Cover Page for the Japanese Translation

WS-I 利用シナリオ

Document Status: Final Specification

Version: 1.01

Date: March 25, 2004

Editors: Scott Werden, WRQ

Colleen Evans, Sonic Software

Marc Goodner, SAP

Translator(日本語訳):

SUZUKI Toshihiro, Oracle Corporation Japan / 鈴木俊宏（日本オラクル株式会社）

YAMAMOTO Yohei, Ricoh Company, Ltd. / 山本陽平（株式会社リコー）

MATSUNO Yohichiroh, Ricoh Company, Ltd. / 松野陽一郎（株式会社リコー）

Translation Reviewers (日本語訳レビュー):

FUJITA Satoru, NEC Corporation / 藤田　悟 (日本電気株式会社)

This document is a translation of a Web Services-Interoperability Organization document. In the event of a disagreement between this translation and the English version, or an omission in this translation, the original English document should be considered the definitive source.

この文書は Web Services-Interoperability Organization (WS-I) の文書の翻訳である。この翻訳と英語版との間に食い違いがあった場合，又は，翻訳に省略がある場合，原文の英語文書が決定版として扱われるべきである。

WS-I Usage Scenarios

Document Status: Final Specification

Version: 1.01

Date: December 9, 2003

Editors: Scott Werden, WRQ

Colleen Evans, Sonic Software

Marc Goodner, SAP

Notice(通告)

ここに含まれる内容は，この内容の著者若しくは開発者又は WS-I が所有又は管理する知的財産権に対するライセンスを，明示的にも暗示的にも示すものではない。ここに含まれる内容はそのまま (AS IS) の形式で提供され，適用可能な法律に許される最大限の範囲において，そのままの形で欠陥を含んだまま (AS IS AND WITH ALL FAULTS) 提供される。この内容の著者及び開発者並びに WS-I は，ここに，明示的，暗示的，又は法定による，市場性，特定目的への適用性，応答の精度若しくは完全性，結果，職人的努力，ウィルスがないこと，又は過失のないことについての，暗示的な保証，義務又は条件を含む (しかし，それに限定されない)，その他すべての保証及び条件を否認する。 さらに，この内容に関する権利，安全な居住，安全な所有，記述への対応又は侵害不在性について，いかなる保証又は条件も存在しない。

この内容の著者若しくは開発者又は WS-I のいずれも，他者に対して，損害の可能性の事前通告の有無にかかわらず，これ又はこの内容についてのその他のいかなる合意によって起こされる損害に対して，契約，不法行為，保証その他による，代替する物資若しくは服務の購入，逸失利益，用途損失，データ損害，又はいかなる偶発的，結果的，直接，間接，若しくは特別な損害を補償する責任を持たない。

Status of this Document(この文書の位置付け)

これは最終仕様 (final specification) である。正誤表又は更新版については，WS-I.org のサイトを参照のこと。

Table of Contents(目次)

Table of Figures(図目次)

Revision History(改定履歴)

Version 1.0 Original version

Version 1.01 Revisions to Board Approval Draft to reflect late changes to the Basic Profile 1.0

1 Introduction(はじめに)

WS-I 利用シナリオ（Usage Scenarios）は、構造化された基礎的な相互運用性の必要条件を識別し、WS-I Basic Profile 1.0 (以後 Basic Profile)[1]の必要条件にシナリオのフローを写像する、相互作用における Web サービスの活用を定義する。シナリオは任意のアプリケーションドメインに依存しない。WS-I Use Case は、特定のアプリケーションの高レベルの定義をモデル化するためにシナリオを使用する。

ここで示されているシナリオは再構成したり拡張したりすることができる。すなわち、これらのシナリオは、ビルディングブロックのように、組み合わせたり、その上に作り上げたりできる基礎的な Web サービスのデザイン・パターンについて記述している。例えば、同期リクエスト/レスポンスのシナリオは、基礎的な（訳注：メッセージの）交換について記述し、SOAP ヘッダを付加することにより拡張することができる。唯一の要求は、拡張も Basic Profile に必ず従わなければならないということである。

1.1 How to use this document(この文書の使い方)

この文書はBasic Profileと共に使用されるWS-Iの使用法のシナリオについて記述している。Basic Profileの制約および要件は直接参照され、読者は参照された情報を解釈するために、この文書を一緒に関連させてBasic Profileを使用することが期待されている。

この文書の中で示された三つのシナリオは、この文書の利用者がシナリオの中の一つ以上を利用した WS-I 準拠の Web サービスアプリケーションを作成することができるように、十分な情報を提供するように意図されている。各シナリオのメッセージおよびサービス記述インスタンスに適用可能なガイドラインと制限の全てが提供されている。

2 Usage Scenario Taxonomy(利用シナリオの分類)

利用シナリオの分類は Basic Profile の制約を適用するための仕組みを定義している。この分類は Web サービスのスタックと、このスタックの層ごとにグループ化され、Web サービスインスタンスが Web サービスの利用シナリオの一部として実行するアクティビティの集合から構成される。Basic Profile の制約は、シナリオの任意のコンポーネント（例えば、Web サービスインスタンスを記述するための WSDL）に加えてそれぞれのアクティビティにも適用される。ここにシナリオ上の二種類の制約がある。

- Web サービスのフローに関与するそれぞれのアクティビティに適用されるフローの制約。これらは XML 中の Web サービスデータモデルの表現、SOAP を利用したメッセージの生成と使用、HTTP を使ったメッセージトランスポートを含んでいる。
- シナリオの記述に適用される記述制約。操作上、Web サービスインスタンスの記述は WSDL そしておそらく UDDI の中に存在する。それゆえ、それらの制約はシナリオを記述した WSDL と UDDI に適用される。

以下に WS-I の利用シナリオの特性を示す。

- それらはシナリオに特有のアクティビティの集まりが共に関連したフロー記述を含み、
- SOAP ヘッダやセキュリティのような任意のコンポーネントを含み、
- WSDL 文書で記述され、
- シナリオ内のそれぞれのアクティビティは Basic Profile によって適用された制約を持ち
- そして実世界の Web サービスの実装を表している。

2.1 Web Service Stack(Web サービススタック)

利用シナリオの分類は Web サービススタックに基づいている。スタックのそれぞれの層は Web サービスインスタンスの基本的な機能的な範囲のうちの一つを表わす。全ての可能な機能的な範囲（例えば、セキュリティや連携）が表現されているわけでなく、最も基本的なものだけである。以下の図にこれらの層を示す。

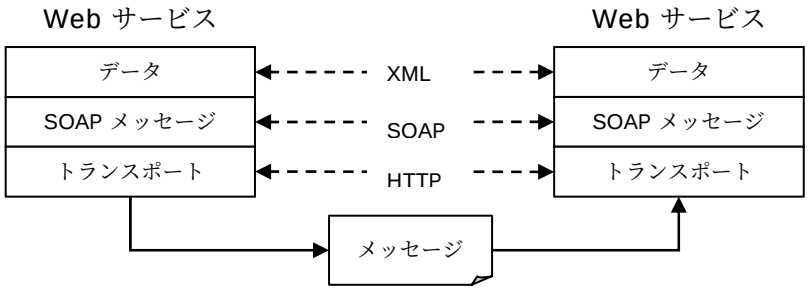

図 2-1 Web サービススタック

Webサービスアプリケーションは、Webサービスインスタンスおよびアプリケーション・ビジネスロジックのようないくつかの論理層を組込む機能を含んでいるかもしれない。Basic Profileおよび利用シナリオはビジネスロジックの中でWebサービススタックの任意の部分の機能性を除いたアプリケーション・ビジネスロジックを扱わない。
Web サービススタックにおけるそれぞれの層の詳細を以下に示す。

データ層

データ層は特定のアプリケーション固有のデータを Web サービスのために選択されたモデルに変換する。データ層は柔軟なデータ型をサポートするのに必要な機能を含んでいる。この層は WSDL 中の `wsdl:types` と `wsdl:message` 定義にマップされる。

SOAP メッセージ層

SOAP メッセージ層は SOAP メッセージを処理し、それらをディスパッチするインフラストラクチャ層である。この層はサービスに要求される質を満たす機能を持つかもしれない。送信側では、メッセージ層が portType とバインディングに定義されたデータモデルに基づいて SOAP メッセージを書き出す。受信側では、メッセージ層が SOAP メッセージを処理し、リクエストを正しいアプリケーションやメソッドにディスパッチする。

トランスポート層

トランスポート層はメッセージを送受信する。Basic Profile に関しては、これは HTTP クライアントおよびサーバー・プラットフォームだけを含んでいる。この層は WSDL 文書中の `wsdl:binding` と `wsdl:port` の定義にマップされる。

2.2 Activities(アクティビティ)

アクティビティの集合は Web サービススタックのそれぞれの層に定義されている。アクティビティは Web サービスを構成する基礎的なオペレーションである。一つのアクティビティは Basic Profile からそれに対して適用されるいくつかの制約を持つ。たとえば、一つのアクティビティを "Send HTTP" とすると、このアクティビティをどのように実現するかのための仕様とガイドラインは、Basic Profileの SOAP 1.1 と HTTP の節に由来する。
以下の表にアクティビティの要約を示す。

層	アクティビティ
データ層	XML の書き出し XML の処理
SOAP メッセージ層	SOAP エンベロープの書き出し SOAP エンベロープの処理 SOAP ボディの書き出し SOAP ボディの処理 SOAP ヘッダの書き出し SOAP ヘッダの処理
トランスポート層	HTTP の送信 HTTP の受信

表 1 – Web サービススタック層によってグループ化されたアクティビティ

2.2.1 Data Layer Activities(データ層アクティビティ)

以下に示すアクティビティはデータ層の一部である:

XMLの書き出し

Web サービスインタラクションの間で交換されるアプリケーションレベルメッセージは下位のトランスポートプロトコルでトランスポート可能な直列化された形式で書き出されなければならない。これらのメッセージはデータモデルの資料(たとえば、WSDL やスキーマ)で宣言されたデータ型と形式を使う。メッセージデータの書き出しはメッセージを受信者に送信するアプリケーションコンポーネントの責任である。

XMLの処理

Web サービスインタラクションの間で交換されるアプリケーションレベルメッセージは、メッセージを受信し、解釈し、受信したメッセージに従って動作する責任のあるアプリケーションコンポーネントに渡されなければならない。アプリケーションコンポーネントはデータモデル文書で宣言された型とフォーマットに従ってメッセージデータを処理する。

2.2.2 SOAP Message Layer Activities(SOAP メッセージ層アクティビティ)

以下に示すアクティビティは SOAP メッセージ層によって実行される。

SOAP エンベロープ

SOAP エンベロープはペイロードを含んだ、その他の SOAP メッセージ部のコンテナである。

- SOAP エンベロープの書き出し
- SOAP エンベロープの処理

SOAP ボディ

SOAP Body はアプリケーションメッセージデータを含むアプリケーション固有の情報のトランスポートに使われる。この層のアクティビティは、データペイロードの書き出しとデータ層アクティビティ節で記述したアクティビティの処理とは異なる。

