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

Version 2024.1
2024-06-06

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

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
InterSystems IRIS Data Platform　Version 2024.1　2024-06-06

インターシステムズ社の製品に関するサポートやご質問は、以下にお問い合わせください：

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

目次

1

SOAP と Web サービスについて

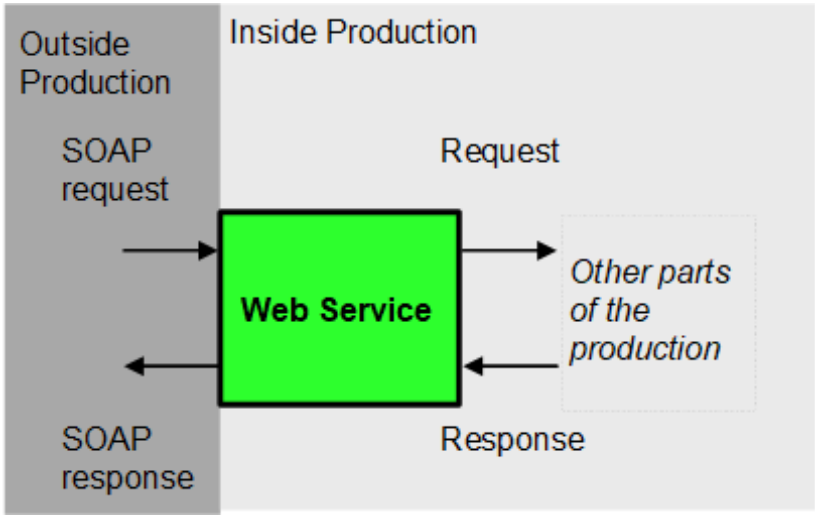
InterSystems IRIS® データ・プラットフォームは、SOAP 1.1 および 1.2 (Simple Object Access Protocol) をサポートします。このサポートを利用して、以下の操作を行えます。

- プロダクションに Web サービスを追加し、プロダクションに SOAP 対応のフロント・エンドを提供します。クライアント・アプリケーションは、SOAP プロトコルを使用して、このプロダクション Web サービスを呼び出すことができます。他の SOAP 対応のアプリケーションから、このようなメソッドを検索して呼び出すことができます。InterSystems IRIS は、データベース内で直接 SOAP メソッドを実行するため、非常に効率的に実行できます。
- プロダクションに Web クライアントを追加します。使用したい Web サービスの WSDL ドキュメントがある場合は、ツールを使用してビジネス・オペレーションやプロキシ・クライアント・クラスを生成できます。InterSystems IRIS Web クライアントは、InterSystems IRIS SOAP 送信アダプタおよび生成されたプロキシ・クライアント・クラスを介して、外部の Web サービスを呼び出します。

SOAP および Web サービスでインターシステムズ製品がサポートする特定の基準 (WSDL 制限など) については、"Web サービスおよび Web クライアントの作成" を参照してください。

1.1 InterSystems IRIS による Web サービスのサポート

プロダクション用に SOAP 対応のフロント・エンドを提供できます。そのためには、Web サービスでありビジネス・サービスでもある、プロダクション Web サービスを作成します。内部的には、通常 Web メソッドが SOAP 要求メッセージを受信し、それらを使用してプロダクション内で必要とされる要求メッセージを作成および送信します。次に応答メッセージを受信し、それらを使用して SOAP 応答メッセージを作成します。



プロダクション Web サービスを作成するために、InterSystems IRIS には、**%SOAP** および **%XML** パッケージに基本プロダクション Web サービス・クラス (**EnsLib.SOAP.Service**)、およびサポート・クラスが用意されています。

InterSystems IRIS は、強力な Web サービスの組み込みサポートを提供します。基本プロダクション Web サービス・クラスは、以下のことを実行できます。

- 受信 SOAP メッセージを検証します。
- SOAP メッセージを解凍し、InterSystems IRIS 表記にデータを変換し、要求メッセージをプロダクション内の宛先に送信する対応メソッドを呼び出します。
- メソッドを実行します。
- 応答メッセージを受信し、応答メッセージ (SOAP メッセージ) を作成して呼び出し側に返信します。

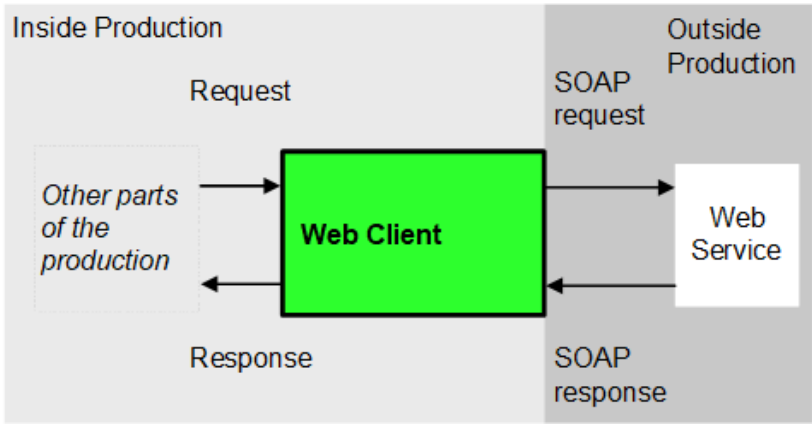
SOAP 仕様には、セッション・サポートは含まれていません。ただし、Web クライアントと使用する Web サービスとの間でセッションを維持することが有用な場合がよくあります。プロダクション Web サービスを使用して、これを行うことができます。Web サービスがセッションを使用する場合、セッション ID が確立され、クライアントからの呼び出しが一度正しく認証されると、その後はサービスに対する繰り返しの呼び出しが許可されます。

また、プロダクション Web サービス・クラスは、任意のビジネス・サービスの完全な機能を提供します。

注釈　　プロダクション Web サービスを作成する際にプロダクション・アダプタは使用しません。

1.2 InterSystems IRIS による Web クライアントのサポート

プロダクション内から外部の Web サービスを呼び出すことができます。そのために、InterSystems IRIS Web クライアントを作成します。



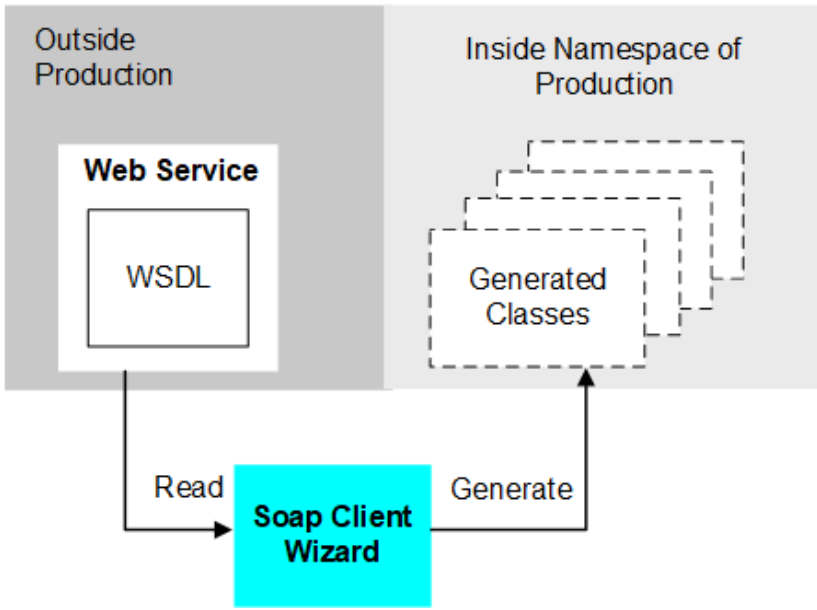
高レベルで、InterSystems IRIS Web クライアントは InterSystems IRIS 要求を受信し、SOAP 要求に変換して適切な Web サービスに送信します。同様に、SOAP 応答を受信して InterSystems IRIS 応答に変換します。

InterSystems IRIS Web クライアントは、以下の要素で構成され、その要素のすべてが SOAP ウィザードを使用してスタジオ内で生成できます。

- プロキシ・クライアント・クラス。Web サービスによって定義される各メソッドに対してプロキシ・メソッドを定義します。プロキシ・クライアントの目的は、Web サービスの場所を指定し、プロキシ・メソッドを保持することです。各プロキシ・メソッドは、対応する Web サービス・メソッドが使用するものと同じシグニチャを使用し、要求に応じてそのメソッドを呼び出します。
- ビジネス・オペレーション。InterSystems IRIS SOAP 送信アダプタを使用して、プロキシ・メソッドを呼び出します。
- サポート・クラス。XML タイプおよびプロダクション・メッセージの定義に必要となります。

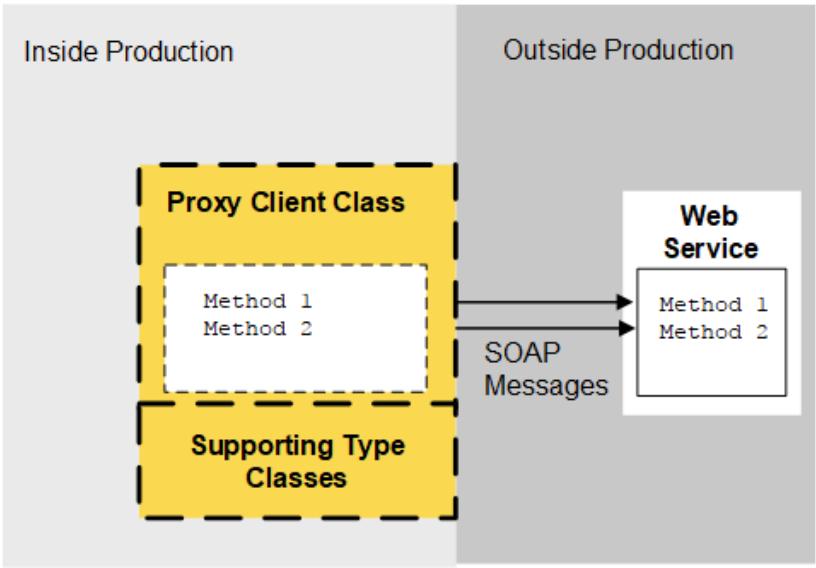
1.2.1 SOAP ウィザード

これらの各要素を理解するには、これらがどのように生成されるかを考慮することが有用です。まず、SOAP ウィザードを使用するときに、対象となる Web サービスの WSDL の URL を入力します。ウィザードが WSDL を読み取り、一連のクラスを生成します。



