
# Web サービスおよび Web クライアントの作成

Version 2023.1
2024-01-02

InterSystems Corporation　1 Memorial Drive　Cambridge MA 02142　www.intersystems.com

Web サービスおよび Web クライアントの作成
InterSystems IRIS Data Platform　Version 2023.1　2024-01-02


InterSystems Worldwide Response Center (WRC)
Tel:　+1-617-621-0700
Tel:　+44 (0) 844 854 2917
Email:　support@InterSystems.com

# 目次

# テーブル一覧

# 1

# InterSystems IRIS の Web サービスおよび Web クライアントの概要

InterSystems IRIS は、SOAP (Simple Object Access Protocol) 1.1 および 1.2 をサポートしています。このサポートは、使いやすく効率的で、SOAP 仕様と完全な互換性があります。このサポートは InterSystems IRIS に組み込まれており、InterSystems IRIS がサポートするすべてのプラットフォームで利用できます。

## 1.1 InterSystems IRIS Web サービスの概要

このセクションでは、InterSystems IRIS Web サービスの概要について説明します。

### 1.1.1 InterSystems IRIS Web サービスの作成

InterSystems IRIS では、次のいずれかの方法で Web サービスを作成できます。

- 既存のクラスに小さな変更をいくつか加えて、Web サービスに変換します。この場合、引数として使用されているオブジェクト・クラスを変更して、**%XML.Adaptor** を拡張し、SOAP メッセージにパッケージ化できるようにする必要もあります。
- 新規の Web サービス・クラスを最初から作成します。
- InterSystems IRIS SOAP ウィザードを使用して、既存の WSDL ドキュメントを読み取り、Web サービス・クラスおよびサポートするすべてのタイプ・クラスを生成します。

  この方法 (WSDL-first の開発) は、WSDL が設計済みで、それに応じた Web サービスを作成する必要がある場合に適用されます。

### 1.1.2 Web アプリケーションの一部としての InterSystems IRIS Web サービス

InterSystems IRIS Web サービスは、Web アプリケーション内で実行する必要があります。このアプリケーションの構成は、管理ポータルで行います。具体的には、Web サービス・クラスを使用する前に、そのクラスを含むネームスペースを使用する Web アプリケーションを定義する必要があります。

この Web アプリケーションの定義と構成の詳細は、"承認ガイド" の "アプリケーション" を参照してください。

## 1.1.3 WSDL

クラス・コンパイラは、Web サービスをコンパイルする際、そのサービス用の WSDL を生成し、利便性を向上させるために、その WSDL を Web サーバ経由で公開します。この WSDL は、WS-I (Web Services Interoperability Organization) によって確立された Basic Profile 1.0 に準拠しています。InterSystems IRIS では、WSDL ドキュメントが特定の URL で動的に処理され、(実行時に追加されるヘッダ要素以外にも) ユーザの Web サービス・クラスのインタフェースに対する変更のすべてが自動的に反映されます。ほとんどの場合は、このドキュメントを使用して、Web サービスと相互運用する Web クライアントを生成できます。

詳細と重要な注意事項は "WSDL の表示" を参照してください。

## 1.1.4 Web サービスのアーキテクチャ

InterSystems IRIS Web サービスの既定の機能を理解するには、Web サービスが認識可能なメッセージである、SOAP メッセージが含まれる HTTP 要求を受け取ったときに、Web サービスで発生するイベントについて理解することが有用です。

まず、特定の URL にリダイレクトされる HTTP 要求のコンテンツを考えてみます。

- HTTP のバージョンや文字セットなどの情報を示す HTTP ヘッダ。

  HTTP ヘッダには SOAP アクションが含まれている必要があります。これは、SOAP HTTP 要求の目的を示す URI です。

  SOAP 1.1 の場合、SOAP アクションは、`SOAPAction` HTTP ヘッダとして含まれます。SOAP 1.2 の場合は、`Content-Type` HTTP ヘッダ内に含まれます。

  SOAP アクションは、通常、着信 SOAP メッセージを転送するのに使用されます。例えば、ファイアウォールは、HTTP での SOAP 要求メッセージを適切にフィルタするためにこのヘッダを使用できます。SOAP はこの URI の形式や具体性、および解決可能性について、何も制限していません。

- GET、POST、HEAD などの HTTP メソッドを含む要求行。この行は、実行されるアクションを示します。
- メッセージ本文。この場合、メソッド呼び出しを含む SOAP メッセージです。より具体的に言うと、この SOAP メッセージは、呼び出すメソッドの名前とその引数に使用される値を示します。このメッセージには、SOAP ヘッダを含めることもできます。

ここで、この要求が送信された場合に実行される処理を確認しましょう。

1. サードパーティの Web サーバがこの要求を受け取ります。
2. この要求は、末尾が **.cls** である URL 宛てになっているので、Web サーバによって Web ゲートウェイに転送されます。
3. Web ゲートウェイで、この URL が検証されます。Web ゲートウェイでは、この URL の一部が Web アプリケーションの論理名として解釈されます。Web ゲートウェイは、Web アプリケーション内の該当する物理位置 (Web サービスのページ) にこの要求を転送します。
4. Web サービス・ページはこの要求を受け取ると、その OnPage() メソッドを呼び出します。
5. Web サービスは、要求に InterSystems IRIS SOAP セッション・ヘッダが含まれているかどうかを確認し、含まれている場合には、該当する SOAP セッションを再開するか、新規のセッションを開始します。

   注釈　この手順では、SOAP セッションとは、InterSystems IRIS SOAP サポートでサポートされるセッションのことです。SOAP 仕様がセッションの標準を定義するわけではありません。ただし、ここで説明するように、InterSystems IRIS SOAP サポートでは、Web クライアントと Web サービス間のセッションの管理に使用できる専用の InterSystems IRIS SOAP セッション・ヘッダが提供されます。

6. Web サービスは、メッセージをアンパックし、それを検証して、すべての入力パラメータを適合する InterSystems IRIS 表記に変換します。変換では、複雑なタイプごとに、その複雑なタイプを表現するオブジェクト・インスタンスが作成され、そのオブジェクトが Web メソッドの入力として使用されます。

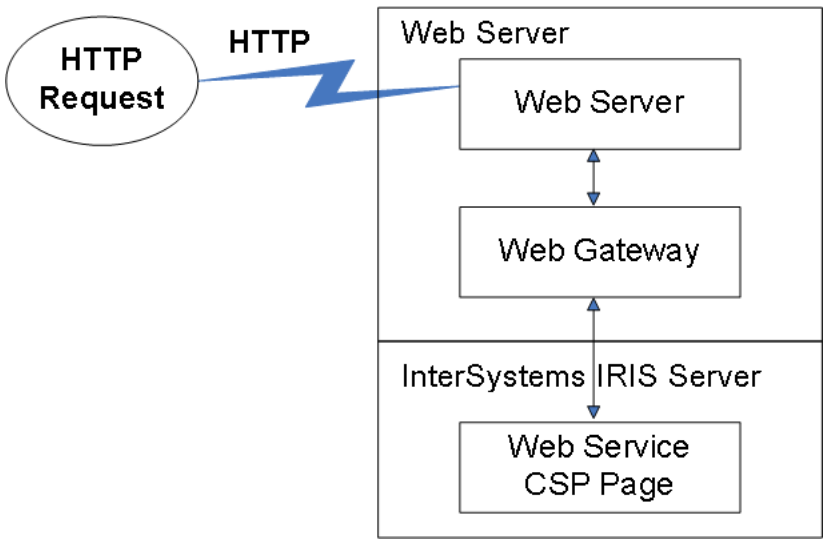
   ここで HTTP ヘッダの SOAP アクションを使用して、メソッドと要求オブジェクトが指定されます。

   Web サービスは、メッセージをアンパックするときに、新しい要求オブジェクトを作成し、そのオブジェクトに SOAP メッセージをインポートします。このプロセスでは、Web サービスは、この Web サービスがコンパイルされたときに作成されたクラス (Web メソッド・ハンドラ・クラス) を使用します。

7. Web サービスは、要求された InterSystems IRIS メソッドを実行し、応答をパッケージ化して、SOAP 応答を作成します。このときに必要に応じて SOAP ヘッダも組み込みます。

8. Web サービスは、SOAP 応答 (XML ドキュメント) を現在の出力デバイスに書き込みます。

次の図は、このフローの外部部分を示しています。



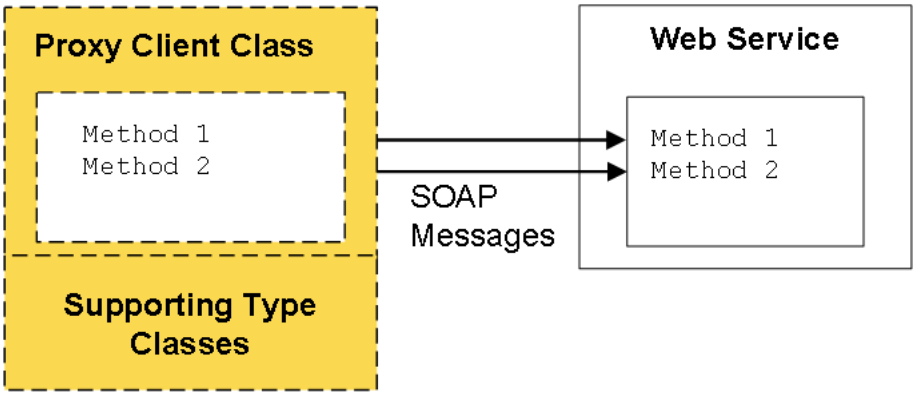

# 1.2 InterSystems IRIS Web クライアントの概要

このセクションでは、InterSystems IRIS Web クライアントの概要について説明します。

## 1.2.1 InterSystems IRIS Web クライアントの作成

InterSystems IRIS では、InterSystems IRIS SOAP ウィザードを使用して既存の WSDL ドキュメントを読み取ることで、Web クライアントを作成します。このウィザードでは、Web クライアント・クラスおよびサポートするすべてのタイプ・クラスが生成されます。

生成された Web クライアント・インタフェースには、Web サービスで定義された各メソッドのプロキシ・メソッドを含むクライアント・クラスがあります。各プロキシでは、対応する Web サービス・メソッドで使用されるものと同じシグニチャが使用されます。インタフェースには、メソッドの入出力として必要な XML タイプを定義するためのクラスも含まれています。



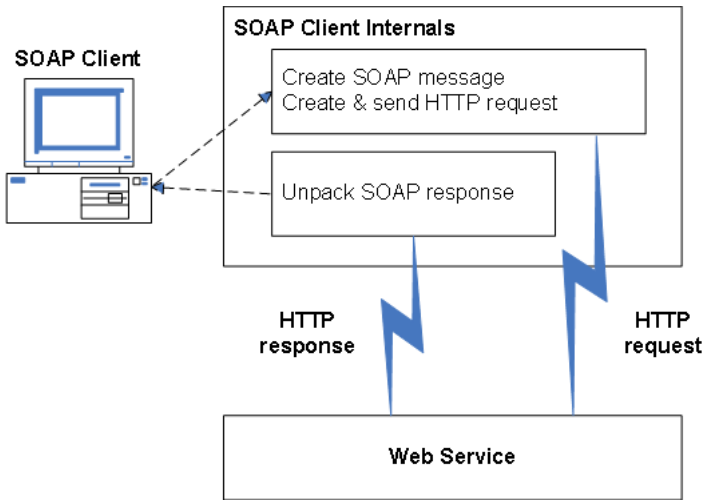

通常は、生成されたクラスをカスタマイズすることはありません。その代わり、Web クライアントの動作を制御し、そのプロキシ・メソッドを呼び出す追加クラスを作成します。

### 1.2.2 Web クライアントのアーキテクチャ

InterSystems IRIS Web クライアントの機能を理解するため、ここでは、ユーザまたは他のエージェントが Web クライアント内のメソッドを呼び出したときに発生するイベントについて考えてみます。

1. まず、Web クライアントで、メソッド呼び出しとその引数値を表す SOAP メッセージが作成されます。
2. 次に、SOAP メッセージを含む HTTP 要求が作成されます。前述のとおり、HTTP 要求には要求行と HTTP ヘッダが含まれます。
3. HTTP 要求が発行され、それが適切な URL に送信されます。
4. Web クライアントは、HTTP 応答を待機し、状態を判断します。
5. Web サービスから SOAP 応答を受信します。
6. SOAP 応答をアンパックします。

以下の図は、このフローを示しています。




## 1.3 その他の機能

InterSystems IRIS Web サービスおよび Web クライアントには、次の機能を追加できます。

- セッションのサポート。前述したように、SOAP 仕様がセッションの標準を定義するわけではありませんが、InterSystems IRIS では、クライアント・サーバ SOAP セッションを作成できます。
- カスタムの SOAP ヘッダ (WS-Addressing ヘッダを含む)、カスタムの SOAP メッセージ本文、およびカスタムの SOAP フォルト。
- MIME 添付。
- MTOM (Message Transmission Optimization Mechanism) の使用。
- Web クライアントと Web サービス間の認証 (ユーザ・ログイン)、および WS-Security 規格における主要部分。
- ポリシー。これにより、サービスまたはクライアントが次の操作をどのように行うかを制御できます。
  - 使用または要求する WS-Security ヘッダ要素の指定。
  - MTOM の使用の指定。
  - WS-Addressing の使用の指定。

  "Web サービスの保護" を参照してください。
- 形式に関するほとんどの要件を満たすように、生成される WSDL ドキュメントを調整するオプション。
- Web クライアントと Web サービス間での HTTP 以外による転送。

サポートされる規格の詳細は、次のセクションを参照してください。

# 1.4 SOAP 標準

このセクションでは、InterSystems IRIS Web サービスおよび Web クライアントの基本規格と WSDL のサポートの詳細を紹介します。

"XML 標準" および "SOAP セキュリティ標準" も参照してください。

## 1.4.1 基本規格

InterSystems IRIS Web サービスおよび Web クライアントは、次の基本規格をサポートします。

- SOAP 1.1 (https://www.w3.org/TR/2000/NOTE-SOAP-20000508/ を参照)、エンコード形式が含まれます。
- SOAP 1.2、セクション 3 SOAP バージョン 1.2 パート 2: Adjuncts で指定されたエンコード形式が含まれます (https://www.w3.org/TR/soap12-part2/)。
- MTOM (Message Transmission Optimization Mechanism) 1.0 (https://www.w3.org/TR/soap12-mtom/)。
- WSDL 1.1。InterSystems IRIS Web サービスは、WS-I (Web Services Interoperability Organization) によって確立された Basic Profile 1.0 に準拠する WSDL ドキュメントを作成します。ただし、InterSystems IRIS Web クライアントは、より汎用的な WSDL ドキュメントに対応できます。

  "InterSystems IRIS における WSDL のサポート" を参照してください。
- クライアント・アクセス専用 (リポジトリ提供なし) の UDDI バージョン 1.0。http://uddi.xml.org/ を参照してください。
- SOAP with Attachments の仕様に基づいてマルチパートまたは関連 MIME メッセージとして扱われる添付 (https://www.w3.org/TR/SOAP-attachments)。

  SOAP with Attachments は SOAP 1.2 および SOAP 1.1 でサポートされています。
- HTTP 1.1 または HTTP 1.0 による転送。

- Web クライアントからの出力は、UTF-8 でのみサポートされています。

## 1.4.2 InterSystems IRIS における WSDL のサポート

InterSystems IRIS では、可能なすべての WSDL ドキュメントをサポートしているわけではありません。変更できない特定の WSDL に対応する Web クライアントの作成が必要なことが多いため、クライアント側により高い柔軟性が与えられます。ここでは、サポートの詳細について説明します。

### 1.4.2.1 生成された WSDL ドキュメント

InterSystems IRIS Web サービスによって生成される WSDL ドキュメントにはヘッダが含まれません。また、InterSystems IRIS で作成できる Web サービスがすべての可能なバリエーションを反映するわけではありません。

SOAP 仕様では、WSDL を生成する Web サービスは一切必要とされていないことに注意してください。

### 1.4.2.2 WSDL の利用

InterSystems IRIS SOAP ウィザードは、考えられるすべての WSDL ドキュメントを処理できるわけではありません。特に次の制限事項があります。

- Caché SOAP クライアント・ウィザードでは、`<fault>` 要素はサポートされません。つまり、バインディングの `<operation>` 要素内に `<fault>` 要素が含まれている場合、その `<fault>` 要素は無視されます。
- 応答メッセージでは、以下の条件のうち 1 つを満たしている必要があります。
  - 各応答メッセージは、対応する要求メッセージと同じネームスペースに置く必要があります。
  - 応答メッセージはすべて、互いに同一のネームスペースに置く必要があります（要求メッセージで使用するネームスペースとは異なる場合があります）。
- InterSystems IRIS SOAP ウィザードでは WSDL のヘッダは処理されません。

SOAP ウィザードでは、WSDL における MIME バインディングを使用できます (https://www.w3.org/TR/wsdl#_Toc492291084)。MIME パーツは無視され、WSDL の残りの部分が処理されます。MIME バインディングを含む WSDL を基盤とする Web サービスまたは Web クライアントを作成する場合は、MIME 添付をサポートする ObjectScript コードを明示的に追加する必要がありますが、この作業はこのドキュメントの対象ではありません。

# 1.5 SAX パーサについての重要な点

InterSystems IRIS SAX パーサは、InterSystems IRIS が SOAP メッセージを受け取るたびに使用されます。その既定の動作を知っておくと役に立ちます。パーサは、以下のようなタスクを行います。

- XML ドキュメントが適格な文書であるかどうかを検証します。
- 指定されたスキーマまたは DTD を使用して、ドキュメントを検証しようとします。

  ここでは、1 つのスキーマには、他のスキーマを参照する `<import>` 要素および `<include>` 要素を含めることができる、ということを覚えておくと役に立ちます。以下はその例です。

```
<xsd:import namespace="target-namespace-of-the-importing-schema"
                    schemaLocation="uri-of-the-schema"/>

<xsd:include schemaLocation="uri-of-the-schema"/>
```

これらの他のスキーマをパーサで使用できる場合以外は、検証は失敗します。特に WSDL ドキュメントの場合は、すべてのスキーマをダウンロードして、修正された場所を使用するように主スキーマを編集することが必要になる場合もあります。

- すべての外部エンティティを含め、すべてのエンティティを解決しようとします (この作業は他の XML パーサでも行います)。場所によっては、このプロセスには時間がかかる場合もあります。特に、Xerces では一部の URL の解決にネットワーク・アクセサが使用され、実装ではブロックする I/O が使用されます。結果的に、タイムアウトは発生せず、ネットワーク・フェッチがエラー状態になって停止する可能性があります (現実にはまず発生しません)。

  必要に応じて、カスタム・エンティティ・リゾルバを作成することもできます。"XML ツールの使用法" の "SAX パーサの使用法のカスタマイズ" を参照してください。

# 2

# SOAP Web サービスの作成

このトピックでは、InterSystems IRIS における Web サービスの作成方法の基本について説明します。

Web サービスに関連する URL の概要は、ここをクリックしてください。

## 2.1 InterSystems IRIS Web サービスの概要

InterSystems IRIS で Web サービスを作成するには、%SOAP.WebService を拡張するクラスを作成します。このクラスによって、SOAP プロトコル経由で呼び出し可能な 1 つ以上のメソッドの作成に必要なすべての機能が提供されます。さらに、このクラスは SOAP に関連するブックキーピング (サービスを記述している WSDL ドキュメントの管理など) の管理を自動化します。

## 2.2 基本要件

InterSystems IRIS で Web サービスを作成するには、次の基本要件を満たす InterSystems IRIS クラスを作成してコンパイルします。

- クラスは %SOAP.WebService を拡張したものである必要があります。
- クラスによって SERVICENAME パラメータを定義する必要があります。このパラメータが定義されていないクラスは、InterSystems IRIS でコンパイルされません。
- このクラスによって、WebMethod キーワードを使用してマークを付けるメソッドまたはクラス・クエリを定義する必要があります。

  重要　ほとんどの場合、Web メソッドはインスタンス・メソッドにする必要があります。通常、Web メソッドでは、Web サービス・インスタンスのプロパティを設定し、そのインスタンスのメソッドを呼び出して (詳細はこの後のトピックで説明)、メソッドの動作を調整する必要があります。クラス・メソッドではこれらのタスクを実行できないので、クラス・メソッドは一般に Web メソッドとしては適していません。

- Web メソッドの場合、メソッド・シグニチャ内の各値に XML プロジェクションがあることを確認してください。例えば、メソッドに以下のシグニチャがあったとします。

  ```
  Method MyWebMethod(myarg as ClassA) as ClassB [ WebMethod ]
  ```

  この場合、ClassA と ClassB の両方に XML 表現が含まれている必要があります。つまり、ほとんどの場合、これらのスーパークラス・リストに %XML.Adaptor が含まれていなければなりません。詳細は、"オブジェクトの XML への

投影" を参照してください。InterSystems IRIS SOAP サポートは、このリストの後に示すように、コレクションおよびストリームに対して特別な処理を行います。

Web メソッドは、一般のメソッドと同様に ByRef キーワードと Output キーワードを指定できます (これらのキーワードの詳細は、"クラスの定義と使用" の "メソッド" を参照してください)。

- これらの引数および返り値内に含まれる可能性のある値について考えてみましょう。XML では、印字不能文字、特に ASCII 32 未満の文字は許可されません (キャリッジ・リターン、改行、およびタブは XML で許可されるので例外です)。

  許可されない印字不能文字を含める必要がある場合、タイプを %Binary、(同等の) %xsd.base64Binary、またはサブクラスとして指定します。この値は、XML にエクスポートされるときに、Base 64 のエンコードに自動的に変換されます (またはインポートされるときに、Base 64 のエンコードから自動的に変換されます)。

- 引数の既定の値を指定する場合に、メソッド・シグニチャを利用しないでください。メソッド・シグニチャを利用すると、既定値は無視され、代わりに NULL 文字列が使用されます。例えば、以下のメソッドを考えてみます。

  ```
  Method TestDefaults(val As %String = "Default String") As %String [ WebMethod ]
  ```

  このメソッドを Web メソッドとして呼び出す場合に、引数を指定しないと NULL 文字列が使用され、値 "Default String" は無視されます。

  その代わりに、メソッドの実装の最初に、値をテストし、必要に応じて目的の既定を使用します。1 つの手法は、以下のとおりです。

  ObjectScript

  ```
  if arg="" {
    set arg="Default String"
  }
  ```

  通常通り、メソッド・シグニチャで既定値を示すことができますが、これは情報を提供することのみを目的とし、SOAP メッセージには影響を与えません。

- Web メソッドで必要なすべての引数に対し、メソッド・シグニチャ内で REQUIRED プロパティ・パラメータを指定します。以下はその例です。

  ```
  Method MyWebMethod(myarg as ClassA(REQUIRED=1)) as ClassB [ WebMethod ]
  ```

既定では、継承されたメソッドはすべて、スーパークラスによって Web メソッドとしてマーク付けされていても、一般メソッドとして扱われます (ただし "既存の InterSystems IRIS Web サービスのサブクラスの作成" を参照してください)。

## 2.2.1 %XML.Adaptor を必要としない入力オブジェクトと出力オブジェクト

ほとんどの場合、Web メソッドへの入力または出力としてオブジェクトを使用するときは、そのオブジェクトは %XML.Adaptor を拡張したものでなければなりません。例外は以下のとおりです。

- オブジェクトが、%ListOfDataTypes、%ListOfObjects、%ArrayOfDataTypes、%ArrayOfObjects、またはサブクラスの場合、InterSystems IRIS SOAP サポートは、そのオブジェクトを %XML.Adaptor が含まれているものとして暗黙的に扱います。これらのクラスのサブクラスを作成する必要はありません。ただし、以下の条件があります。

  - メソッド・シグニチャ内で、以下のように ELEMENTTYPE を指定する必要があります。

    Class Member

    ```
    Method MyMethod() As %ListOfObjects(ELEMENTTYPE="MyApp.MyXMLType") [WebMethod]
    {
       //method implementation
    }
    ```

または、入力引数の場合は以下のとおりです。

Class Member

```
Method MyMethod(input As %ListOfObjects(ELEMENTTYPE="MyApp.MyXMLType")) [WebMethod]
{
   //method implementation
}
```

- ELEMENTTYPE で名前を指定したクラスがオブジェクト・クラスの場合、そのクラスは %XML.Adaptor から継承される必要があります。

- オブジェクトがいずれかのストリーム・クラスの場合、InterSystems IRIS SOAP サポートは、そのオブジェクトを %XML.Adaptor が含まれているものとして暗黙的に扱います。そのストリーム・クラスのサブクラスを作成する必要はありません。

  オブジェクトが文字ストリームの場合、InterSystems IRIS SOAP ツールは、そのタイプが文字列であると見なします。また、オブジェクトがバイナリ・ストリームの場合、ツールはそれを Base 64 でエンコードされたデータとして扱います。したがって、タイプ情報を指定する必要はありません。

### 2.2.2 入力または出力としての結果セットの使用法

結果セットは入力または出力として使用できますが、その方法は対象の Web クライアントによって異なります。

- Web サービスと Web クライアントの両方が InterSystems IRIS を基盤とする場合や、いずれかが .NET を基盤とする場合、特殊な結果セット・クラスである %XML.DataSet を使用できます。詳細は、"SOAP メッセージでのデータセットの使用" を参照してください。または、クラス・クエリを Web メソッドとして使用することもできます。XML 表現は自動的に %XML.DataSet と同じになります。
- Java ベースの Web クライアントで使用できるようにクエリの結果を出力するには、%ListOfObjects サブクラスを使用します。SAMPLES ネームスペースの SOAP.Demo に例があります。

## 2.3 簡単な例

このセクションでは、Web サービスの例、および Web サービスが認識できる要求メッセージと対応する応答メッセージの例を示します。

まず、Web サービスは以下のとおりです。

Class Definition

```
/// MyApp.StockService
Class MyApp.StockService Extends %SOAP.WebService
{

/// Name of the WebService.
Parameter SERVICENAME = "StockService";

/// TODO: change this to actual SOAP namespace.
/// SOAP Namespace for the WebService
Parameter NAMESPACE = "http://tempuri.org";

/// Namespaces of referenced classes will be used in the WSDL.
Parameter USECLASSNAMESPACES = 1;

/// This method returns tomorrow's price for the requested stock
Method Forecast(StockName As %String) As %Integer [WebMethod]
{
    // apply patented, nonlinear, heuristic to find new price
    Set price = $Random(1000)
```

```
    Quit price
}
}
```

Web クライアントからこのメソッドを呼び出すと、クライアントによって SOAP メッセージが Web サービスに送信されます。この SOAP メッセージは以下のようになります (改行とスペースが、読みやすいように追加してあります)。

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:s='http://www.w3.org/2001/XMLSchema'>
  <SOAP-ENV:Body>
    <Forecast xmlns="http://tempuri.org">
     <StockName xsi:type="s:string">GZP</StockName>
    </Forecast>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

メッセージ本文 (<SOAP-ENV:Body> 要素) には、<Forecast> という名前の要素が含まれています。これは、クライアントが呼び出しているメソッドの名前です。<Forecast> には、<StockName> という 1 つの要素が含まれています。この要素の名前は、呼び出している Web メソッドの引数の名前に基づいています。この要素には、その引数の実際の値が含まれます。

Web サービスでは、要求されたアクションを実行し、SOAP メッセージを応答として送信します。応答メッセージは、以下のようになります。

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:s='http://www.w3.org/2001/XMLSchema'>
  <SOAP-ENV:Body>
    <ForecastResponse xmlns="http://www.myapp.org">
      <ForecastResult>799</ForecastResult>
    </ForecastResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

これらの例には、SOAP メッセージ自体の前に置かれる HTTP ヘッダは含まれていません。

# 2.4 Web サービスの生成

Web サービスは以下のいずれかの方法で作成できます。

- 新規クラスを作成するか、またはこのトピックで前述の要件に従うように既存のクラスを編集する
- Web サービス・ウィザードを使用する
- SOAP ウィザードと既存の WSDL を併用する
- 1 つ以上の InterSystems IRIS Web サービスのサブクラスを作成する

## 2.4.1 Web サービス・ウィザードの使用法

Web サービス・ウィザードでは、簡単なスタブが生成されます。

1. **[ファイル]→[新規作成]** をクリックします。

[新規作成] ダイアログ・ボックスが表示されます。

2. [一般] タブをクリックします。
3. [新規 Web サービス] をクリックしてから [OK] をクリックします。

   ウィザードが表示されます。
4. パッケージ名、クラス名、および Web サービス名の値を入力します。これらの入力は必須です。
5. オプションで、ネームスペースの URI を編集します (またはこの初期値を後で変更します)。これは、InterSystems IRIS ネームスペースではなく、XML ネームスペースです。
6. オプションで、別の行にメソッド名のリストを入力します。
7. [OK] をクリックします。

Web メソッドのスタブを含む新しい Web サービス・クラスが作成されました。以下はその例です。

Class Definition

```
/// MyApp.StockService
Class MyApp.StockService Extends %SOAP.WebService
{

/// Name of the WebService.
Parameter SERVICENAME = "StockService";

/// TODO: change this to actual SOAP namespace.
/// SOAP Namespace for the WebService
Parameter NAMESPACE = "http://tempuri.org";

/// Namespaces of referenced classes will be used in the WSDL.
Parameter USECLASSNAMESPACES = 1;

/// TODO: add arguments and implementation.
/// Forecast
Method Forecast() As %String [ WebMethod ]
{
   ;Quit "Forecast"
}

}
```

## 2.4.2 SOAP ウィザードと既存の WSDL の併用

WSDL が設計済みで、その WSDL に合わせて Web サービスを作成する必要があることも考えられます。これは、WSDL-first の開発と呼ばれています。InterSystems IRIS では、3 つの手順でこの開発を行います。

1. SOAP ウィザードを使用して、WSDL を読み取り、Web サービス・クラスおよびサポートするすべてのクラスを生成します。

   このウィザードでは、Web クライアント・クラスも生成できます (こちらのほうがより一般的です)。

   このウィザードの使用法の詳細は、"SOAP ウィザードの使用法" を参照してください。そのセクションで説明する手順に従い、さらにウィザード内の [ウェブ・サービスの作成] オプションを選択してください。

   または、"プログラムによるクライアント・クラスの生成" の説明に従って、%SOAP.WSDL.Reader クラスを使用します。
2. 生成されたクラスを検証して、メソッド・シグニチャ内の %String の値のいずれかに変更を加える必要があるかどうかを確認します。

   ウィザードで WSDL を読み取る場合、InterSystems IRIS では文字列タイプの入出力を %String として表現できることが前提となっていますが、そのように表現できないこともあります。まれに、文字列によっては、最大文字列長の制限を超えることがあります。

   "生成されたクラスをきわめて長い文字列に合わせて調整する方法" を参照してください。

3. 生成された Web サービスで、必要なアクションを実行するようにメソッドを編集します。

   当初、各メソッドは次の例のようなスタブです。

   Class Member

```
Method Add(a As Test.ns2.ComplexNumber, b As Test.ns2.ComplexNumber) As Test.ns2.ComplexNumber
[ Final, SoapAction = "http://www.mynamespace.org/GSOAP.AddComplexWS.Add",
SoapBindingStyle = document, SoapBodyUse = literal, WebMethod ]
{
 // Web Service Method Implementation Goes Here.
}
```

   ウィザードには、Final や SoapBindingStyle などのコンパイラ・キーワードが含まれます。これらのキーワードの値は変更しないでください。

WSDL に WS-Policy 要素が含まれている場合、ウィザードでは、Web サービスの構成クラスも生成されます。既定の構成クラス名は、Web サービス名に `Config` を追加したものです。WS-Policy については、"Web サービスの保護" を参照してください。

## 2.4.3 既存の InterSystems IRIS Web サービスのサブクラスの作成

Web サービスを作成するには、既存の InterSystems IRIS Web サービス・クラスのサブクラスを作成してから、以下のようにクラスに SOAPMETHODINHERITANCE パラメータを追加します。

Class Member

```
PARAMETER SOAPMETHODINHERITANCE = 1;
```

このパラメータの既定値は 0 です。このパラメータが 0 の場合、クラスは Web メソッドを Web メソッドとして継承しません。つまり、それらのメソッドは、一般メソッドとしては使用可能ですが、サブクラスによって定義された Web サービス内の Web メソッドとしてアクセスすることはできません。

このパラメータを 1 に設定した場合、クラスは、Web サービスであるスーパークラスで定義された Web メソッドを使用することができます。

# 2.5 Web サービスのパラメータの指定

Web サービス・クラスで以下のパラメータに適切な値を使用していることを確認します。

注釈　SOAP ウィザードを使用して既存の WSDL から Web サービスを生成する場合、これらのパラメータはいずれも変更しないでください。

### SERVICENAME

Web サービスの名前。この名前は、文字から開始し、英数字のみを使用する必要があります。

このパラメータが定義されていないクラスは、InterSystems IRIS でコンパイルされません。

### NAMESPACE

Web サービスとそのコンテンツが他のサービスと競合しないように、Web サービスにターゲットのネームスペースを定義する URI。初期設定では、`"http://tempuri.org"` に設定されています。これは、SOAP 開発者が開発中に一時的なネームスペースとしてよく使用する URI です。

このパラメータを指定しないと、ターゲットのネームスペースは、`"http://tempuri.org"` となります。

InterSystems IRIS Web サービスには、要求メッセージを別のネームスペースに配置する方法は用意されていません。ただし、InterSystems IRIS Web クライアントにはこのような制限はありません。"メッセージのネームスペース" を参照してください。

RESPONSENAMESPACE

応答メッセージのネームスペースを定義する URI。既定では、NAMESPACE パラメータで指定されるネームスペースと同じです。

InterSystems IRIS Web サービスには、応答メッセージを別のネームスペースに配置する方法は用意されていません。ただし、InterSystems IRIS Web クライアントにはこのような制限はありません。"メッセージのネームスペース" を参照してください。

TYPENAMESPACE

Web サービスによって定義されたタイプのスキーマのネームスペース。このパラメータを指定しない場合、スキーマは Web サービスのターゲットのネームスペース (NAMESPACE、または既定の "http://tempuri.org") に配置されます。

InterSystems IRIS Web サービスには、要求メッセージのタイプを別のネームスペースに配置する方法は用意されていません。InterSystems IRIS Web クライアントにはこのような制限はありません。"タイプのネームスペース" を参照してください。

RESPONSETYPENAMESPACE

応答メッセージで使用されるタイプのネームスペースを定義する URI。既定では、TYPENAMESPACE パラメータで指定されるネームスペースと同じです。

このパラメータは、SoapBindingStyle が "document" (既定) の場合にのみ使用されます。

InterSystems IRIS Web サービスと InterSystems IRIS Web クライアントのいずれについても、応答メッセージのタイプがすべて同一のネームスペースに属している必要があります。

SOAPVERSION

Web サービスの WSDL で通知される SOAP のバージョン (複数可) を指定します。以下の値のいずれかを使用します。

- "" — SOAP 1.1 または 1.2 の場合はこの値を使用します。
- "1.1" — SOAP 1.1 の場合はこの値を使用します。これが既定値です。
- "1.2" — SOAP 1.2 の場合はこの値を使用します。

Web サービスが SOAP 要求を受け取ると、Web サービスの SoapVersion プロパティは、その要求の SOAP バージョンと等しくなるよう更新されます。

"Web サービスで処理する SOAP バージョンの制限" も参照してください。

これらの値が WSDL に及ぼす影響の詳細は、"生成された WSDL の詳細" を参照してください。

# 2.6 カタログおよびテスト・ページについて

Web サービス・クラスをコンパイルすると、クラス・コンパイラによって便利なカタログ・ページが生成されます。そのカタログ・ページを使用して、Web サービスを検証することができます。このカタログ・ページは簡易的な、制限テストのページにリンクしています (生成もされます)。これらのページは、既定では無効になっています。テスト環境でのみ有効にします。

### 2.6.1 カタログおよびテスト・ページへのアクセス

現在使用しているネームスペースに Web アプリケーションが存在しない場合は、カタログおよびテスト・ページにアクセスできません。"Web アプリケーションの一部としての InterSystems IRIS Web サービス" を参照してください。また、既定ではこれらのページにはアクセスできません。これらのページにアクセスできるようにするには、ターミナルを開いて %SYS ネームスペースに移動し、次のコマンドを入力します。

```
set ^SYS("Security","CSP","AllowClass",webapplicationname,"%SOAP.WebServiceInfo")=1
set ^SYS("Security","CSP","AllowClass",webapplicationname,"%SOAP.WebServiceInvoke")=1
```

webapplicationname は、末尾にスラッシュを付けた Web アプリケーションの名前です。例えば、"/csp/mynamespace/" のようになります。

これらのページは、%Development リソースに対する USE 特権を持つユーザとしてログインした場合にのみ使用できます。

### 2.6.2 カタログおよびテスト・ページの表示

カタログ・ページの URL は以下のような構造になっています。

```
base/csp/app/web_serv.cls
```

ここで base は Web サーバのベース URL (必要な場合はポートを含む)、/csp/app は Web サービスがある Web アプリケーションの名前、web_serv は Web サービスのクラス名です。(通常、/csp/app は /csp/namespace です。)以下はその例です。

```
http://localhost:52773/csp/samples/MyApp.StockService.cls
```

### 2.6.3 これらのページに関する注意事項

カタログ・ページには、クラス名、ネームスペース、サービス名、およびクラスと Web メソッドについてのコメントが表示されます。[**サービス詳細**] リンクには、生成された WSDL が表示されます。詳細は "WSDL の表示" を参照してください。次にページに Web メソッドがリンクと共に表示されます (適切な許可がある場合)。指定されたメソッドのリンクにテスト・ページが表示され、そこから限られた方法でメソッドをテストできます。