- SOAP ボディの書き出し
- SOAP の処理

SOAP ヘッダ

SOAP ヘッダは SOAP メッセージを拡張するモジュール機構を提供する。

- SOAP ヘッダの書き出し
- SOAP ヘッダの処理

2.2.3 Transport Layer Activities(トランスポート層アクティビティ)

SOAP メッセージは HTTP または HTTPS トランスポートプロトコルを使って送信されることがある。

- HTTP の送信
- HTTP の受信

2.2.4 Web Service Actors(Web サービスアクタ)

WS-I Web サービスシナリオでは二つの高レベルアクタが存在する。これは SOAP 1.1 で定義されている SOAP Actor とは関係ない。

コンシューマ(Consumer)

コンシューマはプロバイダによって実装されたサービスのリクエストを作成することに責任を負う。

プロバイダ(Provider)

プロバイダはコンシューマのサービスリクエストを待ち、処理することに責任を負う。

2.3 Security(セキュリティ)

利用シナリオは認証、認可、識別、プライバシについて明示的に扱わない。しかしながら、それらの事柄のうちいくつかは Basic Profile と互換性をもつ既存の技術で扱うことができる。たとえば、暗号化されていない HTTP バインディングでなく HTTPS バインディングを使うことができる。アプリケーションレベルのセキュリティは常にメッセージ層の中に加えることができる。また、これは Basic Profile に完全に透過である。

防衛手段は Web サービスアプリケーションのリスクアセスメントを通して最大に適用される。このプロセスを補助するために、共通の脅威と Basic Profile 準拠の緩和戦略の詳細情報については、後半の Security Appendix を見てほしい。それぞれの利用シナリオは、与えられたシナリオに最も適切な関連した共通の脅威と同様に、追加の事柄も記述する節を含んでいる。

3 Usage Scenarios(利用シナリオ)

この節は、Basic Profile を補足するために開発された以下の三つの利用シナリオを定義する。

- 一方向(One-way)
- 同期リクエスト/レスポンス(Synchronous request/response)
- 基本コールバック(Basic callback)

3.1 One-way(一方向)

3.1.1 Description(概要)

コンシューマはプロバイダにリクエストメッセージを送信する。プロバイダはそのメッセージを受信し、処理する。

交換は一方向である。プロバイダからの SOAP レスポンスメッセージは生成または期待されない。下位トランスポートはプロバイダへのメッセージの配信保証を要求しない。トランスポート層によって実装されるプロトコルにかかわらず、コンシューマはメッセージが送信に成功したこと、意図した目的地に到達されたことや、プロバイダによって受信されたことを示すトランスポート層の上の受取り通知を受信しない。

このシナリオは情報損失が重要でない状況へ適用される(たとえば、周期的な状態更新イベントで、もし一つの更新を紛失しても、次の更新イベントが正しい状態を伝えるように提供されている状態監視シナリオ)。

図 3-1 一方向シーケンス

高レベルフロー:

1. コンシューマがプロバイダに向けて HTTP リクエストに束縛された SOAP メッセージを送信することでサービスを呼び出す
2. プロバイダがサービスを実行する。

条件:

- このシナリオはイベントの実行時シーケンスを記述しており、設計または配備のアクティビティについて記述しない。
- データモデル、アプリケーションのセマンティクス、トランスポートバインディングはこのシナリオに先立って全て合意されており、実装されている。
- このシナリオの全ての部分は Basic Profile のガイドラインと勧告に適合して定義されている。
- このシナリオは「構成可能」である。すなわち、このシナリオはより複雑なシナリオの作成のために基礎として使用されてもよい。

3.1.2 Flow(フロー)

2.2 節中で定義されたアクティビティを使用するこのシナリオのためのフローの詳細を、下記に記述する。各項目はフローを完了するために必用な、スタックの一つの層で行われるアクティビティを表現する。コンシューマまたはプロバイダ内のアクティビティの順序は重要ではない。各アクティビティは、Basic Profile から課せられた制約を持つ。

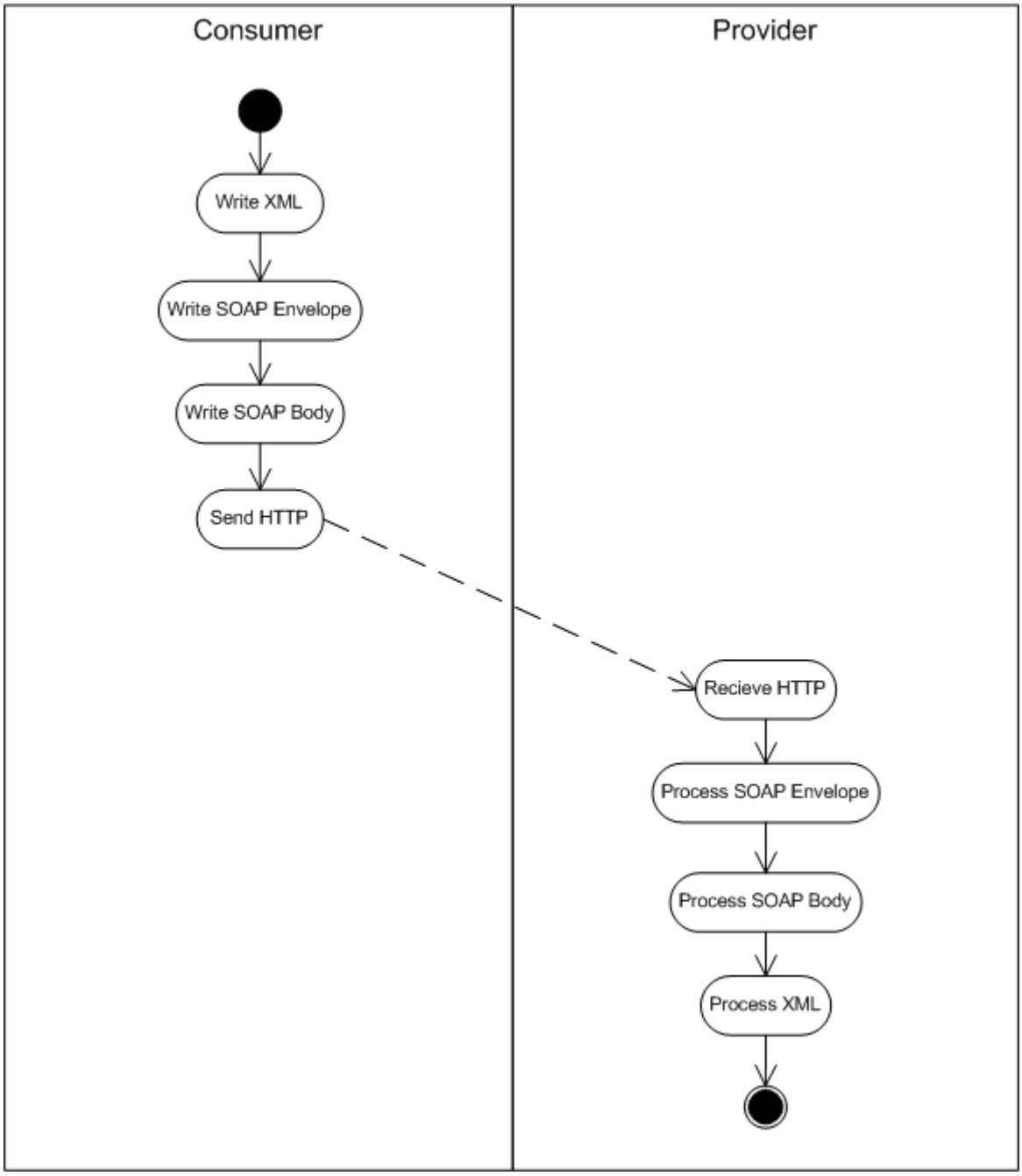

図 3-2 一方向リクエスト

コンシューマが SOAP リクエストを開始する:

- データ層
  - XML を書き出す。データモデルに従ってペイロードが作成される。
- SOAP メッセージ層
  - SOAP エンベロープを書き出す
  - SOAP ボディを書き出す

- トランスポート層
    - HTTP を送信する

プロバイダが SOAP リクエストを受信する:

- トランスポート層
    - HTTP を受信する
- SOAP メッセージ層
    - SOAP エンベロープを処理する
    - SOAP ボディを処理する
- データ層
    - XML を処理する。データモデルにしたがってデータペイロードが処理され、アプリケーションにディスパッチされる

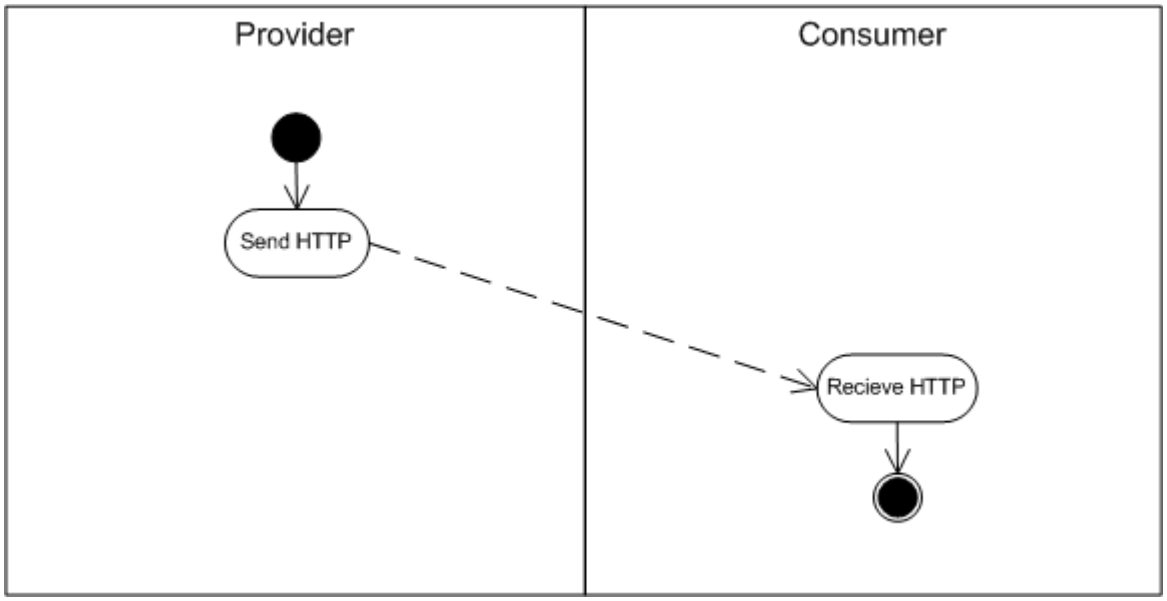

図 3-3 一方向受け取り通知

プロバイダが受信確認を送信する。

- トランスポート層
    - HTTP を送信する。 HTTP レスポンスは SOAP メッセージの処理に呼応していつでも送信されることができることに注意すること。HTTP レスポンスに SOAP エンベロープが含まれることはない。

