# SOAP

高瀬 俊郎
日本アイ・ビー・エム（株）　E30809@jp.ibm.com

## XMLによる企業間取引

XML (eXtensible Markup Language) はプラットフォームに依存しないデータ構造として注目を浴びている．その理由としては，データ表現がテキストベースであり，自己記述的であることから分かりやすいこと，元々国際化対応がなされていること，データ構造が柔軟であり，変化するシステムやビジネス環境に対応することが容易であること，W3C (World Wide Web Consortium) [1] などの標準化団体によって標準化が行われており，その結果，多様なベンダの製品において相互運用性が確保されていること，などが挙げられる．

このようなプラットフォームに依存しない，相互運用可能というXMLの性質により，特にBusiness-to-Business (B2B)，つまりインターネット上での企業間のデータ交換が容易となり，これらの分野におけるXMLの重要性が増してきている．

もちろん，インターネット上でデータを流すのに，XMLでなければならないという理由はない．メールはたいてい普通のテキストであるし，Webページは HTMLというデータフォーマットで流れる．B2Bの世界では，EDI (Electronic Data Interchange) という規格があり，注文書や請求書などビジネス間のデータ交換を標準化してきた．これらの標準規格でなくても，スプレッドシートでよく使われているCSV (Comma Separated Value，コンマで区切られた値の列) や，単純な固定長のデータフォーマットが使われることもある．

しかし，インターネット上には多種多様なプラットフォーム上にこれまた多種多様な言語で実装されたシステムやアプリケーションが存在している．これらのアプリケーション間でのデータのやりとりを行う場合，事前にデータのやりとりを行う企業間で交換するデータ構造のすり合わせをし，システムを適応させる必要がある．

データフォーマットとしてXMLを用いることで，その相互運用性の高さによって，このような事前のシステムの適応のための負荷を大幅に軽減することができる．これによって現在XMLは企業間取引におけるデータフォーマットとして使われるようになってきている．

## XML用メッセージングプロトコルSOAP

XMLは単にデータフォーマットであるから，当然データの交換を行うプロトコルは定義していない．実際のデータが流れるトランスポートプロトコルとしては，インターネットで広く用いられるようになったHTTP (HyperText Transfer Protocol) が使われることが多いであろう．この理由としては，プロトコルがテキストベースであるため理解とデバッグが容易であること，HTTPサーバ上でアプリケーションを書くプログラミングスタイルが，CGI-BIN，Servlet，ASP (Active Server Pages) などによって一般的になってきたこと，HTTPに対してはSSLを用いたその暗号化版であるHTTPSが広くサポートされていること，企業間のデータ交換を行うためには，ファイアウォールを通過しなくてはならないが，HTTPはファイアウォールを通過できるように設定されることが多いため，既存のインフラを変更する必要がないこと，などが挙げられる．

HTTPは基本的にリクエスト・レスポンスの通信モデルを持つ．つまり，1つのリクエストをあるサーバに送るとそのリクエストに対するレスポンスが1つクライア

ントに返信される．返信は同期的に行われ，クライアントはリクエストを送信した後，レスポンスを待って次の処理に進むことになる．しかし，このようなHTTPの通信モデルでは実際の企業間取引においては不十分な場合も考えられる．

まず，メッセージの処理を同期的に行うことが難しい場合が考えられる．たとえば，あるメッセージを処理するときに完全に機械化されているわけではなく，途中に人間の担当者の承認が必要である場合などがこれにあたる．このような場合，メッセージは電子メール（SMTP）のような非同期なプロトコルによって送信され，サーバ側で処理が行われた後に返信が別のメッセージとして非同期的に返されることが考えられる．複数のHTTP接続を使ってこのような処理を実現することは可能であるが，どのメッセージがどのメッセージの返答であるか，などの管理を行う必要がある．