テスト・ページに関する注記

- SOAP 要求は表示できません。
- このテスト・ページは、SOAP 経路を完全にテストするわけではありません。つまり、このトピックの後述部分で説明される SOAP ログなどは作成されません。
- 簡単なリテラル入力のみを受け付けるので、引数がオブジェクト、コレクション、またはデータセットとなっているメソッドの呼び出しには使用できません。

このドキュメントでは、このページについてこれ以上は説明しません。Web サービスをより完全にテストするには、"Web クライアントの作成" の説明に従い、Web クライアントを生成して使用します。

## 2.7 WSDL の表示

Web サービスの定義に %SOAP.WebService を使用すると、その Web サービスを記述した WSDL ドキュメントが作成されて発行されます。Web サービスを変更してリコンパイルするたびに、それに応じて WSDL が自動的に更新されます。このセクションでは、以下の項目について説明します。

- WSDL および WSDL が発行される URL の表示
- WSDL を静的ドキュメントとして生成する際に使用できるメソッド

"InterSystems IRIS における WSDL のサポート" も参照してください。

重要　定義により、Web サービスとその Web クライアントは、それぞれの実装にかかわらず (また、テクノロジの根本的な変更にかかわらず)、共通インタフェースに準拠する必要があります。WSDL とは、このインタフェースに関する標準準拠の記述のことです。次に示す事項に注意してください。

- 実際には、わずかに異なる複数の WSDL ドキュメントで、1 つの SOAP インタフェースを正しく記述できることが多くあります。

  そのため、InterSystems IRIS で生成した WSDL は、InterSystems IRIS のバージョンごとに、わずかに異なることがあります。このような相違点については、このドキュメントでは説明していません。インターシステムズでは、W3C 仕様で要求されている Web サービスと、そのサービスに対応するクライアントとの相互運用性のみを保証しています。

- W3C 仕様では、適合するインタフェースを記述する WSDL を生成できることを、Web サービスまたは Web クライアントのどちらに対しても要求していません。

  利便性のために、WSDL ドキュメントが生成され、特定の URL で提供されます。ただし、WSDL を収容している Web アプリケーションが、パスワード認証や SSL 接続を要求する場合、この方法で WSDL にアクセスすることは実用的とはいえません。このような場合は、WSDL をファイルにダウンロードして、このファイルを代わりに使用する必要があります。また、前述したように、生成された WSDL には、実行時に追加される SOAP ヘッダに関する情報が含まれていません。実行時に追加される SOAP ヘッダに関する情報を WSDL ドキュメントに含める必要がある場合は、WSDL をファイルにダウンロードして、そのファイルを適切に変更してから使用する必要があります。

## 2.7.1 WSDL の表示

Web サービスの WSDL を表示するには、次の URL を使用します。

```
base/csp/app/web_serv.cls?WSDL
```

ここで base は Web サーバのベース URL (必要な場合はポートを含む)、/csp/app は Web サービスがある Web アプリケーションの名前、web_serv は Web サービスのクラス名です。(通常、/csp/app は `/csp/`namespace です。)

注釈　クラス名内のパーセント記号 (%) は、この URL ではアンダースコア文字 (_) に置き換えられます。

以下はその例です。

```
http://localhost:52773/csp/samples/MyApp.StockService.cls?WSDL
```

ブラウザに WSDL ドキュメントが表示されます。以下に例を示します。


重要　すべてのブラウザでスキーマが正しく表示されるわけではありません。実際のスキーマを確認するには、ページのソースの表示が必要になることもあります。例えば、Firefox では右クリックして [**ページのソースを表示**] を選択します。

## 2.7.2 WSDL の生成

WSDL は静的ドキュメントとしても生成できます。**%SOAP.WebService** クラスには、そのために使用できるメソッドがあります。

FileWSDL()

```
ClassMethod FileWSDL(fileName As %String, includeInternalMethods As %Boolean = 1) As %Status
```

ここで、fileName はファイルの名前で、includeInternalMethods は、生成された WSDL に Internal としてマークされた Web メソッドが含まれるかどうかを指定します。

## 2.7.3 WSDL からの Internal Web メソッドの抑制

Web サービスに Internal とマークされている Web メソッドがある場合、既定では WSDL にこれらの Web メソッドが含まれます。これらのメソッドが WSDL に含まれないようにすることができます。これを実行するには、以下のいずれかを実行します。

- Web サービスの FileWSDL() メソッドを使用して WSDL を生成します。前のセクションを参照してください。このメソッドは、WSDL に内部 Web メソッドが含まれるかどうかを制御する引数を提供します。
- Web サービス・クラスの SOAPINTERNALWSDL クラス・パラメータを 0 に指定してください (このクラス・パラメータの既定値は 1 です)。

# 3

# SOAP メッセージのバリエーション

このトピックでは、SOAP メッセージの主なバリエーションと、InterSystems IRIS Web サービスおよび Web クライアントでのそれらの生成方法を説明します。

InterSystems IRIS Web サービスまたは Web クライアントでは、いくつかのキーワードと 1 つのパラメータによって各 Web メソッドで使用されるメッセージ・バリエーションを指定します。Web サービスを手動で作成した場合は、通常はこれらの項目の既定値が適した値となります。一方、Web サービスまたは Web クライアントを SOAP ウィザードを使用して作成した場合は、WSDL による要求に従って値が設定されます。ただし、状況によっては、特定のメッセージ・バリエーションを選択することが必要になることがあります。

## 3.1 概要

SOAP メッセージのモードは以下のいずれかとなり、WSDL によって形式的に決定されます。

- Document/literal ー これは、InterSystems IRIS Web サービスの既定のメッセージ・モードであり、最も広く使用されるモードです。

  このメッセージ・モードでは、ドキュメントスタイルのバインディングとリテラル・エンコード形式を使用します。バインディングとエンコード形式については、サブセクションで簡単に説明します。

- RPC/encoded ー これは、2 番目に広く使用されるモードです。
- RPC/literal ー このモードは、IBM によって広く使用されます。
- Document/encoded ー このモードが使用されることは非常にまれなため、お勧めしません。また、このモードは、WS-I Basic Profile 1.0 に準拠していません。

非公式には、document/literal メッセージには追加のバリエーションが考えられます。それらは、wrapped (InterSystems IRIS での既定値) または unwrapped のいずれかです。ラップされたメッセージでは、メッセージにサブパートを含む単一のパートがあります。これは、複数の引数を取るメソッドの場合に該当します。ラップされたメッセージでは、引数はこのメッセージ内のサブパートになります。ラップされていないメッセージでは、メッセージに複数のパート (引数ごとに 1 つ) があります。

RPC メッセージには、複数のパートが存在する場合があります。

### 3.1.1 バインディング・スタイル

各 Web メソッドは、Web メソッドの出入力に対するバインディング・スタイルを持ちます。バインディング・スタイルは、ドキュメントまたは RPC のいずれかです。バインディング・スタイルは、WSDL バインディングを SOAP メッセージに変換する方法を決定します。また、SOAP メッセージの本文の形式も制御します。

### 3.1.2 エンコード形式

各 Web メソッドは、エンコード形式も持ちます。エンコード形式は、リテラルまたはエンコード (SOAP エンコード) のいずれかです。SOAP 1.1 と SOAP 1.2 では、エンコードの詳細が少し異なります。リテラル形式と SOAP エンコード形式の違いの詳細は、"オブジェクトの XML への投影" を参照してください。

## 3.2 メッセージ・バリエーションの決定方法

InterSystems IRIS Web サービスまたは Web クライアントでは、サービス・クラスやクライアント・クラスの詳細によって、各 Web メソッドで使用されるメッセージ・モードが制御されます。これらの詳細は以下のとおりです。

- SoapBindingStyle クラス・キーワードおよび SoapBindingStyle メソッド・キーワード。メソッド・キーワードが優先されます。
- SoapBodyUse クラス・キーワードおよび SoapBodyUse メソッド・キーワード。メソッド・キーワードが優先されます。
- ARGUMENTSTYLE クラス・パラメータ。

以下のテーブルは、InterSystems IRIS Web メソッド用のメッセージ・モードの決定方法をまとめたものです。

| メッセージ・モード | SoapBindingStyle | SoapBodyUse | ARGUMENTSTYLE |
|---|---|---|---|
| wrapped document/literal | `document` (既定) | `literal` (既定) | `wrapped` (既定) |
| unwrapped document/literal | `document` | `literal` | |
| rpc/encoded | `rpc` | `encoded` | 無視 |
| rpc/literal | `rpc` | `literal` | 無視 |
| document/encoded | `document` | `encoded` | 無視 |

SOAP ウィザードを使用して Web サービス・クラスまたは Web クライアント・クラスを生成する場合、ウィザードでは、開始元の WSDL に応じてそれらのキーワードおよびパラメータの値が設定されます。

重要　手動で作成した Web サービスでは、通常はこれらの既定値が最適な値となります。

SOAP ウィザードを使用して WSDL から Web クライアントまたは Web サービスを作成すると、InterSystems IRIS は、その WSDL に応じて、これらのキーワードを設定します。この値を変更すると、Web クライアントまたは Web サービスは機能しなくなる場合があります。

## 3.3 メッセージ・バリエーションの例

ここでは、参考情報として、各モードのメッセージの例を紹介します (ただし、document/encoded は非推奨のため除きます)。

また、"生成された WSDL の詳細" の "<message>" も参照してください。

### 3.3.1 Wrapped Document/Literal

これは、最も一般的なメッセージ・スタイルであり、InterSystems IRIS Web サービスの既定のメッセージ・スタイルです。

XML

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
                   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
                   xmlns:s='http://www.w3.org/2001/XMLSchema'>
  <SOAP-ENV:Body>
    <MyMethod xmlns="http://www.demoservice.org">
        <A>stringA</A>
        <B>stringB</B>
        <C>stringC</C>
    </MyMethod>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## 3.3.2 Message/Unwrapped Document/Literal

これは、前のスタイルを少し変化させたものです。

XML

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
                   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
                   xmlns:s='http://www.w3.org/2001/XMLSchema'>
  <SOAP-ENV:Body>
      <A xmlns="http://www.demoservice.org">stringA</A>
      <B xmlns="http://www.demoservice.org">stringB</B>
      <C xmlns="http://www.demoservice.org">stringC</C>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## 3.3.3 RPC/Encoded

これは、2 番目に広く使用されるスタイルです。以下に SOAP 1.1 の rpc/encoded メッセージを示します。

XML

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
                   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
                   xmlns:s='http://www.w3.org/2001/XMLSchema'
                   xmlns:SOAP-ENC='http://schemas.xmlsoap.org/soap/encoding/'
                   xmlns:tns='http://www.demoservice.org'
                   xmlns:types='http://www.demoservice.org'>
  <SOAP-ENV:Body SOAP-ENV:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
    <types:MyMethod>
      <A>stringA</A>
      <B>stringB</B>
      <C>stringC</C>
    </types:MyMethod>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP 1.2 では、エンコードの規則が異なるため、以下のようにメッセージが異なります。

XML

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://www.w3.org/2003/05/soap-envelope'
                   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
                   xmlns:s='http://www.w3.org/2001/XMLSchema'
                   xmlns:SOAP-ENC='http://www.w3.org/2003/05/soap-encoding'
                   xmlns:tns='http://www.demoservice.org'
                   xmlns:types='http://www.demoservice.org'>
  <SOAP-ENV:Body>
    <types:MyMethod SOAP-ENV:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <A>stringA</A>
      <B>stringB</B>
      <C>stringC</C>
    </types:MyMethod>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## 3.3.4 RPC/Literal

rpc/literal メッセージの例を次に示します。

### XML

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
                   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
                   xmlns:s='http://www.w3.org/2001/XMLSchema'
                   xmlns:tns='http://www.demoservice.org'>
  <SOAP-ENV:Body>
    <tns:MyMethod>
      <tns:A>stringA</tns:A>
      <tns:B>stringB</tns:B>
      <tns:C>stringC</tns:C>
    </tns:MyMethod>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# 4

# Web クライアントの作成

Web クライアントは、Web サービスにアクセスするソフトウェアです。Web クライアントには、一連のプロキシ・メソッドが用意されています。各プロキシ・メソッドは、Web サービスのメソッドに対応しています。プロキシ・メソッドは、対応する Web サービス・メソッドと同じシグニチャを使用し、必要に応じて Web サービス・メソッドを呼び出します。このトピックでは、InterSystems IRIS で Web クライアントを作成し、使用する方法を説明します。

InterSystems IRIS Web クライアントに対する SOAP 呼び出しをログに記録する方法については "InterSystems IRIS SOAP ログ" を参照してください。

注釈　InterSystems IRIS Web サービスの場合、自動的に生成された WSDL には、SOAP ヘッダ要素に関する情報が含まれていないことがあります。

- HeadersOut プロパティを設定することにより、手動で SOAP ヘッダを追加する場合は、"カスタム・ヘッダ要素の追加と使用" の "サポート対象のヘッダ要素の指定" の手順を実行します。これにより、すべての該当情報が WSDL に含まれるようになります。それ以外の場合は、これが行われなくなるため、WSDL をファイルに保存して、そのファイルを必要に応じて手動で編集することが必要になります。
- SecurityOut プロパティ ("Web サービスの保護" を参照) を設定して、WS-Security ヘッダ要素を追加した場合、WSDL には必要な情報の一部が含まれなくなります。(これは、コンパイル時に WSDL が生成され、その後の実行時にヘッダが追加されるためです。)この場合は、WSDL をファイルに保存して、そのファイルを必要に応じて手動で編集してください。

  多くの理由から、WS-Policy を使用することで、WS-Security 要素の追加が簡潔で容易になります ("ポリシーの作成と使用" を参照)。WS-Policy を使用すると、生成された WSDL には必要な情報がすべて含まれます。
- その他の場合は、生成された WSDL にすべての情報が含まれます。

W3C 仕様では、生成された WSDL を Web サービスが提供することを要求していない点に注意してください。

## 4.1 SOAP ウィザードの概要

InterSystems IRIS Web クライアントを作成するには、スタジオの SOAP ウィザードを使用するか、InterSystems IRIS に用意された対応するクラス・メソッドを使用します。どちらの場合も、入力は WSDL ドキュメントです。このツールでは、Web クライアント・クラスおよびサポートするすべての必要なクラスが生成されます。

このツールは、ほぼすべての WSDL ドキュメントで使用できます。"InterSystems IRIS における WSDL のサポート" を参照してください。

WSDL の URL またはファイル・パスのいずれかを指定できます。

注釈　WSDL が SOAP 1.1 および SOAP 1.2 の両方のサポートを示す場合は、SOAP ウィザードは必要に応じて 2 つのクラスのセットを生成します。

# 4.2 SOAP ウィザードの使用法

特定の Web サービスを記述している WSDL にアクセスできる場合は、スタジオの SOAP ウィザードを使用して、そのサービスの Web クライアントを生成できます。

注釈　プロキシ・サーバが有効になっている場合、スタジオは SOAP ウィザードなどのテンプレートから通信する際に、プロキシ・サーバを使用します。プロキシ・サーバとポートの指定については、"インターネット・ユーティリティの使用法" の "プロキシ・サーバの使用法" を参照してください。

SOAP ウィザードを使用するには:

1. スタジオで、**[ツール]**→**[アドイン]**→**[SOAP ウィザード]** をクリックします。
2. **SOAP ウィザード**の最初の画面で、WSDL の場所および WSDL へのアクセスに必要な SSL 構成を指定します。
   a. **[URL]** または **[FILE]** をクリックして、WSDL の形式を示します。
   b. WSDL URL を入力するか、WSDL ファイルを参照します。
   c. SSL 認証を必要とする URL を指定する場合は (URL が `https` で始まる場合)、以下を行います。
      1. **[SSL構成]** ドロップダウン・リストから SSL 構成を選択します。

         SSL/TLS 構成の作成と管理の詳細は、インターシステムズの "TLS ガイド" を参照してください。

         重要　**[SSL構成]** フィールドでは、ウィザードが WSDL へのアクセスに使用する SSL 構成のみを指定します。

      2. 必要に応じて、**[SSL 接続を行う場合、接続しているシステムのシステム名がサーバ証明書のサーバIDと一致していることを確認してください]** チェック・ボックスのチェックを外します。

         このチェック・ボックスにチェックが付けられている場合、ウィザードは、証明書サーバ名がサーバへの接続に使用される DNS 名と一致するかどうかを判断します。名前が一致していないと、接続は許可されません。この既定の動作は、中間者攻撃の防止を目的としています。RFC 2818 のセクション 3.1 に説明があります。また、RFC 2595 のセクション 2.4 で詳細も参照してください。

   d. **[次へ]** をクリックします。

      ウィザードは WSDL にアクセスしてこれを表示しようとします。

      失敗すると、エラーが表示されます。考えられる原因については、"WSDL を利用する際の問題" を参照してください。

      Tipヒント　何回か試行しても、WSDL URL にアクセスできない場合は、代わりに WSDL をファイルとして保存し、これを参照することができます。

      アクセスに成功した場合は、**[ステップ 2]** 画面が表示されます。

   e. WSDL URL がパスワード認証を要求する場合は、資格情報を指定します。
      1. 使用する資格情報のタイプを選択します。
         - InterSystems IRIS の資格情報の場合は、**[InterSystems IRIS ユーザ名とパスワード]** を選択します。
         - HTTP 基本認証の資格情報の場合は、**[HTTP 認証のユーザ名とパスワード]** を選択します。

2. **[ユーザ名]** フィールドと **[パスワード]** フィールドに入力します。
3. **[やり直してください]** をクリックします。

   ウィザードでエントリは保存されません。

   ユーザ名とパスワードが有効な場合、**[ステップ 2]** 画面が表示されます。

詳細は、"パスワードで保護された WSDL URL の使用法" を参照してください。

3. **SOAP ウィザード**の **[ステップ 2]** 画面で、ウィザードが WSDL からクラスを生成する方法を指定します。
   a. **[クラス生成とコンパイルを制御するオプション]** 領域で設定を構成します。

      設定の詳細は、"クラス生成とコンパイルを制御する SOAP ウィザードのオプション" を参照してください。

   b. **[次へ]** をクリックします。

      **[ステップ 3]** 画面が表示されます。

4. **SOAP ウィザード**の **[ステップ 3]** 画面で、ウィザードが WSDL から生成したクラスをパッケージ化する方法を指定します。
   a. 画面上部で設定を構成して、ウィザードが WSDL の XML ネームスペースからクラス・パッケージを生成する方法を決定します。

      設定の詳細は、"XML ネームスペースに関する SOAP ウィザードのオプション" を参照してください。

   b. 必要に応じて、クラス・パッケージ名を編集します。

      **[パッケージ名]** フィールドの詳細は、"クラス・パッケージ名についての SOAP ウィザードのルール" を参照してください。

   c. **[次へ]** をクリックします。

      ウィザードでクラスが生成、コンパイル、およびリストされます。その後、**[ステップ 4]** 画面が表示されます。

      注釈　スキーマの要素名がアンダースコア (_) で始まる場合、その要素に生成されるクラスのプロパティはパーセント記号 (%) で始まります。

5. **[完了]** をクリックします。

## 4.2.1 クラス生成とコンパイルを制御する SOAP ウィザードのオプション

SOAP ウィザードの **[ステップ 2]** 画面にある以下のオプションを使用して、ウィザードが WSDL から生成するクラスのタイプを指定できます。

**[ウェブサービスに対するクライアントを作成]**

ウィザードが WSDL で定義された Web サービスに対してクライアントとして機能するプロキシ・クラスを生成するかどうかを指定します。

**[ウェブサービスを作成]**

ウィザードが WSDL に基づいて、InterSystems IRIS Web サービスとして機能するクラスを生成するかどうかを指定します。

**[生成されたクラスをコンパイル]**

クラスの生成後にウィザードがクラスをコンパイルするかどうかを指定します。

[生成されたクラスをコンパイル] を選択した場合、[コンパイルフラグ] フィールドでフラグを指定して、コンパイラの動作を制御できます。詳細を確認するには、以下のコマンドを実行します。

ObjectScript

```
Do $System.OBJ.ShowFlags()
```

### [クラス・タイプ]

ウィザードが WSDL から生成するクラスのタイプを指定します。以下のいずれかのオプションを選択できます。

- [永続] – クラスは %Persistent から継承されることを指定します。コレクションはリストとして定義されます。
- [一対多リレーションシップを使用する永続] – クラスは %Persistent から継承されることを指定します。コレクション・プロパティは一体多のリレーションシップとして定義されます。
- [永続で一対多リレーションシップにインデックスを使用] – クラスは %Persistent から継承されることを指定します。コレクション・プロパティは一体多のリレーションシップとして定義され、InterSystems IRIS は各リレーションシップのインデックスを定義します。
- [リレーションシップに親子を使用する永続] – クラスは %Persistent から継承されることを指定します。コレクション・プロパティは親子のリレーションシップとして定義されます。
- [Serial] – クラスは %SerialObject から継承されることを指定します。
- [Registered] – クラスは %RegisteredObject から継承されることを指定します。

### [カスケード削除するために %OnDelete メソッドをクラスに追加する]

永続クラス・タイプについて、生成された各クラス定義に %OnDelete() コールバック・メソッドを実装するかどうかを指定します。[リレーションシップに親子を使用した永続化] を選択した場合は、このオプションを使用しないでください。

生成された %OnDelete() メソッドによって、クラスで参照されるすべての永続オブジェクトが削除されます。

生成されたクラスを変更する場合、必要に応じて必ず、対応する %OnDelete() コールバック・メソッドを変更してください。

### [プロキシクラスパッケージ]

Web クライアントおよび生成されたクラスのパッケージ名です。

既定のパッケージ名は、サービス名です。

既存のパッケージ名を指定した場合、既定では、新たに生成されたクラスと同じ名前を持つ既存のクラスはツールによって上書きされます。

### [ビジネスオペレーション作成]

プロダクションで使用できる、ビジネス・オペレーションおよび関連する要求メッセージ・クラスと応答メッセージ・クラスを生成するかどうかを指定します。

プロダクションの詳細は、"相互運用プロダクションの概要" および "プロダクション内での SOAP サービスおよび Web クライアントの追加" を参照してください。

[ビジネスオペレーション作成] を選択する場合、以下の値を指定する必要があります。

- [ビジネス・オペレーションのパッケージ] – ビジネス・オペレーション・クラスのパッケージ名。
- [要求パッケージ] – 要求メッセージ・クラスのパッケージ名。
- [応答パッケージ] – 応答メッセージ・クラスのパッケージ名。

## 4.2.2 XML ネームスペースに関する SOAP ウィザードのオプション

SOAP ウィザードの [ステップ 3] 画面にある以下のオプションを使用して、WSDL から生成されるクラス・パッケージを構成できます。

### [NAMESPACEクラスパラメータ追加]

NAMESPACE クラス・パラメータが Web サービスのネームスペースに設定されて、生成されたタイプ・クラスに組み込まれるかどうかを指定します。

- 指定されたタイプが属するネームスペースが WSDL で明示的に示されている場合は、**[NAMESPACE クラス・パラメータを追加]** はチェックが付けられた状態で、グレー表示になります。この場合、NAMESPACE クラス・パラメータがそのネームスペースに設定され、生成されたタイプ・クラスに組み込まれます。
- 指定されたタイプのネームスペースを WSDL が示していない場合は、**[NAMESPACE クラス・パラメータを追加]** にチェックを付けるか、外すかを選択できます。

### [ドキュメント形式のウェブ・メソッドで改行しないメッセージ形式を使用]

生成された Web クライアントのメソッドで、改行しないメッセージ形式を使用するかどうかを指定します。このオプションは、SoapBindingStyle を "document" に設定したメソッドにのみ影響します。

以下のいずれかの文が WSDL に当てはまる場合は、このチェック・ボックスにチェックを付けます。

- `<message>` 要素に複数のパートが含まれている。
- 応答メッセージで使用するタイプが複数のネームスペースに属している。

そうでなければ、ウィザードは失敗し、次のようなエラー・メッセージが表示されます。

```
ERROR #6425: Element 'wsdl:binding:operation:msg:input' - message 'AddSoapOut'
Message Style must be used for document style message with 2 or more parts.
```

詳細は、"生成されたクラスの詳細" および "InterSystems IRIS Web クライアント・クラスの使用法" を参照してください。

### [配列プロパティを作成しない]

ウィザードで配列プロパティを生成するかどうかを指定します。

このオプションを選択すると、ウィザードでは、配列プロパティが生成されない代わりに別の形式が生成されます。詳細は、"配列プロパティの作成" を参照してください。

### [ヌル可能な要素に XMLNIL プロパティ・パラメータを生成]

生成されたクラスの使用可能なプロパティに対して、XMLNIL プロパティ・パラメータをウィザードで指定するかどうかを示します。

このオプションは `nillable="true"` で指定された XML 要素に対応する各プロパティに適用されます。このオプションを選択した場合、ウィザードにより `XMLNIL=1` がプロパティ定義に追加されます。それ以外の場合、このパラメータの追加はありません。

このプロパティ・パラメータの詳細については、"オブジェクトの XML への投影" の "空文字列および NULL 値の処理" を参照してください。

### [ヌル可能な要素に対してプロパティパラメータ XMLNILNOOBJECT を生成する]

生成されたクラスの使用可能なプロパティに対して、XMLNILNOOBJECT プロパティ・パラメータをウィザードで指定するかどうかを示します。

このオプションは nillable="true" で指定された XML 要素に対応する各プロパティに適用されます。このオプションを選択した場合、ウィザードにより XMLNILNOOBJECT=1 がプロパティ定義に追加されます。それ以外の場合、このパラメータの追加はありません。このパラメータの詳細は、"オブジェクトの XML への投影" の "空文字列および NULL 値の処理" を参照してください。

[XMLSEQUENCE パラメータに 0 を設定]

ウィザードが、生成されたクラスで XMLSEQUENCE クラス・パラメータを 0 に設定するかどうかを指定します。

ウィザードは既定では、生成されるクラスのこの値を 1 に設定します。これにより、クラスが WSDL のスキーマで指定された要素の順序に従います。この値は、スキーマに、特定の親の同じ名前の複数の要素がある場合に役立ちます。詳細は、"オブジェクトの XML への投影" の "複数の同名のタグを含む XML ドキュメントの処理" を参照してください。

[XMLIGNORENULL パラメータに 1 をセットして生成]

ウィザードが生成されたクラスで XMLIGNORENULL クラス・パラメータを指定するかどうかを示します。

このオプションを選択した場合、ウィザードにより XMLIGNORENULL=1 が、生成される Web クライアント (または Web サービス) などのクラス定義に追加されます。それ以外の場合、このパラメータの追加はありません。

このクラス・パラメータの詳細は、"オブジェクトの XML への投影" の "空文字列および NULL 値の処理" を参照してください。

[バイナリにはストリームを使用]

ウィザードがタイプ xsd:base64Binary の各要素にタイプ %Stream.GlobalBinary のプロパティを生成するかどうかを指定します。

このオプションを選択すると、生成されるプロパティのタイプが %Stream.GlobalBinary になります。あるいは、プロパティのタイプは %xsd.base64Binary になります。

ウィザードは、タイプ xsd:base64Binary の属性を無視します。

[SECURITYIN クラス・パラメータを指定]

生成されたクライアント・クラスで、SECURITYIN クラス・パラメータの値を指定します。

Web サービス・セキュリティを使用している場合、クライアントにそれらの要素を要求するか、単純にそれらを検証するかで、REQUIRE または ALLOW を使用します。そうでない場合は、IGNORE または IGNOREALL が一般に適しています。詳細は、"Web サービスの保護" の "WS-Security ヘッダの検証" を参照してください。

SECURITYIN パラメータは、関連付けられた (そしてコンパイルされた) 構成クラスにセキュリティ・ポリシーがある場合、無視されます。"Web サービスの保護" を参照してください。

## 4.2.3 クラス・パッケージ名についての SOAP ウィザードのルール

SOAP ウィザードでは、クラス・パッケージ名に次のルールを適用します。

- [ウェブ・クライアントのパッケージ] および [ウェブ・サービスのパッケージ] フィールドで指定されたパッケージには、生成された主な Web クライアント・クラスまたは Web サービス・クラスが含まれます。これらの値は、ウィザードの前のページでのエントリによって初期化されます。
- ウィザードでポリシー・クラスも生成される場合 (WSDL に WS-Policy 情報が含まれているため)、そのクラスは既定で同じパッケージに含まれます。別のパッケージを指定するには、[構成サブパッケージ] で値を指定します。

  既定では、このクラス名は、Web クライアント名または Web サービス名に Config を追加したものです。[構成サブパッケージ] に値を指定すると、この生成されたクラスの名前が Web クライアントまたは Web サービスと同じになります (ただし、代わりにこのクラスは指定されたサブパッケージ内に含まれます)。

WS-Policy については、"Web サービスの保護" を参照してください。

- 生成されるどのサポート・クラスに対しても、ウィザードによって WSDL で使用されているすべてのネームスペースが検出されます。既定では、生成されたクラスはパッケージに分類されます (ネームスペースごとに 1 つのパッケージ)。

  必要に応じて、これらのパッケージ名を編集します。

  すべてのネームスペースが生成されたクラスに対応するとは限りません。例えば、この例の WSDL はネームスペース `http://schemas.xmlsoap.org/wsdl`、`http://schemas.xmlsoap.org/wsdl/mime`、および `http://schemas.xmlsoap.org/wsdl/soap` を使用します。この場合、これらのネームスペースには生成されたクラスがなく、対応するパッケージは生成されません。

# 4.3 プログラムによるクライアント・クラスの生成

プログラムによってクライアント・クラスを生成することもできます。これには、%SOAP.WSDL.Reader クラスを使用します。

結果として生成されたクラスとそれらの構成は、SOAP ウィザードを使用した場合と異なることがあります。SOAP ウィザードでは、生成されたクラスのパッケージに対してより詳細な管理を行うことができます。

プログラムによってクライアント・クラスを生成する手順は以下のとおりです。

1. %SOAP.WSDL.Reader のインスタンスを作成します。
2. 必要に応じて、インスタンスの動作を制御するプロパティを設定します。詳細は、%SOAP.WSDL.Reader のクラス・ドキュメントを参照してください。

   WSDL が SSL を使用する場所にある場合、%SOAP.WSDL.Reader は、証明書サーバ名がサーバへの接続に使用される DNS 名と一致するかどうかをチェックする (既定の動作) 点に注意してください。これらの名前が一致していないと、接続は許可されません。この既定の動作は、中間者攻撃の防止を目的としています。RFC 2818 のセクション 3.1 に説明があります。また、RFC 2595 のセクション 2.4 も参照してください。

   このチェックを無効にするには、インスタンスの SSLCheckServerIdentity プロパティを 0 に設定します。
3. %SOAP.WSDL.Reader で直接サポートされていない方法で HTTP 要求を制御できるようにする必要がある場合は、以下の手順を実行します。

   a. %Net.HttpRequest のインスタンスを作成し、必要に応じてプロパティを設定します。

   b. %SOAP.WSDL.Reader のインスタンスに対して、作成した %Net.HttpRequest のインスタンスと同じ HttpRequest プロパティを設定します。

   例えば、これは、サーバで認証が必要な場合に行います。"インターネット・ユーティリティの使用法" の "HTTP 要求の送信" にある "ログイン資格情報の提供" を参照してください。
4. インスタンスの Process() メソッドを呼び出します。

```
method Process(pLocationURL As %String, pPackage As %String = "") as %Status
```

   - pLocationURL は、Web サービスの WSDL の URL、または WSDL ファイルの名前 (完全パスを含む) でなければなりません。Web サービスの構成によっては、適切なユーザ名とパスワードを指定する文字列を追加する必要があります。例を参照してください。
   - pPackage は、生成されたクラスを配置するパッケージの名前です。パッケージを指定しないと、InterSystems IRIS は、サービス名をパッケージ名として使用します。

注釈　このパッケージが既存のパッケージと同じである場合、既定では、既存の同名のクラスが上書きされます。クラス定義が上書きされないようにするには、クラス定義に次の指定を追加します。

Class Member

```
Parameter XMLKEEPCLASS = 1;
```

ターミナル・セッションの例を以下に示します。

```
set r=##class(%SOAP.WSDL.Reader).%New()
GSOAP>set url="http://localhost:52773/csp/gsop/GSOP.AddComplexWS.CLS?WSDL=1"

GSOAP>d r.Process(url)
Compilation started on 11/09/2009 12:53:52 with qualifiers 'dk'
Compiling class AddComplex.ns2.ComplexNumber
Compiling routine AddComplex.ns2.ComplexNumber.1
Compilation finished successfully in 0.170s.

Compilation started on 11/09/2009 12:53:52 with qualifiers 'dk'
Compiling class AddComplex.AddComplexSoap
Compiling routine AddComplex.AddComplexSoap.1
Compiling class AddComplex.AddComplexSoap.Add
Compiling routine AddComplex.AddComplexSoap.Add.1
Compilation finished successfully in 0.363s.
```

WSDL URL は、Web アプリケーションの一部です。Web アプリケーションは、パスワード認証によって保護されている可能性があります。このような場合に WSDL を使用して InterSystems IRIS Web クライアントまたはサードパーティの Web クライアントを生成する方法の詳細は ″パスワードで保護された WSDL URL の使用法″ を参照してください。

どのような場合でも、必要なユーザ名とパスワードを指定した後、ブラウザから WSDL を取得して、これをファイルとして保存し、そのファイルを代わりに使用することもできます。

# 4.4 生成された Web クライアント・クラスの変更

通常は、InterSystems IRIS Web クライアント・クラスを生成した後で、そのクラスを編集することはありません。その代わりに、Web クライアントのインスタンスを作成するコードと、クライアント側のエラー処理を提供するコードを記述します。このセクションでは、生成されたクライアント・クラスに変更を加える際の主な例外について説明します。

″生成されたクラスの Web メソッドに関するその他の注意事項″ と ″InterSystems IRIS での Web クライアントの調整″ も参照してください。

注釈　生成された Web クライアント・クラスのサブクラスは作成しないでください。コンパイラは、適切な実行に必要なサポート・クラスを生成しないため、そのサブクラスは使用できなくなります。

## 4.4.1 生成されたクラスをきわめて長い文字列に合わせて調整する方法

まれに、値の長さが文字列長の制限を超える、きわめて長い文字列やバイナリ値に対応できるように、生成されたクライアント・クラスの編集が必要になることがあります。

SOAP ウィザードで WSDL を読み取る場合、InterSystems IRIS では文字列タイプの入出力を %String として表現できることが前提となっていますが、そのように表現できないこともあります。まれに、文字列が文字列長の制限を超えることがあります。同様に、InterSystems IRIS では XML タイプ base64Binary の入出力を %xsd.base64Binary として表現できることが前提となっていますが、文字列長に関する同様の制限によって、そのように表現できないこともあります。どちらの場合も、WSDL には、その入出力が文字列長の制限を超える可能性があることを示す情報は含まれません。

長すぎる文字列またはバイナリ値が Web クライアントで検出されると、次のいずれかのエラーがスローされます。

- <MAXSTRING> エラー
- データ型検証エラー

```
ERROR #6232: Datatype validation failed for tag your_method_name ...
```

(当然ながら、このエラーはデータ型の不一致が原因で発生することもあります。)

ただし、この問題は簡単に修正できます。生成された Web クライアント・クラス (特に、%SOAP.WebClient から継承されたクラス) のメソッド・シグニチャを、適切なストリーム・クラスを使用するように調整します。

- %String ではなく、%GlobalCharacterStream を使用します。
- %xsd.base64Binary ではなく、%GlobalBinaryStream を使用します。

例えば、引数を取らず、非常に長い文字列を返すメソッド (`WriteIt`) が 1 つ含まれる Web サービス (`MyGiantStringService`) について考えてみます。SOAP ウィザードを使用して Web クライアント・クラスを生成すると、Web クライアント・クラスは最初は次のようになります。

### Class Definition

```
Class GetGiantString.MyServiceSoap Extends %SOAP.WebClient
{

Method WriteIt() As %String
[Final,SoapBindingStyle=document,SoapBodyUse=literal,WebMethod]
{
 Quit ..WebMethod("WriteIt").Invoke($this,"http://tempuri.org/MyApp.MyGiantStringService.WriteIt")
}

}
```

この場合、行う調整は 1 つだけです。`WriteIt` の返りタイプを次のように変更します。

### Class Member

```
Method WriteIt() As %GlobalCharacterStream
[Final,SoapBindingStyle=document,SoapBodyUse=literal,WebMethod]
{
 Quit ..WebMethod("WriteIt").Invoke($this,"http://tempuri.org/MyApp.MyGiantStringService.WriteIt")
}
```

このクラスをコンパイルすると、関連するクラスが必要に応じて自動的に再生成されます。

生成されたタイプのクラス内のプロパティ・タイプの調整が必要な場合があります。例えば、タイプ文字列の要素 `<ContainerPart>` が組み込まれた `<Container>` という要素を Web サービスで使用しているとします。InterSystems IRIS Web クライアント・クラスを生成すると、タイプ %String の `ContainerPart` プロパティによって `Container` クラスが作成されます。Web サービスから `<ContainerPart>` 要素で文字列長の制限を超える文字列が送信された場合、Web クライアントはエラーをスローします。このエラーを回避するには、`ContainerPart` プロパティのタイプを %GlobalCharacterStream に変更します。

## 4.4.2 その他の調整

WSDL が Web サービスの場所を指定しない場合、SOAP ウィザードは Web クライアントの LOCATION パラメータを指定しません。これは一般的なケースではありません。このシナリオでは、Web クライアント・クラスを編集して LOCATION パラメータを含めるようにします。例えば以下のようになります。

