2.　各種プラットフォーム向け SOAP 実装

2.1.　はじめに

SOAP は XML をパケットデータとして用いたサービスやオブジェクト、サーバにアクセスするためのワイヤプロトコルであり、そこで用いられる XML は、拡張可能なマークアップ言語であり、これまでにインターネット上で使用されていた HTML に、SGML の柔軟性とパワーを組み合わせるように、かつ、SGML より単純化されるように設計された。XML はいわゆるテキストデータのため、マシンアーキテクチャ等に依存しない形でデータを構築できる。そのため、これを用いた SOAP では、マシンアーキテクチャや OS に依存しないという特徴を持つことができる。また、XML は SGML より単純化されるように設計されているため、テキスト処理が可能なプログラミング言語ならば、どのようなものでも、XML 処理プログラムが SGML 処理プログラムよりも簡単に構築することが可能であり、それゆえ、プログラミング言語にも依存しないことになる。

以上のような特徴をもつため、SOAP を用いることにより、異なるプラットフォーム間での計算機資源の共有を実現できる。これにより、プログラムから利用可能なアプリケーション・コンポーネントである Web サービスの枠組が構築可能となった。そのため、現在ではさまざまな、SOAP の実装が行われているが、その多くが、Java 等の特定のプログラミング言語による実装のため、プログラミング言語に依存しないという特徴がいかされていない。また、プログラミング言語に依存しない実装も存在はするが、それは、特定 OS 上でしか、実行できないため、プラットフォーム非依存とはなっていない。これらの問題を解決するために、本サブテーマでは、まず Linux を含む UNIX 系 OS と Windows 系 OS で動作する C 言語による実装を行うことにした。UNIX 系 OS と Windows 系 OS を選んだ理由としては、現在稼働しているコンピュータの多く、特に、現在 SOAP を利用可能なものは、ほとんどが上記のカテゴリに含まれるためである。また、C 言語による実装を行う理由としては、ひとつには、C 言語は他のプログラミング言語との親和性が高いこと、また、現在では、C 言語はほとんどのプラットフォームで利用可能なこと、さらに、前述したプラットフォーム上での現在までに作成されたプログラムの多くは、C 言語での実装がなされており、そのため、過去の資産を利用可能となり、かつ、C 言語のプログラマは、Java 等の他のプログラミング言語をおぼえる必要無く利用可能となること、以上三つの理由による。

本サブテーマにおける各プログラミング言語から利用可能な SOAP 実装を OpenSOAP API と呼ぶ。C 言語における OpenSOAP API は図 2.1.1 に示すように三

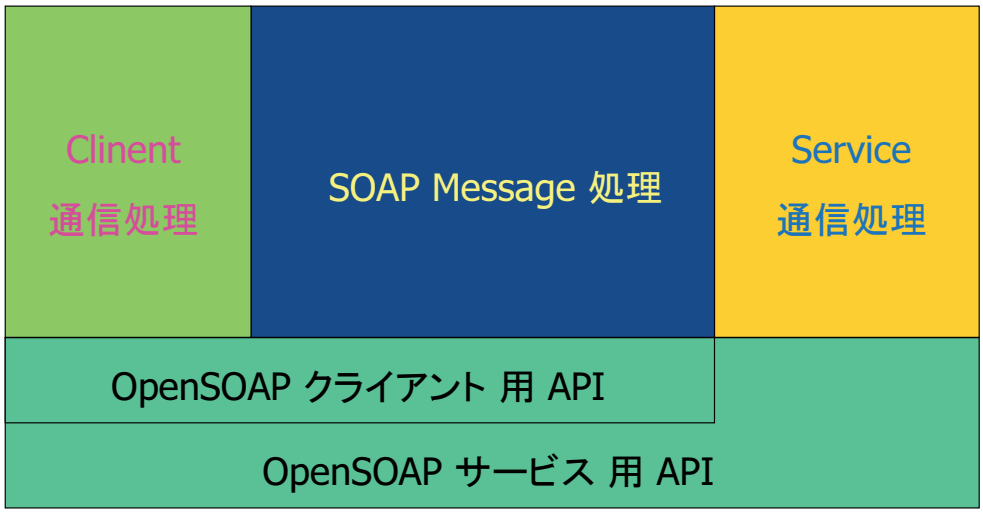

図 2.1.1 OpenSOAP API 構成

つの部分から構成され Client 通信処理と SOAP Message 処理をあわせて、OpenSOAP クライアント用 API、それに、Service 通信処理を追加したものを OpenSOAP サービス用 API と呼ぶ。

前述したように、本サブテーマでは、基本的に C 言語による実装を行うが、Web サービスの新規開発においては、現 在の所、Java 言語による実装が主に行われており、サブテーマ 5「SOAP メッセージ管理技術の実装」における拡張機能を Java 言語においても利用するための Java 言語による実装例を示す。これにより

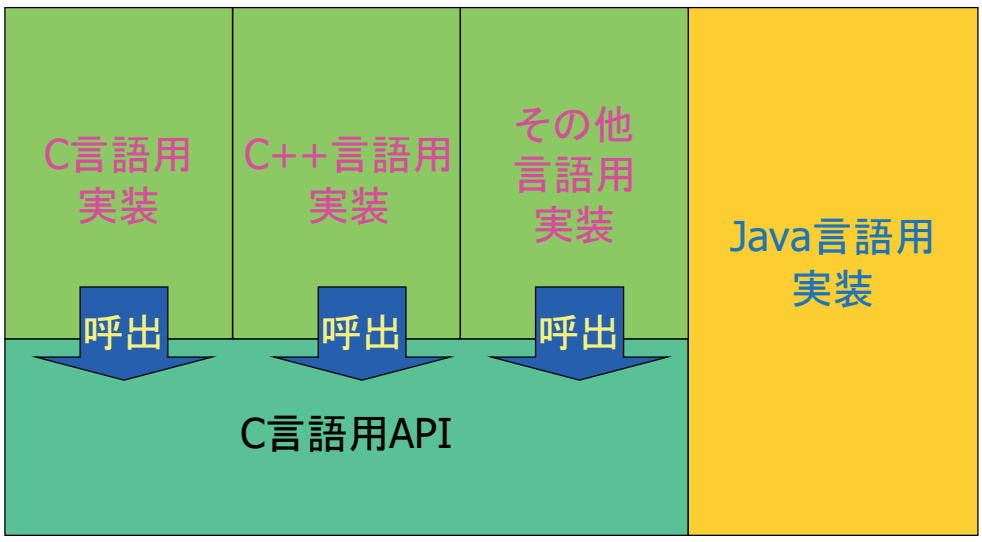

図 2.1.2 OpenSOAP API 言語毎の対応

OpenSOAP API における言語毎の対応は図 2.1.2 のようになる。

また、昨今のＩＴ関連インフラの普及により、企業間および企業と消費者間の電子商取引(B2B、B2C)の市場が加速度的に拡大している。電子商取引はインターネットを介して行われるため、世界中が市場となりうる。それゆえ、顧客が利用する言語としては様々なものが想定される。そのため、Web サービス構築に際して、種々の言語に対応可能なプログラム作成を行うことが必須と考えられる。一方で、電子商取引においては、インターネットを介する故に、様々な不正が入り込む余地がある。したがって、こうした不正を排除できる環境を提供することが、電子商取引分野への安全な適用のための必須条件となる。それゆえ、各種プラットフォーム向け SOAP 実装における機能として様々な言語に対応可能とするための国際化機能と安全な取引を行うためのセキュリティ機能が不可欠と考えられる。そのため、前述の C 言語用実装にたいして、国際化の枠組を実装し、またサブテーマ 3「総合セキュリティ機能の実装」の成果であるセキュリティモジュールとの結合を行う。

2.2.　C 言語用 API の実装

2.2.1.　C 言語用 API 概要

ここでは、本サブテーマのなかの C 言語による OpenSOAP API 実装について述べる。OpenSOAP API というのは、OpenSOAP プロジェクトにおいて、SOAP 仕様をベースとし、ビジネス Web サービスの開発をサポートする様々な機能を付加したミドルウェアのうち、既存のソフトウェアリソースの再利用性とマルチプラットフォーム化を目的とした、Application Programming Interface のことである。OpenSOAP API は、図 2.1.1 に示すように、機能別に分類すると、(1) SOAP Message 処理用 (2) SOAP Client 通信用 (3) SOAP Service 通信用の 3 種類に分類され、そのうちの SOAP Client 通信用と SOAP Message 処理用をあわせたものを OpenSOAP クライアント用 API と呼び、さらに、SOAP Service 通信用をあわせたものを OpenSOAP サービス用 API と呼ぶ。

