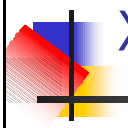
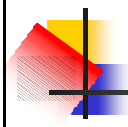


Webサービス構築のための XML/SOAP関連仕様と開発手法

Sarion Systems Research
中村 浩士

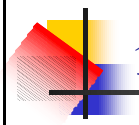
Webサービス構築のためのXML/SOAP関連仕様と開発手法



全体のAgenda(1)

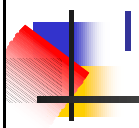
- I. Webサービスとは
- II. Webサービスを支える技術
  1. SOAP: XMLによるデータ転送
  2. WSDL: XMLによるサービス形式の記述
  3. UDDI: SOAPにより自動運用と可用性をサポートするグローバルディレクトリ
  4. Webサービスのセキュリティ

Copyright 2001 © Sarion Systems Research



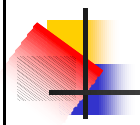
全体のAgenda(2)

- III. Webサービス開発手法
  1. Microsoft: .NET
  2. Apache-SOAP
  3. IBM: Web Services Toolkit + Apache-SOAP
  4. UDDIへの登録と検索
  5. 各種ツールの相互運用性



I. Webサービスとは

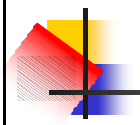
Sarion Systems Research
中村 浩士



「Webサービス」ブーム

- 2000年後半より、各ツールベンダから「Webサービス」サポートツールが発表されている
  - Microsoft: .NET
  - IBM: WebSphere
  - HP、BEA、Borland、SilverStream、富士通…

Webサービス構築のためのXML/SOAP関連仕様と開発手法



「Webサービス」狭義と広義

- 狭義の「Webサービス」
  - Microsoft、IBMを中心に、ツールベンダによりボトムアップに仕様策定
  - SOAP、WSDL、UDDIなどの仕様群
- その他広義の「Webサービス」には
  - ebXML: electronic business XML
    - XMLによる企業間電子商取引の標準化を目指し、標準化団体によりトップダウンに仕様策定
  - RosettaNet、etc…

「Webサービス」って?

- 標準仕様を積極的に活用
  - HTTP、SMTP、XML関連仕様群、etc…

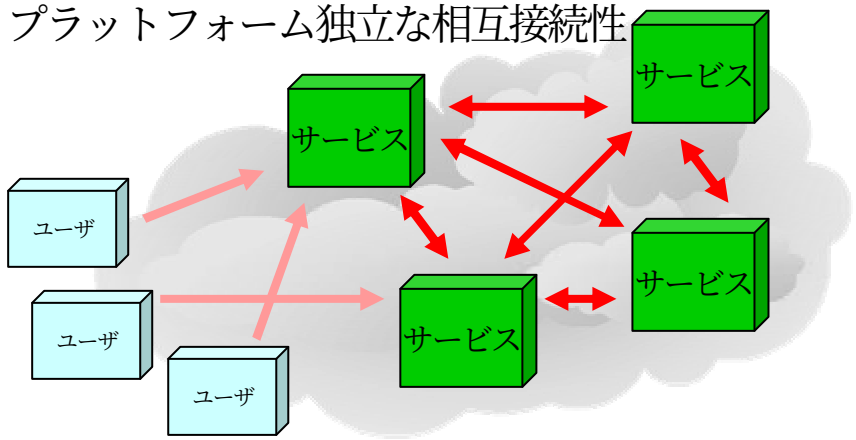
  →プラットフォーム独立
- 疎結合分散アプリケーション環境
  - インターネット上に配置されたサービスの部品を動的に（実行時に）結合

  →柔軟性、敏捷性（agile）

SARION™ SYSTEMS RESEARCH

Webサービスの特徴(1)

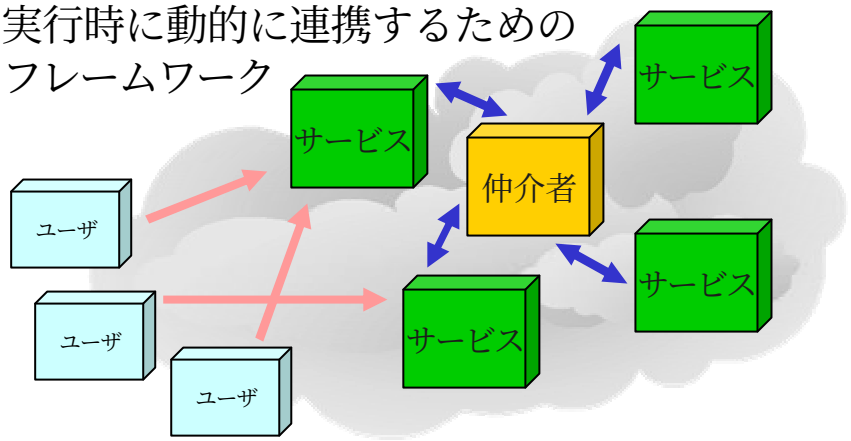
- プラットフォーム独立な相互接続性



SARION™ SYSTEMS RESEARCH

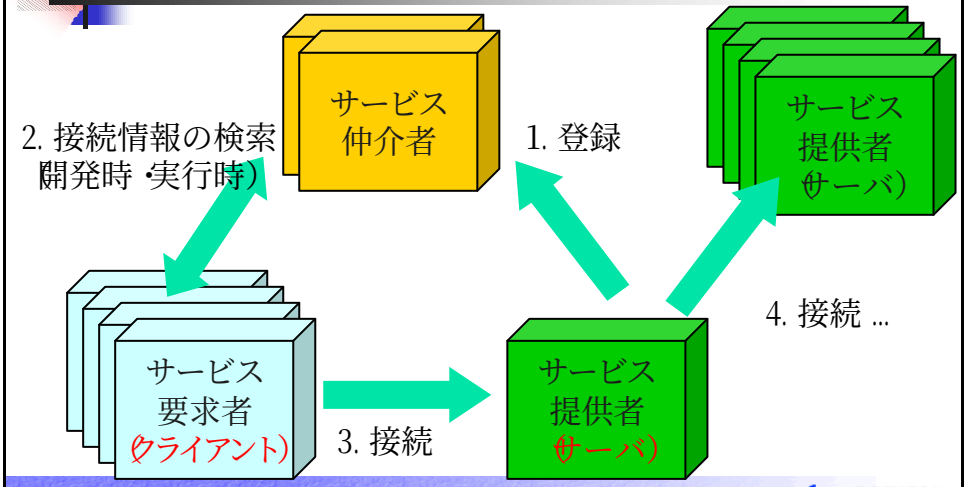
Webサービスの特徴(2)

- 実行時に動的に連携するための
  フレームワーク

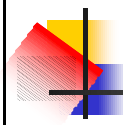


SARION™ SYSTEMS RESEARCH

Webサービス連携の流れ



SARION™ SYSTEMS RESEARCH

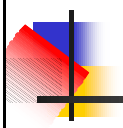


Webサービスを支える仕様

- フレームワークの界面仕様がまず定められ、各種ミドルウェアがサポートする形で発展
  - プラットフォーム独立 相互接続性）に寄与
- SOAP
  - メッセージフォーマットに関する仕様
- WSDL
  - サービス界面仕様の記述フォーマットに関する仕様
- UDDI
  - サービス連携をサポートするディレクトリに関する仕様および実装

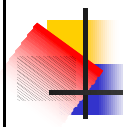
II. 関連仕様解説

Sarion Systems Research
中村 浩士



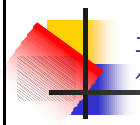
II. 関連仕様解説

1. SOAP: XMLによるデータ転送



SOAPとは?