また，HTTPはブラウザでWebページを表示する程度であれば，通信が途中で切れたとしても大きな問題にはならないことが多い．しかし，企業間取引においては契約にかかわる場合なども想定されるため，Webページの表示よりも大きな信頼性が要求される．たとえば，HTTPではあるリクエストに対するレスポンスが返ってこなかった場合，そのメッセージがリクエストの送信中に途切れて通信先に届かなかったのか，通信先は正常にリクエストを受信し返信したが，その返信の途中で通信が切れたのか確かめる手段はない．このように企業間における取引においてはHTTPだけでは不十分な場合も多い．このような場合，必要に応じてMQ（Message Queuing）などのトランスポートプロトコルと組み合わせることが必要になる．

さらに，HTTPを用いる場合メッセージのルーティングに考慮する必要がある．ファイアウォールは大概HTTPは越えることができるように設定されるため，HTTPを用いることによって企業のイントラネット内から他の企業のサーバに接続することが容易になる．しかし，企業間取引におけるメッセージングを考える場合，最終的なアプリケーションにメッセージが到達するまでにいくつかの処理が行われることが考えられる．たとえば，メッセージの認証や署名の検証，暗号の復号化，メッセージのログへの記録などの一連の処理が最終的なアプリケーションに送られる前に行われるかもしれない．これらの処理は個々のアプリケーションに固有の処理ではないので最終的なアプリケーションが解釈すべきメッセージの本体にこれらのルーティングの情報を記述をするのは好ましくなく，どこか別の部分に記述されることが望まれる．このようなメッセージのルーティングの様子を図-1に示す．

このように，企業間のデータのやりとりを文書のやりとりのように考えると，内容に関係なく一定の共通の書式をくくり出すことができるだろう．メッセージの送信者／受信者，メッセージの送信時刻／受信時刻，メッセージのID，メッセージの認証情報などが考えられる．これらの情報は，メッセージに関するメタ情報であり，特定のアプリケーションに依存しない．よって，これらメタ情報はアプリケーション間でやりとりするメッセージの本体の中で記述するのではなく，メッセージングのプロトコルによってサポートされるべきである．

これらのことを考え合わせると，特定のトランスポートプロトコルに依存することなく，トランスポートの上のレベルでメッセージングの情報を記述できるようなモデルが必要とされる．このようなXMLの企業間取引におけるメッセージングの役割を果たすプロトコルとして現在SOAPというメッセージングプロトコルが注目を浴びてきている．一例として，XMLによるEDIの標準化を進めているebXMLでは，TRP（Transport, Routing, Protocol）作業部会でこのようなメッセージングの仕様を決めようとしているが，一部にSOAPを採用することを決めている．

## SOAP（Simple Object Access Protocol）1.1

SOAPは元々，MicrosoftによってXMLを用いたプラットフォーム独立で軽量なRPC（Remote Procedure Call，遠隔手続き呼び出し）のプロトコルとして設計さ

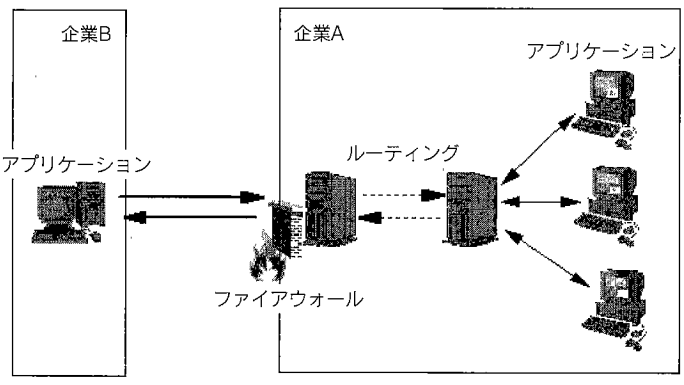


図-1 メッセージのルーティング

れたが，HTTPに強く依存していたため，SOAP1.1[2]ではトランスポート独立なプロトコルとして，2000年5月にMicrosoft，Lotus，IBMが中心となってW3CにNoteとして提案された．Noteとは，Working DraftのようなW3C内のワーキンググループによって議論されたものではないが，W3Cのメンバによって提出されたものがW3Cによって公開された仕様である．