1.2.2 プロキシ・クライアント

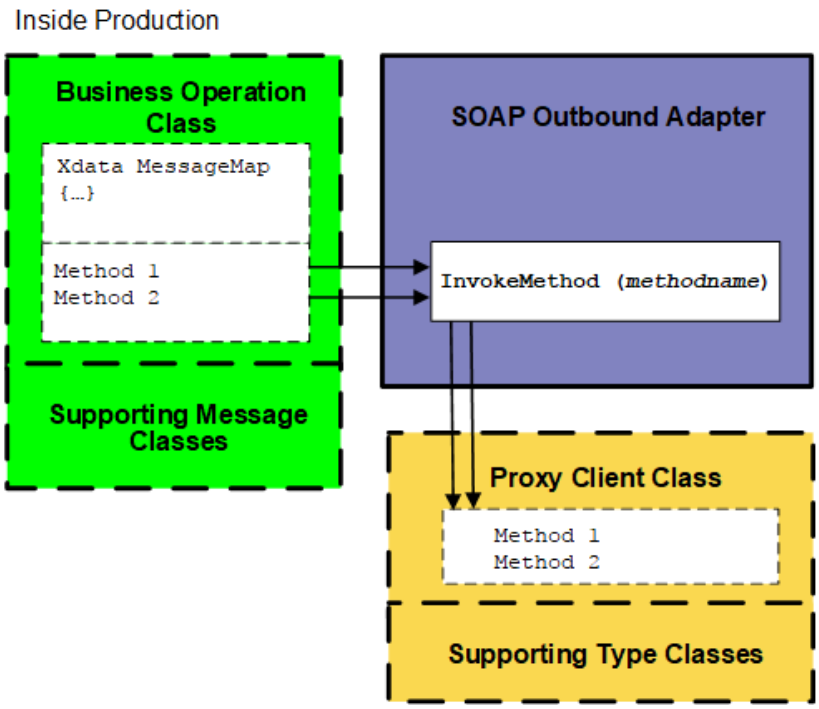
生成されたクラスには、Web サービスの各メソッドに対してプロキシ・メソッドを定義する、プロキシ・クライアント・クラスが含まれます。各プロキシ・メソッドは、Web サービスに SOAP 要求を送信し、対応する SOAP 応答を受信します。



図のように、生成されたクラスには、メソッドの入力や出力に必要な XML タイプを定義するクラスも含まれます。

1.2.3 InterSystems IRIS Web クライアントのビジネス・オペレーション

このウィザードでは、プロキシ・クライアントを呼び出すビジネス・オペレーション・クラス、および必要に応じてメッセージ・タイプを定義するクラスを生成することもできます。以下の図は、これらのクラスが機能するしくみについて示しています。



点線で示されたクラスおよびメソッドは、すべて SOAP ウィザードによって生成されます。

ビジネス・オペレーションは、SOAP 送信アダプタを使用します。SOAP 送信アダプタは、便利な実行時設定および汎用メソッド InvokeMethod() を提供します。プロキシ・クライアント・クラスのプロキシ・メソッドを呼び出すために、ビジネス・オペレーション・クラスは InvokeMethod() を呼び出し、実行するメソッド名および任意の引数を渡します。次に、InvokeMethod() がプロキシ・クライアント・クラスのメソッドを呼び出します。

2

プロダクション内での Web サービスの作成

ここでは、プロダクションの Web サービスである、プロダクション Web サービスを作成する方法を説明します。この操作を行うと、プロダクションに SOAP 対応のインタフェースが提供されます。

このドキュメントに記載されていない設定は、"プロダクションの管理" の "すべてのプロダクションに含まれる設定" を参照してください。

代替方法については、付録 "SOAP 受信アダプタの使用法" を参照してください。

Tipヒント　InterSystems IRIS® では、SOAP を使用する特殊なビジネス・サービス・クラスとユーザのニーズに適したビジネス・サービス・クラスの 1 つも提供されます。そのため、プログラミングの必要がありません。"相互運用プロダクションの概要" の "接続オプション" を参照してください。

2.1 全般的な動作

プロダクション Web サービスは、EnsLib.SOAP.Service またはサブクラスを基本としています。このクラスは、Ens.BusinessService と %SOAP.WebService の両方を拡張します。前者を拡張することで、このクラスはビジネス・サービスとなり、後者を拡張することで、このクラスは Web サービスとしても機能できるようになります。プロダクション Web サービスは、以下のように動作します。

- これは Web サービスであるため、その中で使用できる Web メソッドを記述している WSDL ドキュメント (自動的に生成されます) を保持しています。このサービスは、WSDL に準拠する任意の SOAP メッセージを受信し、これに応じて SOAP 応答を送信できます。
- これはビジネス・サービスなので、このサービスを追加するプロダクションの不可欠な要素となります。監視、エラー・ログ、実行時パラメータ、およびその他のすべてのプロダクション機能が通常どおりに使用できます。

  注釈　プロダクション Web サービスは、プロダクションが実行中でないと (また、ビジネス・サービスが有効化されていないと) 使用できません。

外部とのやり取りは、SOAP 要求メッセージおよび応答メッセージによって行われます。InterSystems IRIS Interoperability 要求メッセージおよび応答メッセージは、プロダクション内で使用されます。

2.2 基本要件

プロダクション内に Web サービスを作成するには、ここに記載されているように新しいビジネス・サービス・クラスを作成します。後で、それをプロダクションに追加して、構成します。

存在しなければ、適切なメッセージ・クラスを作成する必要もあります。"プロダクションの開発" の "メッセージの定義" を参照してください。

ビジネス・サービス・クラスの基本要件を以下に列挙します。

- このクラスは EnsLib.SOAP.Service を拡張します。このクラスは、Ens.BusinessService と %SOAP.WebService の両方を拡張します。前者を拡張することで、このクラスはビジネス・サービスとなり、後者を拡張することで、このクラスは Web サービスとしても機能できるようになります。
- このクラスは ADAPTER パラメータを NULL ("") として定義します。以下に例を示します。

  Class Member

  ```
  Parameter ADAPTER = "";
  ```

  または、以下も同じ意味です。

  Class Member

  ```
  Parameter ADAPTER;
  ```

- このクラスは、その他のパラメータを次の値に指定します。

パラメータ	説明
SERVICENAME	Web サービスの名前。文字から始まり、英数字のみを含む名前でなければなりません。デフォルトのサービス名は "MyProductionRequestWebService" です。
NAMESPACE	サービスおよびそのコンテンツが別のサービスと競合しないように、Web サービスのターゲットの XML ネームスペースを定義する URI。最初は http://tempuri.org に設定されています。これは、SOAP 開発者が開発中に使用した暫定的な URI です。
TYPENAMESPACE	Web サービスによって定義されるタイプにおけるスキーマの XML 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、NAMESPACE で指定されるネームスペースにあります。
RESPONSENAMESPACE	応答メッセージの XML ネームスペースを定義する URI。デフォルトでは、これは NAMESPACE パラメータで指定されるネームスペースと同じです。

- このクラスは、"Web メソッドの定義" で説明するように、Web メソッドを定義します。
- その他のオプションと一般情報は、"プロダクションの開発" の "ビジネス・サービス・クラスの定義" を参照してください。

以下は、一般的なクラスの例です。

Class Definition

```
Class Hospital.MyService Extends EnsLib.SOAP.Service
{

///For this business service, ADAPTER should be "" so that we use the normal SOAP processing
Parameter ADAPTER = "";
```

```
Parameter SERVICENAME = "MyService";

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

Parameter USECLASSNAMESPACES = 1;

Method GetAuthorization(patientID As %Integer, RequestedOperation As %String,
LikelyOutcome As %String) As %Status [ WebMethod ]
{
    set request = ##class(Hospital.OperateRequest).%New()
    set request.PatientID = patientID
    set request.RequestedOperation = RequestedOperation
    set request.LikelyOutcome = LikelyOutcome
    set tSC=..SendRequestSync("Hospital.PermissionToOperateProcess",request,.response)
    // Create the SOAP response, set its properties, and return it.
}

}
```

2.3 InterSystems IRIS で使用する Web メソッドの定義

この節では、InterSystems IRIS Web メソッドの基本的な要件を説明します。

- “基本要件 ” で説明されているとおり、**EnsLib.SOAP.Service** のサブクラスでメソッドを定義します。
- メソッドに WebMethod キーワードでマーキングします。
- すべての引数および戻り値が XML に対応していることを確認します。
  - メソッドが引数または戻り値としてオブジェクトを使用している場合は、そのオブジェクトが XML に対応していることを確認します。つまり、タイプに対するクラス定義が **%XML.Adaptor** を拡張している必要があります。このクラスのデフォルト設定で通常は適切ですが、そうでない場合は、″オブジェクトの XML への投影″ を参照してください。
  - メソッドが引数または戻り値としてデータ・セットを使用している場合は、データ・セットのタイプが **%XML.DataSet** であることを確認する必要があります。
  - 引数または戻り値としてコレクション (**%ListOfObjects** または **%ArrayOfObjects**) を使用するには、コレクションの ELEMENTTYPE パラメータが設定され、XML に対応するクラスを参照することを確認する必要があります。

重要　多くの場合、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。Web メソッド内では、メソッドの動作を調整するために、しばしば Web サービス・インスタンスのプロパティを設定してからメソッドを呼び出すことが必要となります。クラス・メソッドでは、これらの作業をできないため、多くの場合クラス・メソッドは Web メソッドに適していません。

その他の注意事項は、“Web サービスの作成” の “基本要件” を参照してください。

2.3.1 InterSystems IRIS Interoperability Web メソッドの基本的な手順

プロダクション Web サービス内で、Web メソッドは一般的に以下のことを行います。

1. 受信 SOAP メッセージの情報を使って、要求メッセージを作成し、プロパティを設定します。
2. ビジネス・サービスの適切なメソッドを呼び出して、要求をプロダクション内の宛先に送信します。具体的には、SendRequestSync()、SendRequestAsync()、または (あまり一般的ではない) SendDeferredResponse() を呼び出します。詳細は、″プロダクションの開発″ の “要求メッセージの送信” を参照してください。

   これらの各メソッドは、ステータス (具体的には、**%Status** のインスタンス) を返します。
3. 前の手順で返されたステータスを確認して、適切に対応します。

4. 次に、以下の手順を実行します。

   - 成功した場合は、参照によって返された応答メッセージを調べ、そのメッセージを使用して Web メソッドの戻り値を作成します。以前説明したように、戻り値を SOAP 応答としてパッケージ化するには、戻り値が XML に対応していなければなりません。
   - 失敗した場合は、Web サービスの ReturnMethodStatusFault() または ReturnStatusFault() メソッドを呼び出し、SOAP フォールトを返して、Ens.Alert を生成できるようにします。詳細は次の節を参照してください。

2.3.2 呼び出し側へのフォールトの返信

デフォルトでは、Web メソッドの実行中にエラーが発生すると、Web サービスは呼び出し側に SOAP メッセージを返しますが、このメッセージにはフォールトが発生した正確な場所が示されません。以下に例を示します。

```
<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: <INVALID OREF>
          zGetCustomerInfo+10^ESOAP.WebService.1</text>
    </error>
 </detail>
 </SOAP-ENV:Fault>
</SOAP-ENV:Body>
```

