プロダクション内での HTTP アダプタの使用法

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 HTTP アダプタの使用法
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

HTTP アダプタについて

HTTP アダプタ (EnsLib.HTTP.InboundAdapter および EnsLib.HTTP.OutboundAdapter) を使用すると、プロダクションで HTTP 要求および応答を送受信できるようになります。ここでは、これらのアダプタについて簡単に説明します。

注釈　Web サーバと InterSystems IRIS との間の通信の設定には Web ゲートウェイを使用することを強くお勧めしますが、必要に応じて、受信アダプタを使用することもできます。

Tipヒント　InterSystems IRIS® データ・プラットフォームでは、これらのアダプタ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。"接続オプション" を参照してください。

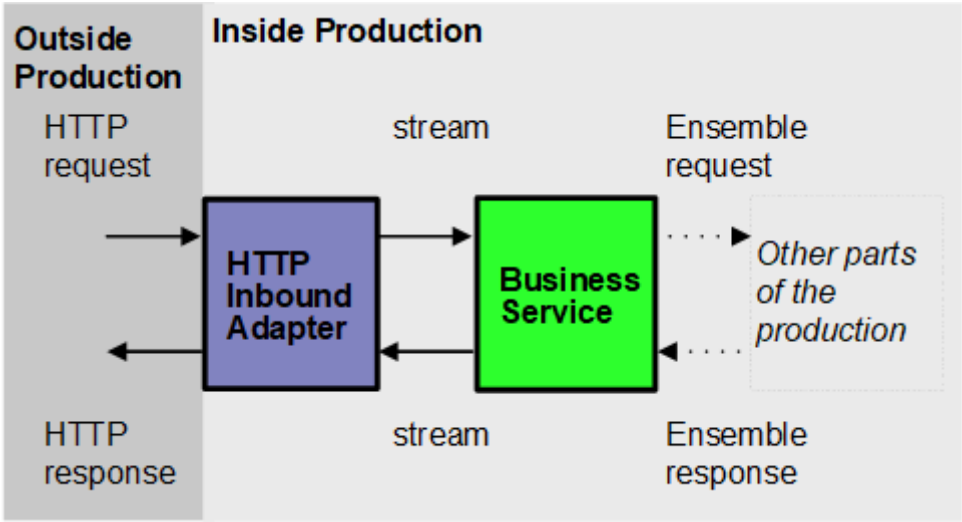
1.1 HTTP 受信アダプタとヘルパ・クラス

EnsLib.HTTP.InboundAdapter は、カスタム・ポート・リスニング、XML リスニング、または生の HTML 処理に使用する HTTP リスナです。受信アダプタは、(標準の Web サーバを使用して HTTP 要求を処理する) CSP ページを使用せずに、プライベート・ポートでリッスンします。Web ゲートウェイの詳細は、"Web ゲートウェイのインストール" を参照してください。

EnsLib.HTTP.InboundAdapter クラスには、以下のような項目の指定に使用する実行時設定が用意されています。

- アダプタが入力をリッスンするローカル・ポート
- アダプタが入力を受け入れる IP アドレスのリスト (使用可能なソースを制限したい場合)
- 受信要求で指定された文字セットを使用するかどうかを指定する設定。また、受信要求で指定された文字セットを使用しない場合は、使用する別の文字セット

受信 HTTP アダプタは指定されたポートでリッスンし、入力を読み取り、関連するビジネス・サービスにストリーム (使用する文字セットに応じて、バイナリまたは文字のいずれか) としてその入力を送信します。ユーザが作成および構成するビジネス・サービスは、このストリームを使用してプロダクションの他の部分と通信します。



HTTP 受信アダプタを使用する場合、%Library.GlobalCharacterStream および %Library.GlobalBinaryStream の 2 つのヘルパ・クラスを使用します。受信アダプタは、関連するビジネス・サービスにストリームを送信します。具体的にはこれは、使用する文字セットに応じて、%Library.GlobalCharacterStream または %Library.GlobalBinaryStream のインスタンスになります。通常、これらのクラスは、ストリームのコンテンツを読み取る、ストリーム長を取得する、単一の行を読み取る、巻き戻す、データを追加する、といった操作に使用できるメソッドを提供します。これらのクラスの詳細は、"ストリームを使用した作業" を参照してください。