SOAPのメッセージ・エンベロープはきわめて単純であり，ヘッダとボディという2つの構成要素しか定義していない．図-2にSOAPメッセージの構造を示す．

それではSOAP1.1の仕様の概略をみてみよう．SOAP1.1の仕様では主にSOAPエンベロープ，SOAP符号化，SOAPのHTTPによる使い方，SOAPのRPC表現などについて定められている．順に概略を解説する．

## ■SOAPエンベロープ

まずSOAPエンベロープ（Envelope）であるが，その名の通りXMLを包む封筒のようなものである．これがSOAPメッセージの一番外側の要素になる．名前空間として"http://schemas.xmlsoap.org/soap/envelope/"が使われる．

SOAPエンベロープにはSOAPヘッダとSOAPボディが1つずつ子要素として入れられる．SOAPヘッダはなくても構わないがSOAPボディは必ず1つなくてはならない．

SOAPヘッダ（Header）には認証やトランザクション管理などの付加情報を記述することができる．SOAPヘッダの子要素としていくつかのヘッダ項目を記述することができるが，これらのヘッダ項目に対してはactor属性やmustUnderstand属性を付加することができる．それぞれのヘッダ項目はすべてが最終的なあて先が処理するとは限らず，ゲートウェイサーバなどのメッセージパス中の仲介者（intermediary）によって処理されることも想定されている．そういった場合，actor属性にそのヘッダ項目を処理すべきサーバのURIを指定する．また，アプリケーションによっては自分が理解できない項目を無視するという処理を行うものがあるかもしれない．しかし，mustUnderstand="1"という属性を持つヘッダ項目が存在する場合，あて先のアプリケーションはそのヘッダ項目を無視してはならない．もしあて先のアプリケーションがそのヘッダ項目を理解できない場合は，MustUnderstandというSOAP Faultを返すことになっている．

SOAPボディ（Body）はメッセージの本体である．この中には任意のXML文書を入れることができる．またSOAPメッセージの処理の過程でエラーが起きた場合には，このBody要素にFaultという要素を入れてレスポンスを返すことになる．

SOAP FaultはSOAPメッセージの処理の過程でエラーが起きたとき，そのエラーを記述するためのものである．SOAP Faultにはfaultcode，faultstring，faultactor，detailなどの内容が含まれる．faultcodeはエラーの原因を表し，VersionMismatch，MustUnderstand，Client，Serverのいずれかの値を指定する．それぞれ，VersionMismatchはエンベロープの名前空間の不一致，MustUnderstandはmustUnderstand="1"のついたヘッダ要素が理解できない，Clientは送られてきたメッセージが間違っている，Serverがサーバ側のエラー（同じメッセージが後の時点で成功する可能性あり）ということを意味している．その他，faultstringには人が読めるようなエラーの説明，faultactorにはエラーの発生もとのURI（最終宛先でない場合は必須）detailにはエラーの詳細（たとえばスタックトレースなど）を記述する．

SOAPメッセージの各要素にはencodingStyle属性をつけることができる．encodingStyle属性には直列化の方式を指定する．直列化の方式とは，たとえばJavaのint変数に入っている値をXMLで表現するときの変換規則のことである．SOAP1.1で規定されている符号化規則（SOAP Encoding）を用いる場合はencodingStyle属性は"http://schemas.xmlsoap.org/soap/encoding/"の値を用いる．encodingStyle属性の有効範囲はXML名前空間宣言と同じで，その要素とその要素のすべての下位要素である．