### Class Member

```
Parameter LOCATION = "http://localhost:52773/csp/gsop/GSOP.AddComplexWS.cls";
```

または、Web クライアント・インスタンスの Location プロパティを、このトピックで後から示すように指定します。

Web クライアント・クラスの他のパラメータを調整して、その他の変更を加える必要がある場合があります。詳細は、"InterSystems IRIS での Web クライアントの調整" を参照してください。

# 4.5 生成された Web クライアント・クラスの使用法

前述のセクションで説明したように、通常は、InterSystems IRIS Web クライアント・クラスを生成した後で、生成されたクラスを編集することはありません。その代わりに、その Web クライアントのインスタンスを作成するコードと、クライアント側のエラー処理を提供するコードを記述します。このコードでは、以下の処理を実行します。

1. Web クライアント・クラスのインスタンスを作成します。
2. インスタンスのプロパティを設定します。ここでは、以下のような項目を制御できます。
   - Web クライアントのエンドポイント (Web クライアントで使用する Web サービスの URL)。制御するには、Location プロパティを設定します。これは、Web クライアント・クラスの LOCATION パラメータをオーバーライドします。
   - プロキシ・サーバを指定する設定。
   - HTTP 基本認証を制御する設定。

   次のセクションと "InterSystems IRIS での Web クライアントの調整" を参照してください。
3. 必要に応じて、Web クライアントのメソッドを呼び出します。
4. クライアント側のエラー処理を実行します。"SOAP フォルトの処理" を参照してください。
5. このトピックで後述するように、オプションで、Web クライアントが受け取る HTTP 応答を検証します。

ターミナルのセッションからの簡単な例を以下に示します。

```
GSOAP>set client=##class(Proxies.CustomerLookupServiceSoap).%New()

GSOAP>set resp=client.GetCustomerInfo("137")

GSOAP>w resp

11@Proxies.CustomerResponse

GSOAP>w resp.Name
Smith,Maria
```

## 4.5.1 例 1 : ラップされたメッセージを使用するクライアントの使用法

この例では、ラップされたメッセージを使用する Web クライアントのラッパ・クラスを作成します。前に示した `GSOAPClient.AddComplex.AddComplexSoap` の例を使用するために、次のようなクラスを作成します。

### Class Definition

```
Class GSOAPClient.AddComplex.UseClient Extends %RegisteredObject
{

ClassMethod Add(arg1 As ComplexNumber, arg2 As ComplexNumber) As ComplexNumber
{
    Set client=##class(AddComplexSoap).%New()
    //uncomment the following to enable tracing
    //set client.Location="http://localhost:8080/csp/gsop/GSOP.AddComplexWS.cls"
    Set ans=client.Add(arg1,arg2)
    Quit ans
}

}
```

クライアント・アプリケーションは、Web メソッドを実行するために、このメソッドを呼び出します。

### 4.5.2 例 2 : ラップされていないメッセージを使用するクライアントの使用法

この例では、ラップされていないメッセージを使用する Web クライアントのラッパ・クラスを作成します。前に示した GSOAPClient.AddComplex.AddComplexSoap の例を使用するために、次のようなクラスを作成します。

#### Class Definition

```
Class GSOAPClient.AddComplexUnwrapped.UseClient Extends %RegisteredObject
{

ClassMethod Add(arg1 As GSOAPClient.AddComplexUnwrapped.s0.ComplexNumber,
arg2 As GSOAPClient.AddComplexUnwrapped.s0.ComplexNumber)
As GSOAPClient.AddComplexUnwrapped.s0.ComplexNumber
{
    //create the Add message
    Set addmessage=##class(GSOAPClient.AddComplexUnwrapped.s0.Add).%New()
    Set addmessage.a = arg1
    Set addmessage.b = arg2

    Set client=##class(AddComplexSoap).%New()

    //send the Add message to client and get response
    Set addresponse=client.Add(addmessage)

    //get the result from the response message
    Set ans=addresponse.AddResult

    Quit ans

}

}
```

このメソッドには、普通に想定されるシグニチャがあります。つまり、このメソッドは 2 つの複素数を受け取り、1 つの複素数を返します。このメソッドは Web クライアントが必要とするメッセージを作成します。このメッセージの要素は 2 つの複素数です。

サンプル・コードを見るとわかるように、ラップされていないメッセージを Web クライアントで使用する場合は、ユーザにわかりやすい形式の引数を Web クライアントで使用するメッセージに変換する必要があるので、作成するコード量が若干多くなります。

## 4.6 Web クライアント・インスタンスのプロパティの調整

Web クライアント・クラスのインスタンスを使用するとき、そのインスタンスのプロパティを指定して動作を制御できます。このセクションでは、一般的に設定することが多いプロパティとその既定値について説明します。

### 4.6.1 Web クライアントのエンドポイントの変更

SOAP ウィザードでは、Web クライアントの LOCATION パラメータを設定することにより、Web クライアントにエンドポイントが自動的に設定されます。既定では、このパラメータは、通信する Web サービスの URL と同じに設定されます。

この設定を変更するには、Web クライアント・インスタンスの Location プロパティを設定します。Location が NULL の場合に、LOCATION パラメータが使用されます。

一般的な使用法の 1 つとして、トレースを有効にするために別のポートを使用するように Location プロパティを設定します。例えば、生成された Web クライアント・クラスでエンドポイントが次のように定義されているとします。

#### Class Member

```
Parameter LOCATION = "http://localhost:52773/csp/gsop/GSOP.AddComplexWS.cls";
```

このクライアントを使用するときに、次の行を追加します。

ObjectScript

```
Set client.Location="http://localhost:8080/csp/gsop/GSOP.AddComplexWS.cls"
```

注釈　WSDL が Web サービスの場所を指定しない場合、SOAP ウィザードは Web クライアントの LOCATION パラメータを指定しません。これは一般的なケースではありません。このシナリオでは、Web クライアント・クラスを編集して LOCATION パラメータを含めるようにするか、または Web クライアント・インスタンスの Location プロパティをここに示すように指定します。

## 4.6.2 SSL を使用するようにクライアントを構成する方法

Web クライアントのエンドポイントで HTTPS プロトコルを使用する場合は、SSL を使用するようにその Web クライアントを構成する必要があります。具体的には以下のとおりです。

- まだ作成していない場合は、管理ポータルを使用して、必要な SSL 接続の詳細を格納する SSL/TLS 構成を作成します。詳細は、"TLS ガイド" の "構成について" を参照してください。
- Web クライアントの SSLConfiguration プロパティを SSL/TLS 構成名に等しくなるように設定します。

プロキシ・サーバ経由でクライアントを接続している場合は、Web クライアントで HttpProxySSLConnect プロパティを 1 に設定する必要もあります。プロキシ・サーバを使用するように InterSystems IRIS Web クライアントを構成する方法については、"Web クライアントの調整" を参照してください。

## 4.6.3 SOAP バージョンの指定

SOAP ウィザードでは、Web サービスの WSDL で使用している SOAP バージョンに基づいて、要求メッセージで使用する SOAP バージョンが自動的に指定されます。具体的には、SOAPVERSION パラメータが設定されます。

この設定を変更するには、Web クライアント・インスタンスの SoapVersion プロパティを設定します。以下の値のいずれかを使用します。

- "" — クライアントは SOAP 1.1 メッセージを送信します。
- "1.1" — クライアントは SOAP 1.1 メッセージを送信します。
- "1.2" — クライアントは SOAP 1.2 メッセージを送信します。

SoapVersion が NULL の場合に、SOAPVERSION パラメータが使用されます。

## 4.6.4 その他の調整

Web クライアント・インスタンスの他のプロパティを設定して、その他の変更を加える必要がある場合があります。詳細は、"InterSystems IRIS での Web クライアントの調整" を参照してください。

# 4.7 HTTP 応答の使用法

既定では、Web クライアント・メソッドを呼び出す場合、HTTP 経由で呼び出されます。HTTP 応答は、Web クライアント・インスタンスの HttpResponse プロパティとして使用できるようになります。このプロパティは %Net.HttpResponse のインスタンスであり、以下のようなプロパティが含まれています。

- Headers には、HTTP 応答のヘッダが含まれます。

- Data は、HTTP 応答のデータを格納する InterSystems IRIS 多次元配列です。
- StatusCode、StatusLine、および ReasonPhrase はステータス情報を提供します。

詳細は、"インターネット・ユーティリティの使用法" を参照するか、%Net.HttpResponse のクラス・ドキュメントを参照してください。

# 5

# SOAP フォルトの処理

このトピックでは、Web サービス内および Web クライアント内でフォルトを処理する方法を説明します。

エラー処理の詳細は、"ObjectScript の使用法" のエラーに関するトピックを参照してください。

SOAPPREFIX パラメータは、すべての SOAP フォルトで使用する接頭語に作用します。"InterSystems IRIS での Web サービスの調整" の "SOAP エンベロープ接頭語の指定" を参照してください。

## 5.1 Web サービスでの既定のフォルト処理

既定では、InterSystems IRIS Web サービスでエラーが検出されると、フォルトを含む標準の SOAP メッセージが返されます。以下に例を示します (SOAP 1.1 の場合)。この例では、SOAP エンベロープは省略されています。

```
<SOAP-ENV:Body>
 <SOAP-ENV:Fault>
  <faultcode>SOAP-ENV:Server</faultcode>
  <faultstring>Server Application Error</faultstring>
  <detail>
     <error xmlns='http://www.myapp.org' >
       <text>ERROR #5002: ObjectScript error: <DIVIDE>zDivide^FaultEx.Service.1</text>
     </error>
  </detail>
</SOAP-ENV:Fault>
</SOAP-ENV:Body>
```

## 5.2 InterSystems IRIS Web サービスでカスタムの SOAP フォルトを返す方法

カスタムの SOAP フォルトを作成して返すには、エラーをトラップする適切なコードの領域内で以下の手順を実行します。

1. 適切な情報を含むフォルト・オブジェクトを作成します。そのためには、Web サービスのメソッドのうち、MakeFault()、MakeFault12()、MakeSecurityFault()、MakeStatusFault() のいずれかを呼び出します。これらは以下のサブセクションで説明しています。

   または、手動でフォルト・オブジェクトを作成することもできます。その手順については、このトピックで後述します。

2. フォルト・オブジェクトを引数として渡すことにより、Web サービスの ReturnFault() メソッドを呼び出します。ReturnFault() が呼び出し元に戻ることはありません。このメソッドは、フォルトを送信し、Web メソッドの処理を終了するだけです。

以下に例を示します。

Class Member

```
Method Divide(arg1 As %Numeric, arg2 As %Numeric) As %Numeric [ WebMethod ]
{
  Try {
    Set ans=arg1 / arg2
    } Catch {

        //<detail> element must contain element(s) or whitespace
        //specify this element by passing valid XML as string argument to MakeFault()
        set mydetail="<mymessage>Division error detail</mymessage>"

        set fault=..MakeFault($$$FAULTServer,"Division error",mydetail)

        // ReturnFault must be called to send the fault to the client.
        // ReturnFault will not return here.
        Do ..ReturnFault(fault)
      }
  Quit ans
}
```

## 5.2.1 フォルトを作成するメソッド

MakeFault()

```
classmethod MakeFault(pFaultCode As %String,
                      pFaultString As %String,
                      pDetail As %String = "",
                      pFaultActor As %String = "") as %SOAP.Fault
```

SOAP 1.1 に適したフォルト・オブジェクトを返します。以下はその説明です。

- pFaultCode は、SOAP フォルトの `<faultcode>` 要素内で使用されます。このプロパティを、"SOAP フォルト・コードのマクロ" に記載されている SOAP 1.1 マクロのいずれかに設定します。
- pFaultString は、SOAP フォルトの `<faultstring>` 要素内で使用されます。ユーザが確認できるように、フォルトの理由を示す文字列を指定します。
- pDetail は、SOAP フォルトの `<detail>` 要素内で使用されます。これを使用して、フォルトの原因に関する情報を指定します。

  指定した場合は、この引数には、`<detail>` 要素内で使用できる有効な XML を含む文字列を指定する必要があります。InterSystems IRIS では、指定した文字列が有効かどうかの確認は行われず、このチェックはアプリケーションの担当です。
- pFaultActor は、フォルト発生の原因となった SOAP メッセージ・パスの SOAP ノードの URI を指定します。

  SOAP メッセージが SOAP メッセージ・パスの複数のノードを通過する場合に、エラーの原因となったノードをクライアントで確認する必要があるときには、これが役立ちます。この高度な使用法については、このドキュメントでは説明していません。

MakeFault12()

```
classmethod MakeFault12(pFaultCode As %String,
                        pFaultString As %String,
                        pDetail As %String = "",
                        pFaultActor As %String = "") as %SOAP.Fault
```

SOAP 1.2 に適したフォルト・オブジェクトを返します。このメソッドは、Web サービスの SoapVersion プロパティが `"1.2"` の場合にのみ使用します。InterSystems IRIS で要求メッセージの SOAP バージョンがどのように扱われるかの詳細は、"Web サービスのパラメータの指定" を参照してください。

また、これらの引数の詳細は、MakeFault() に関する説明を参照してください。

MakeSecurityFault()

```
classmethod MakeSecurityFault(pFaultCode As %String,
                              securityNamespace As %String) as %SOAP.Fault
```

セキュリティの失敗に適したフォルト・オブジェクトを返します。FaultCode を、"FailedAuthentication"、"FailedCheck"、"InvalidSecurity"、"InvalidSecurityToken"、"SecurityTokenUnavailable"、"UnsupportedAlgorithm"、"UnsupportedSecurityToken" のいずれかに指定します。

このセキュリティ・フォルトのネームスペースは、SecurityNamespace プロパティに含まれています。

MakeStatusFault()

```
classmethod MakeStatusFault(pFaultCode As %String,
                            pFaultString As %String,
                            pStatus As %Status = "",
                            pFaultActor As %String = "") as %SOAP.Fault
```

%Status オブジェクトの値に基づいてフォルト・オブジェクトを返します。

pStatus は、使用する %Status オブジェクトです。

その他の引数の詳細は、MakeFault() に関する説明を参照してください。

## 5.2.2 SOAP フォルト・コードのマクロ

以下のテーブルに示すように、SOAP インクルード・ファイル (%soap.inc) では、一部の標準の SOAP フォルト・コードのマクロが定義されます。これらのマクロを使用して、SOAP フォルト・コードを指定できます。このテーブルは、各マクロが適用される SOAP のバージョン (複数可) を示しています。

テーブル 5-1: SOAP フォルト・コードの ObjectScript マクロ

| マクロ | SOAP バージョン | このマクロを使用する場合 |
|---|---|---|
| $$$FAULTVersionMismatch | 1.1 および 1.2 | Web サービスが、必要なエンベロープ要素情報項目ではなく、無効な要素情報項目を含む SOAP メッセージを受け取った場合<br>ネームスペースまたはローカル名のいずれかが一致しない、不一致が発生した場合 |
| $$$FAULTMustUnderstand | 1.1 および 1.2 | Web サービスが、mustUnderstand="true" というマークが付いた予期しない要素を含む SOAP メッセージを受け取った場合 |
| $$$FAULTServer | 1.1 | サーバ側で他のエラーが発生した場合 |
| $$$FAULTClient | 1.1 | クライアントが、不完全なまたは誤った要求をした場合 |
| $$$FAULTDataEncodingUnknown | 1.2 | 受信者にとって不明なデータ・エンコードで引数がエンコードされている場合 |
| $$$FAULTSender | 1.2 | 送信者が不完全な、誤った、またはサポートされていない要求をした場合 |
| $$$FAULTReceiver | 1.2 | 何らかの一時的な状態 (メモリ不足など) のために、受信者がメッセージを処理できない場合 |

# 5.3 フォルト・オブジェクトの手動作成

前のセクションの手順よりも詳細な管理を行う必要がある場合は、以下の手順でカスタムの SOAP フォルトを作成して返すことができます。

1. フォルト・オブジェクトを手動で作成します。

   そのためには、**%SOAP.Fault** (SOAP 1.1 の場合) または **%SOAP.Fault12** (SOAP 1.2 の場合) のインスタンスを作成し、そのプロパティを設定します。手順は以降のセクションを参照してください。

   注釈　すべての場合に **%SOAP.Fault** を使用できます。Web サービスが SOAP 1.2 要求を受信し、フォルトを返す必要がある場合は、Web サービスによってそのフォルトが SOAP 1.2 形式に自動的に変換されます。

2. フォルト・オブジェクトを引数として渡すことにより、Web サービスの ReturnFault() メソッドを呼び出します。ReturnFault()) が呼び出し元に戻ることはありません。このメソッドは、フォルトを送信し、Web メソッドの処理を終了するだけです。

## 5.3.1 SOAP 1.1 フォルト

このセクションでは、SOAP 1.1 フォルトを表す **%SOAP.Fault** に関する情報を提供します。このセクションでは、以下の項目について説明します。

- SOAP 1.1 フォルトの例
- SOAP 1.1 フォルトを表す %SOAP.Fault に関する情報

### 5.3.1.1 SOAP フォルトの例

参照のために、SOAP エンベロープを含めた SOAP 1.1 フォルトの例を以下に示します。

XML

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:s='http://www.w3.org/2001/XMLSchema'
xmlns:flt="http://myfault.org" >
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Server</faultcode>
      <faultstring>Division error</faultstring>
      <detail><mymessage>Division error detail</mymessage></detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

### 5.3.1.2 %SOAP.Fault のプロパティ

InterSystems IRIS では、SOAP 1.1 フォルトを **%SOAP.Fault** のインスタンスとして表します。これには、以下のプロパティがあります。

detail

SOAP フォルトの `<detail>` 要素内で使用されます。これを使用して、フォルトの原因に関する情報を指定します。

指定した場合は、この引数には、`<detail>` 要素内で使用できる有効な XML を含む文字列を指定する必要があります。InterSystems IRIS では、指定した文字列が有効かどうかの確認は行われず、このチェックはアプリケーションの担当です。

faultcode

SOAP フォルトの <faultcode> 要素内で使用されます。このプロパティを、"SOAP フォルト・コードのマクロ" に記載されている SOAP 1.1 マクロのいずれかに設定します。

faultstring

SOAP フォルトの <faultstring> 要素内で使用されます。ユーザが確認できるように、フォルトの理由を示す文字列を指定します。

faultactor

フォルト発生の原因となった SOAP メッセージ・パスの SOAP ノードの URI を指定します。

SOAP メッセージが SOAP メッセージ・パスの複数のノードを通過する場合に、エラーの原因となったノードをクライアントで確認する必要があるときには、これが役立ちます。この高度な使用法については、このドキュメントでは説明していません。

faultPrefixDefinition

SOAP フォルトのエンベロープに追加されるネームスペース接頭語宣言を指定します。以下の形式の値を使用します。

```
xmlns:prefix="namespace"
```

prefix は接頭語で、namespace はネームスペース URI です。

以下はその例です。

ObjectScript

```
 set fault.faultPrefixDefinition = "xmlns:FLT=""http://myfault.com"""
```

%SOAP.Fault クラスには、フォルト・オブジェクトを文字列として返す AsString() メソッドもあります。

## 5.3.2 SOAP 1.2 フォルト

このセクションでは、SOAP 1.2 フォルトを表す %SOAP.Fault12 および関連するクラスに関する情報を提供します。このセクションでは、以下の項目について説明します。

- SOAP 1.2 フォルトの例
- SOAP 1.2 フォルトを表すメイン・クラスである %SOAP.Fault12
- ヘルパー・クラス %SOAP.Fault12.Code
- 別のヘルパー・クラス %SOAP.Fault12.Text

### 5.3.2.1 SOAP フォルトの例

参照のために、SOAP エンベロープを含めた SOAP 1.2 フォルトの例を以下に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:flt="http://myfault.org">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <SOAP-ENV:Code>
        <SOAP-ENV:Value>SOAP-ENV:Receiver</SOAP-ENV:Value>
      </SOAP-ENV:Code>
      <SOAP-ENV:Reason>
        <SOAP-ENV:Text xml:lang="en">Division error</SOAP-ENV:Text>
        <SOAP-ENV:Text xml:lang="it">Errore di applicazione</SOAP-ENV:Text>
        <SOAP-ENV:Text xml:lang="es">Error del uso</SOAP-ENV:Text>
      </SOAP-ENV:Reason>
      <SOAP-ENV:Detail><mymessage>Division error detail</mymessage></SOAP-ENV:Detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

### 5.3.2.2 %SOAP.Fault12 のプロパティ

%SOAP.Fault12 クラスは SOAP 1.2 フォルトを表します。このクラスには、以下のプロパティがあります。

#### Code

次のセクションで説明している %SOAP.Fault12.Code のインスタンス。

#### Detail

SOAP フォルトの <detail> 要素内で使用されます。これを使用して、フォルトの原因に関する情報を指定します。

指定した場合は、この引数には、<detail> 要素内で使用できる有効な XML を含む文字列を指定する必要があります。InterSystems IRIS では、指定した文字列が有効かどうかの確認は行われず、このチェックはアプリケーションの担当です。

#### Node

フォルト発生の原因となった SOAP メッセージ・パスの SOAP ノードの URI を指定します。宛先ノードの場合はオプションです。

SOAP メッセージが SOAP メッセージ・パスの複数のノードを通過する場合に、エラーの原因となったノードをクライアントで確認する必要があるときには、これが役立ちます。SOAP のこの高度な使用法についての説明は、このドキュメントの対象ではありません。

#### Reason

次のセクションで説明している %SOAP.Fault12.Text のインスタンスのリスト。各インスタンスには、理由文字列、および理由文字列の言語または局所性を示す言語コードが含まれています。これらは、<Reason> 要素内で使用されます。

#### Role

ノードが動作していた役割。Node に関する前述の備考を参照してください。

#### faultPrefixDefinition

SOAP フォルトのエンベロープに追加されるネームスペース接頭語宣言を指定します。以下の形式の値を使用します。

```
xmlns:prefix="namespace"
```

prefix は接頭語で、namespace はネームスペース URI です。

以下はその例です。

ObjectScript

```
set fault.faultPrefixDefinition = "xmlns:FLT=""http://myfault.com"""
```

%SOAP.Fault12 クラスには、フォルト・オブジェクトを文字列として返す AsString() メソッドもあります。

### 5.3.2.3 %SOAP.Fault12.Code のプロパティ

%SOAP.Fault12 のインスタンスの Code プロパティの値として %SOAP.Fault12.Code を使用します。%SOAP.Fault12.Code クラスには以下のプロパティがあります。

Subcode

オプションのサブコードです。

Value

指定する値は、サブコードを指定したかどうかによって異なります。

- サブコードを使用した場合、Value には qname と指定します。
- サブコードを使用していない場合、Value には、"SOAP フォルト・コードのマクロ" に記載された SOAP 1.2 マクロのいずれかを指定します。

### 5.3.2.4 %SOAP.Fault12.Text のプロパティ

%SOAP.Fault12 のインスタンスの Reason プロパティのリスト要素として、%SOAP.Fault12.Text を使用します。%SOAP.Fault12.Text クラスには以下のプロパティがあります。

Text

ユーザが確認できるように、フォルトの理由を示す文字列。

lang

フォルト・テキストが表されている言語または局所性に対応するコード。詳細は、W3 の Web サイト (https://www.w3.org/) を参照してください。

## 5.4 フォルト発生時の WS-Addressing ヘッダ要素の追加

フォルトの発生時に WS-Addressing ヘッダ要素を追加するように InterSystems IRIS Web サービスを設定できます。そのためには、Web サービスのフォルト処理に次の手順を追加します。

1. フォルト発生時に使用するフォルトの宛先とフォルトのアクションを選択します。
2. これらを引数として使用して、%SOAP.Addressing.Properties クラスの GetDefaultResponseProperties() メソッドを呼び出します。このメソッドからは、一般に必要となる値が設定された %SOAP.Addressing.Properties のインスタンスが返ります。
3. 必要に応じて、%SOAP.Addressing.Properties のインスタンスのその他のプロパティを設定します。

   詳細は、%SOAP.Addressing.Properties のクラス・ドキュメントを参照してください。

4. Web サービスの FaultAddressing プロパティを %SOAP.Addressing.Properties のインスタンスと等しくなるように設定します。

# 5.5 フォルト発生時のその他のヘッダ要素の追加

フォルト発生時にカスタムのヘッダ要素を追加するように InterSystems IRIS Web サービスを設定することもできます。このカスタムのヘッダ要素は、前のセクションで説明したオプションに追加するオプションとすることができるほか、それらのオプションに代わるものとすることもできます。以下はその方法です。

1. %SOAP.Header のサブクラスを作成します。このサブクラスで、追加データを設定したプロパティを追加します。

   "カスタム・ヘッダ要素の追加と使用" を参照してください。

2. このトピックで既に取り上げた Web サービスのフォルト処理に次の手順を追加します。

   a. ヘッダ・サブクラスのインスタンスを作成します。

      注釈　このようなクラス名になっていても、このオブジェクトはヘッダ全体を表しているわけではなく、実際には SOAP ヘッダ要素です。個々の SOAP メッセージには、複数の要素で構成したヘッダが 1 つあります。

   b. 必要に応じて、そのプロパティを設定します。

   c. Web サービスの FaultHeaders 配列プロパティにこのヘッダ要素を挿入します。そのためには、そのプロパティの SetAt() を呼び出します。指定したキーはヘッダのメイン要素名として使用されます。

例えば、次のようなカスタムのヘッダ・クラスがあるとします。

### Class Definition

```
Class Fault.CustomHeader Extends %SOAP.Header
{

Parameter XMLTYPE = "CustomHeaderElement";

Property SubElement1 As %String;

Property SubElement2 As %String;

Property SubElement3 As %String;

}
```

前述の Web メソッドを以下のように変更します。

### Class Member

```
Method DivideAlt(arg1 As %Numeric, arg2 As %Numeric) As %Numeric [ WebMethod ]
{
  Try {
    Set ans=arg1 / arg2
    } Catch {

        //<detail> element must contain element(s) or whitespace
        //specify this element by passing valid XML as string argument to MakeFault()
        set mydetail="<mymessage>Division error detail</mymessage>"

        set fault=..MakeFault($$$FAULTServer,"Division error",mydetail)

        //Set fault header
        Set header=##class(CustomHeader).%New()
        Set header.SubElement1="custom fault header element"
        Set header.SubElement2="another custom fault header element"
        Set header.SubElement3="yet another custom fault header element"
```

```
        Do ..FaultHeaders.SetAt(header,"CustomFaultElement")

        // ReturnFault must be called to send the fault to the client.
        // ReturnFault will not return here.
        Do ..ReturnFault(fault)
      }
  Quit ans
}
```

Web クライアントから Divide() Web メソッドを呼び出して、分母として 0 を使用すると、Web サービスは次のように応答します。

```
<?xml version='1.0' encoding='UTF-8' standalone='no' ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:s='http://www.w3.org/2001/XMLSchema' xmlns:flt="http://myfault.org" >
  <SOAP-ENV:Header>
<CustomHeaderElement xmlns:hdr="http://www.mynamespace.org">
    <SubElement1>custom fault header element</SubElement1>
    <SubElement2>another custom fault header element</SubElement2>
    <SubElement3>yet another custom fault header element</SubElement3>
</CustomHeaderElement>
</SOAP-ENV:Header>
  <SOAP-ENV:Body>
...
```

ここでは、見やすくするために改行を追加してあります。

# 5.6 InterSystems IRIS Web クライアントでの SOAP フォルトおよびその他のエラーの処理

InterSystems IRIS Web クライアントでは、TRY-CATCH メカニズム、または従来の $ZTRAP メカニズムを使用できます。

どちらの場合も、InterSystems IRIS Web クライアントがエラーを受信すると、特殊変数 $ZERROR と %objlasterror が InterSystems IRIS で次のように設定されます。

- エラーが SOAP フォルトの場合、$ZERROR の値は <ZSOAP> で始まります。また %objlasterror には、受信した SOAP フォルトから形成されるステータス・エラーが格納されます。

  また、クライアント・インスタンスには SoapFault という名前のプロパティがあります。これは、Web サービスで使用している SOAP バージョンに応じて、%SOAP.Fault または %SOAP.Fault12 のインスタンスになります。このプロパティでこの情報を使用できます。%SOAP.Fault および %SOAP.Fault12 の詳細は、前のセクションを参照してください。
- エラーが SOAP フォルトではない場合、標準のエラー処理を行う必要があります (通常は $ZERROR を使用)。処理方法はユーザが指定します。

## 5.6.1 例 1 : Try-Catch

以下のメソッドでは、TRY-CATCH を使用しています。

Class Member

```
ClassMethod Divide(arg1 As %Numeric, arg2 As %Numeric) As %Numeric
{
    Set $ZERROR=""
    Set client=##class(FaultClient.DivideSoap).%New()

    Try {
        Set ans=client.Divide(arg1,arg2)
        }
    Catch {
        If $ZERROR["<ZSOAP>" {
            Set ans=%objlasterror
            }
            Else {
            Set ans=$$$ERROR($$$ObjectScriptError,$ZERROR)
        }
    }

    Quit ans
}
```

このメソッドでは、%systemInclude インクルード・ファイルで定義されているシステム・マクロを使用しているため、このメソッドを含むクラスの先頭は以下のようになります。

```
Include %systemInclude
```

## 5.6.2 例 2 : $ZTRAP

次の例では、従来の $ZTRAP メカニズムを使用しています。この場合、InterSystems IRIS Web クライアントがエラーを受信すると、特殊変数 $ZTRAP が示すラベル位置に制御が移動します (そのラベルが定義されている場合)。

Class Member

```
ClassMethod DivideWithZTRAP(arg1 As %Numeric = 1, arg2 As %Numeric = 2) As %Numeric
{
    Set $ZERROR=""
    Set $ZTRAP="ERRORTRAP"
    Set client=##class(FaultClient.DivideSoap).%New()
    Set ans=client.Divide(arg1,arg2)
    Quit ans

    //control goes here in case of error
ERRORTRAP
    if $ZERROR["<ZSOAP>"
    {
         quit client.SoapFault.Detail
        }
    else
    {
        quit %objlasterror
       }
}
```

## 5.6.3 SSL ハンドシェイクのエラー

InterSystems IRIS Web クライアントが SSL 接続を使用しているときに SSL ハンドシェイク・エラーが発生した場合、クライアントの SSLError プロパティにその SSL エラーを説明するテキストが格納されます。

# 6

# 添付での MTOM の使用法

SOAP 要求と応答のメッセージに添付を追加できます。この実行方法としては、MTOM (Message Transmission Optimization Mechanism) に対する InterSystems IRIS サポートの使用が推奨されます。

ポリシー内で MTOM の使用について指定することもできます。"Web サービスの保護" を参照してください。

また、パッケージ化の後で gzip を使用してメッセージを圧縮するように InterSystems IRIS Web サービスと Web クライアントを構成することもできます。"InterSystems IRIS での Web サービスの調整" と "InterSystems IRIS での Web クライアントの調整" を参照してください。

## 6.1 添付および SOAP メッセージのパッケージ化

一般に添付は、バイナリ・データの伝送に使用されます。InterSystems IRIS の SOAP サポートでは、SOAP メッセージをパッケージ化する方法が 3 種類用意されています。詳細を説明する前に、このパッケージ化方法を確認しておきます。

- すべてのパーツをインラインにしてメッセージをパッケージ化します (添付なし)。バイナリ・データのすべてに Base 64 のエンコードを使用します。

  Web サービスが MTOM 要求を受信する場合 (サービスが MTOM 応答で返信する) を除き、これは InterSystems IRIS Web サービスおよび Web クライアントの既定の動作です。

- MTOM (Message Transmission Optimization Mechanism) 仕様に従ってメッセージをパッケージ化します。この場合、すべてをインライン化するメッセージと比較して若干コンパクトになります。現在、SOAP メッセージについてはこちらの方法が推奨されます。

  この方法を採用した場合、状況に応じて SOAP メッセージが自動的にパッケージ化されます。つまり、MIME パーツが必要に応じて作成され、ユーザの操作を必要とせずにメッセージに追加されます。

  また、InterSystems IRIS で MTOM パッケージを作成する際の既定として、添付を使用してバイナリ・ストリームが出力され、バイナリ文字列 (**%Binary** または **%xsd.base64Binary**) がインラインで出力されます。この動作は制御が可能です。

  MTOM の仕様へのリンクは、"SOAP 標準" を参照してください。

- SOAP With Attachments 仕様に従ってメッセージをパッケージ化します。この場合、すべてをインライン化するメッセージと比較して若干コンパクトになります。

  この方法を採用した場合は、MIME パーツを作成し、そのパーツにデータを入力し、必要に応じて MIME ヘッダを指定して、パーツを SOAP メッセージに添付するという手動作業が必要になります。一般に、この方法では MTOM を採用した場合より多くの作業が必要です。"SOAP With Attachments の使用法" を参照してください。

## 6.1.1 すべてのパーツをインライン化した SOAP メッセージ (既定)

SOAP メッセージをパッケージ化する既定の方法は、すべての要素をインライン・パーツとして組み込む方法です (つまり、添付はありません)。バイナリ・データもすべて Base 64 でエンコードされたデータとしてインラインに組み込まれます。以下に例を示します (読みやすくするため改行およびスペースが追加されています)。

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2008 21:57:50 GMT
Server: Apache
SET-COOKIE: CSPSESSIONID-SP-8080-UP-csp-gsoap-=003000010000248
guobl000000K7opwlDlY$XbvrGR1eYZsA--; path=/csp/gsop/;
CACHE-CONTROL: no-cache
EXPIRES: Thu, 29 Oct 1998 17:04:19 GMT
PRAGMA: no-cache
TRANSFER-ENCODING: chunked
Connection: close
Content-Type: text/xml; charset=UTF-8

1d7b
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:s='http://www.w3.org/2001/XMLSchema'>
<SOAP-ENV:Body>
  <DownloadResponse xmlns="http://www.filetransfer.org">
     <DownloadResult><Filename>sample.pdf</Filename>
     <IsBinary>true</IsBinary>
     <BinaryContents>
        [very long binary content not shown here]
     </BinaryContents></attachment></Upload>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

このパッケージでは MIME が使用されず、また、メッセージ範囲もないことがわかります。

## 6.1.2 MTOM パッケージ化を使用した SOAP メッセージ

SOAP メッセージをパッケージ化する 2 つ目の方法は、MTOM (Message Transmission Optimization Mechanism) の仕様に従って MIME パーツを使用する方法です。バイナリ・データは、独立した MIME パートに配置され、Base 64 エンコードは使用されません。SOAP メッセージには、必要に応じて独立したパーツへの参照が組み込まれます。以下に例を示します (読みやすくするため改行およびスペースが追加されています)。

```
HTTP/1.1 200 OK
Date: Wed, 19 Nov 2008 21:54:57 GMT
Server: Apache
SET-COOKIE: CSPSESSIONID-SP-8080-UP-csp-gsoap-=003000010
000247guhlx000000NW1KN5UtWg$CWY38$bbTOQ--; path=/csp/gsop/;
CACHE-CONTROL: no-cache
EXPIRES: Thu, 29 Oct 1998 17:04:19 GMT
MIME-VERSION: 1.0
PRAGMA: no-cache
TRANSFER-ENCODING: chunked
Connection: close
Content-Type: multipart/related; type="application/xop+xml";
boundary=--boundary388.5294117647058824932.470588235294118--;
start="<0.B1150656.EC8A.4B5A.8835.A932E318190B>"; start-info="text/xml"

1ddb
----boundary388.5294117647058824932.470588235294118--
Content-Type: application/xop+xml; type="text/xml"; charset="UTF-8"
Content-Transfer-Encoding: 8bit
Content-Id: <0.B1150656.EC8A.4B5A.8835.A932E318190B>

<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:s='http://www.w3.org/2001/XMLSchema'>
<SOAP-ENV:Body>
<DownloadResponse xmlns="http://www.filetransfer.org">
<DownloadResult>
   <Filename>sample.pdf</Filename>
   <IsBinary>true</IsBinary>
   <BinaryContents>
      <xop:Include href="cid:1.B1150656.EC8A.4B5A.8835.A932E318190B"
```

```
              xmlns:xop="http://www.w3.org/2004/08/xop/include"/>
   </BinaryContents></DownloadResult></DownloadResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
----boundary388.5294117647058824932.470588235294118--
Content-Id: <1.B1150656.EC8A.4B5A.8835.A932E318190B>
Content-Transfer-Encoding: binary
CONTENT-TYPE: application/octet-stream


[very long binary content not shown here]
```

既定のパッケージと比べて次のような違いがあります。

- メッセージには MIME パーツがあり、したがって境界が含まれます。
- MIME パーツには `Content-ID` 属性があります。
- SOAP 本文では、`BinaryContents` 要素はそのコンテンツ ID への参照で構成されています。

## 6.1.3 SOAP With Attachments

SOAP メッセージをパッケージ化する 3 つ目の方法は、SOAP With Attachments 仕様を使用する方法です。この場合も MIME パーツが使用されますが、メッセージのパッケージ化方法は MTOM と少し異なります。以下に例を示します (読みやすくするため改行およびスペースが追加されています)。

```
HTTP/1.1 200 OK
Date: Mon, 09 Nov 2009 17:47:36 GMT
Server: Apache
SET-COOKIE: CSPSESSIONID-SP-8080-UP-csp-gsoap-=
000000010000213eMwn70000004swjTo4cGuInLMU1n7jaPg--; path=/csp/gsop/;
CACHE-CONTROL: no-cache
EXPIRES: Thu, 29 Oct 1998 17:04:19 GMT
MIME-VERSION: 1.0
PRAGMA: no-cache
TRANSFER-ENCODING: chunked
Connection: close
Content-Type: multipart/related; type="text/xml";
 boundary=--boundary2629.3529411764705883531.411764705882353--

1ca2
----boundary2629.3529411764705883531.411764705882353--
Content-Type: text/xml; charset="UTF-8"
Content-Transfer-Encoding: 8bit

<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:s='http://www.w3.org/2001/XMLSchema'>
  <SOAP-ENV:Body><DownloadBinaryResponse xmlns="http://www.filetransfer.org">
<DownloadBinaryResult>MQ==</DownloadBinaryResult></DownloadBinaryResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
----boundary2629.3529411764705883531.411764705882353--
Content-Transfer-Encoding: binary
Content-Type: application/octet-stream

%PDF-1.4
%âãÏÓ
86 0 obj
<</Length 87 0 R
/Filter /FlateDecode
>>
stream
[stream not shown]
```