1.2 HTTP 送信アダプタとヘルパ・クラス

EnsLib.HTTP.OutboundAdapter は、HTTP 要求をプロダクションの外部に送信し HTTP 応答を受信するアダプタです。このアダプタは、以下のような項目を制御する設定を提供します。

- アダプタが HTTP 要求を送信するサーバおよびポート
- 指定されたサーバおよびポートで、要求するリソースの URL パス
- サーバへの接続に使用するオプションの TLS 構成
- アダプタが要求をルーティングするプロキシ・サーバを指定するためのオプション情報

このアダプタは、HTTP POST、GET、および PUT アクションを送信するメソッドを提供します。

- 主要なメソッドは PostFormData() および GetFormData() です。各メソッドで、応答オブジェクトの出力引数、フォーム変数名のカンマ区切りのリスト、およびフォーム変数引数の変数数値 (1 つの変数数値がカンマ区切りリスト内の各名前に対応) を使用できます。1 つのフォーム変数に複数の値を設定する場合は、リスト内で同じ名前を複数回使用できます。もちろん、フラットなスカラ・コンテンツ (通常の Web ページなど) を要求するために、フォーム変数なしでこれらのメソッドを使用することもできます。
- フォーム変数の複雑なセットが使用されている場合は、PostFormDataArray() および GetFormDataArray() メソッドを使用します。これらのメソッドでは、変数引数リストの代わりに、多次元配列が使用できます。多次元配列は、指定されたフォーム変数の複数の値を、名前リスト内の複数のエントリとしてではなく、配列のサブノードとして提供できるため、情報の組織化に役立ちます。また、位置ではなくフォーム変数名によって配列にインデックスを割り当てることができます。
- その他の特別な HTTP 要求 (PUT など) を使用する必要がある場合や、フォーム変数や Cookie 以外の HTTP 要求の面でカスタマイズをする必要がある場合には、低レベルのワーカ・メソッドである SendFormDataArray() を使用できます。

このアダプタは、Cookie を管理するためのプロパティおよびメソッドも提供します。

HTTP 送信アダプタを使用する場合、以下の 2 つのヘルパ・クラスを使用します。

- HTTP 送信アダプタは、%Net.HttpRequest を使用して、送信する HTTP 要求をカプセル化します。

  PostFormData()、GetFormData()、PostFormDataArray()、または GetFormDataArray() を使用する場合、要求はアダプタによって自動的に作成されるので、直接アクセスすることはできません。

  ただし、SendFormDataArray() を使用する場合は、%Net.HttpRequest クラスのインスタンスを作成して、そのプロパティを設定し、それを使用してそのメソッドで送信する HTTP 要求を初期化できます。アダプタを介して設定できない (プロキシ認証などの) HTTP 要求のプロパティを設定する必要がある場合は、この方法を使用します。

- すべての場合で、HTTP 応答は %Net.HttpResponse のインスタンスにカプセル化されます。%Net.HttpResponse クラスは HTTP ヘッダにアクセスするためのメソッドを提供すると共に、応答 (ストリーム・オブジェクト) の本文、理由コード、サーバの HTTP のバージョンなどを含むプロパティも提供します。

2

HTTP 受信アダプタの使用法

ここでは、HTTP 受信アダプタ (EnsLib.HTTP.InboundAdapter) の既定の動作、およびプロダクションでのこのアダプタの使用法について説明します。

Tipヒント InterSystems IRIS® データ・プラットフォームでは、このアダプタ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。"接続オプション" を参照してください。

