表示用の画面(販売員用、お客様用のタイプもあり)、POS キーボードを備える。

- 商品:

  お客様が購入する物品あるいはサービス。お客様は商品を購入するために対価として現金あるいは他の手段により決済を行う。

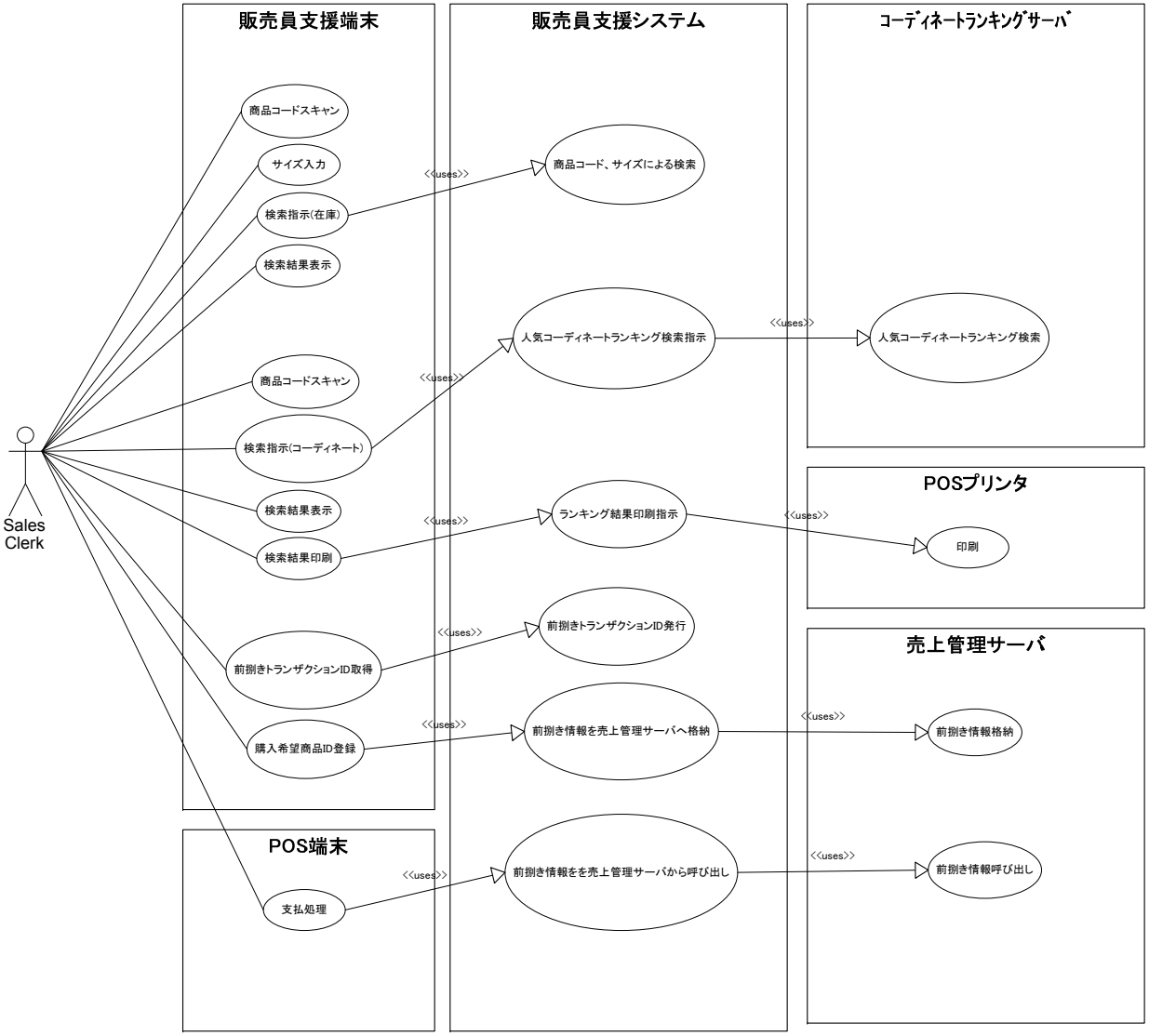
  システムからみた場合、商品は ID、名称、価格などの副次情報を持つ。.

- 在庫:

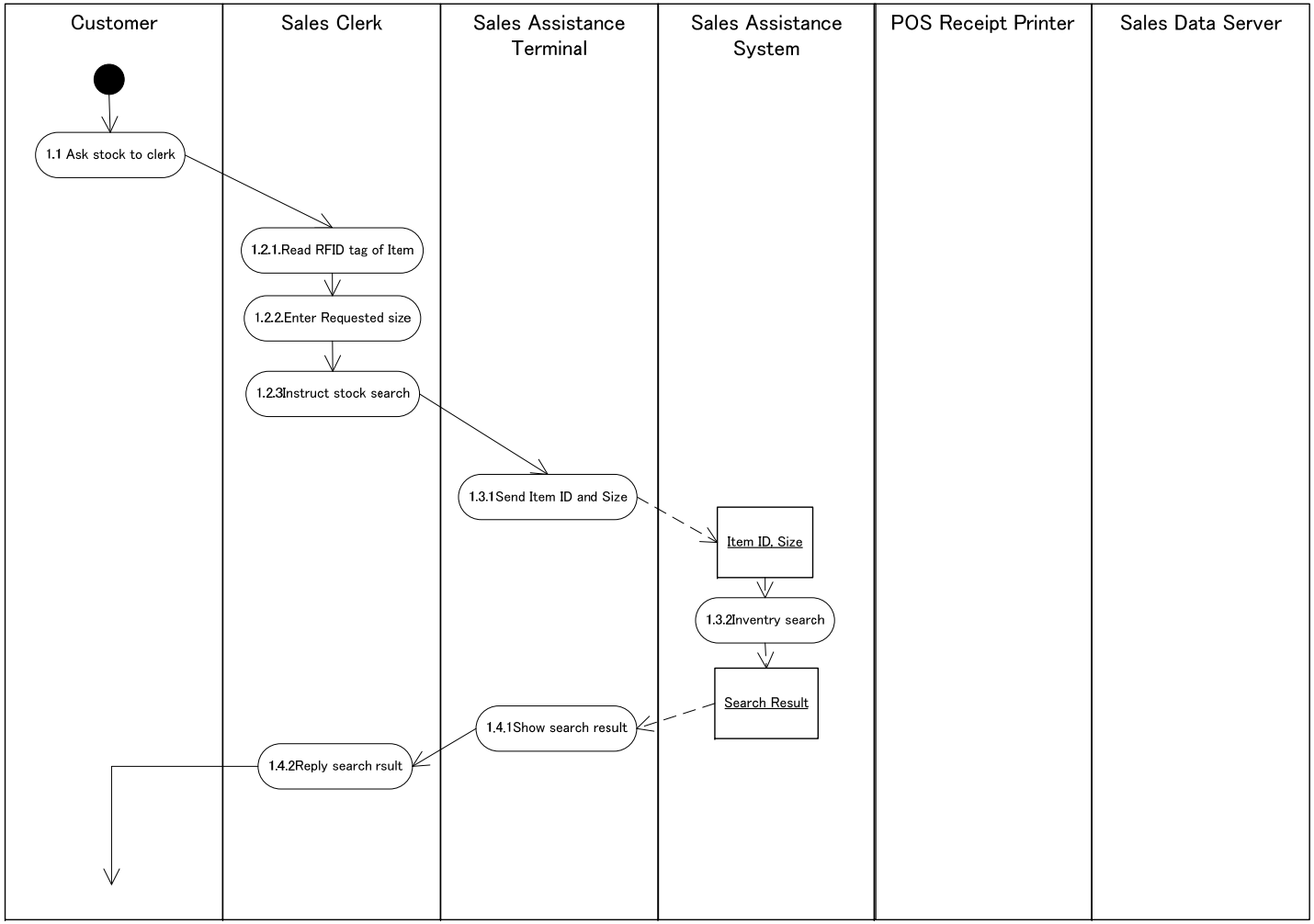
  店頭あるいはバックヤードや倉庫等に保持している商品群。

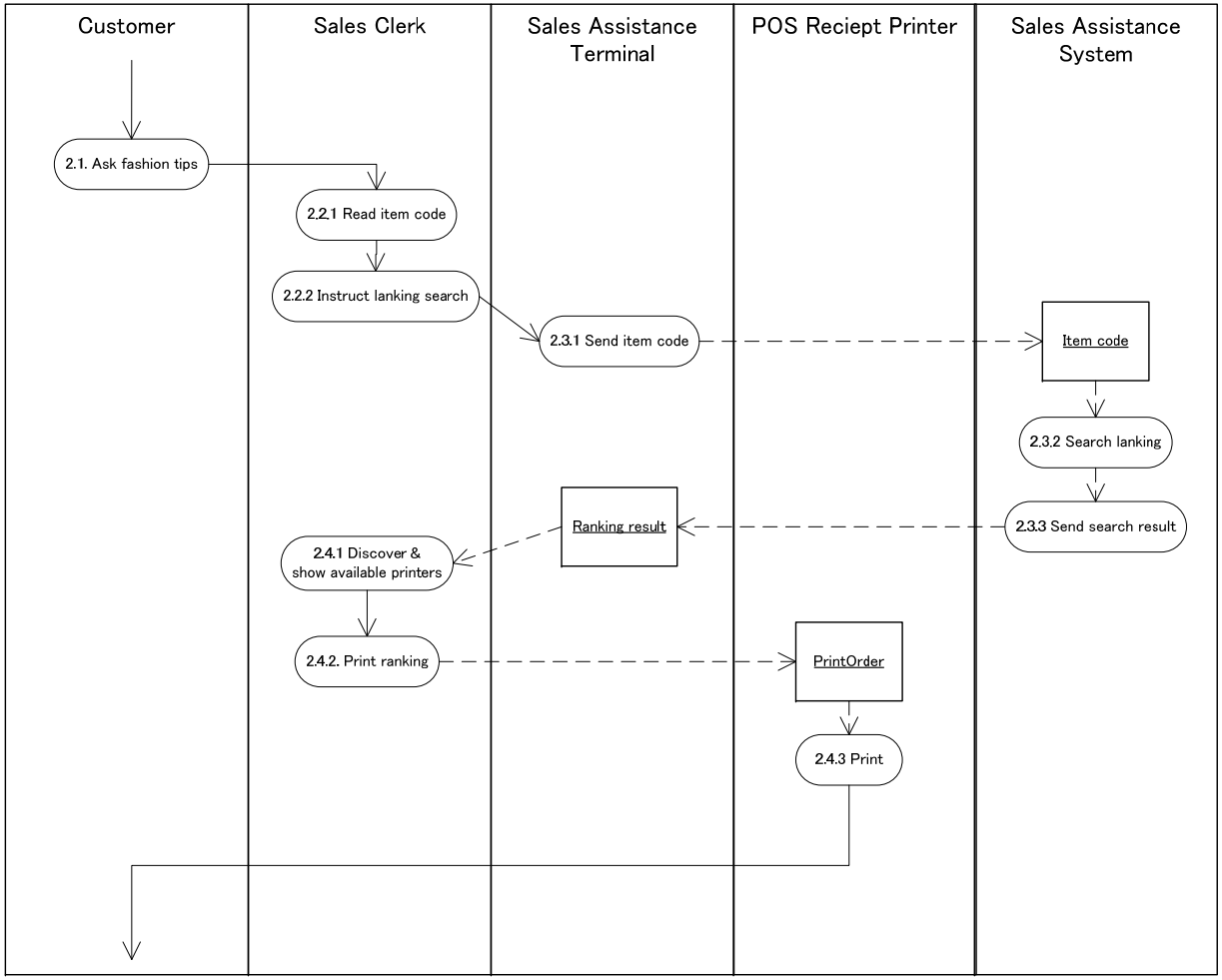
- コーディネートランキングなどの商品関連情報

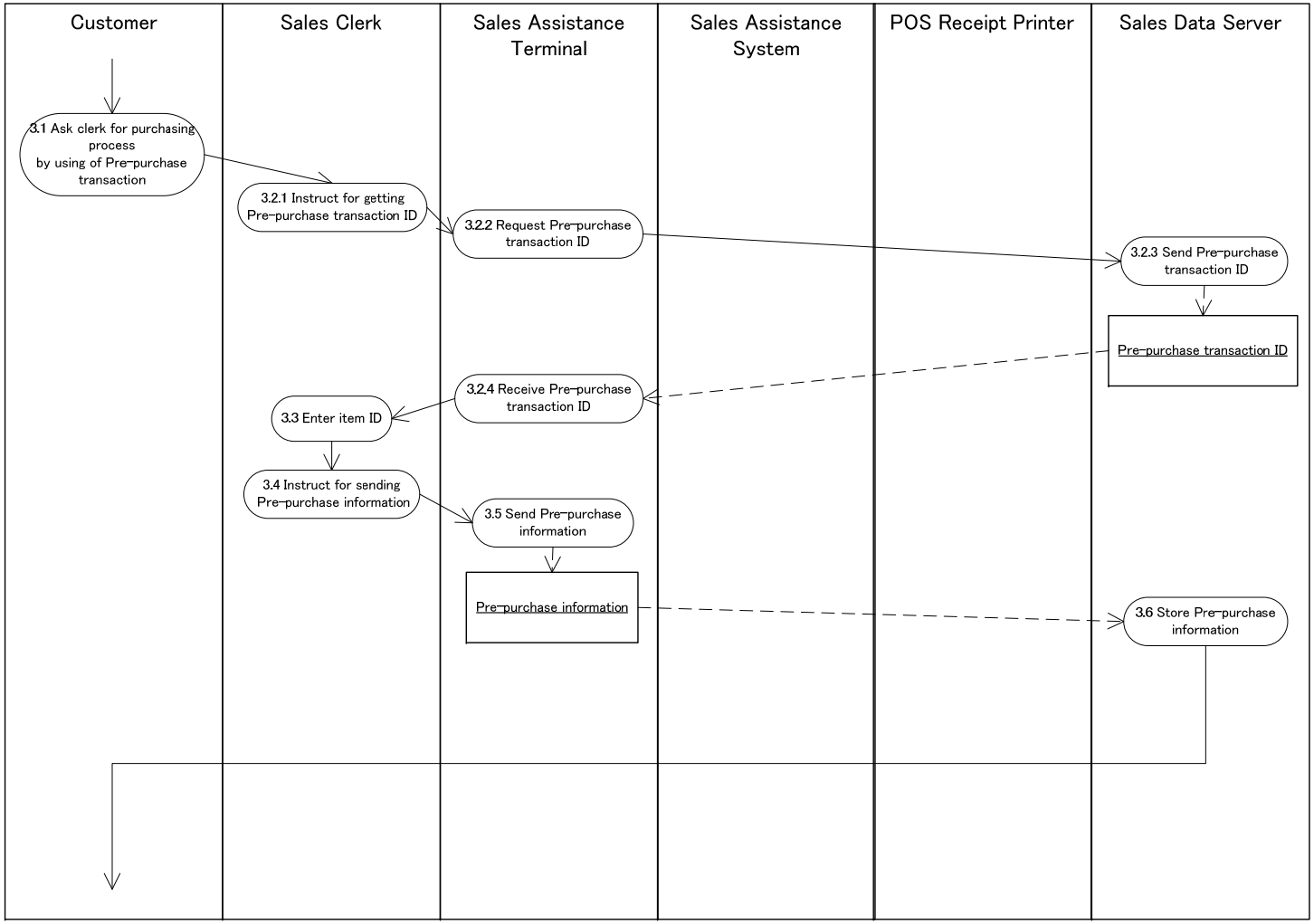
  お客様が商品購入時に参考とする情報。ランキングなどの情報はコーディネートの流行や人気度を表すものであり、お客さまにとっては購入時に参考となる。