MTOM と同じように境界文字列があり、添付は MIME パートです。一方、MTOM と異なる点として、その MIME パートにはコンテンツ ID がなく、SOAP の本文にもその MIME パートの参照が存在しません。

# 6.2 InterSystems IRIS Web サービスおよび Web クライアントの既定の動作

既定では、InterSystems IRIS Web サービスは次のように動作します。

- MTOM パッケージで要求を受け取ると、Web サービスはその応答を MTOM パッケージとして送信します。

  また、Web サービス・インスタンスの **IsMTOM** プロパティは、1 に設定されます。
- MTOM パッケージ以外の要求を受け取ると、Web サービスはその応答を MTOM パッケージ以外で送信します。

既定では、InterSystems IRIS Web クライアントは次のように動作します。

- 要求を MTOM パッケージとして送信しません。
- 応答が MTOM パッケージであるかどうかに関わらず、応答を処理します。

  応答が MTOM パッケージである場合、Web クライアント・インスタンスの **IsMTOM** プロパティは、1 に設定されます。応答が MTOM パッケージでない場合は、**IsMTOM** プロパティは変更されません。

# 6.3 応答を MTOM パッケージとして強制する

InterSystems IRIS Web サービスに対して、すべての応答を MTOM パッケージとして送信するように強制できます。これを実行するには、次のいずれかの操作を行います。

- InterSystems IRIS Web サービス・クラスで MTOMREQUIRED パラメータを 1 に設定します。
- InterSystems IRIS Web サービス・インスタンスで **MTOMRequired** プロパティを 1 に設定します。これは、Web メソッド内または **OnPreWebMethod()** コールバック内で実行できます。このコールバックの概要は、"Web サービスのコールバックのカスタマイズ" を参照してください。
- MTOM パッケージを送信するように、Web サービスのポリシー文を添付します。そのためには、Web サービス・クラスを参照する構成クラスを作成およびコンパイルし、このポリシー内で MTOM の使用を有効にします。"Web サービスの保護" を参照してください。

  このようなポリシー文を添付すると、MTOMREQUIRED の値は無視され、**MTOMRequired** は 1 に設定されます。

## 6.3.1 WSDL への影響

MTOMREQUIRED および **MTOMRequired** は、Web サービスの WSDL には何も影響しません。

MTOM を参照するポリシー文は、WSDL に影響します。ポリシー文を追加する場合、Web クライアントの再生成が必要です。InterSystems IRIS Web クライアントでは、クライアント・クラスを再生成する代わりに、MTOM ポリシー文をクライアントに添付するだけですみます。

# 6.4 要求を MTOM パッケージとして強制する

InterSystems IRIS Web クライアントに対して、すべての要求を MTOM パッケージとして送信するように強制できます。これを実行するには、以下のいずれかの操作を行います。

- InterSystems IRIS Web クライアント・クラスで MTOMREQUIRED パラメータを 1 に設定します。
- InterSystems IRIS Web クライアント・インスタンスで **MTOMRequired** プロパティを 1 に設定します。
- MTOM パッケージを送信するように、Web クライアントにポリシー文を添付します。そのためには、Web サービス・クライアントを参照する構成クラスを作成およびコンパイルし、このポリシー内で MTOM の使用を有効にします。"Web サービスの保護" を参照してください。

  このようなポリシー文を添付すると、MTOMREQUIRED の値は無視され、**MTOMRequired** は 1 に設定されます。

### 6.4.1 WSDL への影響

MTOMREQUIRED および **MTOMRequired** は、この Web クライアントによって使用される Web サービスの WSDL に何らかの変更があるとは想定していません。

MTOM を参照するポリシー文は、WSDL に影響します。つまり、Web サービスが要求した場合のみ、MTOM ポリシー文をクライアントに追加することになります。

## 6.5 MTOM パッケージ化の制御

既定では、InterSystems IRIS が MTOM パッケージを作成する際に以下の規則が使用されます。

- バイナリ文字列 (**%Binary** または **%xsd.base64Binary**) はインラインで出力されます。
- バイナリ・ストリームは添付を使用して出力されます。

この既定は、MTOM プロパティ・パラメータを使用して変更できます。

- 1 は、このプロパティが添付として出力されることを意味します。
- 0 は、このプロパティがインラインとして出力されることを意味します。

Web サービスまたは Web クライアントが MTOM を使用しない場合、MTOM プロパティ・パラメータは何にも影響しません。

また、このプロパティ・パラメータは Web サービスの WSDL には影響しません。

## 6.6 例

この例では、バイナリ・ファイルを受信して呼び出し元に返送する InterSystems IRIS Web サービスを示します。

対応する Web クライアントは、ハードコードされたファイル名を持つファイルを送信し、Web サービスから同じファイルを受信します。そして、そのファイルを新しい名前で保存して、それが正常に送信されたことを実証します。

### 6.6.1 Web サービス

Web サービスは、次のとおりです。

### Class Definition

```
/// Receive an attachment and send it back
Class MTOM.RoundTripWS Extends %SOAP.WebService
{

///  Name of the web service.
Parameter SERVICENAME = "RoundTrip";

///  SOAP namespace for the web service
Parameter NAMESPACE = "http://www.roundtrip.org";

///  Receive an attachment and send it back
Method ReceiveFile(attachment As %GlobalBinaryStream) As %GlobalBinaryStream [ WebMethod ]
{
  Set ..MTOMRequired=1
  Quit attachment
}

}
```

## 6.6.2 Web クライアント

生成される Web クライアント (`MTOMClient.RoundTripSoap`) には、メソッド `ReceiveFile()` が組み込まれます。これによって同じ名前の Web メソッドが起動されます。このメソッドは、最初は次のようになっています。

### Class Member

```
Method ReceiveFile(attachment As %xsd.base64Binary) As %xsd.base64Binary
[ Final, SoapBindingStyle = document, SoapBodyUse = literal, WebMethod ]
{
 Quit ..WebMethod("ReceiveFile").Invoke($this,"http://www.roundtrip.org/MTOM.RoundTripWS.ReceiveFile",

 .attachment)
}
```

送信するファイルが文字列長の制限を超える可能性があるため、メソッド・シグニチャを次のように調整します。

### Class Member

```
Method ReceiveFile(attachment As %GlobalBinaryStream) As %GlobalBinaryStream
[ Final, SoapBindingStyle = document, SoapBodyUse = literal, WebMethod ]
{
 Quit ..WebMethod("ReceiveFile").Invoke($this,"http://www.roundtrip.org/MTOM.RoundTripWS.ReceiveFile",

 .attachment)
}
```

既定では、Web クライアントで MTOM は不要です。つまり、MTOMREQUIRED パラメータは定義されていません。

このプロキシ・クライアントを使用するには、次のクラスを作成します。

### Class Definition

```
Include %systemInclude

Class MTOMClient.UseClient
{

/// For this example, hardcode what we are sending
ClassMethod SendFile() As %GlobalBinaryStream
{
  Set client=##class(MTOMClient.RoundTripSoap).%New()
  Set client.MTOMRequired=1

  //reset location to port 8080 to enable tracing
  Set client.Location="http://localhost:8080/csp/gsop/MTOM.RoundTripWS.cls"

  //create file
  Set filename="c:\sample.pdf"
  Set file=##class(%Library.FileBinaryStream).%New()
  Set file.Filename=filename

  //create %GlobalBinaryStream
```

```
  Set attachment=##class(%GlobalBinaryStream).%New()
  Do attachment.CopyFrom(file)

  //call the web service
  Set answer=client.ReceiveFile(attachment)

  //save the received file to prove we made the round trip successfully
  Set newfilename="c:\roundtrip"_$h_"sample.pdf"
  Set newfile=##class(%Library.FileBinaryStream).%New()
  Set newfile.Filename=newfilename
  Do newfile.CopyFromAndSave(answer)

  Quit answer
}

}
```

# 7

# SOAP With Attachments の使用法

InterSystems IRIS Web クライアントおよび Web サービスでは、前のトピックで説明した、InterSystems IRIS がサポートする MTOM ではなく、同じく InterSystems IRIS がサポートする SOAP With Attachments を使用して、SOAP メッセージに添付を追加したり、添付を使用したりできます。

この方法では、添付として使用される MIME パーツをコードから直接管理する必要があるため、必要な作業が多くなります。

SOAP With Attachments 規格の仕様は、"SOAP 標準" を参照してください。

## 7.1 添付の送信

SOAP With Attachments 規格に対する InterSystems IRIS サポートを使用する場合、添付の送信に以下の処理を使用します。

1. 添付を作成します。添付を作成する手順は以下のとおりです。
   a. 添付データを表すストリーム・オブジェクトを使用します。使用するクラスは、ストリーム・データの取得に必要なインタフェースによって異なります。例えば、ファイルのコンテンツをストリームに読み取るには %Library.FileCharacterStream を使用できます。
   b. %Net.MIMEPart のインスタンスである MIME パーツを作成します。
   c. 作成した MIME パーツで以下を実行します。
      - Body プロパティをストリーム・オブジェクトに設定します。または、%Net.MIMEPart インスタンスのリストである Parts プロパティを設定します。
      - SetHeader() メソッドを呼び出して、MIME パーツの `Content-Transfer-Encoding` ヘッダを設定します。これは、必ず送信するデータのタイプに合わせて設定してください。

2. Web サービスまたは Web クライアントに添付を追加します。指定の添付を追加するには、以下の手順で MIME パートを適切なプロパティに挿入します。
   - Web クライアントから添付を送信する場合は、Web クライアントの Attachments プロパティを更新します。
   - Web サービスから添付を送信する場合は、Web サービスの ResponseAttachments プロパティを更新します。

   これらのプロパティはそれぞれ、通常のリスト・インタフェース (SetAt() メソッド、Count() メソッド、GetAt() メソッドなど) を使用したリストです。
3. 添付のコンテンツが記述されるように、Web クライアントまたは Web サービスの適切なプロパティを更新します。

- ContentId
- ContentLocation

# 7.2 添付の使用

InterSystems IRIS Web サービスや Web クライアントで添付のある SOAP メッセージ (SOAP With Attchments 仕様に準拠) を受信した場合は、次のように処理されます。

- 添付が適切なプロパティに挿入されます。
  - Web サービスでは、着信した添付が Attachments プロパティに配置されます。
  - Web クライアントでは、着信した添付が ResponseAttachments プロパティに配置されます。

  これらのプロパティはそれぞれ、通常のリスト・インタフェース (SetAt() メソッド、Count() メソッド、GetAt() メソッドなど) を使用したリストです。リスト要素のそれぞれは、%Net.MIMEPart のインスタンスです。MIME パートには他の MIME パートを入れ子に組み込むことができます。添付の構造とコンテンツの決定は、コードで行う必要があります。
- ContentID プロパティと ContentLocation プロパティが更新され、着信 SOAP メッセージの `Content-ID` ヘッダおよび `Content-Location` ヘッダが反映されます。

Web サービスまたは Web クライアントはこれらのプロパティへのアクセスが可能で、これによって添付へのアクセスも可能になります。

# 7.3 例

このセクションでは、添付を相互に送信する Web サービスと Web クライアントの例を紹介します。

## 7.3.1 Web サービス

Web サービスには、次の 2 つのメソッドがあります。

- `UploadAscii()` は、ASCII 添付を受信して保存します。
- `DownloadBinary()` は、要求側にバイナリ添付を送信します。

クラス定義は以下のとおりです。

### Class Definition

```
Class GSOAP.FileTransferWS Extends %SOAP.WebService
{

///  Name of the web service.
Parameter SERVICENAME = "FileTransfer";

///  SOAP namespace for the web service
Parameter NAMESPACE = "http://www.filetransfer.org";

/// Namespaces of referenced classes will be used in the WSDL.
Parameter USECLASSNAMESPACES = 1;

///  Receive an attachment and save it
Method UploadAscii(filename As %String = "sample.txt") As %Status [WebMethod]
{
```

```
  //assume 1 attachment; ignore any others
  Set attach=..Attachments.GetAt(1)

  Set file=##class(%FileCharacterStream).%New()
  Set file.Filename="c:\from-client"_$H_filename

  //copy attachment into file
  Set status=file.CopyFrom(attach.Body)
  If $$$ISERR(status) {do $System.Status.DisplayError(status)}
  Set status= file.%Save()
  Quit status
}

///  Create an attachment and send it in response to the web client call
Method DownloadBinary(filename As %String = "sample.pdf") As %Status [WebMethod]
{
  //use a file-type stream to read file contents
  Set file=##class(%Library.FileBinaryStream).%New()
  Set file.Filename="c:\"_filename

  //create MIMEpart and add file to it
  Set mimepart=##class(%Net.MIMEPart).%New()
  Set mimepart.Body=file

  //set header appropriately for binary file
  Do mimepart.SetHeader("Content-Type","application/octet-stream")
  Do mimepart.SetHeader("Content-Transfer-Encoding","binary")

  //attach
  Set status=..ResponseAttachments.Insert(mimepart)
  Quit status
}

}
```

## 7.3.2 Web クライアント

Web クライアント・アプリケーションには、次の 2 つのメソッドがあります。

- `UploadAscii()` は、ASCII ファイルを Web サービスに送信します。
- `DownloadBinary()` は、Web サービスを呼び出し、応答としてバイナリ・ファイルを受信します。

生成された Web クライアント・クラス (`GSOAPClient.FileTransfer.FileTransferSoap`) には、メソッド `UploadAscii()` および `DownloadBinary()` が組み込まれます。これによって先行の Web サービスの対応メソッドが起動されます。クラスは表示されません。

Web クライアント・アプリケーションには、生成された Web クライアント・クラスを使用する次のクラスも組み込まれます。

### Class Definition

```
Include %systemInclude

Class GSOAPClient.FileTransfer.UseClient
{

ClassMethod DownloadBinary(filename As %String = "sample.pdf") As %Status
{
  Set client=##class(GSOAPClient.FileTransfer.FileTransferSoap).%New()

  //call web method
  Set ans=client.DownloadBinary(filename)

  //get the attachment (assume only 1)
  Set attach=client.ResponseAttachments.GetAt(1)

  //create a file and copy stream contents into it
  Set file=##class(%FileBinaryStream).%New()
  //include $H in the filename to make filename unique
  Set file.Filename="c:\from-service"_$H_filename
  Set status=file.CopyFrom(attach.Body)
  If $$$ISERR(status) {do $System.Status.DisplayError(status)}
  Set status= file.%Save()
  Quit status
}
```

```
ClassMethod UploadAscii(filename As %String = "sample.txt") As %Status
{
  Set client=##class(GSOAPClient.FileTransfer.FileTransferSoap).%New()

  //use a file-type stream to read file contents
  Set file=##class(%Library.FileCharacterStream).%New()
  Set file.Filename="c:\"_filename

  //create MIME part, add file as Body, and set the header
  Set mimepart=##class(%Net.MIMEPart).%New()
  Set mimepart.Body=file
  Do mimepart.SetHeader("Content-Transfer-Encoding","7bit")

  //attach to client and call web method
  Do client.Attachments.Insert(mimepart)
  Set status=client.UploadAscii(filename)
  Quit status
}


}
```

# 8

# カスタム・ヘッダ要素の追加と使用

このトピックでは、カスタムの SOAP ヘッダ要素の追加方法および使用方法を説明します。

フォルト発生時のヘッダ要素の追加の詳細は、"SOAP フォルトの処理" を参照してください。

WS-Addressing ヘッダ要素については、別の場所で説明します。WS-Security ヘッダ要素については、"Web サービスの保護" を参照してください。

## 8.1 InterSystems IRIS の SOAP ヘッダ要素入門

SOAP メッセージには、ヘッダ (<Header> 要素) を含めることができます。ヘッダは、一連のヘッダ要素で構成されます。以下はその例です。

```
<SOAP-ENV:Envelope>
   <SOAP-ENV:Header>
      <MyHeaderElement>
         <Subelement1>abc</Subelement1>
         <Subelement2>def</Subelement2>
      </MyHeaderElement>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
...
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

多くの場合、各ヘッダ要素が非公式にヘッダと呼ばれます。ただし、これは厳密には正しくありません。なぜなら、メッセージ自体に含まれるヘッダの数は最大でも 1 つであり、このヘッダには常に <Header> と適切なネームスペース接頭語が使用されるためです。ヘッダには、WS-Security ヘッダ要素、WS-Addressing ヘッダ要素、および独自のカスタム・ヘッダ要素を格納することができます。

ヘッダ要素には、SOAP メッセージを受信する Web サービスまたは Web クライアントで使用可能な追加情報が含まれます。ここに示す例では、この情報は XML 要素内に含まれています。上記の例には含まれていませんが、ヘッダ要素には XML 属性を含めることもできます。SOAP 規格では、受信者が SOAP メッセージをどのように処理するかを示す 3 つの標準属性 (mustUnderstand、actor、および encodingStyle) が指定されています。

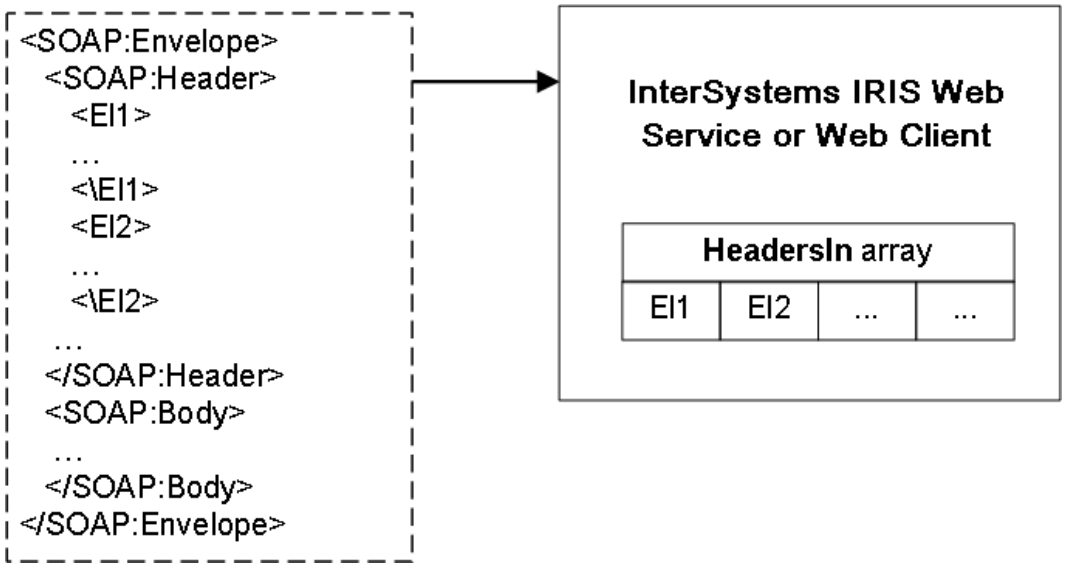
### 8.1.1 InterSystems IRIS で SOAP ヘッダを表す方法

InterSystems IRIS では、各ヘッダ要素を %SOAP.Header のインスタンス、またはそのサブクラスのいずれかとして表します。%SOAP.Header は XML 対応クラスであり、そのプロパティは標準的なヘッダ要素属性 (mustUnderstand、actor、および encodingStyle) に対応します。

InterSystems IRIS には、WS-Addressing および WS-Security で使用するための %SOAP.Header の特殊なサブクラスが用意されています。カスタム・ヘッダ要素を表すには、%SOAP.Header の独自のサブクラスを作成します。

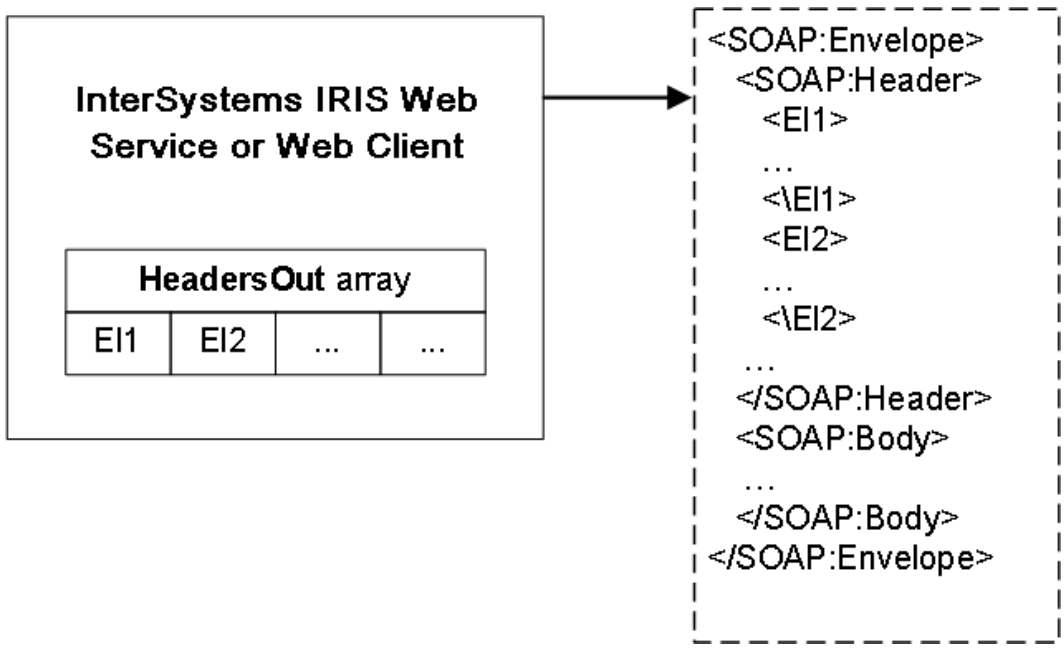
InterSystems IRIS Web サービスまたは Web クライアントが SOAP メッセージを受け取ると、そのメッセージをインポートして処理します。この手順では、カスタム・ヘッダ要素の指定されたヘッダがメッセージに含まれる場合、InterSystems IRIS は、そのヘッダ要素をサポート対象のヘッダ要素のリスト (次のサブセクションを参照) と比較します。

次に、サービスまたはクライアントは、以下のように該当する各ヘッダ要素クラスのインスタンスを作成し、それらを配列に挿入して、その配列を自身の HeadersIn プロパティに配置します。




InterSystems IRIS Web サービスまたは Web クライアントは、それらのヘッダ要素を使用するために HeadersIn プロパティにアクセスすることができます。SOAP メッセージに <Header> 要素が含まれていなかった場合、HeadersIn プロパティの Count() は 0 です。

同様に、InterSystems IRIS Web サービスまたは Web クライアントが SOAP メッセージを送信する前に、HeadersOut プロパティを更新して、発信メッセージに組み込むカスタム要素が格納されるようにする必要があります。HeadersOut Count() が 0 の場合、発信 SOAP メッセージに <Header> 要素は組み込まれません。



カスタム・ヘッダ要素の場合は、HeadersIn プロパティと HeadersOut プロパティを常に使用します。

その他 (カスタム以外) のヘッダ要素の場合は、以下のように詳細が異なります。

- WS-Addressing には、HeadersIn プロパティと HeadersOut プロパティではなく、AddressingIn プロパティと AddressingOut プロパティを使用します。"WS-Addressing ヘッダ要素の追加と使用" を参照してください。
- WS-Security ヘッダ要素には、WS-Policy 機能を使用してください ("Web サービスの保護" を参照)。

  または、SecurityIn プロパティと SecurityOut プロパティを直接使用します ("Web サービスの保護" を参照)。一般に、こちらのほうが作業が多くなります。

  (WS-Security ヘッダ要素は HeadersIn プロパティと HeadersOut プロパティにも格納されていますが、これらのプロパティを通じて、それらの要素にアクセスしたり、それらの要素を設定したりすることはお勧めできません)。
- InterSystems IRIS SOAP セッションのサポートでは、HeadersIn プロパティと HeadersOut プロパティが使用されます。"SOAP セッション管理" を参照してください。

## 8.1.2 サポート対象のヘッダ要素

InterSystems IRIS Web サービスおよび Web クライアントでは、WS-Addressing ヘッダおよび WS-Security ヘッダは自動的にサポートされますが、その他のヘッダは自動的にサポートされるわけではありません。

InterSystems IRIS Web サービスまたは Web クライアントでサポート対象のヘッダ要素を指定するには、クラスに XData ブロックを追加して、クラス・パラメータ USECLASSNAMESPACES を指定します。この XData ブロックは、サポート対象の要素をリストします。このクラス・パラメータにより、適用可能なタイプが WSDL に含まれるようになります。"サポート対象のヘッダ要素の指定" を参照してください。

## 8.1.3 ヘッダ要素と WSDL

Web サービスの WSDL は、その Web サービスでサポートされていて、その Web サービスと通信する Web クライアントで許可されているヘッダ要素を通知します。

InterSystems IRIS Web サービスの場合、生成された WSDL には、SOAP ヘッダ要素に関する情報が含まれていないことがあります。

- HeadersOut プロパティを設定して、手動で SOAP ヘッダを追加する場合は、"サポート対象のヘッダ要素の指定" の説明に従い、それらのヘッダを XData ブロックで宣言します。また、Web サービス・クラスのクラス・パラメータ USECLASSNAMESPACES を 1 に指定してください。

  これらの手順を実行すると、すべての該当情報が WSDL に含まれるようになります。それ以外の場合は、これが行われなくなるため、WSDL をファイルに保存して、そのファイルを必要に応じて手動で編集することが必要になります。
- SecurityOut プロパティ ("Web サービスの保護" を参照) を設定して、WS-Security ヘッダを追加した場合、WSDL には必要な情報の一部が含まれなくなります。(これは、コンパイル時に WSDL が生成され、その後の実行時にヘッダが追加されるためです。)この場合は、WSDL をファイルに保存して、そのファイルを必要に応じて手動で編集してください。

  多くの理由から、WS-Policy を使用することで、WS-Security 要素の追加が単純で簡単になります。WS-Policy を使用すると、生成された WSDL には必要な情報がすべて含まれます。
- その他の場合は、生成された WSDL にすべての情報が含まれます。

W3C 仕様では、生成された WSDL を Web サービスが提供することを要求していない点に注意してください。

### 8.1.4 必須のヘッダ要素

特定のヘッダ要素が mustUnderstand=1 と指定する場合、その要素は必須と見なされ、受信者はその要素をサポートする必要があります。受信者は、必須ヘッダ要素をすべて認識しない限り、メッセージを処理することはできません。

SOAP 規格に従って、InterSystems IRIS では、必須でありながらサポート対象ではないヘッダ要素を含む SOAP メッセージが拒否されます。具体的には、InterSystems IRIS Web サービスまたは Web クライアントは、mustUnderstand=1 と指定されたヘッダ要素を含むメッセージを受信したときに、そのヘッダ要素がサポート対象でない場合には、SOAP フォルトを発行してそのメッセージを無視します。

## 8.2 カスタム・ヘッダ要素の定義

SOAP ウィザードを使用して、特定の WSDL に基づいて InterSystems IRIS Web サービスまたは Web クライアントを作成すると、必要に応じて任意のヘッダ要素を表すためのクラスが生成されます。

一方、Web サービスまたは Web クライアントを手動で作成する場合には、任意のカスタム・ヘッダ要素を表すためのクラスを手動で定義する必要があります。そのためには、以下の操作を実行します。

1. カスタム・ヘッダ要素ごとに、%SOAP.Header のサブクラスを作成します。
2. ヘッダ要素のネームスペースを示す NAMESPACE パラメータを指定します。
3. ヘッダ要素の名前を示す XMLNAME パラメータを指定します。
4. このサブクラスで、必要なヘッダ情報を含めるようにプロパティを定義します。既定では、プロパティは、<Header> 要素内の要素に投影されます。
5. オプションで、このヘッダ要素の形式を制御する XMLFORMAT パラメータを指定します。既定では、ヘッダ要素は常にリテラル形式です (SOAP エンコードではありません)。

以下はその例です。

#### Class Definition

```
Class Scenario1.MyHeaderElement Extends %SOAP.Header
{

Parameter NAMESPACE = "http://www.myheaders.org";

Parameter XMLNAME = "MyHeader";

Property Subelement1 As %String;

Property Subelement2 As %String;

}
```

このヘッダ要素は、SOAP メッセージ内で以下のように表示されます。

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope [parts omitted]>
  <SOAP-ENV:Header>
    <MyHeader xmlns="http://www.myheaders.org"
                    xmlns:hdr="http://www.myheaders.org">
      <Subelement1>abc</Subelement1>
      <Subelement2>def</Subelement2>
    </MyHeader>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
   [omitted]
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

任意のオブジェクト・クラスの XML プロジェクションをカスタマイズする方法の詳細は、"オブジェクトの XML への投影" を参照してください。

SOAP ウィザードの詳細は、"SOAP ウィザードの使用法" を参照してください。

# 8.3 SOAP メッセージへのカスタム・ヘッダ要素の追加

Web サービスまたは Web クライアントからカスタム・ヘッダ要素を SOAP メッセージに追加するには、SOAP メッセージを送信する前に次の操作を実行します。

1. ヘッダ・オブジェクトのインスタンスを作成します。
2. 必要に応じてヘッダ・オブジェクトのプロパティを設定します。オプションで、actor プロパティと mustUnderstand プロパティも設定します。
3. 発信ヘッダ配列の HeadersOut プロパティに新しいヘッダを追加します。このプロパティは、通常の配列インタフェース (SetAt() メソッド、Count() メソッド、GetAt() メソッドなど) を使用した配列です。

注釈　ユーティリティ・メソッドでこれらの手順を実行する場合、そのメソッドがインスタンス・メソッドであり、抽象クラスなどではなく、インスタンス化可能なクラスのメンバである必要があります。

また、Web サービスまたは Web クライアントのクラス内に、ヘッダ要素を追加するユーティリティ・メソッドがあるとします。

Class Member

```
Method AddMyHeaderElement(mustUnderstand=0)
{
    Set h=##class(MyHeaderElement).%New()
    Set h.Subelement1 = "abc"
    Set h.Subelement2 = "def"
    If mustUnderstand {Set h.mustUnderstand=1}
    Do ..HeadersOut.SetAt(h,"MyHeaderElement")
}
```

そして、このユーティリティ・メソッドを使用する各 Web メソッドからそのユーティリティ・メソッドを呼び出すことができるとします。以下はその例です。

Class Member

```
/// Divide arg1 by arg2 and return the result
Method Divide(arg1 As %Numeric, arg2 As %Numeric) As %Numeric [ WebMethod ]
{
    //main method code here
    //...

    do ..AddMyHeaderElement()

    Quit ans
}
```

この Web メソッドを呼び出すと、SOAP 応答にヘッダが追加されます。

# 8.4 サポート対象のヘッダ要素の指定

InterSystems IRIS Web サービスおよび Web クライアントでは、WS-Addressing ヘッダ要素および WS-Security ヘッダ要素は自動的にサポートされますが、その他のヘッダ要素は自動的にサポートされるわけではありません。

InterSystems IRIS Web サービスまたは Web クライアントでサポートされるヘッダ要素を指定するには、以下の手順を実行します。

- それらのヘッダ要素を表すクラスを定義します。その手順は、"カスタム・ヘッダ要素の定義" を参照してください。
- ヘッダ要素クラスを、Web サービスまたは Web クライアントのヘッダ要素に関連付けます。

  これは、以下の 2 つの方法のいずれかで行うことができます。

  - Web サービスまたは Web クライアントのクラスに XData ブロックを追加します。この XData ブロックで、特定のヘッダ要素とそれらに対応するヘッダ要素クラス間の関連付けを指定します。

    さらに、そのサービスまたはクライアントのクラスでクラス・パラメータ USECLASSNAMESPACES を 1 (推奨値) に設定する場合、生成された WSDL で、このヘッダ情報が使用されます。

  - Web サービスまたは Web クライアントのクラスで、SOAPHEADERS パラメータを指定します。このパラメータで、特定のヘッダ要素とそれらに対応するヘッダ要素クラス間の関連付けを指定します。

    注釈　この手法は柔軟性があまりなく、生成された WSDL には影響しないため、現在は非推奨となっています。

  以降のセクションで詳細を説明します。

# 8.5 XData ブロックでのサポート対象のヘッダ要素の指定

SOAP ウィザードを使用して、特定の WSDL に基づいて InterSystems IRIS Web サービスまたは Web クライアントを作成すると、その SOAP メッセージでサポート対象の任意のヘッダ要素を表すための XData ブロックがそのクラスに生成されます。(SOAP ウィザードの詳細は、"SOAP ウィザードの使用法" を参照してください。)

一方、Web サービスまたはクライアントを手動で作成する場合には、この XData ブロックも手動で指定する必要があります。

以下の簡単な例で説明します。

```
XData NewXData1
{
<parameters xmlns="http://www.intersystems.com/configuration">
   <request>
      <header name="ServiceHeader" class="NewHeaders.MyCustomHeader"/>
   </request>
   <response>
      <header name="ExpectedClientHeader" class="NewHeaders.MyCustomHeader"/>
   </response>
</parameters>
}
```

## 8.5.1 詳細

この XData ブロックの要件は、以下のとおりです。

- XData ブロックには、どのような名前でも指定できます。その名前 (この例では `NewXData1`) は使用されません。

  SOAP ウィザードでは、このブロックの作成時に `parameters` という名前が使用されます。

- 最上位の要素は、`<parameters>` であることが必要です。
- `<parameters>` 要素とそのすべての子要素 (とさらにそれらの子) は、`"http://www.intersystems.com/configuration"` ネームスペース内に存在している必要があります。

- <parameters> 要素には、以下の子要素を含めることができます。
  - <request> ー すべての要求メッセージに関連付けられるヘッダ要素を決定します。すべての要求メッセージに共通するヘッダ要素の場合に使用します。

    この要素には、該当するヘッダ要素ごとに子要素 <header> が必要です。
  - <response> ー すべての応答メッセージに関連付けられるヘッダ要素を決定します。すべての応答メッセージに共通するヘッダ要素の場合に使用します。

    この要素には、該当するヘッダ要素ごとに子要素 <header> が必要です。
  - <methodname> ー methodname という名前の Web メソッドに関連付けられるヘッダ要素を決定します。

    この要素には、以下の子要素を含めることができます。
    - <header> ー この Web メソッドの要求メッセージと応答メッセージに関連付けられるヘッダ要素を決定します。両方の場合に共通するヘッダ要素の場合に使用します。
    - <request> ー この Web メソッドの要求メッセージに関連付けられるヘッダ要素を決定します。

      この要素には、該当するヘッダ要素ごとに子要素 <header> が必要です。
    - <response> ー この Web メソッドの応答メッセージに関連付けられるヘッダ要素を決定します。

      この要素には、該当するヘッダ要素ごとに子要素 <header> が必要です。

- この XData ブロックでは、各 <header> 要素は、ヘッダ要素を表すために使用される InterSystems IRIS クラスにヘッダ要素を関連付けます。この要素には以下の属性が含まれます。

| 属性 | 目的 |
|---|---|
| name | ヘッダ要素の名前。 |
| class | そのヘッダ要素を表す InterSystems IRIS クラス。 |
| alias | (オプション) Web サービスまたは Web クライアントの HeadersIn 配列内のそのヘッダ要素のキー。既定値は、name 属性に指定された値です。 |

XData ブロック内での <header> 要素の位置によって、適用先のメッセージが示されます。

## 8.5.2 カスタム・ヘッダの継承

この Web サービスのサブクラスを作成する場合は、そのサブクラスが、メソッド固有ではないヘッダ情報を継承します。ヘッダ情報は、<parameters> の直接の子要素である <request> または <response> 要素に含まれています。これは、SOAPMETHODINHERITANCE が 0 の場合も同じです。

## 8.5.3 例

別の例を以下に示します。

### Class Member

```
XData service
{
<parameters xmlns="http://www.intersystems.com/configuration">
  <response>
    <header name="Header2" class="User.Header4" alias="Header4"/>
    <header name="Header3" class="User.Header3"/>
  </response>
  <method name="echoBase64">
```

```
    <request>
      <header name="Header2" class="User.Header4" alias="Header4"/>
      <Action>http://soapinterop.org/Round2Base.Service.echoBase64Request</Action>
    </request>
    <response>
      <header name="Header2" class="User.Header2" alias="Header2"/>
      <header name="IposTransportHeader" class="ipos.IposTransportHeader"/>
      <Action>http://soapinterop.org/Round2Base.Service.echoBase64Result</Action>
    </response>
  </method>
  <method name="echoString">
    <request>
      <Action>http://soapinterop.org/Round2Base.Service.echoStringRequest</Action>
    </request>
    <response>
      <Action>http://soapinterop.org/Round2Base.Service.echoStringAnswer</Action>
    </response>
  </method>
</parameters>
}
```

# 8.6 SOAPHEADERS パラメータでのサポート対象のヘッダ要素の指定

サポート対象のヘッダ要素を指定する従来の方法では、Web サービスまたは Web クライアントのクラスに SOAPHEADERS パラメータを含めます。

このパラメータは、ヘッダ仕様のコンマで区切られたリストと同じである必要があります。各ヘッダ仕様の形式は、以下のとおりです。

```
headerName:headerPackage.headerClass
```

headerName はサポート対象のヘッダの要素名で、headerPackage.headerClass は、そのヘッダを表すクラスのパッケージとクラスの完全な名前です。以下はその例です。