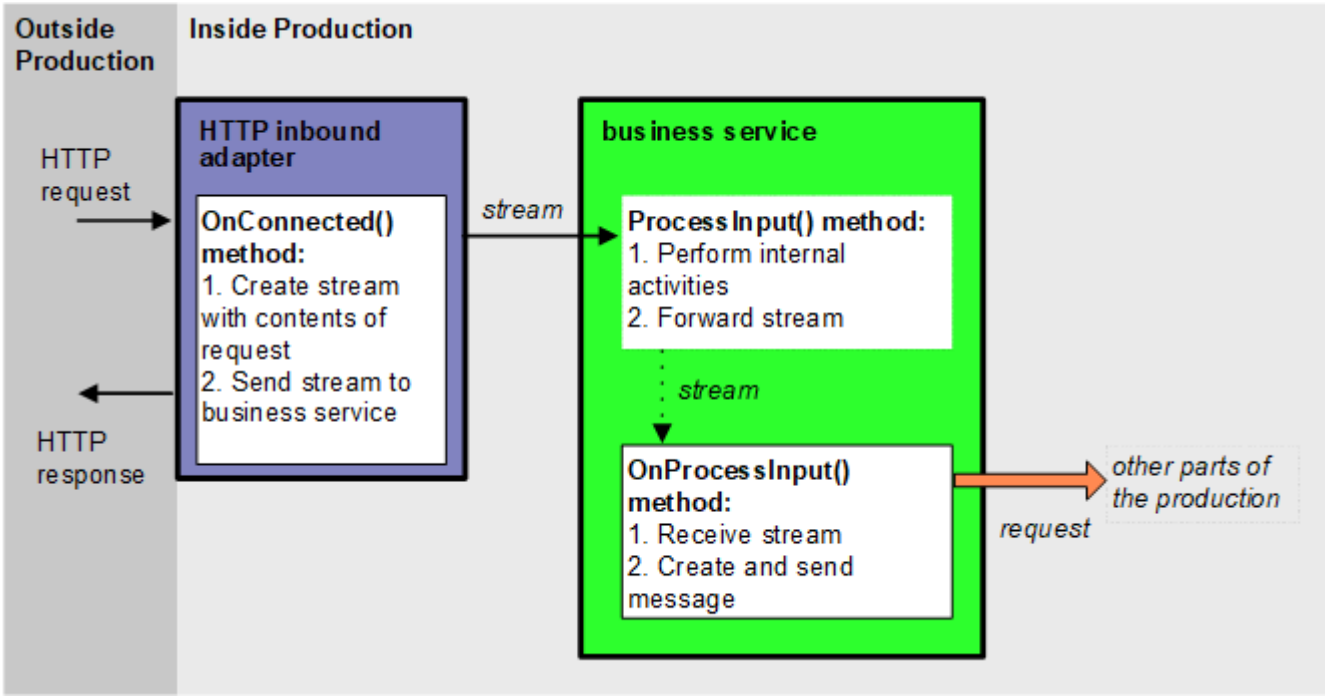
2.1 全般的な動作

EnsLib.HTTP.InboundAdapter は、カスタム・ポート・リスニング、XML リスニング、または生の HTML 処理に使用する HTTP リスナです。(構成した Web サーバを使用して HTTP 要求を処理する) CSP ページを使用せずに、プライベート・ポートでリッスンしたい場合に、このアダプタを使用します。

まず、このクラスには、以下のような項目の指定に使用する実行時設定が用意されています。

- アダプタが入力をリッスンするローカル・ポート
- アダプタが入力を受け入れる IP アドレスのリスト (使用可能なソースを制限したい場合)
- 受信要求で指定された文字セットを使用するかどうかを指定する設定。また、受信要求で指定された文字セットを使用しない場合は、使用する別の文字セット

受信 HTTP アダプタはローカル・マシン上のポートをリッスンし、入力を読み取り、関連するビジネス・サービスにストリームとしてその入力を送信します。ユーザが作成および構成するビジネス・サービスは、このストリームを使用してプロダクションの他の部分と通信します。以下の図は、全体的なフローを示しています。



詳細は、以下のとおりです。

1. アダプタは HTTP メッセージを受信し、TCP 接続を開きます。(HTTP は TCP 接続を介して送信されるヘッダと本文データのフォーマットです。)
2. アダプタが接続されると、その OnConnected() メソッドが実行され、最初に、使用する文字セットが決定されます。デフォルトでは、受信 HTTP 要求で指定された文字セットが使用されます。詳細は、"使用する文字セットの指定" を参照してください。
3. アダプタは、文字セットに対して適切な変換テーブルを選択します。
4. アダプタは、入力の本文を読み取り、変換し、新しいストリーム・オブジェクト内に配置します。
   - アダプタが非バイナリの文字セットを使用している場合、ストリームのタイプは %GlobalCharacterStream です。
   - アダプタがバイナリの文字セットを使用している場合、ストリームのタイプは %GlobalBinaryStream です。