Web メソッドでエラーを確認し、ReturnMethodStatusFault() または ReturnStatusFault() を使用します。以下のように、エラーが発生した場合には、メッセージに含まれる情報が多くなります。

```
<SOAP-ENV:Body>
 <SOAP-ENV:Fault>
  <faultcode>SOAP-ENV:Method</faultcode>
  <faultstring>Server Application Error</faultstring>
  <faultactor>ESOAP.WebService</faultactor>
  <detail>
    <error xmlns='http://www.myapp.org' >
     <text>ERROR <Ens>ErrException:
           <DIVIDE>zGetCustomerRequest+8^ESOAP.MyOperation.1 -
           logged as '13 Jul 2007' number 4 @'    set x=100/0'</text>
   </error>
 </detail>
 </SOAP-ENV:Fault>
</SOAP-ENV:Body>
```

ReturnMethodStatusFault() および ReturnStatusFault() メソッドは、呼び出し側に SOAP フォールトを返信し、(設定に応じて) アラートを作成する例外を生成します。これらのメソッドには、以下のシグニチャがあります。

```
ClassMethod ReturnStatusFault(pCode As %String,
                              pStatus As %Status)

ClassMethod ReturnMethodStatusFault(pStatus As %Status)
```

ここで、

- pCode は、SOAP フォールトの <faultcode> 要素で使用する、エラー・コードを表す文字列です。ReturnMethodStatusFault() メソッドは、汎用エラー・コード `SOAP-ENV:Method` を使用します。
- pStatus は、返される SOAP フォールトで使用するステータスです。これは、SOAP フォールトの詳細を作成するために使用されます。

また、これらのメソッドは SOAP フォールトの <faultactor> 要素も設定します。

2.3.3 例

簡単な例を以下に示します。

Class Member

```
Method GetCustomerInfo(ID As %Numeric) As ESOAP.SOAPResponse [WebMethod]
{
    //create request message with given ID
    set request=##class(ESOAP.CustomerRequest).%New()
    set request.CustomerID=ID

    //send request message
    set sc= ..SendRequestSync("GetCustomerInfoBO",request,.response)
    if $$$ISERR(sc) do ..ReturnMethodStatusFault(sc)

    //use info from InterSystems IRIS response to create SOAP response
    set soapresponse=##class(ESOAP.SOAPResponse).%New()
    set soapresponse.CustomerID=response.CustomerID
    set soapresponse.Name=response.Name
    set soapresponse.Street=response.Street
    set soapresponse.City=response.City
    set soapresponse.State=response.State
    set soapresponse.Zip=response.Zip

    quit soapresponse
}
```

2.4 WSDL の表示

InterSystems IRIS では、Web サービスは InterSystems IRIS Web アプリケーション内で実行されます。その Web アプリケーションは、ユーザが選択した Web サーバ (管理ポータルを処理するのと同じ Web サーバ) により処理されます。

InterSystems IRIS は、Web サービスについて記述する WSDL ドキュメントを自動的に作成および発行します。Web サービスの修正やリコンパイルを行うたびに、InterSystems IRIS は対応する WSDL を自動的に更新します。

URL の形式は以下のとおりです。<baseURL> はインスタンスのベース URL です。

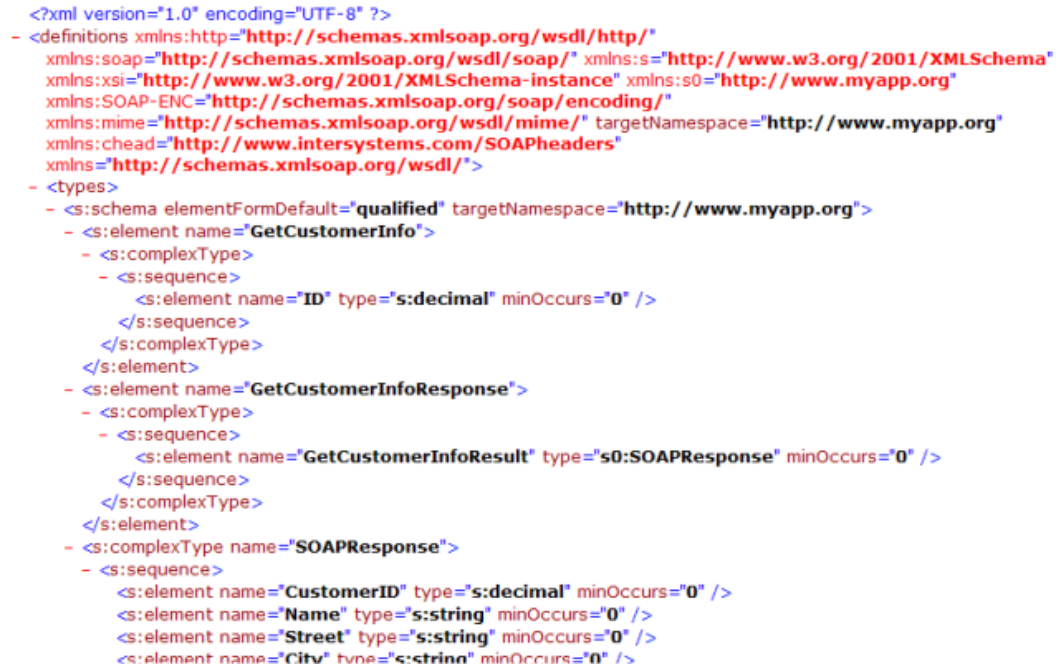
```
https:<baseURL>/csp/app/web_serv.cls?WSDL
```

/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。(一般的に、/csp/app は /csp/namespace です。namespace は、Web アプリケーションとプロダクションが存在するネームスペースです。)

以下に例を示します。

```
https://devsys/csp/mysamples/MyApp.StockService.cls?WSDL
```

ブラウザに WSDL ドキュメントが XML ドキュメントとして表示されます。以下に例を示します。



2.5 Web サービス例

以下の例は、顧客 ID から顧客情報を検索するために使用するプロダクション Web サービスを示しています。

Class Definition

```
Class ESOAP.WebService Extends EnsLib.SOAP.Service
{

Parameter ADAPTER;

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

Parameter SERVICENAME = "CustomerLookupService";

Method GetCustomerInfo(ID As %Numeric) As ESOAP.SOAPResponse [WebMethod]
{
    //create request message with given ID
    set request=##class(ESOAP.CustomerRequest).%New()
    set request.CustomerID=ID

    //send request message
    set sc= ..SendRequestSync("GetCustomerInfoBO",request,.response)
    if $$$ISERR(sc) do ..ReturnMethodStatusFault(sc)

    //use info from InterSystems IRIS response to create SOAP response
    set soapresponse=##class(ESOAP.SOAPResponse).%New()
    set soapresponse.CustomerID=response.CustomerID
    set soapresponse.Name=response.Name
    set soapresponse.Street=response.Street
    set soapresponse.City=response.City
    set soapresponse.State=response.State
    set soapresponse.Zip=response.Zip

    quit soapresponse
}

}
```

SOAP 応答クラスは以下のとおりです。

Class Definition

```
///
Class ESOAP.SOAPResponse Extends (%RegisteredObject, %XML.Adaptor)
{

Property CustomerID As %Numeric;
Property Name As %String;
Property Street As %String;
Property City As %String;
Property State As %String;
Property Zip As %Numeric;

}
```

以下の点に注意してください。

- 例の Web メソッド (`GetCustomerInfo`) は SendRequestSync() を使用して、プロダクションの別の場所にあるビジネス・オペレーションと通信します。このメソッドは応答メッセージを受信し、それを使用して SOAP 応答メッセージを作成します。
- SOAP 応答クラスは、対応するプロダクション・メッセージ応答クラスと同じプロパティを持ちます。ただし、プロダクション・メッセージ応答と異なり、SOAP 応答クラスは XML 対応であり非永続的です。

2.6 SOAP セッションの有効化

SOAP 仕様には、セッション・サポートは含まれていません。ただし、Web クライアントと使用する Web サービスとの間でセッションを維持することが有用な場合がよくあります。プロダクション Web サービスを使用して、これを行うことができます。Web サービスがセッションを使用する場合、セッション ID が確立され、クライアントからの呼び出しが一度正しく認証されると、その後はサービスに対する繰り返しの呼び出しが許可されます。

SOAP セッションのサポートは、SOAPSESSION クラス・パラメータによって制御されます。デフォルトは 0 です。これは、Web サービスがセッションを使用しないことを意味します。

SOAP セッションを有効化するには、EnsLib.SOAP.Service のサブクラスを作成し、このサブクラスで SOAPSESSION を 1 に設定します。このサブクラスをプロダクション Web サービスの基本クラスにします。

SOAP セッションの詳細は、ドキュメントの "Web サービスおよび Web クライアントの作成" を参照してください。

2.7 その他のオプション

プロダクション Web サービスは %SOAP.WebService を拡張するので、そのクラスによって提供されるすべての SOAP サポートを使用できます。このサポートには、以下のカスタマイズ・オプションが含まれます。

- SOAP ヘッダのカスタマイズ
- SOAP メッセージにおけるアタッチメントの受け渡し
- SOAP メッセージのバインディング・スタイルをドキュメント・スタイル (デフォルト) から RPC スタイルに変更
- メッセージのエンコーディング・スタイルをリテラル・エンコーディング (デフォルト) から SOAP エンコーディングに変更
- SOAP メッセージで使用される XML タイプのカスタマイズ
- Web メソッドの呼び出しに使用される `SOAPAction` ヘッダのカスタマイズ
- 要素が限定要素かどうかの制御 (Web サービスの elementFormDefault 属性の制御)

- NULL 引数のフォームの制御 (省略要素ではなく空の要素とする)
- 戻り値ではなく出力パラメータを持つ Web メソッドの記述

これらのオプションおよびその他のオプションについては、ドキュメント・セットの "Web サービスおよび Web クライアントの作成" を参照してください。

2.8 Web サービスの追加と構成

プロダクションにプロダクション Web サービス (ビジネス・サービス) を追加するには、管理ポータルを使用して以下の操作を行います。

1. カスタム・クラスのインスタンスをプロダクションに追加します。

   重要　　構成名が、パッケージを含む完全なクラス名と同じであることを確認します。これは、プロダクション Web サービスを実行するための要件です。

2. ビジネス・サービスを有効化します。
3. プール・サイズの設定を 0 にします。

   その他の設定については、"プロダクションの構成" を参照してください。
4. プロダクションを実行します。

3

プロダクション内での Web クライアントの作成

ここでは、InterSystems IRIS® Interoperability Web クライアントの作成方法を説明します。高レベルで、InterSystems IRIS Interoperability Web クライアントはプロダクション内の他の場所から InterSystems IRIS Interoperability 要求を受信し、SOAP 要求に変換して適切な Web サービスに送信します。同様に、SOAP 応答を受信して InterSystems IRIS 応答に変換し、プロダクション内の場所に送信します。