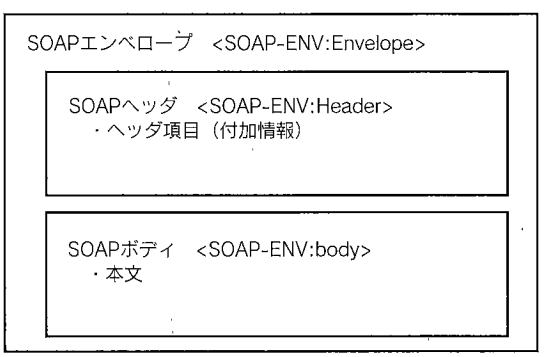


図-2　SOAPメッセージの構造

### ■SOAP符号化（SOAP Encoding）

SOAP符号化ではRPCでやりとりされるデータ，たとえば，あるプログラミング言語のさまざまなデータ型をどのようにXMLで記述するかを定めている．基本的にはXML Schema Part2: Datatypesにそのまま従うものである．しかしSOAPでは必ずこの符号化規則を使わなくてはならないということではなく，他の符号化を用いてもよい．このSOAP1.1で規定されている符号化を用いるときには前述のようにencodingStyle属性に"http://schemas.xmlsoap.org/soap/encoding/"という値を与えておく．SOAP符号化ではXML Schema Part2: DatatypesのBuilt-in datatypesの節で規定されるすべての型，多態的アクセサ（実行時に型が決定される），構造体，配列などの型について規定されている．

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

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Header>
    <t:Transaction
      xmlns:t="some-URI"
      SOAP-ENV:mustUnderstand="1">
        0005721
    </t:Transaction>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <symbol>DEF</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

図-3　リクエストメッセージの例

### ■SOAPのHTTPによる使い方

SOAPはトランスポートに対して独立した仕様であるので，トランスポートには何を使用しても構わない．しかし，特にHTTPを用いてSOAPメッセージのやりとりをするときの規定がいくつか定められている．たとえば，HTTPをトランスポートとして使う場合，HTTPSのように暗号化して使うことが一般的に考えられる．しかし，ゲートウェイサーバやプロキシサーバにまったく情報が読めないと困るので，メッセージをどこにルーティングするかを決定するためのヒントをSOAPActionというHTTPヘッダに書き込むことができることになっている．また，リクエストを処理している間にSOAPエラーが発生した場合，SOAP HTTPサーバはHTTPコード500 "Internal Server Error"を発行し，そのレスポンスはSOAP Fault要素を持つSOAPメッセージを含まなければならないと決められている．

### ■SOAPのRPC表現

SOAPはRPCのみに使われるというわけではないが，SOAPメッセージを用いてRPCを行う場合の規定がいくつか仕様に定められている．メソッドコールとレスポンスはSOAPボディによって運ばれること，メソッドコールは引数をメンバとする構造体にし，メソッド名と同じ名前をつけること，メソッドレスポンスは戻り値をメンバとする構造体，名前は慣例として「メソッド名＋Response」とすること，パラメータのアクセサはパラメータ名と同じ名前，パラメータの型と同じ型を持ち，メソッドの形式と同じ順序で出現すること，戻り値アクセサの名前は何でもよいが，型は戻り値の型と同じとすること，などが定められている．

以上がSOAP1.1の仕様の概略である．詳細は仕様そのものを参照してもらうとして，次の章では実際のSOAPのメッセージをみていくことにする．

## SOAPメッセージの例

実際のSOAPのメッセージの例を図-3に示す．この例はHTTPをトランスポートとした場合のリクエストメッセージである．

XMLは自己記述形式であるので何のリクエストを出しているかはSOAPをあまり知らない人でも見て大体わかるだろう．この例では"DEF"というシンボルで表される企業の株価の終値を要求している．

簡単に解説すると，まずいくつかのHTTPヘッダがある．Content-Typeは"text/xml"である．SOAPActionは，SOAP HTTPリクエストの意図を指示するために使用されるのであった．HTTPにおけるSOAPリクエストメッセージを適切にフィルタリングするゲートウェイなどのサーバによって使われるわけだが，この例のように値が空文字（""）である場合は，SOAPメッセージの意図がHTTP Request-URIによって指示されることを意味している．