コンシューマが受信確認を受信する。

- トランスポート層
    - HTTP を受信する(状態コード)。これは Web サービススタックの上の層では無視される。

3.1.3 Flow Constraints(フロー制約)

このシナリオのフロー制約を以下に示す。

- 5.1 節で定義されているように、XML を書き込む

- 5.3 節で定義されているように、SOAP エンベロープを書き込む
- 5.5 節で定義されているように、SOAP ボディを書き込む
- 5.9 節で定義されているように、HTTP を送信する。 このシナリオ特有のさらなる制約は R2714 である。
- 5.10 節で定義されているように HTTP を受信する。 このシナリオ特有のさらなる制約は R2727 と R2750 である。
- 5.4 節で定義されているように、SOAP エンベロープを処理する
- 5.6 節で定義されているように、SOAP ボディを処理する
- 5.2 節で定義されているように、XML を処理する

3.1.3.1 Error conditions and SOAP Fault(エラー条件と SOAP Fault)

SOAPレスポンスメッセージがないので、SOAP Faultはこのシナリオでは生成されない。もしプロバイダでエラーが起きた場合は、プロバイダとコンシューマはそれぞれR2714を固守しなければならない。

3.1.3.2 SOAP Headers(SOAP ヘッダ)

このシナリオでは、SOAP ヘッダの使用は任意である。もし使用する場合、5.7 節と 5.8 節の中でそれぞれ定義されるように、SOAP ヘッダの書き込みと、SOAP ヘッダの処理アクティビティのための制約に従わなければならない。

3.1.4 Description Constraints(記述の制約)

WSDL は一方向シナリオの定義内に少なくとも以下の内容を持っているべきである。全ての節が要求されるとは限らない。しかし WSDL 中にある場合、それぞれは下に示されるようなガイドラインに従うべきである。WSDL の一般的な制約は 5.11 節に記述されている。Basic Profile によって課されたその他の制約は以下に一覧される。

3.1.4.1 types

この節は要求されない。もし存在する場合は、データモデルの詳細に依存するであろう。

WSDL の types についての制約は 5.12 節に挙げられている。

3.1.4.2 messages

データモデル(doc/literal または rpc/literal)に依存している。ただ一つのメッセージ(一つの入力)が定義されている。

3.1.4.2.1Document messages(ドキュメントメッセージ)

ドキュメントメッセージ部はスキーマ要素定義から構成される(R2204 を見よ)

```
<wsdl:message …>
        <wsdl:part name=”Input” element=”..”>
</wsdl:message>
```

3.1.4.2.2RPC messages(RPC メッセージ)

RPC メッセージ部はスキーマ型宣言から構成される(R2203 を見よ)

```
<wsdl:message …>
        <wsdl:part name=”Input“ type=”..” />
```

```
</wsdl:message>
```

メッセージの制約は 5.13 節に挙げられている。

3.1.4.3 portTypes

一方向送信プリミティブが使われなければならない。文法を以下に示す。

```
<wsdl:portType ..>
        <wsdl:operation …>
                <wsdl:input …/>
        </wsdl:operation>
</wsdl:portType>
```

その他の制約は 5.14 節に挙げられている。

3.1.4.4 binding

`wsdl:binding` 節は HTTP トランスポートを伴った SOAP バインディング拡張を使わなければならない。`wsdl:portType` 中で定義された同じオペレーション型がバインディング節で使われなければならない。

```
<wsdl:binding …>
        <soap:binding style=”rpc|document” transport=http://schemas.xmlsoap.org/soap/http>
                <wsdl:input …>
                </wsdl:input>
        </soap:binding>
</wsdl:binding>
```

その他の制約は 5.15 節に挙げられている。

3.1.4.5 port

`soap:address` 要素はエンドポイントの URL と共に明記されなければならない。

```
<wsdl:port>
        <soap:address location=”uri” />
</wsdl:port>
```

その他の制約は 5.16 節に挙げられている。

3.1.5 UDDI

このシナリオでパターン化された Web サービスの広告はベストプラクティス文書 “Using WSDL in a UDDI Registry, Version 1.07” を踏襲する。`uddi:tModel` はメッセージオペレーションのための `wsdl:binding` を含むファイルを参照する Web サービスの型を表現している。`uddi:bindingTemplate` はサービスエンドポイントを保存し、Web サービスタイプ用の `uddi:tModel(s)` を参照する。

この方法による Web サービスの広告はUDDI Inquiry API セットによってサポートされる紹介パターン(http://www.uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.pdf を見よ)を使ったディスカバリを可能にする。これらはブラウズ(browse)パターン、ドリルダウン(drill-down)パターン、インボケーション(invocation)パターンを含む。

一般的な UDDI の制約は 5.17 節に挙げられている。

3.1.6 Security(セキュリティ)

この節では付録1(Basic Profile 準拠の対抗策に関する追加の情報があるかもしれない)で定義されているこの利用シナリオに最も関連する脅威を確認する。
セキュリティを実装するために HTTPS が使われる場合、追加の制約を適用するかもしれない。これらは Profile 要件: R5000, R5001, R5010 である。付録1はセキュリティのための追加のガイドラインを含む。
この段階で、特定の脅威を書かないことは、この利用シナリオに非常に適切なこととして認識された。

3.2 Synchronous Request/Response(同期リクエスト/レスポンス)

3.2.1 Description(概要)

コンシューマがリクエストメッセージをプロバイダに送信する。プロバイダはそのメッセージを受信し、処理し、レスポンスを返信する。

以下の図は、同期リクエスト/レスポンス利用シナリオにおけるコンシューマとプロバイダの間の高レベルなインタラクションを示している。

図 3-4 同期リクエスト/レスポンスシーケンス

高レベルフロー:

1. コンシューマがプロバイダに向けて HTTP リクエストに束縛された SOAP メッセージを送信することでサービスを呼び出す
2. プロバイダがサービスを実行し、コンシューマに向けて HTTP レスポンスに束縛された SOAP メッセージを送信する

条件:

1. このシナリオはイベントの実行時シーケンスであり、設計または配備のアクティビティを伴わない。
2. データモデル、アプリケーションのセマンティクス、トランスポートバインディングはこのシナリオに先立って全て合意されており、実装されている。

3. このシナリオは「構成可能」である。すなわち、このシナリオはより複雑なシナリオの作成のために基礎として使用されてもよい。
4. リクエストメッセージとレスポンスメッセージは HTTP を通して同期している。

3.2.2 Flow(フロー)

2.2 節中で定義されたアクティビティを使用するこのシナリオのためのフローの詳細を下記に記述する。各項目はフローを完了するために必用な、スタックの一つの層で行われるアクティビティを表現する。コンシューマまたはプロバイダ内のアクティビティの順序は重要ではない。各アクティビティは、Basic Profile から課せられた制約を持つ。

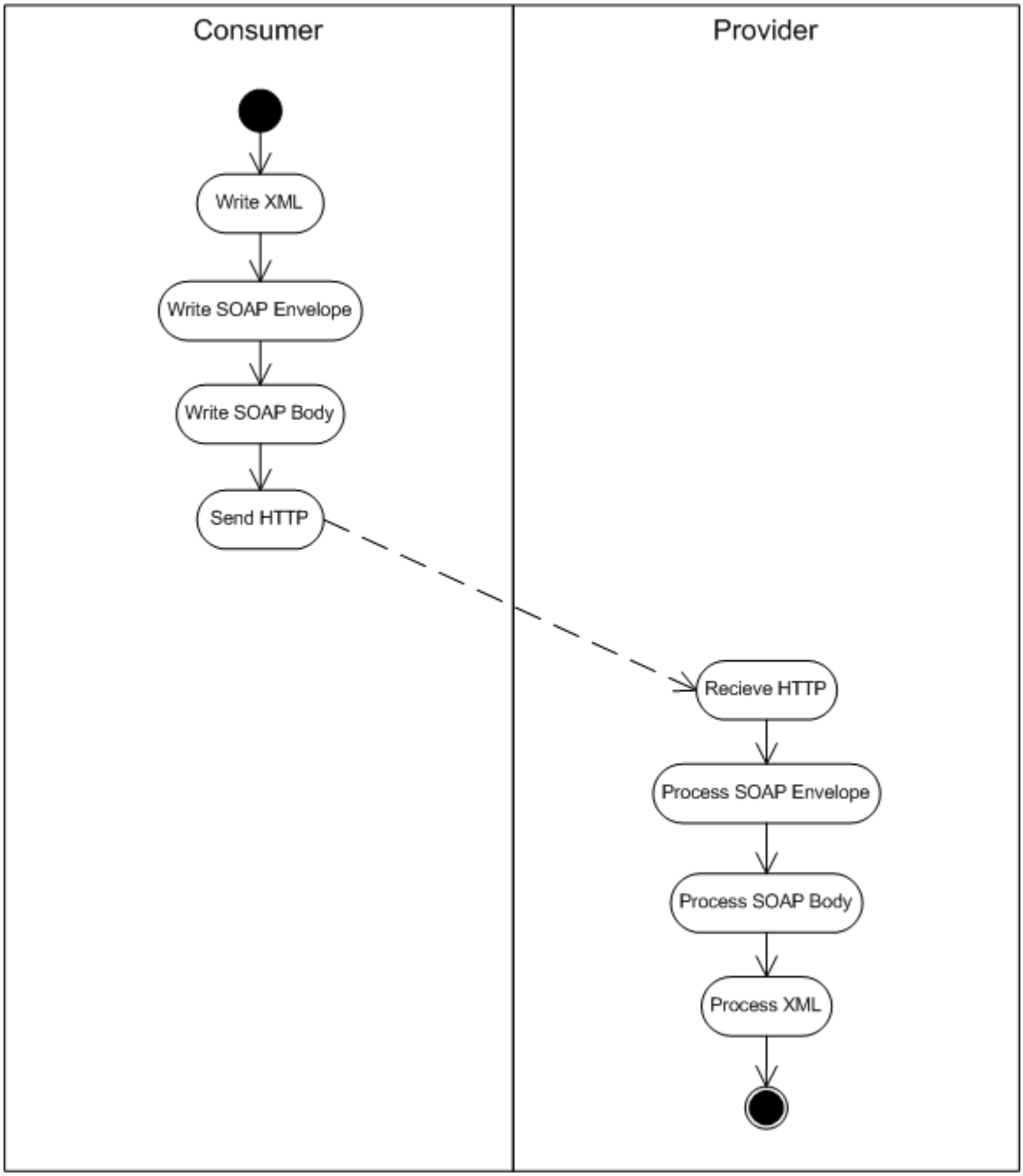

図 3-5 同期リクエスト

コンシューマが SOAP リクエストを開始する:

- データ層
    - XML を書き出す。データモデルに従ってペイロードが作成される。
- SOAP メッセージ層
    - SOAP エンベロープを書き出す
    - SOAP ボディを書き出す
- トランスポート層
    - HTTP を送信する

プロバイダが SOAP リクエストを受信する:

- トランスポート層
    - HTTP を受信する
- SOAP メッセージ層
    - SOAP エンベロープを処理する
    - SOAP ボディを処理する
- データ層
    - XML を処理する。データモデルにしたがってデータペイロードが処理され、アプリケーションにディスパッチされる

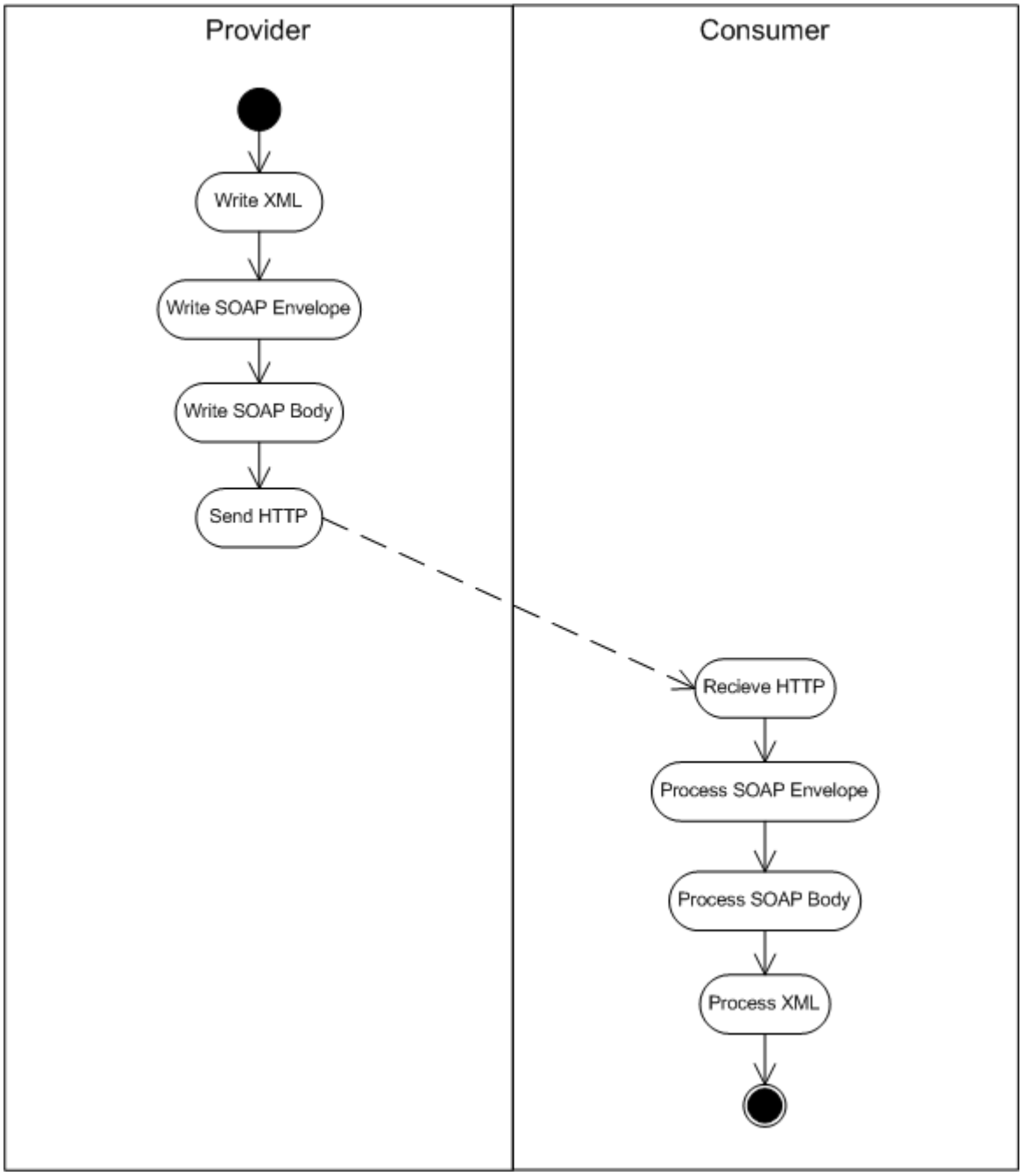

図 3-6 同期レスポンス

プロバイダが SOAP レスポンスを生成する

- データ層
  - XML を書き出す。データモデルに従ってペイロードが作成される。
- SOAP メッセージ層
  - SOAP エンベロープを書き出す
  - SOAP ボディを書き出す

- トランスポート層
    - HTTP を送信する

コンシューマが SOAP レスポンスを受信する。

- トランスポート層
    - HTTP を受信する
- SOAP メッセージ層
    - SOAP エンベロープを処理する
    - SOAP ボディを処理する
- データ層
    - XML を処理する。データモデルにしたがってデータペイロードが処理され、アプリケーションにディスパッチされる

3.2.3 Flow Constraints(フロー制約)

下記のアクティビティはこの利用シナリオで参照された制約を持っている。

- 5.1 節で定義されているように、XML を書き込む
- 5.3 節で定義されているように、SOAP エンベロープを書き込む
- 5.5 節で定義されているように、SOAP ボディを書き込む
- 5.9 節で定義されているように、HTTP を送信する
- 5.10 節で定義されているように HTTP を受信する
- 5.4 節で定義されているように、SOAP エンベロープを処理する
- 5.6 節で定義されているように、SOAP ボディを処理する
- 5.2 節で定義されているように、XML を処理する

3.2.3.1 Errors and SOAP Faults(エラーと SOAP Fault)

SOAP 処理中に起きるエラーは SOAP 1.1 仕様による SOAP Fault メッセージによって伝えられる。このシナリオは SOAP Fault をサポートする。これは、3.2.3 節と 3.3.4.1 節で定義された全ての制約が適用され、追加の制約が以下のアクティビティに課されるということである。

Web サービスプロバイダは Basic Profile の以下の制約とガイドラインを守らなければならない:

- Fault 生成の振る舞い: R2724, R2725
- `soap:fault` の書き出し: R1000, R1001, R1004, R1031, R2742, R2743
- HTTP SOAPAction: R1119
- HTTP 状態コード: R1126
- WSDL 記述への要件: R2728, R2742, R2743, R2754

Web サービスコンシューマは Basic Profile の以下の制約とガイドラインに従わなければならない:

- soap:fault の処理: R1002, R1003, R1016

3.2.3.2 SOAP Headers(SOAP ヘッダ)

このシナリオでは、SOAP ヘッダの使用は任意である。 もし使用する場合、5.7 節と 5.8 節の中でそれぞれ定義されるように、SOAP ヘッダの書き込みと、SOAP ヘッダの処理アクティビティのための制約に従わなければならない。

3.2.4 Description Constraints(記述の制約): WSDL

WSDL は同期リクエスト/レスポンスシナリオの定義内に少なくとも以下の内容を持っているべきである。全ての節が要求されるとは限らない。しかし WSDL 中にある場合、それぞれは下に示されるようなガイドラインに従うべきである。WSDL の一般的な制約は 5.11 節に記述されている。Basic Profile によって課されたその他の制約は以下に一覧される。

3.2.4.1 types

この節は要求されない。もし存在する場合は、データモデルの詳細に依存するであろう。

WSDL の types についての制約は 5.12 節に挙げられている。

3.2.4.2 messages (メッセージ)

データモデル(doc/literal または rpc/literal)に依存している。少なくとも二つのメッセージ(一つの入力と一つの出力)が定義されていなければならない。任意に、Fault メッセージも定義されるかもしれない。

3.2.4.2.1Document messages(ドキュメントメッセージ)

ドキュメントメッセージ部はスキーマ要素定義から構成される(R2204 を見よ)

```
<wsdl:message …>
        <wsdl:part name=”..” element=”..”>
        <wsdl:part name=”..” element=”..”/>
</wsdl:message>
```

3.2.4.2.2RPC messages(RPC メッセージ)

RPC メッセージ部はスキーマ型宣言から構成される(R2203 を見よ)

```
<wsdl:message …>
        <wsdl:part name=” “ type=”..” />
        <wsdl:part name=” “ type=”..”/>
</wsdl:message>
```

メッセージの制約は 5.13 節に挙げられている。

3.2.4.3 portTypes

リクエスト/レスポンスプリミティブが使われなければならない。文法を以下に示す。

```
<wsdl:portType ..>
        <wsdl:operation …>
                <wsdl:input …/>
                <wsdl:output …/>
```

```
        </wsdl:operation>
</wsdl:portType>
```

portType についての制約は 5.14 節に挙げられている。

3.2.4.4 binding

wsdl:binding 節は HTTP トランスポートを伴った SOAP バインディング拡張を使わなければならない。wsdl:portType 中で定義された同じオペレーション型がバインディング節で使われなければならない。

```
<wsdl:binding …>
        <soap:binding style=”rpc|document” transport=http://schemas.xmlsoap.org/soap/http>
                <wsdl:input …>
                </wsdl:input>
                <wsdl:output …>
                </wsdl:output>
        </soap:binding>
</wsdl:binding>
```

バインディングについての制約は 5.15 節に挙げられている。

3.2.4.5 port(ポート)

soap:address 要素はエンドポイントの URL と共に明記されなければならない。

```
<wsdl:port>
        <soap:address location=”uri” />
</wsdl:port>
```

ポート定義についての制約は 5.16 節に挙げられている。

3.2.5 UDDI

このシナリオでパターン化された Web サービスの広告はベストプラクティス文書 “Using WSDL in a UDDI Registry, Version 1.07” を踏襲する。uddi:tModel はメッセージオペレーションのための wsdl:binding を含むファイルを参照する Web サービスの型を表現している。uddi:bindingTemplate はサービスエンドポイントを保存し、Web サービスタイプ用の uddi:tModel(s) を参照する。

この方法による Web サービスの広告はUDDI Inquiry API セットによってサポートされる紹介パターン(http://www.uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.pdf を見よ)を使ったディスカバリを可能にする。これらはブラウズ(browse)パターン、ドリルダウン(drill-down)パターン、インボケーション(invocation)パターンを含む。

一般的な UDDI の制約は 5.17 節に挙げられている。

3.2.6 Security(セキュリティ)

この節では付録1(Basic Profile 準拠の対抗策に関する追加の情報があるかもしれない)で定義されているこの利用シナリオに最も関連する脅威を確認する。
セキュリティを実装するために HTTPS が使われる場合、追加の制約を適用するかもしれない。これらは Profile 要件: R5000, R5001, R5010 である。付録1はセキュリティのための追加のガイドラインを含む。
この段階で、特定の脅威を書かないことは、この利用シナリオに非常に適切なこととして認識された。

3.3 Basic Callback(基本コールバック)

3.3.1 Description(概要)

基本コールバックシナリオは、Web サービスのための非同期メッセージ交換の形式を実現する。これは、2 つの同期リクエスト/レスポンスペアの構成によって遂行される。メッセージは、コンシューマによって提供される相関性情報によって関連づけられる。コンシューマは、さらに、プロバイダにコールバックサービスの位置のエンドポイント情報も提供する。コールバックサービスの定義は公表された Web サービス記述の中でプロバイダによって定義され、コンシューマによって実装される。