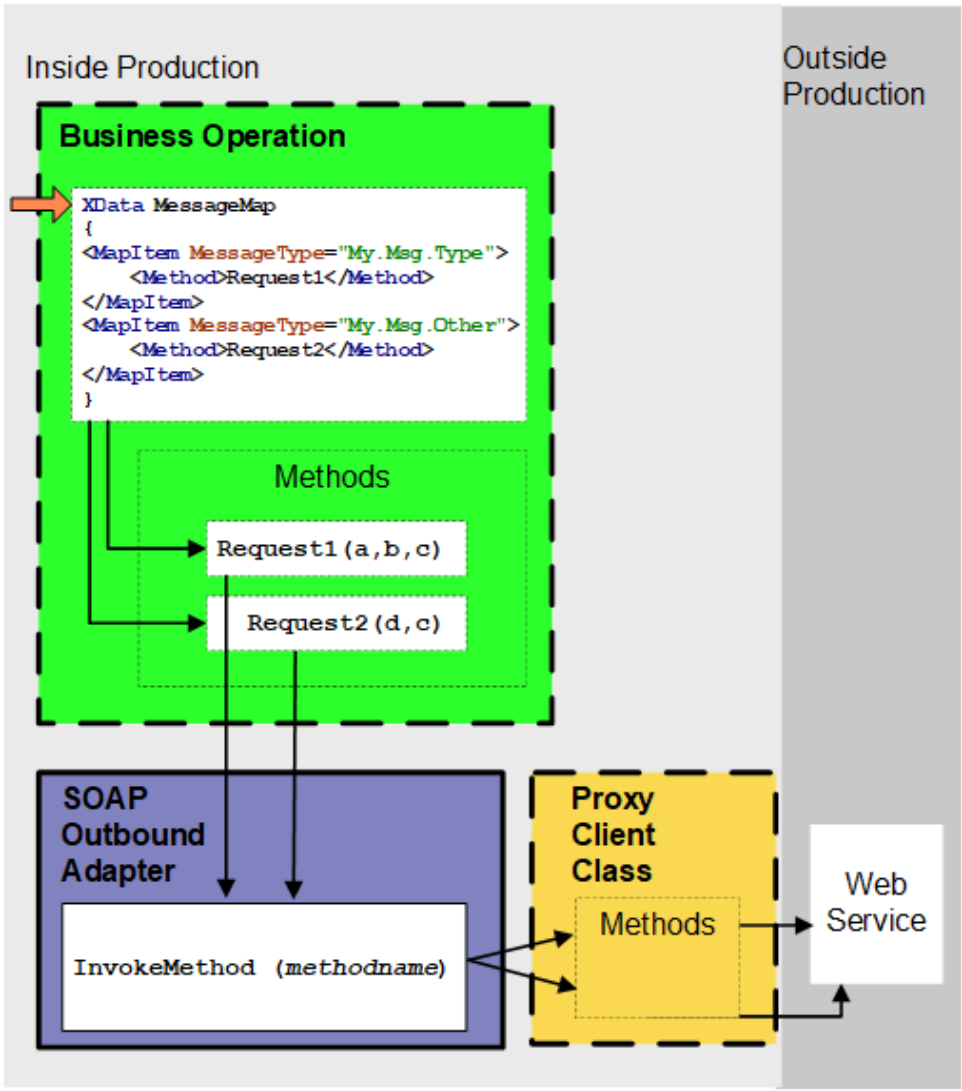
Tipヒント InterSystems IRIS では、SOAP を使用する特殊なビジネス・サービス・クラスとユーザのニーズに適したビジネス・サービス・クラスの 1 つも提供されます。そのため、プログラミングの必要がありません。″相互運用プロダクションの概要″ の “接続オプション” を参照してください。

3.1 概要

InterSystems IRIS Interoperability Web クライアントは、以下の要素で構成されます。

- プロキシ・クライアント・クラス。Web サービスによって定義される各メソッドに対してプロキシ・メソッドを定義します。各プロキシ・メソッドは、対応する Web サービス・メソッドが使用するものと同じシグニチャを使用し、要求に応じてそのメソッドを呼び出します。
- ビジネス・オペレーション。InterSystems IRIS Interoperability SOAP 送信アダプタを使用して、プロキシ・クライアントを呼び出します。
- サポート・クラス。XML タイプおよびプロダクション・メッセージの定義に必要となります。

以下の図は、ビジネス・オペレーション、アダプタ、およびプロキシ・クライアント・クラスが連携して機能するしくみについて示しています。サポート・クラスはここには示されていません。



上の図で、点線で示されたすべての項目は、スタジオの SOAP クライアント・ウィザードで生成できます。このコードは、必要に応じて編集することができます。

これらの要素の詳細は、このドキュメントの “InterSystems IRIS による Web クライアントのサポート” を参照してください。

3.2 基本的な手順

この節では、InterSystems IRIS Interoperability Web クライアントを作成するための基本的な手順を説明します。

InterSystems IRIS Interoperability Web クライアントを作成するには、スタジオで以下の操作を行います。

1. SOAP ウィザードを使用して、ビジネス・オペレーション・クラス、プロキシ・クライアント・クラス、およびサポート・クラスを生成します。このツールについては、″SOAP ウィザードの使用法″ で説明します。
2. メソッドの入力および出力のタイプを調整する必要があるかどうかを確認します。

3.2.1 SOAP ウィザードの使用法

SOAP ウィザードを使用して InterSystems IRIS Interoperability Web クライアントを生成するには、以下の操作を行います。

1. スタジオで、適切なネームスペースにいることを確認します。
2. **[ツール]→[アドイン]→[SOAP ウィザード]** の順にクリックします。
3. 最初の画面で、使用する Web サービスの WSDL ドキュメントの URL 全体を入力します。
4. **[次へ]** をクリックします。
5. **[パッケージでビジネス・オペレーションを作成]** の左側のチェックボックスにチェックを付けます。このオプションは、ウィザードで、プロキシ・クライアントを呼び出すビジネス・オペレーションクラス、およびそのビジネス・オペレーションと共に使用するメッセージ・クラスを定義するためのものです。
6. **[パッケージでビジネス・オペレーションを作成]** で、必要に応じて、サブパッケージ名を BusOp から別の名前に変更します。
7. **[オプションのパッケージ名]** に、プロキシ・クライアントおよび関連するクラスのパッケージ名を入力します。デフォルトのパッケージ名は、サービス名です。“生成されるクラスと XMLKEEPCLASS” も参照してください。
8. **[クラスの種類]** で、プロキシ・クライアント・クラスの一般的なタイプ、つまり永続またはシリアル (デフォルト) を選択します。
9. **[次へ]** をクリックします。ウィザードによってクラスが生成およびコンパイルされ、これらのクラスのリストが表示されます。
10. **[終了]** をクリックします。

3.2.1.1 生成されるクラスと XMLKEEPCLASS

SOAP ウィザードによって、一連のクラスが生成されます。これらのクラスの詳細については後述します。

ツールがプロキシ・クライアント・クラスおよびサポート・クラスを作成する、パッケージを指定します。このパッケージが既存のパッケージと同じ場合、デフォルトでは、ツールは既存の同じ名前のクラスを上書きします。ウィザードによってクラス定義が上書きされないようにするには、そのクラスに XMLKEEPCLASS パラメータを追加し、このパラメータの値を 1 に設定します。

3.2.1.2 Process() メソッドの使用法

ウィザードを使用する代わりに、%SOAP.WSDL.Reader クラスの Process() メソッドを使用できます。このメソッドを使用するには、以下を実行します。

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。

プロパティ	目的
CompileFlags	生成されたクラスをコンパイルするときに使用するフラグを指定します。最初の式は "dk" で、$System.OBJ.ShowFlags() を使用して利用可能なフラグに関する情報を取得します。
MakePersistent	このプロパティが 1 の場合、プロキシ・クライアントは永続オブジェクトです。それ以外の場合は登録オブジェクトです。最初の式は 0 です。
MakeSerial	このプロパティが 1 で、MakePersistent が 1 の場合、プロキシ・クライアントはシリアル・クラスです。最初の式は 0 です。

プロパティ	目的
OutputTypeAttribute	WSDL リーダが、生成するプロキシ・クライアントの OUTPUTTYPEATTRIBUTE パラメータをどのように設定するかを制御します。これは、SOAP メッセージの xsi:type 属性の使用を制御します。ドキュメント・セットの "Web サービスおよび Web クライアントの作成" を参照してください。
MakeBusinessOperation	ビジネス・オペレーション、および関連する要求オブジェクトおよび応答オブジェクトを生成するかどうかを指定します。このビジネス・オペレーションの ADAPTER 設定は、EnsLib.SOAP.OutboundAdapter です。このオプションは、相互運用対応ネームスペース内で作業している場合にのみ機能します。

その他のプロパティについては、%SOAP.WSDL.Reader のクラスに関するドキュメントを参照してください。

3. Process() メソッドを呼び出し、以下の引数を提供します。
   - 最初の引数は、Web サービスの WSDL の URL または WSDL ファイルの名前 (完全なパスを含む) 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提供する文字列を追加しなければならない場合があります。以下の例を参照してください。
   - オプションの 2 つ目の引数は、リーダが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。

3.3 InterSystems IRIS Interoperability Web クライアントに対して生成されるクラス

この節では、SOAP ウィザードによって生成されるクラスについての情報を提供します。

例として、以下の条件を満たす `MyService` という名前の Web サービスについて考えてみましょう。

- ホストは `https://devsys/csp/mysamples/MyApp.AddService.CLS`
- この Web サービスのターゲット XML ネームスペースは `http://www.myapp.org`
- この Web サービスは `AddService` という名前の Web メソッドを定義します。このメソッドは、引数として 2 つの複素数を受け取り、結果を返します。

ここでは、SOAP ウィザードを使用して、この Web サービスの InterSystems IRIS Interoperability Web クライアントを生成すると想定します。このクライアントのクラスのパッケージを `MyClient` として指定すると、SOAP ウィザードは以下のクラスを生成し、これらすべてを現在のプロジェクトに追加します。

- ビジネス・オペレーションを定義する、`MyClient.BusOp.MyServiceSoap` クラスを生成します。

```
Class MyClient.BusOp.MyServiceSoap Extends Ens.BusinessOperation
{

Parameter ADAPTER = "EnsLib.SOAP.OutboundAdapter";

Method Add(pRequest As MyClient.BusOp.AddRequest,
Output pResponse As MyClient.BusOp.AddResponse) As %Library.Status
{
   Set ..Adapter.WebServiceClientClass = "MyClient.MyServiceSoap"
   Set tSC = ..Adapter.InvokeMethod("Add",.AddResult,
pRequest.a,pRequest.b) Quit:$$$ISERR(tSC) tSC

   Set tSC = pRequest.NewResponse(.pResponse) Quit:$$$ISERR(tSC) tSC
   Set pResponse.AddResult=AddResult
   Quit $$$OK
}
```

```
XData MessageMap
{
<MapItems>
    <MapItem MessageType="MyClient.BusOp.AddRequest">
        <Method>Add</Method>
    </MapItem>
</MapItems>
}

}
```