```
Parameter SOAPHEADERS = "MyHeaderElement:Scenario1Client.MyHeaderElement"
```

このリストでは、その Web サービスまたは Web クライアントに対する SOAP 要求でサポートされるすべてのヘッダが特定され、それぞれのマッピング先のクラスが表されます。

この従来の手法を使用する場合は、以下の点に注意してください。

- Web サービスの場合、この手法は、生成された WSDL に影響しません。
- 特定の Web メソッドに異なるヘッダ要素を指定することはできません。
- SOAP ウィザードでは、生成された Web サービスおよび Web クライアントのクラスに SOAPHEADERS パラメータが生成されなくなりました。
- この手法は非推奨となっています。

## 8.6.1 カスタム・ヘッダの継承

この Web サービスのサブクラスを作成する場合は、そのサブクラスが SOAPHEADERS パラメータを継承します。これは、SOAPMETHODINHERITANCE が 0 の場合も同じです。

# 8.7 ヘッダ要素の使用法

要求メッセージの受信後に特定の SOAP ヘッダ要素を使用するには、サービスまたはクライアントの **HeadersIn** プロパティを使用します。

サポート対象のヘッダ要素ごとに、Web サービスまたは Web クライアントは、適切なヘッダ・クラスのインスタンスを作成して、そのヘッダを着信ヘッダ配列である **HeadersIn** プロパティに追加します。このプロパティは、通常の配列インタフェース (SetAt() メソッド、Count() メソッド、GetAt() メソッドなど) を使用した配列です。Web サービスまたは Web クライアントは、これらのヘッダを必要に応じて処理できるようになります。

注釈　ヘッダ要素のネームスペースは、リストのヘッダ要素のマッチングには使用されません。ただし、SOAP メッセージのヘッダ要素のネームスペースは、ヘッダ要素サブクラスの NAMESPACE パラメータで指定されたものと同じである必要があります。異なっていると、メッセージのインポート時にエラーが発生します。

# 9

# WS-Addressing ヘッダ要素の追加と使用

このトピックでは、WS-Addressing ヘッダ要素の追加方法および使用方法を説明します。

この標準の詳細は、"SOAP 標準" を参照してください。

"フォルト発生時の WS-Addressing ヘッダ要素の追加" も参照してください。

## 9.1 概要

SOAP 1.1 と SOAP 1.2 の WS-Addressing 規格で定められているように、WS-Addressing ヘッダ要素を SOAP メッセージに追加することができます。これには、以下のいずれかを実行します。

- Web サービスまたはクライアントの WSADDRESSING パラメータを "AUTO" に指定します。このオプションにより、WS-Addressing ヘッダ要素の既定のセットが追加されます。これについては、次のサブセクションで説明します。
- 次のサブセクションで説明するように、WSADDRESSING を "OFF" (既定値) に指定し、WS-Addressing ヘッダ要素を手動で追加します。
- WS-Addressing ヘッダ要素を含むように Web サービスまたはクライアントのポリシーを作成します。そのためには、Web サービスまたはクライアントを参照する構成クラスを作成およびコンパイルし、このポリシー内で WS-Addressing を有効にします。"Web サービスの保護" を参照してください。

  このようなポリシーを添付すると、InterSystems IRIS では、同じ既定の WS-Addressing ヘッダ要素のセットが既定で使用されるようになります。WS-Addressing ヘッダ要素を手動で作成して追加することも可能です。

  このようなポリシーを添付すると、WSADDRESSING の値は無視されます。

## 9.2 WSDL への影響

Web サービスの場合、WSADDRESSING パラメータは、生成された WSDL に影響しません。同様に、Web クライアントに対してこれを指定した場合、WSDL を変更する必要はありません。

WS-Addressing を参照するポリシー文は、WSDL に影響します。ポリシー文を追加する場合、Web クライアントの再生成が必要です。InterSystems IRIS Web クライアントでは、クライアント・クラスを再生成する代わりに、WS-Addressing ポリシー文をクライアントに添付するだけですみます。

# 9.3 既定の WS-Addressing ヘッダ要素

このセクションでは、InterSystems IRIS で使用される既定の WS-Addressing ヘッダ要素の例について説明します。

## 9.3.1 要求メッセージにおける既定の WS-Addressing ヘッダ要素

このセクションで前に説明したとおりに WS-Addressing を有効にすると、Web クライアントの要求メッセージには次の WS-Addressing ヘッダ要素が含まれます。

- To：宛先アドレス
- Action：SoapAction
- MessageID：一意の UUID
- ReplyTo：匿名

以下はその例です。

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
                   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
                   xmlns:s='http://www.w3.org/2001/XMLSchema'
                   xmlns:wsa='http://www.w3.org/2005/08/addressing'>
   <SOAP-ENV:Header>
      <wsa:Action>http://www.myapp.org/GSOAP.DivideAddressingWS.Divide</wsa:Action>
      <wsa:MessageID>urn:uuid:91576FE2-4533-43CB-BFA1-51D2B631453A</wsa:MessageID>
      <wsa:ReplyTo>
        <wsa:Address xsi:type="s:string">http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>

      </wsa:ReplyTo>
      <wsa:To>http://localhost:8080/csp/gsop/GSOP.DivideAddressingWS.cls</wsa:To>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <Divide xmlns="http://www.myapp.org">
         <arg1 xsi:type="s:decimal">1</arg1>
         <arg2 xsi:type="s:decimal">7</arg2>
      </Divide>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## 9.3.2 応答メッセージにおける既定の WS-Addressing ヘッダ要素

このセクションで前に説明したとおりに WS-Addressing を有効にすると、要求メッセージに WS-Addressing ヘッダ要素が含まれているときに、Web サービスの応答メッセージには次の WS-Addressing ヘッダ要素が含まれます。

- To：匿名
- Action：SoapAction_"Response"
- MessageID：一意の UUID
- RelatesTo：要求の MessageID

以下はその例です。

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
                   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
                   xmlns:s='http://www.w3.org/2001/XMLSchema'
                   xmlns:wsa='http://www.w3.org/2005/08/addressing'>
   <SOAP-ENV:Header>
      <wsa:Action>http://www.myapp.org/GSOAP.DivideAddressingWS.DivideResponse</wsa:Action>
      <wsa:MessageID>urn:uuid:577B5D65-D7E3-4EF7-9BF1-E8422F5CD739</wsa:MessageID>
      <wsa:RelatesTo>urn:uuid:91576FE2-4533-43CB-BFA1-51D2B631453A</wsa:RelatesTo>
      <wsa:To>http://www.w3.org/2005/08/addressing/anonymous</wsa:To>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <DivideResponse xmlns="http://www.myapp.org">
         <DivideResult>.14285714285714285</DivideResult>
      </DivideResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# 9.4 手動での WS-Addressing ヘッダ要素の追加

既定の WS-Addressing ヘッダ要素を使用する代わりに、独自の要素を手動で作成して追加することができます。そのためには、以下の操作を実行します。

1. %SOAP.Addressing.Properties のインスタンスを作成し、必要に応じてプロパティを指定します。詳細は、クラスリファレンスを参照してください。
2. Web サービスまたはクライアントの AddressingOut プロパティを %SOAP.Addressing.Properties のこのインスタンスと等しくなるように設定します。

注釈　AddressingOut プロパティを設定すると、Web サービスまたは Web クライアントでは、添付されたポリシーで指定されている WS-Addressing 要素ではなく、このプロパティの WS-Addressing ヘッダ要素が使用されます。

# 9.5 WS-Addressing ヘッダ要素の処理

InterSystems IRIS Web サービスまたは Web クライアントで WS-Addressing ヘッダ要素を含むメッセージを受信すると、サービスまたはクライアントの AddressingIn プロパティは、%SOAP.Addressing.Properties のインスタンスと等しくなるように更新されます。Web サービスまたは Web クライアントは、その AddressingIn プロパティの詳細を検証できるようになります。

%SOAP.Addressing.Properties の詳細は、クラス・リファレンスを参照してください。

# 10

# SOAP セッション管理

SOAP Web サービスは本質的にステートレスのため、セッションを管理しません。ただし、Web クライアントと Web クライアントが使用する Web サービスとの間のセッション管理が有用な場合がよくあります。InterSystems IRIS SOAP サポートは、これを実行するための方法を提供します。

"InterSystems IRIS での Web クライアントの調整" の "カスタムの HTTP 要求の指定" も参照してください。

また、"生成された WSDL の詳細" の "InterSystems IRIS SOAP セッションの WSDL の相違点" を参照してください。

## 10.1 SOAP セッションの概要

Web クライアントと InterSystems IRIS Web サービスとの間のセッションを管理できます。このサポートは以下のツールで構成されています。

- Web セッション管理 (InterSystems IRIS および InterSystems Web ゲートウェイで管理)
- 簡単な専用ヘッダである InterSystems IRIS SOAP セッション・ヘッダ

全体的な流れは、以下のとおりです。

1. Web クライアントは、初期メッセージを Web サービスに送信します。このメッセージには、InterSystems IRIS SOAP セッション・ヘッダは含まれません。
2. Web サービスは、メッセージを受信し、新しい Web セッションを開始します。
3. Web サービスは、その返信を送信するときに、メッセージに InterSystems IRIS SOAP セッション・ヘッダを追加します。
4. 応答を受信した Web クライアントでは、その SOAP セッション・ヘッダを検出し、セッションの cookie を抽出する必要があります。Web クライアントでは、以降のメッセージを送信するときに、その cookie を使用してメッセージに SOAP セッション・ヘッダを作成する必要があります。

   メモ：

   - クライアントが InterSystems IRIS Web クライアントである場合、セッション cookie は自動的にその Web クライアントの SessionCookie プロパティに保存されます。また、そのクライアントのインスタンスで SOAP セッション・ヘッダが自動的に作成され、そのクライアントから送信するすべてのメッセージにそのヘッダが追加されます。
   - セッションで発生するすべての SOAP メッセージに同じクライアント・インスタンスを使用する場合は、この手順が .NET Web クライアントでも自動的に実行されます。その他のクライアント・プラットフォームでは追加のコードが必要な場合があります。

5. Web サービスは、次の応答を受信し、Web セッションを続行して、Web サービスによる応答時に SOAP セッション・ヘッダを再度追加します。

ログ・アウトのメソッドを追加する必要はありません。Web セッションは、短い時間の経過後 (Web アプリケーションのタイムアウト期間が経過した後) にタイムアウトします。

# 10.2 セッションの有効化

SOAP セッションに対する InterSystems IRIS サポートを使用するには、InterSystems IRIS Web サービスを使用している必要があります。

- Web クライアントが InterSystems IRIS を使用して作成されている場合、SOAP セッションのサポートを有効にするために必要な手順は 1 つだけです。Web サービス・クラスで、SOAPSESSION パラメータを 1 に設定します。
- Web クライアントの作成にサードパーティのツールを使用している場合は、初期応答で InterSystems IRIS SOAP セッション・ヘッダ要素を検出する必要があり、またセッションの有効期間中のすべての要求にこのヘッダ要素が Web クライアントによって追加されるようにする必要があります。このヘッダ要素の形式は、以下のとおりです。

  XML

```
<csp:CSPCHD xmlns:csp="http://www.intersystems.com/SOAPheaders"><id>value of CSPCHD
token</id></csp:CSPCHD>
```

# 10.3 セッション情報の使用

セッションが有効な場合、Web サービスは、%CSP.Session のインスタンスである、変数 %session を使用できます。このオブジェクトのプロパティには、システム情報およびユーザが追加するように選択した情報が含まれます。以下は、一般的に使用されるプロパティの一部です。

- SessionID ― このセッションの一意の識別子。
- EndSession ― 通常、これは 0 です。セッションを終了するには、このプロパティを 1 に設定します。
- Data ― カスタム・データの格納に使用される InterSystems IRIS 多次元配列。
- NewSession ― このセッションが新しいセッションの場合は 1 です。
- AppTimeout ― セッションのタイムアウト値を秒単位で指定します。

%session オブジェクトには、その他の多くのプロパティを初め、セッションに関連するタスクに使用できるいくつかのメソッドがあります。詳細は、%CSP.Session のクラス・ドキュメントを参照してください。

# 11

# InterSystems IRIS バイナリ SOAP 形式の使用法

InterSystems IRIS SOAP サポートは、オプションで、専用のバイナリ SOAP 形式を提供します。これは、大きな SOAP メッセージを送受信するときにメッセージのサイズを最小化する場合に有用です。

InterSystems IRIS Web サービスは、InterSystems IRIS バイナリ SOAP 形式または通常の SOAP 形式のいずれかで SOAP 要求を受信できます。この動作を有効にするためにパラメータは必要ありません。InterSystems IRIS Web クライアントは、バイナリ SOAP 形式を使用するように構成されている場合にのみ、この形式を使用します。

"生成された WSDL の詳細" の "InterSystems IRIS バイナリ SOAP 形式の WSDL の相違点" も参照してください。

注釈　InterSystems IRIS Web サービスまたは Web クライアントがこの専用のバイナリ SOAP 形式を使用する場合、その Web サービスまたは Web クライアントで WS-Security 機能または WS-Policy 機能を使用することはできません。"Web サービスの保護" を参照してください。

## 11.1 概要

以下に示すように、InterSystems IRIS バイナリ SOAP は HTTP メッセージで使用されます。

- このメッセージは POST メソッドを使用します。
- コンテンツ・タイプは常に `"application/octet-stream"` です。
- 本文は、専用プロトコルを使用したオブジェクトのバイナリ表現です。
- バイナリ SOAP 要求には、次の形式の HTTP ISCSoap ヘッダが含まれます。

```
ISCSoap: NAMESPACE/Package.Class.Method
```

- SOAP セッションがサポートされます。セッション情報は、標準の Web セッション cookie を使用することによって管理されます。ただし、SOAP Web クライアントと Web サービスの **SessionCookie** プロパティはサポートされません。これは、バイナリ SOAP では、CSPCHD 専用 SOAP ヘッダが使用されないためです。

以下は、バイナリ SOAP 要求の例を示しています。

```
POST /csp/gsop/GSOP.WebServiceBinary.cls HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; InterSystems IRIS;)
Host: localhost:8080
Connection: Close
ISCSoap: http://www.myapp.org/GSOAP.WebServiceBinary.Divide
Content-Type: application/octet-stream
Content-Length: 90

00085hdBinaryClient.MyAppSoap.Dividearg1arg2t
```

影響を受けるのは、SOAP エンベロープとその内容のみです。HTTP ヘッダには影響しません。

# 11.2 InterSystems IRIS Web サービスの WSDL の拡張

すべての InterSystems IRIS Web サービスが、InterSystems IRIS バイナリ SOAP 形式または通常の SOAP 形式のいずれかで SOAP 要求を受信できます。InterSystems IRIS Web サービスがバイナリ要求を受け取ると、その Web サービスはバイナリ応答を送信します。それ以外の場合、Caché Web サービスは通常の応答を送信します。この動作を有効にするためにパラメータは必要ありません。

Web サービスの WSDL を拡張すると、次の処理を実行できます。

1. Web サービスが通常の SOAP 形式に加えて InterSystems IRIS バイナリ SOAP 形式をサポートしていることを WSDL で公開する。
2. InterSystems IRIS バイナリ SOAP 形式の使用に関する情報を WSDL に含める。

これにより、InterSystems IRIS Web クライアントは、必要に応じてこの形式のメッセージを適切に送信できるようになります。

このように InterSystems IRIS Web サービスの WSDL を拡張するには、Web サービスの SOAPBINARY パラメータを 1 に設定します。

変更の詳細は、"生成された WSDL の詳細" の "InterSystems IRIS バイナリ SOAP 形式の WSDL の相違点" を参照してください。

# 11.3 バイナリ SOAP を使用するように InterSystems IRIS Web クライアントを再定義する方法

InterSystems IRIS バイナリ SOAP 形式を使用するように既存の InterSystems IRIS Web クライアントを再定義できます。そのためには、Web クライアントの SOAPBINARY パラメータまたは **SoapBinary** プロパティを 1 に設定します。その他の変更が必要になることもあります。"生成された WSDL の詳細" の "InterSystems IRIS バイナリ SOAP 形式の WSDL の相違点" を参照してください。

# 11.4 文字セットの指定

Web クライアントの **SoapBinaryCharset** プロパティは、Web サービスの InterSystems IRIS 文字セット (例 : Unicode、Latin1) を指定します。クライアント・マシンとサービス・マシンの文字セットが同じ場合、文字列は RAW で送信され、それ以外の場合は UTF8 のエンコードで送信されます。

SoapBinaryCharset プロパティは、既定値の SOAPBINARYCHARSET パラメータに設定され、このパラメータは既定値の NULL に設定されます。NULL では常に文字列が UTF8 に変換されます。

# 11.5 InterSystems IRIS バイナリ SOAP 形式の詳細

以下に示すように、バイナリ SOAP の API は、XML SOAP とは異なります。

- InterSystems IRIS サーバの場合 :
  - バイナリ SOAP は、ISCSoap HTTP ヘッダの存在によって示されます。
  - Web サービスの Initialize() メソッドは呼び出されません。
  - 初期実装では、標準の %request.Content ストリームが使用されます。
  - ログインは、URL に添付されている IRISUsername および IRISPassword クエリ・パラメータを使用して実行されます。バイナリ SOAP でログイン・ページが返されることはありません。
  - 無効なログインが発生すると、%SOAP.Fault のインスタンスが返されます。
- %Net.HttpRequest の応答は以下のようになります。
  - 呼び出されるメソッドに Web クライアント・クラスの SoapBinary プロパティを設定することにより、バイナリ SOAP 要求が示されます。
  - 要求は、標準の EntityBody ストリームを使用して送信されます。
  - 応答は、HttpResponse の Data プロパティで返されます。

# 12

# SOAP メッセージでのデータセットの使用

このトピックでは、%XML.DataSet について説明します。これは、Web サービスと Web クライアントの両方が InterSystems IRIS を基盤とする場合や、いずれかが .NET を使用する場合に、SOAP メッセージで使用できる XML 対応のデータセットです。他の SOAP ベンダは、データセットをサポートしていません。

## 12.1 データセットについて

データセットとは、Microsoft 社によって定義され (また、.NET で使用され)、InterSystems IRIS でもサポートされている XML 形式の結果セットです。Web サービスと Web クライアントの両方が InterSystems IRIS を基盤とする場合や、いずれかが .NET を使用する場合、データセットを Web メソッドへの入力または Web メソッドからの出力として使用できます。他の SOAP テクノロジでは、この形式は認識されません。

InterSystems IRIS で Web サービスまたは Web クライアントを使用して作業する場合には、データセットを表現するために **%XML.DataSet** (またはカスタム・サブクラス) を使用します。**%XML.DataSet** の使用の詳細は、クラス参照を参照してください。

注釈　**%XML.DataSet** クラスは単一のテーブルのみをサポートします。つまり、それが使用するクエリは単一のテーブルのみを返すことができます。

サンプル Web サービス **SOAP.Demo** (**SAMPLES** ネームスペース内にあります) では、InterSystems IRIS でのデータセットの例を示します。具体的には、以下の Web メソッドでデータセットを使用します。

- GetByName()
- GetDataSetByName()
- QueryByName()

Java ベースの Web クライアントで使用できるようにクエリの結果を出力するには、**%ListOfObjects** サブクラスを使用します。**SOAP.Demo** に例があります。

## 12.2 typed データセットの定義

データセットではすべて、取得するデータを指定するクエリが使用されます。コンパイル時にクエリが既知のとき、データセットは typed となり、それ以外の場合は untyped となります。typed データセットは多くの場面で便利に使うことができます。例えば、.NET では、typed データセットによって Microsoft Visual Studio のコードを実行できます。

typed データセットを定義するには、%XML.DataSet のサブクラスを作成し、QUERYNAME パラメータと CLASSNAME パラメータを指定します。これらのパラメータは共に、特定の 1 つの SQL クエリを参照します。データセットのスキーマを生成すると、%XML.DataSet ではカスタム・データ型の LogicalToXSD() メソッドなど、クラスとプロパティのメタデータが考慮されます。

注釈　%XML.DataSet をメソッドの返り値として使用する場合、その値の XML タイプは DataSet です。一方、%XML.DataSet のサブクラスを返り値として使用する場合、この値の XML タイプはそのサブクラスの名前です。この動作は、他の XML 対応のクラスのものと同じであり、WSDL に記述される XML タイプに影響を及ぼします。"オブジェクトの XML への投影" の "XML タイプへの投影の制御" を参照してください。

# 12.3 データセットの形式の制御

既定では、データセットは、Microsoft DiffGram 形式で書き込まれ、XML スキーマが前に付きます。以下に例を示します。

```
<SOAP-ENV:Body>
  <Get0Response xmlns="http://www.myapp.org">
    <Get0Result>
    <s:schema id="DefaultDataSet" xmlns=""
    attributeFormDefault="qualified"
    elementFormDefault="qualified"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
      <s:element name="DefaultDataSet" msdata:IsDataSet="true">
        <s:complexType>
          <s:choice maxOccurs="unbounded">
            <s:element name="GetPeople">
              <s:complexType>
                <s:sequence>
                  <s:element name="Name" type="s:string" minOccurs="0" />
                  <s:element name="DOB" type="s:date" minOccurs="0" />
                </s:sequence>
              </s:complexType>
            </s:element>
          </s:choice>
        </s:complexType>
      </s:element>
    </s:schema>

    <diffgr:diffgram
    xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
    xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
      <DefaultDataSet xmlns="">
        <GetPeople diffgr:id="GetPeople1" msdata:rowOrder="0">
          <Name>Quine,Howard Z.</Name>
          <DOB>1965-11-29</DOB>
        </GetPeople>
...
      </DefaultDataSet>
    </diffgr:diffgram>
   </Get0Result>
 </Get0Response>
</SOAP-ENV:Body>
```

%XML.DataSet クラスには、この形式を制御する以下のオプションがあります。

- DATAONLY パラメータと DiffGram プロパティは、出力が DiffGram 形式であるかどうかを制御します。前述のとおり、既定では、出力は DiffGram 形式です。%XML.DataSet をサブクラスに指定し、DATAONLY を 1 に設定した場

合、または DiffGram を 0 に設定した場合は、出力が DiffGram 形式ではなくなります。この場合の XML データセットの本文は次のようになります。

```
<SOAP-ENV:Body>
  <Get0Response xmlns="http://www.myapp.org">
    <Get0Result>
      <GetPeople xmlns="">
        <Name>Quine,Howard Z.</Name>
        <DOB>1965-11-29</DOB>
      </GetPeople>
      <GetPeople xmlns="">
...
    </Get0Result>
  </Get0Response>
</SOAP-ENV:Body>
```

DiffGram 形式の場合とは対照的に、既定ではスキーマは出力されず、出力に <diffgram> 要素が含まれません。

- NeedSchema プロパティは、出力に XML スキーマを含めるかどうかを制御します。DiffGram 形式を使用していると、既定でスキーマが出力されますが、DiffGram 形式を使用していないと、既定ではスキーマが出力されません。スキーマの出力を強制するには、NeedSchema を 1 に設定します。またはスキーマの出力を抑止するには、このプロパティを 0 に設定します。
- DiffGram 形式を使用すると、WriteEmptyDiffgram プロパティによって、データセットに行が含まれない場合の <diffgram> 要素の内容が制御されます。以下に示すように、既定では (または WriteEmptyDiffgram が 0 の場合は)、<diffgram> 要素には空の要素が含められます。

```
...
<diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
   <DefaultDataSet xmlns="">
   </DefaultDataSet>
</diffgr:diffgram>
...
```

対照的に、WriteEmptyDiffgram が 1 のときは、<diffgram> 要素には何も含まれません。

```
...
<diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
</diffgr:diffgram>
...
```

DiffGram 形式を使用していない場合、このプロパティによる影響はありません。

- DiffGram 形式を使用すると、DataSetName プロパティによって、<diffgram> 要素に含まれる要素の名前が制御されます。前述の例のとおり、既定では、この要素は <DefaultDataSet> と名付けられます。DiffGram 形式を使用していない場合、このプロパティによる影響はありません。

%XML.DataSet は、CaseSensitive プロパティも提供します。これは、同じ名前の Microsoft データセット・プロパティに対応します。互換性の理由から、既定値は False です。

# 12.4 データセットとスキーマの XML としての表示

%XML.DataSet を拡張するデータセットには、XML の生成に使用できるユーティリティ・メソッドがあります。これらのメソッドはすべて現在のデバイスに書き込みます。

- WriteXML() は、XML としてデータセットを書き込みます。オプションで、XML スキーマが前に付きます。このメソッドには、最上位の要素の名前、ネームスペースの使用、NULL の扱いなどを制御するオプションの引数があります。前のセクションの設定で指定されているように、既定では、このメソッドはデータセットの形式を考慮します。出力が DiffGram 形式かどうかなどを制御するオプションの引数に値を指定することによって、この結果を上書きできます。詳細は、%XML.DataSet のクラス・ドキュメントを参照してください。

- XMLExport() は、データセットの XML スキーマを書き込み、その後ろに XML としてのデータセットを書き込みます。
- WriteSchema() は、データセットの XML スキーマのみを書き込みます。
- XMLSchema() は、データセット・クラスの Microsoft 専用の XML 表現を書き込みます。

XML 対応のオブジェクトからの XML スキーマの生成については、"XML ツールの使用法" を参照してください。

# 12.5 WSDL への影響

InterSystems IRIS Web サービスで **%XML.DataSet** を Web メソッドに対する入力または出力として使用する場合、WSDL に影響を与え、InterSystems IRIS および .NET 以外のクライアントで WSDL の利用に問題が発生します。

typed データセットの場合、WSDL では (**<types>** セクション内に) 以下の要素が含まれます。

```
<s:element name="GetDataSetByNameResponse">
   <s:complexType>
      <s:sequence>
        <s:element name="GetDataSetByNameResult" type="s0:ByNameDataSet" />
      </s:sequence>
   </s:complexType>
</s:element>
<s:complexType name="ByNameDataSet">
   <s:sequence>
     <s:any namespace="http://tempuri.org/ByNameDataSet" />
   </s:sequence>
</s:complexType>
```

untyped データセットの場合、WSDL では以下の要素が含まれます。

```
<s:element name="GetByNameResponse">
   <s:complexType>
      <s:sequence>
         <s:element name="GetByNameResult" type="s0:DataSet" />
      </s:sequence>
   </s:complexType>
</s:element>
<s:complexType name="DataSet">
   <s:sequence>
      <s:element ref="s:schema" />
   <s:any />
   </s:sequence>
</s:complexType>
```

後者の場合、InterSystems IRIS または .NET 以外のツール内で Web クライアントを生成しようとすると、そのツールに関する十分な情報がないためにエラーが発生します。Metro の場合は、WSDL を利用しようとする前に、追加のスキーマ情報をロードできます。そのためには、**wsimport** と呼ばれるコマンドライン・ツールを使用できます。この手法によって、クライアントを生成するのに十分な情報を提供することができます。

ただし、どの場合でも、クライアントが適切な形式でメッセージを解釈または生成できるようにコードを記述するには、相当の作業が必要です。

# 13

# InterSystems IRIS での Web サービスの調整

このトピックでは、InterSystems IRIS Web サービスを調整するさまざまな方法を説明します。

基本情報の詳細は、"SOAP Web サービスの作成" の "Web サービスのパラメータの指定" を参照してください。

"オブジェクトの XML への投影" の "XML スキーマへの投影の制御" で、"コレクション・プロパティの XML スキーマへの投影" に記載されている ALLOWREDUNDANTARRAYNAME Web サービス・クラスのパラメータも参照してください。

## 13.1 オンライン WSDL へのアクセスの無効化

既定では、以下の形式の URL によって InterSystems IRIS Web サービスの WSDL を表示できます。

```
base/csp/app/web_serv.cls?WSDL
```

base は Web サーバのベース URL (必要に応じてポートも指定)、/csp/app は Web サービスが存在する Web アプリケーションの名前、web_serv は Web サービスのクラス名です。

この方法で WSDL にアクセスする機能を無効にするには、Web サービスの SOAPDISABLEWSDL パラメータを 1 に指定します。SOAPDISABLEWSDL が 1 の場合でも、FileWSDL() メソッドを使用して WSDL を静的ファイルとして生成することができる点に注意してください。

## 13.2 ユーザ名およびパスワードの要求

パスワードを要求するように InterSystems IRIS Web サービスを構成するには、親の Web アプリケーションを構成して、パスワード認証が使用されると同時に、認証されていないアクセスが許可されないようにします。

## 13.3 XML タイプの制御

WSDL は、引数に XML タイプを定義し、Web サービスのすべてのメソッドの値を返します。InterSystems IRIS Web サービスの場合、タイプは次のように指定されます。

- InterSystems IRIS タイプが、単純なタイプ (%String など) に対応する場合、対応する適切な XML タイプが使用されます。

- InterSystems IRIS タイプが XML 対応クラスに対応する場合、そのクラスの XMLTYPE パラメータによって XML タイプの名前が指定されます。このパラメータが指定されていない場合は、クラス名 (パッケージなし) が XML タイプ名として使用されます。

  また、WSDL は対応するクラス定義の情報を使用することによってこのタイプを定義します。

- InterSystems IRIS タイプがその他のクラスに対応する場合、そのクラス名 (パッケージなし) が XML タイプ名として使用されます。また、WSDL は、このタイプを定義しません。

詳細は、"オブジェクトの XML への投影" を参照してください。

"InterSystems IRIS における WSDL のサポート" も参照してください。

# 13.4 スキーマとタイプのネームスペースの制御

ここでは、WSDL のスキーマのネームスペース、およびスキーマ内で定義されるすべてのタイプのネームスペースの制御方法について説明します。

## 13.4.1 スキーマのネームスペースの制御

Web サービスの TYPENAMESPACE パラメータは、Web サービスのスキーマのターゲット・ネームスペースを制御します。

TYPENAMESPACE が NULL の場合、スキーマは、Web サービスの NAMESPACE パラメータによって指定されるネームスペースに配置されます。WSDL は以下のようになります。

```
<?xml version='1.0' encoding='UTF-8' ?>
...
<types>
<s:schema elementFormDefault='qualified'
targetNamespace = 'http://www.myapp.org'>
...
```

TYPENAMESPACE を URI に設定すると、タイプのネームスペースとしてその URI が使用されます。この場合、WSDL は以下のようになります。

```
<?xml version='1.0' encoding='UTF-8' ?>
...
<types>
<s:schema elementFormDefault='qualified'
targetNamespace = 'http://www.mytypes.org'>
...
```

## 13.4.2 タイプのネームスペースの制御

スキーマ内で参照されるすべてのタイプで、ネームスペースへの割り当て方法に以下の規則が適用されます。

- Web サービスの USECLASSNAMESPACES パラメータが 0 (既定) の場合、タイプのネームスペースはスキーマと同じになります。前のセクションを参照してください。

- Web サービスの USECLASSNAMESPACES パラメータが 1 の場合 (さらに、Web サービスがドキュメント・バインディング・スタイルを使用する場合)、それぞれのタイプは、対応するタイプ・クラスの NAMAESPACE パラメータで指定されるネームスペースに配置されます。

  特定のタイプで、そのタイプ・クラスの NAMESPACE パラメータが NULL の場合、そのタイプはスキーマと同じネームスペースに配置されます。前のセクションを参照してください。

  バインディング・スタイルの詳細は、"バインディング・スタイル" を参照してください。

# 13.5 タイプのドキュメントの追加

既定では、Web サービスの WSDL には、Web サービスで使用するタイプのドキュメントは含まれません。

WSDL のスキーマの <annotation> 要素内にタイプのクラス・ドキュメントを追加するには、Web サービスの INCLUDEDOCUMENTATION パラメータを 1 に指定します。

このパラメータによって WSDL が Web サービスおよびその Web メソッドにコメントを追加することはありません。それらのコメントを自動的に WSDL に追加するためのオプションはありません。

# 13.6 SOAP エンベロープへのネームスペース宣言の追加

指定の Web サービスによって送信された SOAP メッセージの SOAP エンベロープ (<SOAP-ENV:Envelope> 要素) にネームスペース宣言を追加するには、その Web サービスの各 Web メソッドを変更して、それが Web サービスの %AddEnvelopeNamespace() メソッドを呼び出すようにします。このメソッドには、以下のシグニチャがあります。

```
Method %AddEnvelopeNamespace(namespace As %String,
                             prefix As %String,
                             schemaLocation As %String,
                             allowMultiplePrefixes As %Boolean) As %Status
```

以下はその説明です。

- namespace は追加するネームスペースです。
- prefix はこのネームスペースで使用するオプションの接頭語です。この引数を省略すると、接頭語が生成されます。
- schemaLocation はこのネームスペースのためのオプションのスキーマの場所です。
- allowMultiplePrefixes は、指定されたネームスペースが異なる接頭語で複数回宣言可能かどうかを制御します。この引数が 1 の場合、指定されたネームスペースは異なる接頭語で複数回宣言可能です。この引数が 0 の場合、同じネームスペースに異なる接頭語で複数の宣言を追加すると、最後に指定された接頭語のみが使用されます。

# 13.7 必要な要素および属性のチェック

既定では、InterSystems IRIS Web サービスは Required とマークされたプロパティに対応する要素と属性が存在するかどうかをチェックしません。Web サービスでそのような要素と属性の存在をチェックするようにするには、Web サービスの SOAPCHECKREQUIRED パラメータを 1 に設定します。互換性の理由から、このパラメータの既定値は 0 です。

# 13.8 NULL 文字列の引数が持つ形式の制御

通常、引数を省略すると、InterSystems IRIS Web サービスから送信する SOAP メッセージでは、対応する要素が省略されます。これを変更するには、Web サービス・クラスで XMLIGNORENULL パラメータを 1 に設定します。この場合、SOAP メッセージには空の要素が含まれます。

注釈　　このパラメータは、タイプが %String の Web メソッド引数にのみ作用します。

# 13.9 SOAP 応答のメッセージ名の制御

Web メソッドから受け取った応答で使用するメッセージ名を制御できます。既定では、メッセージ名は、Web メソッド名の末尾に `Response` を追加したものです。次の例は、`Divide` という Web メソッドからの応答を示しています。応答メッセージの名前は、`DivideResponse` です。

XML

```
<SOAP-ENV:Body>
   <DivideResponse xmlns="http://www.myapp.org">
       <DivideResult>.5</DivideResult>
   </DivideResponse>
</SOAP-ENV:Body>
```

別の応答メッセージ名を指定するには、Web メソッド定義内に SoapMessageName キーワードを設定します。

指定された Web メソッドを呼び出す SOAP メッセージの名前は変更できません。このメッセージの名前は、メソッドの名前です。ただし、HTTP 要求での指定に従い、SOAP アクションをオーバーライドできます。後述の ″HTTP SOAP アクションおよび要求メッセージ名のオーバーライド″ を参照してください。

# 13.10 HTTP SOAP アクションおよび要求メッセージ名のオーバーライド

HTTP 経由で Web メソッドを呼び出す場合、HTTP ヘッダには SOAP アクションが含まれている必要があります。これは、SOAP HTTP 要求の目的を示す URI です。SOAP 1.1 の場合、SOAP アクションは、`SOAPAction` HTTP ヘッダとして含まれます。SOAP 1.2 の場合は、`Content-Type` HTTP ヘッダ内に含まれます。

SOAP アクションは、SOAP HTTP 要求の目的を示すものです。値は、目的を特定する URI です。通常は、着信 SOAP メッセージの転送に使用されます。例えば、ファイアウォールは、HTTP での SOAP 要求メッセージを適切にフィルタするためにこのヘッダを使用できます。

InterSystems IRIS で作成される Web メソッドの場合、`SOAPAction` HTTP ヘッダの形式は、既定で次のようになります (SOAP 1.1 の場合)。

```
SOAPAction: NAMESPACE/Package.Class.Method
```

NAMESPACE は、Web サービスの NAMESPACE パラメータの値です。Package.Class.Method は Web メソッドとして使用するメソッドの名前です。以下はその例です。

```
SOAPAction: http://www.myapp.org/GSOAP.WebService.GetPerson
```

これをオーバーライドするには、Web メソッドの定義内で SoapAction メソッドのキーワードに値を指定します。SOAP 要求の目的を識別する引用符付きの文字列として指定します。一般的なシナリオでは、Web サービス内の各 Web メソッドは SoapAction に一意の値 (ある場合) を指定します。

SoapAction がこの Web サービス内で一意でない場合、各メソッドは SoapRequestMessage メソッド・キーワードの一意の値を持っている必要があります。このキーワードにより、要求メッセージの SOAP 本文内に最上位要素名を指定します。SoapRequestMessage は、wrapped document/literal メッセージにのみ影響することに注意してください。

# 13.11 要素が修飾されるかどうかの指定

Web サービスの ELEMENTQUALIFIED パラメータは、WSDL のスキーマの `elementFormDefault` 属性の値を制御します。具体的には以下のとおりです。

- ELEMENTQUALIFIED が 1 の場合、`elementFormDefault` は `"qualified"` です。
- ELEMENTQUALIFIED が 0 の場合、`elementFormDefault` は `"unqualified"` です。

このパラメータの既定の設定は、SoapBodyUse クラス・キーワードの値によって異なります。"クラス定義リファレンス" を参照してください。通常、SoapBodyUse は "literal" です。これは、ELEMENTQUALIFIED が 1 であることを意味します。

修飾要素と未修飾要素の違いとその例については、"オブジェクトの XML への投影" を参照してください。

# 13.12 メッセージ部分で要素とタイプのどちらを使用するかの制御

Web サービスには、SOAP メッセージのメッセージ・パートの的確な形式を制御する別のパラメータ (XMLELEMENT) があります。具体的には以下のとおりです。

- XMLELEMENT が 1 の場合、`<part>` 要素には、`name` と `element` という属性が含まれます。この場合、WSDL には以下のようにサンプルの `<message>` 要素が含まれます。

  XML