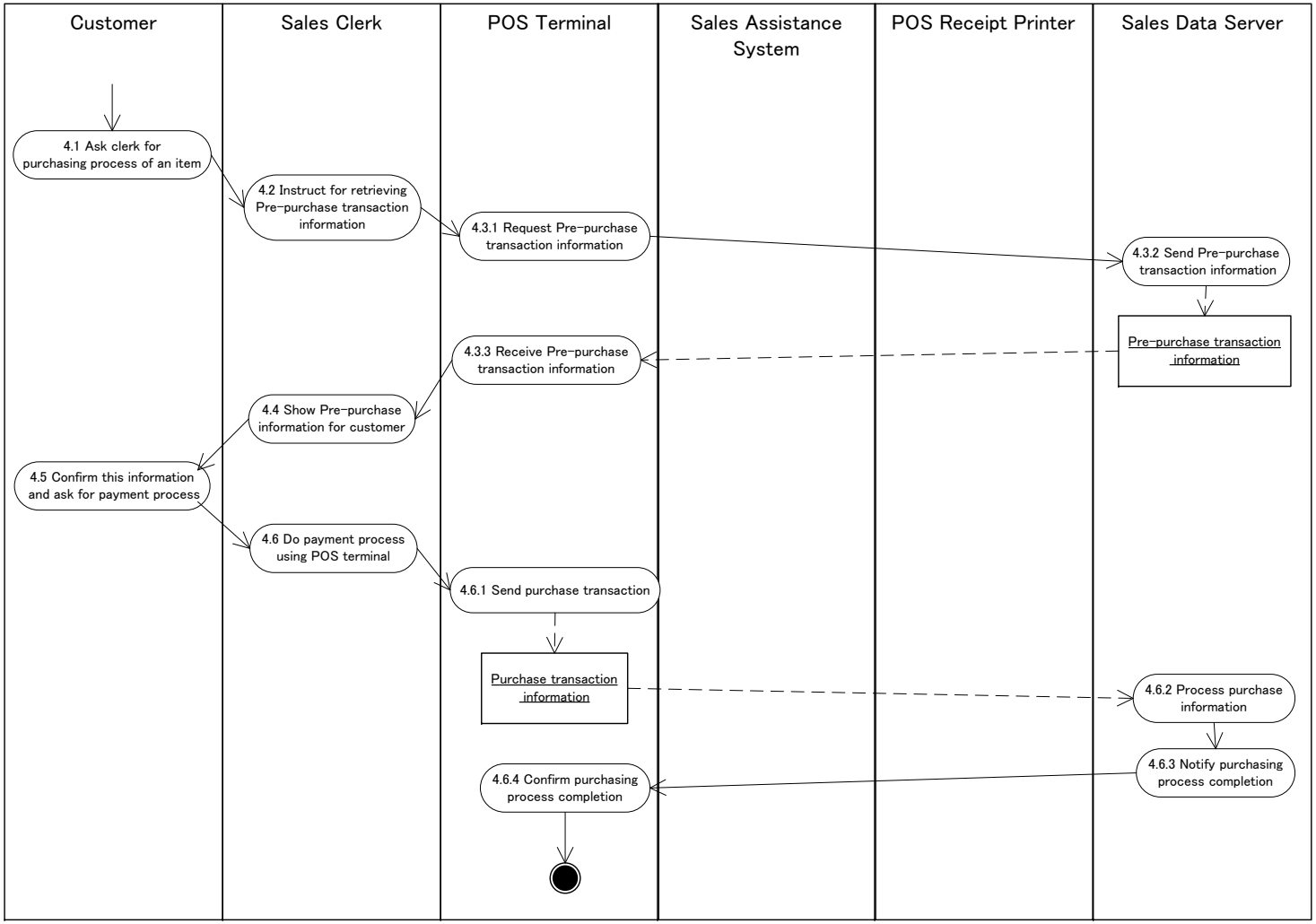


Activity Diagram









3.7　サブスコープ：複合商業施設における一括決済

Contributor: 株式会社ソリマチ技研

百貨店、ショッピングモールなどの複合商業施設で、各小売店舗毎に決済を行う代わりに、一括して、専用の決済コーナー、もしくは任意の小売店舗のＰＯＳで決済を行う。

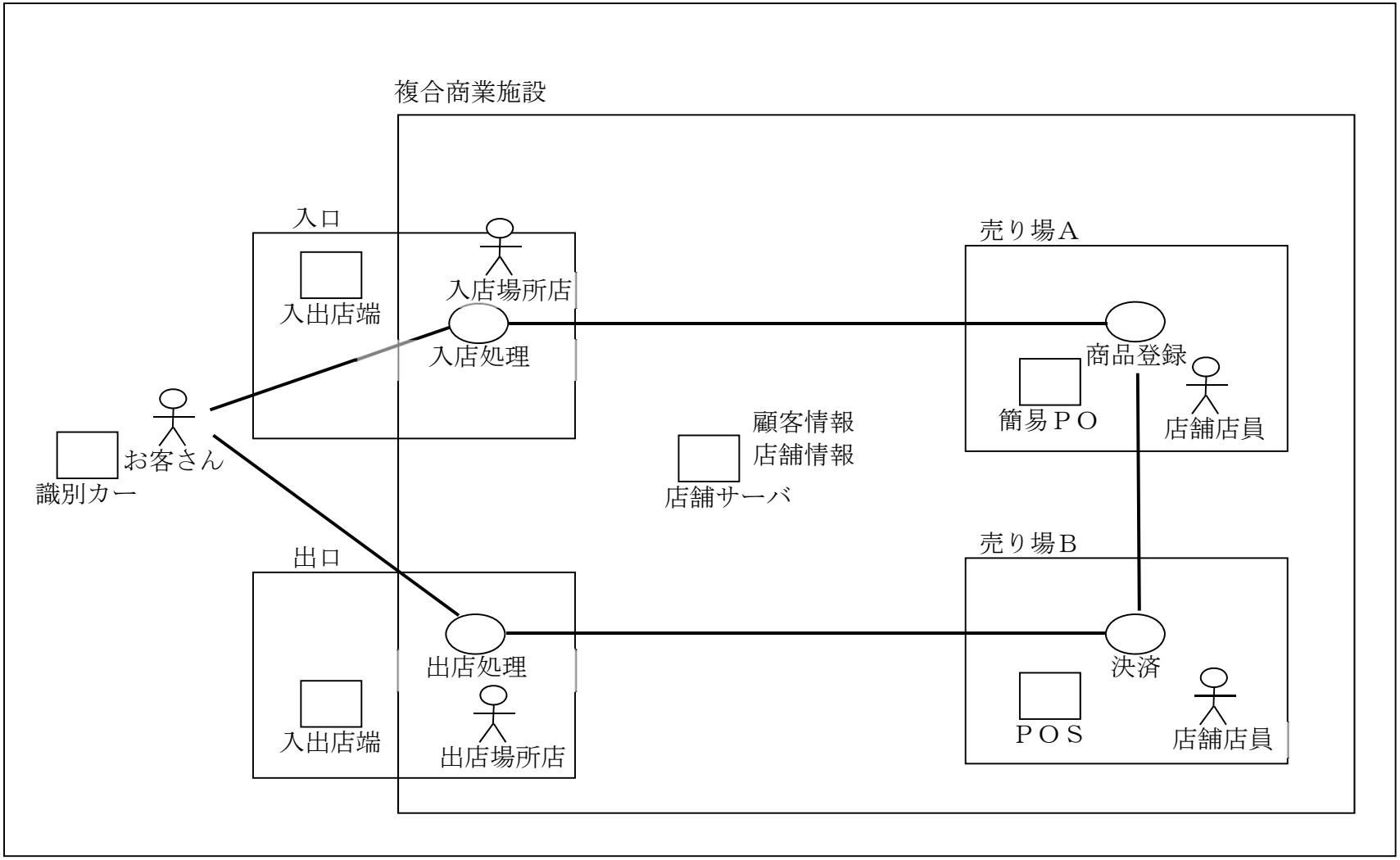
このサブスコープは、顧客が商品を持ち帰るか、それとも配送するかに依存しない。

配送する場合、任意の小売店舗にて住所を１度記入すれば、その商業施設内のすべての小売店舗で購入した商品を一括発送する。

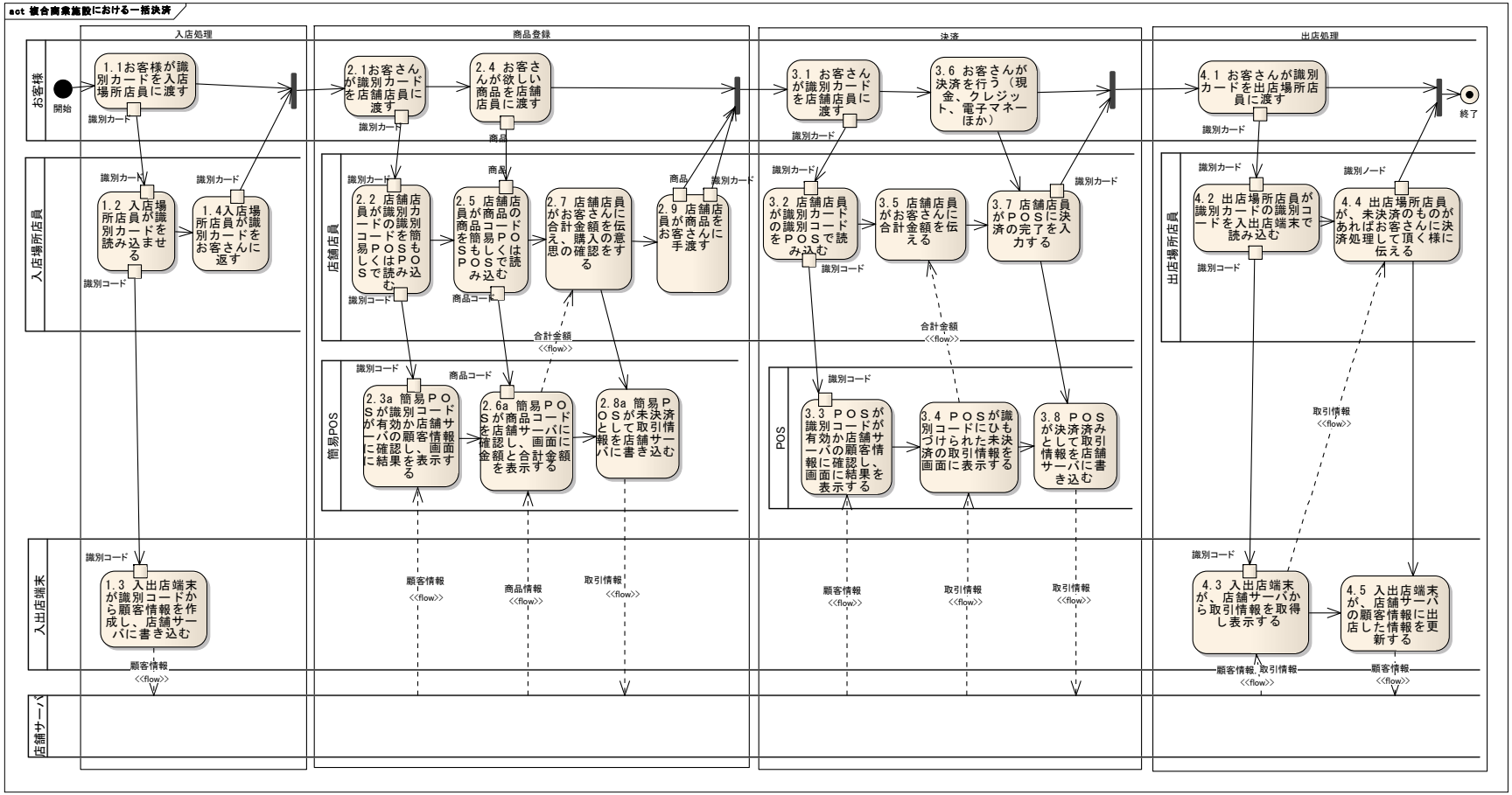
エンティティの定義:

分類	エンティティ	説明
顧客	お客さん	
店員	入店場所店員	複合商業施設の入口で入店処理を行う店員
	出店場所店員	複合商業施設の出口で出店処理を行う店員
	店舗店員	複合商業施設内の小売店舗で、商品登録処理、決済処理を行う店員
端末	識別カード	お客さんを識別するカード 会員カード、クレジットカード、即時発行カード
	入出店端末	入店場所、出店場所で、識別カードの読み込み、発行を行う端末 識別カードを読み込み可能なカードリーダを備えるハンディタイプを想定する

	簡易ＰＯＳ	商品登録処理だけが可能なＰＯＳ 識別カードを読み込み可能なカードリーダと商品バーコードを読み込むバーコードスキャナを備えた端末で、ハンディタイプを想定する
	ＰＯＳ	商品登録処理と決済処理が可能なＰＯＳ 識別カードを読み込み可能なっカードリーダと商品バーコードを読み込むバーコードスキャナを備え、決済を行うことができる据え置き型のＰＯＳ
	店舗サーバ	店舗内の簡易ＰＯＳ、ＰＯＳのデータを一元管理するサーバ
データ	顧客情報	識別カード、識別コードとお客さんの情報
	取引情報	識別コード、商品コード、決済状態からなる情報



Activity Diagram



3.8 サブスコープ：店内 KIOSK の装置監視と問題発生時の BackOffice 連動

Contributor: 富士通フロンテック株式会社

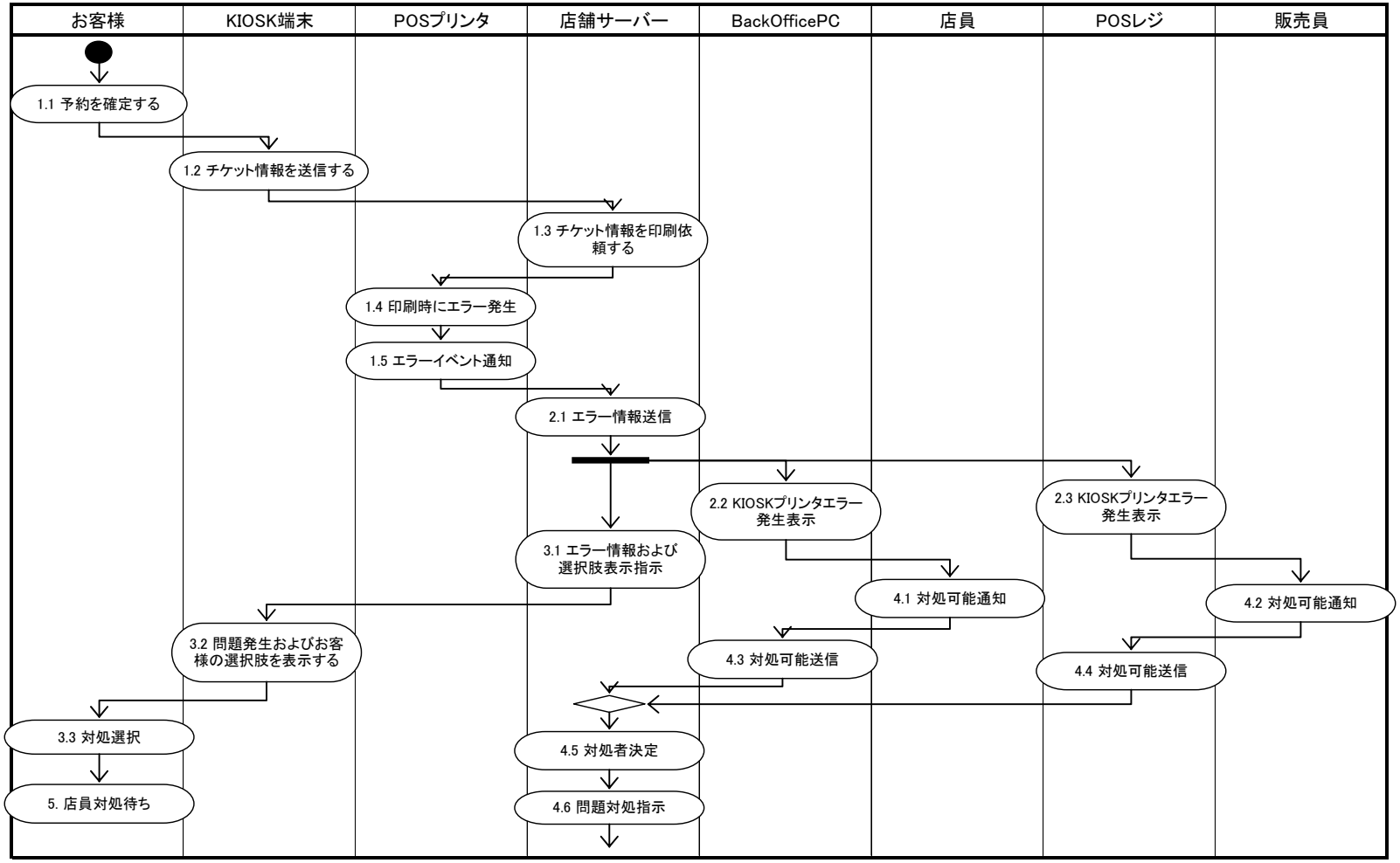
お客様が、店内 KIOSK で各種情報（映画館や飛行機の座席予約情報など）を見てチケット予約した後、決済のためのカード読み取りや結果を POS プリンタで出力しようとして問題発生した場合の対処。

エンティティの定義:

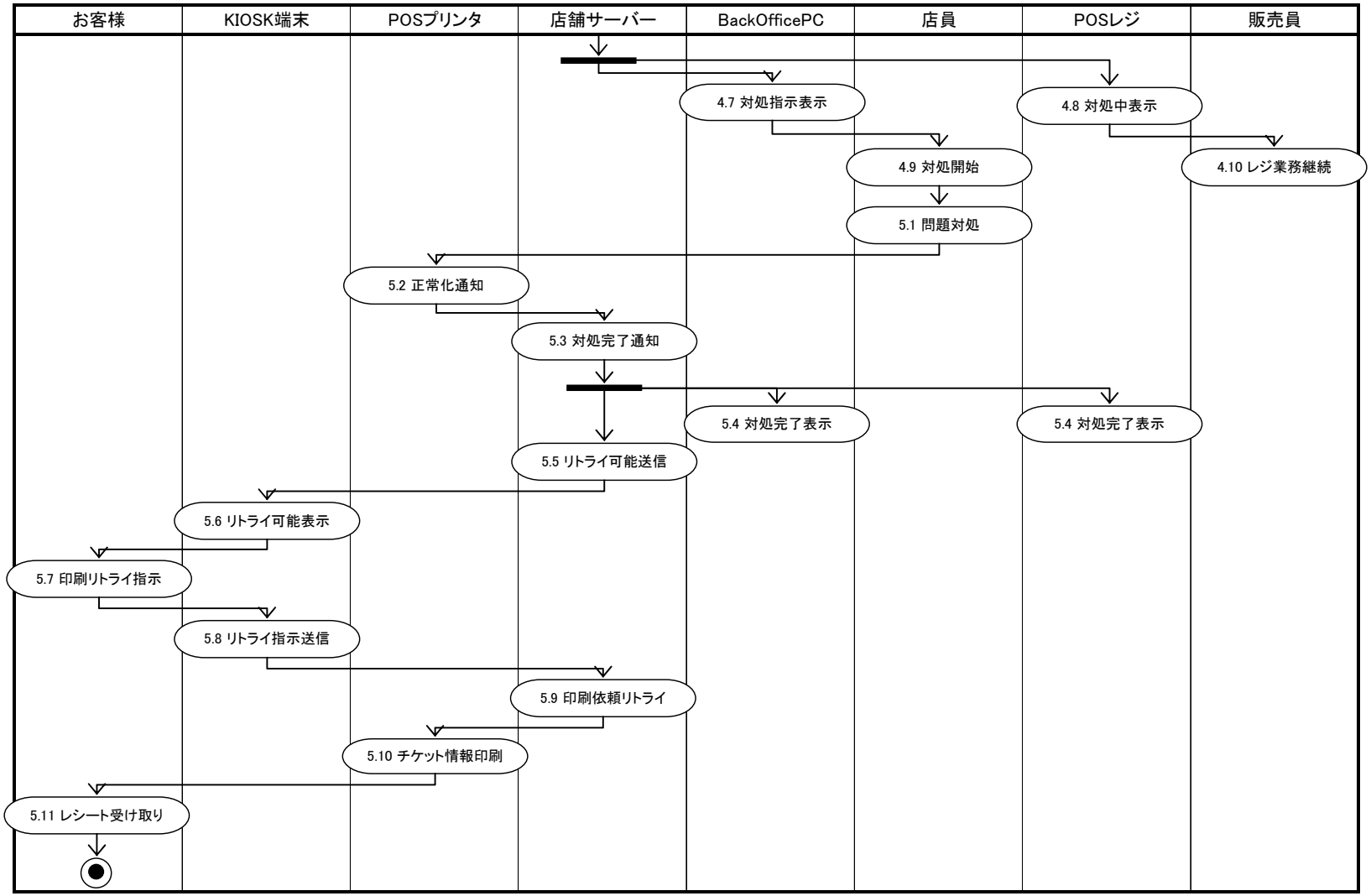
- お客　　お客は、チケット予約や商品購入を行う人である。
- KIOSK 端末１　KIOSK 端末は、各種情報の表示やチケット予約を管理する端末である。
- KIOSK 端末２　複数台並んでいてコーナーを構成している。
- KIOSK 端末３
- POS プリンタ　POS プリンタは、KISOK 端末の一部で取引領収書やチケット予約レシートを出力するための装置である。
- チケット予約情報　チケット予約情報は、KIOSK 端末より管理される座席予約情報などの管理情報である。
- 取引情報　取引情報は、POS レジより管理される会計に対する実績情報である。
- 店舗サーバ　店舗サーバは KIOSK 端末上のサービスを制御するサーバである。
- BackOfficePC　BackOfficePC は店舗のバックヤードで事務等の各種作業を行う端末である。
- POS レジ　POS レジは、商品の会計を行うための端末である。
- 販売員　販売員は、POS レジの操作を行う人である。
- 店員　店員は、POS レジ以外で作業を行う人である。
- 電子バリューR/W　電子バリューR/W は KIOSK 端末の一部で電子化された金銭やチケット等の入出力を行う装置である。