- プロキシ・クライアント・クラス、MyClient.AddServiceSOAP クラスを生成します。

  Class Definition

```
Class MyClient.AddServiceSoap Extends %SOAP.WebClient
{

/// This is the URL used to access the web service.
Parameter LOCATION = "https://devsys/csp/mysamples/MyApp.AddService.cls";

/// This is the namespace used by the Service
Parameter NAMESPACE = "http://www.myapp.org";

/// Use xsi:type attribute for literal types.
Parameter OUTPUTTYPEATTRIBUTE = 1;

/// This is the name of the Service
Parameter SERVICENAME = "AddService";

Method Add(a As MyClient.ComplexNumber, b As MyClient.ComplexNumber)
   As MyClient.ComplexNumber [ Final,
   SoapBindingStyle = document, SoapBodyUse = literal, WebMethod ]
{
 Quit ..WebMethod("Add").Invoke($this,
      "http://www.myapp.org/MyApp.AddService.Add",.a,.b)
}

}
```

- ビジネス・オペレーションに必要な要求および応答メッセージ・クラスを生成します。要求クラスは以下のとおりです。

  Class Definition

```
Class MyClient.BusOp.AddRequest Extends Ens.Request
{

Parameter RESPONSECLASSNAME = "MyClient.BusOp.AddResponse";

Property a As MyClient.ComplexNumber;

Property b As MyClient.ComplexNumber;

}
```

  応答クラスは以下のとおりです。

  Class Definition

```
Class MyClient.BusOp.AddResponse Extends Ens.Response
{

Property AddResult As MyClient.ComplexNumber;

}
```

- 最後に、MyClient.ComplexNumber クラスを生成します。このクラスは複素数を定義し、その他のクラスによって使用されます。

Class Definition

```
/// Created from: http://devsys/csp/mysamples/MyApp.AddService.CLS?WSDL=1
Class MyClient.ComplexNumber Extends (%RegisteredObject, %XML.Adaptor)
{
Parameter XMLNAME = "ComplexNumber";

Parameter XMLSEQUENCE = 1;

Property Real As %xsd.double(XMLNAME = "Real") [ SqlFieldName = _Real ];

Property Imaginary As %xsd.double(XMLNAME = "Imaginary");

}
```

これらのクラスをコンパイルすると、コンパイラは、Web サービスで定義される各メソッドに対するクラスも生成します。これらのクラスはプロジェクトに自動的には追加されず、内容は変更される可能性があります。これらのクラスは %SOAP.ProxyDescriptor を拡張します。ユーザは、このクラスに対して自分でサブクラスを作成しないでください。

3.4 手動でのビジネス・オペレーション・クラスの作成

SOAP ウィザードによって生成されるビジネス・オペレーション・クラスを使用する代わりに、独自のクラスを作成できます。この節では、その方法について説明します。以下について説明します。

- ビジネス・オペレーション・クラスの基本要件
- メソッドの基本要件
- プロキシ・メソッドを呼び出すための具体的なテクニック (例を含む)
- ここで使用されるアダプタ・プロパティおよびメソッドについての参照情報

3.4.1 クラスの基本要件

ビジネス・オペレーション・クラスの基本要件を以下に列挙します。

- ビジネス・オペレーション・クラスは、Ens.BusinessOperation を拡張するものでなければなりません。
- クラスの ADAPTER パラメータは EnsLib.SOAP.OutboundAdapter である必要があります。
- クラスの INVOCATION パラメータは、使用する呼び出しスタイルを指定する必要があります。以下のいずれかを使用します。
  - Queue は、メッセージが 1 つのバックグラウンド・ジョブ内で作成され、元のジョブが解放された段階でキューに配置されることを意味します。その後、メッセージが処理された段階で、別のバックグラウンド・ジョブがそのタスクに割り当てられます。これは最も一般的な設定です。
  - InProc は、メッセージが、作成されたジョブと同じジョブで生成、送信、および配信されることを意味します。このジョブは、メッセージが対象に配信されるまで送信者のプールに解放されません。これは特殊なケースのみに該当します。
- 以下の節で説明するように、クラスでは、プロキシ・クライアントの各メソッドに対して 1 つのメソッドを定義します。

- クラスでは、各メソッドに対して 1 つのエントリを含むメッセージ・マップを定義します。メッセージ・マップは、以下の構造を持つ XData ブロック・エントリです。

```
XData MessageMap
{
<MapItems>
  <MapItem MessageType="messageclass">
    <Method>methodname</Method>
  </MapItem>
  ...
</MapItems>
}
```

また、ビジネス・オペレーションが使用するプロダクション・メッセージ・クラスも定義する必要があります。

3.4.2 メソッドの基本要件

ビジネス・オペレーション・クラスにおいて、メソッドがプロキシ・メソッドを呼び出す必要があります。一般的な要件は以下のとおりです。

1. このメソッドは、呼び出しているプロキシ・メソッドと同じシグニチャを持つ必要があります。
2. このメソッドには、WebMethod キーワードでマーキングする必要があります。
3. このメソッドは、プロキシ・クライアントによって期待される、SoapBindingStyle キーワードおよび SoapBodyUse キーワードを設定する必要があります。(つまり、対応するプロキシ・メソッドのシグニチャと同じ値を使用します。)
4. このメソッドは、アダプタの **WebServiceClientClass** プロパティを設定する必要があります。このプロパティが設定されている場合、プロキシ・クライアントのインスタンスが作成され、アダプタの **%Client** プロパティ内に配置されます。
5. このメソッドは、次の節で説明するいずれかの方法で、対応するプロキシ・メソッドを呼び出す必要があります。
6. このメソッドは、ステータスを確認する必要があります。
7. 次に、以下の手順を実行します。
   - 成功した場合は、(要求の NewResponse() メソッドを介して) 新しい応答メッセージを作成し、適切にプロパティを設定します。
   - 失敗した場合は、エラーが表示されて停止します。

3.4.3 プロキシ・メソッドを実行する方法

ビジネス・オペレーションにおいて、メソッドが、プロキシ・クライアント・クラスのプロキシ・メソッドを実行する必要があります。これには何通りかの方法があるので、例を使用して説明します。この節では、`GetStock` という名前の Web メソッドを持つサンプル Web サービスを使用します。このメソッドは、ストック・シンボル (文字列) を受け取り、数値を返します。SOAP ウィザードを使用して、`GetStock` という名前のメソッドを含むプロキシ・クライアント (`GetStock.StockServiceSoap`) を生成済みであると想定します。

また、以下のようにメッセージ・クラスを作成したものとします。

Class Definition

```
Class GetStock.BusOp.GetQuoteRequest Extends Ens.Request
{

Parameter RESPONSECLASSNAME = "GetStock.BusOp.GetQuoteResponse";

Property StockName As %String;

}
```

および

Class Definition

```
Class GetStock.BusOp.GetQuoteResponse Extends Ens.Response
{

Property StockValue As %Numeric;

}
```

プロキシ・メソッド GetStock を実行するために、ビジネス・オペレーション・クラスは、以下のいずれかを行うことができます。

- アダプタの InvokeMethod() メソッドを呼び出し、実行するプロキシ・メソッドの名前、および任意の数の引数を指定します。この場合、引数は 1 つだけ (pRequest.StockName) です。以下に例を示します。

  ### Class Member

  ```
  Method GetQuote1(pRequest As GetStock.BusOp.GetQuoteRequest,
  Output pResponse As GetStock.BusOp.GetQuoteResponse) As %Status
  {
   set ..Adapter.WebServiceClientClass = "GetStock.StockServiceSoap"

   set status = ..Adapter.InvokeMethod("GetQuote",.answer,pRequest.StockName)
   if $$$ISERR(status) quit status

   set pResponse=##class(GetStock.BusOp.GetQuoteResponse).%New()
   set pResponse.GetQuoteResult=answer
   quit $$$OK
  }
  ```

  SOAP ウィザードを使用してビジネス・オペレーションを生成すると、そのビジネス・オペレーションではこのメソッドが使用されます。

- アダプタの %Client プロパティにアクセスします。このプロパティによってプロキシ・クライアント・クラスのインスタンスが与えられ、そのプロパティのプロキシ・メソッドが実行されます。%Client プロパティは、WebServiceClientClass プロパティの設定時に設定されます。この場合、%Client には、文字列ストック・シンボルを受け取る、GetQuote という名前のメソッドがあります。以下に例を示します。

  ### Class Member

  ```
  Method GetQuote2(pRequest As GetStock.BusOp.GetQuoteRequest,
  Output pResponse As GetStock.BusOp.GetQuoteResponse) As %Status
  {
   set ..Adapter.WebServiceClientClass = "GetStock.StockServiceSoap"

   set client=..Adapter.%Client
   set answer=client.GetQuote("GRPQ")

   set pResponse=##class(GetStock.BusOp.GetQuoteResponse).%New()
   set pResponse.GetQuoteResult=answer
   quit $$$OK
  }
  ```

  この方法を使用する場合、InterSystems IRIS の再試行ロジックにはアクセスできません。

- アダプタの WebMethod() メソッドを呼び出すことによって、プロキシ・メソッド・オブジェクトを作成します。このオブジェクトのプロパティを適切に設定します (名前のある引数ごとに 1 つのプロパティ)。ここでは、WebMethod() は、1 つのプロパティ (StockName) を持つオブジェクトを返します。必要に応じてプロパティを設定した後、オブジェクトの Invoke() メソッドを呼び出します。以下に例を示します。

Class Member

```
Method GetQuote3(pRequest As GetStock.BusOp.GetQuoteRequest,
Output pResponse As GetStock.BusOp.GetQuoteResponse) As %Status
{
 set ..Adapter.WebServiceClientClass = "GetStock.StockServiceSoap"


 set proxymethod=..Adapter.WebMethod("GetQuote")
 set proxymethod.StockName=pRequest.StockName


 set status=..Adapter.Invoke(proxymethod)
 if $$$ISERR(status) quit status


 set pResponse=##class(GetStock.BusOp.GetQuoteResponse).%New()
 set pResponse.GetQuoteResult=proxymethod.%Result
 quit $$$OK
}
```

この場合、任意の数だけ引数を提供できます。

3.4.4 参照情報

この節では、これまでに説明したアダプタ・プロパティおよびメソッドについての参照情報を提供します。

%Client プロパティ

```
%SOAP.WebClient
```

プロキシ・クライアントの関連するインスタンス (%SOAP.WebClient のインスタンス)。このプロパティは、アダプタの WebServiceClientClass プロパティの設定時に設定されます。

InvokeMethod() メソッド

```
Method InvokeMethod(pMethodName As %String,
       Output pResult As %RegisteredObject,
       pArgs...) As %Status
```

すべての引数を渡してプロキシ・クライアント・クラスの指定されたメソッドを呼び出して、ステータスを返します。出力は、2 番目の引数として参照によって返されます。

WebMethod() メソッド

```
Method WebMethod(pMethodName As %String) As %SOAP.ProxyDescriptor
```