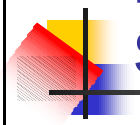
- 目的: 疎結合な分散環境において構造化データを交換
- メッセージングフォーマット
  - データフォーマットとしてXMLを採用
  - トランスポートに依存しない
- 単純で高い拡張性を持つ
- 直行する4つの仕様群から成る



規定外の内容

- 提供されるサービス情報の定義（→WSDL）
- 分散オブジェクト管理
- トランザクション管理
- セキュリティ管理
- メッセージのboxcarringとbatching
  - 複数メッセージの一括送信、一括処理

SARION™ SYSTEMS RESEARCH



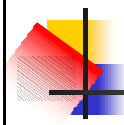
RPC、HTTPと共にSOAPを利用する例

- 例: ウェブサーバによる株価終値サービス

```
price GetLastTradePrice( codeNasdaq );
```

NASDAQコードをパラメータとして呼び出すと
株価終値を返すサービス

SARION™ SYSTEMS RESEARCH

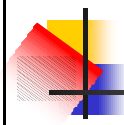


SOAP-RPC/HTTP request

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<Envelope>
  <Body>
    <GetLastTradePrice>
      <codeNasdaq>ABC</codeNasdaq>
    </GetLastTradePrice>
  </Body>
</Envelope>
```

※Namespaceなどの詳細は省略している

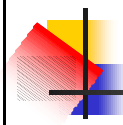
Webサービス構築のためのXML/SOAP関連仕様と開発手法



SOAP-RPC/HTTP response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<Envelope>
  <Body>
    <GetLastTradePriceResponse>
      <price>34.5</price>
    </GetLastTradePriceResponse>
  </Body>
</Envelope>
```

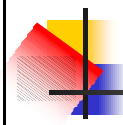
※Namespaceなどの詳細は省略している



SOAP/1.1仕様の構成

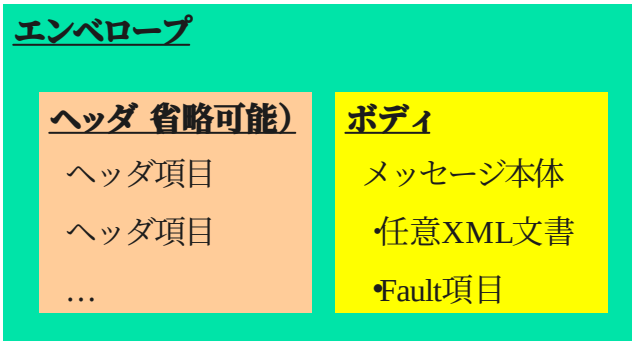
1. SOAP Envelope（SOAP/1.2: Part1で規定）
   - SOAPメッセージの構造を規定
2. SOAP Encoding（SOAP/1.2: Part2で規定）
   - 構造化データのXML文書化規則
3. HTTP binding（SOAP/1.2: Part2で規定）
   - HTTP用のトランスポートバインディング
4. RPC適用（SOAP/1.2: Part2で規定）
   - SOAPでRPCコールを実現する方法

Webサービス構築のためのXML/SOAP関連仕様と開発手法



(1) SOAP Envelope

SOAPメッセージの構造を規定する



Copyright 2001 © Sarion Systems Research

SOAPエンベロープのサンプル

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    :
  </SOAP-ENV:Header>

  <SOAP-ENV:Body>
    :
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


ボディにXML文書を入れたサンプル

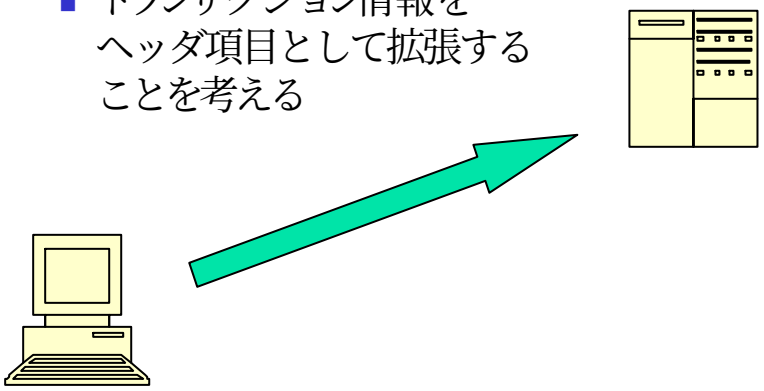
```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body
    xmlns:n="http://tempuri.org/notificationSubscriptionSample">
    <n:tickerSymbol>ABC</n:tickerSymbol>
    <n:notificatoinFrequency>P2D</n:notificatoinFrequency>
    <n:notificatoinDuration>P30D</n:notificatoinDuration>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Faultのサンプル

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Body>
   <SOAP-ENV:Fault>
    <faultcode>SOAP-ENV:Server</faultcode>
    <faultstring>Server Error</faultstring>
    <detail>
     <myfaultdetails>
      <message>Illegal code Nasdaq: ABC</message>
      <errorcode>1001</errorcode>
     </myfaultdetails>
    </detail>
   </SOAP-ENV:Fault>
</SOAP-ENV:Envelope>
```


SOAPヘッダの利用

- トランザクション情報を
  ヘッダ項目として拡張する
  ことを考える



SOAPヘッダを使用した拡張

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <t:Transaction xmlns:t="some-URI" SOAP-ENV:mustUnderstand="1">
      5
    </t:Transaction>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    :
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


SOAP encodingStyle属性

- Header、Body内のエンコード方式を指定
- 例: Body要素に、後述するSOAP Encoding仕様によりエンコードされたXML文書を収める

```
<SOAP-ENV:Body
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <...>
    (SOAP Encoding仕様に従ってエンコードして作られたXML文書)
  </...>
</SOAP-ENV:Body>
```