Activity Diagram

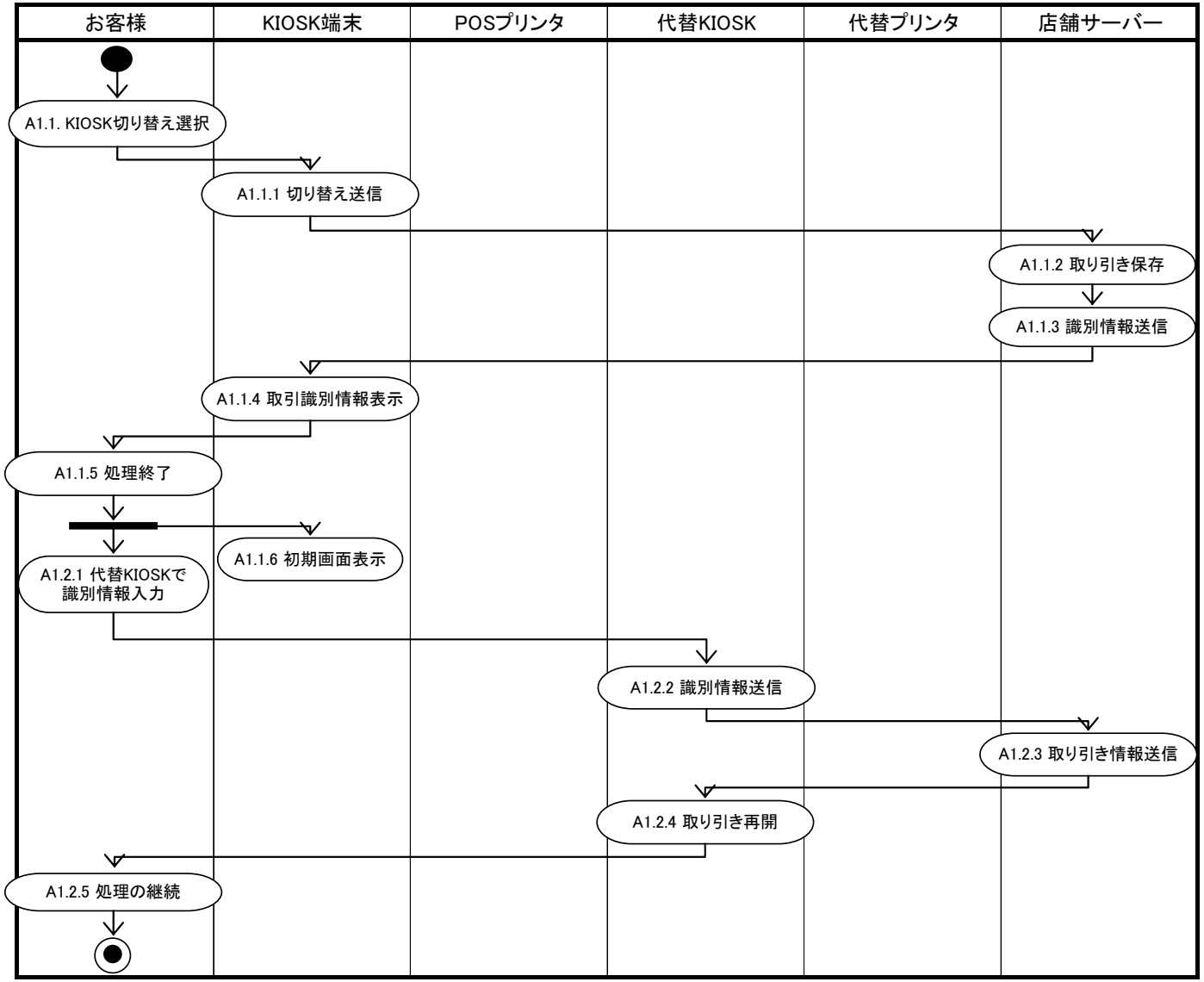
正常シナリオ 1/2



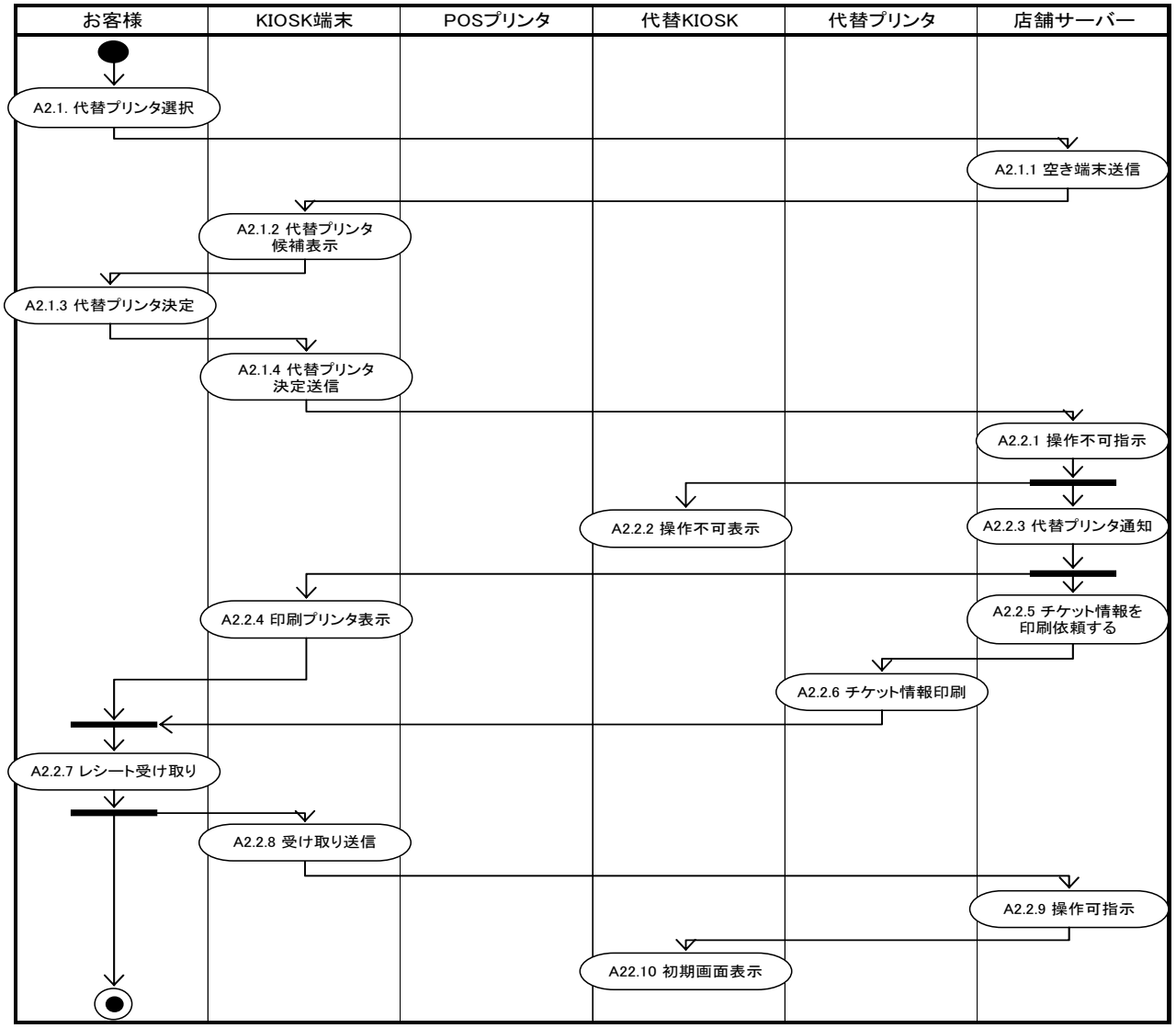
正常シナリオ 2/2



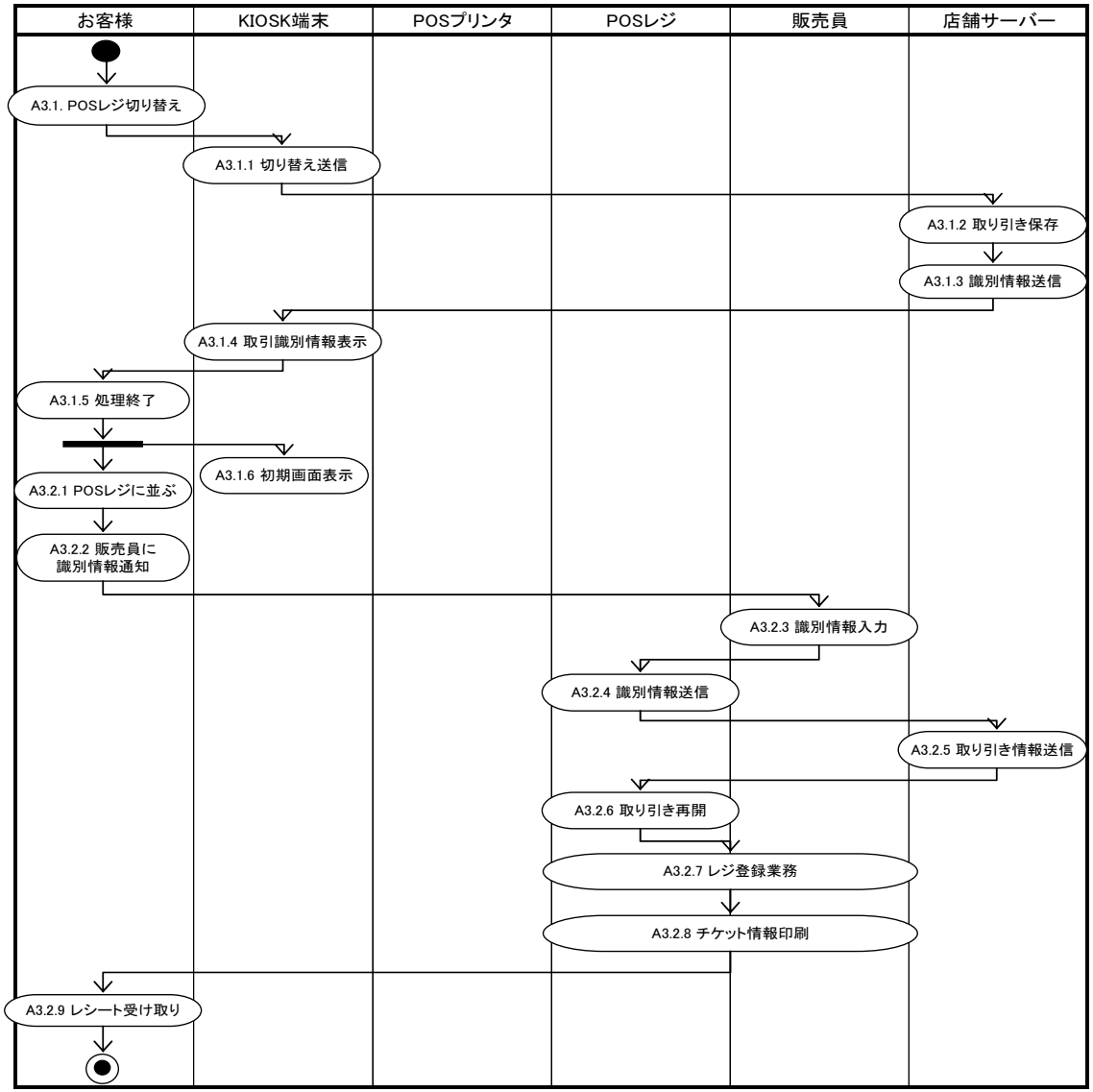
代替シナリオ1



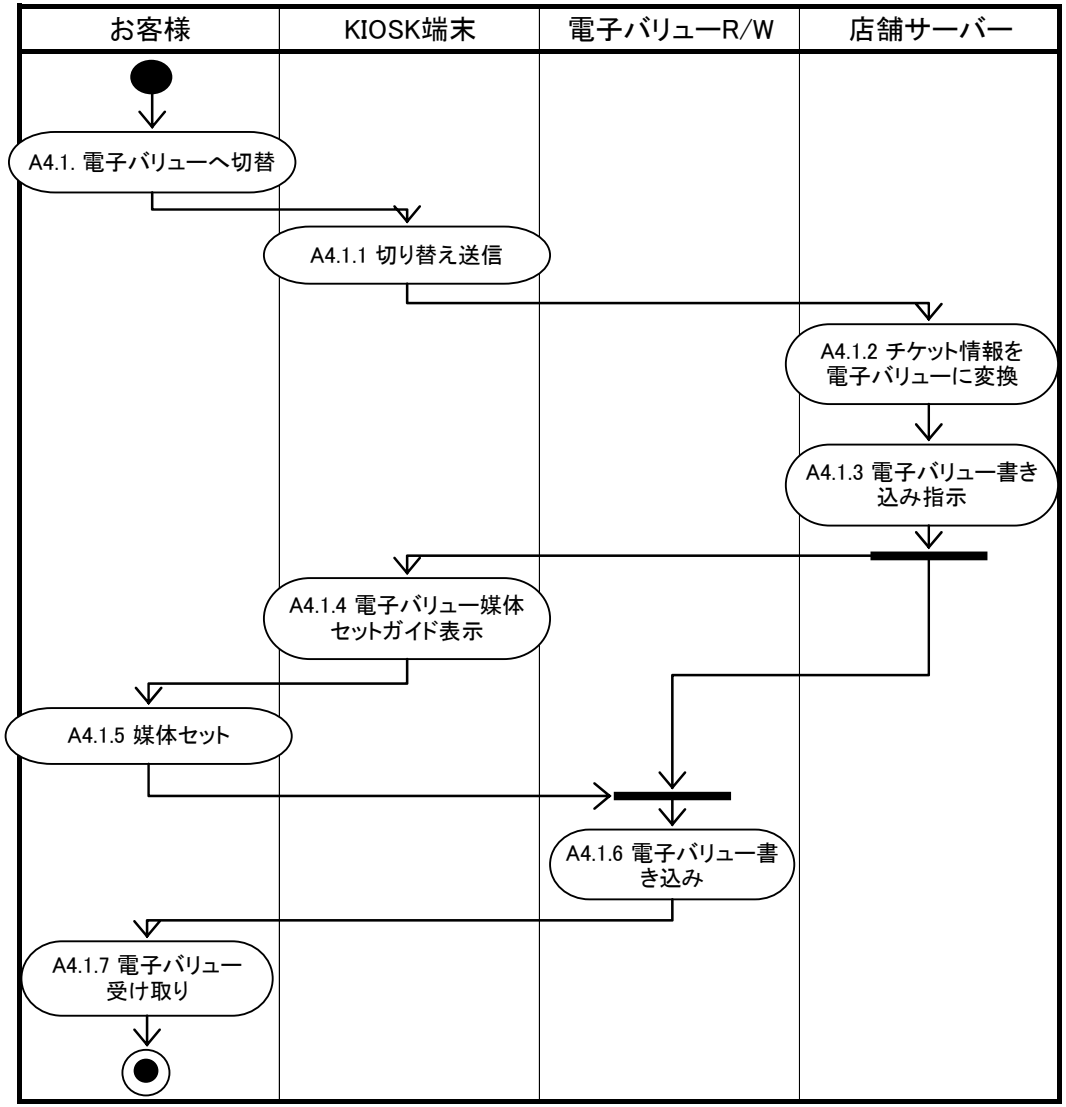
代替シナリオ2



代替シナリオ3



代替シナリオ4



3.9 サブスコープ：異業種間の連携を考慮した POS システム

Contributor: スター精密株式会社

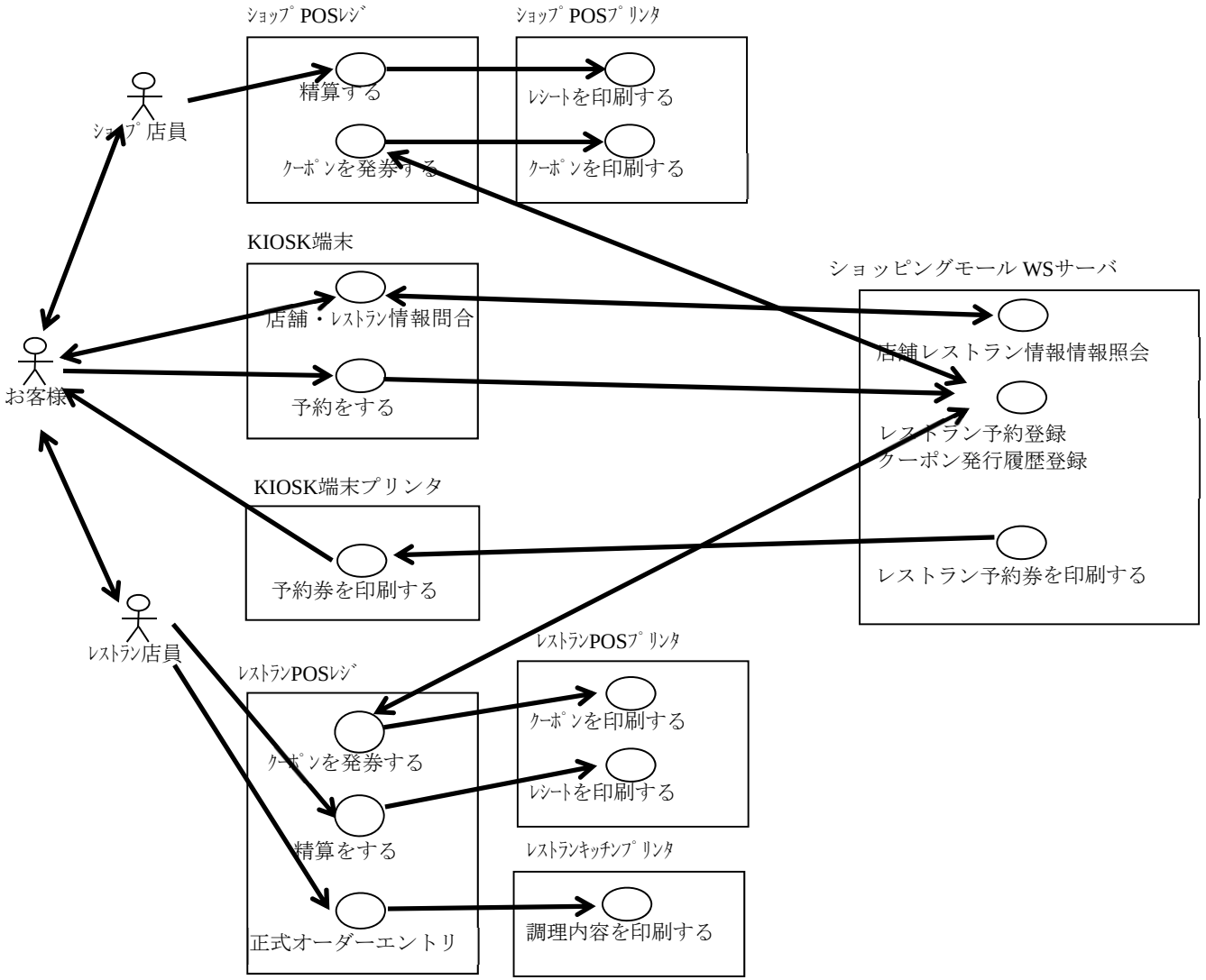
お客様が、小売店又は飲食店にて物品の購入もしくは食事を行う。

レジで精算時にレシート発行と同時に異業種(レストランなど)で利用可能なクーポンを発行する。

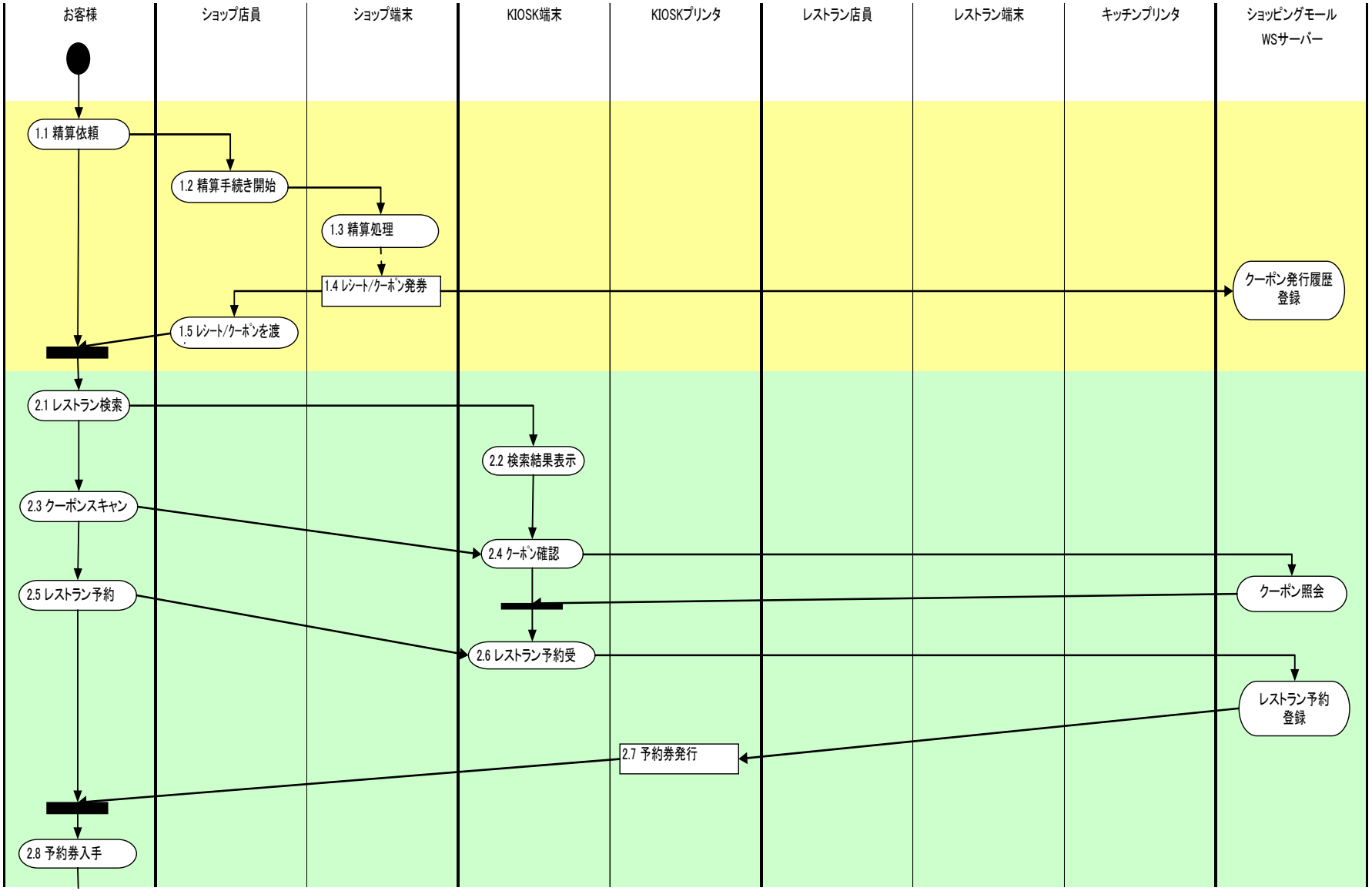
ショッピングモール内 KIOSK にて本クーポンを利用したレストラン予約などが行える。