  ```
  <message name="GetPersonSoapOut">
    <part name="GetPersonResult" element="s0:Person" />
  </message>
  ```

- XMLELEMENT が 0 の場合、`<part>` 要素には、`name` と `type` という属性が含まれます。この場合、WSDL には以下のようにサンプルの `<message>` 要素が含まれます。

  XML

  ```
  <message name="GetPersonSoapOut">
    <part name="GetPersonResult" type="s0:Person" />
  </message>
  ```

このパラメータの既定の設定は、SoapBodyUse クラス・キーワードの値によって異なります。"クラス定義リファレンス" を参照してください。通常、SoapBodyUse は "literal" です。これは、XMLELEMENT が 1 であることを意味します。

# 13.13 xsi:type 属性の使用の制御

既定では、InterSystems IRIS SOAP メッセージには、最上位タイプの場合にのみ xsi:type 属性が追加されます。以下はその例です。

```
<?xml version="1.0" encoding="UTF-8" ?>
...
<types:GetPersonResponse>
<GetPersonResult href="#id1" />
</types:GetPersonResponse>
<types:Person id="id1" xsi:type="types:Person">
<Name>Yeats,Clint C.</Name>
<DOB>1944-12-04</DOB>
</types:Person>
...
```

これらの例では、見やすくするために改行を追加してあります。SOAP メッセージ内のすべてのタイプでこの属性を使用するには、OUTPUTTYPEATTRIBUTE パラメータまたは OutputTypeAttribute プロパティを 1 に設定します。いずれの場合も以下のように出力されます。

```
<?xml version="1.0" encoding="UTF-8" ?>
...
<types:GetPersonResponse>
<GetPersonResult href="#id1" />
</types:GetPersonResponse>
<types:Person id="id1" xsi:type="types:Person">
<Name xsi:type="s:string">Yeats,Clint C.</Name>
<DOB xsi:type="s:date">1944-12-04</DOB>
</types:Person>
...
```

このパラメータは Web サービスの WSDL には影響しません。

# 13.14 エンコード形式でのインライン参照の使用の制御

エンコードされた形式では、任意のオブジェクト値プロパティが参照として含められ、参照オブジェクトが SOAP メッセージに個別の要素として書き込まれます。

代わりに、エンコードされたオブジェクトをインラインで書き込む場合は、REFERENCESINLINE パラメータまたは ReferencesInline プロパティに 1 を指定します。

プロパティはパラメータよりも優先されます。

# 13.15 SOAP エンベロープ接頭語の指定

既定では、InterSystems IRIS Web サービスは、送信する SOAP メッセージのエンベロープで接頭語 SOAP-ENV を使用します。別の接頭語を指定できます。そのためには、Web サービスの SOAPPREFIX パラメータを設定します。例えば、このパラメータを MYENV に設定すると、次に示すように、Web サービスのメッセージにこの接頭語が追加されます。

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<MYENV:Envelope xmlns:MYENV='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:s='http://www.w3.org/2001/XMLSchema'>
  <MYENV:Body>
   <DivideResponse xmlns="http://www.myapp.org">
      <DivideResult>.5</DivideResult>
   </DivideResponse>
  </MYENV:Body>
</MYENV:Envelope>
```

SOAPPREFIX パラメータは、Web サービスで生成されるすべての SOAP フォルトで使用される接頭語にも影響を及ぼします。

このパラメータは Web サービスの WSDL には影響しません。

# 13.16 Web サービスで処理する SOAP バージョンの制限

既定では、InterSystems IRIS Web サービスは SOAP バージョン 1.1 または 1.2 を使用する SOAP 要求を処理できます。Web サービスを変更して特定の SOAP バージョンに対する SOAP 要求のみを処理できるようにするには、REQUESTVERSION パラメータを設定します。このパラメータは、"1.1"、"1.2"、または "" に設定できます。このパラメータが "" の場合、Web サービスは既定の動作になります。

SOAPVERSION パラメータは Web サービスによってサポートされるバージョンに影響しないことに注意してください。このパラメーターは、単に WSDL 内で通知されるバージョンを制御するします。

# 13.17 gzip で圧縮された応答の送信

InterSystems IRIS Web サービスでは応答メッセージを gzip で圧縮できます。gzip はインターネット上で広く提供されている無償の圧縮プログラムです。他の何らかのメッセージのパッケージ化 (MTOM パッケージの作成など) の後、この圧縮が実行されます。Web サービスでこの圧縮を実行するには、GZIPOUTPUT パラメータを 1 に設定します。

このパラメータは Web サービスの WSDL には影響しません。

この変更を行う場合は、対応する解凍プログラムである gunzip を使用して Web クライアント側でメッセージが自動解凍できることを確認してください。

Web クライアントが InterSystems IRIS Web クライアントの場合は、Web クライアントへ送信する前の着信メッセージが Web ゲートウェイによって自動的に解凍されます。

# 13.18 単方向 Web メソッドの定義

通常、Web メソッドを実行すると、メソッドに返りタイプがなく、InterSystems IRIS での実行時に何も返さない場合であっても、SOAP メッセージは返されます。この SOAP 応答メッセージの一般的な形式は以下のとおりです。

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:s='http://www.w3.org/2001/XMLSchema'>
  <SOAP-ENV:Body>
   <MethodNameResponse xmlns="http://www.myapp.org"></MethodNameResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

まれに、一方向に Web メソッドを定義することが必要な場合があります。そのようなメソッドは値を返すことがなく、要求メッセージに SOAP 応答は必要ではありません。単方向の Web メソッドを定義するには、メソッドの返りタイプを %SOAP.OneWay として定義します。この場合は以下のようになります。

- WSDL は、この Web メソッドに対して定義される出力を定義しません。
- Web サービスは、SOAP メッセージを返しません (サービスによってヘッダ要素が追加される場合は除く。サブセクションを参照)。つまり、HTTP 応答メッセージには XML コンテンツが含まれません。

注釈　通常、単方向メソッドは使用されません。返りタイプがないメソッドの場合でも、要求と応答のペアを使用する方法のほうがより一般的であり、広くサポートされ、必要とされています。

"生成された WSDL の詳細" の "単方向 Web メソッドの WSDL の相違点" を参照してください。

## 13.18.1 単方向の Web メソッドおよび SOAP ヘッダ

Web メソッドによってヘッダ要素が追加される場合、HTTP 応答には以下のような XML コンテンツが含まれます。

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/' ...
  <SOAP-ENV:Header>
     header elements as set by the web service
 </SOAP-ENV:Header>
  <SOAP-ENV:Body></SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## 13.18.2 動的に Web メソッドを単方向にする方法

Web メソッドが単方向になるように動的に再定義することもできます。そのためには、Web メソッドの定義内で Web サービスの ReturnOneWay() を呼び出します。以下はその例です。

Class Member

```
Method HelloWorldDynamic(oneway as %Boolean = 0) As %String [ WebMethod ]
{
  If oneway {Do ..ReturnOneWay() }
  Quit "Hello world "
}
```

引数が 0 の場合、この Web メソッドは、本文に "Hello world" という文字列が含まれる SOAP 応答を返します。この引数が 1 の場合、このメソッドは SOAP 応答を返しません。

# 13.19 バイナリ・データへの改行の追加

InterSystems IRIS Web サービスで、%Binary または %xsd.base64Binary タイプのプロパティに自動改行を含めるようにできます。そのためには、以下のいずれかを実行します。

- Web サービス・クラスで、BASE64LINEBREAKS パラメータを 1 に設定します。
- Web サービス・クラス・インスタンスに対して、Base64LineBreaks プロパティを 1 に設定します。このプロパティの値は、BASE64LINEBREAKS パラメータによって設定される値より優先されます。

パラメータおよびプロパティの場合、既定値は 0 です。既定では、InterSystems IRIS Web サービスには %Binary または %xsd.base64Binary タイプのプロパティに対する自動改行は含まれません。

# 13.20 SOAP メッセージへのバイト・オーダー・マークの追加

既定では、InterSystems IRIS Web サービスによって送信されるメッセージの先頭に BOM (バイト・オーダー・マーク) はありません。

メッセージはバイト・オーダーの問題がない UTF-8 としてエンコードされるため、通常 BOM は必要ありません。ただし、SOAP メッセージに BOM を組み込むことが必要であったり、推奨される状況があります。この BOM は単にメッセージが UTF-8 であることを示すものです。

InterSystems IRIS Web サービスで送信されるメッセージに BOM を追加するには、サービスの RequestMessageStart プロパティを設定します。このプロパティは、メッセージの先頭に組み込むパーツのコンマ区切りのリストにする必要があります。これらのパーツは、以下のとおりです。

- DCL は、XML 宣言です。

  ```
  <?xml version="1.0" encoding="UTF-8" ?>
  ```

- BOM は、UTF-8 BOM です。

既定は "DCL" です。

実際には、RequestMessageStart を以下の値のいずれかにすることができます。

- "DCL"
- "BOM"
- "BOM,DCL"

# 13.21 タイムアウト時間のカスタマイズ

Web ゲートウェイは、InterSystems IRIS Web サービスからの応答メッセージの送信を一定の時間待機します。タイムアウト時間の設定の詳細は、"Web ゲートウェイ・ガイド" の "Web ゲートウェイの既定パラメータの構成" を参照してください。

ある状況では、特定の Web メソッドが完了できるまでに長い時間が必要であることがわかっている場合があります。この場合、そのメソッドのタイムアウト時間を指定できます。これには、その Web メソッドの定義の先頭付近に、Web サービスの Timeout プロパティを設定する行を追加します。秒単位でタイムアウト期間を指定します。例えば、既定のタイムアウト時間が 3 分で、そのタイムアウト時間を 5 分にする必要がある場合は、以下のように指定できます。

```
Method LongRunningMethod(Input) as %Status [ WebMethod ]
{
   set ..Timeout=300; this method will not time out until 5 minutes
   //method implementation here
}
```

# 13.22 プロセス・プライベート・グローバルを使用して非常に大きいメッセージをサポートする方法

既定では、InterSystems IRIS Web サービスは通常、要求や応答を解析するときにローカル配列メモリを使用します。代わりに、プロセス・プライベート・グローバルを強制的に使用させることができます。これにより、Web サービスは非常に大きいメッセージを処理できるようになります。

そのためには、以下のように Web サービス・クラスの USEPPGHANDLER パラメータを指定します。

```
Parameter USEPPGHANDLER = 1;
```

このパラメータが 1 の場合、Web サービスは常に、要求や応答を解析するときにプロセス・プライベート・グローバルを使用します。このパラメータが 0 の場合、Web サービスは常に、この目的のためにローカル配列メモリを使用します。このパラメータが設定されていない場合、Web サービスは既定の動作になります。既定の動作は通常ローカル配列メモリの使用です。

# 13.23 Web サービスのコールバックのカスタマイズ

コールバック・メソッドをオーバーライドすることによって、InterSystems IRIS Web サービスの動作をカスタマイズできます。

### OnRequestMessage()

Web サービスが要求メッセージを受信したときに、セキュリティ・エラーがない場合に呼び出されます。このコールバックは、セキュリティ・エラーが発生した場合には呼び出されません。システムは、セキュリティ処理の実行後、エンベロープのエラーのチェック後、および WS-Addressing ヘッダ内に指定されているアクションの処理後 (ある場合) に、このコールバックを呼び出します。このコールバックは、未加工の SOAP 要求のログを記録するなどのタスクで役立ちます。

このメソッドには、以下のシグニチャがあります。

```
Method OnRequestMessage(mode As %String, action As %String, request As %Stream.Object)
```

以下はその説明です。

- mode は、SOAP 要求のタイプを指定します。これは、"SOAP" または "binary" のいずれかを指定します。
- action は、SOAPAction ヘッダの値を格納します。
- request は、ストリームの SOAP 要求メッセージを格納します。

このメソッドは、%CSP.Session のインスタンスであるオブジェクト %request を使用できます。このオブジェクトの内容は以下のとおりです。

- Content プロパティには、未加工の要求メッセージが格納されます。
- NextMimeData() インスタンス・メソッドによって、個々の MIME パートの取得が可能になります (MIME SOAP 要求の場合)。

このメソッドは、Web サービス・インスタンスのプロパティも使用できます。以下のプロパティは初期化の際に設定されます。

- ImportHandler プロパティには、解析済み SOAP メッセージの DOM が格納されます。

- SecurityIn プロパティには、WS-Security ヘッダ要素が格納されます。詳細は、"Web サービスの保護" を参照してください。
- SecurityNamespace プロパティには、WS-Security ヘッダ要素のネームスペースが格納されます。
- SOAP フォルトが生成された場合、SoapFault プロパティが設定されます。

OnRequestMessage() 内でフォルトを返すには、SoapFault プロパティを設定します。ReturnFault() メソッドは呼び出さないでください。

OnPreWebMethod()

Web メソッドが実行される直前に呼び出され、既定では何もしません。このメソッドは引数を取らないので値を返すことができません。このため、Web メソッドと同じ方法で SOAP フォルトを返すことを除き、このメソッドでは Web サービスの実行を変更できません。

このメソッドは、%request、%session、および Web サービスのプロパティを使用できます。Web サービスの MsgClass プロパティは、Web メソッドの引数を含むメッセージ記述子クラスです。

OnPostWebMethod()

Web メソッドの実行直後に呼び出され、既定では何もしません。このメソッドは引数を取らないので値を返すことができません。このため、このメソッドは Web メソッドの実行を変更したり、値を返したりできません。主に、OnPreWebMethod() によって作成される必要な構造を削除するために、このメソッドをカスタマイズします。

# 13.24 Web サービスのカスタム転送の指定

ここで説明するように、既定では、InterSystems IRIS Web サービスは、指定した方法で転送するよう応答します。この動作をカスタマイズできます。

## 13.24.1 背景

InterSystems IRIS Web サービスが SOAP メッセージを受け取ると、この Web サービスは、OnSOAPRequest() クラス・メソッドを実行します。既定では、このメソッドは以下を実行します。

1. Initialize() メソッドを呼び出すことにより、Web サービス・インスタンスを初期化します。このメソッドは、着信 SOAP メッセージを解析し、情報の複数の部分を参照によって返し、セキュリティ・ヘッダを処理します。%SOAP.WebService クラスのドキュメントを参照してください。
2. SoapFault などの Web サービス・インスタンスのプロパティを設定します。
3. 応答ストリームを初期化します。
4. Web サービスの Process() メソッドを呼び出して、このメソッドに SOAP アクションおよび呼び出すメソッドを渡します。
5. Reset() メソッドを呼び出すことにより、Web サービス・インスタンスをリセットします。
6. 結果を応答ストリームにコピーします。

## 13.24.2 Web サービスのカスタム転送の定義

独自の転送を使用して Web サービスを実装するには、転送を使用して SOAP メッセージをストリームとして取得し、Web サービス・クラスをインスタンス化して、その OnSOAPRequest() クラス・メソッドを呼び出します。

OnSOAPRequest() メソッドは、要求を Web サービスに転送して、応答を取得する必要があります。エラーを示す場合は、応答ストリームで SOAP フォルトを返します。このメソッドのシグニチャは、以下のようにする必要があります。

```
Method OnSOAPRequest(action,requestStream, responseStream)
```

以下はその説明です。

1. action は、SOAP アクションを指定する **%String** です。最後の "." の後のアクション文字列の部分は、正しい記述子クラスを使用するためのメソッド名として使用されます。アクションが NULL の場合、SOAP 本文の最初の要素 (ラップ要素) の要素名が、メソッド名として使用されます。
2. requestStream は、XML 指示文のエンコード属性に従ってエンコードされた SOAP 要求メッセージを含むストリームです。
3. responseStream は、UTF-8 でエンコードされた応答 SOAP メッセージを含む SOAP 応答として生成された文字ストリームです。OnSOAPRequest() を呼び出す前にこの引数を作成して、このメソッド呼び出しでこの引数を渡すことができます。または、この引数を、参照によって渡される変数にすることもできます。この場合、OnSOAPRequest() では、応答が組み込まれた **%FileCharacterStream** のインスタンスと等しくなるように、これを設定する必要があります。

# 13.25 Web サービスのカスタム処理の定義

まれに、着信メッセージの処理と応答メッセージの作成にカスタム処理を使用する InterSystems IRIS Web サービスを定義すると便利な場合があります。これらのシナリオでは、ProcessBodyNode() メソッドか ProcessBody() メソッドのいずれかを Web サービスに実装します。このセクションでは、詳細について説明します。

## 13.25.1 概要

カスタム処理では、手動で着信メッセージを解析して応答を作成します。要件は以下のとおりです。

- Web サービスで、必要なシグニチャを持つ Web メソッドを定義します。これは、Web サービスの WSDL を設定するために行います。それらの Web メソッド (またはその一部) は、スタブとすることができます。メソッドは、ProcessBodyNode() または ProcessBody() が 0 を返す場合のみ実行されます。
- また、Web サービスに、以下のメソッドのいずれかを実装します。
  - ProcessBodyNode() ー このメソッドは、SOAP 本文を **%XML.Node** のインスタンスとして受け取ります。InterSystems IRIS XML ツールを使用してこのインスタンスを扱い、応答メッセージを作成します。SOAP エンベロープは、**%XML.Node** のこのインスタンスの **Document** プロパティにあります。
  - ProcessBody() ー このメソッドは、SOAP 本文をストリームとして受け取ります。SOAP 本文は XML ドキュメントではなく XML フラグメントであるため、InterSystems IRIS XML ツールをその読み取りに使用することはできません。代わりに、ObjectScript 関数を使用してそのストリームを解析し、必要な部分を抽出します。

  これらのメソッドを両方とも定義する場合、ProcessBodyNode() メソッドは無視されます。

いずれの場合も、作成する応答メッセージは Web サービスの WSDL と整合性がある必要があります。

## 13.25.2 ProcessBodyNode() の実装

ProcessBodyNode() メソッドには、以下のシグニチャがあります。

```
method ProcessBodyNode(action As %String, body As %XML.Node,
       ByRef responseBody As %CharacterStream) as %Boolean
```

以下はその説明です。

- action は、着信メッセージに指定されている SOAP アクションです。
- body は、SOAP <Body> を含む %XML.Node のインスタンスです。
- responseBody は、%Library.CharacterStream のインスタンスとしてシリアル化された応答本文です。このストリームは参照によって渡され、最初は空です。

このメソッドを Web サービスに実装する場合、このメソッドで以下を実行する必要があります。

1. action および分岐を適切に検証します。以下はその例です。

   ObjectScript

   ```
   if action["action1" {
    //details
   }
   ```

2. SOAP <Envelope> にアクセスする必要がある場合は (例えば、そのネームスペース宣言にアクセスするなど)、body の Document プロパティを使用します。これは、SOAP エンベロープを DOM (ドキュメント・オブジェクト・モデル) として表現する %XML.Document のインスタンスと等しくなります。

   それ以外の場合は、body を直接使用します。

3. ここで、以下のオプションがあります。

   - %XML.Writer を使用して本文を文字列として記述し、操作できるようにします。以下はその例です。

     ObjectScript

     ```
     set writer=##class(%XML.Writer).%New()
     do writer.OutputToString()
     do writer.DocumentNode(body)
     set request=writer.GetXMLString(.sc)
     // check returned status and continue
     ```

   - 必要に応じて、%XML.Document または %XML.Node のメソッドを使用し、ドキュメントをナビゲートします。同様に、%XML.Document または %XML.Node のプロパティを使用し、ドキュメントの現在の部分に関する情報にアクセスします。
   - XPath 式を使用して、データを抽出します。
   - XSLT 変換を実行します。

   詳細は、"XML ツールの使用法" を参照してください。これらのクラス内のメソッドによって返されるステータスをチェックし、エラーが発生した場合のトラブルシューティングを簡素化するようにします。

4. 要求の処理中にエラーが発生したら、ReturnFault() メソッドを使用する通常の方法でフォルトを返します。
5. 応答ストリームの Write() メソッドを使用して、<Body> の子要素となる XML フラグメントを記述します。
6. 応答ストリームが作成される場合は、1 を返します。それ以外の場合は 0 を返しますが、これによって、指定されたアクションに関連付けられた Web メソッドが実行されます。

   以下はその例です。

   ObjectScript

   ```
   if action["action1" {
     //no custom processing for this branch
     quit 0
    } elseif action["action2" {
       //details
       //quit 1
    }
   ```

## 13.25.3 ProcessBody() の実装

ProcessBody() メソッドには、以下のシグニチャがあります。

```
method ProcessBody(action As %String, requestBody As %CharacterStream,
                   ByRef responseBody As %CharacterStream) as %Boolean
```

以下はその説明です。

- action は、着信メッセージに指定されている SOAP アクションです。
- requestBody は、SOAP <Body> 要素を含む %Library.CharacterStream のインスタンスです。ストリームには、完全な XML ドキュメントではなく、XML フラグメントが含まれます。
- responseBody は、%Library.CharacterStream のインスタンスとしてシリアル化された応答本文です。このストリームは参照によって渡され、最初は空です。

このメソッドを Web サービスに実装する場合、このメソッドで以下を実行する必要があります。

1. action および分岐を適切に検証します。以下はその例です。

   ObjectScript

   ```
   if action["action1" {
    //details
   }
   ```

2. requestBody の Read() メソッドを使用して、SOAP <Body> を取得します。以下はその例です。

   ObjectScript

   ```
   set request=requestBody.Read()
   ```

3. $EXTRACT などのツールを使用して、このストリームを解析します。以下はその例です。

   ObjectScript

   ```
   set in1="<echoString xmlns=""http://soapinterop.org/xsd""><inputString>"
   set in2="</inputString></echoString>"
   set contents=$extract(request,$length(in1)+1,*-$length(in2))
   ```

4. 要求の処理中にエラーが発生したら、ReturnFault() メソッドを使用する通常の方法でフォルトを返します。
5. 応答ストリームの Write() メソッドを使用して、<Body> の子要素となる XML フラグメントを記述します。以下はその例です。

   ObjectScript

   ```
   set in1="<echoString xmlns=""http://soapinterop.org/xsd""><inputString>"
   set in2="</inputString></echoString>"
   set request=requestBody.Read()
   if ($extract(request,1,$length(in1))'=in1) || ($extract(request,*-$length(in2)+1,*)'=in2) {
     do responseBody.Write("Bad Request: "_request)
     quit 1
   }

   set out1="<echoStringResponse xmlns=""http://soapinterop.org/xsd""><echoStringResult>"
   set out2="</echoStringResult></echoStringResponse>"
   do responseBody.Write(out1)
   do responseBody.Write($extract(request,$length(in1)+1,*-$length(in2)))
   do responseBody.Write(out2)
   ```

6. 応答ストリームが作成される場合は、1 を返します。それ以外の場合は 0 を返しますが、これによって、指定されたアクションに関連付けられた Web メソッドが実行されます。

# 14

# InterSystems IRIS での Web クライアントの調整

通常は、InterSystems IRIS Web クライアント・クラスを生成した後で、そのクラスを編集することはありません。その代わりに、そのクラスのインスタンスを作成するコードと、クライアント側のエラー処理を提供するコードを記述します。このトピックでは、InterSystems IRIS Web クライアントを調整するさまざまな方法について説明します。この調整方法は、Web クライアントのインスタンスを変更するか、一般的ではありませんが生成されたクラスを変更するかのどちらかになります。

注釈　生成された Web クライアント・クラスのサブクラスは作成しないでください。コンパイラは、適切な実行に必要なサポート・クラスを生成しないため、そのサブクラスは使用できなくなります。

## 14.1 Web クライアントのキープ・アライブを無効にする方法

既定では、InterSystems IRIS Web クライアント・インスタンスを再利用して複数の要求メッセージを送信すると、InterSystems IRIS は (HTTP 1.1 キープ・アライブ接続を使用して) 1 つの HTTP 転送ですべてのメッセージを送信します。具体的には、InterSystems IRIS は TCP/IP ソケットを閉じてから再び開く必要がないように、開いたままにします。このキープ・アライブ動作を無効にするには、以下のいずれかを実行します。

- Web クライアントのインスタンスを削除し、新しいインスタンスを作成して使用します。
- 最初のメッセージの送信後に、クライアントの HttpRequest.SocketTimeout プロパティを 0 に設定します。以下はその例です。

```
Set client.HttpRequest.SocketTimeout=0
```

注釈　WS-ReliableMessaging メッセージを使用し、SSL/TLS を使用して Web サーバと通信するには、キープ・アライブを無効にしないでください。WS-ReliableMessaging については、"Web サービスの保護" を参照してください。

## 14.2 NULL 文字列の引数が持つ形式の制御

通常、引数を省略すると、InterSystems IRIS Web クライアントから送信する SOAP メッセージでは、対応する要素が省略されます。これを変更するには、Web クライアント・クラスで XMLIGNORENULL パラメータを 1 に設定します。この場合、SOAP メッセージには空の要素が含まれます。

注釈　このパラメータは、タイプが %String の Web メソッド引数にのみ作用します。

# 14.3 クライアントのタイムアウトの制御

InterSystems IRIS Web クライアントに対して、2 つの異なるタイムアウト期間を制御できます。

- Web クライアントの Timeout プロパティは、読み取りのタイムアウトを意味します。このプロパティは、Web クライアントが応答を待機する長さを秒単位で指定します。

  このプロパティが指定されていない場合、Web クライアントは、%Net.HttpRequest クラスの Timeout プロパティで指定されている既定値を使用します。この既定値は 30 秒です。

  プロキシ・サーバを使用している場合は、このプロパティにより、プロキシからの応答をクライアントが待機する長さが制御されます。

- OpenTimeout プロパティは、オープン・タイムアウトを指定します。これは、TCP/IP 接続のオープンを待機する秒数です。このプロパティが指定されていない場合は、Timeout によって指定された値が使用されます。

# 14.4 プロキシ・サーバの使用法

InterSystems IRIS Web クライアントは、プロキシ・サーバ経由で Web サービスと通信できます。これを設定するために、使用するプロキシ・サーバを示すように Web クライアント・インスタンスのプロパティを指定します。これらのプロパティは、以下のとおりです。

### HttpProxyServer

使用するプロキシ・サーバのホスト名を指定します。このプロパティが NULL でない場合、HTTP 要求はこのマシンに向けられます。

既定のプロキシ・サーバの指定については、"インターネット・ユーティリティの使用法" の "プロキシ・サーバの使用法" を参照してください。

### HttpProxyPort

プロキシ・サーバ上の接続先ポートを指定します。

既定のプロキシ・ポートの指定については、"インターネット・ユーティリティの使用法" の "プロキシ・サーバの使用法" を参照してください。

### HttpProxyHTTPS

プロキシ・サーバを使用している場合、およびそのプロキシ・サーバで HTTPS がサポートされる場合は、これを True に指定します。

HTTPS を使用している場合は、クライアントの SSLConfiguration プロパティを SSL/TLS 構成の名前に設定する必要もあります。セキュリティの詳細は、"SSL を使用するようにクライアントを構成する方法" を参照してください。

### HttpProxyAuthorization

Web クライアントがそのクライアント自体をプロキシ・サーバで認証する必要がある場合は、これを必須の `Proxy-Authorization` ヘッダ・フィールドとして指定します。

HttpProxyTunnel

Web クライアントが、ターゲットの HTTP サーバへのプロキシ経由のトンネルを確立する必要がある場合は、これを True に指定します。True の場合、要求は HTTP CONNECT コマンドを使用してトンネルを確立します。プロキシ・サーバのアドレスは、HttpProxyServer プロパティと HttpProxyPort プロパティから取得されます。エンドポイントの URL で https: プロトコルを使用している場合は、トンネルが確立されたときに InterSystems IRIS が SSL 接続をネゴシエートします。この場合、トンネルによってターゲット・システムとの直接接続が確立されるため、HttpProxyHTTPS プロパティは無視されます。

# 14.5 HTTP ヘッダの設定

Web クライアントによって送信される HTTP ヘッダをさらに制御する必要がある場合は、%SOAP.WebClient の次のメソッドを使用できます。

SetHttpHeader()

HTTP 要求にヘッダを追加します。`Content-Type`、`Content-Encoding`、および `Content-Length` の各ヘッダは、HTTP のメイン・ヘッダではなく、本文の一部です。`Content-Length` ヘッダは読み取り専用なので、設定することはできません。`Connection` ヘッダも設定できません。このクラスは、永続接続をサポートしていないためです。

ResetHttpHeaders()

すべての HTTP ヘッダをクリアします。

"HTTP 認証の使用法" も参照してください。

# 14.6 使用する HTTP バージョンの指定

既定では、InterSystems IRIS Web クライアントは HTTP/1.1 を使用します。代わりに HTTP/1.0 を使用することができます。そのためには、クライアントの HttpVersion プロパティを `"1.0"` に設定します。

# 14.7 SSL サーバ名チェックの無効化

既定では、InterSystems IRIS Web クライアントは、SSL を介してサーバに接続するときに、証明書サーバ名と、そのサーバへの接続に使用した DNS 名が一致していることを確認します (このチェックは、RFC 2818 セクション 3.1 で説明されています。ワイルドカード・サポートは RFC 2595 で説明されていますが、ブラウザは一般的にサーバ名をチェックし、InterSystems でも同様に実行します)。

このチェックを無効にするには、クライアントの SSLCheckServerIdentity プロパティを `0` に設定します。

# 14.8 xsi:type 属性の使用の制御

既定では、InterSystems IRIS SOAP メッセージには、最上位タイプの場合にのみ xsi:type 属性が追加されます。以下はその例です。

```
<?xml version="1.0" encoding="UTF-8" ?>
...
<types:GetPersonResponse>
<GetPersonResult href="#id1" />
</types:GetPersonResponse>
<types:Person id="id1" xsi:type="types:Person">
<Name>Yeats,Clint C.</Name>
<DOB>1944-12-04</DOB>
</types:Person>
...
```

これらの例では、見やすくするために改行を追加してあります。この属性を SOAP メッセージに含まれるすべてのタイプに使用するには、以下のどちらかを実行します。

- Web クライアントのインスタンスで、OutputTypeAttribute プロパティを 1 に設定します。
- Web クライアント・クラスで、OUTPUTTYPEATTRIBUTE パラメータを 1 に設定します。

同じ出力は以下のようになります。

```
<?xml version="1.0" encoding="UTF-8" ?>
...
<types:GetPersonResponse>
<GetPersonResult href="#id1" />
</types:GetPersonResponse>
<types:Person id="id1" xsi:type="types:Person">
<Name xsi:type="s:string">Yeats,Clint C.</Name>
<DOB xsi:type="s:date">1944-12-04</DOB>
</types:Person>
...
```

プロパティはパラメータよりも優先されます。

# 14.9 エンコード形式でのインライン参照の使用の制御

エンコードされた形式では、任意のオブジェクト値プロパティが参照として含められ、参照オブジェクトが SOAP メッセージに個別の要素として書き込まれます。

代わりに、エンコードされたオブジェクトをインラインで書き込む場合は、ReferencesInline パラメータまたはReferencesInline プロパティに 1 を指定します。プロパティはパラメータよりも優先されます。

# 14.10 エンベロープ接頭語の指定

既定では、InterSystems IRIS Web クライアントは、送信する SOAP メッセージのエンベロープで接頭語 SOAP-ENV を使用します。別の接頭語を指定できます。そのためには、Web クライアント・クラスの SOAPPREFIX パラメータを設定します。例えば、このパラメータを MYENV に設定すると、SOAP-ENV ではなく、この接頭語が Web クライアントのメッセージに使用されます。

# 14.11 SOAP エンベロープへのネームスペース宣言の追加

指定の Web クライアントによって返された SOAP 応答の SOAP エンベロープ (<SOAP-ENV:Envelope> 要素) にネームスペース宣言を追加するには、Web メソッドを呼び出す前に Web クライアントの %AddEnvelopeNamespace() メソッドを呼び出します。このメソッドには、以下のシグニチャがあります。

```
Method %AddEnvelopeNamespace(namespace As %String,
                             prefix As %String,
                             schemaLocation As %String,
                             allowMultiplePrefixes As %Boolean) As %Status
```

以下はその説明です。

- namespace は追加するネームスペースです。
- prefix はこのネームスペースで使用するオプションの接頭語です。この引数を省略すると、接頭語が生成されます。
- schemaLocation はこのネームスペースのためのオプションのスキーマの場所です。
- allowMultiplePrefixes は、指定されたネームスペースが異なる接頭語で複数回宣言可能かどうかを制御します。この引数が 1 の場合、指定されたネームスペースは異なる接頭語で複数回宣言可能です。この引数が 0 の場合、同じネームスペースに異なる接頭語で複数の宣言を追加すると、最後に指定された接頭語のみが使用されます。

# 14.12 gzip で圧縮された応答の送信

InterSystems IRIS Web クライアントでは応答メッセージを gzip で圧縮できます。gzip はインターネット上で広く提供されている無償の圧縮プログラムです。他の何らかのメッセージのパッケージ化 (MTOM パッケージの作成など) の後、この圧縮が実行されます。これを Web クライアントで実行するには、以下のどちらかを実行します。

- Web クライアントのインスタンスで、GzipOutput プロパティを 1 に設定します。
- Web クライアント・クラスで、GZIPOUTPUT パラメータを 1 に設定します。

この操作時は、Web サービス側で対応する解凍プログラムである gunzip を使用してメッセージが自動解凍できることを確認してください。(Web サービスが InterSystems IRIS Web サービスの場合は、Web サービスへ送信する前の着信メッセージが Web ゲートウェイによって自動的に解凍されます。)

# 14.13 SOAP アクションに対する引用符の使用 (SOAP 1.1 のみ)

SOAP 1.1 の要求メッセージでは、HTTP ヘッダに次のような SOAPAction 行が含まれます。

```
POST /csp/gsop/GSOP.DivideWS.cls HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; InterSystems IRIS;)
Host: localhost:8080
Connection: Close
Accept-Encoding: gzip
SOAPAction: http://www.mynamespace.org/GSOAP.DivideWS.Divide
Content-Length: 397
Content-Type: text/xml; charset=UTF-8

...
```

既定では、SOAPAction の値は引用符で囲まれません。この値を引用符で囲むには、Web クライアント・クラスで SOAPACTIONQUOTED に 1 を指定します。この指定により、要求メッセージの HTTP ヘッダが次のようになります。

```
POST /csp/gsop/GSOP.DivideWS.cls HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; InterSystems IRIS;)
Host: localhost:8080
Connection: Close
Accept-Encoding: gzip
SOAPAction: "http://www.mynamespace.org/GSOAP.DivideWS.Divide"
Content-Length: 397
Content-Type: text/xml; charset=UTF-8

...
```

SOAP 1.2 では、SOAPACTIONQUOTED パラメータは機能しません。これは、要求メッセージに SOAPAction 行がないためです。次に示すように、SOAP アクションは必ず引用符で囲まれて Content-Type 行に記述されます。

```
Content-Type: application/soap+xml;
   charset=UTF-8; action="http://www.mynamespace.org/GSOAP.DivideWS.Divide"
```

注釈　この例では、PDF 形式にしたこのコンテンツでページに行が適切に表示されるように、意図的な改行を適用しています。

# 14.14 HTTP のステータス 202 をステータス 200 と同じように処理する方法

既定では、HTTP 応答に SOAP エンベロープが含まれていない場合、InterSystems IRIS Web クライアントは、HTTP 応答ステータス 202 のみを使用する標準の WS-I Basic Profile に従います。

HTTP ステータス 202 を HTTP ステータス 200 と同じように処理する場合は、クライアントの HttpAccept202 プロパティを 1 に設定します。実際に返されるステータスを確認するには、クライアントの HttpResponse.StatusCode プロパティをチェックします。

WS-I Basic Profile では、この方法をサポートしていますが、お勧めはしません。“Profile が両方のステータス・コードを受け入れるのは、一部の SOAP 実装が HTTP プロトコル実装をほとんど制御できず、これらの応答ステータス・コードのいずれを送信するか制御できないためです。“

# 14.15 単方向 Web メソッドの定義

通常、Web クライアントが Web サービスを呼び出すと、メソッド自体に返りタイプがなく、InterSystems IRIS での実行時に何も返さない場合であっても、SOAP メッセージが返されます。

まれに、一方向に Web メソッドを定義することが必要な場合があります。そのようなメソッドは値を返すことがなく、メッセージに SOAP 応答は必要ではありません。

注釈　通常、単方向メソッドは使用されません。返りタイプがないメソッドの場合でも、要求と応答のペアを使用する方法のほうがより一般的であり、広くサポートされ、必要とされています。

単方向の Web メソッドを定義するには、メソッドの返りタイプを %SOAP.OneWay として定義します。WSDL では、この Web メソッドに対して定義される出力が定義されません。また、Web サービスでは、SOAP メッセージが返されません。

# 14.16 バイナリ・データへの改行の追加

InterSystems IRIS Web サービスで、%Binary または %xsd.base64Binary タイプのプロパティに自動改行を含めるようにできます。そのためには、以下のいずれかを実行します。

- Web サービス・クラスで、BASE64LINEBREAKS パラメータを 1 に設定します。
- Web サービス・クラス・インスタンスに対して、Base64LineBreaks プロパティを 1 に設定します。このプロパティの値は、BASE64LINEBREAKS パラメータによって設定される値より優先されます。

パラメータおよびプロパティの場合、既定値は 0 です。既定では、InterSystems IRIS Web サービスには %Binary または %xsd.base64Binary タイプのプロパティに対する自動改行は含まれません。