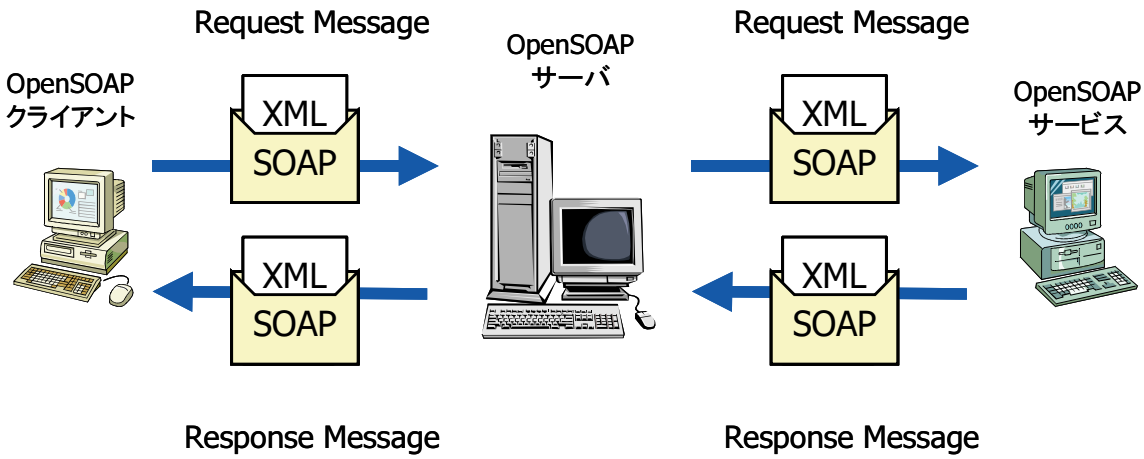

図 2.2.1 OpenSOAP API の処理

OpenSOAP API の処理の流れを図 2.2.1 に示す。処理手順は以下のようになる。

(1) OpenSOAP クライアント側は OpenSOAP の Request Message を作成する。
Request Message を作成するには、OpenSOAP Message 処理の API を利用する。

(2) Request Message を OpenSOAP サーバへ送る。
Request Message をサーバへ送るときは、OpenSOAP Client 構築用の API を利用し、パラメータ型の変換や、OpenSOAP サーバとの通信を行う。

(3) OpenSOAP サービス側が OpenSOAP サーバからの Request Message を受け取る。
Request Message を受け取るには、OpenSOAP Service 構築用の API を利用し、パラメータ型の変換や、OpenSOAP サーバとの通信を行う。

(4) OpenSOAP サービス側は受け取った Request Message に対して、解析し、サービスを実行する。
OpenSOAP Message 処理用の API を利用して、Request Message を解析し、また OpenSOAP Service 構築用 API を利用し、サービスを実行する。

(5) サービスの実行結果を Response Message で返信する。
返信するには、まず OpenSOAP Message 処理用 API を利用し、Response Message を作成する。次は、SOAP Service 構築用 API を利用し、OpenSOAP サーバを通信を行い、Response Message を送る。

(6) クライアント側は OpenSOAP サーバからの Response Message を受け取る。
Response Message を受け取るには、OpenSOAP Client 構築用 API を利用し、OpenSOAP サーバと通信を行う。

(7) クライアント側は受け取った Response Message に対して、解析し、実行結果を取得する。
Response Message に対する解析には、OpenSOAP Message 処理用 API を利用し、

実行結果の型を変換して、値を取得する。

以上の記述から、OpenSOAP API の実装は、以下の三つの部分からなることが分かる。
(1) OpenSOAP Message 処理用 API の実装
(2) OpenSOAP Client 構築用 API の実装
(3) OpenSOAP Service 構築用 API の実装

以下、上記の三つの実装について、説明する。

2.2.2. OpenSOAP Message 処理用 API の実装

OpenSOAP は、SOAP をベースとし、ビジネス Web サービスの開発をサポートする様々な機能を付加したミドルウェアである。従って、OpenSOAP で扱う Message は SOAP の仕様に従う。SOAP Message の構成は SOAP Version1.2 において、図 2.2.2 に示すようになる。図 2.2.3 に示す具体的な例で説明すると、OpenSOAP Message はルート要素の Envelope 部からなる。Envelope 部の下に、SOAP Header 部と SOAP Body 部がある。更に、SOAP Header 部と SOAP Body 部の直下にそれぞれ SOAP Block 部がある。このような構成の OpenSOAP Message を処理するために、以下の API を開発した。
(1) Envelope 部操作用 API
(2) Block 部操作用 API
(3) XML 要素部操作用 API
(4) Namespace 操作用 API
(5) 属性(Attribute)操作用 API
(6) Serializer API
(7) 文字列オブジェクト操作用 API
(8) 文字列キー連想配列操作用 API
つぎに、上記それぞれについて説明を行う。

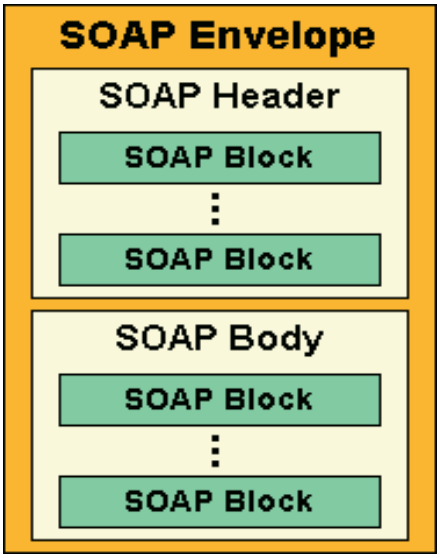

図 2.2.2 SOAP Message の構成

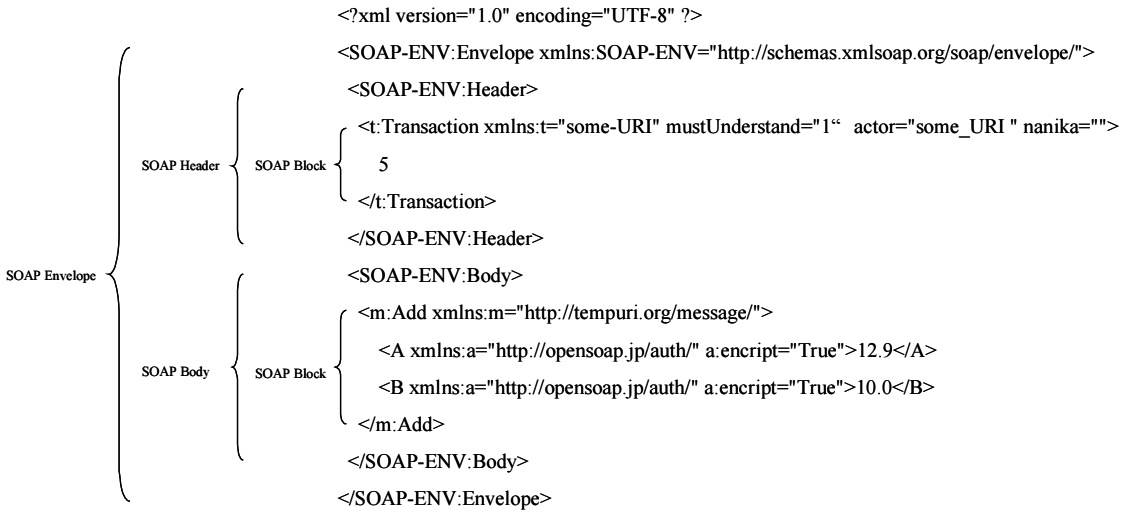

図 2.2.3 OpenSOAP Message Sample（Add Request Message）

Envelope 部操作用 API

以下の API は SOAP Envelope の操作を提供する。

・ OpenSOAP Envelope 作成と初期化

関数名　OpenSOAPEnvelopeCreate

概　要　SOAP Envelope 領域の確保と初期化を行う。

・ OpenSOAP Envelope 作成と初期化(MB)
　関数名　OpenSOAPEnvelopeCreateMB
　概　要　マルチバイト文字列で指定したバージョンの SOAP Envelope 領域の確保と初期化を行う。

・ OpenSOAP Envelope 作成と初期化(WC)
　関数名　OpenSOAPEnvelopeCreateWC
　概　要　ワイドキャラクタ文字列で指定したバージョンの SOAP Envelope 領域の確保と初期化を行う。

・ 文字エンコーディング指定による ByteArray からの SOAP Envelope 作成
　関数名　OpenSOAPEnvelopeCreateCharEncoding
　概　要　指定した文字エンコーディングとして ByteArray を読み込み、それにより、SOAP Envelope を作成する。

・ SOAP Envelope の開放
　関数名　OpenSOAPEnvelopeRelease
　概　要　SOAP Envelope 領域を開放する。

・ ヘッダブロックの追加(MB)
　関数名　OpenSOAPEnvelopeAddHeaderBlockMB
　概　要　マルチバイト文字列を用いて Envelope に Header ブロックを追加する。