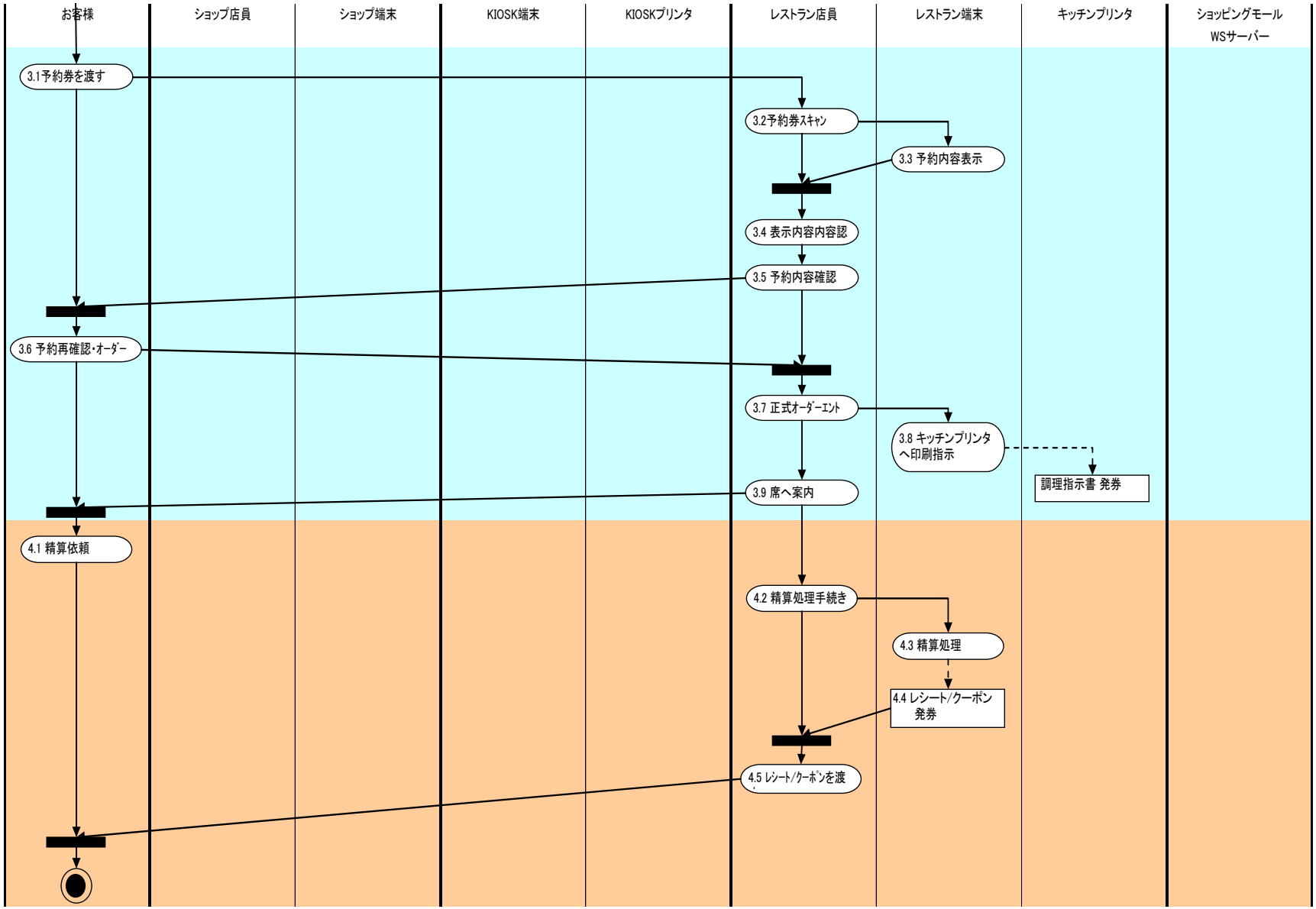
エンティティの定義:

お客様	商品購入や、レストランを利用する人。
ショップ店員	ショップでの POS レジ操作を行う人。
ショップ KIOSK 端末	ショッピングモール内のショッピングのお得な情報検索表示や、レストランの予約を行う為の端末。お客様が直接操作する。
ショップ情報システム	ショップ内の受発注、顧客管理を行う情報システム。
ショップ POS 端末	ショップ内に設置された POS 端末。操作者は、ショップ店員。
レストラン店員	レストランにてお客様の席の手配、料理のオーダーを受ける人。
レストラン情報システム	レストラン店内の受発注、座席管理などを行う情報システム。
レストラン端末	レストラン店内の受発注、座席管理などを行う情報システムの端末。操作者はレストラン店員。
レストランキッチンプリンタ	レストランのキッチンに配置されたプリンタ。料理すべき内容を適時印刷を行う。
Web Base Server System	Web Base でのデータ通信をサポートしたショッピングモール内共用の POS 相互調停システム



Activity Diagram



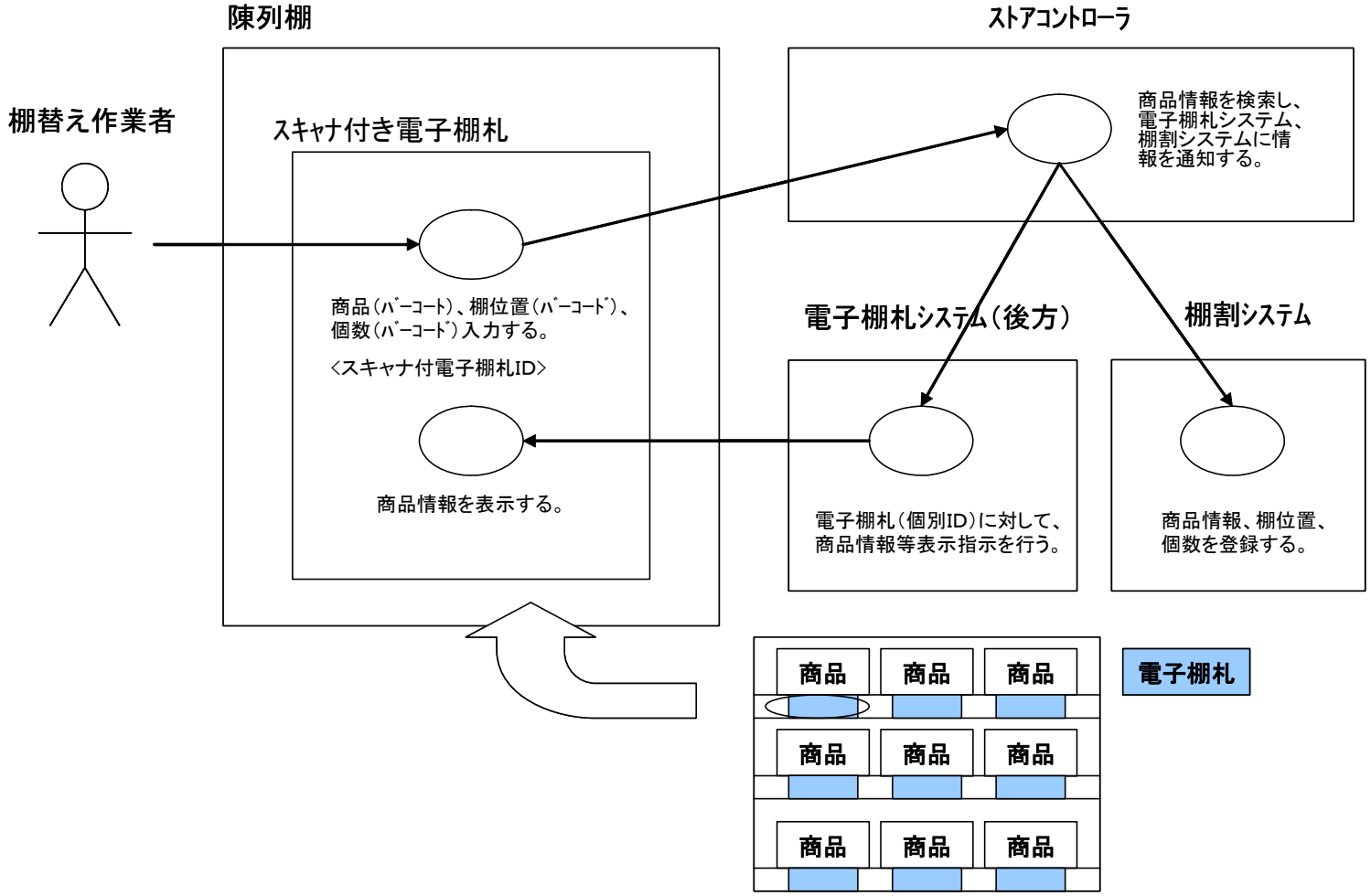


3.10 サブスコープ：陳列棚と後方システムとの連動

Contributor: オムロンソフトウェア株式会社

エンティティの定義（その１）：

- 棚替え作業者　：棚替え作業者は、棚割りをしながら該当商品を棚に陳列する人である。
- スキャナ付電子棚札などの商品　：スキャナ付電子棚札は、商品や棚の位置を示すバーコードを読み取るとともに商品価格情報を表示する機器である。（但し、現状は存在しないデバイスである）
- ストアコントローラ　：商品情報（商品名、売価、数量など）を管理するサーバである。
- 棚割システム　：陳列棚の棚割り状況を管理するサーバである。
- 電子棚札システム（後方）：スキャナ付電子棚札へ情報表示指示や表示状況を管理するサーバである。

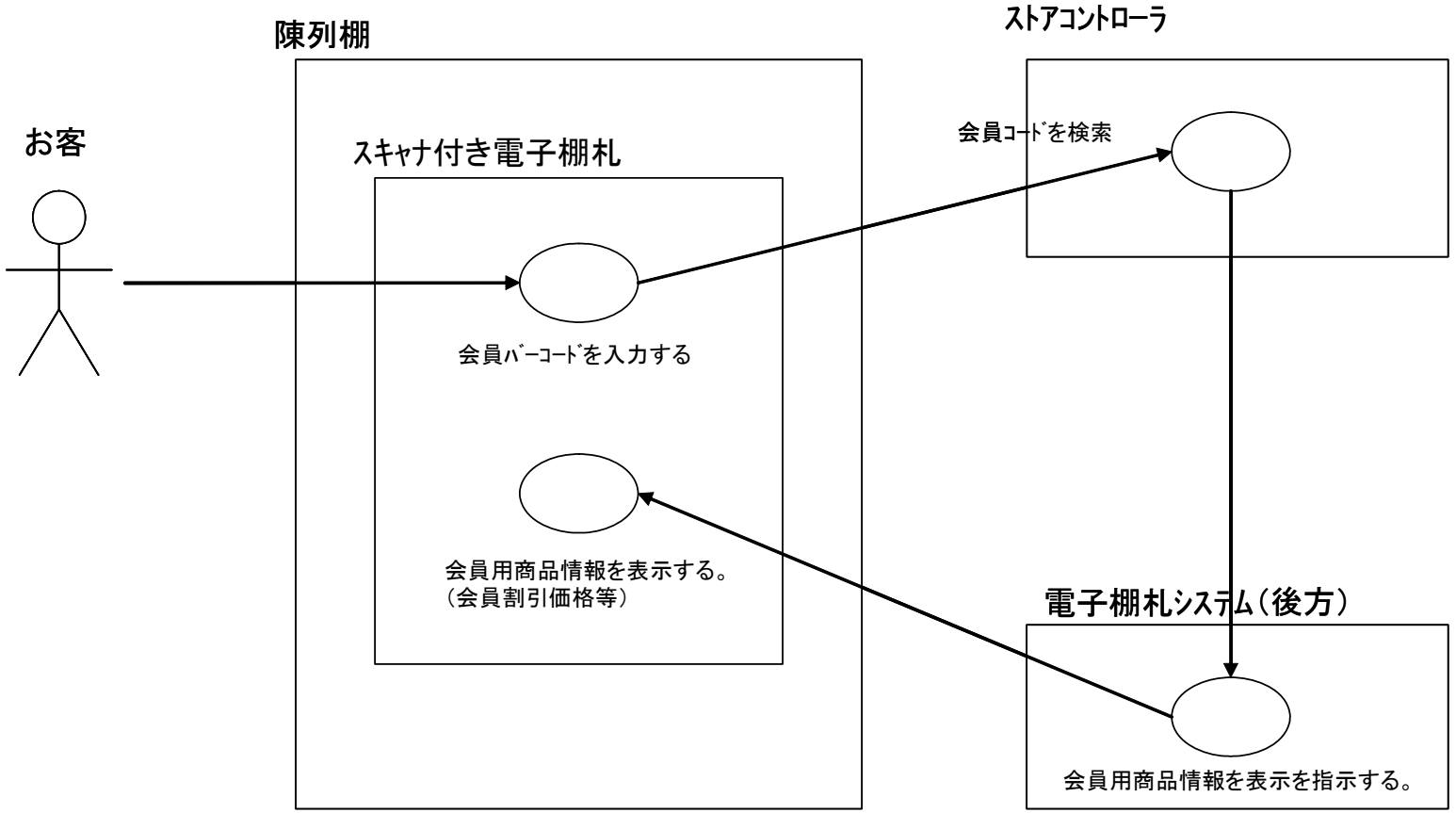


エンティティの定義（その２）：

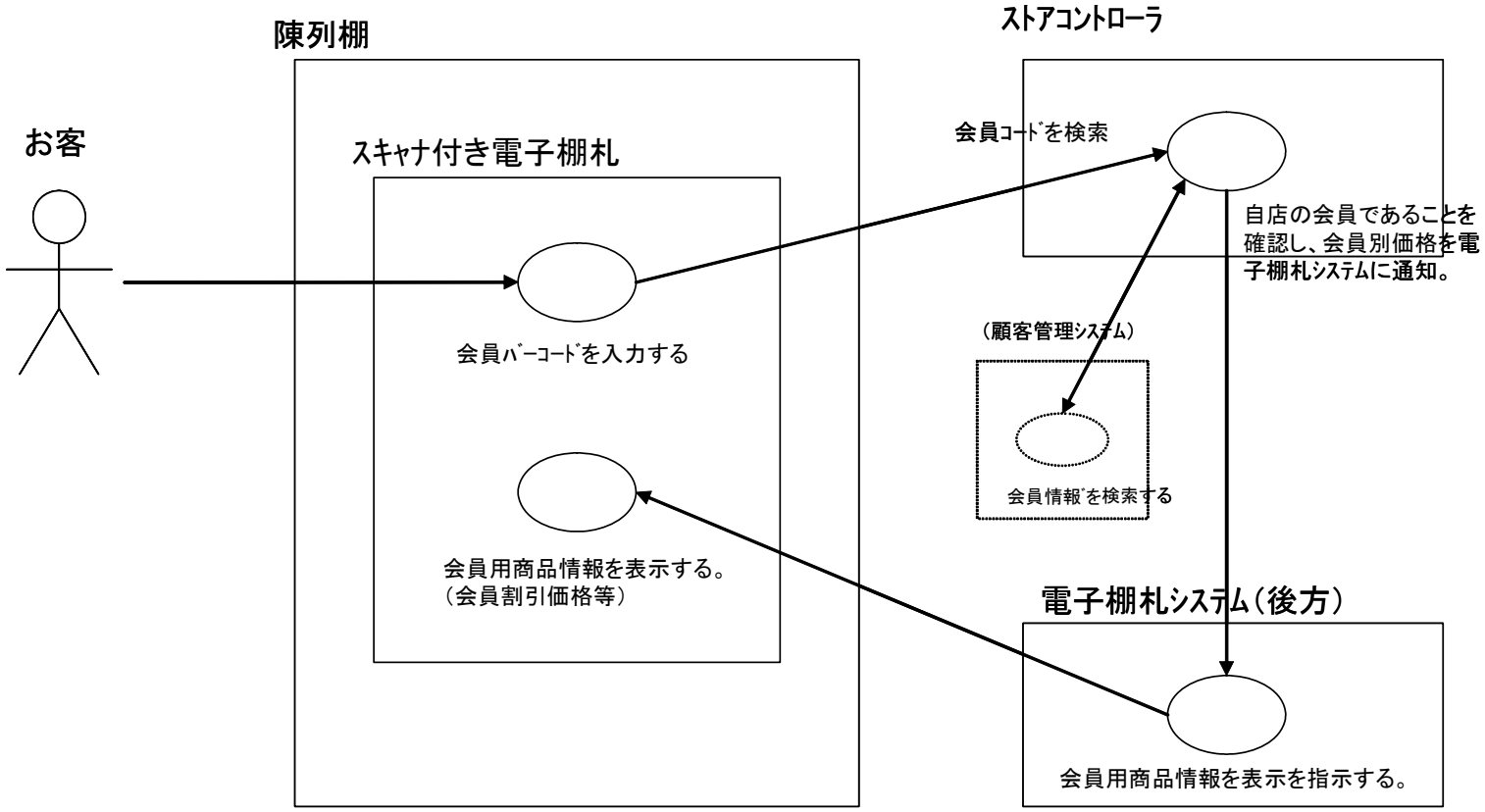
- 顧客　　：顧客は、該当商品の売価（会員売価含む）などの商品情報を電子棚札から得る人である。
- スキャナ付電子棚札　　：スキャナ付電子棚札は、会員バーコードを読み取るとともに商品価格などの商品情報を表示