メッセージ本体は<Envelope>の中に<Header>と

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

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetLastTradePriceResponse xmlns:m="Some-URI">
      <Price>34.5</Price>
    </m:GetLastTradePriceResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

図-4　レスポンスメッセージの例

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<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>
        <e:myfaultdetails xmlns:e="Some-URI">
          <message>
            My application didn't work
          </message>
          <errorcode>
            1001
          </errorcode>
        </e:myfaultdetails>
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

図-5　エラーメッセージの例

<Body>が1つずつあることが分かる．SOAP-ENVというのはSAOPエンベロープの名前空間である．ヘッダ項目の<Transaction>にはmustUnderstand="1"という属性がついているのでこの項目をあて先のアプリケーションは無視してはいけないことになる．もし理解できない場合はSOAP Faultを返さなくてはならない．また，<Envelope>の属性にencodingStyle="http://schemas.xmlsoap.org/soap/encoding/"とあるので，このメッセージ全体がSOAP1.1で定められたSOAP符号化を用いて直列化されていることが分かる．

図-4にこのリクエストに対するレスポンスメッセージを示す．SOAPメッセージのやりとりが成功して，終値が返ってきている．このメッセージには<Header>がないが，ヘッダは必要がなければなくてもかまわない．

図-5にリクエストメッセージの処理に失敗したときのレスポンスメッセージを示す．<Body>の中に<Fault>が入っている．HTTPのコードは500である．<faultcode>がServerとなっているのでこのリクエストメッセージは後で実行すると成功するかもしれない．<detail>にはエラーの詳細が記述される．これはサーバ側のアプリケーションによって作られたものである．

いかがであろうか．SOAPのメッセージのやりとりの雰囲気をつかんでいただけたであろうか．SOAPはXMLで記述されているためにメッセージそのものを見るだけでも何をしようとしているのかが分かり，デバッグも容易になる．またXMLで記述することで多様なプラットフォーム間を相互に繋ぎ合わせることも可能である．

## Apache-SOAPを使ってみる

SOAPの実装に関しては多くのものがリリースされているがここではオープンソースのJavaによる実装であるApache-SOAP[3]を使ってみる．Apache-SOAPは元々IBMがSOAP4J（SOAP for Java）として公開していたものが，HTTPサーバで有名なApache Software Foundationに寄贈され，オープンソースソフトウェアとして開発が引き続き進められている．

Apache-SOAPはSOAPをJavaで利用するためのライブラリである．ApacheのXML Apache Projectの中で公開されている．

Apache-SOAPにはサーバ用とクライアント用の両方のAPIを含む．その他にも，実際にやりとりされているSOAPメッセージをTCPのレベルで観察することができるツールや，ブラウザからSOAPサーバにサービスを登録することができるJSPによる管理ツールなどを含んでいる．

SOAPはトランスポートに依存しないプロトコルであるのでHTTP以外のトランスポートを利用することが可能である．実際Apache-SOAPでもHTTPの他にSMTPもサポートしているが，ここではやはり一番一般的であるであろうHTTPの例を示す．

Apache-SOAPではサーバ側のRPCサービスとして一般のJavaのクラスを登録できるようになっている．サ

ーバにサービスを登録することを配置（deploy）という．例として，図-6のようなJavaのクラスを用いる．

このクラスを呼び出すクライアントのコードの一部分を図-7に示す．Callオブジェクトを生成し，次の値をセットしている．

• TargetObjedtURI: 呼び出すサービスのIDであるURNの値．

• EncodingStyleURI: "http://schemas.xmlsoap.org/soap/encoding/"はSOAP1.1で定義されている符号化規則を使うことを表す．