・ ヘッダブロックの追加(WC)
　関数名　OpenSOAPEnvelopeAddHeaderBlockWC
　概　要　ワイドキャラクタ文字列を用いて Envelope に Header ブロックを追加する。

・ ヘッダブロックの順次取得
　関数名　OpenSOAPEnvelopeGetNextHeaderBlock
　概　要　指定したヘッダブロックの次のヘッダブロックを取得する。指定したブロックを NULL にした場合は、最初のブロックを取得する

・ ヘッダブロックの名前指定取得(MB)

関数名　OpenSOAPEnvelopeGetHeaderBlockMB

概　要　マルチバイト文字列で指定した名前のヘッダブロックを取得する。

・ ヘッダブロックの名前指定取得(WC)

関数名　OpenSOAPEnvelopeGetHeaderBlockWC

概　要　ワイドキャラクタ文字列で指定した名前のヘッダブロックを取得する。

・ ボディブロックの追加(MB)

関数名　OpenSOAPEnvelopeAddBodyBlockMB

概　要　マルチバイト文字列を用いて Envelope に Body ブロックを追加する。

・ ボディブロックの追加(WC)

関数名　OpenSOAPEnvelopeAddBodyBlockWC

概　要　ワイドキャラクタ文字列を用いて Envelope に Body ブロックを追加する。

・ ボディブロックの順次取得

関数名　OpenSOAPEnvelopeGetNextBodyBlock

概　要　指定したボディブロックの次のボディブロックを取得する。指定したブロックを NULL にした場合は、最初のブロックを取得する

・ ボディブロックの名前指定取得(MB)

関数名　OpenSOAPEnvelopeGetBodyBlockMB

概　要　マルチバイト文字列で指定した名前のボディブロックを取得する。

・ ボディブロックの名前指定取得(WC)

関数名　OpenSOAPEnvelopeGetBodyBlockWC

概　要　ワイドキャラクタ文字列で指定した名前のボディブロックを取得する。

・ SOAP Envelope の指定文字エンコーディング変換
　関数名　OpenSOAPEnvelopeGetCharEncodingString
　概　要　指定した文字エンコーディングにデータ変換する。

Block 部操作用 API

以下の API は SOAP ヘッダブロックおよびボディブロックに対する操作を提供する。

・ SOAP Block 名の取得(MB)
　関数名　OpenSOAPBlockGetName
　概　要　マルチバイト文字列で指定した型の値として Block 直下の値を取得する。

・ SOAP Block の値の取得(MB)
　関数名　OpenSOAPBlockGetValueMB
　概　要　マルチバイト文字列で指定した型の値として Block 直下の値を取得する。

・ SOAP Block の値の取得(WC)
　関数名　OpenSOAPBlockGetValueWC
　概　要　ワイドキャラクタ文字列で指定した型の値として Block 直下の値を取得する。

・ SOAP Block の値の設定(MB)
　関数名　OpenSOAPBlockSetValueMB
　概　要　マルチバイト文字列で指定した型の値として Block 直下の値を設定する。

・ SOAP Block の値の設定(WC)
　関数名　OpenSOAPBlockSetValueWC
　概　要　ワイドキャラクタ文字列で指定した型の値として Block 直下の値を設定する。

・ SOAP Block の Namespace の設定(MB)

関数名　OpenSOAPBlockSetNamespaceMB

概　要　マルチバイト文字列で指定した Namespace を SOAP Block に設定する。

・ SOAP Block の Namespace の設定(WC)

関数名　OpenSOAPBlockSetNamespaceWC

概　要　ワイドキャラクタ文字列で指定した Namespace を SOAP Block に設定する。

・ SOAP Block の Namespace の取得

関数名　OpenSOAPBlockGetNamespace

概　要　SOAP Block の Namespace を取得する。

・ SOAP Block の Namespace 比較(MB)

関数名　OpenSOAPBlockSetNamespaceMB

概　要　マルチバイト文字列で指定した Namespace URI と SOAP Block の Namespace URI を比較し、同じか判定する。

・ SOAP Block の Namespace 比較(WC)

関数名　OpenSOAPBlockSetNamespaceWC

概　要　ワイドキャラクタ文字列で指定した Namespace URI と SOAP Block の Namespace URI を比較し、同じか判定する。

・ 属性値(Attribute)の設定(MB)

関数名　OpenSOAPBlockAddAttributeMB

概　要　マルチバイト文字列で指定した属性値情報に従って、属性値を設定する。

・ 属性値(Attribute)の設定(WC)

関数名　OpenSOAPBlockAddAttributeWC

概　要　ワイドキャラクタ文字列で指定した属性値情報に従って、属性値を設定する。

・ 属性(Attribute)の取得(MB)

関数名　OpenSOAPBlockGetAttributeMB

概　要　マルチバイト文字列で指定した属性名に従って、属性を取得する。

・ 属性(Attribute)の取得(WC)

関数名　OpenSOAPBlockGetAttributeWC

概　要　ワイドキャラクタ文字列で指定した属性名に従って、属性を取得する。

・ 子要素(Element)の値の取得(MB)

関数名　OpenSOAPBlockGetChildValueMB

概　要　マルチバイト文字列で指定した Block の子要素名と型名によりその値を取得する。

・ 子要素(Element)の値の取得(WC)

関数名　OpenSOAPBlockGetChildValueWC

概　要　ワイドキャラクタ文字列で指定した Block の子要素名と型名によりその値を取得する。

・ 子要素とその値の設定(MB)

関数名　OpenSOAPBlockSetChildValueMB

概　要　マルチバイト文字列で指定した Block の子要素名と型名により子要素とその値を設定する。

・ 子要素とその値の設定(WC)

関数名　OpenSOAPBlockSetChildValueWC

概　要　ワイドキャラクタ文字列で指定した Block の子要素名と型名により子要素とその値を設定する。

・ 子要素の順次取得

関数名　OpenSOAPBlockGetNextChild

概　要　指定した子要素の次を取得する。指定した子要素が NULL の場合は、最初の子要素を取得する。

・ 子要素の追加(MB)

関数名　OpenSOAPBlockAddChildMB

概　要　マルチバイト文字列で指定した名前で Block の子要素名を追加する。

・ 子要素の追加(WC)

関数名　OpenSOAPBlockAddChildWC

概　要　ワイドキャラクタ文字列で指定した名前で Block の子要素名を追加する。

・ 名前に合致した子要素の取得(MB)

関数名　OpenSOAPBlockGetChildMB

概　要　マルチバイト文字列で指定した名前で Block の子要素名を取得する。

・ 名前に合致した子要素の取得(WC)

関数名　OpenSOAPBlockGetChildWC

概　要　ワイドキャラクタ文字列で指定した名前で Block の子要素名を取得する。

・ Mustunderstand 属性の取得

関数名　OpenSOAPBlockGetMustunderstandAttr

概　要　Mustunderstand 属性を取得する。

・ Mustunderstand 属性の設定

関数名　OpenSOAPBlockSetMustunderstandAttr

概　要　Mustunderstand 属性を設定する。

・ Mustunderstand 属性の削除

関数名　OpenSOAPBlockClearMustunderstandAttr

概　要　Mustunderstand 属性を削除する。

・ Actor 属性の取得

関数名　OpenSOAPBlockGetActorAttr

概　要　Actor 属性を取得する。

・ Actor 属性の設定(MB)
　関数名　OpenSOAPBlockSetActorAttrMB
　概　要　マルチバイト文字列で指定された値で Actor 属性を設定する。

・ Actor 属性の設定(WC)
　関数名　OpenSOAPBlockSetActorAttrWC
　概　要　ワイドキャラクタ文字列で指定された値で Actor 属性を設定する。

・ Actor 属性の削除
　関数名　OpenSOAPBlockClearActorAttr
　概　要　Actor 属性を削除する。

・ Actor 属性を next に設定
　関数名　OpenSOAPBlockSetActorAttrNext
　概　要　Actor 属性を next を意味する値に設定する。

・ Actor 属性が next か判定
　関数名　OpenSOAPBlockIsActorAttrNext
　概　要　Actor 属性が next か判定する。

XML 要素部操作用 API

以下の API は SOAP ブロックの子要素となる XML 要素に対する操作を提供する。

・ XML 要素の作成
　関数名　OpenSOAPXMLElmCreate
　概　要　XML 要素を作成する。

・ XML 要素の Namespace 設定(MB)
　関数名　OpenSOAPXMLElmSetNamespaceMB
　概　要　マルチバイト文字列で指定した Namespace を XML 要素に設定する。

・ XML 要素の Namespace 設定(WC)

関数名　OpenSOAPXMLElmSetNamespaceWC

概　要　ワイドキャラクタ文字列で指定した Namespace を XML 要素に設定する。

・ XML 要素から Namespace 定義の検索(MB)