実行時に、コンシューマは初期 SOAP リクエストをリクエスト/レスポンスシーケンスでプロバイダに送信し、プロバイダは次に即時の受取通知を送り返す。その後に、プロバイダはコンシューマによって送信された初期リクエストへの返答データを含んだ最終リクエスト/レスポンスシーケンスを開始する。以下の図は基本コールバックシナリオのコンシューマとプロバイダの間の高レベルなインタラクションを示している。

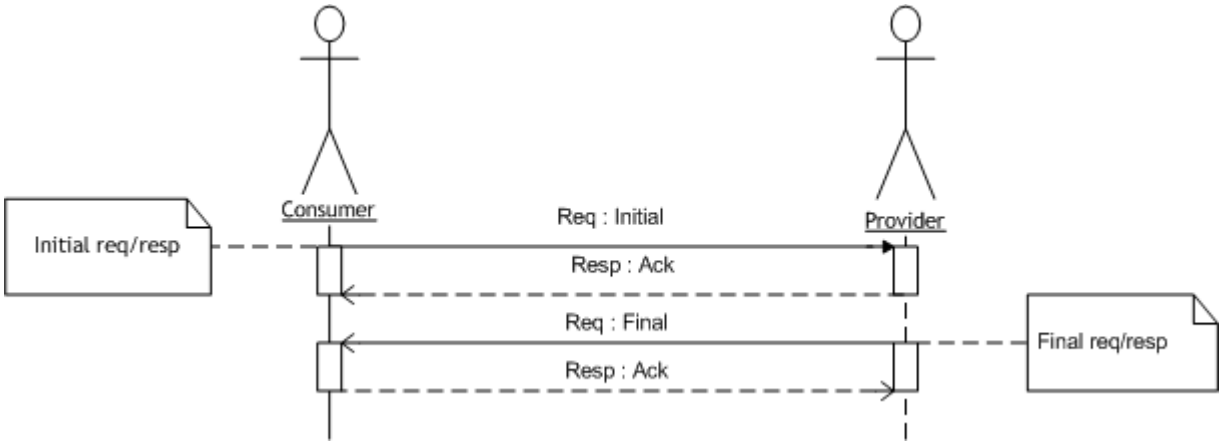

図 3-7 基本コールバックシーケンス

高レベルフロー:

1. コンシューマが HTTP リクエストにバインドした SOAP メッセージをプロバイダに送信することでサービスを開始する(“初期リクエスト”)
2. プロバイダが HTTP レスポンスにバインドした SOAP メッセージで受取りをコンシューマに通知する(“初期レスポンス”)
3. プロバイダが HTTP リクエストにバインドした初期リクエストの結果を含む SOAP メッセージをコンシューマに送信することで交換を完了する(“最終リクエスト”または”コールバック”)
4. コンシューマがコールバックメッセージを HTTP レスポンスにバインドした SOAP メッセージで受取りを通知する(“最終レスポンス”)

条件:

- このシナリオはイベントの実行時シーケンスであり、設計または配備のアクティビティを伴わない。
- データモデル、アプリケーションのセマンティクス、コールバック相関機構、トランスポートバインディングはこのシナリオに先立って全て合意されており、実装されている。
- このシナリオの全ての部分は Basic Profile のガイドラインと勧告に適合して定義されている。
- このシナリオは「構成可能」である。すなわち、このシナリオはより複雑なシナリオの作成のために基礎として使用されてもよい。

3.3.2 Details(詳細)

基本コールバックシナリオは二つの同期リクエスト/レスポンスインタラクションで構成されている。コンシューマとプロバイダは多くの未解決のリクエストを持っているかもしれないので、双方でどのコールバックがどの初期リクエストに対応するのかを確認する機構が必要になる。これはコールバックとリクエストの相互関係を示すために使うことのできる SOAP ペイロード中の発注番号などのビジネスデータ、またはメッセージ ID などを使うことで達成できる。

コールバックサービスを発動するために、コンシューマはコールバックエンドポイントでプロバイダと通信しなければならない。コールバックエンドポイントは、実行時に最初の SOAP メッセージに入れてコンシューマからプロバイダに送信したり、デプロイ時に送信したり、双方が合意しているディスカバリ機構を使って送信したりすることができる。

最初と最後のリクエスト/レスポンスのための Web サービス定義(すなわち portTypes)はひとつの WSDL 文書で定義してもよい。これは必須ではないが、それらを同じ文書におくことはクライアントの契約と期待をより効果的に伝える。二つの企業が単一の文書を保守したくないようなゆるやかに結合した場面では、最初と最後のリクエスト/レスポンスの組は分割された WSDL 文書で定義される必用がある。どちらの場合でも、プロバイダとコンシューマの Web サービスの portType が定義される必用がある。記述はプロバイダ Web サービスのポートを含んでもよい(MAY)。そして、コンシューマ Web サービスを定義したポートは含まない(DOES NOT)。最終サービスのアドレスはあらかじめわからないので、最終リクエスト/レスポンスの portType の WSDL port は定義されない。これは、その代りとして前に記述したコンシューマによって伝達される。

3.3.3 Flow(フロー)

2.2 節中で定義されたアクティビティを使用するこのシナリオのためのフローの詳細を下記に記述する。各項目はフローを完了するために必用な、スタックの一つの層で行われるアクティビティを表現する。コンシューマまたはプロバイダ内のアクティビティの順序は重要ではない。各アクティビティは、Basic Profile から課せられた制約を持つ。

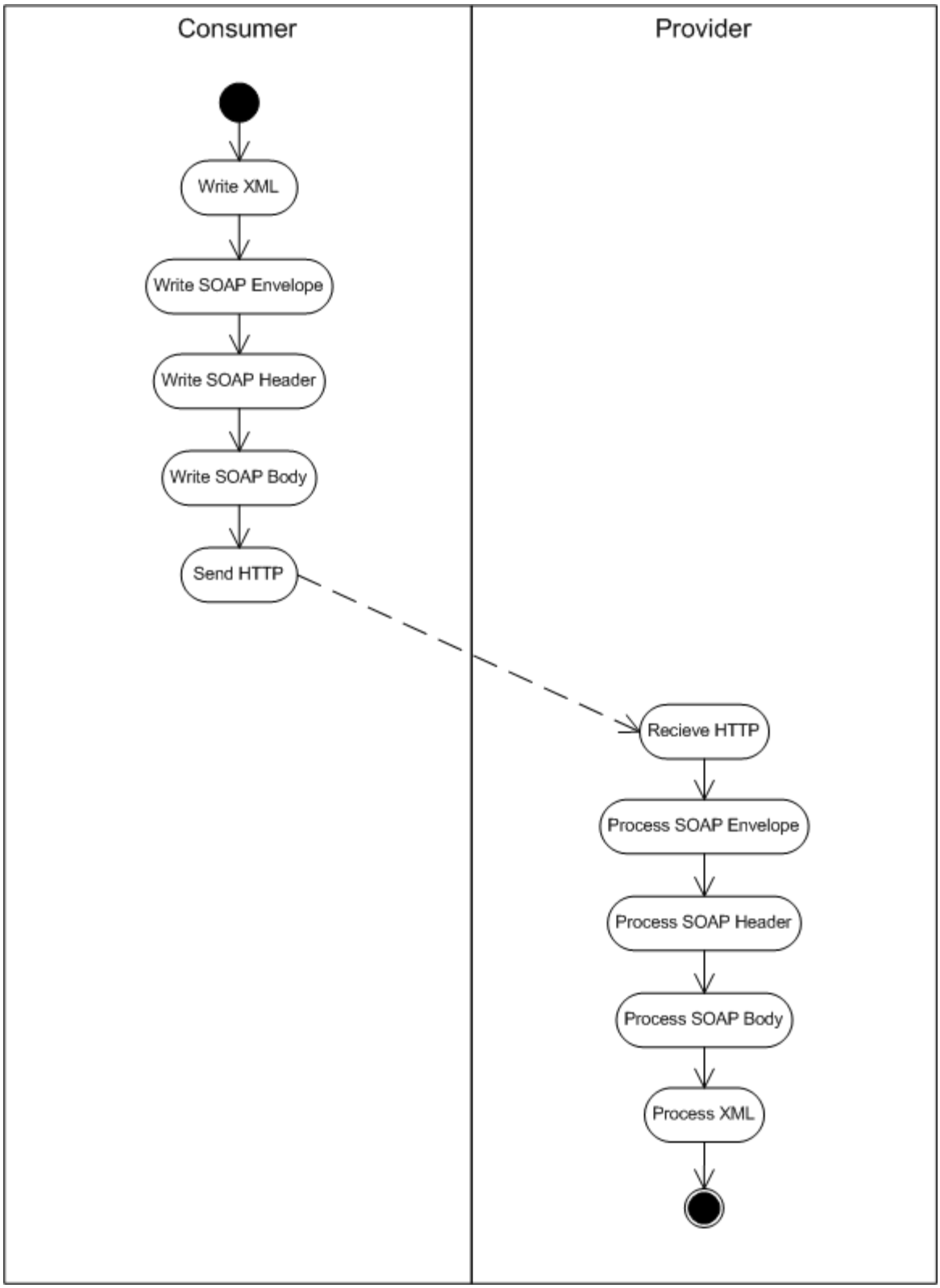

図 3-8 基本コールバックコンシューマリクエスト

コンシューマが SOAP リクエストを開始する:

- データ層
    - XML を書き出す。データモデルに従ってペイロードが作成される。
- SOAP メッセージ層
    - SOAP エンベロープを書き出す
    - [SOAP ヘッダを書き出す (相関性情報が SOAP ヘッダで伝えられる場合)]
    - SOAP ボディを書き出す
- トランスポート層
    - HTTP を送信する

プロバイダが初期 SOAP リクエストを受信する。

- トランスポート層
    - HTTP を受信する
- SOAP メッセージ層
    - SOAP エンベロープを処理する
    - [SOAP ヘッダを処理する (相関性情報が SOAP ヘッダで伝えられる場合)]
    - SOAP ボディを処理する
- データ層
    - XML を処理する。データモデルにしたがってデータペイロードが処理され、アプリケーションにディスパッチされる