(2) SOAP Encoding

- SOAP EnvelopeのHeader、Bodyに収めるXML文書の記述方式の一つ
- 構造化データとXML文書の変換規則
  - 構造化データ → XML文書
    - 例: プログラミング言語における構造体をXML文書として記述
  - 構造化データ ← XML文書
    - 例: XML文書から構造体を復元

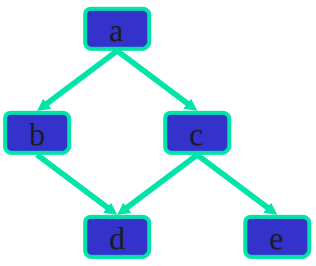
SARION™ SYSTEMS RESEARCH

SOAP Encoding

型定義

```
Struct a { b, c };
Struct c { d, e };
```

構造化データ



XML文書

```
<a>
  <b>
    <d href="#d1" />
  </b>
  <c>
    <d href="#d1" />
    <e>1.0F</e>
  </c>
</a>
<d id="d1">ABC</d>
```


SARION™ SYSTEMS RESEARCH

単純型のSOAP Encoding

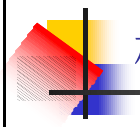
- XML Schema Part2: Datatypesに従う
- 例: 文字列 "NAKAMURA, Hiroshi" ⇔

```
<Person
   xsi:type="xsd:string"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
   xmlns:xsd="http://www.w3.org/2001/XMLSchema>
 NAKAMURA, Hiroshi
</Person>
```

- 例: 浮動小数点 1.23E12 ⇔

```
<Rate xsi:type="xsd:float">1.23E12</Rate>
```


SARION™ SYSTEMS RESEARCH

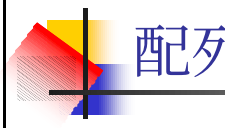


構造体のSOAP Encoding

- 例: 構造体 book { author, text { abstract, preface, intro } } ⇔

```
<book>
  <author>Henry Ford</author>
  <text>
    <abstract>Abstraction</abstract>
    <preface>Prefatory text</preface>
    <intro>This is a book.</intro>
  </text>
</book>
```

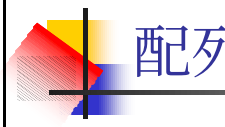

SARION™ SYSTEMS RESEARCH



配列のSOAP Encoding(1)

- 例: 配列 numbers[] ⇔

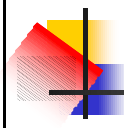
```
<numbers SOAP-ENC:arrayType="xsd:int[2]">
  <number>3</number>
  <number>4</number>
</numbers>
```

配列のSOAP Encoding(2)

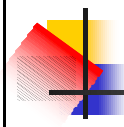
- 例: 2次元配列 a[2,3] ⇔

```
<a SOAP-ENC:arrayType="xsd:string[2,3]">
  <b>行1列1</b>
  <b>行1列2</b>
  <b>行1列3</b>
  <b>行2列1</b>
  <b>行2列2</b>
  <b>行2列3</b>
</a>
```



多重参照のSOAP Encoding

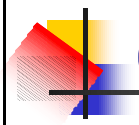
```
<book>
  <title>Title</title>
  <firstAuthor>
    <name>FIRST</name>
    <address href="#Address-1"/>
  </firstAuthor>
  <secondAuthor>
    <name>SECOND</name>
    <address href="#Address-1"/>
  </secondAuthor>
</book>

<Address id="Address-1">...</Address>
```

(3) HTTP binding

HTTP用のトランスポートバインディング

- ファイアウォール対策として、HTTPヘッダにSOAPActionを入れてもよい
- 受信側（HTTPサーバ側）は、正しく受信したら200を返す
- エラーが起こったらHTTP的には500、SOAP的にはFault項目を返す

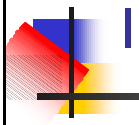


(4) RPC適用

SOAPでRPCコールを実現する方法

- RPCコール … [in]、[in/out]パラメータをメンバとする構造体
- RPCレスポンス … 戻り値、[out]、[in/out]パラメータをメンバとする構造体
- エラー通知 … Fault項目を利用

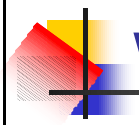
Webサービス構築のためのXML/SOAP関連仕様と開発手法



II. 関連仕様解説

2. WSDL: XMLによる
サービス形式の記述

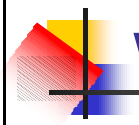
Copyright 2001 © Sarion Systems Research



WSDLとは?

- 目的: Webサービスコンポーネントが提供するサービスについて記述
  - サービスクライアントが開発時、実行時に利用
    - 開発時:
      サービスのインターフェイス詳細・呼び出し方式
    - 実行時: サービスの提供場所
- サービスの抽象化と再利用を図る

Webサービス構築のためのXML/SOAP関連仕様と開発手法



WSDL仕様の構成

1. サービス定義仕様
   - 一般的なサービス記述方式を規定
2. 各種メッセージングフォーマット・トランスポートへの対応付け規定
   - SOAPバインディング拡張
   - HTTP GET&POSTバインディング拡張
   - MIMEバインディング拡張

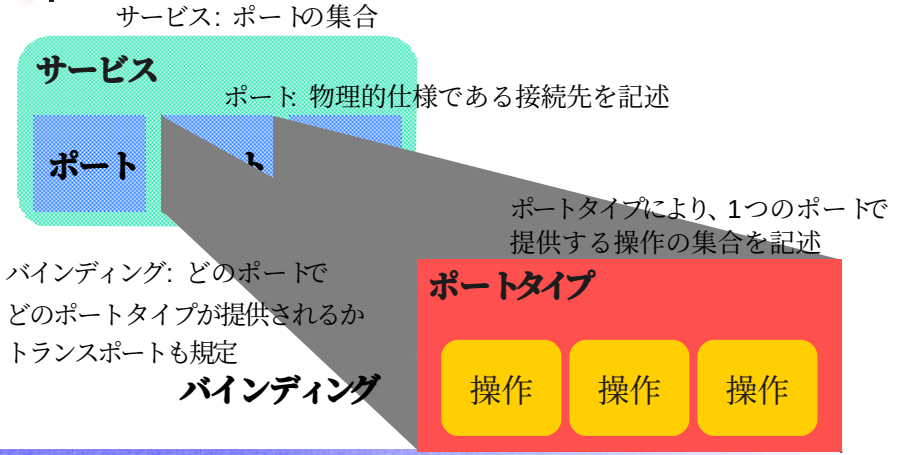
WSDLサービス定義仕様で用いられる概念

- サービス(service)
- ポート(port)
- バインディング(binding)
- ポートタイプ(portType)
- 操作(operation)
- メッセージ(mssage)
- タイプ(types)