関数名　OpenSOAPXMLElmSearchNamespaceMB

概　要　マルチバイト文字列で指定した Namespace を XML 要素自身とその親要素において定義されているか検索する。

・ XML 要素から Namespace 定義の検索(WC)

関数名　OpenSOAPXMLElmSearchNamespaceWC

概　要　ワイドキャラクタ文字列で指定した Namespace を XML 要素自身とその親要素において定義されているか検索する。

・ XML 要素の Namespace 取得

関数名　OpenSOAPXMLElmGetNamespace

概　要　ワイドキャラクタ文字列で指定した Namespace を XML 要素自身とその親要素において定義されているか検索する。

・ XML 要素の Namespace 定義(MB)

関数名　OpenSOAPXMLElmDefineNamespaceMB

概　要　マルチバイト文字列で指定した Namespace を XML 要素自身に定義する。

・ XML 要素の Namespace 定義(WC)

関数名　OpenSOAPXMLElmDefineNamespaceWC

概　要　ワイドキャラクタ文字列で指定した Namespace を XML 要素自身に定義する。

・ XML 要素に対する属性値(Attribute)の設定(MB)

関数名　OpenSOAPXMLElmAddAttributeMB

概　要　マルチバイト文字列で指定した属性値情報に従って、属性値を設定する。

・ XML 要素に対する属性値(Attribute)の設定(WC)
　関数名　OpenSOAPXMLElmAddAttributeWC
　概　要　ワイドキャラクタ文字列で指定した属性値情報に従って、属性値を設定する。

・ XML 要素に対する名前指定による属性の取得(MB)
　関数名　OpenSOAPXMLElmGetAttributeMB
　概　要　マルチバイト文字列で指定した属性名に従って、属性を取得する。

・ XML 要素に対する名前指定による属性の取得(WC)
　関数名　OpenSOAPXMLElmGetAttributeWC
　概　要　ワイドキャラクタ文字列で指定した属性名に従って、属性を取得する。

・ 子要素の順次取得
　関数名　OpenSOAPXMLElmGetNextChild
　概　要　指定した子要素の次を取得する。指定した子要素が NULL の場合は、最初の子要素を取得する。

・ 子要素の追加(MB)
　関数名　OpenSOAPXMLElmAddChildMB
　概　要　マルチバイト文字列で指定した名前で子要素名を追加する。

・ 子要素の追加(WC)
　関数名　OpenSOAPXMLElmAddChildWC
　概　要　ワイドキャラクタ文字列で指定した名前で子要素名を追加する。

・ 名前による子要素の取得(MB)
　関数名　OpenSOAPXMLElmGetChildMB
　概　要　マルチバイト文字列で指定した名前で子要素名を取得する。

・ 名前による子要素の取得(WC)

関数名　OpenSOAPXMLElmGetChildWC

概　要　ワイドキャラクタ文字列で指定した名前で子要素名を取得する。

・ XML 要素の値の取得(MB)

関数名　OpenSOAPXMLElmGetValueMB

概　要　マルチバイト文字列で指定した型名で要素の値を取得する。

・ XML 要素の値の取得(WC)

関数名　OpenSOAPXMLElmGetValueWC

概　要　ワイドキャラクタ文字列で指定した型名で要素の値を取得する。

・ XML 要素の値の設定(MB)

関数名　OpenSOAPXMLElmSetValueMB

概　要　マルチバイト文字列で指定した型名で要素の値を設定する。

・ XML 要素の値の設定(WC)

関数名　OpenSOAPXMLElmSetValueWC

概　要　ワイドキャラクタ文字列で指定した型名で要素の値を設定する。

・ XML 要素文字エンコーディングデータ変換

関数名　OpenSOAPXMLElmGetCharEncodingString

概　要　XML 要素を指定した文字エンコーディングデータに変換する。

・ XML 要素名取得

関数名　OpenSOAPXMLElmGetNameString

概　要　XML 要素名を取得する。

・ 属性の順次取得

関数名　OpenSOAPXMLElmGetNextChild

概　要　指定した属性の次を取得する。指定した属性が NULL の場合は、最初の属性を取得する。

Namespace 操作用 API

以下の API は Namespace に対する操作を提供する。

・ Namespace の作成(MB)

関数名　OpenSOAPXMLNamespaceCreateMB

概　要　マルチバイト文字列で指定した Namespace 設定値で Namespace を作成する。

・ Namespace の作成(WC)

関数名　OpenSOAPXMLNamespaceCreateWC

概　要　ワイドキャラクタ文字列で指定した Namespace 設定値で Namespace を作成する。

・ Namespace の開放

関数名　OpenSOAPXMLNamespaceRelease

概　要　Namespace 領域を開放する。

・ Namespace URI の取得

関数名　OpenSOAPXMLNamespaceGetURI

概　要　Namespace URI を取得する。

・ Namespace Prefix の取得

関数名　OpenSOAPXMLNamespaceGetPrefix

概　要　Namespace Prefix を取得する。

属性(Attribute)操作用 API

以下の API は属性(Attribute)に対する操作を提供する。

・ 属性の作成(MB)

関数名　OpenSOAPXMLAttrCreateMB

概　要　マルチバイト文字列で指定した名前の属性を作成する。

・ 属性の作成(WC)

関数名　OpenSOAPXMLAttrCreateWC

概　要　ワイドキャラクタ文字列で指定した名前の属性を作成する。

・ 属性の Namespace 設定(MB)

関数名　OpenSOAPXMLAttrSetNamespaceMB

概　要　マルチバイト文字列で指定した Namespace を属性に設定する。

・ 属性の Namespace 設定(WC)

関数名　OpenSOAPXMLAttrSetNamespaceWC

概　要　ワイドキャラクタ文字列で指定した Namespace を属性に設定する。

・ 属性の Namespace 取得

関数名　OpenSOAPXMLAttrGetNamespace

概　要　属性の Namespace を取得する。

・ 属性の値の取得(MB)

関数名　OpenSOAPXMLAttrGetValueMB

概　要　マルチバイト文字列で指定した型に適合した値を取得する。

・ 属性の値の取得(WC)

関数名　OpenSOAPXMLAttrGetValueWC

概　要　ワイドキャラクタ文字列で指定した型に適合した値を取得する。

・ 属性の値の設定(MB)

関数名　OpenSOAPXMLAttrSetValueMB

概　要　マルチバイト文字列で指定した型に適合した値を設定する。

・ 属性の値の設定(WC)

関数名　OpenSOAPXMLAttrSetValueWC

概　要　ワイドキャラクタ文字列で指定した型に適合した値を設定する。

・ 属性名の取得

関数名　OpenSOAPXMLAttrGetName

概　要　属性名を取得する。

Serializer API

以下の API は Serializer/Deserializer に対する操作を提供する。

・ Serializer/Deserializer の登録(MB)

関数名　OpenSOAPSerializerRegisterMB

概　要　マルチバイト文字列で指定した型の Serializer / Deserializer を登録する。

・ Serializer/Deserializer の登録(WC)

関数名　OpenSOAPSerializerRegisterWC

概　要　ワイドキャラクタ文字列で指定した型の Serializer / Deserializer を登録する。

・ Serializer の取得(MB)

関数名　OpenSOAPGetSerializerMB

概　要　マルチバイト文字列で指定した型の Serializer を取得する。

・ Serializer の取得(WC)

関数名　OpenSOAPGetSerializerWC

概　要　ワイドキャラクタ文字列で指定した型の Serializer を取得する。

・ Deserializer の取得(MB)

関数名　OpenSOAPGetDeserializerMB

概　要　マルチバイト文字列で指定した型の Deserializer を取得する。

・ Deserializer の取得(WC)

関数名　OpenSOAPGetDeserializerWC

概　要　ワイドキャラクタ文字列で指定した型の Deserializer を取得する。

文字列オブジェクト操作用 API

以下の API は OpenSOAP 文字列に対する操作を提供する。

・ OpenSOAP 文字列の作成

関数名　OpenSOAPStringCreate

概　要　OpenSOAP 文字列を作成する。

・ 初期値付 OpenSOAP 文字列の作成(MB)

関数名　OpenSOAPStringCreateWithMB

概　要　マルチバイト文字列で指定した値を初期値として OpenSOAP 文字列を作成する。

・ 初期値付 OpenSOAP 文字列の作成(WC)

関数名　OpenSOAPStringCreateWithWC

概　要　ワイドキャラクタ文字列で指定した値を初期値として OpenSOAP 文字列を作成する。

・ OpenSOAP 文字列の開放

関数名　OpenSOAPStringRelease

概　要　OpenSOAP 文字列を開放する。

・ 文字列長さの取得(MB)

関数名　OpenSOAPStringGetLengthMB

概　要　マルチバイト文字列としての文字列長さを取得する。

・ 文字列長さの取得(WC)