指定されたメソッドに対応するオブジェクトを返します。このオブジェクトには、各メソッド引数に対応する 1 つのプロパティがあります。Invoke() メソッドを使用する前に、このプロパティを設定します。%SOAP.ProxyDescriptor の詳細は、クラス参照を参照してください。

Invoke() メソッド

```
Method Invoke(pWebMethod As %SOAP.ProxyDescriptor) As %Status
```

指定されたメソッドを呼び出し、ステータスを返します。

3.5 Web クライアントの追加と構成

InterSystems IRIS Interoperability Web クライアントをプロダクションに追加するには、管理ポータルを使用して以下の操作を行います。

1. カスタム・ビジネス・オペレーション・クラス、具体的には SOAP ウィザードによって生成されるビジネス・オペレーション・クラスのインスタンスを、プロダクションに追加します。
2. ビジネス・オペレーションを有効化します。
3. 以下の説明に従って、関連するアダプタの実行時設定に適切な値を指定します。
4. プロダクションを実行します。

以降の節では、InterSystems IRIS Interoperability Web クライアントの実行時設定について説明します。実行時設定は、次のように複数の一般的なグループに分けられます。

- 基本設定
- 資格情報に関する設定
- TLS の使用を制御する設定
- プロキシ・サーバの使用を制御する設定

このドキュメントに記載されていない設定は、"プロダクションの管理" の "すべてのプロダクションに含まれる設定" を参照してください。

3.5.1 基本設定の指定

以下の設定は、InterSystems IRIS Interoperability Web クライアントの基本的な情報を指定します。

- ウェブサービスURL
- ウェブサービスクライアントクラス
- 応答タイムアウト

3.5.2 資格情報の指定

Web サービスにアクセスするのに、ユーザ名とパスワードが必要な場合があります。通常、InterSystems IRIS Interoperability SOAP クライアントは、以下のいずれかの方法で Web サービスにログインできます。

- WS-Security ユーザ認証を使用できます。この場合、SOAP 要求に WS-Security ヘッダを含めます。このヘッダに、ユーザ名とパスワードを含めます。[SOAP認証情報] 設定の値を指定すると、プロキシ・クライアントによってこの操作が自動的に行われます。

  注意　Web クライアントと Web サービスとの間で TLS を使用していることを確認します。WS-Security ヘッダはクリア・テキストで送信されるので、TLS が使用されていない場合、この方法は安全ではありません。

- 基本的な HTTP ユーザ認証を使用することができます。WS-Security ほど安全性は高くありませんが、この方法が必要な場合もあります。この場合、SOAP 要求の HTTP ヘッダにユーザ名とパスワードを含めます。[認証情報] 設定の値を指定すると、プロキシ・クライアントによってこの操作が自動的に行われます。

使用する Web サービスに適した方法を使用してください。

3.5.3 TLS 構成の指定

Web サーバが TLS をサポートしている場合、これを使用して接続できます。そのためには、[SSL構成] 設定に対して値を指定します。

注釈　また、Web サービスが `https://` を使用する URL にあることを確認する必要があります。Web サービスの場所は [ウェブ・サービスURL] 設定によって決まります。この設定が指定されていない場合、InterSystems IRIS Interoperability Web クライアントは、プロキシ・クライアント・クラスの LOCATION パラメータによって指定される URL に Web サービスがあると仮定します。

3.5.4 プロキシ・サーバの指定

プロキシ・サーバを介して Web サービスと通信できます。これを設定するには、**[プロキシ設定]** グループの Proxy Server およびその他の設定を使用します。

A

SOAP サービス向けのプロダクションの構成

この付録では、Web ポートを通じて HTTP および SOAP サービスを使用できるようにシステムを構成する方法を簡単に説明します。ここで紹介する情報は、これらのサービス用の開発システムやテスト・システムをセットアップするのに役立ちます。これらのトピックに関する包括的な情報は、ドキュメントに記載されています。詳細は、″Configuring System-Wide Settings″ を参照してください。

InterSystems IRIS® の開発システムまたはテスト・システムを HTTP または SOAP サービス向けにセットアップするには、以下の手順を実行します。

1. InterSystems IRIS をロックダウンされたインストール環境にインストールした場合は、スタジオへのアクセスは無効になっています。管理ポータルを開いてスタジオへのアクセスを有効にします。
   a. InterSystems IRIS ランチャーから管理ポータルを開始します。管理ポータルにアクセスするには、_system ではなく Windows のログイン・ユーザ名を使用する必要があります。インストール時に指定したパスワードを入力します。
   b. **[システム管理]**、**[セキュリティ]**、**[サービス]** の順に選択して、[サービス] ポータル・ページを表示します。
   c. デフォルトでは、**[%Services_Bindings]** サービスは無効になっています。このサービス名を選択し、**[サービス有効]** チェック・ボックスにチェックを付けて、設定を保存します。
2. 既存のネームスペースを使用しない場合は、新しいネームスペースを作成します。
   a. **[システム管理]**、**[構成]**、**[システム構成]**、**[ネームスペース]** の順に選択して、[ネームスペース] ポータル・ページを表示します。
   b. **[新規ネームスペース作成]** ボタンをクリックして、ネームスペースの名前を指定します (例 : SERVICESNS)。
   c. グローバル用データベースについて **[新規データベース作成]** ボタンをクリックします。
   d. データベース・ウィザードで、グローバル用データベースの名前を入力します (例 : SERVICES_GDB)。ウィザードはこの名前を使用して、このデータベース用のディレクトリを作成します。
   e. **[次へ]** ボタンを 2 回クリックして、[データベースリソース] フォームを表示します。**[新規リソース作成]** ラジオ・ボタンを選択します。ウィザードで [新規リソース作成] フォームが表示されます。表示されている名前 (%DB_SERVICES_GDB など) をそのまま使用して、[パブリック許可] の [読み込み] と [書き込み] というチェック・ボックスのチェックが外れていることを確認します。[データベースリソース] フォームの **[保存]** ボタンをクリックして、[データベースウィザード] フォームの **[完了]** ボタンをクリックします。
   f. ルーチン用データベースについて手順 c から e を繰り返します。
   g. **[保存]** ボタンをクリックして、ネームスペースの作成を完了します。
   h. **[閉じる]** をクリックしてログを閉じます。
3. 空のロールを作成して、このロールを不明ユーザに割り当てます。

a. **[システム管理]**、**[セキュリティ]**、**[ロール]** の順に選択して、[ロール] ポータル・ページを表示します。

b. **[新規ロール作成]** ボタンをクリックして、ロールに名前を付けて (例 : Services_Role)、**[保存]** ボタンをクリックします。

c. **[メンバ]** タブを選択して、[不明ユーザ] を選択して、右矢印をクリックして、**[割り当てる]** ボタンをクリックします。

4. Web ポートへの呼び出しを処理する Web アプリケーションを定義します。この Web アプリケーションの名前によって、サービスを呼び出す URL のルートが指定されます。単一の Web アプリケーションは複数のビジネス・サービスをサポートできますが、これらのビジネス・サービスはすべて、同じクラスを持っているか、その Web アプリケーションのディスパッチ・クラスのサブクラスを持っている必要があります。

a. **[システム管理]**、**[セキュリティ]**、**[アプリケーション]**、**[Web アプリケーション]** の順に選択して、**[Web アプリケーション]** ポータル・ページを表示します。**[新規 Web アプリケーション作成]** ボタンをクリックします。

b. この Web アプリケーションに名前を付けます (例 : `/weatherapp` や `/math/sum`)。名前の先頭には / (スラッシュ) を付加する必要があります。

c. **[ネームスペース]** で、当該プロダクションが実行されているネームスペースを指定します (例 : SERVICESNS)。[ネームスペースのデフォルト・アプリケーション] チェックボックスのチェックは外したままにします。

d. [アプリケーション]、[CSP/ZEN]、および [着信 Web サービス] の各チェック・ボックスにはチェックを付けてもかまいません。

e. [必須リソース] フィールドと [ID でグループ分け] フィールドは空白のままにします。

f. **[許可された認証方法]** 行の [認証なし] チェック・ボックスにチェックを付けます。

   またはこれに代わる方法について、このリストの後のヒントを参照してください。

g. [ディスパッチ・クラス] でコンポーネント・クラスを指定します (例 : EnsLib.REST.GenericService や EnsLib.SOAP.GenericService)。

h. [保存] をクリックします。

i. [マッチングロール] タブを選択します。

j. **[マッチングロールを選択:]** フィールドで、前の手順で作成したロールを選択します。

k. **[選択されたマッチングロールに対して追加するターゲットロールを選択]** フィールドで、当該ネームスペースのグローバルとルーチンに関連付けられた 1 つまたは複数のロールを選択します。これらのグローバルとルーチンは、同じデータベース内にあっても別々のデータベース内にあってもかまいません。お使いのサービスが別の InterSystems IRIS データベースにアクセスする場合は、そのデータベースのロールも選択する必要があります。 複数のロールを選択するには、Ctrl キーを押しながら選択します。

   注釈　グローバル用データベースには、二次データベースおよび対応するロールも存在する場合があります (例 : %DB_GDBSECONDARY)。この二次データベースにはパスワードが保管されます。パススルー・サービスとパススルー・オペレーションのためにこのデータベースにアクセスする必要はありませんが、パスワード・アクセスを使用するカスタム Web サービスを作成する場合は、二次データベースのロールをターゲット・データベースに追加する必要もあります。

l. 選択したロールがハイライト表示された状態で、右矢印をクリックしてこれらのロールを **[選択済み]** テキスト・ボックスに移動します。

m. **[割り当てる]** ボタンをクリックします。

これで、システム構成が完了しました。

Tipヒント ユーザを認証する場合は、以下の操作を行います。

1. EnsLib.SOAP.GenericService を拡張する新規クラスを作成します。
2. このクラスで、パラメータ SECURITYIN を ALLOW または REQUIRE に設定してオーバーライドします。
3. ビジネス・サービスの構成時に、この新規クラスを使用します。
4. サービスを呼び出す際に、WS-Security ユーザ名トークンを使用します。
5. Web アプリケーションの構成で、**[認証なし]** オプションをクリアします。

B

SOAP 受信アダプタの使用法

ここでは、クラス EnsLib.SOAP.InboundAdapter について簡単に説明します。このクラスは EnsLib.SOAP.Service (“プロダクション内での Web サービスの作成” を参照) の代替として使用できます。

相互運用プロダクションで Web サービスを作成する標準的な方法では、EnsLib.SOAP.Service のサブクラスを作成し、Web サーバをプロダクション Web サーバとして設定します。この方法により、市販の Web サーバおよび InterSystems IRIS® SOAP フレームワークによって提供されるすべての SOAP 機能とセキュリティ機能をシステムで利用できるようになります。一方、EnsLib.SOAP.InboundAdapter を使用すると構成が容易になり、軽量化を実現できますが、この方法では、前述した正式な Web サポート機構が使用されません。また、アダプタは、標準の InterSystems IRIS SOAP フレームワークで行われるようには WSDL およびテスト・ページを公開しません。