  する機器である。（但し、現状は存在しないデバイスである）
- ストアコントローラ　　：商品情報（商品名、売価、会員価格、数量など）を管理するサーバである。

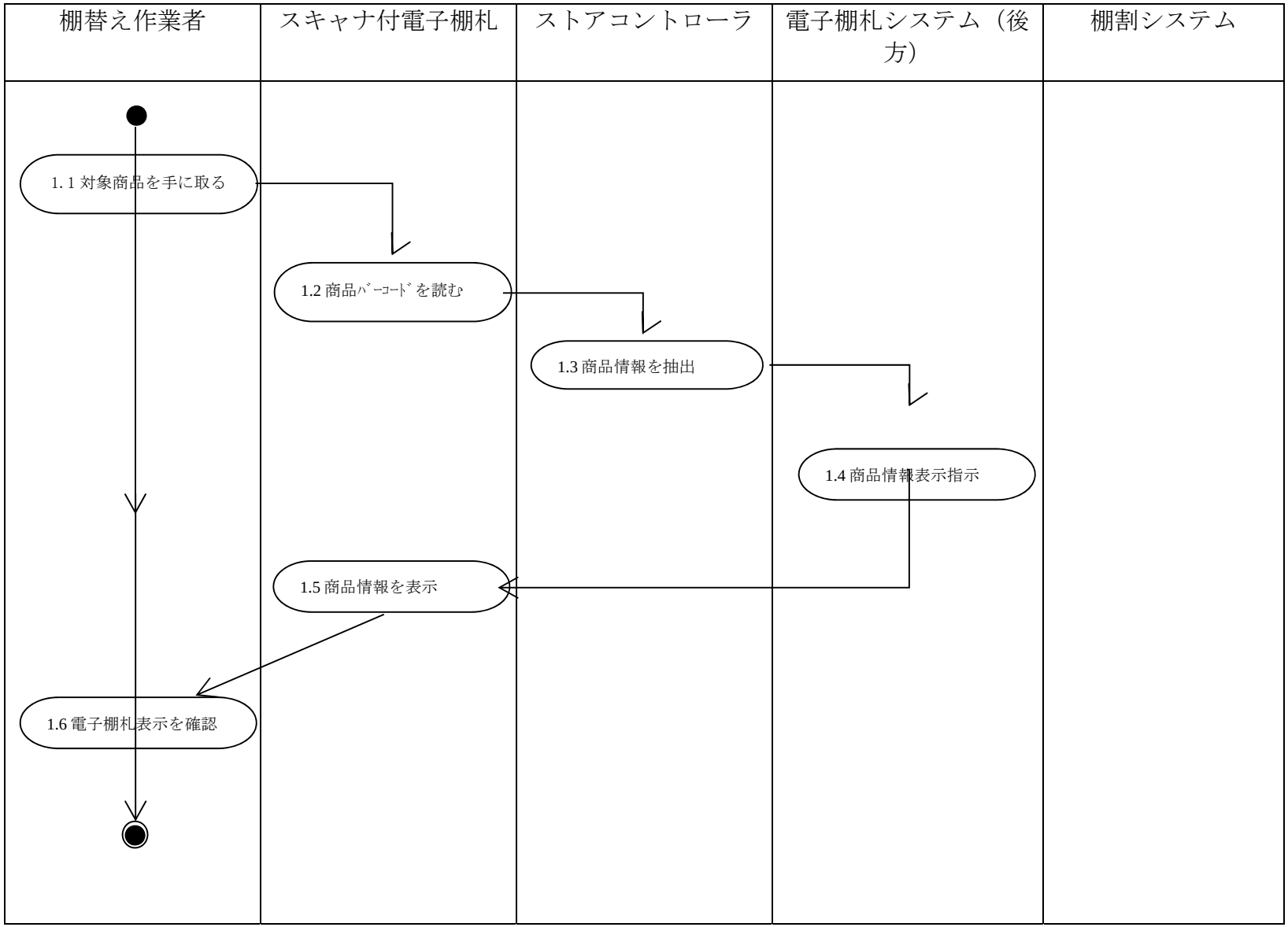
電子棚札システム（後方）：スキャナ付電子棚札へ情報表示指示や表示状況を管理するサーバである。

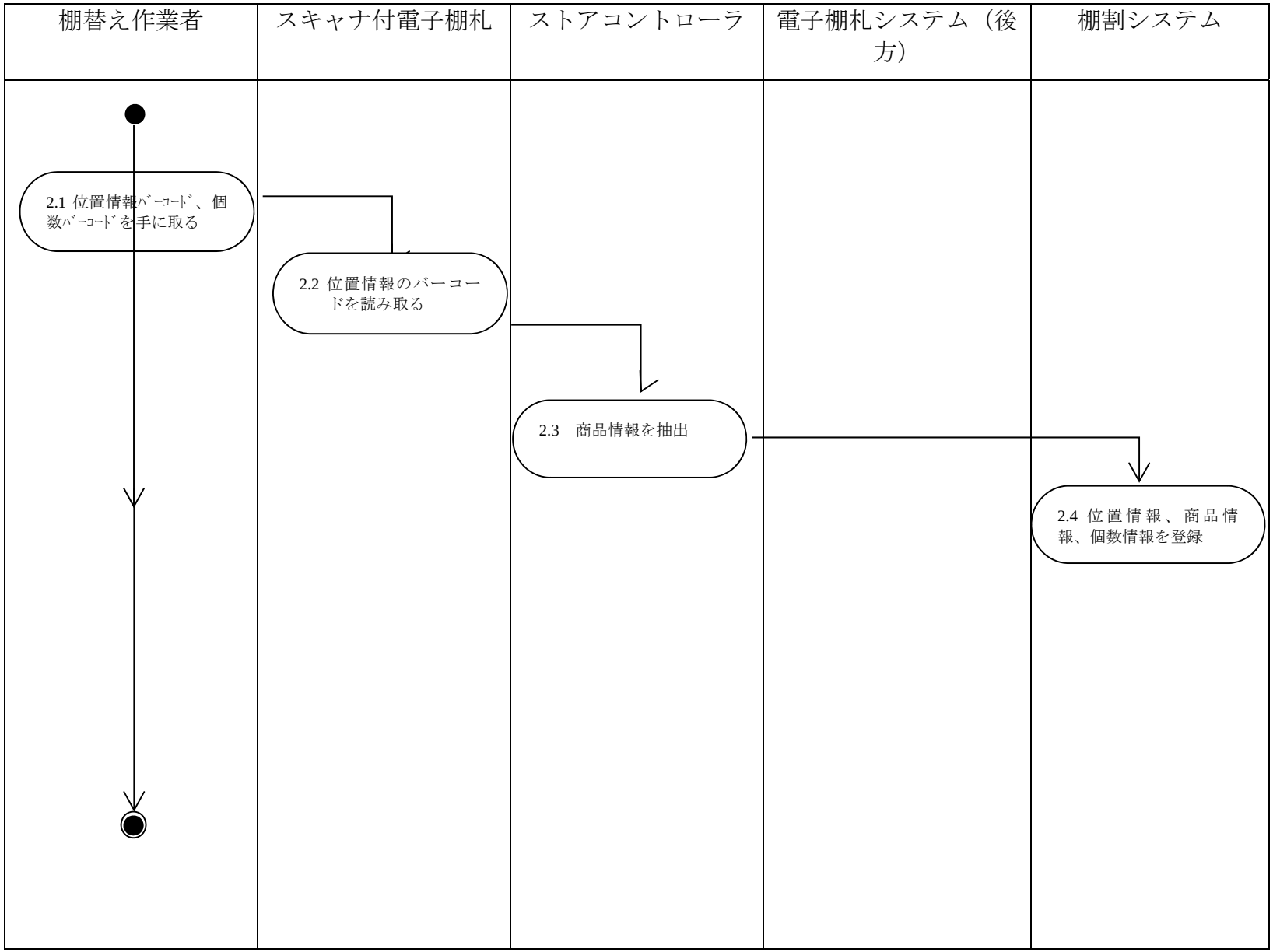


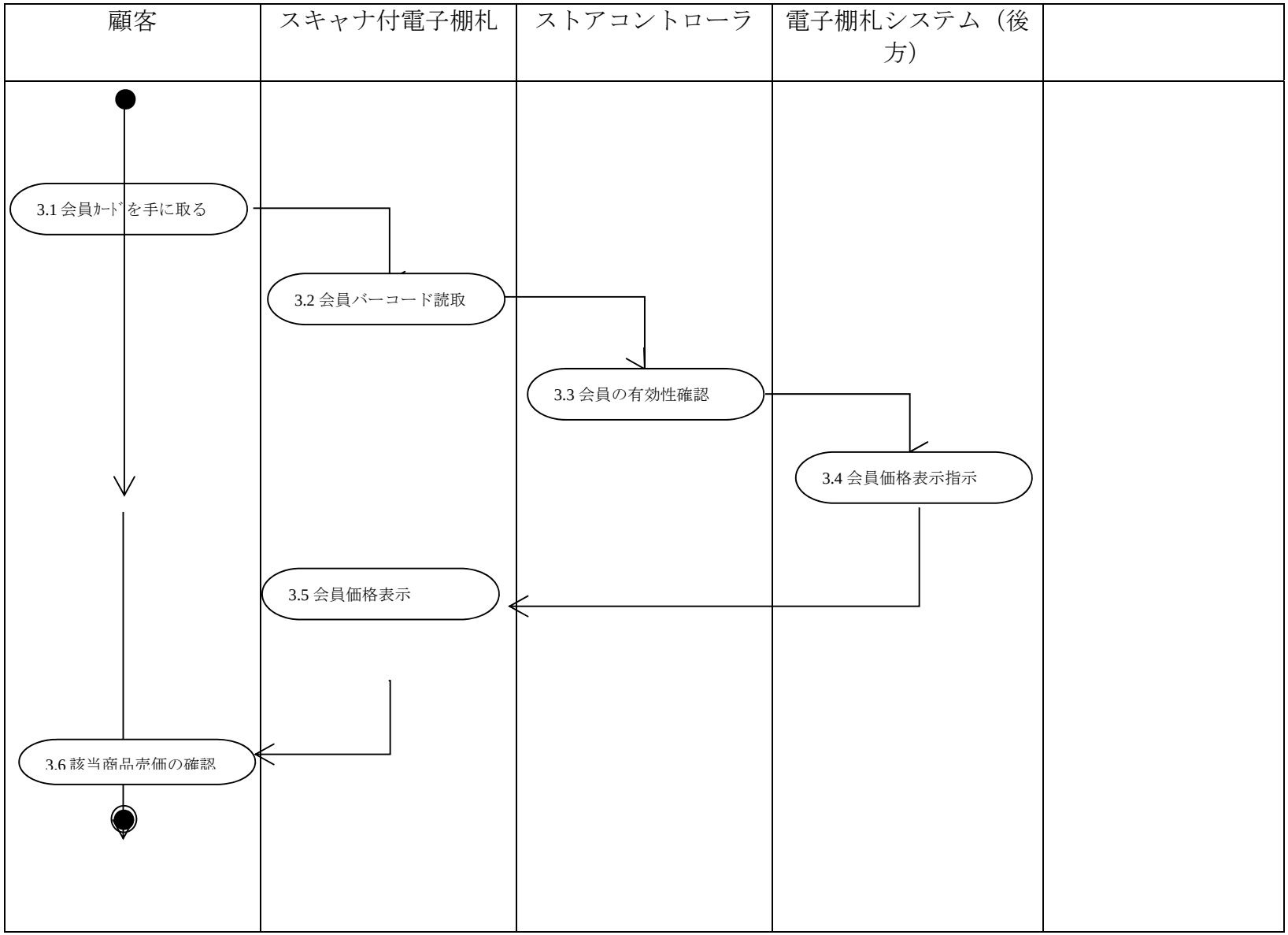
<Alternative method>



Activity Diagram

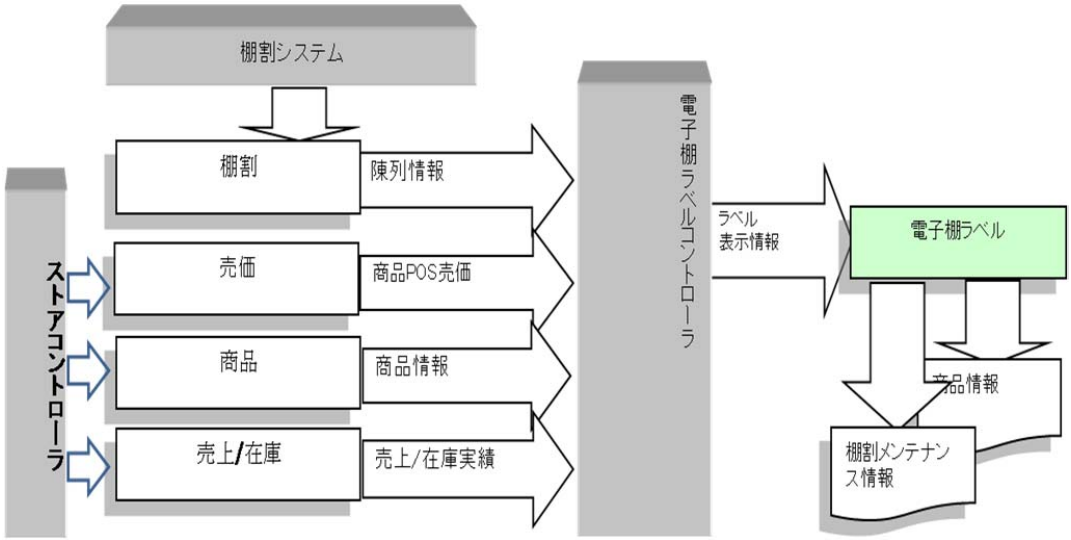






3.11　サブスコープ：電子棚ラベルと棚割情報連携

Contributor: 株式会社リテイルサイエンス



対象とするエンティティ

- 棚割（棚割システム）
- 電子棚ラベルコントローラ
- 電子棚ラベルデバイス
- 棚割メンテナンス情報

対象としないエンティティ

- ストアコントローラ
- 売価
- 商品
- 売上/在庫

利害関係者と関心

Stakeholders	Concerns
棚割管理者	棚割管理者は、店別の陳列データを作成、管理する責任者である。棚割管理者は、店舗が商品の改廃を計画通りに、正確に実施してもらい、売場の生産性を高めたいと考えている。
店舗オペレータ	店舗オペレータは、棚割管理者が計画した棚割を実際に店頭の棚上に反映する義務がある。その際、できるだけ、効率的に、間違いのない作業が行えることを願っている。

サブスコープの定義

電子棚ラベルに関するエンティティ群を全体のスコープとし、電子棚ラベルの位置情報と商品情報の紐付けを管理し、電子棚ラベルに表示させるためのアプリケーション間、及び、アプリケーションとデバイスのインタフェース、電子棚ラベルが表示する棚割メンテナンス情報について、をサブスコープとする。

対象となるエンティティの定義

- 棚割（棚割システム）

  外部の棚割システムなどで作成された小売企業内の店別売場別の棚割情報をインポートして保持し、電子棚ラベルコントローラが管理している電子棚ラベルデバイスと、店頭の棚上に陳列される商品の位置との紐付けを提供するアプリケーションである。場合により、棚割システムそのものが拡張され、利用される場合もある。

  「棚割」は、従来、商品コードと棚割上の位置情報を管理することが、主要な要件だが、ここでは、さらに、電子棚ラベルデバイスのシリアル番号と商品コード、商品の位置情報を管理することによって、棚上の特定の位置に取り付けられている電子棚ラベルデバイスに対して、商品の情報を提供することを支援する。

  「棚割」は、電子棚ラベルデバイスを制御するコントローラへ、陳列情報を反映させるためのトリガーを持っているか、「棚割」のなかに、電子棚ラベルデバイスへ情報を提供するコントローラ機能を持たせる場合もある。

- 電子棚ラベルコントローラ
  このコントローラは、電子棚ラベルデバイスのシリアル番号と、それが表示する情報を管理する。電子棚ラベルコントローラは、「棚割」と連携し、指定された電子棚ラベルデバイスが表示する情報を電子棚ラベルへと送信する。

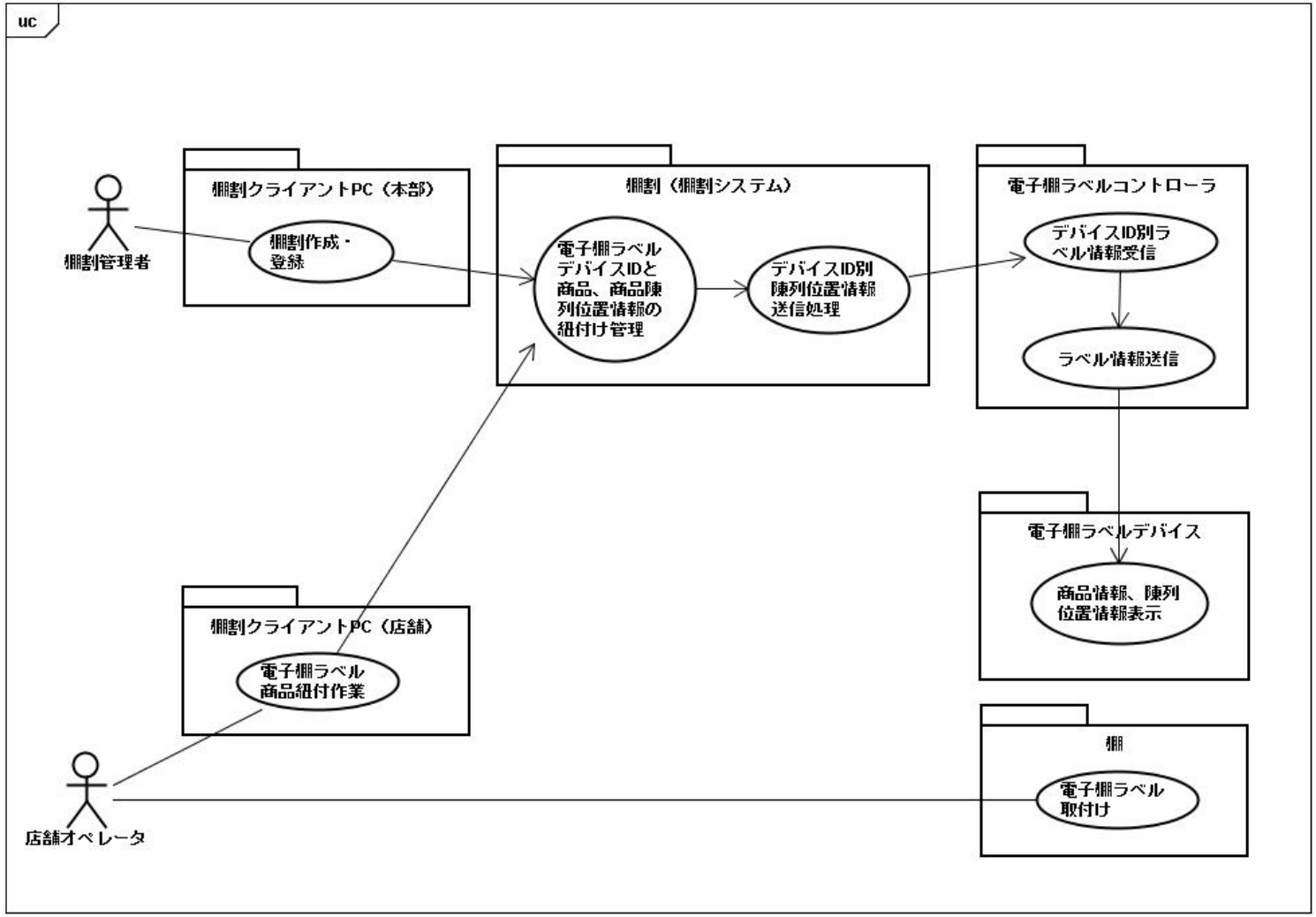
  情報として代表的な項目は、商品名、価格、発注単位などの基本情報と、今回対象とする、棚割メンテナンス業務を支援する「棚割メンテナンス情報」を保持する。