関数名　OpenSOAPStringGetLengthWC

概　要　ワイドキャラクタ文字列としての文字列長さを取得する。

・ 領域を確保した文字列の取得(MB)

関数名　OpenSOAPStringGetStringMBWithAllocator

概　要　マルチバイト文字列としての文字列を領域を確保して取得する。

・ 領域を確保した文字列の取得(WC)

関数名　OpenSOAPStringGetStringWCWithAllocator

概　要　ワイドキャラクタ文字列としての文字列を領域を確保して取得する。

・ 文字列の取得(MB)

関数名　OpenSOAPStringGetStringMB

概　要　マルチバイト文字列としての文字列を取得する。

・ 文字列の取得(WC)

関数名　OpenSOAPStringGetStringWC

概　要　ワイドキャラクタ文字列としての文字列を取得する。

・ 文字エンコーディングを指定した文字列の取得

関数名　OpenSOAPStringGetCharEncodingString

概　要　文字エンコーディングを指定しての文字列を取得する。

・ 文字列の設定(MB)

関数名　OpenSOAPStringSetStringMB

概　要　マルチバイト文字列としての文字列を設定する。

・ 文字列の設定(WC)

関数名　OpenSOAPStringSetStringWC

概　要　ワイドキャラクタ文字列としての文字列を設定する。

・ 文字エンコーディングを指定した文字列の設定

関数名　OpenSOAPStringSetCharEncodingString

概　要　文字エンコーディングを指定しての文字列を設定する。

・ 書式による整形(MB)

関数名　OpenSOAPStringFormatMB

概　要　マルチバイト文字列を書式に用いて整形した文字列を設定する。

・ 書式による整形(WC)

関数名　OpenSOAPStringFormatWC

概　要　ワイドキャラクタ文字列を書式に用いて整形した文字列を設定する。

・ 文字列比較(MB)

関数名　OpenSOAPStringCompareMB

概　要　マルチバイト文字列と比較する。

・ 文字列比較(WC)

関数名　OpenSOAPStringCompareWC

概　要　ワイドキャラクタ文字列と比較する。

・ 文字列比較

関数名　OpenSOAPStringCompare

概　要　OpenSOAP 文字列同士を比較する。

・ 文字列検索(MB)

関数名　OpenSOAPStringFindStringMB

概　要　マルチバイト文字列としての文字列を検索する。

・ 文字列検索(WC)

関数名　OpenSOAPStringFindStringWC

概　要　ワイドキャラクタ文字列としての文字列を検索する。

・ 文字列検索

関数名　OpenSOAPStringFindString

概　要　文字列を検索する。

・ 条件付文字列検索

関数名　OpenSOAPStringFindIfStringIndex

概　要　条件関数に基づき文字列を検索する。

・ 文字列置換(MB)
　関数名　OpenSOAPStringReplaceStringMB
　概　要　マルチバイト文字列としての文字列を置換する。

・ 文字列置換(WC)
　関数名　OpenSOAPStringReplaceStringWC
　概　要　ワイドキャラクタ文字列としての文字列を置換する。

・ 文字列置換
　関数名　OpenSOAPStringReplaceString
　概　要　文字列を置換する。

・ 文字列消去
　関数名　OpenSOAPStringClear
　概　要　文字列を消去する。

・ 文字列追加(MB)
　関数名　OpenSOAPStringAppendMB
　概　要　マルチバイト文字列を追加する。

・ 文字列追加(WC)
　関数名　OpenSOAPStringAppendWC
　概　要　ワイドキャラクタ文字列を追加する。

・ 文字列複製
　関数名　OpenSOAPStringDuplicate
　概　要　文字列を複製する。

・ 部分文字列の取得
　関数名　OpenSOAPStringGetSubstring
　概　要　部分文字列を取得する。

・ 文字エンコーディング指定データ変換
　関数名　OpenSOAPStringGetSubstring
　概　要　指定した文字エンコーディングデータに変換する。

・ サイズ指定付き文字エンコーディング指定データ変換

関数名　OpenSOAPStringGetSubstring

概　要　指定した文字エンコーディングデータに指定したサイズ分変換する。

文字列キー連想配列操作用 API

以下の API は文字列キー連想配列に対する操作を提供する。

・ 文字列キー連想配列の作成

関数名　OpenSOAPStringHashCreate

概　要　文字列キー連想配列を作成する。

・ 文字列キー連想配列の開放

関数名　OpenSOAPStringHashRelease

概　要　文字列キー連想配列を開放する。

・ 文字列キー連想配列の全要素の消去

関数名　OpenSOAPStringHashRelease

概　要　文字列キー連想配列の全要素を消去する。

・ 文字列キー連想配列の要素の削除

関数名　OpenSOAPStringHashRemoveKey

概　要　文字列キー連想配列の要素をキーを指定して削除する。

・ 文字列キー連想配列への値の設定(MB)

関数名　OpenSOAPStringHashSetValueMB

概　要　マルチバイト文字列キーによる値を設定する。

・ 文字列キー連想配列への値の設定(WC)

関数名　OpenSOAPStringHashSetValueWC

概　要　マルチバイト文字列キーによる値を設定する。

・ 文字列キー連想配列への値の設定

関数名　OpenSOAPStringHashSetValue

概　要　OpenSOAP 文字列キーによる値を設定する。

・ 文字列キー連想配列からの値の取得(MB)
　関数名　OpenSOAPStringHashGetValueMB
　概　要　マルチバイト文字列キーによる値を取得する。

・ 文字列キー連想配列からの値の取得(WC)
　関数名　OpenSOAPStringHashGetValueWC
　概　要　マルチバイト文字列キーによる値を取得する。

・ 文字列キー連想配列からの値の取得
　関数名　OpenSOAPStringHashGetValue
　概　要　OpenSOAP 文字列キーによる値を取得する。

・ 文字列キー連想配列大きさの取得
　関数名　OpenSOAPStringHashGetSize
　概　要　文字列キー連想配列の大きさを取得する。

・ 文字列キー連想配列全キーの取得
　関数名　OpenSOAPStringHashGetKeys
　概　要　文字列キー連想配列の全キーを取得する。

・ 文字列キー連想配列全値の取得
　関数名　OpenSOAPStringHashGetValues
　概　要　文字列キー連想配列の全値を取得する。

・ 文字列キー連想配列全値に対する関数の適用
　関数名　OpenSOAPStringHashApplyToValues
　概　要　文字列キー連想配列の全値に対して関数を適用する。

2.2.3.　OpenSOAP Client 構築用 API の実装

OpenSOAP Client 構築用 API では、クライアント側が OpenSOAP サーバと通信するために必要な API を中心に開発した。以下の API がある。

(1) クライアントソケット用の API

この API は、BSD ソケットインターフェイスを利用した、TCP によるクライアント機能に特化したインターフェースを提供する。すなわち、サーバへの接

続、切断。接続後の読み込みおよび書き込みである。これを用いて以下のトランスポート用 API の実装を行っている。

(2) トランスポート用の API

この API は SOAP トランスポートとして必要なインターフェースを提供する。具体的には接続時に、URL による指定を行い、SOAP メッセージの送受信を行うインターフェースとなっている。

2.2.4. OpenSOAP Service 構築用 API の実装

OpenSOAP Service 構築用 API では、OpenSOAP サービス側が OpenSOAP サーバと通信するために必要な API、または、OpenSOAP サービスの登録と実行用の API を中心に開発した。以下の API がある。

(1) Connect 用 API

この API 実装は、実際には、(3)サービス実行用 API に隠蔽されており、ユーザが意識して使う必要はないようになっている。具体的には、OpenSOAP サーバとのプロトコルは、HTTP のミニマムセットとして規定しており、HTTP サーバと類似の動作をするように実装を行った。

(2) サービス登録用 API

この API では、OpenSOAP サービスを提供するために必要な情報を設定する。具体的には、OpenSOAP サービスの名前、種別、また、サービスを実行する処理の登録も行う。これらの設定情報に基づいて、(3) サービス実行用 API により適切に OpenSOAP サービスが実行される。

(3) サービス実行用 API

この API は(2)サービス登録用 API で行った設定にしたがってサービスを実行するインターフェースを一つだけ提供している。ユーザはこの API を呼び出すだけで、OpenSOAP サービスを提供できるようになる。

2.3. マルチプラットフォーム対応

OpenSOAP API 実装は、まず Linux を含む Unix 系 OS と Windows 系 OS で動作するように行った。実際にこれらに対応することができれば、多くのプラットフォームでの SOAP の利用が可能となるためである。しかしながら、Unix 系 OS とひとくくりにしているが、実際には、様々な亜種が存在し、その亜種間での差異があるために、限られた期間内での対応は困難であった。その解決のため、GNU Autoconf, Automake, Libtool (以下、これら 3 つをまとめて GNU Autotools と呼ぶ)を利用することにより、この問題への解決を試みた。GNU Autotools とは、現存する様々な Unix 亜種上で動作するプログラムを開発するために、Unix 亜種間の