B.1 メモ

SOAP 受信アダプタ (EnsLib.SOAP.InboundAdapter) は、Web サーバ・ソフトウェアを必要としません。代わりに、InterSystems IRIS スーパー・サーバを使用して TCP リスナ・ジョブを生成します。これによって、フォアグラウンド・ウィンドウでサービスを実行できるので、デバッグに便利です。(これを行うには、サービスをローカルで実行する必要があります。また、PoolSize 設定が 1、JobPerConnection 設定が偽であることを確認します。)TLS もサポートします。

EnsLib.SOAP.InboundAdapter は、指定されたポートで HTTP 入力をリッスンします。アダプタが入力を受信すると、以下の動作が行われます。

1. HTTP `SOAPaction` ヘッダを抽出します。
2. 入力の本文を含むストリーム (%Library.GlobalBinaryStream) を作成します。
3. 指定された `SOAPaction` に対応する Web メソッドを呼び出します。

このアダプタには、以降の SOAP 呼び出しでも永続的な接続を使用できる利点があります。また、JobPerConnection に 0 を指定してこのアダプタを使用すると、以降の複数の SOAP サービス呼び出しにわたる接続でも、インスタンス化で負荷が発生する Xpath パーサなどのリソースを引き続き保持できます。

B.2 開発タスク

InterSystems IRIS SOAP 受信アダプタを使用するには、スタジオで新しいビジネス・サービス・クラスを記述およびコンパイルします。以下は基本要件のリストです。

- このクラスは EnsLib.SOAP.Service を拡張します。このクラスは、Ens.BusinessService と %SOAP.WebService の両方を拡張します。前者を拡張することで、このクラスはビジネス・サービスとなり、後者を拡張することで、このクラスは Web サービスとしても機能できるようになります。
- このクラスは、SERVICENAME およびその他のパラメータの値を提供する必要があります。これについては、“基本要件”を参照してください。
- このクラスは、“InterSystems IRIS で使用する Web メソッドの定義”で説明したように、Web メソッドを定義する必要があります。

オプションとして、アダプタ経由の呼び出しに対するサポートをサービスで無効にするには、以下をクラスに追加します。

```
Parameter ADAPTER="";
```

B.3 構成タスク

管理ポータルを使用して、以下の操作を行います。

1. カスタム・クラスのインスタンスをプロダクションに追加します。

   重要　　構成名が、パッケージを含む完全なクラス名と同じであることを確認します。これは、プロダクション Web サービスを実行するための要件です。

2. ビジネス・サービスを有効化します。
3. PoolSize 設定を 1 に設定し、アダプタが TCP リスナを使用できるようにします。
4. StayConnected 設定を 0 にします。そうしないと、サーバが接続をドロップするのを待機している間に、クライアントがタイムアウト期間を超えてハングアップする場合があります。
5. その他の設定を適切に指定します。“設定の参照先”の“SOAP 受信アダプタに関する設定”を参照してください。
6. プロダクションを実行します。

C
古い Web サービスの相違点

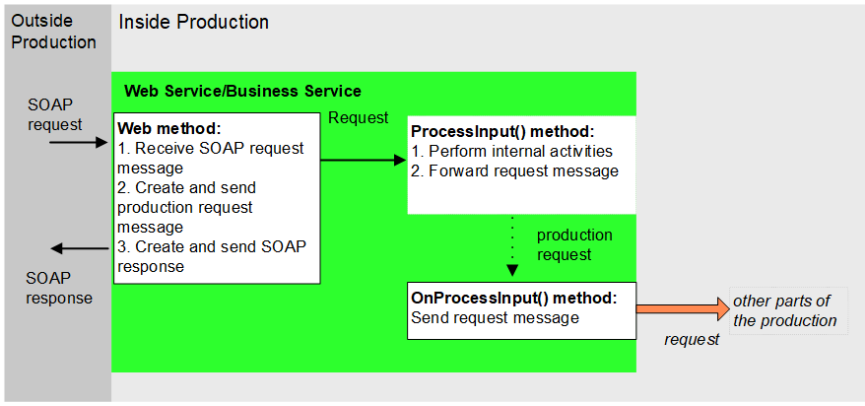
前のリリースでは、InterSystems IRIS® Web メソッドから直接 SendRequestSync()、SendRequestAsync()、または SendDeferredResponse() を呼び出すことはできませんでした。別の方法が必要でした。この付録では、この別の方法を使ったコードを維持する場合の利点について詳細に説明します。

C.1 概要

前のリリースでは、InterSystems IRIS Web メソッドから直接 SendRequestSync()、SendRequestAsync()、または SendDeferredResponse() を呼び出すことはできませんでした。代わりに、このドキュメントで前述した要件のほかに、プロダクション Web サービスに対する 2 つの要件がありました。

- 各 Web メソッドは、必要に応じて ProcessInput() メソッドを呼び出し、適切な要求プロダクション・メッセージを渡して、応答メッセージを受信しなければなりませんでした。
- Web サービス・クラスでは、OnProcessInput() コールバック・メソッドを定義する必要がありました。このメソッドでは、SendRequestSync()、SendRequestAsync()、または SendDeferredResponse() を呼び出します。

以下の図は、このシナリオでの要求メッセージの全体的なフローを示しています。



C.2 OnProcessInput() メソッドの実装

OnProcessInput() メソッドには、以下のシグニチャがあります。

```
Method OnProcessInput(pInput As %RegisteredObject,
                      ByRef pOutput As %RegisteredObject,
                      ByRef pHint As %String) As %Status
```

ここで、

1. pInput は、送信する要求メッセージです。
2. pOutput は、返信として送信される応答メッセージです。
3. pHint は、InterSystems IRIS 要求の処理方法を決定するために使用できるオプション文字列です。詳細は、“pHint 引数の使用”の項を参照してください。

以下に例を示します。

Class Member

```
Method OnProcessInput(pInput As %RegisteredObject, ByRef pOutput As %RegisteredObject) As %Status
{
    set sc= ..SendRequestSync("Lookup",pInput,.pOutput)
    Quit sc
}
```

C.2.1 pHint 引数の使用

Web サービスに複数のメソッドが定義されている場合で、それらをプロダクション内の異なる宛先に送信する必要がある場合、次のように ProcessInput() および OnProcessInput() メソッドのオプションのヒント引数を使用していました。

1. ProcessInput() を呼び出すときに、この呼び出しを行った Web メソッドを示すヒント引数の値を使用していました。以下に例を示します。

```
Method GetCustomerInfo(ID As %Numeric) As ESOAP.SOAPResponse [ WebMethod ]
{
    //create request message with given ID
    set request=##class(ESOAP.CustomerRequest).%New()
    set request.CustomerID=ID

    //send request message
    //ProcessInput() calls OnProcessInput(), which actually
    //sends the message
    set sc=..ProcessInput(request,.response,"GetCustomerInfo")
...

    quit soapresponse
}
```

2. OnProcessInput() 内で、ヒント引数を使用してフローを決定していました。以下に例を示します。

Class Member

```
Method OnProcessInputAlt(pInput As %RegisteredObject,
ByRef pOutput As %RegisteredObject, pHint As %String) As %Status
{
    if pHint="GetCustomerInfo"{
        set sc= ..SendRequestSync("GetCustomerInfoBO",pInput,.pOutput)
        }
    elseif pHint="GetStoreInfo" {
            set sc= ..SendRequestSync("GetStoreInfoBO",pInput,.pOutput)
    }
    Quit sc
}
```

SOAP アダプタ設定

ここでは、SOAP アダプタの参照情報を提供します。

″プロダクションの管理″ の “すべてのプロダクションに含まれる設定” も参照してください。

SOAP 受信アダプタに関する設定

SOAP 受信アダプタ EnsLib.SOAP.InboundAdapter の設定に関する参照情報を提供します。このアダプタを必要としない“プロダクション内での Web サービスの作成”も参照してください。

概要

受信 SOAP アダプタには以下の設定があります。

グループ	設定
基本設定	[ポート]、[呼び出し間隔]
接続設定	[標準リクエスト有効]、[アダプタ URL]、[接続毎のジョブ]、[許可IPアドレス]、[OS接続受け付けキューサイズ]、[接続を維持]、[読み込みタイムアウト]、[SSL構成]、[ローカル・インターフェース]
遅延応答サポート	[遅延同期リクエストをサポート]、[クライアント応答待ちタイムアウトを上書き]、[ゲートウェイタイムアウト]
追加設定	SuperSession ID の生成

残りの設定はすべてのビジネス・サービスに共通しています。詳細は、“すべてのビジネス・サービスに含まれる設定”を参照してください。

アダプタ URL

サービスが要求を受け取る特定の URL。カスタム・ローカル・ポート上の SOAP 受信アダプタによって呼び出される SOAP サービスでは、この設定によって、標準の csp/namespace/classname スタイルの URL の代わりにカスタム URL を使用できるようになります。

Allowed IP Addresses

接続を受け入れるリモート IP アドレスのカンマ区切りのリストを指定します。アダプタは、ドット付き 10 進数形式による IP アドレスを受け入れます。オプションで :port の指定がサポートされています。192.168.1.22 または 192.168.1.22:3298 のいずれかのアドレス形式で指定できます。

注釈　IP アドレス・フィルタリングは、一般アクセスできるシステムではなくプライベート・ネットワーク上のアクセスを制御するための手段です。IP アドレス・フィルタリングを唯一のセキュリティ・メカニズムとして利用することは推奨されません。攻撃者は IP アドレスをスプーフィング (偽装) できるからです。

ポート番号を指定すると、その他のポートからの接続は拒否されます。

この文字列の先頭に感嘆符 (!) が付加されている場合、受信アダプタは、受信接続要求を待つことなく接続を開始します。受信アダプタは指定されたアドレスへの接続を開始し、次にメッセージを待機します。この場合、指定されるアドレスは 1 つだけで、ポートが指定された場合は Port 設定の値より優先されます。それ以外の場合は Port 設定が使用されます。

呼び出し間隔

アダプタが構成されたソースからの受信データをリッスンする秒数を指定します。この秒数を超えると、プロダクション・フレームワークからのシャットダウン信号を確認します。

アダプタは、入力を検出すると、データを取得してビジネス・サービスに渡します。ビジネス・サービスはデータを処理し、アダプタは直ちに新規入力の待機を開始します。プロダクションが実行中であり、ビジネス・サービスが有効化され、アクティブになるようにスケジュールされている場合、このサイクルは常に継続されます。