SARION™ SYSTEMS RESEARCH

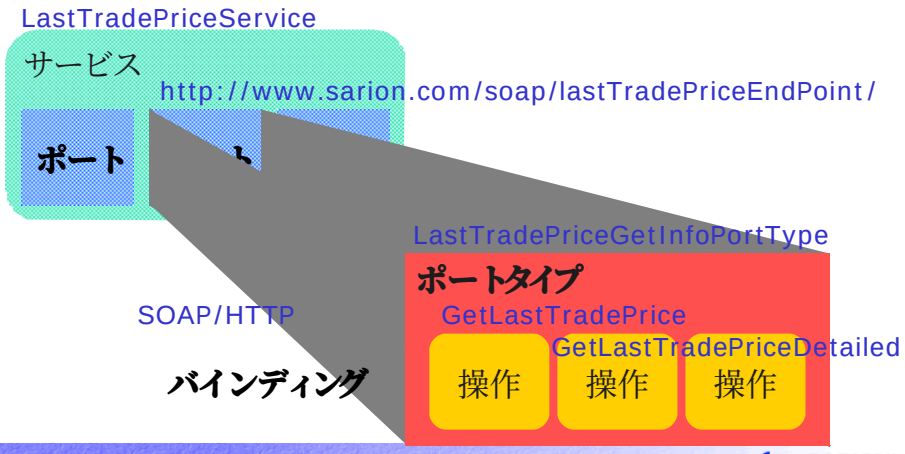
サービス定義の構成要素(1)

サービス: ポートの集合

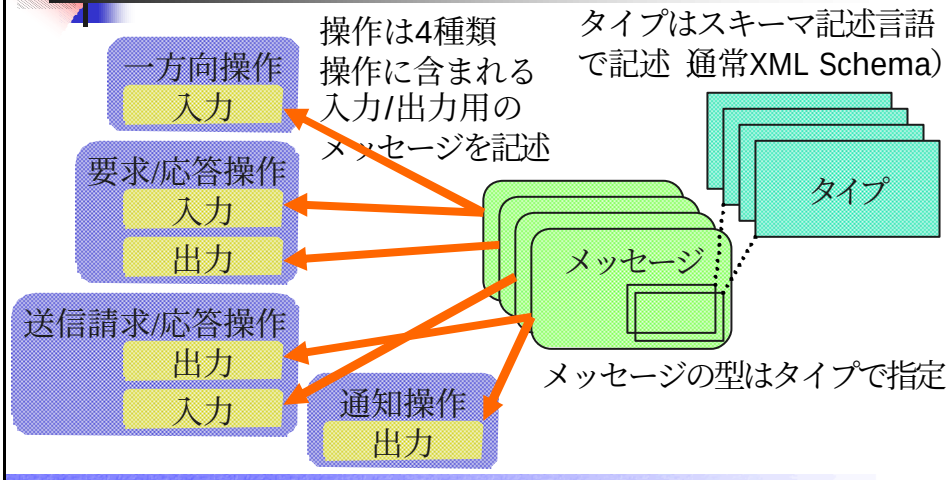


SARION™ SYSTEMS RESEARCH

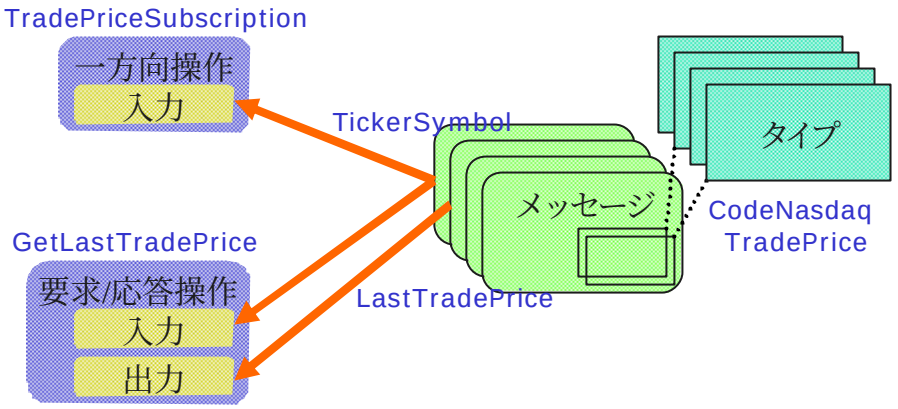
サービス定義の構成要素(1):例



サービス定義の構成要素(2)



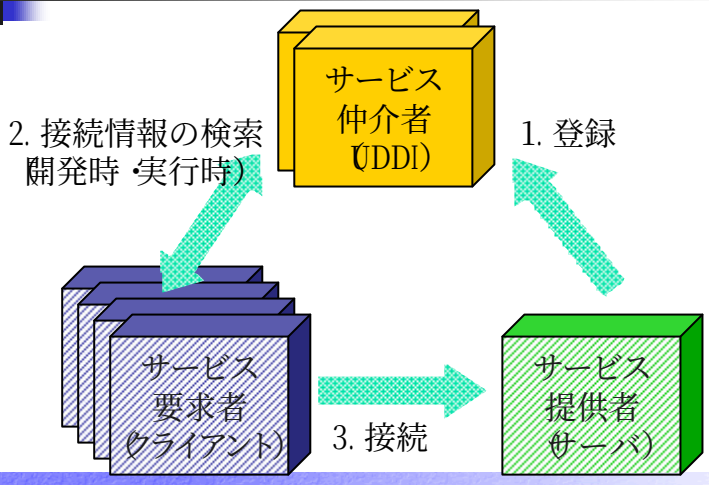
サービス定義の構成要素(2):例



II. 関連仕様解説

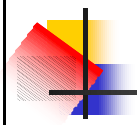
3. UDDI: SOAPにより
自動運用と可用性をサポートする
グローバルディレクトリ

Webサービスのアーキテクチャ



UDDIとは?

- 目的: Webサービスのディレクトリとして、仲介者に求められる機能を提供すること
- 登録と検索のAPI仕様を公開し、また実際に運用している
- 人だけでなく、プログラムからも利用可能(SOAPによる接続インターフェイス)
- サービスの接続情報をWSDLにより記述可能



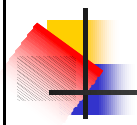
ビジネスレジストリ

- 3種類のディレクトリ

企業情報: 企業名、連絡先、etc.