差異を吸収するためのツールである。これらを利用することにより、数多くの Unix 系 OS への対応がやりやすくなった。また、Linux が POSIX に準拠していることと併せて、現在では数多くのユーザが存在することから、Unix 系 OS での主な開発は Linux 上で行うこととした。さらに、Windows の POSIX 互換関数の利用することで、非互換関数の実装や、代用を行うことにより、C プリプロセッサの機能を用い、Windows 系 OS との統合を試みた。その結果、Unix 系 OS と Windows 系 OS で、同一のインターフェースを提供することに成功した。

2.4. 国際化対応

XML では、Unicode を採用することで、国際化に対応しようとしている。そのため、OpenSOAP API では、Unicode 処理を含めた、国際化プログラミングへの対応が不可欠であった。

2.4.1. SOAP 仕様における国際化

SOAP では XML をパケットデータとして用いている。XML とは拡張可能なマークアップ言語であり、これまでにインターネット上で使用されていた HTML に、SGML の柔軟性とパワーを組み合わせるように、かつ、SGML より単純化されるように設計された。また、XML は設計目標の一つに、国際化をあげている。XML 自体は、いわゆるテキストデータだが、使用する文字に関しては、7 ビットの ASCII 文字セットには限定せず、“タブ、復帰、改行、Unicode と ISO/IEC 10646 が定める正当な文字”と規定している。また、文字符号化方式については、UTF-8 および UTF-16 については、必ず処理可能なことが規定されており、それ以外の符号化方式については、IANA に登録済の符号化方式を宣言することで使用可能になっている。このように、XML 自体は国際化を意図して設計され、そのように規定されているので、それを用いた SOAP 仕様自体も、国際化対応を行っていると考えられる。

2.4.2. OpenSOAP API における国際化対応

SOAP 自体に国際化対応がなされているとしても、それを利用するソフトウェアが自動的に国際化対応となるわけではない。ソフトウェアの国際化とは、世界の様々な言語や習慣に対応するために、ソフトウェアをできるだけ共通化/汎用化することにより個別対応を少なくし、効率よく開発を行えるようにすることである。そのため、OpenSOAP API では、マルチプラットフォーム化も考慮し、プラットフォーム間で異なる国際化フレームワークを実装に隠蔽し、統一したインターフェースを提供することで対応した。

2.5. 他のプログラミング言語用実装

C 言語での実装を行うことにより、C++を始めとした他のプログラミング言語での利用が可能となった。また、Java 言語においては、OpenSOAP クライアントおよびサービスを、ネイティブコードにより実装可能とした。

2.5.1. Java 言語用実装

Java 言語に関しては、既存の RPC クライアント、OpenSOAP 拡張である非同期クライアント、それと、OpenSOAP サーバ用サービスの 3 種類について、四則演算サービスを例に実装を行った。

2.6. セキュリティモジュールとの結合

OpenSOAP により利用されるメッセージ、すなわち SOAP Message は、インターネットを介してやり取りされる。インターネットは外部に開かれたネットワークであるため、SOAP Message は、その伝達経路上の悪意を持った第三者の介在による取引リスクにさらされる。

OpenSOAP を B2B, B2C での実用に供するためには、様々な不正が SOAP Message に対して行われる可能性を排除するセキュリティ技術の実装が必要となる。

そのため、サブテーマ 3「総合セキュリティ機能の実装」で行われた実装を OpenSOAP API として利用可能とするため、インターフェースを統一し、OpenSOAP API への組み込みを行った。これにより、SOAP Message に対する暗号化/復号化と認証処理が OpenSOAP API の元で可能となった。

2.7. サンプルコードの作成

OpenSOAP API の処理とサンプルコードの作成手順を説明する。

ここでは、Hello Sample を例として、基本の部分を説明する。

2.7.1. OpenSOAP API の処理

OpenSOAP API の処理は、大きく分けて、クライアントプログラムの作成とサービスプログラムの作成がある。以下詳しく説明する。

2.7.2. 共通の処理

クライアントプログラムの作成とサービスプログラムの作成には、共通する処理がある(図 2.7.1)。それは：

(1) OpenSOAP API の初期化処理

プログラム作成の初めに、OpenSOAP API の初期化を行う。以下の関数を利用

する。

OpenSOAPInitialize(NULL);

(2) OpenSOAP API の終了処理

プログラム作成の最後に、OpenSOAP API を終了させる。以下の関数を利用する。

OpenSOAPUltimate();

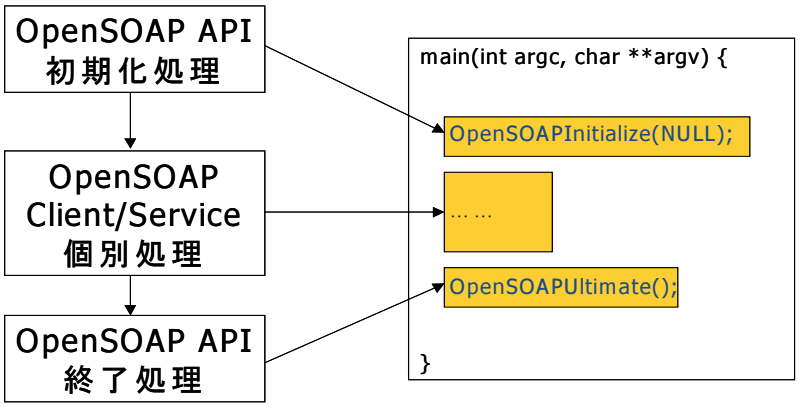

図 2.7.1 OpenSOAP API 処理の共通処理

2.7.3. クライアントプログラムの作成

クライアントプログラムの作成は、三つの作業に分けられる。

(1) リクエストの作成

(2) SOAP サービスの呼び出し

(3) レスポンスの解析

以下、三つの作業について、説明する。

リクエストの作成

リクエスト作成の目的は図 2.7.2 に示すような OpenSOAP Request Message を作成することである。

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Body>
      <m:Hello xmlns:m="http://namespaces.opensoap.jp">
         <MyName>foo</MyName>
      </m:Hello>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

図 2.7.2 OpenSOAP Request Message（Hello Sample）

リクエストの作成は図 2.7.3 に示している。手順は：

(1) Envelope の作成

(2) Body Block の追加

(3) Namespace の設定

(4) 値の設定

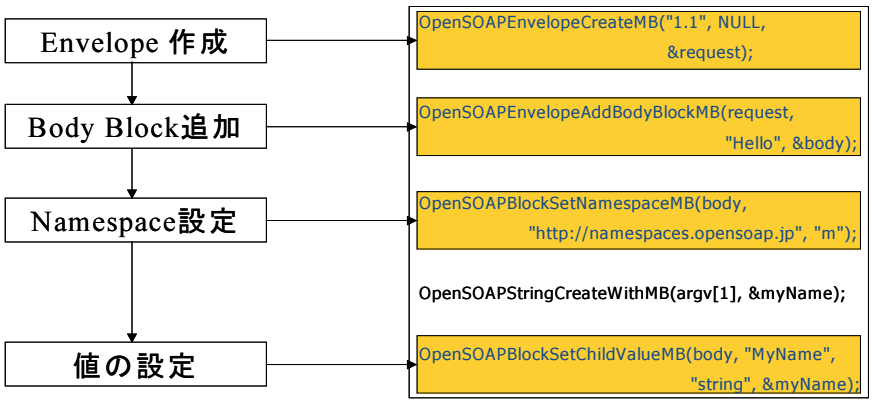

図 2.7.3 Client Program のリクエスト作成

以下、それぞれの手順について詳しく説明する。

(1) Envelope の作成

Envelope の作成には、以下の関数を利用する。

OpenSOAPEnvelopeCreateMB(“1.1”, NULL, &request);

図 2.7.2 の Request Message で説明すると、Request Message 中のルート要素<Envelope></Envelope>を作成し、入力したバージョンパラメータにより、Envelope 要素の Namespace を作成する。

(2) Body Block の追加

Body Block の追加には、以下の関数を利用する。

OpenSOAPEnvelopeAddBodyBlockMB(request, “Hello”, &body);

図 2.7.2 の Request Message で説明すると、Response Message 中の<SOAP-ENV:Body></SOAP-ENV:Body>、<Hello></Hello>の二つ要素を作成する。Body Block が存在していない場合、この関数を利用すると、まず<SOAP-ENV:Body></ SOAP-ENV:Body>の SOAP Body 要素を作成する。次は SOAP Body 要素直下の Body Block<Hello></Hello>を作成する。もし Body Block が既に一個以上存在している場合、SOAP Body 要素直下の Body Block だけを作成する。

(3) Namespace の設定

Namespace の設定には、以下の関数を利用する。

OpenSOAPBlockSetNamespaceMB(body, “http://namespaces.opensoap.jp”, “m”);

