
# InterSystems IRIS での Web サービスの調整

Version 2024.1
2024-06-03

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

InterSystems IRIS での Web サービスの調整
InterSystems IRIS Data Platform　Version 2024.1　2024-06-03


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

# 目次

# 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® データ・プラットフォーム 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。

InterSystems IRIS® データ・プラットフォーム Web サービスの基本情報の詳細は、"Web サービスのパラメータの指定" を参照してください。

# 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® データ・プラットフォーム 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。

# 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® データ・プラットフォーム 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。

# 4

# スキーマとタイプのネームスペースの制御

ここでは、InterSystems IRIS® データ・プラットフォーム Web サービスの 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。

## 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'>
...
```

## 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 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。

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。

# 5
# タイプのドキュメントの追加

既定では、InterSystems IRIS® データ・プラットフォーム 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。

# 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。

# 7

# 必要な要素および属性のチェック

既定では、InterSystems IRIS® データ・プラットフォーム 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。

# 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® データ・プラットフォーム 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。

# 9

# SOAP 応答のメッセージ名の制御

InterSystems IRIS® データ・プラットフォーム Web サービスでは、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。"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。

# 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® データ・プラットフォーム Web サービスの 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。

# 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、以下のようになります。

- 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。

# 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パラメータ (XMLELEMENT) があります。具体的には、以下のようになります。

- 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

# xsi:type 属性の使用の制御

既定では、InterSystems IRIS® データ・プラットフォーム 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。以下に例を示します。

```
<?xml version="1.0" encoding="UTF-8" ?>
...
<types:GetPersonResponse>
<GetPersonResult href="#id1" />
</types:GetPersonResponse>
<types:Person id="id1" xsi:type="types:Person">
<Name>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。

# 14

# エンコード形式でのインライン参照の使用の制御

InterSystems IRIS® データ・プラットフォーム Web サービスにおいて、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。

# 15

# SOAP エンベロープ接頭語の指定

既定では、InterSystems IRIS® データ・プラットフォーム 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。

# 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® データ・プラットフォーム 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。このパラメータは、単に WSDL 内で通知されるバージョンを制御します。

# 17

# gzip で圧縮された応答の送信

InterSystems IRIS® データ・プラットフォーム 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。

# 18

# 単方向 Web メソッドの定義

InterSystems IRIS® データ・プラットフォーム Webサービスの場合、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、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。

"単方向 Web メソッドの WSDL の相違点" を参照してください。

## 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>
```

# 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。

# 19

# バイナリ・データへの改行の追加

InterSystems IRIS® データ・プラットフォーム 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。

# 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® データ・プラットフォーム 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"`

# 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® データ・プラットフォーム 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。それには、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
}
```

# 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® データ・プラットフォーム 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。

# 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® データ・プラットフォーム 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。

# 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® データ・プラットフォーム 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。

## 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。

## 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、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。

# 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® データ・プラットフォーム 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。ここでは、その詳細を説明します。

## 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。

## 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
    }
   ```

# 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。