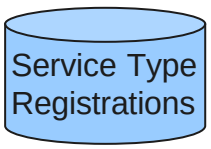
企業分類情報: 既存の標準に基づく3つ分類情報
NAICS: 北米産業分類コード、
UN/SPSC: 製品・サービスのタイプを分類、所在地

企業が提供するWebサービスの技術情報
提供しているサービス（サービスタイプ識別子）、
およびその接続情報



サービスタイプレジストリ

- Webサービスの仕様を登録するレジストリ

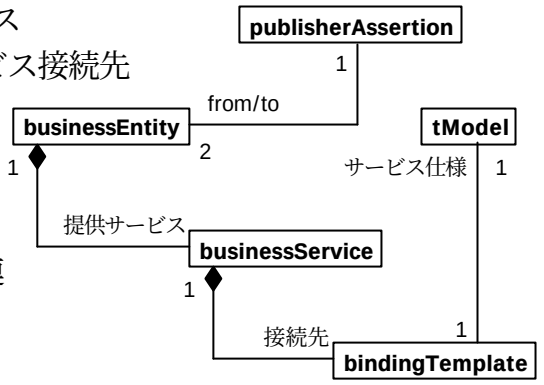


サービスタイプ:
サービスを利用するクライアントの
実装にあたり、開発者が読むべき
文書へのポインタ

tModelKey: サービスタイプ識別子

UDDIの内部データ構造

- businessEntity: 企業
- businessService: サービス
- bindingTemplate: サービス接続先
- tModel: サービス仕様
- publisherAssertion: 関連

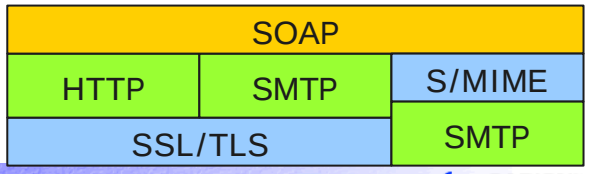


関連仕様解説

4. Webサービスのセキュリティ

Webサービスのセキュリティ

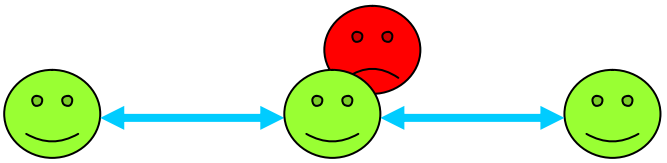
- 通信のセキュリティを考える場合、基本は下位のセキュリティレイヤ
- 既存のセキュリティレイヤ
  - HTTP: SSL/TLS
  - E-mail: SSL/TLS、S/MIME



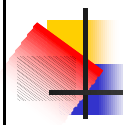
Webサービス構築のためのXML/SOAP関連仕様と開発手法

セキュリティレイヤの利用

- 既存のセキュリティレイヤを利用することにより、2点間の通信路の保護が可能
- 複数の通信路にまたがる場合、下位レイヤのインターネット標準には、保護可能なものがない



Copyright 2001 © Sarion Systems Research



Webサービスのセキュリティ

- 大抵はSSL・TLS、S/MIME（下位レイヤのセキュリティ）で間に合う
- 防ぎたい脅威の種類によっては防げない
  - なりすまし防止
  - 指定者以外への情報漏洩を防ぐ
  - 確実に一度だけ（リプレーアタック防止）

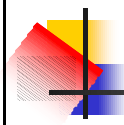
  →現状では、SOAPヘッダを利用した拡張により対応
  XML-SignatureをSOAPヘッダに付加する仕様も存在

SARION™ SYSTEMS RESEARCH

III. Webサービス開発手法

Sarion Systems Research
中村 浩士

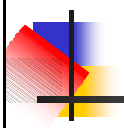
SARION™ SYSTEMS RESEARCH



Agenda

1. Microsoft: .NET
2. Apache-SOAP
3. IBM: Web Services ToolKit + Apache-SOAP
4. UDDIへの登録と検索
5. 各種ツールの相互接続性

Webサービス構築のためのXML/SOAP関連仕様と開発手法

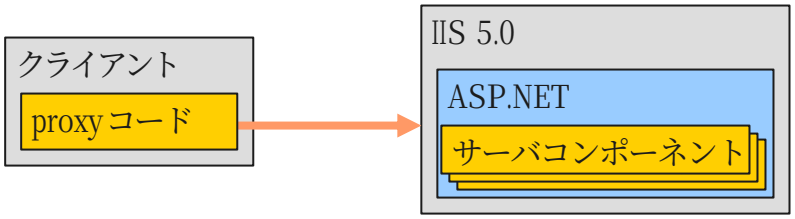


III. Webサービス開発手法

1. Microsoft: .NET

Copyright 2001 © Sarion Systems Research

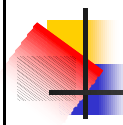
ASP.NET利用時の構成



サーバコンポーネントの作成

- ASP.NETによる開発
  - 以下のソースをabroad.asmxなどとして保存
  - [WebMethod]という宣言を追加するだけ

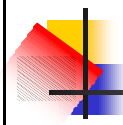
```
<%#@ WebService Language="C#" Class="Abroad" %>
using System.Web.Services;
public class Abroad {
   [WebMethod]
   public Hotel GetHotelInfo(string CRSCode, string HotelCode, string CityCode) {
      …
   }
}
```



クライアントの作成

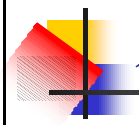
- VisualStudio.NETによる開発
  - VisualStudio.NETで参照設定を行う
  - クライアント側コードからnewして呼び出す

```
…
   Abroad anAbroad = new Abroad();
   Hotel aHotel = anAbroad.GetHotelInfo(crsCode, hotelCode, cityCode);
…
```

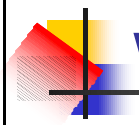
バックグラウンド(1)

- サーバ側asmxファイルの作成により、WSDLファイルが自動生成される
- クライアント側ツールにより、参照するWSDLファイルからproxyコードが生成される



バックグラウンド(2)

- proxyコード
  - クライアントコードからのメソッド呼び出しをSOAPリクエストに変換し、ASP.NETに送信
  - SOAPレスポンスをメソッドの戻り値に変換し、クライアントコードに返す
- ASP.NET
  - SOAPリクエストを元に、asmxファイル内のコード（例ではC#）のメソッドを呼び出す
  - メソッドの戻り値をSOAPレスポンスに変換し、proxyコードに返す



.NETに含まれるWebサービスソリューション

- ASP.NET Web Services
  - 上記例のように、ASP経由でサービスを公開・利用する場合
- .NET Remoting
  - COM同士の通信にSOAPを利用することもできる
  - オブジェクト参照のやり取りも可能
- ATL Server Web Services
  - C++ ATL Serverでサービスを公開する場合