# 14.17 SOAP メッセージへのバイト・オーダー・マークの追加

既定では、InterSystems IRIS Web クライアントによって送信されるメッセージの先頭に BOM (バイト・オーダー・マーク) はありません。

メッセージはバイト・オーダーの問題がない UTF-8 としてエンコードされるため、通常 BOM は必要ありません。ただし、SOAP メッセージに BOM を組み込むことが必要であったり、推奨される状況があります。この BOM は単にメッセージが UTF-8 であることを示すものです。

InterSystems IRIS Web クライアントで送信されるメッセージに BOM を追加するには、クライアントの RequestMessageStart プロパティを設定します。このプロパティは、メッセージの先頭に組み込むパーツのコンマ区切りのリストにする必要があります。これらのパーツは、以下のとおりです。

- DCL は、XML 宣言です。

```
<?xml version="1.0" encoding="UTF-8" ?>
```

- BOM は、UTF-8 BOM です。

既定は "DCL" です。

実際には、RequestMessageStart を以下の値のいずれかにすることができます。

- "DCL"
- "BOM"
- "BOM,DCL"

# 14.18 解析時にプロセス・プライベート・グローバルを使用する方法

既定では、InterSystems IRIS Web クライアントは通常、要求や応答を解析するときにローカル配列メモリを使用します。代わりに、プロセス・プライベート・グローバルを強制的に使用させることができます。これにより、Web クライアントは非常に大きいメッセージを処理できるようになります。

そのためには、以下のように Web サービス・クラスの USEPPGHANDLER パラメータを指定します。

```
Parameter USEPPGHANDLER = 1;
```

このパラメータが 1 の場合、Web クライアントは常に、要求や応答を解析するときにプロセス・プライベート・グローバルを使用します。このパラメータが 0 の場合、Web クライアントは常に、この目的のためにローカル配列メモリを使用します。このパラメータが設定されていない場合、Web クライアントは既定の動作になります。既定の動作は通常ローカル配列メモリの使用です。

このパラメータは実行時にオーバーライドできます。そのためには、Web クライアント・インスタンスの UsePPGHandler プロパティを設定します。

# 14.19 カスタム SOAP メッセージの作成

特別な場合として、Web クライアントでカスタム SOAP メッセージの送信が必要なことがあります。基本的な要件は以下のとおりです。

1. %SOAP.WebRequest のサブクラスを作成し、その LOCATION パラメータまたは Location プロパティを設定します。
2. このサブクラスで、SOAP メッセージを送信するメソッドを作成します。このメソッドは、%Library.CharacterStream のインスタンスを作成し、送信する SOAP メッセージをそのインスタンスに配置します。メッセージの形式が正しいことを確認する必要があります。
3. 次に、このメソッドによって SendSOAPBody() メソッドが起動されます。

```
method SendSOAPBody(Action As %String,
                    OneWay As %Boolean = 0,
                    Request As %CharacterStream,
                    ByRef Response) as %Status
```

   - Action は、実行する SOAP アクションの名前を指定する文字列です。
   - OneWay は、メッセージが単方向かどうかを制御する true/false フラグです。
   - Request は、現在のロケールの文字セットで SOAP 要求の本文を組み込む %Library.CharacterStream のインスタンスです。
   - Response は、参照によって文字ストリームか、%XML.Node のインスタンスとして返される応答です。

     SendSOAPBody() を呼び出したときに Response が NULL である場合は、メソッドによって Response が %Library.CharacterStream のインスタンスに設定されます。このストリームには、現在のロケールの文字セットで SOAP 応答の本文が格納されます。

     SendSOAPBody() を呼び出したときに Response が %Library.CharacterStream のインスタンスである場合は、メソッドによって Response が更新され、現在のロケールの文字セットで SOAP 応答の本文が組み込まれます。

     SendSOAPBody() を呼び出したときに Response が %XML.Node のインスタンスである場合は、メソッドによって Response が更新され、本文 DOM がポイントされます。

%SOAP.WebRequest は %SOAP.WebClient のサブクラスであるため、他のパラメータおよびプロパティの設定が必要な場合があります。SOAP ヘッダを追加することもできます。これについては、別途説明しています。詳細は、%SOAP.WebRequest のクラス・ドキュメントを参照してください。

# 14.20 カスタムの HTTP 要求の指定

既定では、InterSystems IRIS Web クライアントを使用する場合、Web クライアントでは HTTP を使用して SOAP メッセージが Web サービスに転送され、その応答が受信されます。Web クライアントでは、自動的に HTTP 要求が作成されて、送信されますが、カスタムの HTTP 要求を作成することができます。そのためには、次の手順を実行します。

1. %Net.HttpRequest のインスタンスを作成し、必要に応じてプロパティを設定します。このクラスについては、"インターネット・ユーティリティの使用法" を参照するか、%Net.HttpRequest のクラス・ドキュメントを参照してください。
2. Web クライアントの HttpRequest プロパティをこのインスタンスと等しくなるように設定します。

これは、同一セッション内で SOAP サービスへの複数呼び出しをサポートする場合に特に便利です。既定では、InterSystems IRIS Web クライアントでは、同一セッションを使用した SOAP サービスへの複数呼び出しはサポートされません。この問題に対処するには、%Net.HttpRequest のインスタンスを新規作成し、それを Web クライアントの HttpRequest プロパティとして使用します。この変更により、同一の HTTP 要求がすべての呼び出しで再使用されるように強制され、次の要求に対する応答ですべての cookie が返されます。

# 14.21 Web クライアントのコールバックのカスタマイズ

コールバック・メソッドをオーバーライドすることによって、InterSystems IRIS Web クライアントの動作をカスタマイズできます。

### %OnSOAPRequest()

```
Method %OnSOAPRequest(mode As %String,
                      client As %SOAP.WebClient,
                      action As %String,
                      oneWay As %Boolean,
                      method As %String,
                      requestStream As %BinaryStream)
```

Web クライアントが (実際の SOAP 要求を作成する) 転送クラスの DoSOAPRequest() メソッドを呼び出す直前に呼び出されます。既定の DoSOAPRequest() メソッドは、%SOAP.WebClient に含まれていて、要求/応答に HTTP を使用します。

- mode は、SOAP 要求のタイプ ("SOAP" または "binary") を指定します。
- client は、Web クライアント・インスタンスの OREF です。
- action は、SOAPAction ヘッダの値を格納します。
- oneWay は、送信される本文がない場合には true です。
- method 引数は、呼び出されている Web メソッドの名前です。
- requestStream 引数は、ストリームの SOAP 要求メッセージを格納します。

### %OnSOAPResponse()

```
Method %OnSOAPResponse(mode As %String,
                       client As %SOAP.WebClient,
                       action As %String,
                       oneWay As %Boolean,
                       method As %String,
                       requestStream As %BinaryStream,
                       responseStream As %BinaryStream,
                       sc As %Status)
```

Web クライアントが転送クラスの DoSOAPRequest() メソッドを呼び出した後に呼び出されます。sc 引数は、転送クラスの DoSOAPRequest() メソッドによって返されたステータスです。その他の引数は、%OnSOAPRequest() のものと同じです。

%OnSOAPFinished()

```
Method %OnSOAPFinished(mode As %String, client As %SOAP.WebClient, method As %String, sc As
%Status)
```

Web クライアントがその処理をすべて実行した後に呼び出されます。sc 引数は、呼び出された Web メソッドによって返されたステータスです。mode 引数、client 引数、および method 引数は、その他のコールバック・メソッドのものと同じです。

# 14.22 Web クライアントからのカスタム転送の指定

既定では、InterSystems IRIS Web クライアントを使用する場合、Web クライアントでは HTTP を使用して SOAP メッセージが Web サービスに転送され、その応答が受信されます。ユーザ独自の転送クラスを定義し、使用できます。

## 14.22.1 背景

使用する Web サービスと通信するには、InterSystems IRIS Web クライアントに転送クラスが必要です。転送クラスは、通信に関連するパラメータ、プロパティ、およびメソッドを備えています。全体的な通信の動作は、以下のとおりです。

1. Web クライアント・プロキシ・メソッドが実行されている場合、Web クライアント・インスタンスは Transport プロパティの値をチェックします。

   このプロパティが NULL の場合、Web クライアント・インスタンスは、それ自体を転送クラスのインスタンスとして使用します。このようなクラスを定義した場合、代わりに、Transport プロパティを他の適切なクラスのインスタンスと等しくなるように設定できます。

2. Web クライアント・インスタンスは、転送クラスの DoSOAPRequest() メソッドを実行して、次の引数を渡します。

   a. Web クライアント・クラスの OREF

   b. SOAP アクションを指定する文字列

   c. UTF-8 でエンコードされた要求を含むストリーム

   d. 応答を含むストリーム (参照による)

3. Web クライアント・インスタンスは、結果のステータスをチェックし、それに応じて動作します。

HTTP 転送の場合、DoSOAPRequest() メソッドには以下のロジックがあります。

1. 要求オブジェクト (%Net.HttpRequest のインスタンス) を作成し、そのプロパティを設定します。ここで、このメソッドは、Web クライアント・インスタンスのプロパティの値を使用します。特に HttpRequestHeaderCharset プロパティとその他の HTTP 関連のプロパティの値を使用します。
2. SOAP 要求のヘッダを確認した上で要求オブジェクトのヘッダを初期化します。
3. 要求オブジェクトの Post() メソッドを実行します。これは、HTTP 転送に適したアクションです。
4. 応答を取得し、それを返します。

重要　%SOAP.WebClient の DoSOAPRequest() メソッドは直接的に使用しないでください。この動作および以降の処理は保証されません。前述の概要は、一般的なヒントとして提供しています。

### 14.22.2 InterSystems IRIS Web クライアントのカスタム転送の定義

InterSystems IRIS Web クライアントでカスタム転送を使用できるようにするには、カスタムの転送クラスを定義します。次に、Web クライアントのインスタンスを作成した後に、その Transport プロパティを転送クラスのインスタンスと等しくなるように設定します。

転送クラスに関する要件は、以下のとおりです。

- 転送クラスは、インスタンス化可能 (つまり、非抽象) である必要があります。
- 転送クラスは、以下で説明するように DoSOAPRequest() メソッドを実装する必要があります。

DoSOAPRequest() メソッドは、要求を Web サービスに転送して、応答を取得します。このメソッドのシグニチャは、以下のようにする必要があります。

```
Method DoSOAPRequest(webClient,action,requestStream, responseStream) As %Status
```

- webClient は、Web クライアント・クラスの OREF です。
- action は、SOAP アクションを指定する %String です。
- requestStream は、UTF-8 でエンコードされた要求を含むストリームです。
- responseStream は、DoSOAPRequest() が応答の記述に使用する %FileBinaryStream 引数です。このストリームでは、?xml 指示文のエンコード属性で指定した文字セットにデータを含める必要があります。UTF-8 の使用をお勧めします。

## 14.23 SAX パーサのフラグ指定

InterSystems IRIS Web クライアントが Web サービスを呼び出すとき、Web クライアントは、InterSystems IRIS に付属のサードパーティ製品である SAX パーサを内部で使用します。使用するパーサにフラグを設定するために、Web クライアントの SAXFlags プロパティを設定できます。

パーサのフラグ自体については、"XML ツールの使用法" を参照してください。

## 14.24 WS-Security ログイン機能の使用法

認証を必要とする Web サービスが InterSystems IRIS Web クライアントで使用されており、新しい WS-Security 機能を使用する必要がない場合は、従来の単純な WS-Security ログイン機能を使用できます。

WS-Security ログイン機能を使用する手順は以下のとおりです。

1. Web クライアントと、Web サービスをホストする Web サーバの間で、SSL が使用されていることを確認します。WS-Security ヘッダがクリア・テキストで送信されるので、この手法では、SSL を使用しないとセキュリティで保護されません。"SSL を使用するようにクライアントを構成する方法" を参照してください。
2. Web クライアントの WSSecurityLogin() メソッドを呼び出します。このメソッドは、ユーザ名とパスワードを受け入れ、WS-Security ユーザ名トークンをクリア・テキストのパスワードを使用して生成し、WS-Security ヘッダを Web 要求に追加します。
3. Web メソッドを呼び出します。

この方法では、次の SOAP メッセージ 1 件にのみセキュリティ・トークンが追加されます。

新しい WS-Security 機能については、"Web サービスの保護" を参照してください。

# 14.25 HTTP 認証の使用法

Web サービスによっては、WS-Security ではなく HTTP 認証が必要なものがあります ("Web サービスの保護" で説明しています)。こういった Web サービスに向けて、InterSystems IRIS では以下の HTTP 認証スキームがサポートされています。

1. Negotiate (RFC 4559 および RFC 4178 で説明されている SPNEGO および Kerberos)
2. NTLM (NT LAN Manager 認証プロトコル)
3. Basic (RFC 2617 で説明されている Basic アクセス認証)

HTTP 1.0 では、Basic 認証のみが使用されることに注意してください。その他の認証スキームには、単一接続内での複数の往復が必要ですが、HTTP 1.0 ではこれが許可されないためです。

HTTP 認証を使用するには、以下の操作を行います。

- Web メソッドを呼び出す前に、Web クライアントの **HttpUsername** プロパティと **HttpPassword** プロパティを設定します。
- 使用するスキーム (このスキームはサーバによって許可されることがわかっている) を示した初期ヘッダがクライアントから送信されるようにするには、Web メソッドを呼び出す前に **HttpInitiateAuthentication** プロパティを設定します。このプロパティの値として、"インターネット・ユーティリティの使用法" の "HTTP 要求の送信" にある "ログイン資格情報の提供" の説明に従い、認証スキーム名を指定します。
- クライアントで試行するスキームのリストをカスタマイズするには、Web メソッドを呼び出す前に **HttpInitiateAuthentication** プロパティを設定します。このプロパティの値として、"インターネット・ユーティリティの使用法" の "HTTP 要求の送信" にある "ログイン資格情報の提供" の説明に従い、コンマ区切りの名前リストを使用します。

重要　Basic 認証が使用される可能性がある場合、Web サービスをホストする Web サーバと Web クライアントとの間で、必ず SSL を使用します。Basic 認証では、資格情報は Base 64 のエンコード形式で送信されるため、簡単に読み取ることができます。"SSL を使用するようにクライアントを構成する方法" を参照してください。

# 15

# InterSystems IRIS での SOAP の問題のトラブルシューティング

このトピックでは、InterSystems IRIS で SOAP に関する問題の原因を特定する際に役立つ情報を提供します。

明らかにセキュリティに関係する問題については、"Web サービスの保護" の "セキュリティの問題のトラブルシューティング" を参照してください。まれな状況として SOAP クライアントで HTTP 認証を使用する場合は、認証のログ記録を有効にすることができます。"インターネット・ユーティリティの使用法" の "HTTP 要求の送信" にある "ログイン資格情報の提供" を参照してください。

## 15.1 トラブルシューティングに必要な情報

SOAP の問題の原因を特定するには、通常、以下の情報が必要です。

- WSDL およびこれが参照するすべての外部ドキュメント。
- (メッセージ関連の問題の場合) 何らかの形式のメッセージ・ロギングおよびトレース。以下のオプションがあります。

| オプション | SSL/TLS で使用可能かどうか | HTTP ヘッダを表示するかどうか | コメント |
|---|---|---|---|
| InterSystems IRIS SOAP ログ | はい | 必要に応じて | セキュリティ・エラーの場合、このログは、SOAP フォルトに含まれるものよりも詳細を表示します。 |
| Web ゲートウェイ・トレース | はい | はい | MTOM (MIME 添付) を使用する SOAP メッセージの問題の場合、HTTP ヘッダを表示することは重要です。 |
| サードパーティのトレース・ツール | いいえ | ツールに依存 | トレース・ツールの中には、送信された実際のパケットなど、下位レベルの詳細を表示するものもあります。これらの情報は、トラブルシューティングの際に重要になる場合があります。 |

これらのオプションについては、以降のサブセクションで説明しています。

また、フォルトを正しく処理して最適な情報を受け取るようにすることは非常に有効です。"SOAP フォルトの処理" を参照してください。

### 15.1.1 InterSystems IRIS SOAP ログ

InterSystems IRIS ネームスペースとの間でやり取りされる SOAP 呼び出しをログに記録するには、そのネームスペースで ^ISCSOAP グローバルの次のノードを設定します。

| ノード | 目的 |
|---|---|
| ^ISCSOAP("Log") | ログに含めるデータのタイプを指定します。以下の値のいずれかを使用します (大文字と小文字が区別されます)。<br>・ "i" — 着信メッセージをログに記録します。<br>・ "o" — 発信メッセージをログに記録します。<br>・ "s" — セキュリティ情報をログに記録します。このオプションは、SOAP フォルトに通常格納されるものよりも詳細を提供します。SOAP フォルトは、セキュリティに対する後続の攻撃を防ぐために意図的にあいまいな表現になっています。<br>・ "h" — SOAP ヘッダのみをログに記録します。"h" を "i" や "o" と組み合わせる必要があります。"h" を "i" と共に使用する場合、ログには着信メッセージの SOAP エンベロープ要素とヘッダ要素のみが含まれます。同様に、"h" を "o" と共に使用する場合、ログには発信メッセージの SOAP エンベロープ要素とヘッダ要素のみが含まれます。対応する SOAP 本文要素はログに記録されません。<br>・ "H" — HTTP ヘッダをログに記録します。"H" を "i" や "o" と組み合わせる必要があります。"H" を "i" と共に使用する場合、ログには着信メッセージの HTTP ヘッダが含まれます。同様に、"H" を "o" と共に使用する場合、ログには発信メッセージの HTTP ヘッダが含まれます。SOAP データのほかに、HTTP ヘッダがログに記録されます。<br>例えば "iosh" のように、これらの値を任意に組み合わせた文字列も使用できます。 |
| ^ISCSOAP("LogFile") | 作成するログ・ファイルの完全なパスとファイル名を指定します。 |

ログには送信者または受信者が適切に示されるので、そのやり取りにどの Web サービスまたは Web クライアントが関係しているのかを確認することができます。

次に、ログ ファイルの一部の例を示します (この例では、読みやすいように改行が追加されています)。

```
01/05/2010 13:27:02 **********************
Output from web client with SOAP action = http://www.mysecureapp.org/GSOAP.AddComplexSecureWS.Add
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
...
  <SOAP-ENV:Header>
     <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">


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

```
**** Output HTTP headers for Web Client
User-Agent: Mozilla/4.0 (compatible; InterSystems IRIS;)
Host: hostid
Accept-Encoding: gzip


**** Input HTTP headers for Web Client
HTTP/1.1 200 OK
CACHE-CONTROL: no-cache
CONTENT-ENCODING: gzip
CONTENT-LENGTH: 479
CONTENT-TYPE: application/soap+xml; charset=UTF-8
...


01/05/2010 13:27:33 ********************
Input to web client with SOAP action = http://www.mysecureapp.org/GSOAP.AddComplexSecureWS.Add

ERROR #6059: Unable to open TCP/IP socket to server localhost:8080
string
```

以下の点に注意してください。

- InterSystems IRIS XML ツールを使用すると、署名が行われた XML ドキュメントのシグニチャを検証して、暗号化された XML ドキュメントを解読できます。これらのタスクをそのネームスペースで実行する場合、ログにはそれらのタスクの詳細も含まれます。"XML ツールの使用法" を参照してください。
- InterSystems IRIS SOAP ログは、何もメッセージが送信されない (つまり、サービスとクライアントが同じマシン上に存在する) 場合でも SOAP 呼び出しをキャプチャします。
- サーバ・エラーが発生した場合、SOAP ログへの書き込みは停止します。代わりにメッセージ・ログを参照してください。詳細は、"監視ガイド" の "ログ・ファイルの監視" を参照してください。
- タスク・マネージャの `CheckLogging` タスクは、SOAP のロギングが長時間放置される (既定では 2 日間) とアラートを作成します。

## 15.1.2 Web ゲートウェイの HTTP トレース

[ウェブゲートウェイ管理] ページでは、HTTP 要求と HTTP 応答をトレースできます。"Web ゲートウェイ・ガイド" の "HTTP トレース機能の使用法" を参照してください。

## 15.1.3 サードパーティのトレース・ツール

Web サービスのテストには、Wireshark、ProxyTrace、tcpTrace、XMLSpy、soapUI、Web Service Studio Express などのトレース・ツールを使用できます。これらのツールには、無償で利用できるものと、ライセンスが必要なものがあります。インターシステムズでは、これらのツールのいずれについてもその使用を特に推奨しているわけではありません。これらのツールの情報は一般情報として提供しています。

トレース・ツールを使用すると、実際のメソッド呼び出し、および応答を確認できます。トレース・セッションは、特定のポートで待ち受けし、そのポートで受信するメッセージを表示し、それらのメッセージを宛先ポートに転送し、応答を表示し、待ち受けるポートに応答を転送します。

例えば、`http://localhost:52773/csp/gsop/GSOP.Divide.CLS` に Web サービスがあるとします。

また、そのサービスと対話するために作成された Web クライアントがあるとします。この Web クライアントの LOCATION パラメータは、`"http://localhost:52773/csp/gsop/GSOP.Divide.CLS"` に等しくなります。

クライアントとサービスの間のメッセージをトレースするには、以下の 2 つの操作が必要です。

- トレース・ツールで、ポート 8080 (例) で待ち受けし、宛先ポート 52773 を使用するトレース・セッションを開始します。
- Web クライアントで、ポートとして 52773 の代わりに 8080 を使用するように LOCATION パラメータを編集し、再コンパイルします。

または、Web クライアントを呼び出すコードで、次のようにその Web クライアントの Location プロパティを変更します。

ObjectScript

```
//reset location to port 8080 to enable tracing
set client.Location="http://localhost:8080/csp/gsop/GSOP.DivideWS.cls"
```

これで、Web クライアントを使用するときに、トレース・ツールがクライアントと Web サービスの間のメッセージを傍受して表示するようになります。この例を次に示します。

```
POST /csp/gsop/GSOP.Divide.cls HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; IRIS;)
Host: localhost:8080
Connection: Close
SOAPAction: http://www.mynamespace.org/GSOP.Divide.Divide
Content-Length: 401
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
```

```
HTTP/1.1 200 OK
Date: Fri, 18 Apr 2008 15:52:33 GMT
Server: Apache
SET-COOKIE:
CSPSESSIONID-SP-8080-UP-csp-gsop-=00000001000026fgoIxI000000nRuHG3uXQjvSOYWJ7z2ZVw--;
path=/csp/gsop/;
CACHE-CONTROL: no-cache
EXPIRES: Thu, 29 Oct 1998 17:04:19 GMT
PRAGMA: no-cache
CONTENT-LENGTH: 378
Connection: close
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:s='http://www.w3.org/2001/XMLSchema'>
  <SOAP-ENV:Body>
<DivideResponse
xmlns="http://www.mynamespace.org"><DivideResult>.5</DivideResult></DivideResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

上の部分は、クライアントによって送信される要求を示しています。下の部分は、Web サービスによって送信される応答を示しています。

# 15.2 WSDL を利用する際の問題

次のような理由で、SOAP ウィザードで (または、対応する %SOAP.WSDL.Reader クラスを使用中に) エラーが表示されることがあります。

- 指定した WSDL URL では SSL 証明書を使用した認証が必要であるが、SSL 構成が指定されていないか、間違った SSL 構成が指定されている。この場合、次のようなエラー・メッセージが表示されます。

```
ERROR #6301: SAX XML Parser Error: invalid document structure
while processing Anonymous Stream at line 1 offset 1
```

  このエラーを修正するには、適切な SSL 構成を指定します。

- 指定した WSDL URL で、ユーザ名とパスワードを使用した認証が必要である。この場合、次のようなエラー・メッセージが表示されます。

```
ERROR #6301: SAX XML Parser Error: Expected entity name for reference
while processing Anonymous Stream at line 10 offset 27
```

  注釈　　行とオフセット値は、前のシナリオとは異なることがあります。

  このエラーを修正するには、"SOAP ウィザードの使用法" と "パスワードで保護された WSDL URL の使用法" の説明に従って、ユーザ名とパスワードを指定します。

- WSDL に外部で定義されたエンティティへの参照が含まれ、それをウィザードが 10 秒のタイムアウト時間の前に解決できない。この場合、次のようなエラー・メッセージが表示されます。

```
ERROR #6416: Element 'wsdl:definitions' - unrecognized wsdl element 'porttype'
```

  このエラーを修正するには、WSDL で次のような `<import>` および `<include>` 指示文を確認します。

```
<import namespace="http://example.com/stockquote/definitions"
           location="http://example.com/stockquote/stockquote.wsdl"/>
```

  指示文が見つかったら、以下の手順を実行します。

  1. プライマリ WSDL をファイルにダウンロードします。
  2. 参照する WSDL をファイルにダウンロードします。
  3. 参照する WSDL の新しい場所を参照するようにプライマリ WSDL を編集します。

  同様に、次のような相対 URL を使用して、WSDL が他のドキュメントを参照するかどうかを確認できます。

```
xmlns:acme="urn:acme.com.:acme:service:ServiceEndpointInterface"
```

  WSDL をファイルにダウンロードした場合、相対参照は使用できません。代わりに、参照先ドキュメントもダウンロードして、その新しい場所を指すように WSDL を編集する必要もあります。

- WSDL に複数のパートを伴う `<message>` 要素が含まれ、ドキュメント・スタイルのバインディングを使用する。この場合、次のようなエラー・メッセージが表示されます。

```
ERROR #6425: Element 'wsdl:binding:operation:msg:input' - message 'AddSoapOut'
Message Style must be used for document style message with 2 or more parts.
```

  このエラーを修正するには、ウィザードの使用時に **[ドキュメント形式のウェブ・メソッドで改行しないメッセージ形式を使用]** オプションを選択します。

- WSDL が無効。この場合、次のようなエラー・メッセージが表示されます。

```
ERROR #6419: Element 'wsdl:binding:operation' - inconsistent
soap:namespace for operation getWidgetInfo
```

エラー・メッセージは、WSDL の問題を明示しています。この例では、次の WSDL の抜粋の <operation> 要素がエラーを引き起こしました。

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://acme.acmecorp.biz:9999/widget/services"
                  xmlns="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" =
                 [parts omitted]>
  <wsdl:message name="getWidgetInfoRequest">
  </wsdl:message>
  <wsdl:message name="getWidgetInfoResponse">
    <wsdl:part name="getWidgetInfoReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:portType name="Version">
    <wsdl:operation name="getWidgetInfo">
      <wsdl:input message="impl:getWidgetInfoRequest" name="getWidgetInfoRequest"/>
      <wsdl:output message="impl:getWidgetInfoResponse" name="getWidgetInfoResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="VersionSoapBinding" type="impl:Version">
    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="getWidgetInfo">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getWidgetInfoRequest">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                       namespace="http://acmesubsidiary.com"
                       use="encoded"/>
      </wsdl:input>
      <wsdl:output name="getWidgetInfoResponse">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"

                       namespace="http://acme.acmecorp.biz:9999/widget/services"
                       use="encoded"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  [parts omitted]
```

この場合、<operation> の <input> パートで要求メッセージ (getVersionRequest) がネームスペース "http://acmesubsidiary.com" に含まれていることが示されているにもかかわらず、WSDL の前の部分でこのメッセージが "http://acme.acmecorp.biz:9999/widget/services" のように Web サービスのターゲット・ネームスペースであることが示されていることが問題です。

無効な WSDLドキュメントが、有効な XML ドキュメントである可能性もあるため、純粋な XML ツールを使用して WSDL を検証することは十分なテストとはいえません。いくつかのサードパーティ製 WSDL 検証ツールが提供されており、SOAP ウィザードから返される情報を使用して、WSDL を直接検証することもできます。

- WSDL に、InterSystems IRIS でサポートされていない機能が含まれている。詳細は、"WSDL の利用" を参照してください。

# 15.3 メッセージを送信する際の問題

InterSystems IRIS Web サービスまたは Web クライアントと SOAP メッセージを送受信する際に問題が発生した場合、以下の一般的なシナリオのリストを検討してください。

- SOAP メッセージに、文字列長の制限を超える、きわめて長い文字列またはバイナリ値が含まれている可能性がある。この場合、InterSystems IRIS によって以下のいずれかのエラーがスローされます。
  - 〈MAXSTRING〉 エラー
  - データ型検証エラー (これには、他の原因がある可能性もあります):

    ```
    ERROR #6232: Datatype validation failed for tag your_method_name ...
    ```

ウィザードで WSDL を読み取る場合、InterSystems IRIS では文字列タイプの入出力を %String として表現できることが前提となっています。同様に、InterSystems IRIS では XML タイプ base64Binary の入出力を %xsd.base64Binary として表現できることが前提となっています。WSDL 内には、この入出力が文字列長の制限を超えている可能性があることを SOAP ウィザードに通知するための情報が含まれていません。

"生成されたクラスをきわめて長い文字列に合わせて調整する方法" を参照してください。この情報は Web クライアントと Web サービスの両方に当てはまります。

- Web サービスまたは Web クライアントが WS-Security ヘッダを受け取った可能性があるが、それらを認識できるように構成されていない。この場合、以下のような汎用エラーが発行されます。

```
<ZSOAP>zInvokeClient+269^%SOAP.WebClient.1
```

このようなエラーには、他の原因がある可能性もあります。このようなエラーが発生した場合は、まずメッセージに WS-Security ヘッダが含まれていないか確認し、含まれていた場合には、以下を Web サービスまたは Web クライアントに追加してリコンパイルしてください。

### Class Member

```
Parameter SECURITYIN="REQUIRE";
```

また、InterSystems IRIS によって (構成クラス内に) セキュリティ・ポリシーが生成された場合は、足りない詳細を提供するためにそのポリシーの編集が必要になることがあります。"Web サービスの保護" の "生成されたポリシーの編集" を参照してください。これを行わないと、上記のような汎用エラーが発生します。

- Web サービスまたは Web クライアントが、SOAP 仕様に従って、必要とされるものよりもさらに具体的なメッセージ形式を要求している可能性がある (これは、InterSystems IRIS に含まれないサービスまたはクライアントの場合に発生する可能性があります)。インターシステムズでは、以下のシナリオが発生しました。ここでは、(ほぼ) 最も一般的なものから最も一般的でないものまでをリストしています。
  - Web サービスまたは Web クライアントで、メッセージがそのメッセージ内のすべての要素に xsi:type 属性を指定することを必要としている。この属性の使用を指定するには、"xsi:type 属性の使用の制御" を参照してください。この情報は、Web サービスと Web クライアントの両方に当てはまります。
  - NULL 文字列値に対して、Web サービスまたは Web クライアントが NULL 要素を (省略するのではなく) 要求している。この回避策として、NULL 文字列の引数の形式を制御できます。"NULL 文字列の引数が持つ形式の制御" を参照してください。この情報は、Web サービスと Web クライアントの両方に当てはまります。
  - Web サービスまたは Web クライアントが特定のネームスペース接頭語を要求している。InterSystems IRIS には、一般に、ネームスペース接頭語を指定する方法は用意されていません。

    ただし、SOAP エンベロープの場合は、使用する接頭語を指定できます。"SOAP エンベロープ接頭語の指定" を参照してください。この情報は、Web サービスと Web クライアントの両方に当てはまります。
  - Web クライアントが、SOAP アクションを引用符で囲むよう要求している。この回避策については、"SOAP アクションに対する引用符の使用 (SOAP 1.1 のみ)" を参照してください。
  - Web サービスまたはクライアントが、各 SOAP メッセージの冒頭に BOM (バイト・オーダー・マーク) を要求している。SOAP メッセージはバイト・オーダーの問題がない UTF-8 としてエンコードされるため、BOM は必要ありません。"SOAP メッセージへのバイト・オーダー・マークの追加" を参照してください。この情報は、Web サービスと Web クライアントの両方に当てはまります。

これらの問題を示す症状は、使用しているサードパーティ製品によって異なります。

- Web サービスまたは Web クライアントが WSDL に準拠していない可能性がある。これは、InterSystems IRIS Web サービスや Web クライアントでは発生しませんが、他のシナリオで発生することが考えられます。インターシステムズでは、以下のシナリオを認識しています。
  - メッセージに含まれる要素が WSDL によって要求されるネームスペース内にない。

- メッセージに含まれる要素が WSDL と同じ順序でない。

サービスまたはクライアントが WSDL に準拠しているかどうかを確認するには、メッセージを WSDL と照合します。

または、サードパーティの Web サービスの場合、その Web サービスが WSDL に準拠しているかどうかを確認するには、以下を実行すると便利です。

1. サードパーティ製ツールを使用して、Web クライアントを作成します。
2. メッセージをその Web クライアントから送信します。
   - これが成功した場合、Web サービスはその WSDL と整合性のあるメッセージを想定および送信し、問題の原因が他にあることが考えられます。その場合、このクライアントから送信されたメッセージを InterSystems IRIS クライアントから送信されたメッセージと照合します。
   - これが成功しなかった場合、Web サービスはその WSDL と整合性のあるメッセージを想定および送信しないことが考えられます。

- Web サービスまたは Web クライアントが、InterSystems IRIS でサポートされていない形式のメッセージを送信する可能性がある。使用している WSDL を検証し、それが InterSystems IRIS でサポートされていることを確認することは有用です。"WSDL の利用" を参照してください。これらの詳細は、InterSystems IRIS で徐々に変更されてきていることに注意してください。

# A

# Web サービスの URL の概要

このトピックは、InterSystems Web サービスに関連する URL をまとめたものです。

## A.1 Web サービスの URL

InterSystems IRIS Web サービスに関連する URL は次のとおりです。

**Web サービスのエンド・ポイント**

```
base/csp/namespace/web_serv.cls
```

以下はその説明です。

- base は、Web サーバのベース URL です (必要に応じてポートが含まれます)。
- /csp/namespace は、Web サービスが存在する Web アプリケーションの名前です。
- web_serv は、Web サービスのクラス名です。

以下はその例です。

```
http://localhost:52773/csp/samples/MyApp.StockService.cls
```

**WSDL**

```
base/csp/app/web_serv.cls&WSDL
```

以下はその例です。

```
http://localhost:52773/csp/samples/MyApp.StockService.cls?WSDL
```

これらの URL はいずれも、/csp/namespace Web アプリケーションの一部であることに注意してください。

## A.2 パスワードで保護された WSDL URL の使用法

既存の InterSystems IRIS Web サービスの WSDL URL を使用して、InterSystems IRIS またはサードパーティのツールで Web クライアントを作成できます。ただし、Web サービスの親 Web アプリケーションでパスワード認証が要求されてい

る場合、WSDL にアクセスするには、有効なユーザ名とパスワードを WSDL URL の中で指定する必要があります。それには、URL に `&IRISUsername=username&IRISPassword=password` を追加します。次に例を示します。

```
http://localhost:52773/csp/samples/MyApp.StockService.cls?WSDL&IRISUsername=
_SYSTEM&IRISPassword=SYS
```

さらに、サードパーティのツールを使用して Web クライアントを作成する際、ツールがログイン後に URL リダイレクトを使用する場合は、`&IRISNoRedirect=1` を追加する必要があります。例えば、ログイン後に、.NET が URL リダイレクトを実行するとします。この場合、.NET Web クライアントの WSDL URL 形式は次のようになります。

```
http://localhost:52773/csp/samples/MyApp.StockService.cls?WSDL&IRISUsername=
_SYSTEM&IRISPassword=SYS&IRISNoRedirect=1
```

数回試行しても、パスワードで保護された WSDL URL から Web クライアントを生成できない場合は、以下の代替案を検討します。

- 有効なユーザ名とパスワードを指定してブラウザから WSDL を取得し、WSDL をファイルとして保存して、そのファイルを使用して Web クライアントを生成します。
- Web サービスが WSDL への継続的なアクセスを提供する必要がある場合は、パスワードで保護されていない Web アプリケーションを作成して WSDL にアクセスできるようにします。
- CSP/ZEN を使用して WSDL を提供する従来のアプリケーションが存在し、その **[ログイン CSRF 攻撃を防ぐ]** 設定が有効になっている場合、安全と判断できるのであれば、一時的にこの設定を無効にします。詳細は、このタイプの従来のアプリケーションの設定に関するコンテンツを参照してください。

# B

# 生成された WSDL の詳細

このトピックでは、参考として、InterSystems IRIS Web サービスのサンプル WSDL ドキュメントから一部をいくつか取り上げ、キーワードとパラメータがそれらの部分に及ぼす影響に関する情報を示します。

Web メソッドのシグニチャも WSDL に影響しますが、このトピックではその詳細には触れません。

WSDL は、Web サービスで使用するあらゆる XML 対応クラスの XML プロジェクションの影響も受けます。"オブジェクトの XML への投影" を参照してください。

注釈　Web サービスにコンパイル済みのポリシー構成クラスがある場合、<binding> セクションには、<wsp:Policy> 形式の要素も含まれます。このドキュメントでは、ポリシーがどのように WSDL に影響するかについては触れません。それは WS-SecurityPolicy などの仕様によって決まるからです。

ポリシー構成については、"Web サービスの保護" を参照してください。

利便性のために WSDL ドキュメントが生成されますが、これは W3C 仕様では要求されていません。これに関する重要な注意事項は、"WSDL の表示" を参照してください。

## B.1 WSDL ドキュメントの概要

Web サービスには、マシンが読み取ることができる形式で記述されているインタフェース定義である WSDL ドキュメントがあります。WSDL ドキュメントは、Web サービス記述言語の規格に従って XML で記述されます。これにより、Web サービスとそのクライアントの対話方法に関するコントラクトが定義されます。

WSDL には、以下を定義する追加要素を記述したルート <definitions> 要素があります。

- Web サービスの入力または出力に必要な XML タイプまたは要素の定義。これは、基本の XML タイプに関して定義されます。<types> 要素には、XML タイプ、要素、またはその両方を必要に応じて定義する <schema> 要素が 1 つ以上記述されています。
- Web サービスで使用するメッセージの定義。各 Web メソッドには 1 つまたは 2 つのメッセージが必要です。Web メソッドを呼び出す要求メッセージと、応答で使用する応答メッセージです。各メッセージは、XML タイプまたは XML 要素に関して定義されます。
- Web サービスで使用するポート・タイプの定義。各ポートは、1 つまたは複数の処理を定義します。処理は、Web メソッドに対応しており、対応するメッセージを使用します。

  一般に、WSDL には複数の <portType> 要素を含めることができますが、InterSystems IRIS Web サービスの WSDL に含まれるのは 1 つだけです。

- Web サービスの bindings 要素。これは、特定のポート・タイプによって定義された処理とメッセージの、メッセージ形式とプロトコルの詳細を定義します。

  一般に、WSDL には複数の <binding> 要素を含めることができますが、InterSystems IRIS Web サービスの WSDL に含まれるのは 1 つだけです。

- 前述のコンポーネントに関する、Web サービスの正式な定義。この定義には、Web サービスを呼び出す URL などが含まれます。

サービス、スキーマ、およびメッセージはすべて XML ネームスペースに関連付けられます。これらをすべて 1 つのネームスペースに置くことも、複数のネームスペースに置くこともできます。SOAP に対する InterSystems IRIS サポートでは、想定可能なすべてのバリエーションがサポートされるわけではありません。"SOAP 標準" を参照してください。

# B.2 サンプルの Web サービス

このトピックでは、次のサンプル Web サービスの WSDL を抜粋して示します。

### Class Definition