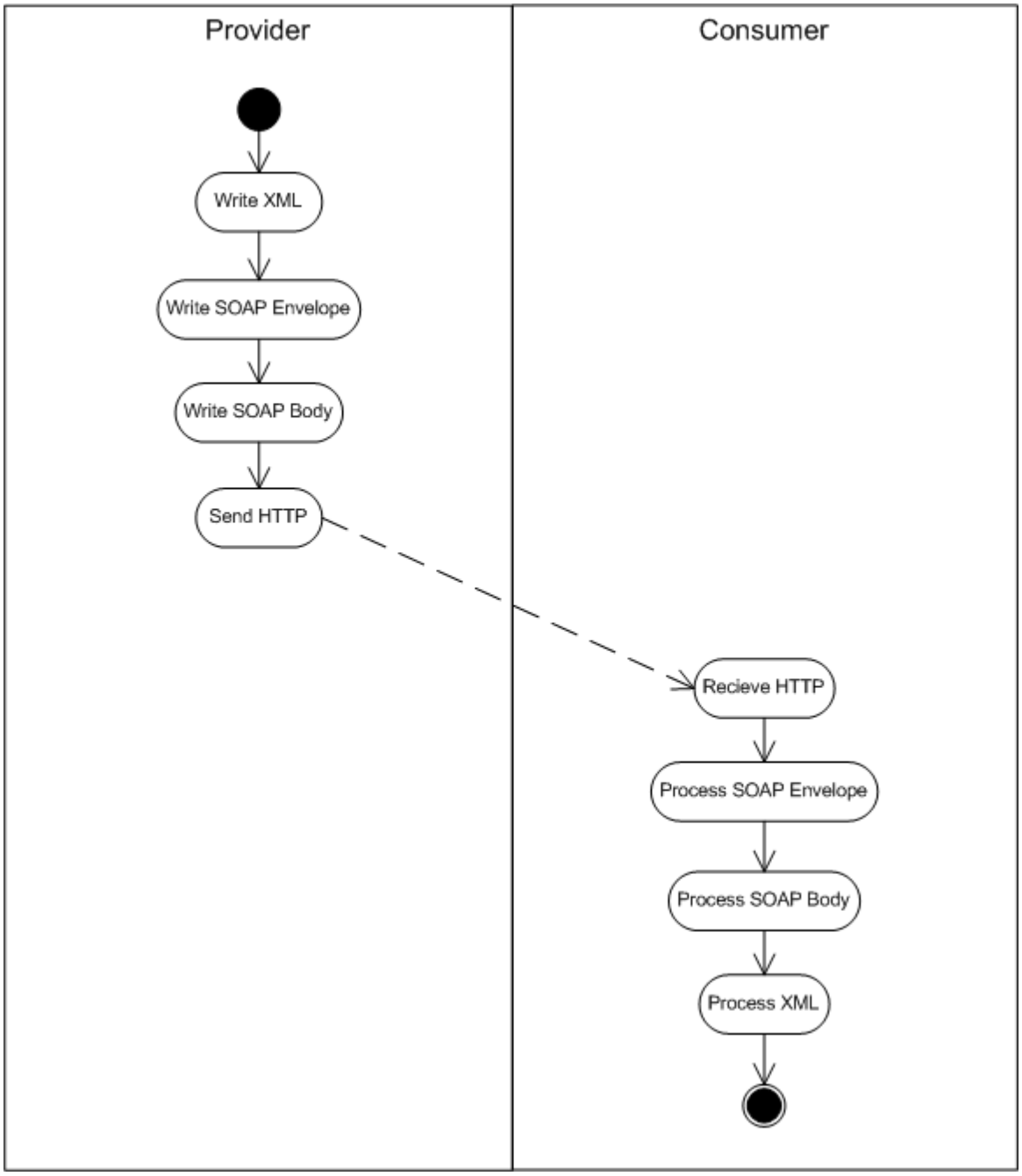

図 3-9 基本コールバックプロバイダ Acknowledgement

プロバイダが受取り(レスポンス)メッセージを生成する:

- データ層
  - XML を書き出す。データモデルに従ってペイロードが作成される。
- SOAP メッセージ層
  - SOAP エンベロープを書き出す
  - SOAP ボディを書き出す

- トランスポート層
    - HTTP を送信する

コンシューマが受取り(レスポンス)メッセージを受信する:

- トランスポート層
    - HTTP を受信する
- SOAP メッセージ層
    - SOAP エンベロープを処理する
    - SOAP ボディを処理する
- データ層
    - XML を処理する。データモデルにしたがってデータペイロードが処理され、アプリケーションにディスパッチされる

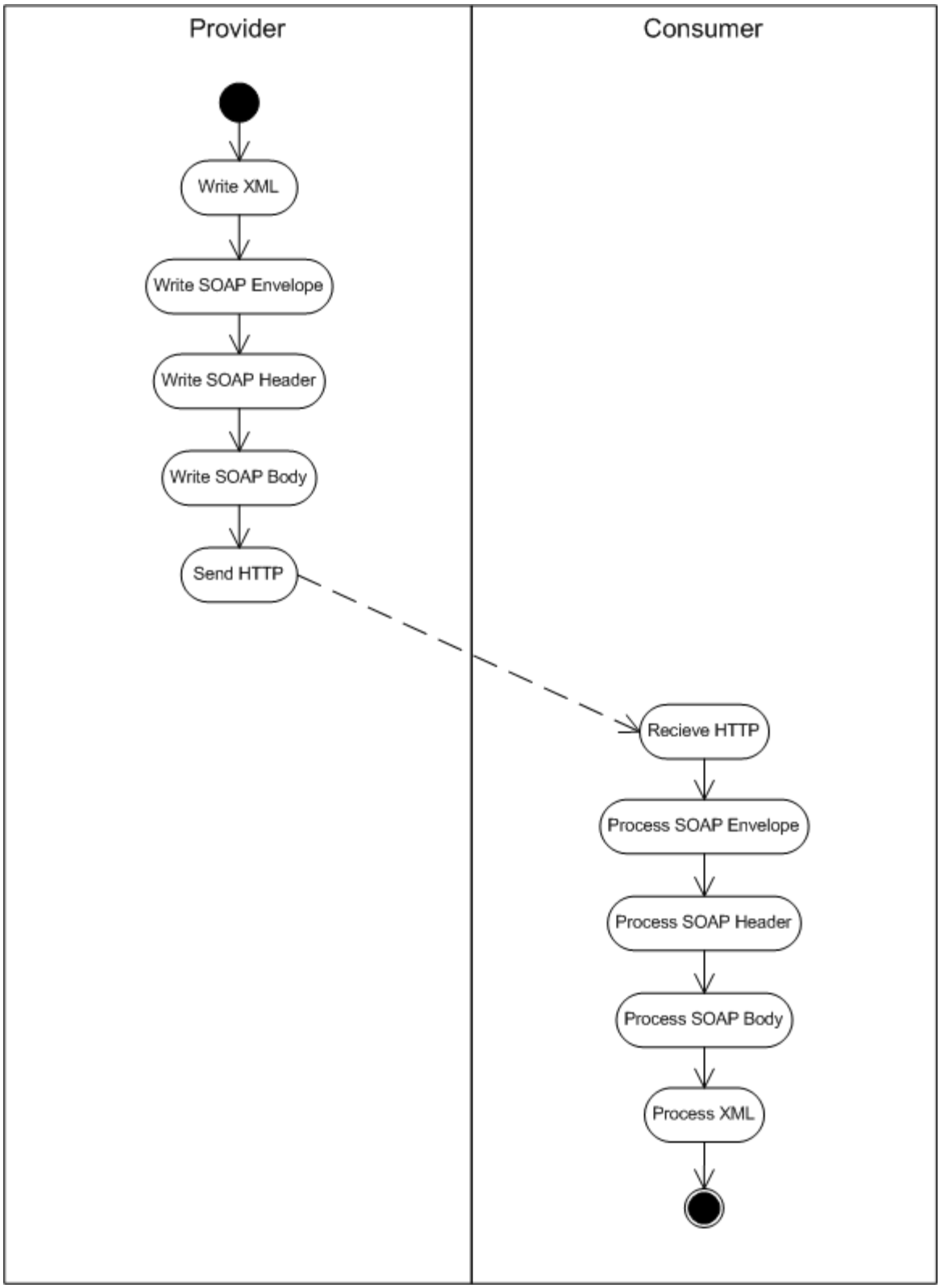

図 3-10 基本コールバックプロバイダリクエスト

プロバイダが SOAP リクエストを初期化する。

- データ層
    - XML を書き出す。データモデルに従ってペイロードが作成される。
- SOAP メッセージ層
    - SOAP エンベロープを書き出す
    - [SOAP ヘッダを書き出す (相関性情報が SOAP ヘッダで伝えられる場合)]
    - SOAP ボディを書き出す
- トランスポート層
    - HTTP を送信する

コンシューマが SOAP リクエストを受信する。

- トランスポート層
    - HTTP を受信する
- SOAP メッセージ層
    - SOAP エンベロープを処理する
    - [SOAP ヘッダを処理する (相関性情報が SOAP ヘッダで伝えられる場合)]
    - SOAP ボディを処理する
- データ層
    - XML を処理する。データモデルにしたがってデータペイロードが処理され、アプリケーションにディスパッチされる

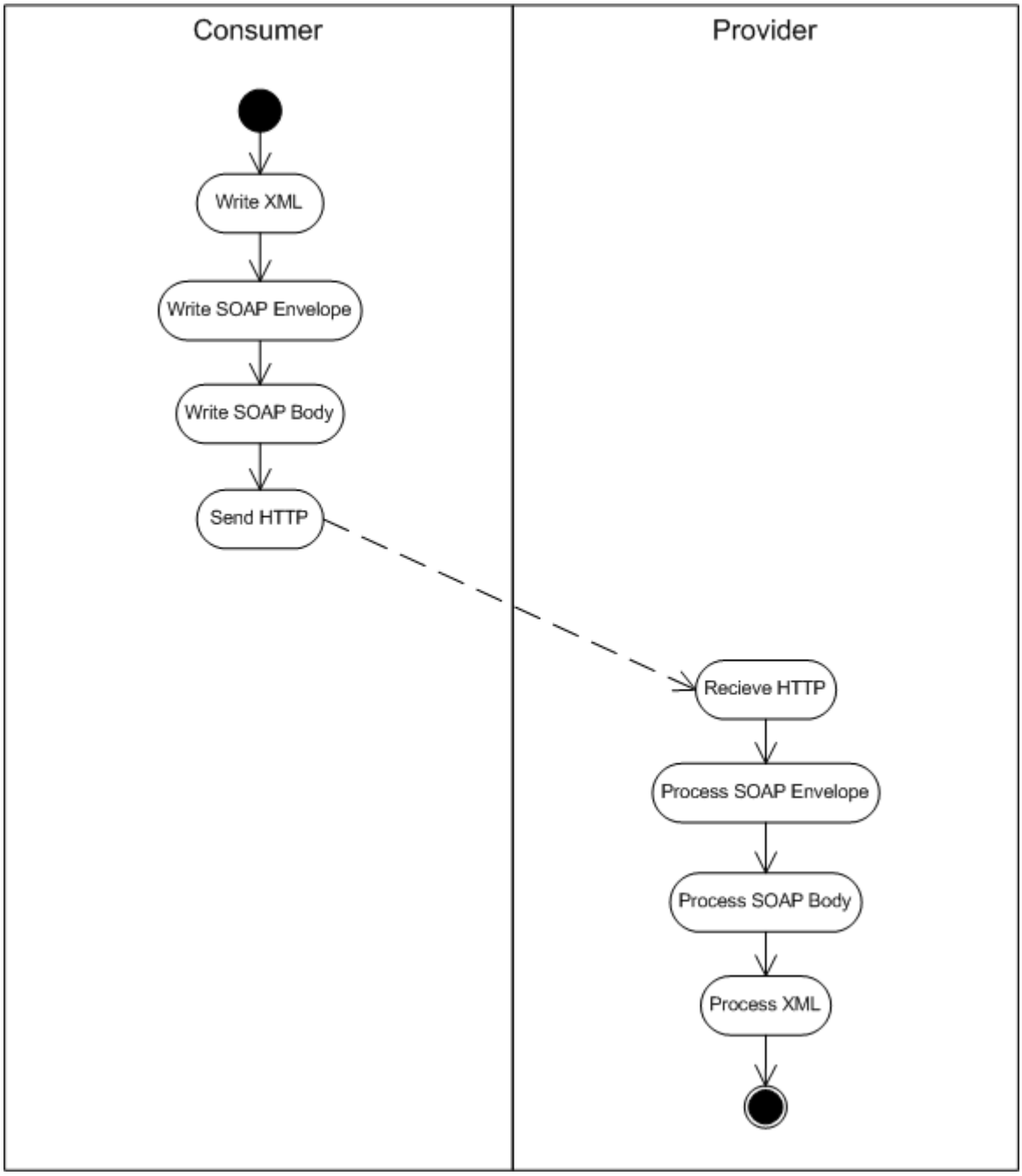

図 3-11 基本コールバックコンシューマ Acknowledgement

コンシューマが受取り(レスポンス)メッセージを生成する:

- データ層
  - XML を書き出す。データモデルに従ってペイロードが作成される。
- SOAP メッセージ層
  - SOAP エンベロープを書き出す
  - SOAP ボディを書き出す

- トランスポート層
    - HTTP を送信する

プロバイダが SOAP 受取り(レスポンス)メッセージを受信する:

- トランスポート層
    - HTTP を受信する
- SOAP メッセージ層
    - SOAP エンベロープを処理する
    - SOAP ボディを処理する
- データ層
    - XML を処理する。データモデルにしたがってデータペイロードが処理され、アプリケーションにディスパッチされる

3.3.4 Flow Constraints(フロー制約)

下記のアクティビティはこの利用シナリオで参照された制約を持っている。

- 5.1 節で定義されているように、XML を書き込む
- 5.3 節で定義されているように、SOAP エンベロープを書き込む
- [5.7 節で定義されているように、SOAP ヘッダを書き込む (相関性情報が SOAP ヘッダで伝えられる場合)]
- 5.5 節で定義されているように、SOAP ボディを書き込む
- 5.9 節で定義されているように、HTTP を送信する
- 5.10 節で定義されているように HTTP を受信する
- 5.4 節で定義されているように、SOAP エンベロープを処理する
- [5.8 節で定義されているように、SOAP ヘッダを処理する (相関性情報が SOAP ヘッダで伝えられる場合)]
- 5.6 節で定義されているように、SOAP ボディを処理する
- 5.2 節で定義されているように、XML を処理する

3.3.4.1 Errors and SOAP Faults(エラーと SOAP Fault)

同期リクエスト/レスポンスシナリオ 3.2.3.1 で記述された Fault 生成の制約と振舞いはこのシナリオにも適用する。

3.3.5 Description Constraints(記述の制約)

WSDL は基本コールバックシナリオの定義内に少なくとも以下の内容を持っているべきである。全ての節が要求されるとは限らない。しかし WSDL 中にある場合、それぞれは下に示されるようなガイドラインに従うべきである。下に定義された WSDL は単一の文書内に含まれており、基本コールバックの初期・最終リクエスト/レスポンスの両方を記述する。Basic Profile によって WSDL に課された制約もリストされている。

WSDL の一般的な制約は 5.11 節に記述されている。Basic Profile によって課されたその他の制約は以下に一覧される。

3.3.5.1 types

Application Data(アプリケーションデータ)

この節はデータモデルの詳細に依存し、アプリケーションデータ型に加えて相関性情報を含んでいる。

3.3.5.2 messages (メッセージ)

データモデル(doc/literal または rpc/literal)に依存している。以下に示すメッセージと部分(parts)が典型的にはこの利用シナリオのために定義される。

- 初期リクエストメッセージ
- 初期レスポンスメッセージ
- 最終リクエストメッセージ
- 最終レスポンスメッセージ

3.3.5.2.1Document and RPC style(ドキュメントスタイルと RPC スタイル)

以下は Document と RPC スタイルの間の違いに関するいくつかの一般的な問題である。簡単のために、このシナリオに必用なメッセージは全て Document スタイルとして定義されているとする。ただし、これは必須ではない。

Document messages(ドキュメントメッセージ)

ドキュメントメッセージ部はスキーマ要素定義から構成される(R2204 を見よ)

```
<wsdl:message …>
       <wsdl:part name=”InitialRequest” element=”..”>
</wsdl:message>
```

RPC messages(RPCメッセージ)

RPC メッセージ部はスキーマ型宣言から構成される(R2203 を見よ)

しかしながら、SOAP ヘッダまたは Fault で使われるパートは要素として定義されなければならない。

```
<wsdl:message …>
       <wsdl:part name=”InitialRequest“ type=”..” />
</wsdl:message>
```

メッセージの制約は 5.13 節に挙げられている。

3.3.5.3 portTypes

初期および最終シーケンスの両方において、リクエスト/レスポンスプリミティブが使われなければならない。

3.3.5.3.1Provider

```
<wsdl:portType name="ProviderPortType">
       <wsdl:operation name="…">
              <wsdl:input…
              </wsdl:input>
              <wsdl:output…
              </wsdl:output>
       </wsdl:operation>
</wsdl:portType>
```

3.3.5.3.2Consumer

```
<wsdl:portType name="ConsumerPortType">
        <wsdl:operation name="…">
                <wsdl:input…
                </wsdl:input>
                <wsdl:output…
                </wsdl:input>
        </wsdl:operation>
</wsdl:portType>
```

portType についての制約は 5.14 節に挙げられている。

3.3.5.4 binding

wsdl:binding 節は HTTP トランスポートを伴った SOAP バインディング拡張を使わなければならない。wsdl:portType 中で定義された同じオペレーション型がバインディング節で使われなければならない。初期シーケンスと最終シーケンスの二つのバインディングが定義される。

3.3.5.4.1Provider(プロバイダ)

```
<wsdl:binding name="ProviderSoapBinding" type="tns:ProviderPortType">
<soap:binding style="document|rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="…">
        <soap:operation/>
        <wsdl:input…
        </wsdl:input>
        <wsdl:output…
        </wsdl:output>
</wsdl:operation>
</wsdl:binding>
```

3.3.5.4.2Consumer(コンシューマ)

```
<wsdl:binding name="ConsumerSoapBinding" type="tns:ConsumerPortType">
<soap:binding style="document|rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="submitFinalReq">
        <soap:operation/>
        <wsdl:input…
        </wsdl:input>
        <wsdl:output…
        </wsdl:output>
</wsdl:operation>
</wsdl:binding>
```

バインディングについての制約は 5.15 節に挙げられている。

3.3.5.5 port

初期リクエスト/レスポンスのためのただ一つのポートが定義される。最終シーケンスのポートバインディングは明記されない。 soap:address 要素はエンドポイントの URL と共に明記されなければならない。

```
<wsdl:port>
       <soap:address location="uri" />
</wsdl:port>
```

ポート定義についての制約は 5.16 節に挙げられている。

3.3.6 UDDI

この利用シナリオで必用とされる二つの Web サービス実装があるが、初期 Web サービスだけが UDDI で広告される。最終 Web サービスのコールバックエンドポイントは初期リクエストの開始者によって知られ、アクセスされなければならないので、ディスカバリ可能でなくてよい。初期リクエストでのコールバックエンドポイントとの通信はこれを達成する。

このシナリオでパターン化された Web サービスの広告はベストプラクティス文書 "Using WSDL in a UDDI Registry, Version 1.07" を踏襲する。 uddi:tModel は初期オペレーションのための wsdl:binding を含むファイルを参照する Web サービスの型を表現している。 Web サービスタイプのこの部分に一致する wsdl:binding は、WSDL ファイルの URL に xpointer ベースの断片識別子を付加したものを使って参照される。初期シーケンスのためのuddi:bindingTemplate はサービスエンドポイントを保存し、Web サービスタイプ用の uddi:tModel(s) を参照する。

このシナリオの最終シーケンスは初期シーケンスに関係した二つの当事者の間で起るので、最終シーケンスの広告やディスカバリは求められたり必用とされたりしない。

この方法による基本コールバック Web サービスの広告はUDDI Inquiry API セットによってサポートされる紹介パターン(http://www.uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.pdf を見よ)を使ったディスカバリを可能にする。これらはブラウズ(browse)パターン、ドリルダウン(drill-down)パターン、インボケーション(invocation)パターンを含む。

一般的な UDDI の制約は 5.17 節に挙げられている。

3.3.7 Security(セキュリティ)

この節では付録1(Basic Profile 準拠の対抗策に関する追加の情報があるかもしれない)で定義されているこの利用シナリオに最も関連する脅威を確認する。
セキュリティを実装するために HTTPS が使われる場合、追加の制約を適用するかもしれない。これらは Profile 要件: R5000, R5001, R5010 である。付録 1 はセキュリティのための追加のガイドラインを含む。

この段階で、Replay は潜在的な候補として調査されているが、特定の脅威を書かないことは、この利用シナリオに非常に適切なこととして認識された。

4 Appendix 1 – Security(付録 1 – セキュリティ)

この節では、Web サービス共通の脅威を詳細化し、Basic Profile に従った可能な対抗策を示す。ここで詳細化される対抗策は、あなたの Web サービスアプリケーションのリスク評価を終えてから、最適に適用されるものである。

ここで述べられている情報は、Web サービス開発者に立ちはだかるセキュリティ問題を網羅的、または百科事典的に取り扱うようなことを意図していない。むしろ、伝統的なセキュリティ問題と Web サービスアーキテクチャ上での発生する問題の共通部分を単に探索するといったセキュリティ問題の中間評価を提供する程度のものである。

4.1 Authentication(認証)

認証は、身元の証明を検証するための仕組み、または手続きである。認証の仕組みを使うことで、Web サービスはリクエストまたはレスポンスのメッセージの送り手を断定し、メッセージを実行することができる。パスワード方式、Secure Socket Layer(SSL)、ケルベロス、公開鍵方式など、多くの認証方式や認証プロトコルが開発されている。それぞれの機構は、利点と制限を持っている。相互運用性の観点から、それぞれの機構はさまざまな挑戦を導入する。Web サービスは、通常、相互運用可能な認証方式を提供するためにソフトウェアプラットフォームを信頼することが出来る。また、協力するビジネス実体のコミュニティ内部でのシングルサインオンを提供するために、ドメインを超えて認証情報を共有したいかもしれない。