詳細: http://www.microsoft.com/japan/developer/net/soap/hawksoap.asp

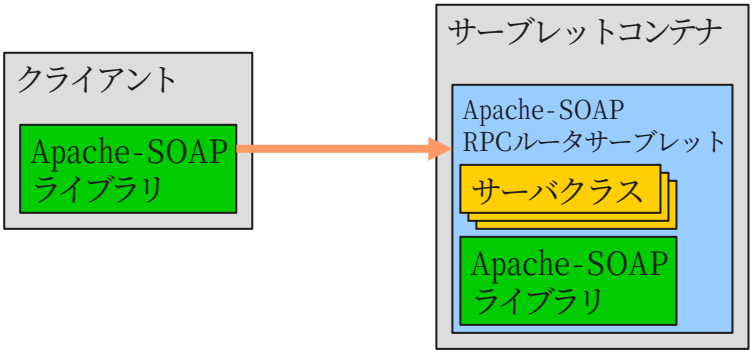
III. Webサービス開発手法

2. Apache-SOAP

SARION™ SYSTEMS RESEARCH

Apache-SOAP利用時の構成

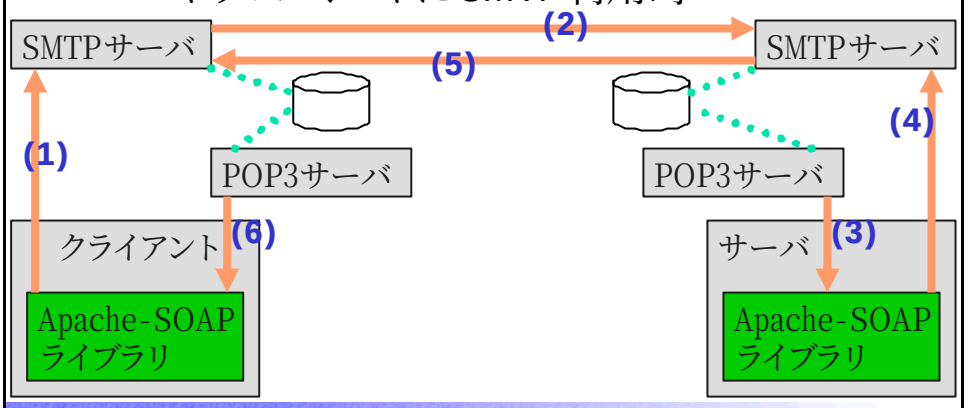
- トランスポートにHTTP利用時



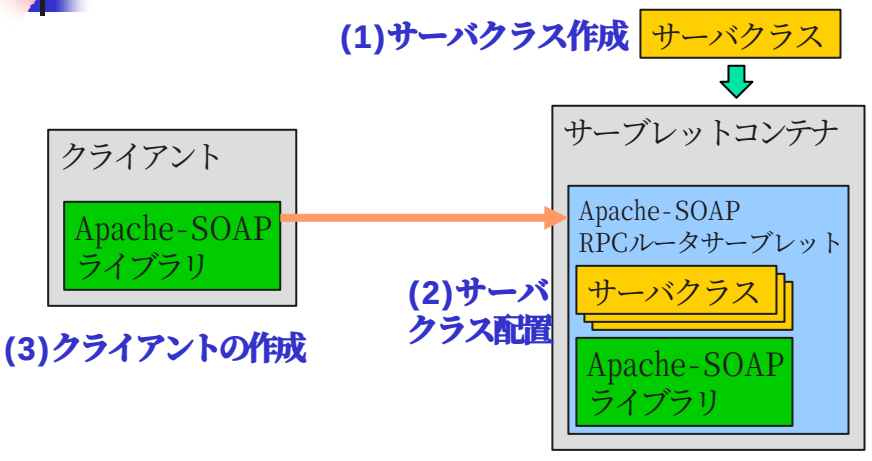
SARION™ SYSTEMS RESEARCH

Apache-SOAP利用時の構成

- トランスポートにSMTP利用時



Apache-SOAP利用時の開発の流れ



(1)サーバクラス作成

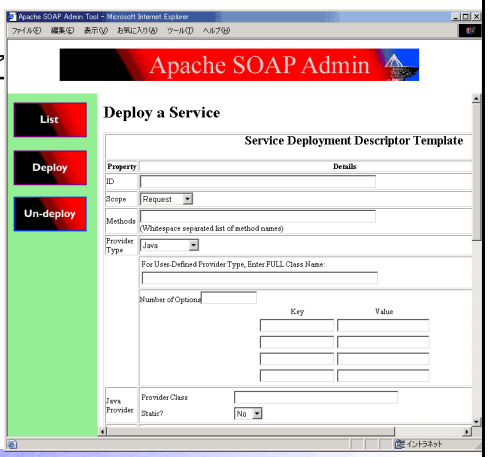
- 通常のJavaのクラスとして作成する
  - 例: ExchangeというクラスのgetRateというメソッドをサービスとして公開する

```
public class Exchange {
        public float getRate( String country1, String country2 ) {
        System.out.println( "getRate( " + country1 + ", " + country2 + " )" );
        return 123.45F;              // ToDo: ちゃんと計算しなさい!
    }
}
```


SARION™ SYSTEMS RESEARCH

(2)サーバクラス配置

- サーバクラスをサーブレットコンテナ内に配置し、サービスとしてアクセス可能に
  - 配置情報（Javaのクラス名、公開メソッド名他）をWebページで指定
  - 同様の情報をXMLファイルで定義し、コマンドラインから設定することも可能



SARION™ SYSTEMS RESEARCH

(3)クライアントの作成

- Dynamic invocation

```
Call call = new Call();  // ←資料にありませんでしたすいません。追加です。
URL url = new URL( "http://services.xmethods.net/soap" );
String urn = "urn:xmethods-CurrencyExchange";
SOAPHTTPConnection st =  new SOAPHTTPConnection();
call.setSOAPTransport( st );
call.setTargetObjectURI ( urn );
```

下準備

```
call.setMethodName( "getRate" );
call.setEncodingStyleURI ( Constants.NS_URI_SOAP_ENC );
Vector params = new Vector();
params.addElement( new Parameter( "country1", String.class, "USA", null ));
params.addElement( new Parameter( "country2", String.class, "Japan", null ));
call.setParams( params );  // ↑"JPN"→"Japan"、また順序が逆。
```

メソッド名とパラメータの設定

```
Response response = call.invoke( url, "" );
if ( !response.generatedFault() ) {
    Parameter result = response.getReturnValue();
    System.out.println( "Result = " + result.getValue() );
}
else { ...エラー処理... }
```