  電子棚ラベルデバイスの性能上、電子棚ラベルコントローラが、デバイス専用の通信手段に合わせたコントロール機能を持つ場合もある。また、商品の価格など、POS システムのストアコントローラから、マスタ情報、現在売価情報を LAN 経由で同期している。

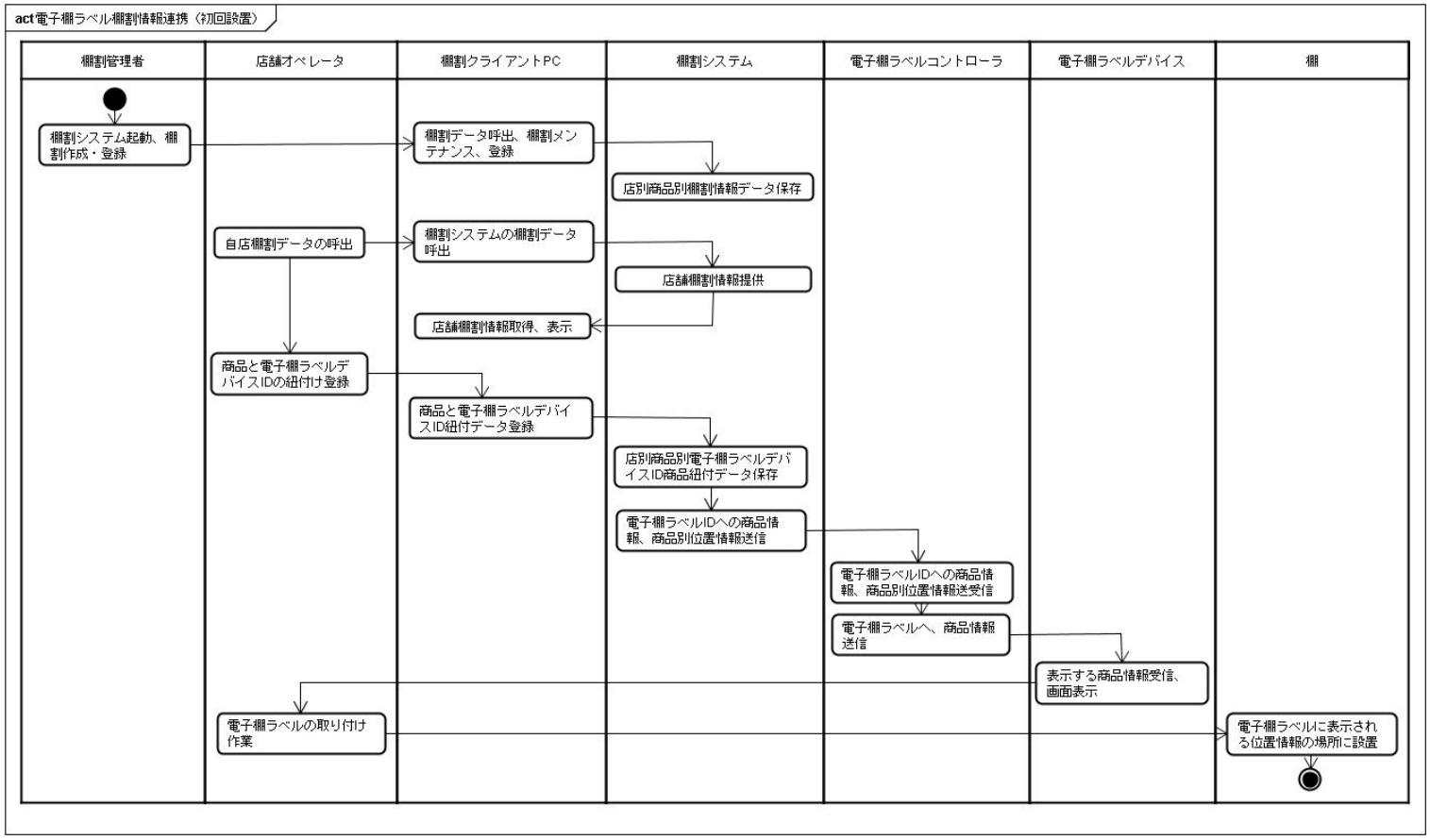
- 電子棚ラベルデバイス
  顧客、または従業員へ、商品についての情報を表示する装置で、電子棚ラベルのシリアル番号を管理する電子棚ラベルコントローラにより、表示内容が制御される。

  今回の目的としては、商品名、バーコードも液晶ディスプレーなどで表示可能なものが望まれる。また、顧客向けの情報と従業員向けの情報を切り替える必要性がある場合も考えられる。電子棚ラベルデバイスとの通信手段は、将来的には、直接、デバイスに無線 LAN 接続、Bluetooth 接続などで通信できることが、汎用的な利用に有利になる可能性があるが、現状、コスト、普及の側面から、デバイス専用の電子棚ラベルコントローラを経由し、有線、無線による赤外線通信のものを利用が先行するだろう。

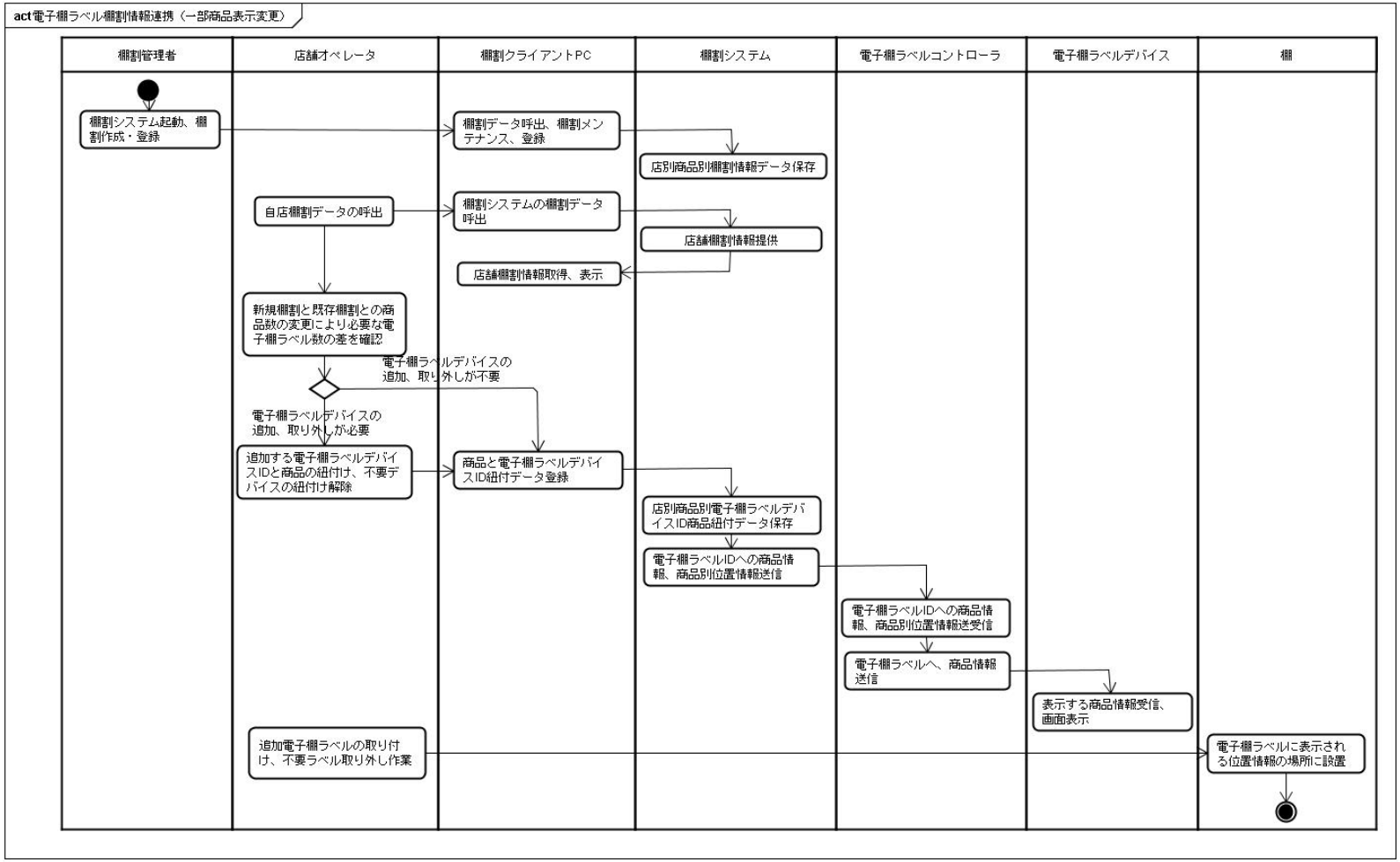
- 棚割メンテナンス情報
  棚割メンテナンス情報は、電子棚ラベルデバイスが表示する、棚割メンテナンスに関する情報である。電子棚ラベルが表示する情報には、商品についての説明として、商品名、価格、商品コード（バーコード）が必要だが、今回のスコープとしては、それに加え、主に従業員が必要とする棚割メンテナンスに関する情報（陳列位置情報（ゴンドラ番号、棚段番号、陳列順）、フェイス数、商品適用期間など）の情報が必要である。



Activity Diagram (High level use case)



Activity Diagram (Extended use case 1)



3.12　サブスコープ：セルフ給油

Contributor: NEC インフロンティア株式会社

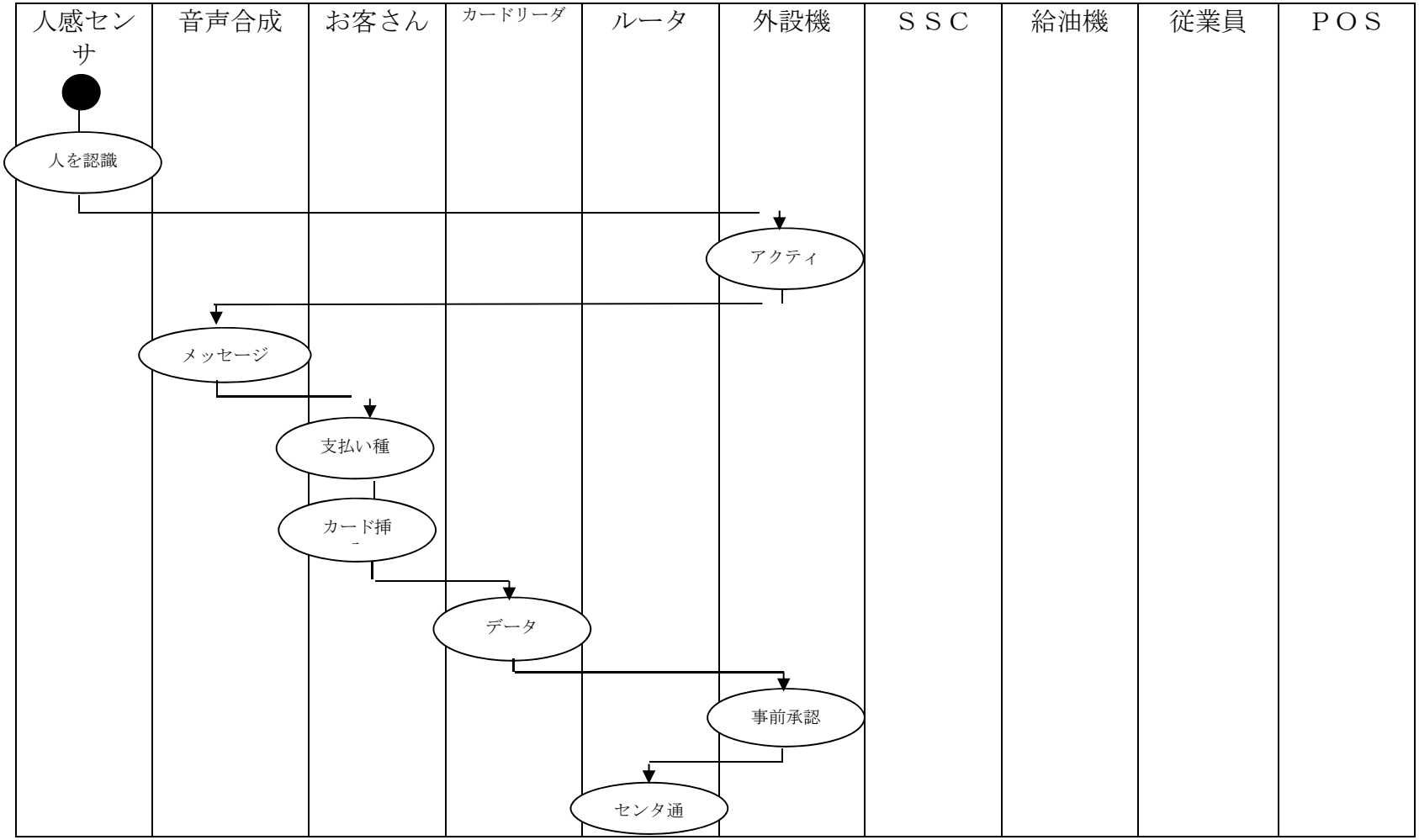
ＳＳ(Service Station)にてセルフ給油処理を行う。

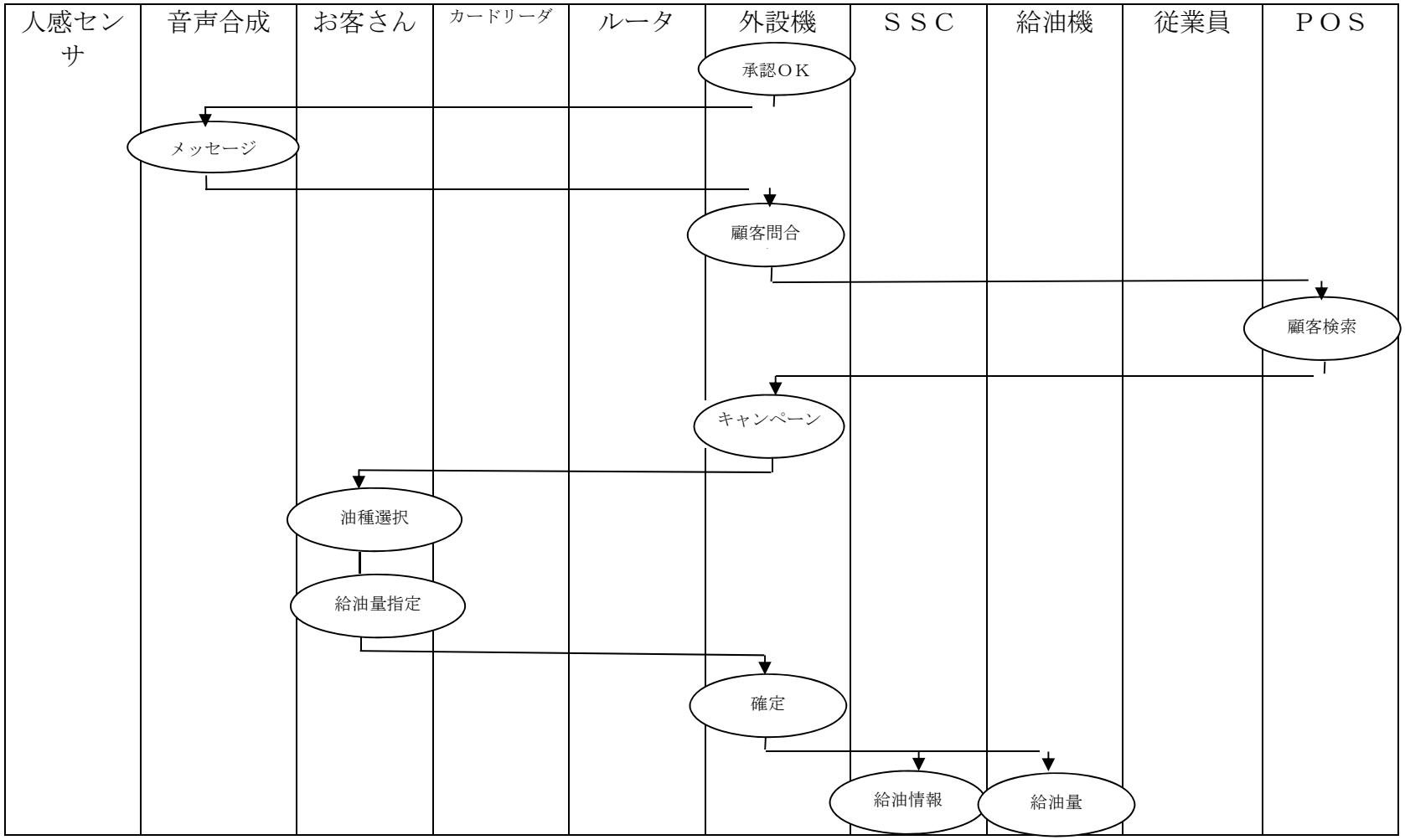
エンティティの定義:

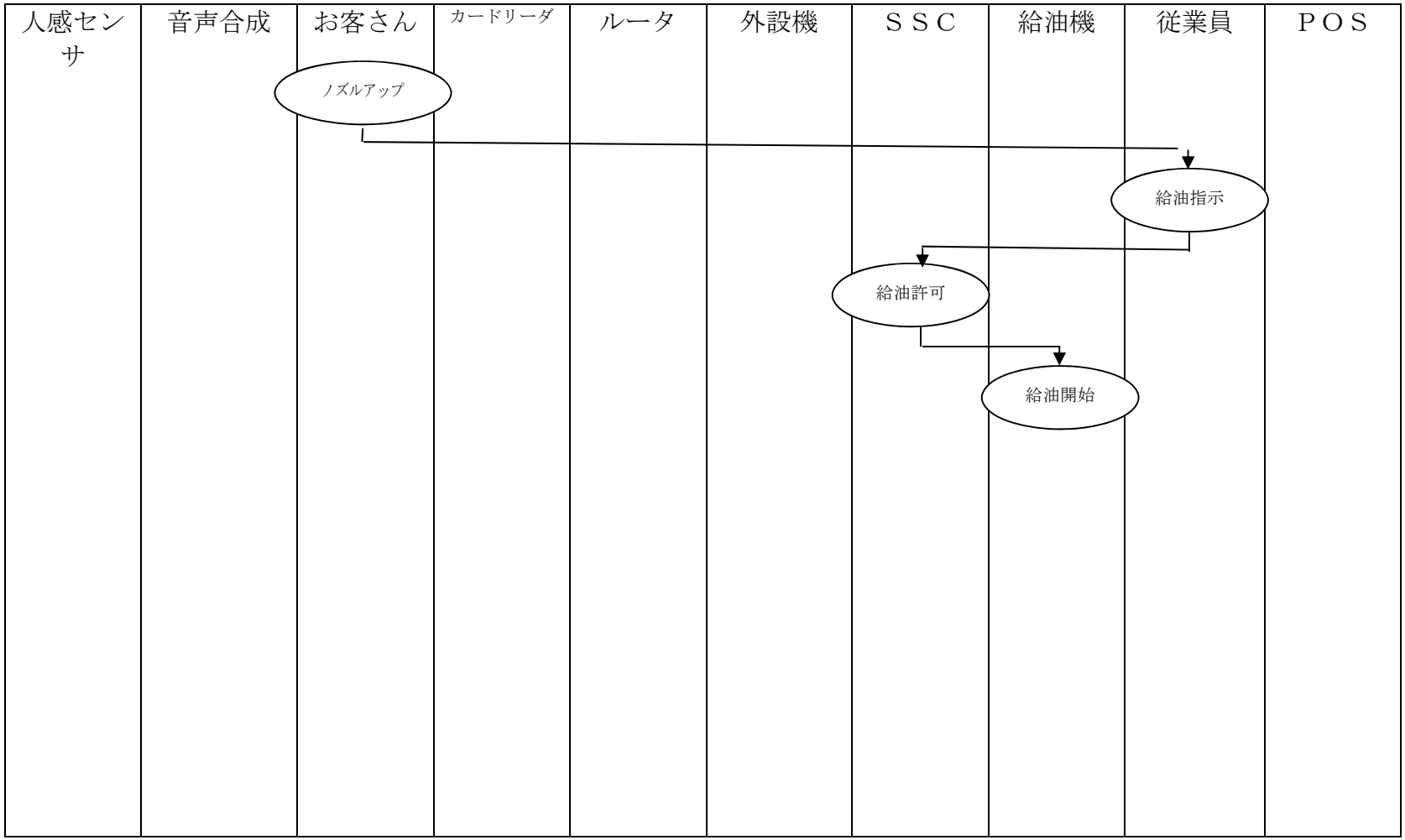
お客さん	SS にてセルフ給油を実施する。
車	燃料を給油される。
従業員	お客様の給油処理に対して、給油許可と緊急停止の SSC より実施する。
クレジットカード	クレジットで給油処理を行う。
電子マネー	電子マネーで給油処理を行う。
現金処理	現金で給油処理を行う。
ＳＳＣ（Self Service Console）	給油機の給油許可と給油禁止を制御する。
POS	給油処理、発注、仕入れ、単品売上、カード発行、顧客管理、商品管理を行う。
カードリーダ	マグネチックカードのリードと作成を行う
非接触リーダ	電子マネーを処理する
人感センサー	人が近づくと外設機をアクティブにする。
音声合成装置	次の処理を音声で通知する。
プリンタ	給油が完了したらレシートを印字する。
入金機	現金処理時に決済するお金を入金する。
釣銭機	給油処理終了時におつりを出力する。
外設機	屋外にて、決済処理、油種の指定、給油量の指定を行う。

	クレジット決済時は、事前承認処理を実施する。
	電子決済は、与信確認を行う。
	現金決済時は入金処理と釣銭支払いを釣銭機に指示する。
	給油完了時にプリンタからレシートを出力する。
計量機	給油量を計測する。
給油機	車に燃料を給油する。
油面計	タンク内の各油種の残量を計測する
洗車機	車を洗車する。
ルータ	クレジットセンターとの通信、集計センターとの通信を行う。
Forecourt Device 通信 BOX	ＰＯＳ、外設機、ＳＳＣと Forecourt の各機器とのブリッジを行う、
店舗サーバ	顧客管理、売上管理レポート出力、元売との情報交換を行う、
顧客情報	車に関する情報（点検、車検、オイル交換）
	顧客に関する情報の管理（趣味、住所、家族構成、年齢等々）
取引データ	給油量、支払い区分、単価、

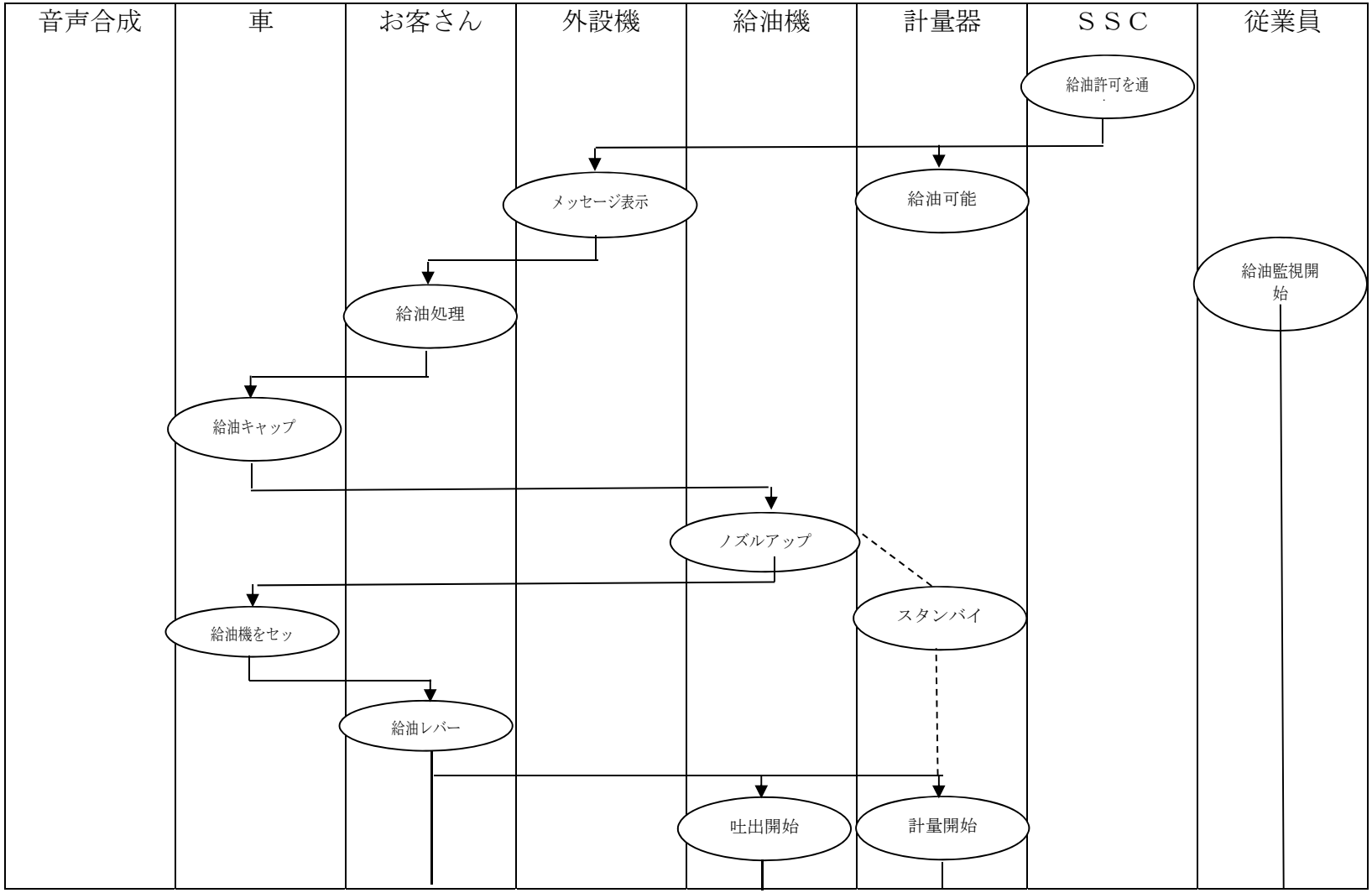
Activity Diagram

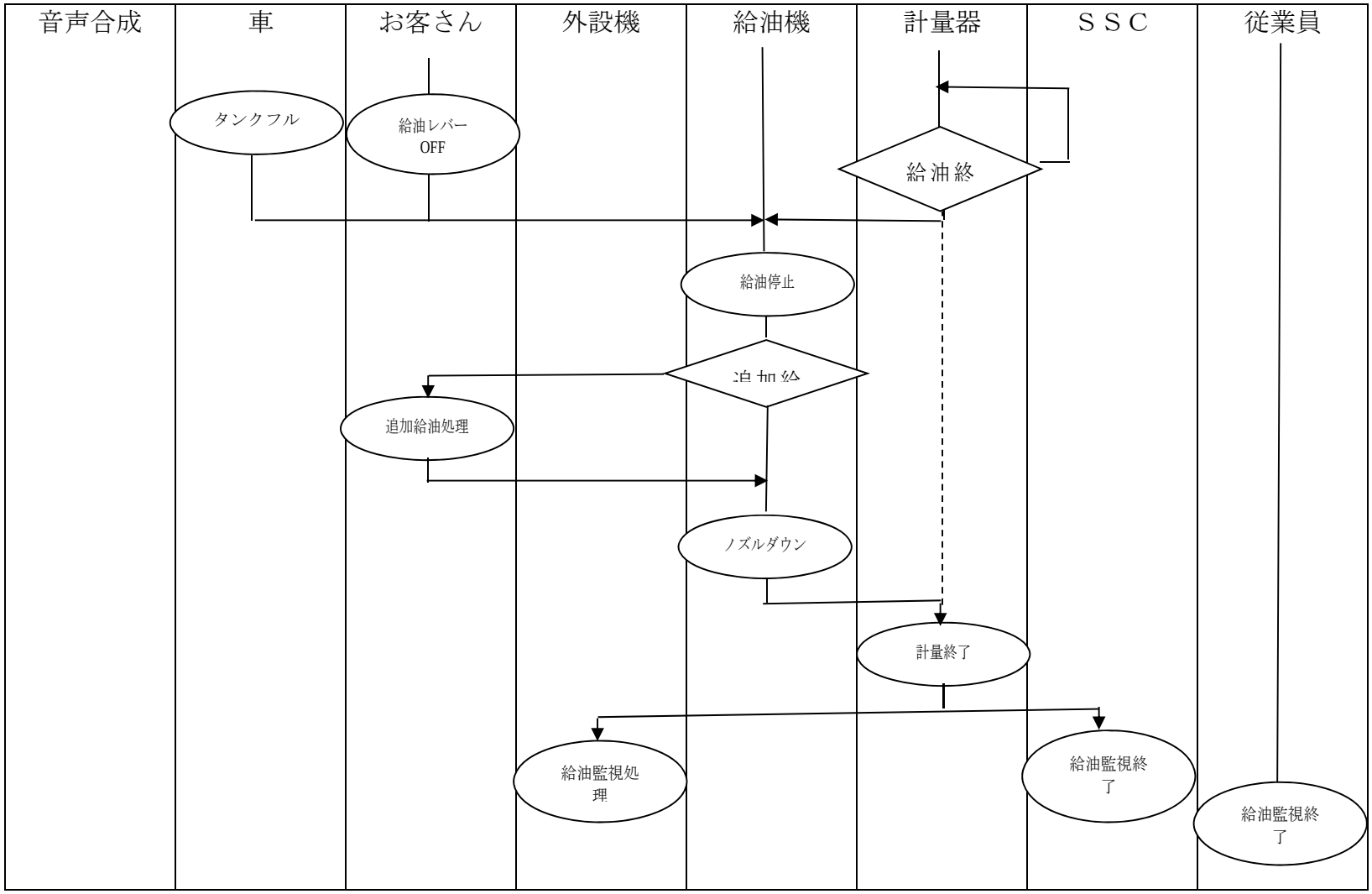




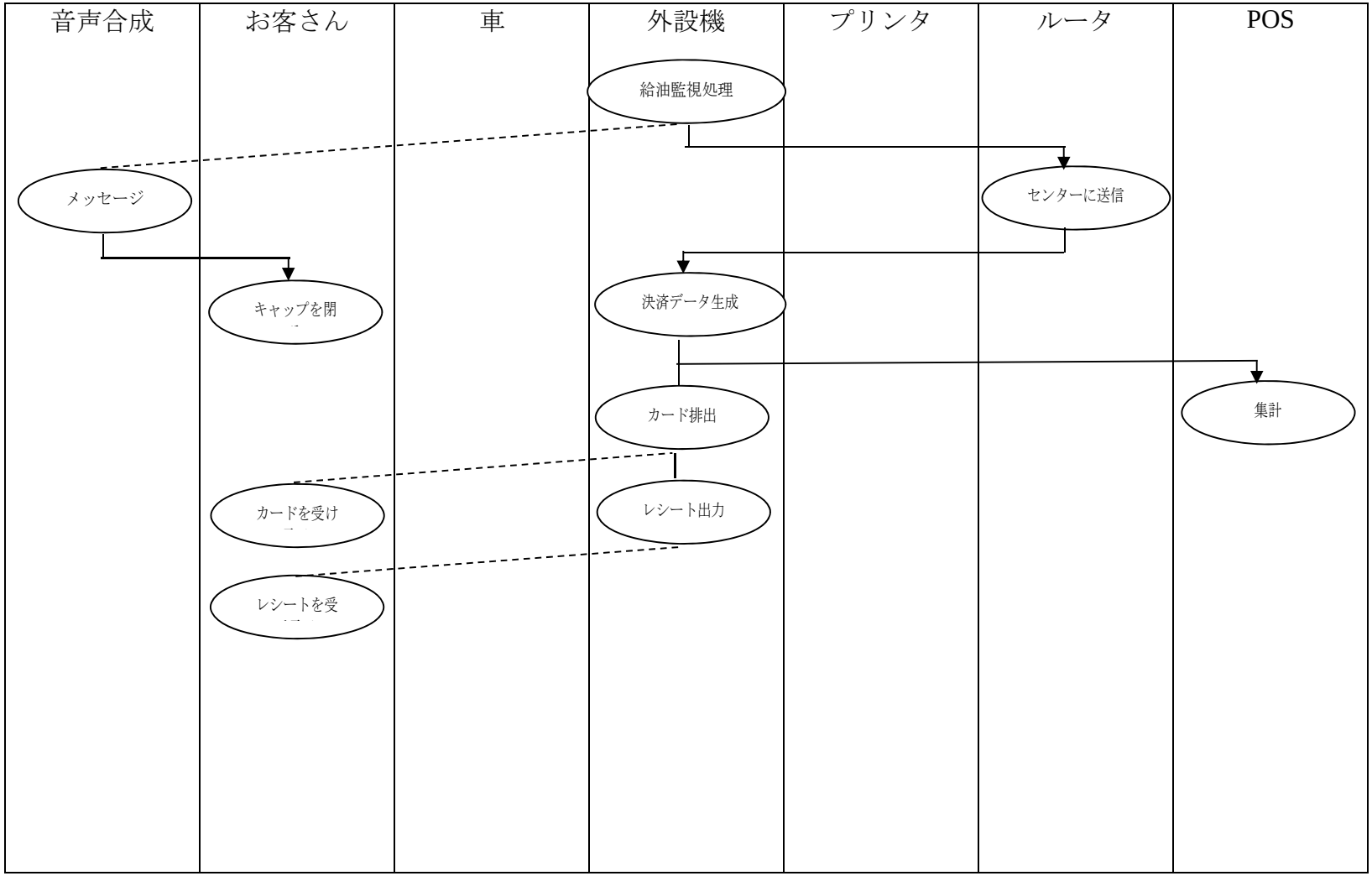


Activity Diagram





Activity Diagram



4. 文書履歴

バージョン履歴

Ver	日時	章	変更点
0.1	2007-09-02	全て	初版
0.2	2007-09-11	全て	参照の修正、表現の修正、Web サービスとそれ以外の要素の分離
0.3	2008.07.31	全て	文書の再構成、各章の内容に関する注意書き追加
0.4	2008.08.26	セキュリティ	特定のセキュリティ要件追加、WS-POS フローの章、対象としない範囲
0.5	2008.09.24	全て	イベント、セキュリティ、サービス記述、探索
1.1	2011.05.15	全て	次世代 POS 研究会のフィールドテストからの知見を仕様書に反映；WCF と JAX-WS を用いた WSDL 生成に関する記述更新と、相互運用性を確実にするための XMLPOS の記述更新；すべての UnifiedPOS 1.14 周辺デバイスの.Net が生成した例としての WS-POS クラスダイアグラムを追加；開発者を助けるソフトウェアのサポートファイルを追加
1.2	2013.01.22	注記した場所	バージョン 1.2 の仕様書は、バージョン 1.1 の明確化と、必要な機能強化を説明する新しい章や、既存の章の更新が含まれている。それらの詳細が以下に、関連する章へのリンクとともに示されている。 • フロントページのバージョンと発行日の更新 • Web サービスコンポーネントの更新 Page 9 • “サービス記述と探索”の章の更新 Page **エラー! ブックマークが定義されていません。**. • “WS-POS ビヘイビアモデル”を説明する新しい章の追加 Page 16. • “プロパティ、メソッドとイベント”を説明する新しい章の追加 Page 17. • “WS-POS 通信モデル”を説明する新しい章の追加 Page 17. • “セッションマネージャとデバイス制御”を説明する新しい章の追加 Page 18. • “WS-POS セッションマネージャ概念の導入“を説明する新しい章の追加 Page 19. • “WS-POS セッションの識別“を説明する新しい章の追加 Page 20. • “WS-POS セッション確立の典型的なシーケンス” を説明する新しい章の追加 Page 20. • “WS-POS サービスメソッドの呼出しとプロパティの使用” を説明する新しい章の追加 Page 23. • “複数の WS-POS コンシューマによる単一の WS-POS プロバイダに対する Claim “の章を更新 Page 23. • “WS-POS メソッドとデバイスメソッド” を説明する新しい章の追加 Page 29.