• MethodName: 呼び出すメソッド名．

• Params: 引数のリスト．セットするVectorには引数の数だけParameterオブジェクトを追加する．

これらをセットした後，invokeメソッドを呼ぶことでSOAPのサービスを呼ぶことができる．invokeメソッドの引数はSOAPサーバのURLとSOAPActionである．SOAPActionはHTTPヘッダに追加される．これによってHTTPサーバがフィルタリングなどを行ってもよい．レスポンスはResponseオブジェクトに格納される．

ここではRPCの例を示したが，SOAPを単にXMLを運ぶメッセージングの機構として使いたい場合，Apache-SOAPではMessageというクラスによってその機能が提供されており，XMLのDOM（Document Object Model）オブジェクトをSOAPボディにセットすることができる．

## SOAPの現状と今後

### ■SOAP関連規格

SOAPに関連した規格としていくつか提案されてきている．ここではともにW3CのNoteとしてすでに提案されているSOAP Messages with AttachmentsとSOAP Security Extensions: Digital Signatureについて簡単に紹介する．

SOAP Messages with Attachments[4]は画像などの非XMLデータをメッセージに添付してmultipart MIMEで送る場合の表現方法である．SOAPはメッセージングのフォーマットであるので，今後企業間の取引などに使われていくことを考えるとXMLだけを運ぶというわけにはいかない．画像やバイナリのデータをやりとりする必要も出てくるであろう．そういった場合にそれらのバイナリデータをどのように添付するかの仕様である．メールにファイルを添付する場合に使われるMIMEエンコードをそのまま用いている．

SOAP Security Extensions: Digital Signature[5]はSOAPエンベロープのヘッダ内に電子署名の情報を記述するためのデータフォーマットを規定している．電子署名は第三者による「改竄」を防いだり，作成者の「否認不可性」を証明するものである．これは企業間取引においては非常に重要になる．「否認不可性」については，たとえば企業Aが企業Bに何らかの注文書を作成し送信したとする．企業Bがその注文書を処理し請求書を発行する際に，企業Aに注文書を作成した事実を否認されると困る．SOAPメッセージに電子署名を付加しておくことでこのような「しらばくれ」を防ぐことができる．

### ■XML Protocol

XML Protocol[6]は2000年9月からW3Cによるワーキンググループが活動を開始している．2001年7月9日にSOAP Version 1.2（http://www.w3.org/TR/soap12/）および，XML Protocol Abstract Model（http://www.w3.org/TR/xmlp-am/）がWorking Draftとして公開された．今後，最終的な仕様であるRecommendationへ向けての作業が行われていくことになる．

```
public class StockQuote{
  public float getQuote(String symbol){
    System.out.println( "getQuote( " + symbol + " )" );
    return 107.82F; // always return the same value
  }
}
```

図-6　StockQuote.java

```
import org.apache.soap.rpc.*; //Call,Parameter,Response

public class StockQuoteClient{
  public static void main(String[] args) {
    .....
    Call call = new Call();
    call.setTargetObjectURI ("urn:demo:stock");
    call.setEncodingStyleURI("http://schemas.xmlsoap.org/soap/encoding/");
    call.setMethodName("getQuote");
    Vector params = new Vector();
    params.addElement(new Parameter("symbol", String.class, "IBM", null));
    call.setParams(params);
    Response resp = call.invoke(new URL("http://localhost/soap/servlet/rpcrouter"), "");
    .....
  }
}
```

図-7　StockQuoteClient.java

これらはXMLでのメッセージのやりとりを行う上での形式を与えようとするもので，SOAP1.1をもっと一般化して，広い視点から再構築するものである．SOAP1.1はXMLをやりとりするうえで非常にシンプルなフォーマットを与えているが，たとえば，actorとmustUnderstand属性の解釈，actionURIの意味などまだ曖昧な部分は多い．これらの仕様ではSOAP1.1に欠けている部分，不十分な部分をきちんと定義しようとしている．Recommendationが出てくるのはまだ先のことになるかもしれないが，W3Cにおいては，SOAP Version 1.2が認められた仕様として標準化が進められるであろう．