呼び出しと戻り値の取得

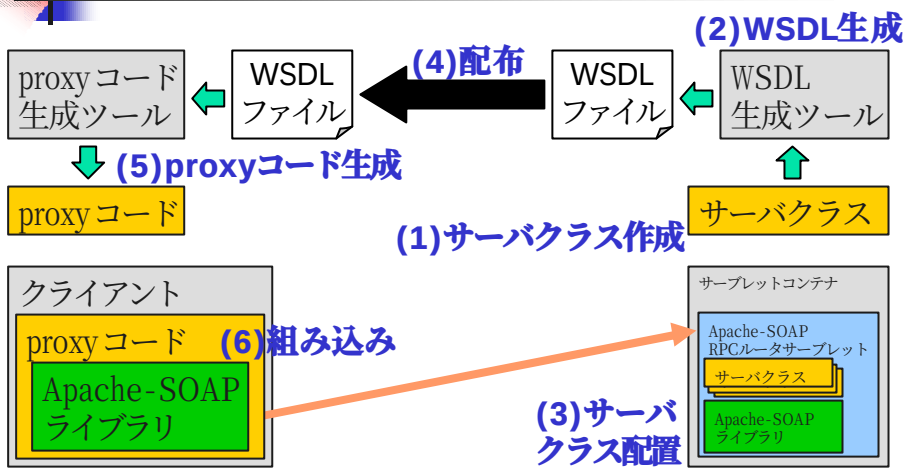
SARION™ SYSTEMS RESEARCH

III. Webサービス開発手法

3. IBM: Web Services ToolKit + Apache-SOAP

SARION™ SYSTEMS RESEARCH

WSTK利用時の開発の流れ



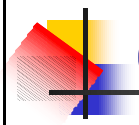
SARION™ SYSTEMS RESEARCH

Webサービス構築のためのXML/SOAP関連仕様と開発手法

(2)WSDL生成

- WSTKのwsdlgenツールにより生成可能
  - Javaのクラスファイル、EJB Jarファイル、COMのタイプライブラリからWSDLファイルを生成
  - ウィザード形式でまずクラスもしくはインタフェイスを指定し、次に公開するメソッドを指定する。
- 3つのファイルが生成される
  - Exchange_Service.wsdl
  - Exchange_Service-interface.wsdl
  - DeploymentDescriptor.xml

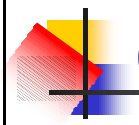
SARION™ SYSTEMS RESEARCH



(3)サーバクラス配置

- サーバクラスをサーブレットコンテナ内に配置し、サービスとしてアクセス可能に
  - wsdlgenが生成した、配置情報設定ファイルDeploymentDescriptor.xmlを利用

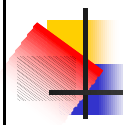
```
D:¥home¥WSDemo¥sample> java
    org.apache.soap.server.ServiceManagerClient
    http://localhost:8080/soap/servlet/rpcrouter deploy
    DeploymentDescriptor.xml
```

(5)proxyコード生成

- Exchange_Service.wsdlを入手し、WSTKのproxygenコマンドを実行

```
D:¥home¥WSDemo¥sample>proxygen Exchange_Service.wsdl
>> Importing http://localhost:8080/wsdl/Exchange_Service-interface.wsdl ..
>> Transforming WSDL to NASSL ..
>> Generating proxy ..
Created file D:¥home¥WSDemo¥sample¥Exchange_ServiceProxy.java
Compiled file D:¥home¥WSDemo¥sample¥Exchange_ServiceProxy.java
Done.

D:¥home¥WSDemo¥sample>
```



(6)クライアントへの組み込み

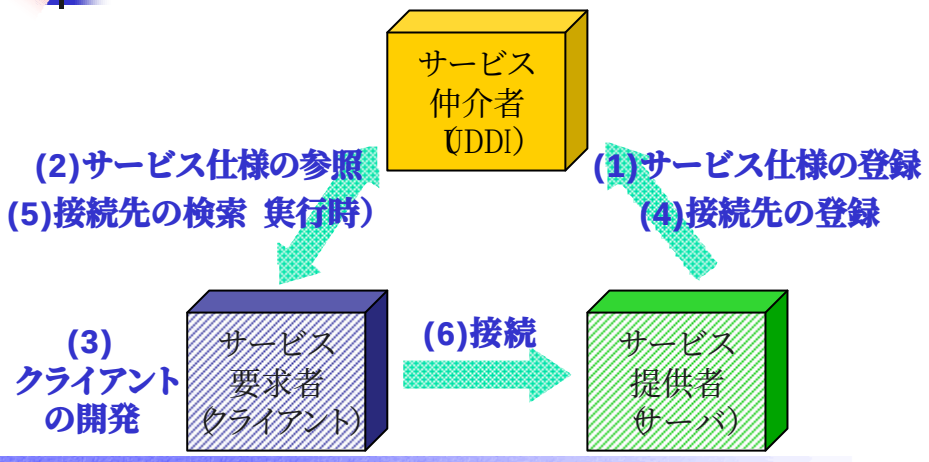
- proxyクラスをnewしてメソッドを呼ぶ

```
public class ClientProxy {
   public static void main( String[] argv ) throws Exception {
      Exchange_ServiceProxy proxy = new Exchange_ServiceProxy();
      float f = proxy.getRate( "USA", "Japan" );
      System.out.println( f );
   }
}
```


III. Webサービス開発手法

4. UDDIへの登録と検索

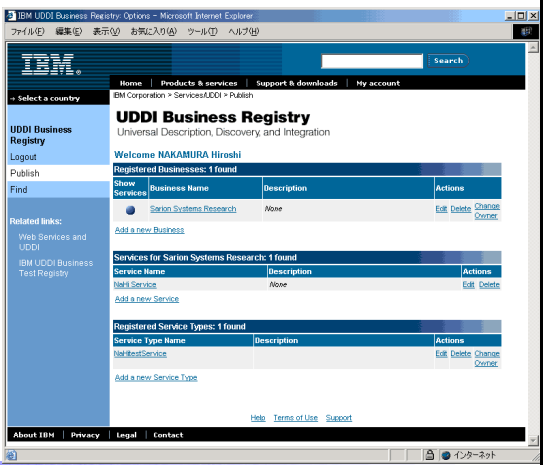
UDDI利用時の流れ



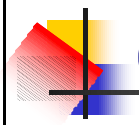
SARION™ SYSTEMS RESEARCH

(1)(2)(4)サービス仕様と接続先の登録・参照

- ブラウザから登録・参照
- WSTK付属のUDDI4Jを使い、SOAPインタフェイスを利用することも可能



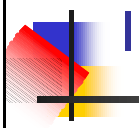
SARION™ SYSTEMS RESEARCH



(5)接続先の検索

- SOAPインタフェイスを利用し、tModelKey（サービス仕様のID）から接続先を取得する手順
  - 与えられたtModelKey（サービス仕様のID）から、find_business APIによりbusinessInfo（簡略版企業情報）を得る
  - 取得した企業IDから、find_service APIによりserviceInfo（簡略版サービス情報）を得る
  - 取得したサービスIDから、get_serviceDetail APIにより businessService（サービス情報）を得る
  - 取得した接続先IDから、get_bindingDetail APIにより bindingDetail（接続先情報）を得る
- UDDI4Jを利用するとSOAP呼び出しが若干楽になるが、基本的なナビゲーション手順は変わらない