1.2	2013.01.22 2013.03.26	注記した場所	• “双方向通信を用いた WS-POS イベントハンドリング “の章を更新 Page 30. • “ポーリングを用いた WS-POS イベントハンドリング “を説明する新しい章の追加 Page 35. • “WS-POS サービスのネットワーク接続管理の考慮” を説明する新しい章の追加 Page 40. • “WS-POS サービスにおけるネットワーク接続管理（イベント—双方向通信の場合）”を説明する新しい章の追加 Page 44. • “WS-POS サービスにおけるネットワーク接続管理（イベント—ポーリングの場合）”を説明する新しい章の追加 Page 52. • “WS-POS メソッドリファレンス(UnifiedPOS UML 形式)” を説明する新しい章の追加 Page 60. • “WSPOSEvent と WSPOSEventResponse “を説明する新しい章の追加 Page .70 • “XMLPOS への変更” を説明する新しい章の追加 Page 82 • “メソッド呼び出しのパラメータとして渡すファイルパス” を説明する新しい章の追加 Page 85. • “後方互換性” を説明する新しい章の追加 Page 107. • “文書履歴“の章を更新 Page 171. • “サポートファイル”を 6 章に追加 Page 174. • 双方向通信時のイベントの取り扱いについての記述追加 Page 30 • 新しい章「WS-POS 双方向通信 イベントリファレンス」を追加 Page 77

5. 用語集

単語	説明

6. 参考文書とサポートファイル

6.1 参考文書

文書	説明
WS-I Basic Profile V1.2	http://www.ws-i.org/Profiles/BasicProfile-1.2.html
SOAP 1.1	http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
SOAP HTTP Binding	http://www.w3.org/TR/2007/REC-soap12-part2-20070427/#soapinhttp
SOAP HTTP Status Codes, Error Conditions	http://www.w3.org/TR/2007/REC-soap12-part2-20070427/#http-reqbindwaitstate
WS-Addressing 1.0	http://www.w3.org/Submission/2004/SUBM-ws-addressing-20040810/
WSDL 1.1	http://www.w3.org/TR/wsdl
WS-MetadataExchange Publication	http://specs.xmlsoap.org/ws/2004/09/mex/WS-MetadataExchange.pdf
UDDI V3	http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm
TLS V1.2	http://www.ietf.org/internet-drafts/draft-ietf-tls-rfc4346-bis-04.txt
WS-Security V1.1	http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
Using WSDL in a UDDI Registry, Version 2.0.2	http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v2.htm
UnifiedPOS 1.13	www.NRF-ARTS.org

6.2 サポートファイル

.開発者が WS-POS1.1 互換アプリケーションを作成する際に、以下の SDK ソフトウェアファイルが入手可能。（参考：www.NRF-ARTS.org ）

WS-POS Ver. 1.1 ファイルリスト

1) WS-POS 1.1 対応 XML スキーマ (XMLPOS 1.13 を利用)

File Name: WS-POS1.1XSD.zip

2) WS-POS 1.1 対応 WSDL ファイル (Independent files and implementation dependent files)

File Name: WS-POS1.1WSDL.zip

3) WS-POS 1.1 WCF コントラクト

File Name: WS-POS1.1WCFContract.zip

4) WS-POS 1.1 JAX-WS コントラクト

File Name: WS-POS1.1JAX-WSContract.zip

5) WS-POS 1.1 WCF サンプルプログラム

File Name: WS-POS1.1WCFSample.zip

6) WS-POS 1.1 JAX-WS サンプルプログラム

File Name: WS-POS1.1JAX-WSSample.zip

***Important Notice For Condition of Use**:*

***ARTS** and **User** agree that the Software is provided "**AS IS**" and that **ARTS** makes no warranty as to the Software. **ARTS** DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, RELATED TO THE SOFTWARE, ITS USE OR ANY INABILITY TO USE IT, THE RESULTS OF ITS USE AND THIS AGREEMENT.*

(利用条件に関する重要な注意（日本語参考訳）

***ARTS** と **User** は上記ソフトウェアが"そのまま"の状態で提供され、**ARTS** は該ソフトウェアに対し無保証であることに合意します。 **ARTS** は表記されている、いないにかかわらず、市場での暗黙の保証や特定の目的に合致していることに制限されないことも含め、本ソフトウェアに関する、その使い方や、利用上のいかなる不都合、利用したことと本合意事項による結果に関するすべての保証を放棄します。）*

6.3 サポートファイル　　*Version 1.2 で追加*

開発者が WS-POS1.2 互換アプリケーションを作成する際に、以下の SDK ソフトウェアファイルが入手可能。（参考：www.NRF-ARTS.org ）

WS-POS Ver. 1.2 File Lists

1) WS-POS 1.2 対応 XML スキーマ　(XMLPOS 1.14*** を利用)

File Name: WS-POS1.2XSD.zip

2) WS-POS 1.2 対応 WSDL ファイル (Independent files and implementation dependent files)

File Name: WS-POS1.2WSDL.zip

3) WS-POS 1.2 WCF コントラクト

File Name: WS-POS1.2WCFContract.zip

4) WS-POS 1.2 JAX-WS コントラクト

File Name: WS-POS1.2JAX-WSContract.zip

5) WS-POS 1.2 クラス図

****注：WS-POS 1.2 は XMLPOS 1.14 を取り込んだ UnifiedPOS 1.14 のサポートを考慮して作成されています。ただし、WS-POS 1.2 は、UnifiedPOS 1.14 で追加された機能を必要としない場合や、現時点で XMLPOS1.14, POS for .Net 1.14, Common Control Objects 1.14、JavaPOS サービス 1.14 が対応されていない状況があるため、旧バージョンである UnifiedPOS 1.13 と 1.12 にも対応します。もし選択する余地があるなら、UnifiedPOS 1.14 対応で実装することを強く推奨します。*

***Important Notice For Condition of Use**:*

***ARTS** and **User** agree that the Software is provided "**AS IS**" and that **ARTS** makes no warranty as to the Software. **ARTS** DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, RELATED TO THE SOFTWARE, ITS USE OR ANY INABILITY TO USE IT, THE RESULTS OF ITS USE AND THIS AGREEMENT.*

(利用条件に関する重要な注意（日本語参考訳）

***ARTS** と **User** は上記ソフトウェアが"そのまま"の状態で提供され、**ARTS** は該ソフトウェアに対し無保証であることに合意します。 **ARTS** は表記されている、いないにかかわらず、市場での暗黙の保証や特定の目的に合致していることに制限されないことも含め、本ソフトウェアに関する、その使い方や、利用上のいかなる不都合、利用したことと本合意事項による結果に関するすべての保証を放棄します。）*

7. WS-POS クラス図 *Version 1.2 で更新*

Version 1.2 に着手するにあたり、UnifiedPOS 仕様で説明されている 36 種類の POS 周辺デバイスに対応した WS-POS サービスを説明するクラス図の例は、6.3 章に記載されている通りサポートファイルの一つに分離した。Web サービスの実装に必要なプロセスとデータの相互関係を理解するための助けになるドキュメントである。

UML の例に関する重要な注意

クラス図の例は Microsoft Visual Studio 2010 開発ツールを使って、C#のコードから抽出されたものである。その結果、プログラムに依存した言語としての特徴は C#のみに有効である。同等の Java 開発ツールを使って Java 実装に対する似たようなクラス図の作成も可能だが、プログラムに依存した言語としての特徴は、記載されているものとはことなるだろう。

ここで意図するのは、標準的な UML 表記を用いて、UnifiedPOS が定義するプロパティ、メソッド、イベント、データとの相対的な関係を示すことである。

周辺デバイスはアルファベット順に記述されており、UnifiedPOS 標準のそれぞれの対応する周辺デバイスの章と一致している。

8. アプリケーション開発サポート *Version 1.2 で更新*

以下では、WS-POS 対応スキーマがどのように UnifiedPOS のプロパティ、メソッド、イベントに対して WSDL や C# Web サービスコントラクト、Java Web サービスコントラクトと対応づけられるかの例を示す。

まず最初に、読み書き可能なプロパティである”**DeviceEnable**”と”**checkHealth**”メソッドが UnifiedPOS デバイスカテゴリーの”Belt”に対して、WSDL 文書としてはどのように表現されるかの例を示す。

次に、それに関連する Java と C#クラスが、JAX-WS と WCF ツールによって生成される。Belt に対する、すべてのその他のプロパティ、メソッドは、読みやすさのために省略(‘…’として)している。

イベントは別の WSDL ファイルとして指定され、この例では含まれていないことに注意すること。

/*---------- BeltV1.2.0.wsdl start ----------*/

```
        <?xml version="1.0" encoding="utf-8"?>
        <wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                          xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
                          xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
                          xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
                          xmlns:wsap="http://schemas.xmlsoap.org/ws/2004/08/addressing/policy"
                          xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
                          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                          xmlns:tns="http://www.nrf-arts.org/UnifiedPOS/Belt/"
                          xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
                          xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
                          xmlns:wsa10="http://www.w3.org/2005/08/addressing"
                          xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
                                     name="BeltService"
                                     targetNamespace="http://www.nrf-arts.org/UnifiedPOS/Belt/"
                          xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
          <wsdl:types>
                  <xsd:schema targetNamespace="http://www.nrf-arts.org/UnifiedPOS/Belt/Imports">
                  <xsd:import namespace="http://www.nrf-arts.org/UnifiedPOS/Belt/" schemaLocation="BeltV1.13.1.xsd" />
             </xsd:schema>
          </wsdl:types>

        ...

        <wsdl:message name="Belt_GetDeviceEnabled_InputMessage">
```

```
    <wsdl:part name="parameters" element="tns:GetDeviceEnabled" />
</wsdl:message>

  <wsdl:message name="Belt_GetDeviceEnabled_OutputMessage">
    <wsdl:part name="parameters" element="tns:GetDeviceEnabledResponse" />
  </wsdl:message>

  <wsdl:message name="Belt_GetDeviceEnabled_UposException_FaultMessage">
    <wsdl:part name="detail" element="tns:UposException" />
  </wsdl:message>

  <wsdl:message name="Belt_SetDeviceEnabled_InputMessage">
    <wsdl:part name="parameters" element="tns:SetDeviceEnabled" />
  </wsdl:message>

  <wsdl:message name="Belt_SetDeviceEnabled_OutputMessage">
    <wsdl:part name="parameters" element="tns:SetDeviceEnabledResponse" />
  </wsdl:message>

  <wsdl:message name="Belt_SetDeviceEnabled_UposException_FaultMessage">
    <wsdl:part name="detail" element="tns:UposException" />
  </wsdl:message>

...
  <wsdl:message name="Belt_CheckHealth_InputMessage">
    <wsdl:part name="parameters" element="tns:CheckHealth" />
  </wsdl:message>

  <wsdl:message name="Belt_CheckHealth_OutputMessage">
    <wsdl:part name="parameters" element="tns:CheckHealthResponse" />
  </wsdl:message>

  <wsdl:message name="Belt_CheckHealth_UposException_FaultMessage">
    <wsdl:part name="detail" element="tns:UposException" />
  </wsdl:message>

 ...
<wsdl:portType name="Belt">

...
  <wsdl:operation name="GetDeviceEnabled">

       <wsdl:input wsaw:Action="http://www.nrf-arts.org/UnifiedPOS/Belt/GetDeviceEnabled"
          message="tns:Belt_GetDeviceEnabled_InputMessage" />

      <wsdl:output wsaw:Action="http://www.nrf-arts.org/UnifiedPOS/Belt/GetDeviceEnabledResponse"
          message="tns:Belt_GetDeviceEnabled_OutputMessage" />
```

```
            <wsdl:fault wsaw:Action="http://www.nrf-arts.org/UnifiedPOS/Belt/UposException" name="UposException"
                message="tns:Belt_GetDeviceEnabled_UposException_FaultMessage" />

        </wsdl:operation>

        <wsdl:operation name="SetDeviceEnabled">

          <wsdl:input wsaw:Action="http://www.nrf-arts.org/UnifiedPOS/Belt/SetDeviceEnabled"
              message="tns:Belt_SetDeviceEnabled_InputMessage" />

          <wsdl:output wsaw:Action="http://www.nrf-arts.org/UnifiedPOS/Belt/SetDeviceEnabledResponse"
              message="tns:Belt_SetDeviceEnabled_OutputMessage" />

          <wsdl:fault wsaw:Action="http://www.nrf-arts.org/UnifiedPOS/Belt/UposException" name="UposException"
              message="tns:Belt_SetDeviceEnabled_UposException_FaultMessage" />

        </wsdl:operation>

    ...
        <wsdl:operation name="CheckHealth">

          <wsdl:input wsaw:Action="http://www.nrf-arts.org/UnifiedPOS/Belt/CheckHealth"
               message="tns:Belt_CheckHealth_InputMessage" />

          <wsdl:output wsaw:Action="http://www.nrf-arts.org/UnifiedPOS/Belt/CheckHealthResponse"
               message="tns:Belt_CheckHealth_OutputMessage" />

          <wsdl:fault wsaw:Action="http://www.nrf-arts.org/UnifiedPOS/Belt/UposException" name="UposException"
               message="tns:Belt_CheckHealth_UposException_FaultMessage" />

        </wsdl:operation>

    ...
      </wsdl:portType>

    </wsdl:definitions>
/*---------- BeltV1.1.0.wsdl end ----------*/
```

WCF ツールである”svcutil.exe”を使い、以下の C#での WCF コントラクトが作成される：

/*--------- Belt.cs start ----------*/

```
        using System;
        using System.Collections.Generic;
        using System.Runtime.Serialization;
        using System.ServiceModel;
        using System.Text;

        namespace UnifiedPOS.Belt
        {
            [ServiceContract(Namespace = "http://www.nrf-arts.org/UnifiedPOS/Belt/")]
            public interface Belt
            {
                                ...

                [OperationContract(Action = "http://www.nrf-arts.org/UnifiedPOS/Belt/GetDeviceEnabled",
                    ReplyAction = "http://www.nrf-arts.org/UnifiedPOS/Belt/GetDeviceEnabledResponse")]

                [FaultContract(typeof(UposException), Action = "http://www.nrf-arts.org/UnifiedPOS/Belt/UposException", Name =
        "UposException")]
                    bool GetDeviceEnabled();

                [OperationContract(Action = "http://www.nrf-arts.org/UnifiedPOS/Belt/SetDeviceEnabled",
                    ReplyAction = "http://www.nrf-arts.org/UnifiedPOS/Belt/SetDeviceEnabledResponse")]

                [FaultContract(typeof(UposException), Action = "http://www.nrf-arts.org/UnifiedPOS/Belt/UposException", Name =
        "UposException")]
                    void SetDeviceEnabled(bool DeviceEnabled);


                                ...

                [OperationContract(Action = "http://www.nrf-arts.org/UnifiedPOS/Belt/CheckHealth",
                    ReplyAction = "http://www.nrf-arts.org/UnifiedPOS/Belt/CheckHealthResponse")]

                [FaultContract(typeof(UposException), Action = "http://www.nrf-arts.org/UnifiedPOS/Belt/UposException", Name =
        "UposException")]
                    void CheckHealth(HealthCheckLevel Level);


                                ...
            }


                                ...
            [DataContract(Namespace = "http://www.nrf-arts.org/UnifiedPOS/Belt/")]
```

```
        public enum HealthCheckLevel
          {
              [EnumMember]
                 External,
              [EnumMember]
                 Interactive,
              [EnumMember]
                 Internal,
          }

                    ...
    }
```

/*--------- Belt.cs end ----------*/

Java では JAX-WS ツールである”wsimport”を使って生成される JAX-WS コントラクトは、以下のようになる：

/*--------- Belt.java start ----------*/

```
    package org.nrf_arts.unifiedpos.belt;

    import javax.jws.WebMethod;
    import javax.jws.WebParam;
    import javax.jws.WebResult;
    import javax.jws.WebService;
    import javax.xml.ws.Action;
    import javax.xml.ws.FaultAction;
    import javax.xml.ws.RequestWrapper;
    import javax.xml.ws.ResponseWrapper;


    @WebService(name = "Belt", targetNamespace = "http://www.nrf-arts.org/UnifiedPOS/Belt/")
    public interface Belt
        {

                          ...
```

```
@Action(input = "http://www.nrf-arts.org/UnifiedPOS/Belt/GetDeviceEnabled",
       output = "http://www.nrf-arts.org/UnifiedPOS/Belt/GetDeviceEnabledResponse",
        fault =
          {
            @FaultAction(className = org.nrf_arts.unifiedpos.belt.FaultException.class,
                         value = "http://www.nrf-arts.org/UnifiedPOS/Belt/UposException")
          })

@WebMethod(operationName = "GetDeviceEnabled",
                  action = "http://www.nrf-arts.org/UnifiedPOS/Belt/GetDeviceEnabled")

@WebResult(name = "GetDeviceEnabledResult",
           targetNamespace = "http://www.nrf-arts.org/UnifiedPOS/Belt/")

@RequestWrapper(localName = "GetDeviceEnabled",
                targetNamespace = "http://www.nrf-arts.org/UnifiedPOS/Belt/",
                className = "org.nrf_arts.unifiedpos.belt.GetDeviceEnabled")

@ResponseWrapper(localName = "GetDeviceEnabledResponse",
                targetNamespace = "http://www.nrf-arts.org/UnifiedPOS/Belt/",
                className = "org.nrf_arts.unifiedpos.belt.GetDeviceEnabledResponse")

public Boolean getDeviceEnabled()
  throws FaultException;

@Action(input = "http://www.nrf-arts.org/UnifiedPOS/Belt/SetDeviceEnabled",
       output = "http://www.nrf-arts.org/UnifiedPOS/Belt/SetDeviceEnabledResponse",
        fault = {@FaultAction(className = org.nrf_arts.unifiedpos.belt.FaultException.class,
                  value = "http://www.nrf-arts.org/UnifiedPOS/Belt/UposException")
                 })

@WebMethod(operationName = "SetDeviceEnabled",
                  action = "http://www.nrf-arts.org/UnifiedPOS/Belt/SetDeviceEnabled")

@RequestWrapper(localName = "SetDeviceEnabled",
          targetNamespace = "http://www.nrf-arts.org/UnifiedPOS/Belt/",
                className = "org.nrf_arts.unifiedpos.belt.SetDeviceEnabled")

@ResponseWrapper(localName = "SetDeviceEnabledResponse",
           targetNamespace = "http://www.nrf-arts.org/UnifiedPOS/Belt/",
                 className = "org.nrf_arts.unifiedpos.belt.SetDeviceEnabledResponse")

public void setDeviceEnabled(
           @WebParam(name = "DeviceEnabled",
          targetNamespace = "http://www.nrf-arts.org/UnifiedPOS/Belt/")

       Boolean deviceEnabled)throws FaultException;
```

```
                    ...

        @Action(input="http://www.nrf-arts.org/UnifiedPOS/Belt/CheckHealth",
                output="http://www.nrf-arts.org/UnifiedPOS/Belt/CheckHealthResponse",
                 Fault={@FaultAction(className=org.nrf_arts.unifiedpos.belt.FaultException.class,
                          value="http://www.nrf-arts.org/UnifiedPOS/Belt/UposException")
                        })

        @WebMethod(operationName = "CheckHealth",
                           action = "http://www.nrf-arts.org/UnifiedPOS/Belt/CheckHealth")

        @RequestWrapper(localName = "CheckHealth",
                    targetNamespace = "http://www.nrf-arts.org/UnifiedPOS/Belt/",
                          className = "org.nrf_arts.unifiedpos.belt.CheckHealth")

        @ResponseWrapper(localName = "CheckHealthResponse",
                     targetNamespace = "http://www.nrf-arts.org/UnifiedPOS/Belt/",
                           className = "org.nrf_arts.unifiedpos.belt.CheckHealthResponse")
        public void checkHealth(
                     @WebParam(name = "Level",
                    targetNamespace = "http://www.nrf-arts.org/UnifiedPOS/Belt/")
                    HealthCheckLevel level)
                    throws FaultException;

                    ...
    }
/*--------- Belt.java end ----------*/
```

Because JAX-WS has no possibility to declare a DataContract as in C#, for WSDL input and output actions, separate classes are generated providing the XML data access:JAX-WS では C#での DataContract 宣言が行われないため、WSDL の入出力アクションに対しては、XML データへのアクセスと共に、別々のクラスが生成される。

/*--------- CheckHealth.java start ----------*/

```
    package org.nrf_arts.unifiedpos.belt;

    import javax.xml.bind.annotation.XmlAccessType;
    import javax.xml.bind.annotation.XmlAccessorType;
    import javax.xml.bind.annotation.XmlElement;
    import javax.xml.bind.annotation.XmlRootElement;
    import javax.xml.bind.annotation.XmlType;



    @XmlAccessorType(XmlAccessType.FIELD)
```

```
        @XmlType(name = "", propOrder = { "level" })

        @XmlRootElement(name = "CheckHealth")

        public class CheckHealth
           {

               @XmlElement(name = "Level")
               protected HealthCheckLevel level;

               public HealthCheckLevel getLevel()
                {
                   return level;
                }

               public void setLevel(HealthCheckLevel value)
                {
                   this.level = value;
                }
          }
```

/*--------- CheckHealth.java end ----------*/

しかしながら、CheckHealth メソッドはデータを返さないため、CheckHealthResponse クラスは空白となる：

/*--------- CheckHealthResponse.java start ----------*/

```
        package org.nrf_arts.unifiedpos.belt;

        import javax.xml.bind.annotation.XmlAccessType;
        import javax.xml.bind.annotation.XmlAccessorType;
        import javax.xml.bind.annotation.XmlRootElement;
        import javax.xml.bind.annotation.XmlType;


        @XmlAccessorType(XmlAccessType.FIELD)
        @XmlType(name = "")
        @XmlRootElement(name = "CheckHealthResponse")
        public class CheckHealthResponse
           {
           }
```

/*--------- CheckHealthResponse.java end ----------*/

/*-- Example End*/