```
Class WSDLSamples.BasicWS Extends %SOAP.WebService
{

Parameter SERVICENAME = "MyServiceName";

Parameter NAMESPACE = "http://www.mynamespace.org";

Parameter USECLASSNAMESPACES = 1;

///  adds two complex numbers
Method Add(a As ComplexNumber, b As ComplexNumber) As ComplexNumber [ WebMethod ]
{
    Set sum = ##class(ComplexNumber).%New()
    Set sum.Real = a.Real + b.Real
    Set sum.Imaginary = a.Imaginary + b.Imaginary

    Quit sum
}
}
```

この Web サービスは次のクラスに基づいています。

### Class Definition

```
///  A complex number
Class WSDLSamples.ComplexNumber Extends (%RegisteredObject, %XML.Adaptor)
{

/// real part of the complex number
Property Real As %Double;

/// imaginary part of the complex number
Property Imaginary As %Double;

}
```

特に明記のない限り、このトピックでは、この Web サービスの WSDL を抜粋して取り上げています (この Web サービスのバリエーションを使用している場合もあります)。

# B.3 ネームスペース宣言

WSDL の残りの部分を詳細に検証する前に、残りの部分によって使用されるネームスペース宣言を確認すると役に立ちます。<definitions> 要素には、WSDL で使用するネームスペースごとにネームスペース宣言が 1 つ記述されています。このトピックの最初に示したサンプルの Web サービスの場合、ネームスペース宣言は次のようになります。

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:s0="http://www.mynamespace.org"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
targetNamespace="http://www.mynamespace.org">
```

以下のパラメータはネームスペース宣言に影響します。

- Web サービスの NAMESPACE パラメータ。

  このパラメータは、Web サービスのターゲット・ネームスペースを示す targetNamespace 属性で使用します。

  NAMESPACE パラメータを指定していない場合、targetNamespace は "http://tempuri.org" になります。

- Web サービスの SOAPVERSION パラメータ。

  これは自動的に追加される SOAP ネームスペースに影響します。

  既定では、SOAPVERSION は 1.1 です。

  SOAPVERSION が 1.2 の場合、WSDL には次の内容が記述されます。

  ```
  <definitions ...
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
  ...
  ```

  SOAPVERSION が "" の場合、WSDL には次の内容が記述されます。

  ```
  <definitions
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
  ...
  ```

- Web サービスのその他のネームスペース・キーワードとパラメータ、および Web サービスで使用するあらゆる XML 対応クラスのネームスペース・キーワードとパラメータ

  これらの項目については以降のセクションで説明しています。

  これらのネームスペースは、WSDL の残りの部分との整合性を維持するために、必要に応じて宣言します。

また、その他のネームスペース (http://schemas.xmlsoap.org/wsdl/soap/ など) が、必要に応じて自動的に追加されます。

ネームスペースの接頭語は自動的に選択され、カスタマイズはできません。

# B.4 <service>

WSDL を検証する際、最後から開始して最初まで確認していくことは有用です。

WSDL 内の最後の要素は <service> 要素であり、これは Web サービスを定義するものです。このトピックの最初に示したサンプルの Web サービスの場合、この要素は次のようになります。

XML

```
<service name="MyServiceName">
    <port name="MyServiceNameSoap" binding="s0:MyServiceNameSoap">
        <soap:address location="http://localhost:52773/csp/gsop/WSDLSamples.BasicWS.cls"/>
    </port>
</service>
```

この要素は、次のように指定されます。

- Web サービスの SERVICENAME パラメータは、<service> 要素の name 属性として使用されます。"Web サービスのパラメータの指定" を参照してください。

  このパラメータは <port> 要素の name 属性と binding 属性にも影響します。単独で制御することはできません。

- binding 属性は、ネームスペース宣言でリストされている s0 ネームスペース内のバインディングを参照します。このネームスペースは、Web サービスの NAMESPACE パラメータによって指定されます。

- Web サービス・クラスの URL によって、<soap:address> 要素の location 属性が制御されます。

# B.5 <binding>

WSDL では、<service> 要素の前に、<binding> 要素が含まれます。これらはそれぞれ、特定の <portType> 要素によって定義された処理とメッセージの、メッセージ形式とプロトコルの詳細を定義します。

一般に、WSDL には複数の <binding> 要素を含めることができますが、InterSystems IRIS Web サービスの WSDL に含まれるのは 1 つだけです。

このトピックの最初に示したサンプルの Web サービスの場合、この要素は次のようになります。

XML

```
<binding name="MyServiceNameSoap" type="s0:MyServiceNameSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
                  style="document"/>
    <operation name="Add">
        <soap:operation soapAction="http://www.mynamespace.org/WSDLSamples.BasicWS.Add"
                        style="document"/>
        <input>
            <soap:body use="literal"/>
        </input>
        <output>
            <soap:body use="literal"/>
        </output>
    </operation>
</binding>
```

この要素は、次のように指定されます。

- <binding> 要素の name 属性は、自動的に <service> 要素と一致するようになります (変更しても意味がありません)。

  ```
  <binding name="MyServiceNameSoap"  ...
  ```

- <binding> 要素の type 属性は、ネームスペース宣言でリストされている s0 ネームスペース内の <portType> 要素を参照します。このネームスペースは、Web サービスの NAMESPACE パラメータによって指定されます。

- 各 <operation> 要素の name 属性は、Web メソッドの名前に基づきます (変更しても意味がありません)。

  ```
  <operation name="Add"> ...
  ```

- Web メソッドに SoapAction キーワードを指定した場合、その値は、処理の soapAction 属性に使用されます。以下はその例です。

  ```
  ...
  <operation name="Add">
      <soap:operation soapAction="mysoapaction" style="document"/>
  ...
  ```

- メソッドの返りタイプを %SOAP.OneWay として定義すると、この要素に影響します。"単方向 Web メソッドの WSDL の相違点" を参照してください。
- Web サービスで SOAPBINARY パラメータに 1 を指定すると、この要素に影響します。"InterSystems IRIS バイナリ SOAP 形式の WSDL の相違点" を参照してください。
- Web サービスで SOAPSESSION パラメータに 1 を指定すると、この要素に影響します。"InterSystems IRIS SOAP セッションの WSDL の相違点" を参照してください。

  SoapBindingStyle クラス・キーワード、SoapBindingStyle メソッド・キーワード、および SoapBindingStyle クエリ・キーワードは、<binding> 要素に影響します。これについては、"クラス定義リファレンス" に説明があります。これらのキーワードでは、document および rpc の値を指定できます。

- SoapBodyUse クラス・キーワード、SoapBodyUse メソッド・キーワード、および SoapBodyUse クエリ・キーワードは、<binding> 要素に影響します。これについては、"クラス定義リファレンス" に説明があります。これらのキーワードでは、literal および encoded の値を指定できます。

注釈　Web サービスにコンパイル済みのポリシー構成クラスがある場合、<binding> セクションには、<wsp:Policy> 形式の要素も含まれます。このドキュメントでは、ポリシーがどのように WSDL に影響するかについては触れません。それは WS-SecurityPolicy などの仕様によって決まるからです。

ポリシー構成については、"Web サービスの保護" を参照してください。

# B.6 <portType>

WSDL では、<binding> セクションの前に、<portType> 要素が含まれます。これらの要素はそれぞれ、<binding> 要素に対して単一のアドレスを指定することで個々のエンドポイントを定義します。<portType> 要素は、抽象操作の名前付きセットおよび関連する抽象メッセージです。

一般に、WSDL には複数の <portType> 要素を含めることができますが、InterSystems IRIS Web サービスの WSDL に含まれるのは 1 つだけです。

このトピックの最初に示したサンプルの Web サービスの場合、<portType> 要素は次のようになります。

XML

```
<portType name="MyServiceNameSoap">
    <operation name="Add">
        <input message="s0:AddSoapIn"/>
        <output message="s0:AddSoapOut"/>
    </operation>
</portType>
```

この要素のすべての特性は、WSDL の他の部分でも自動的に一貫性が保たれます。独立した制御は必要ありません。

# B.7 〈message〉

<portType> 要素の前の <message> 要素は、処理で使用されるメッセージを定義します。WSDL は通常、Web メソッドごとに 2 つの <message> 要素を含んでいます。このトピックの最初に示したサンプルの Web サービスの場合、これらの要素は次のようになります。

```
<message name="AddSoapIn">
    <part name="parameters" element="s0:Add"/>
</message>
<message name="AddSoapOut">
    <part name="parameters" element="s0:AddResponse"/>
</message>
```

この要素は、次のように指定されます。

- <message> 要素の name 属性は Web メソッド名に基づきます。
- メソッドのバインディング・スタイルは、前述のとおり、<binding> 要素によって決まります。これにより、メッセージが複数のパートを持てるかどうかが決まります。
  - バインディング・スタイルが ″document″ の場合、既定では各メッセージはパートを 1 つのみ持ちます。以下はその例です。

    ```
    <message name="AddSoapIn">
        <part name="parameters" .../>
    </message>
    ```

    ARGUMENTSTYLE パラメータが ″message″ の場合、メッセージは複数のパートを持つことができます。以下はその例です。

    ```
    <message name="AddSoapIn">
       <part name="a" .../>
       <part name="b" .../>
    </message>
    ```

  - バインディング・スタイルが ″rpc″ の場合、メッセージは複数のパートを持つことができます。以下はその例です。

    ```
    <message name="AddSoapIn">
        <part name="a" .../>
        <part name="b" .../>
    </message>
    ```

- <soap:body> 要素の use 属性は、前述の <binding> 要素で指定されたとおり、<part> 要素のメッセージの内容を決定します。
  - use 属性が "literal" の場合、<part> 要素には、element 属性が含まれます。以下はその例です。

    ```
    <part name="parameters" element="s0:Add"/>
    ```

    別の例を示します。

    ```
    <part name="b" element="s0:b"/>
    ```

  - use 属性が "encoded" の場合、<part> 要素には、element 属性ではなく type 属性が含まれます。以下はその例です。

    ```
    <part name="a" type="s0:ComplexNumber"/>
    ```

- メッセージが参照する要素またはタイプの名前は、次のセクションで説明しているように決定されます。

- それらの要素およびタイプが属するネームスペースは、次のセクションで説明しているように決定されます。
- "単方向 Web メソッドの WSDL の相違点" も参照してください。
- Web サービスで SOAPSESSION パラメータに 1 を指定すると、この要素に影響します。"InterSystems IRIS SOAP セッションの WSDL の相違点" を参照してください。

# B.8 〈types〉

WSDL では、<message> 要素の前に、<types> 要素が含まれます。これは、メッセージが使用するスキーマを定義します。<types> 要素には、Web サービスおよびそのクライアントで使用する要素、タイプ、またはその両方を定義する <schema> 要素が 1 つ以上記述されています。このトピックの最初に示したサンプルの Web サービスの場合、この要素は次のようになります。

XML

```
<types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://www.mynamespace.org">
        <s:element name="Add">
            <s:complexType>
                <s:sequence>
                    <s:element minOccurs="0" name="a" type="s0:ComplexNumber"/>
                    <s:element minOccurs="0" name="b" type="s0:ComplexNumber"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:complexType name="ComplexNumber">
            <s:sequence>
                <s:element minOccurs="0" name="Real" type="s:double"/>
                <s:element minOccurs="0" name="Imaginary" type="s:double"/>
            </s:sequence>
        </s:complexType>
        <s:element name="AddResponse">
            <s:complexType>
                <s:sequence>
                    <s:element name="AddResult" type="s0:ComplexNumber"/>
                </s:sequence>
            </s:complexType>
        </s:element>
    </s:schema>
</types>
```

以降のサブセクションでは、主なバリエーションについて説明します。

- <types> の name 属性
- <types> でのネームスペースの使用
- <types> セクションで使用可能なその他のバリエーション

注釈　<types> セクションも、Web サービスで使用するあらゆる XML 対応クラスに定義した XML プロジェクションの影響を受けます。XML プロジェクションによって、ネームスペースの使用、NULL 処理、特殊文字の処理などの課題が決まります。"オブジェクトの XML への投影" を参照してください。

SoapBindingStyle キーワードおよび SoapBodyUse キーワードは、WSDL の他のパートに影響し、それらのパートは <types> セクションの構造を決定します。

## B.8.1 name 属性

各 <schema> 要素は、メッセージ・スタイルに応じて、要素、タイプ、またはその両方で構成できます。要素またはタイプはそれぞれ、次のように指定する name 属性を持ちます。

- Web メソッドに対応している項目の場合、その項目の name 属性はその Web メソッドの名前 (Add など) と一致します。この属性は変更できません。
- 引数または返り値として使用する XML 対応クラスに対応している項目の場合、その項目の name 属性はそのクラスの XML プロジェクションで決まります。詳細は、"オブジェクトの XML への投影" を参照してください。
- 応答メッセージに対応している項目の場合、その項目の name 属性の形式は、既定では method_nameResponse (AddResponse など) になります。

  ドキュメント・スタイルのバインディングを使用する Web メソッドの場合、その Web メソッドの SoapMessageName キーワードを指定することで、これをオーバーライドできます。
- <schema> の下位レベル項目については、name 属性は自動的に設定され、独立して制御することはできません。

例えば、サンプルの Web メソッドを次のように編集したとします。

Class Member

```
Method Add(a As ComplexNumber, b As ComplexNumber)
As ComplexNumber [ WebMethod, SoapMessageName = MyResponseMessage]
{
    Set sum = ##class(ComplexNumber).%New()
    Set sum.Real = a.Real + b.Real
    Set sum.Imaginary = a.Imaginary + b.Imaginary

    Quit sum
}
```

この場合、<types> セクションは次のようになります。

XML

```
<types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://www.mynamespace.org">
        <s:element name="Add">
            <s:complexType>
                <s:sequence>
                    <s:element minOccurs="0" name="a" type="s0:ComplexNumber"/>
                    <s:element minOccurs="0" name="b" type="s0:ComplexNumber"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:complexType name="ComplexNumber">
            <s:sequence>
                <s:element minOccurs="0" name="Real" type="s:double"/>
                <s:element minOccurs="0" name="Imaginary" type="s:double"/>
            </s:sequence>
        </s:complexType>
        <s:element name="MyResponseMessage">
            <s:complexType>
                <s:sequence>
                    <s:element name="AddResult" type="s0:ComplexNumber"/>
                </s:sequence>
            </s:complexType>
        </s:element>
    </s:schema>
</types>
```

詳細は、"SOAP 応答のメッセージ名の制御" を参照してください。"オブジェクトの XML への投影" も参照してください。

## B.8.2 〈types〉 のネームスペース

Web サービスの次のパラメータは、<types> セクションでのネームスペースの使用に影響します。

- TYPENAMESPACE を指定している場合、このパラメータによって <schema> 要素の targetNamespace 属性が制御されます。
- TYPENAMESPACE を指定していない場合、targetNamespace 属性は NAMESPACE パラメータで指定されます。

- RESPONSETYPENAMESPACE は、使用する応答のタイプの targetNamespace 属性を制御します。
- USECLASSNAMESPACES は、サポートするタイプ・クラスで指定されたネームスペースも <types> で使用するかどうかを制御します。

各 XML 対応クラスの NAMESPACE パラメータも、WSDL の <types> 要素に影響します。

前出の Web サービスの次のバリエーションを取り上げます。

### Class Definition

```
Class WSDLSamples.Namespaces Extends %SOAP.WebService
{

Parameter SERVICENAME = "MyServiceName";

Parameter NAMESPACE = "http://www.mynamespace.org";

Parameter RESPONSENAMESPACE = "http://www.myresponsenamespace.org";

Parameter TYPENAMESPACE = "http://www.mytypes.org";

Parameter RESPONSETYPENAMESPACE = "http://www.myresponsetypes.org";

Parameter USECLASSNAMESPACES = 1;

///  adds two complex numbers
Method Add(a As ComplexNumberNS, b As ComplexNumberNS) As ComplexNumberNS [ WebMethod ]
{
    Set sum = ##class(ComplexNumberNS).%New()
    Set sum.Real = a.Real + b.Real
    Set sum.Imaginary = a.Imaginary + b.Imaginary

    Quit sum
}

}
```

クラス WSDLSamples.ComplexNumberNS は次のようになります。

### Class Definition

```
///  A complex number
Class WSDLSamples.ComplexNumberNS Extends (%RegisteredObject, %XML.Adaptor)
{

Parameter NAMESPACE = "http://www.complexnumbers.org";

Property Real As %Double;

Property Imaginary As %Double;

}
```

この Web サービスの WSDL では、<types> パートは次のようになります。

### XML

```
<types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://www.mytypes.org">
        <s:import namespace="http://www.complexnumbers.org"/>
        <s:element name="Add">
            <s:complexType>
                <s:sequence>
                    <s:element minOccurs="0" name="a" type="ns2:ComplexNumberNS"/>
                    <s:element minOccurs="0" name="b" type="ns2:ComplexNumberNS"/>
                </s:sequence>
            </s:complexType>
        </s:element>
    </s:schema>
    <s:schema elementFormDefault="qualified" targetNamespace="http://www.complexnumbers.org">
        <s:complexType name="ComplexNumberNS">
            <s:sequence>
                <s:element minOccurs="0" name="Real" type="s:double"/>
                <s:element minOccurs="0" name="Imaginary" type="s:double"/>
```

```
            </s:sequence>
        </s:complexType>
    </s:schema>
    <s:schema elementFormDefault="qualified" targetNamespace="http://www.myresponsetypes.org">
        <s:import namespace="http://www.complexnumbers.org"/>
        <s:element name="AddResponse">
            <s:complexType>
                <s:sequence>
                    <s:element name="AddResult" type="ns2:ComplexNumberNS"/>
                </s:sequence>
            </s:complexType>
        </s:element>
    </s:schema>
</types>
```

## B.8.3 使用可能なその他のバリエーション

以下に示すその他のパラメータも <types> 要素に影響します。

- Web サービスの INCLUDEDOCUMENTATION パラメータが 1 の場合、タイプ・クラスに追加したコメントを含む <annotation> 要素が <types> セクションに追加されます (このコメントの先頭には 3 つのスラッシュを記述する必要があります)。

  既定では、INCLUDEDOCUMENTATION は 0 です。

  例えば、サンプルの Web サービスを編集して次の記述を追加したとします。

  Class Member

  ```
  Parameter INCLUDEDOCUMENTATION = 1;
  ```

  この場合、<types> セクションは次のようになります。

  ```
  ...
  <s:complexType name="ComplexNumber">
      <s:annotation>
          <s:documentation>A complex number</s:documentation>
      </s:annotation>
      <s:sequence>
          <s:element minOccurs="0" name="Real" type="s:double">
              <s:annotation>
                  <s:documentation>real part of the complex number</s:documentation>
              </s:annotation>
          </s:element>
          <s:element minOccurs="0" name="Imaginary" type="s:double">
              <s:annotation>
                  <s:documentation>imaginary part of the complex number</s:documentation>
              </s:annotation>
          </s:element>
      </s:sequence>
  </s:complexType>
  ...
  ```

- Web サービスで SOAPBINARY パラメータに 1 を指定すると、<types> 要素に影響します。"InterSystems IRIS バイナリ SOAP 形式の WSDL の相違点" を参照してください。
- Web サービスで SOAPSESSION パラメータに 1 を指定すると、この要素に影響します。"InterSystems IRIS SOAP セッションの WSDL の相違点" を参照してください。
- SoapTypeNameSpace キーワードを指定すると、WSDL のこのパートに影響します。"クラス定義リファレンス" を参照してください。
- 引数で REQUIRED パラメータに 1 を指定すると、WSDL に引数の minOccurs=1 が含まれます。このクラスのパラメータについては、"基本要件" を参照してください。
- メソッドに SoapRequestMessage キーワードが指定されている場合、対応する要素の名前は、そのメソッドの名前ではなく、SoapRequestMessage キーワードの値です。

- ALLOWREDUNDANTARRAYNAME パラメータの影響については、"オブジェクトの XML への投影" の "XML スキーマへの投影の制御" で "コレクション・プロパティの XML スキーマへの投影" を参照してください。

# B.9 メソッド・シグニチャのバリエーションによる WSDL のバリエーション

ここでは、メソッド・シグニチャのバリエーションによって生じる WSDL のバリエーションをいくつか示します。

## B.9.1 参照による返り値または出力パラメータとしての返り値

参照によって値を返す場合、または出力パラメータとして値を返す場合には、Web メソッドのシグニチャ内で ByRef キーワードまたは Output キーワードを必要に応じて使用します。この変更は、スキーマと SOAP 応答メッセージに影響を及ぼします。

例えば、2 つの異なる Web サービスのメソッドから、次のような Web メソッド・シグニチャがあるとします。

```
//from web service 1
Method HelloWorld() As %String [ WebMethod ]

//from web service 2
Method HelloWorld(ByRef myarg As %String) [ WebMethod ]
```

最初の Web サービスでは、<types> セクションは次のようになります。

XML

```
<types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://www.helloworld.org">
        <s:element name="HelloWorld1">
            <s:complexType>
                <s:sequence/>
            </s:complexType>
        </s:element>
        <s:element name="HelloWorld1Response">
            <s:complexType>
                <s:sequence>
                    <s:element name="HelloWorld1Result" type="s:string"/>
                </s:sequence>
            </s:complexType>
        </s:element>
    </s:schema>
</types>
```

参照によって値を返す 2 つ目の Web サービスでは、<types> セクションに、応答メッセージに対応するタイプのバリエーションがあります。

```
<types>
...
        <s:element name="HelloWorld2Response">
            <s:complexType>
                <s:sequence>
                    <s:element minOccurs="0" name="myarg" type="s:string"/>
...
```

これは、<HelloWorld2Response> メッセージに含まれる要素が <myarg> であることを示しています。この要素は、メッセージ・シグニチャの引数の名前に対応します。一方、この要素は通常 <methodnameResult> です。

Output キーワードの代わりに ByRef キーワードを使用しても、WSDL に同じ影響が及ぼされます。

これらのキーワードの詳細は、"クラスの定義と使用" の "メソッド" を参照してください。

# B.10 InterSystems IRIS Web サービスの WSDL のその他のバリエーション

このセクションでは、InterSystems IRIS Web サービスに使用できる、WSDL のその他のバリエーションについて説明します。

## B.10.1 InterSystems IRIS SOAP セッションの WSDL の相違点

Web サービスで SOAPSESSION パラメータに 1 を指定すると、WSDL に次のように影響します。

- <binding> 要素では、各 <operation> 要素の <input> 要素と <output> 要素に次の追加の下位要素があります。

  XML

  ```
  <soap:header message="s0:IRISSessionHeader" part="CSPCHD" use="literal"/>
  ```

  以下はその例です。

  ```
  <operation name="Add">
     <soap:operation soapAction="http://www.mynamespace.org/WSDLSamples.BasicWS.Add" style="document"/>

      <input>
          <soap:body use="literal"/>
          <soap:header message="s0:IRISSessionHeader" part="CSPCHD" use="literal"/>
      </input>
      <output>
          <soap:body use="literal"/>
          <soap:header message="s0:IRISSessionHeader" part="CSPCHD" use="literal"/>
      </output>
  </operation>
  ```

- WSDL に次の追加の <message> 要素があります。

  XML

  ```
  <message name="IRISSessionHeader">
      <part name="CSPCHD" element="chead:CSPCHD"/>
  </message>
  ```

- <types> 要素に次の追加項目があります。

  XML

  ```
  <s:schema elementFormDefault="qualified" targetNamespace="http://www.intersystems.com/SOAPheaders">

      <s:element name="CSPCHD">
          <s:complexType>
              <s:sequence>
                  <s:element name="id" type="s:string"/>
              </s:sequence>
          </s:complexType>
      </s:element>
  </s:schema>
  ```

- ネームスペース宣言に次の追加項目があります。

  ```
  xmlns:chead="http://www.intersystems.com/SOAPheaders"
  ```

## B.10.2 InterSystems IRIS バイナリ SOAP 形式の WSDL の相違点

SOAPBINARY パラメータに 1 を指定した InterSystems IRIS Web サービスでは、WSDL の機能が次のように拡張されます。

- `<binding>` 要素に、SOAP バイナリ・サポートを示す拡張の子要素があります。

  ```
  <isc:binding charset="isc_charset">
  ```

  isc_charset は、Web サービスの InterSystems IRIS ネームスペースの InterSystems IRIS 文字セット (Unicode、Latin1 など) です。

  以下はその例です。

  ```
  <isc:binding charset="Unicode">
  ```

- WSDL の `<types>` セクションでは、各 `<complexType>` 要素に次の拡張属性があります。

  ```
  <complexType isc:classname="service_name:class_name" ...>
  ```

  service_name は Web サービス名、class_name はこの複雑なタイプに対応する InterSystems IRIS クラス名です。以下はその例です。

  ```
  <s:complexType isc:classname="AddComplex:GSOAP.ComplexNumber" name="ComplexNumber">
    <s:sequence>
      <s:element minOccurs="0" name="Real" type="s:double"/>
      <s:element minOccurs="0" name="Imaginary" type="s:double"/>
    </s:sequence>
  </s:complexType>
  ```

- また、`<types>` セクションでは、`<element>` 要素と `<simpleContent>` 要素にも、必要に応じて次の形式の拡張属性があります。

  ```
  <element isc:property="property_name" ....>
  <simpleContent isc:property="property_name" ....>
  ```

  property_name は、その要素にマップする InterSystems IRIS プロパティの名前です(WSDL では、XMLPROJECTION を `"content"` に設定したプロパティには、`<simpleContent>` 要素が使用されます)。

  以下はその例です。

  ```
  <s:element minOccurs="0" name="RealType" isc:property="Real" type="s:double"/>
  ```

  この属性は、プロパティ名と異なる XML 名を使用できるようにする XMLNAME プロパティ・パラメータをプロパティで使用している場合にのみ設定されます。上の例は、次のプロパティを持つクラスを使用する Web サービスの WSDL から抽出したものです。

  ```
  Property Real As %Double (XMLNAME = "RealType");
  ```

  `isc:property` 属性を使用すると、生成されたクライアント・クラスのプロパティ名を、サービス・クラスのプロパティ名と同じに指定できます。現在、ユーザが選択した内容または `substitutionGroup` を使用する場合、また、タイプ・クラスに XMLNAME パラメータがあるラップ要素の場合は例外です。

- ネームスペース宣言に次の追加項目があります。

  ```
  xmlns:isc="http://www.intersystems.com/soap/"
  ```

これらの WSDL の拡張機能は、XML スキーマ、WSDL、および WS-I Basic Profile の各仕様の下で有効であり、これらの仕様に準拠したすべての Web クライアント・ツールキットでは無視されることが想定されています。

注釈　InterSystems IRIS Web サービスまたは Web クライアントが InterSystems IRIS バイナリ SOAP 形式を使用する場合、その Web サービスまたは Web クライアントで WS-Security 機能または WS-Policy 機能を使用することはできません。"Web サービスの保護" を参照してください。

## B.10.3 単方向 Web メソッドの WSDL の相違点

メソッドの返りタイプを %SOAP.OneWay として定義している場合、WSDL は既定のものと以下の点が異なります。

- <binding> 要素では、メソッドの <operation> 要素に <output> 要素がありません。
- <portType> 要素では、メソッドの <operation> 要素に <output> 要素がありません。
- 応答メッセージには <message> 要素がありません。

# C

# 生成されたクラスの詳細

このトピックでは、参考情報として、SOAP ウィザードによって生成されるクラスについて説明します。

## C.1 生成されるクラスの概要

SOAP ウィザードでは、以下のようなクラスが生成されます。

- SOAP ウィザードでのユーザの選択に応じて、Web クライアント・クラス、Web サービス・クラス、またはその両方が生成されます。Web クライアント・クラスが作成された場合、それは %SOAP.WebClient を拡張したものです。Web サービス・クラスが作成された場合、それは %SOAP.WebService を拡張したものです。

  これらのクラスにはそれぞれ、WSDL で定義された Web メソッドごとに 1 つの Wb メソッドがあります。Web クライアントの場合、このメソッドは以下の例のようになります。

  Class Member

  ```
  Method DemoMethod() As %String [ Final, SoapBindingStyle = document,
  SoapBodyUse = literal, WebMethod ]
  {
   Quit ..WebMethod("DemoMethod").Invoke($this,"http://tempuri.org/Demo.MyService.DemoMethod")
  }
  ```

  Web サービスの場合、このメソッドは以下の例のようになります。

  Class Member

  ```
  Method DemoMethod() As %String [ Final,
  SoapAction = "http://tempuri.org/Demo.MyService.DemoMethod",
  SoapBindingStyle = document, SoapBodyUse = literal, WebMethod ]
  {
   // Web Service Method Implementation Goes Here.
  }
  ```

- Web メソッドへの入力または出力として使用される複雑なタイプごとに、SOAP ウィザードによって XML 対応クラスが生成されます。
- 上記のタイプのコンポーネントとなる複雑なタイプごとに、SOAP ウィザードによって XML 対応クラスが生成されます。

  SOAP ウィザードではこれが再帰的に行われるため、最も複雑でないタイプのプロパティは単純なデータ型プロパティとなり、これは XSD タイプと直接対応します。

SOAP ウィザードでは、これらのクラスに、エンコードおよびバインディング・スタイル、ネームスペースの割り当て、およびその他の項目を指定するのに必要な、クラスおよびメソッドのキーワードとパラメータを指定します。

# C.2 エンコードおよびバンディング・スタイルを制御するキーワード

SOAP ウィザードでは、生成された Web クライアント・クラスおよび Web サービス・クラスに、指定の WSDL との連携に必要なエンコードおよびメッセージ・スタイルを制御する以下のキーワードを指定します。

- SoapBodyUse クラス・キーワード
- SoapBodyUse メソッド・キーワード
- SoapBindingStyle クラス・キーワード
- SoapBindingStyle メソッド・キーワード

Web クライアントまたは Web サービスが WSDL に従わなくなるため、これらのキーワードは変更しないでください。詳細は、"クラス定義リファレンス" を参照してください。

# C.3 ネームスペースの割り当てを制御するパラメータとキーワード

SOAP ウィザードは、生成されたクラスに、ネームスペース割り当てを制御するためのパラメータとキーワードを使用します。以下のサブセクションでは、メッセージのネームスペースとタイプのネームスペースについて説明します。

Web クライアントまたは Web サービスが WSDL に従わなくなるため、これらの値は変更しないでください。SoapNameSpace および SoapTypeNameSpace の詳細は、"クラス定義リファレンス" を参照してください。

## C.3.1 メッセージのネームスペース

SOAP ウィザードでは、次の値を指定して、SOAP メッセージで使用されるネームスペースを制御します。

テーブル III-1: Web クライアントまたは Web サービスによって送信される SOAP メッセージのネームスペース

| アイテム | SOAP ウィザードで指定される値 |
|---|---|
| NAMESPACE (クラス・パラメータ) | すべての要求メッセージで同一のネームスペースが使用される場合の要求メッセージのネームスペース。 |
| SoapNameSpace (メソッド・キーワード) | 各要求メッセージで異なるネームスペースが使用される場合の各要求メッセージのネームスペース。 |
| RESPONSENAMESPACE (クラス・パラメータ) | 応答メッセージのネームスペース。これを指定しないと、応答メッセージは NAMESPACE パラメータで指定されるネームスペースに配置されます。SoapNameSpace キーワードは、応答メッセージのネームスペースに影響を及ぼすことはありません。 |

## C.3.2 タイプのネームスペース

SOAP ウィザードでは、次に示すように、メッセージ・タイプがネームスペースに自動的に割り当てられます。

テーブル III-2: Web クライアントおよび Web サービスで使用されるタイプのネームスペース

| アイテム | SOAP ウィザードで指定される値 |
|---|---|
| TYPENAMESPACE（クラス・パラメータ） | すべてのメソッドが同一のネームスペースのタイプを参照する場合に、SOAP ウィザードによってこのパラメータが設定されます。 |
| RESPONSETYPENAMESPACE（クラス・パラメータ） | SOAP ウィザードでは、WSDL がドキュメントスタイルのバインディングを使用し、応答メッセージが要求メッセージとは異なるネームスペースのタイプを使用する場合に、このパラメータが設定されます。このパラメータは、クラスのすべてのメソッドに適用されます。応答タイプはすべて、相互に同一のネームスペースに属すると見なされます。 |
| SoapTypeNameSpace（メソッド・キーワード） | <s:schema> 要素の targetNamespace 属性の値。各メソッドが異なるネームスペースのタイプを使用している場合に、SOAP ウィザードによって、メソッドごとにこのキーワードが設定されます。<br>このキーワードは、RESPONSETYPENAMESPACE パラメータをオーバーライドしません。 |

# C.4 配列プロパティの作成

既定では、特定のシナリオで、SOAP ウィザードによって配列型プロパティが作成されます。[配列プロパティ以外を作成] オプションを使用すると、別の構造を持つプロパティを必要に応じて作成できます。

具体的には、以下のタイプを含む WSDL を考えてみます。

```
<s:complexType name="Obj">
    <s:sequence>
        <s:element minOccurs="0" name="MyProp" type="test:Array"/>
    </s:sequence>
</s:complexType>
<s:complexType name="Array">
    <s:sequence>
        <s:element maxOccurs="unbounded" minOccurs="0" name="Item" nillable="true" type="test:Item"/>
    </s:sequence>
</s:complexType>
<s:complexType name="Item">
    <s:simpleContent>
        <s:extension base="s:string">
<s:attribute name="Key" type="s:string" use="required"/>
        </s:extension>
    </s:simpleContent>
</s:complexType>
```

既定では、SOAP ウィザードによって以下のクラスが生成されます。

### Class Definition

```
Class testws.Obj Extends (%RegisteredObject, %XML.Adaptor)
{

Parameter ELEMENTQUALIFIED = 1;

Parameter NAMESPACE = "http://testws.org";

Parameter XMLNAME = "Obj";

Parameter XMLSEQUENCE = 1;

Property MyProp As array Of %String(MAXLEN = "", XMLITEMNAME = "Item",
XMLKEYNAME = "Key", XMLNAME = "MyProp", XMLPROJECTION = "COLLECTION");

}
```

SOAP ウィザードを使用する際に [**配列プロパティ以外を作成**] を選択すると、生成された `Obj` クラス内の `MyProp` プロパティが代わりに以下のように定義されます。

### Class Member

```
Property MyProp As list Of testws.Item(XMLITEMNAME = "Item", XMLNAME = "MyProp", XMLPROJECTION =
"COLLECTION");
```

このプロパティは、SOAP ウィザードによって同様に生成された以下のクラスを参照します。

### Class Definition

```
Class testws.Item Extends (%SerialObject, %XML.Adaptor)
{

Parameter ELEMENTQUALIFIED = 1;

Parameter NAMESPACE = "http://testws.org";

Parameter XMLNAME = "Item";

Parameter XMLSEQUENCE = 1;

Property content As %String(MAXLEN = "", XMLNAME = "content", XMLPROJECTION = "CONTENT");

Property Key As %String(MAXLEN = "", XMLNAME = "Key", XMLPROJECTION = "ATTRIBUTE")
[ Required, SqlFieldName = _Key ];

}
```

# C.5 生成されたクラスの Web メソッドに関するその他の注意事項

このセクションでは、生成される Web クライアント・クラスの Web メソッドに関するその他の注意事項を示します。

- InterSystems IRIS は、各メソッド名がメソッド名の制限より短く、かつ一意であることを確認します (メソッド名の長さについては、"サーバ側プログラミングの入門ガイド" の "一般的なシステム制限" を参照してください)。WSDL でのメソッド名がこの制限より長い場合、生成される Web メソッドの名前は、WSDL での名前と同じになりません。

  使用されるアルゴリズムは文書化されておらず、予告なしに変更される場合があります。

- 有効な WSDL の指定のメソッドで `soapAction` 属性が `""` である場合、生成されるクライアント・クラスに次のいずれかの変更を加えて、このメソッドが機能するようにする必要があります。

  - SOAPACTIONQUOTED パラメータを 1 に設定します。
  - 生成されるクライアント・クラスで Web メソッドを編集します。このメソッドは、元は次のような内容です。

    ```
    Quit ..WebMethod("HelloWorld").Invoke($this,"")
    ```

    これを次のように編集します。

    ```
    Quit ..WebMethod("HelloWorld").Invoke($this,"""")
    ```

  または、Web クライアントを生成する前に WSDL を編集します。この場合、`soapAction` 属性を、`""` ではなく、`""""` となるように編集します。