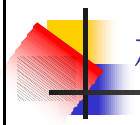
Webサービス構築のためのXML/SOAP関連仕様と開発手法



III. Webサービス開発手法

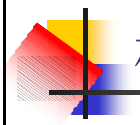
5. 各種ツールの相互接続性

Copyright 2001 © Sarion Systems Research



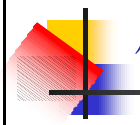
相互接続性

- SOAPBuilders Interoperability Lab:
  <URL:http://www.xmethods.net/ilab/>
  <URL:http://www.whitemesa.com/interop.htm>
  - 各種SOAP実装によるRPCサーバが、定められたメソッドを提供
  - 各実装によるRPCクライアントが接続し、随時結果を公開



相互接続性の現状

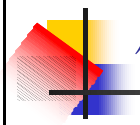
- SOAP/1.1に関しては、正常系での利用であれば問題のないレベル
  - 以下のケースでは問題が生じる実装もある
    - sparse array（疎な配列）
    - intにnullを渡した場合?
    - CR（¥r）が文字列に含まれている
- WSDLに関しての相互接続実験は始まったばかり



仕様(1)

[1] XML 1.0: http://www.fxis.co.jp/DMS/sgml/xml/rec-xml.html
[2] XML Schema: http://www.w3.org/TR/xmlschema-0/
[3] HTTP/1.1: http://www.ietf.org/rfc/rfc2616.txt
[4] SMTP: http://www.ietf.org/rfc/rfc0821.txt
[5] SOAP/1.1: http://www.jp.ibm.com/developerworks/link/soap.html
[6] SOAP/1.2: http://www.w3.org/TR/soap12-part1/ 英）
[7] WSDL/1.1: http://www.microsoft.com/japan/developer/workshop/xml/general/wsdl.asp
[8] UDDI: http://www.uddi.org/ 英）日本語版もあり）
[9] WSFL: http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf 英）

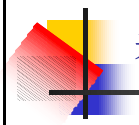
Webサービス構築のためのXML/SOAP関連仕様と開発手法



仕様(2)

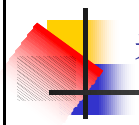
[10] XLANG: http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm 英）
[11] SOAP Security Extensions: Digital Signature: http://www.w3.org/TR/SOAP-dsig/ 英）
[12] RELAX: http://www.y-adagio.com/public/standards/tr_relax_c/toc.htm
[13] TREX: http://www.thaiopensource.com/trex/ 英）
[14] XML-Signature: http://www.w3.org/TR/xmldsig-core/ 英）
[15] SOAP-RP: http://www.gotdotnet.com/team/xml_wsspecs/soap-rp/default.html 英）
[16] XML Protocol (XMLP) Requirements: http://www.w3.org/TR/xmlp-reqs/ 英）
[17] XML Protocol Abstract Model: http://www.w3.org/TR/xmlp-am/ 英）

Copyright 2001 © Sarion Systems Research



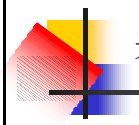
参考サイト(1)

[1] Microsoft .net: http://www.microsoft.com/japan/net/
[2] IBM e-businnes: http://www-6.ibm.com/jp/e-business/
[3] SUN ONE: http://www.sun.com/software/sunone/ 英）
[4] CORBA: http://www.corba.org/ 英）
[5] COM: http://www.microsoft.com/japan/com/
[6] EJB: http://java.sun.com/products/ejb/ 英）
[7] MQSeries: http://www-6.ibm.com/jp/software/mqseries/
[8] W3C: http://www.w3.org/
[9] SOAPセキュリティ拡張:電子署名: http://www-6.ibm.com/jp/developerworks/xml/010330/j_soap-dsig.html
[10] UDDI Overview: http://www.uddi.org/pubs/UDDI_Overview_Presentation.ppt 英）
[11] ebXML: http://www.ebxml.org
[12] W3C: XML Protocol Activity: http://www.w3.org/2000/xp/



参考サイト(2)

[13] Uche Ogbuji, "SOAP アプリケーションで WSDL を活用する方法": http://www-6.ibm.com/jp/developerworks/webservices/001201/j_ws-soap-index.html
[14] 丸山宏, "e-Businessを支える情報技術": http://www-6.ibm.com/jp/developerworks/webservices/010119/b2b.html
[15] Dave Fisco, "IBMのWebサービス・アーキテクチャーのデビュー": http://www-6.ibm.com/jp/developerworks/web/001013/j_w-int.html
[16] Rod Smith, "Web Servicesのチェックポイント": http://www-6.ibm.com/jp/developerworks/webservices/010202/j_ws-check.html
[17] 丸山宏, "ゾーンリーダーのコラム" (UDDI関連) : http://www-6.ibm.com/jp/developerworks/xml/000908/x000908.html
[18] 丸山宏, "XML、SOAP、UDDIによる B2B/eマーケットプレイスの標準化と将来展望": http://www-6.ibm.com/jp/developerworks/xml/000929/000929xml.pdf
[19] Doug Tidwell, "UDDI4J: Web Servicesの縁結び": http://www-6.ibm.com/jp/developerworks/webservices/010406/j_ws-uddi4j.html



参考サイト(3)

[20] Graham Glass, "Web Servicesの進化と革命 第4回": http://www-6.ibm.com/jp/developerworks/webservices/010413/j_ws-peer4.html
[21] SOAP-ml-JP: http://www.sarion.co.jp/ml/soap/
[22] Webサービス同好会: http://objectclub.esm.co.jp/webservice/home.html
[23] Apache SOAP実装: http://xml.apache.org/soap/ 英）
[24] Microsoft SOAP実装: http://msdn.microsoft.com/soap/default.asp 英）
[25] IBM WSDL実装: http://www.alphaworks.ibm.com/tech/wsdltoolkit 英）
[26] IBM Web Services実装（UDDI含む）: http://www.alphaworks.ibm.com/tech/webservicestoolkit 英）
[27] Ruby実装: http://www.jin.gr.jp/~nahi/Ruby/SOAP4R/RELEASE_ja.html
[28] その他実装: http://www.soaprpc.com/software/ 英）