## ■SOAPを用いたサービスの例

SOAPを用いることによってその相互運用性により，異なるプラットフォーム間でのアプリケーションの連携が可能となる．たとえば，Linux上のJavaによるアプリケーションとWindows上のVisualBasicによるアプリケーションを結合することは容易ではない．しかしSOAPを用いたメッセージを受け付けるようにすることで，どのようなプラットフォームからでもアクセスが可能であるようなアプリケーションを構築することが可能となる．

現時点では一般に公開されたSOAPを用いたアプリケーションはまだまだ多いとはいえない．しかし，いくつか現時点ですでに利用可能なサービスがインターネット上に存在する．

XMethodsのサイト（http://www.xmethods.net/）には数十のSOAPを用いたサービスがリストされている．それぞれのサービスは株価や為替，天気などを提供するものや，単位を変換するなど単純なRPCのサービスが多いが，注目すべきはサービスを提供しているサーバの実装が多種にわたっていることである．これらのサービスに繋いでみれば使っているクライアントの実装との相互運用性がすぐに確かめられる．

UDDIレジストリはWebサービスを発見するためのディレクトリの機能を提供するものであるが，UDDIのサービスそのものもSOAPのサービスとして提供されている．UDDIレジストリにUDDIの仕様で定められた形式のXMLをSOAPで包んで送ることで，サービスを検索，登録することができる．現在すでにIBMとMicrosoftのサイトでサービスが提供されている．

XKMS（XML Key Management Specification）[7]はPKI（Public Key Infrastructure）の証明書の検証や鍵管理の機能を提供するサービスの仕様である．2001年3月にはW3CにNoteとして提出された．この仕様はサービスを提供する手段としてSOAPを前提としている．まだ実装をリリースしているベンダは少ないが，この仕様には多くのベンダが賛同を表明しているため，近い将来，実際にサービスが提供されるようになると思われる．

## ■SOAPの今後

SOAPを用いる一番のメリットはやはりその相互運用性である．異なるプラットフォーム間でメッセージのやりとりを容易に行うことができるようになれば企業間の取引において大きなインパクトを与えるであろう．とはいえ，インターネット上で企業間にまたがるSOAPによるメッセージの交換を行うには解決されなくてはならない課題がいくつかあることも事実である．まず，HTTPでメッセージングの途中で通信が切れてしまうような信頼性の問題やそういった場合の回復方法などの問題がある．また，インターネット上でメッセージのやりとりをする場合は常にセキュリティが問題になってくるであろう．電子署名の技術などもすでに実用化されているが，普及にはもう少しかかりそうである．しかし，SOAPは企業内のイントラネットや特定の企業間でのアプリケーションを連携する仕組みとして利用することも可能である．企業内ならばサービスの信頼性やセキュリティに関する問題も生じにくく，現段階の技術でも十分適用可能であると考えられる．いずれにせよ，今後のインターネットによる企業間取引を考えた場合，プラットフォームの異なる既存のアプリケーションを変更することなく，お互いに結びつけることを可能にするSOAPのような技術は必要不可欠なものとなってくるであろう．

参考文献
1) The World Wide Web Consortium (W3C), http://www.w3.org/
2) Simple Object Access Protocol (SOAP) 1.1, http://www.w3.org/TR/SOAP/, 日本語訳, http://www-6.ibm.com/jp/developerworks/link/soap.html
3) Apache-SOAP, http://xml.apache.org/soap/
4) SOAP Messages with Attachments, http://www.w3.org/TR/SOAP-attachments
5) SOAP Security Extensions: Digital Signature, http://www.w3.org/TR/SOAP-dsig/, 日本語訳, http://www-6.ibm.com/jp/developerworks/link/soap-dsig.html
6) XML Protocol, http://www.w3.org/2000/xp/
7) XML Key Management Specification (XKMS), http://www.w3.org/TR/xkms/

（平成13年6月25日受付）