既定値は 5 秒です。最小値は 0.1 秒です。

標準リクエスト有効

この設定が真の場合、アダプタは、通常の方法 (TCP 接続のバイパス) で SOAP 要求も受信できます。デフォルトは偽です。

ゲートウェイタイムアウト

外部 TCP ソケット・システムに想定するタイムアウトを秒数で指定します。

IRIS Web アプリケーションを通じて受信要求が得られる場合、既定値は“IRIS”です。この場合は、要求ヘッダに記述された IRIS の Web ゲートウェイ・タイムアウト値をアダプタで使用できます。それ以外の場合、既定値は 60 秒になります。

SuperSession ID の生成

このプロパティは SuperSessionID をメッセージに含めるかどうかを制御します。SuperSessionID を使用すると、複数のネームスペースにまたがるメッセージを識別できます。このプロパティを設定すると、ビジネス・サービスはまず受信メッセージの HTTP ヘッダをチェックして SuperSession ID を探します。SuperSessionID の値がある場合にはこの値を使用し、ない場合には新しい SuperSession の値を生成します。プロダクション・メッセージに SuperSesssion 値を設定します。また、呼出元に送信する HTTP 応答で値を返すこともできます。

接続ごとのジョブ

この設定が真の場合、アダプタは受信 TCP 接続ごとに処理を行う新しいジョブを生成して、複数の接続の同時処理を可能にします。偽の場合、各接続に対する新しいジョブは生成されません。デフォルトは真です。

ローカル・インタフェース

接続に必要なネットワーク・インタフェースを指定します。リストから値を選択するか、値を入力してください。空の値は、任意のインタフェースが使用できることを意味します。

OS接続受け付けキューサイズ

オペレーティング・システムで開いておく必要がある受信接続の数を指定します。一度に 1 つの接続のみが予想される場合は 0 に設定します。多数のクライアントが次々と接続する場合には大きな数値を設定します。

クライアント応答待ちタイムアウトを上書き

クライアントに想定する応答待ち時間を秒数で指定します。

既定値は 0 で、この場合はクライアントで設定されたタイムアウトが使用されます。

クライアントが IRIS 相互運用プロダクション SOAP オペレーションである場合は、HTTP ヘッダにクライアントの応答時間が記述されます。

ポート

アダプタが SOAP 要求をリッスンしているローカル・マシン上の TCP ポートを指定します。オペレーティング・システムで一時的な送信接続用に使用されるポート範囲内のポート番号を指定することは避けてください。

読込タイムアウト

リモート TCP ポートからの最初のデータを受信した後、連続した受信 TCP 読み取りオペレーションを待機する秒数を指定します。既定値は 5 秒です。指定できる値の範囲は 0 ～ 600 秒 (最大 10 分) です。

SSL構成

この接続の認証に使用する既存の TLS 構成の名前。これは、サーバ構成である必要があります。

TLS 構成を作成して管理するには、管理ポータルを使用します。インターシステムズの "TLS ガイド" を参照してください。[SSL/TLS構成を編集] ページの最初のフィールドは [構成名] です。この文字列を [SSL構成] の値として使用します。

接続を維持

要求の間に、接続が開いた状態を維持するかどうかを指定します。

- この設定が 0 の場合、アダプタは各イベント後に直ちに切断します。
- この設定が -1 の場合、アダプタは起動時に自動接続し、接続したままになります。
- この設定には正の値も使用できますが、その場合の値はアイドル時間 (秒) になるので、ポーリングによって機能するこのアダプタでは役に立ちません。アイドル時間がポーリング間隔 CallInterval よりも長い場合、アダプタは常に接続された状態のままです。アイドル時間がポーリング間隔よりも短い場合は、ポーリング間隔ごとにアダプタの切断と再接続が発生します。

遅延同期リクエストをサポート

遅延同期要求を有効にするかどうかを指定します。既定では無効です。

クライアントが独自の `ClientRequestKey` と `ClientRetryRequestKey` を指定している場合、そのクライアントが最初の応答待ちでタイムアウトして要求を再試行すると、このサービスで実行される SendRequestSync は、同じ要求が複数回実行されないようにしようとします。

SOAP 送信アダプタに関する設定

SOAP 送信アダプタ EnsLib.SOAP.OutboundAdapter の設定に関する参照情報を提供します。

概要

送信 SOAP アダプタには以下の設定があります。

グループ	設定
基本設定	[ウェブサービスURL]、[ウェブサービスクライアントクラス]、[SOAP認証情報]、[認証情報]
接続設定	[SSL構成]、[SSL チェック・サーバ ID]
プロキシ設定	[プロキシ・サーバ]、[プロキシ・ポート]、[HTTPSプロキシ]、[プロキシHTTPトンネル]、[プロキシHTTP SSL接続]
遅延応答サポート	[HTTP ヘッダにリクエストキーを含める]
追加設定	[応答タイムアウト]、[HTTP バージョン]、[接続タイムアウト]、SendSuperSession

残りの設定はすべてのビジネス・オペレーションに共通しています。詳細は、“すべてのビジネス・オペレーションに含まれる設定”を参照してください。

ConnectTimeout

サーバへの接続が開かれるまで待つ秒数を指定します。デフォルト値は 5 です。

この時間内に接続が開かれない場合、アダプタは、Failure Timeout を Retry Interval で割って得られる回数を上限として再試行を繰り返します。

Credentials

HTTP ヘッダで使用するユーザ名とパスワードを含むプロダクション認証情報の ID を指定します。プロダクション認証情報の作成方法は、"プロダクションの構成" を参照してください。

HTTP バージョン

アダプタがサーバに送信する HTTP 要求の中で報告する必要がある HTTP バージョン。

HTTP ヘッダにリクエストキーを含める

HTTP ヘッダに要求キーを記述するかどうかを指定します。既定では無効です。

チェックを付けると、このキーは `VND.InterSystems.IRIS.RequestKey` または `VND.InterSystems.IRIS.RetryRequestKey` で、その値はシステムの GUID、現在のネームスペース、および現在の要求ヘッダ ID になります。

HTTP ヘッダには `VND.InterSystems.IRIS.ResponseTimeout` もあり、その値は応答タイムアウトの値です。

この設定は、遅延応答サポートを有効にした IRIS 相互運用プロダクション SOAP サービスと共に使用されることを意図しています。

再試行の際に要求のペイロードがホスト・クラスによって変更され、元の要求の場合とは異なる応答が想定される場合、この設定は有効にしないようにします。

ProxyHTTPS

プロキシ (ある場合) が実際の HTTP/HTTPS サーバとの通信に HTTPS を使用するかどうかを指定します。

プロキシHTTPトンネル

アダプタが HTTP CONNECT コマンドを使用して、プロキシ経由でターゲットの HTTP サーバへのトンネルを確立するかどうかを指定します。真の場合、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、[プロキシ・サーバ] および [プロキシ・ポート] の各プロパティから取得されます。HTTPS プロキシの SSL Connect が真の場合、トンネルが確立されると、InterSystems IRIS® は TLS 接続をネゴシエートします。デフォルト値は偽です。

ProxyPort

プロキシ・サーバを使用する場合、HTTP 要求を送信するプロキシ・サーバ・ポートを指定します。デフォルト値は 80 です。

ProxyServer

HTTP 要求を送信する際に使用するプロキシ・サーバ (ある場合) を指定します。

プロキシHTTP SSL接続

アダプタがプロキシへのプロキシ TLS 接続を使用するかどうかを指定します。最終的なエンドポイントまで TLS を使用するかどうかは、Web サービスの場所を示す URL のプロトコル部分によって決まります。

ResponseTimeout

リモート Web サーバからの応答取得に対するタイムアウトを指定します (サーバへの接続を開く場合のタイムアウトは ConnectTimeout で設定します)。デフォルト値は 30 です。

SendSuperSession

SendSuperSession はブーリアン設定であり、送信アダプタが HTTP ヘッダ内に SuperSession ヘッダを作成するかどうか、および識別子をそのヘッダに割り当てるかどうかを制御します。メッセージを検索するときに、SuperSession 値を使用して、あるプロダクション内のメッセージを、別のプロダクション内の関連メッセージと突き合わせることができます。プロダクション内では、ビジネス・サービス、プロセス、およびオペレーションの間でメッセージは SessionId を用いて転送されるので、追跡は簡単です。しかし、メッセージが SOAP メッセージによってビジネス・オペレーションからいったん離れ、別のプロダクションに入ると、そのメッセージを受け取ったプロダクションは新しい SessionId を割り当てます。

SendSuperSession を選択すると、SOAP 送信アダプタは以下を実行します。

1. メッセージの Ens.MessageHeaderBase.SuperSession プロパティに空の値があるかどうかを確認します。空の値が含まれている場合、アダプタが新しい値を生成して SuperSession プロパティにそれを格納します。
2. SuperSession プロパティの値を、送信メッセージのプライベート InterSystems.Ensemble.SuperSession HTTP ヘッダに格納します。

SOAP 受信アダプタは、メッセージを受信すると、受信 HTTP メッセージ・ヘッダ内の SuperSession 値を確認します。値が存在する場合、Ens.MessageHeaderBase.SuperSession プロパティを設定します。このプロパティは、メッセージが別のプロダクション・コンポーネントに渡されるときも保持されます。

注釈　SuperSession を使用してプロダクション間でのメッセージの追跡を自動化するツールはありません。

SOAPCredentials

SOAP 要求の WS-Security ヘッダで使用するユーザ名とパスワードを含むプロダクション認証情報の ID を指定します。WS-Security のサポートに関する詳細は、ドキュメント・セットの "Web サービスの保護" を参照してください。

SSLCheckServerIdentity

TLS 接続を行う際に、証明書内のサーバ ID が接続先システムの名前と一致することがアダプタによってチェックされることを指定します。デフォルトでは、このチェックが行われるように指定されます。TLS 証明書で指定された名前が DNS 名と一致しないテスト・システムと開発システムについては、この設定をオフにします。

SSLConfig

この接続の認証に使用する既存の TLS 構成の名前。Web クライアントから通信が開始されるため、クライアント TLS 構成を選択します。

TLS 構成を作成して管理するには、管理ポータルを使用します。インターシステムズの "TLS ガイド" を参照してください。[SSL/TLS構成を編集] ページの最初のフィールドは [構成名] です。この文字列は [SSL構成] 設定の値として使用します。

注釈　また、Web サービスが `https://` を使用する URL にあることを確認する必要があります。Web サービスの場所は WebServiceURL 設定によって決まります。この設定が指定されていない場合、InterSystems IRIS Web クライアントは、プロキシ・クライアント・クラスの LOCATION パラメータによって指定される URL に Web サービスがあると仮定します。

WebServiceClientClass

プロキシ・クライアント・クラスの完全名 (パッケージを含む) を指定します。具体的には、実際に Web サービスと SOAP メッセージの送受信を行うクラスです。

WebServiceURL

Web サービスがある URL を指定します。この設定が指定されていない場合、アダプタは、プロキシ・クライアント・クラスで宣言されたデフォルトの場所 (LOCATION パラメータ) を使用します。詳細は WebServiceClientClass 設定を参照してください。この URL が `https://` を使用している場合にのみ、TLS が機能します。