認証要求は通常サービスリクエスタとサービスプロバイダの間で非対称である。それゆえ、Web サービスに対する認証は、以下のように再分割することができる。

4.1.1 Request Authentication(リクエスト認証)

脅威

ユーザ認証しない Web サービスに対する脅威として、認証していない実体によるデータまたはリソースへのアクセスと”man-in-the-middle”攻撃が考えられる。Man-in-the-middle 攻撃では、認証されていない実体がリクエスタとレスポンダ間のメッセージを横取りし、データを傍受し操作することが可能である。非常に複雑な Web サービス環境において、Web サービスプロバイダはトランザクションに含まれるすべての関係者を認証することができないかもしれない。そして、それゆえ他の Web サービスへ信頼を委譲することが要求されるかもしれない。

対抗策

Web サービスは、リクエストの送り手を認証するべきである。Web サービスがリクエストを認証するべき特殊な状況では、裏に潜んだ状態の変更やサービス利用料の請求、サービスから得られた情報に権限が与えられるなどのケースが考えられる。

リクエスタの認証は適切な対抗策である。SSL/TLS のクライアント認証の一部に含まれる電子署名を認めることで、クライアント認証を実行することができる。

4.1.2 Response Authentication(レスポンス認証)

脅威

サービスリクエスタへの攻撃は、期待されるサービスのレスポンスに似せたメッセージを返す偽のサービスを間に挟むことである。たとえば、”man-in-the-middle”攻撃は、本物のレスポンスを偽のレスポンスで代用して、リクエスト/レスポンスの不整合を引き起こすだろう。

対抗策

サービスの認証が適切な対抗策である。SSL/TLS 接続はサーバ認証を提供し、サーバ間のトランザクションを欺くような Web サービスプロバイダからの攻撃を十分防ぐことができる。

4.2 Authorization(認可)

認可とは、あるサービスプロバイダやある信頼された実体が、別のある実体に許可する能力を決定するプロセスである。認証がどの実体がある Web サービスにアクセスできるかどうかを決める間、認可は Web サービスのどの特徴が認証された実体からアクセスできるかどうかを決定する。認証された実体でさえ、Web サービスによって提供された機能の一部しかアクセスできないような制限を課されなければならない場合もある。

4.2.1 Request Authorization(リクエスト認可)

脅威

コンピュータリソースや保護されたデータへの認可なしのアクセス。

対抗策

認可の機構を適用する。サービスプロバイダによって特別なリクエスタに割り当てられた認可に基づいて、Web サービスリクエストは遂行される。ある Web サービスでは、ポリシーを通じて認可要求を理解しあう必要があるかもしれない。

シンプルな Web サービスでは、ひとつの認可レベルしか持たないかもしれない。たとえば、私は認識できるトークンを用いて認証されたすべてのユーザのためにプロセス X を実行する意思がある。しかしながら、コンシューマのある範囲に対してサービスを提供するような Web サービスのために、より複雑な機構が必要となるかもしれない。

4.3 Confidentiality （機密性）

脅威

認可なしのアクセスを通じて機密情報を危険にさらす。　メッセージング環境（反対はセッション環境）において、Web サービスのメッセージ保護に関する性質を評価することは重要である。なぜなら、Web サービスは送られているデータの最終的な目的地や完全な経路を知らないからである。中継を通過し、データが保護されていない場合、メッセージに含まれる機密性のある内容を中継者が読むかもしれないし、特別なメッセージ（あるタイプのメッセージやある頻度で含まれるメッセージなど）が送られるという単なる事実によって機密情報を推定することができるかもしれない。

対抗策

暗号化は機密性への不正アクセスに対する基本的な防御である。暗号化の適用方法は大きく変わりうる。SSL/TLS はセッション持続している間、メッセージを暗号化する。しかし、それぞれのエンドポイント上では、完全に複合化されてしまうだろう。この状況の例外としては、SSL プロキシトンネルがある。クライアントプロキシは、セキュアなサーバに対して接続をおこなう際に、セキュアなトランザクションを介在せずに双方向でデータをコピーする。

Basic Profile のスコープ外ではあるが、適合性を残しながら２点間の機密性に関する問題に焦点をあてる方法がある。たとえば、XML 暗号は選択的にエレメントや全メッセージを暗号化するといった使い方ができる。多くの設定が存在するが、SOAP ヘッダに “ 平文で ” 他の情報を残したまま、メッセージを暗号させることができる。

4.4 Data Integrity（データ完全性）

脅威

データの完全性を失うことで、認可せずにリクエストまたはレスポンスの修正が行われる。Web サービスに対する脅威としては、データの悪意ある改ざんや偶発的な変更がある。

対抗策

SSL/TLS を用いている場合、メッセージ交換の間はデータ完全性が保障されている。

まだスコープ外ではあるが、Basic Profile を満たす他のテクニックとしては、XML 電子署名を用いてデータ完全性を証明するといった電子署名やメッセージダイジェストを利用するという手もある。これらは、完全な XML メッセージや XML 電子署名の仕様に従った XML 文書の一部に適用することができる。

4.5 Replay（リプレイ）

脅威

Web サービスに対する基本的な攻撃として、かつて妥当であったメッセージを使いまわすという試みがある。セキュリティトークンのような Web サービスメッセージのある要素も妥当なリクエストやレスポンスとしての効果を与えるために異なるメッセージの一部として使いまわすことが出来る。

対抗策

すべてのメッセージ上に universally unique identifiers(uuid)を割り振り、メッセージのタイムスタンプやキャッシングすることによって、リプレイ攻撃を検出することができる。

4.6 Logging and Auditing（ロギングと監査）

上記のセキュリティ問題すべてに対する最高の対抗策の一つとして、強固な監査/ロギングの機構がある。認証機構と組み合わせることにより、監査とロギングの機構は、トラストポリシーに対する実行時の違反行為をビジネス契約と契約法の信用基盤がオフラインで矯正できるような一連の証拠を提示することができる。

4.7 Other Risks (その他のリスク)

すべてのネットワークアプリケーションと同様に、 Web サービスは以下に示す一般的なネットワークセキュリティ脆弱性にさらされている。

- 認可されていない利用者がネットワーク資源への直接のアクセスを獲得してしまう
- 妥当な XML メッセージの中に送信されているウィルスまたはトロイの木馬プログラム
- Web サービスプロバイダによる内部リソースのミスコンフィグレーションや不整合
- 既知の脆弱性の利用
- サービス停止(DoS)攻撃

5 Appendix 2 – Constraints(付録 2-制約)

この節では、第 2 節で区別したそれぞれのフローアクティビティとそれぞれのシナリオに対して、Basic Profile でリストアップされている制約とのマッピングを提供する。それぞれのアクティビティを実行すると、リストアップされた制約は、制約の詳細を満たしていることを確認するために Basic Profile を参照するべきである。

5.1 Write XML(XML の書き出し)

- SOAP メッセージの XML 表現: R4001, R1008, R1009, R1010, R1012, R1013

5.2 Process XML(XML の処理)

- SOAP メッセージの XML 表現: R4001, R1008, R1009, R1010, R1012, R1013, R1015, R1017

5.3 Write SOAP Envelope(SOAP エンベロープの書き出し)

- エンベロープ構造: R1011, R2714

5.4 Process SOAP Envelope(SOAP エンベロープの処理)

- エンベロープ用件: R1011, R1015, R1028, R2714

5.5 Write SOAP Body (SOAP ボディの書き出し)

- SOAP メッセージの XML 表現: R1005, R1006, R1007, R1011, R1014, R2735, R2737
- SOAP 処理モデル: R1025, R1029, R1030
- RPC メッセージ: R2729

5.6 Process SOAP Body (SOAP ボディの処理)

- SOAP メッセージの XML 表現: R1005, R1006, R1007, R1014, R1017, R1028, R1029, R1030

5.7 Write SOAP Header (SOAP ヘッダの書き出し)

- SOAP メッセージの XML 表現: R4001, R1005, R1008, R1009, R1010, R1012, R1013,
- SOAP 処理モデル: R1027
- HTTP での SOAP の利用: R1109
- ヘッダ部: R2738, R2739, R2751, R2752, R2753

5.8 Process SOAP Header (SOAP ヘッダの処理)

- SOAP メッセージの XML 表現: R1012, R1005, R1008, R1009, R1010, R1012, R1013, R1015, R1017,
- SOAP 処理モデル: R1025, R1026, R1027, R1028, R1029, R1030,

5.9 Send HTTP (HTTP の送信)

以下の制約は HTTP と HTTPS の両方に適用する。

- 一般: R1108, R1140, R1141, R1132
- 状態コード: R1106, R1107, R1111, R1112, R1113, R1114, R1115, R1116, R1124, R1125, R1126, R1130
- SOAPAction ヘッダ: R1109, R2713, R2744, R2745
- クッキー: R1120, R1121, R1122, R1123

5.10 Receive HTTP(HTTP の受信)

以下の制約は HTTP と HTTPS の両方に適用する。

- 一般: R1110, R1140, R2746
- 状態コード: R1107, R1111, R1112, R1113, R1114, R1115, R1116, R1124, R1125, R1126, R1130, R1131
- SOAPAction ヘッダ: R1119
- クッキー: R1120, R1121, R1122, R1123

5.11 General WSDL Constraints (一般的な WSDL 制約)

- インスタンスの記述: R0001
- ドキュメントの WSDL へのインポート: R2001, R2002, R2003, R2004, R2005, R2007, R2008, R2009, R2010, R2011

- WSDL の全体構造上の制約: R2020, R2021, R2022, R2023, R2024, R2025, R2026, R2027, R2028, R4002, R4003, R4004
- WSDL 拡張: R2747, R2748

5.12 Constraints on WSDL types(WSDL types の制約)

- QNames 利用上の制約: R2101, R2102
- 配列型の宣言に関する制約: R2110, R2111, R2112, R2113
- XML Schema の利用: R2105, R2114, R2800, R2801

5.13 Constraints on WSDL messages(WSDL message の制約)

- bindings と parts に関する制約: R2201, R2202, R2203, R2204, R2206, R2207, R2208, R2210
- portType の制約: R2209

5.14 Constraints on WSDL portTypes(WSDL portType の制約)

- message の伝送路上での表現: R2301, R2302, R2305, R2306, R2710, R2712
- operations 上の制約: R2303, R2304

5.15 Constraints on WSDL Bindings(WSDL binding の制約)

- 構造: R2029
- 許可するバインディング: R2401, R2700
- トランスポートの制約: R2701, R2702
- soap:style の制約: R2705
- soap:use の制約: R2706, R2707
- portTypes との関係: R2709, R2718
- SOAPAction の利用: R2713
- soap:namespace 属性の利用: R2716, R2717, R2726
- フォールトとヘッダの制約: R2719, R2720, R2721, R2722, R2723, R2740, R2741, R2749

5.16 Constraints on WSDL Port(WSDL port の制約)

- 許可するバインディング: R2711

5.17 General UDDI constraints(一般的な UDDI の制約)

- インスタンスの記述: R0001
- バインディングテンプレートの制約: R3100
- tModels の制約: R3002, R3003, R3005, R3010, R3011

6 References(参考文献)

[1] WS-I Basic Profile version 1.0 from www.ws-i.org.