図 2.7.2 の Request Message で説明すると、Namespace の prefix を“m”に、Namespace の URI を“ http://namespaces.opensoap.jp”にして、Message 中の<Hello></Hello>の要素に Namespace を設定する。

(4) 値の設定

値の設定には、以下の関数を利用する。

OpenSOAPBlockSetChildValueMB(body, “MyName”, “string”, &myName);

図 2.7.2 の Request Message で説明すると、<Hello></Hello>の要素に子要素として<MyName></MyName>要素を追加し、要素の値として string 型の myName を設定する。MyName の具体的な値は“foo”。

SOAP サービスの呼び出し

SOAP サービスの呼び出しというのは、OpenSOAP のクライアント側が作成した Request Message を送るには、通信するためのトランスポートオブジェクトの作成、接続先の設定、SOAP サービスの呼出とトランスポートの開放という一連の作業である(図 2.7.4)。以下詳しく説明する。

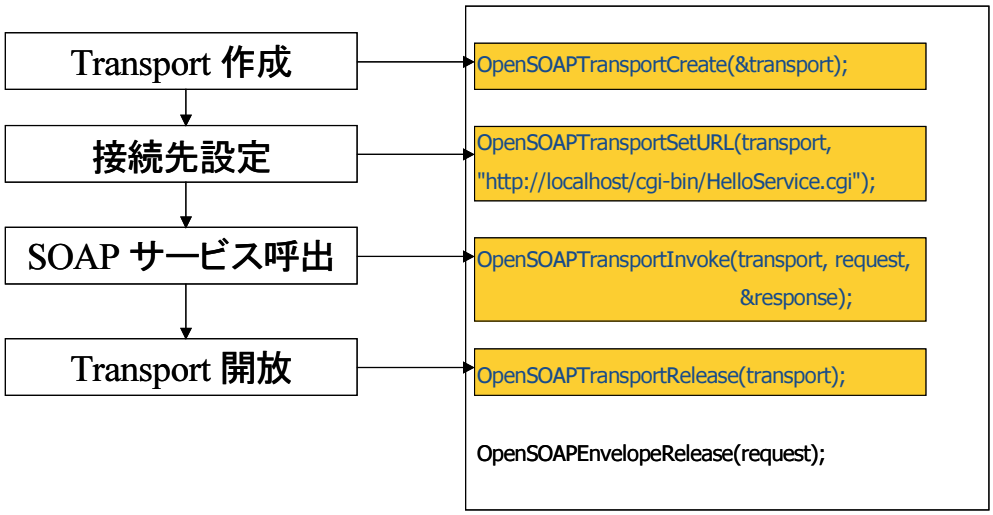

図 2.7.4 Client Program の SOAP Service 呼出

(1) Transport の作成

Transport の作成には、通信用のトランスポートオブジェクトを作成する。以下の関数を利用する。

OpenSOAPTransportCreate(&transport);

(2) 接続先の設定

接続先の設定には、通信先の URL を設定する。以下の関数を利用する。

OpenSOAPTransportSetURL(transport,
"http://localhost/cgi-bin/HelloService.cgi");

(3) SOAP サービスの呼出

SOAP サービスの呼出では、設定したトランスポートの URL へ Request Message を送り、また SOAP サービスの実行結果を乗せている Response Message を受け取る。以下の関数を利用する。

OpenSOAPTransportInvoke(transport, request, &response);

(4) Transport の開放

Transport の開放では、SOAP サービスから Response Message を受け取った後、作成したトランスポートオブジェクトを開放する。以下の関数を利用する。

OpenSOAPTransportRelease(transport);

レスポンスの解析

レスポンスの解析というのは、受け取った Response Message を解析し、実行結果を Message から取り出す作業である。図 2.7.2 の Request Message に対して、図 2.7.5 に示す Response Message を受け取る。この Response Message に対する解析はレスポンス Block 取得、レスポンス値取得、値の処理という三つの手順からなる(図 2.7.6)。

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <m:HelloResponse xmlns:m="http://namespaces.opensoap.jp">
      <Reply>Hello, foo!</Reply>
    </m:HelloResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

図 2.7.5 OpenSOAP Response Message(Hello Sample)

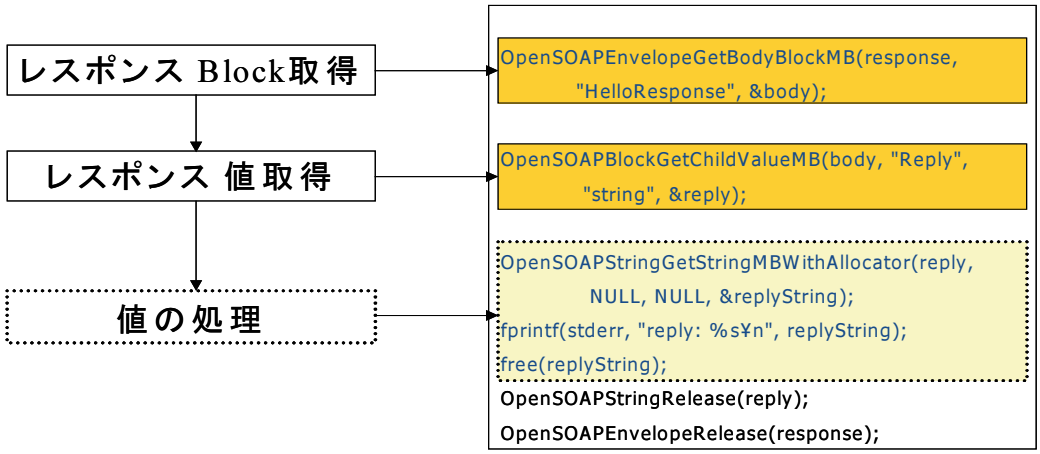

図 2.7.6 Client Program のレスポンス解析

(1) レスポンス Block 取得

レスポンス Block の取得では、Response Message 中の Body Block を取得する。図 2.7.2 に示す Response Message で説明すると、要素の<SOAP-ENV:Body></SOAP-ENV:Body>直下の Body Block <HelloResponse></HelloResponse>を取得する。以下の関数を利用する。

OpenSOAPEnvelopeGetBodyBlockMB(response, “HelloResopnse”, &body);

(2) レスポンス値の取得

レスポンス値の所得では、Response Message に含まれている実行の結果を取り出す。図 2.7.2 に示す Response Message で説明すると、Body Block <HelloResponse></HelloResponse>の子要素<Reply></Reply>の値”Hello,foo”を取得する。以下の関数を利用する。

OpenSOAPBlockGetChildValueMB(body, “Reply”, “string”, &reply);

(3) 値の処理

値の処理では、取得した値に対して、OpenSOAP String 型から一般の String 型に変換する。図 2.7.2 に示す Response Message で説明すると、取得した OpenSOAP String 型の“reply”に対して、一般の String 型の“replyString”に変換する。以下の関数を利用する。

OpenSOAPStringGetStringMBWithAllocator(reply, NULL, NULL, &replyString);

2.7.4. サービスプログラムの作成

サービスプログラムの作成は、三つの作業に分けられる。

(1) サービスの登録及び実行

(2) サービス関数内のリクエスト解析

(3) サービス関数内のレスポンス作成

以下、三つの作業について、説明する。

サービスの登録および実行

サービスの登録及び実行は図 2.7.7 に示す四つの手順になる。

(1) サービスの作成

(2) サービス関数の登録

(3) サービスの実行

(4) サービスの開放

以下、それぞれの手順について、説明する。

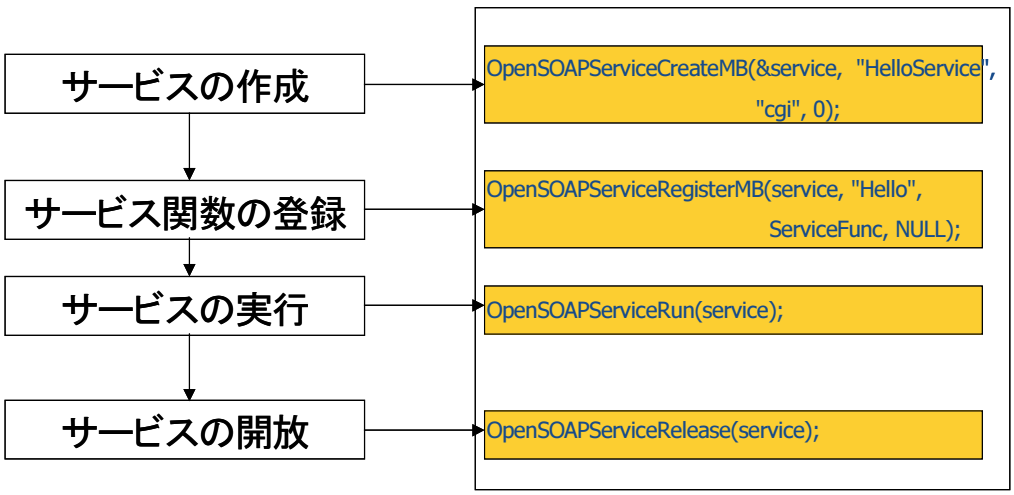

図 2.7.7 Service Program のサービス登録及び実行

(1) サービスの作成

サービスの作成では、サービスの名前と接続タイプを指定して、OpenSOAP Service オブジェクトを生成する。以下の関数を利用する。

OpenSOAPServiceCreateMB(&service, “HelloService”, “cgi”, 0);

(2) サービス関数の登録

サービス関数の登録では、生成されたサービスオブジェクトにサービスの機能を名前で指定して登録する。以下の関数を利用する。

OpenSOAPServiceRegisterMB(service, “Hello”, ServiceFunc, NULL);

(3) サービスの実行

サービスの実行では、Request Message を受け取り、ユーザが要求するサービスを実行させ、実行結果を Response Message として返す。以下の関数を利用する。

OpenSOAPServiceRun(service);

(4) サービスの開放

サービスの開放では、サービスの実行が完了してから、サービスオブジェクトを開放する。以下の関数を利用する。

OpenSOAPServiceRelease(service);

サービス関数内のリクエスト解析

サービス関数内のリクエスト解析とは、受け取った Request Message に対して解析し、パラメータの値を取得することである。図 2.7.8 に処理の流れを示す。

サービス関数内のリクエスト解析は、リクエスト Block の取得、リクエスト値取得、値の処理という三つの手順からなる。以下詳しく説明する。

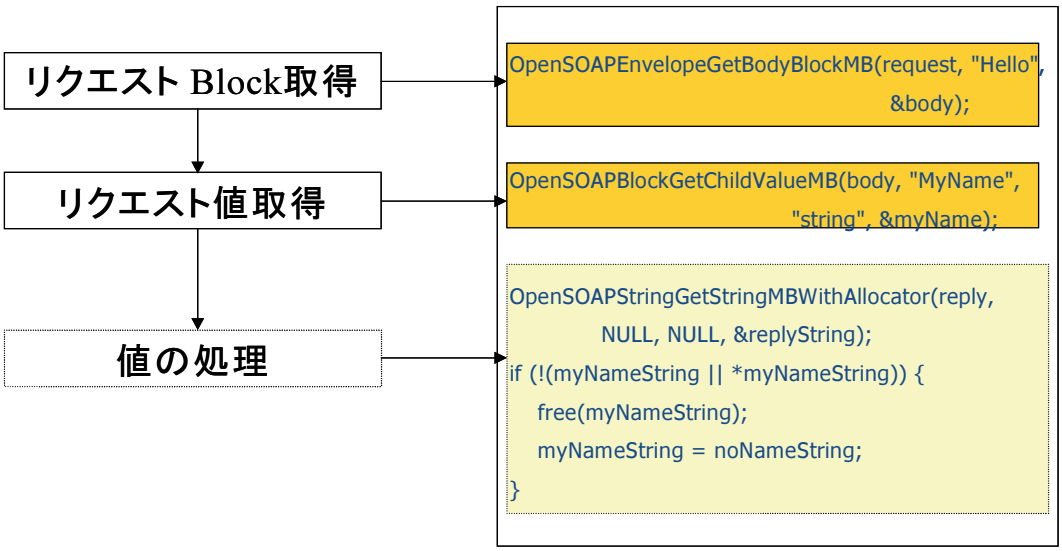

図 2.7.8 Service Program サービス関数内のリクエスト解析

(1) リクエスト Block の取得

リクエスト Block の取得では、指定した名前の Body Block を取得する。図 2.7.2 に示す Request Message で説明すると、Body Block の<Hello></Hello>を取得する。以下の関数を利用する。

OpenSOAPEnvelopeGetBodyBlockMB(request, “Hello”, &body);

(2) リクエスト値の取得

リクエスト値の取得では、サービスを実行するためのパラメータを取得する。図 2.7.2 に示す Request Message で説明すると、Body Block <Hello></Hello>の子要素<MyName></MyName>の値を取得する。以下の関数を利用する。

OpenSOAPBlockGetChildValueMB(body, “MyName”, “string”, &myName);

(3) 値の処理

値の処理では、取得してきたパラメータ myName の型 OpenSOAP String を一般の String 型の myNameString に変換する。以下の関数を利用する。

OpenSOAPStringGetStringMBWithAllocator(myName,
NULL, NULL, &myNameString);

サービス関数内のレスポンス作成

サービス関数内のレスポンス作成とは、サービスの実行結果を送るための Response Message(図 2.7.5)を作成することである。図 2.7.9 に処理の流れを示す。

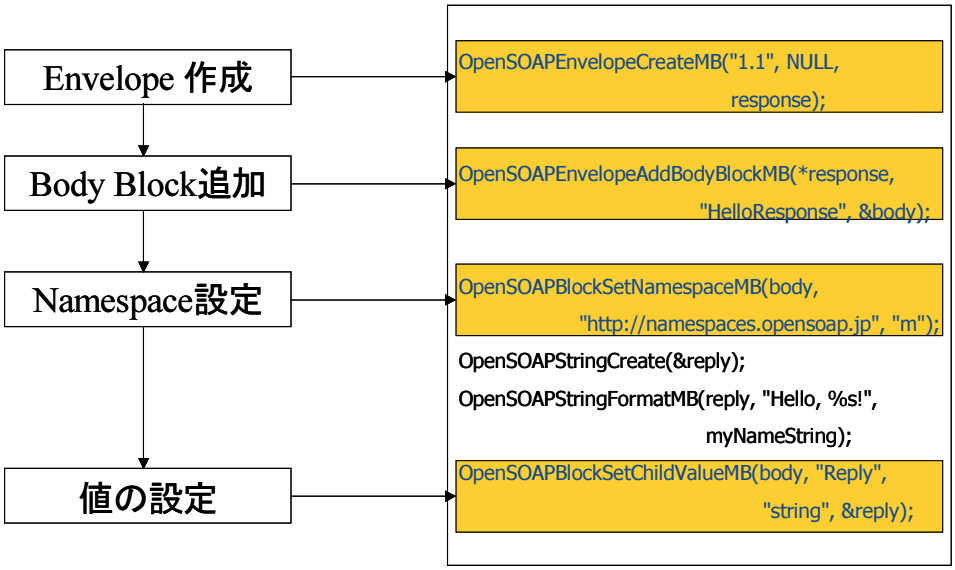

図 2.7.9 Service Program サービス関数内のレスポンス作成

サービス関数内のレスポンス作成は、Envelope 作成、Body Block の追加、Namespace の設定、値の設定という四つの手順からなる。以下詳しく説明する。

(1) Envelope の作成

Envelope の作成では、Response Message のルート要素<Envelope></Envelope>、またバージョンパラメータにより、Envelope 要素の Namespace を作成する。以下の関数を利用する。

OpenSOAPEnvelopeCreateMB(“1.1”, NULL, &response);

(2) Body Block の追加

Body Block の追加では、Response に関する Body Block をルート要素の直下に追加する。以下の関数を利用する。

OpenSOAPEnvelopeAddBodyBlockMB(response, “HelloResponse”, &body);

図 2.7.5 の Response Message で説明すると、Message 中の<SOAP-ENV:Body> </SOAP-ENV:Body>、<HelloResponse> </HelloResponse>の二つ要素を作成する。Body

Block が存在していない場合、このの関数を利用すると、まず<SOAP-ENV:Body></SOAP-ENV:Body>の SOAP Body 要素を作成する。次は SOAP Body 要素直下の Body Block<HelloResponse></HelloResopnse>を作成する。もし Body Block が既に一個以上存在している場合、SOAP Body 要素直下の Body Block だけを作成する。

(3) Namespace の設定

Namespace の設定では、Body Block に必要な Namespace を設定する。以下の関数を利用する。

OpenSOAPBlockSetNamespaceMB(body, “http://namespaces.opensoap.jp”, “m”);

図 2.7.5 の Response Message で説明すると、Namespace の prefix を“m”に、Namespace の URI を “http://namespaces.opensoap.jp” にして、Message 中の <HelloResponse> </HelloResponse>の要素に Namespace を設定する。

(4) 値の設定

値の設定では、サービス実行結果用の要素を作成し、サービス実行結果を要素の値として設定する。以下の関数を利用する。

OpenSOAPBlockSetChildValueMB(body, “Reply”, “string”, &reply);

図 2.7.5 の Response Message で説明すると、<HelloResponse></HelloResponse>の要素に子要素として<Reply></Reply>要素を追加し、要素の値として string 型の reply を設定する。MyName の具体的な値は “Hello, foo!” となる。