   %GlobalCharacterStream および %GlobalBinaryStream は非推奨になっていますが、このような使用法は引き続きサポートされています。このユース・ケースで別のストリーム・クラスに置き換えることはお勧めできません。

   また、アダプタは各 HTTP ヘッダを抽出し、ストリームの Attributes プロパティにそのヘッダを追加します。このプロパティは多次元配列です。これについては後述します。

   また、URL にフォーム・パラメータが含まれている場合、これらのパラメータは次のように渡されます。
   - HTTP 要求が GET 要求の場合、"Params" 添え字にある Attributes 配列に追加されます。
   - HTTP 要求が POST 要求の場合、要求本文にフォーム変数が書き込まれます。
5. 次に、アダプタは、ビジネス・サービス・クラスの内部 ProcessInput() メソッドを呼び出し、ストリームを入力引数として渡します。
6. ビジネス・サービス・クラスの内部 ProcessInput() メソッドが実行されます。このメソッドは、すべてのビジネス・サービスが必要とする内部情報の保持など、基本的なプロダクション・タスクを実行します。ビジネス・サービス・クラスが継承するこのメソッドは、カスタマイズや上書きを行いません。
7. 次に、ProcessInput() メソッドがカスタムの OnProcessInput() メソッドを呼び出し、ストリーム・オブジェクトを入力として渡します。このメソッドの要件については、この後の "OnProcessInput() メソッドの実装" で説明します。

8. ProcessInput() または OnProcessInput() からエラーが返された場合、プロダクションはビジネス・サービスの OnErrorStream() メソッドを呼び出します。

応答メッセージは、同じパスを逆向きにたどります。

2.2 HTTP 受信アダプタを使用するビジネス・サービスの作成

このアダプタをプロダクションで使用するには、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。"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。

- ビジネス・サービス・クラスは Ens.BusinessService 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.HTTP.InboundAdapter である必要があります。
- HTTP POST 要求からのフォーム変数を解析する必要がある場合は、以下のように OnInit() コールバック・メソッドを実装します。

  Class Member

  ```
  Method OnInit() As %Status
  {
      Set ..Adapter.ParseBodyFormVars=1
      Quit 1
  }
  ```

  この手順は、GET 要求からのフォーム変数を解析する場合には必要ありません。この後で説明するように、これらは自動的に解析され、Attributes プロパティとして生成されます。
- このクラスは OnProcessInput() メソッドを実装します。これについては "OnProcessInput() メソッドの実装" で説明します。
- クラスには OnErrorStream() メソッドを実装できます。"OnErrorStream() メソッドの実装" を参照してください。
- その他のオプションと一般情報は、"ビジネス・サービス・クラスの定義" を参照してください。

以下の例は、ビジネス・サービス・クラスの全体的な構造を示しています。OnProcessInput() メソッドの詳細は、アダプタが使用している文字セットによって異なります。アダプタが非バイナリの文字セットを使用している場合、全般的な構造は以下のようになります。

Class Definition

```
Class EHTTP.NewService1 Extends Ens.BusinessService
{
Parameter ADAPTER = "EnsLib.HTTP.InboundAdapter";

Method OnProcessInput(pInput As %GlobalCharacterStream, pOutput As %RegisteredObject) As %Status
{
   set tsc=$$$OK
   //your code here
   Quit tsc
}
}
```

または、アダプタがバイナリの文字セットを使用している場合、OnProcessInput() メソッドは以下のようになります。

Class Member

```
Method OnProcessInput(pInput As %GlobalBinaryStream, pOutput As %RegisteredObject) As %Status
{
   set tsc=$$$OK
   //your code here
   Quit tsc
}
```

2.3 OnProcessInput() メソッドの実装

カスタム・ビジネス・サービス・クラスにおいて、OnProcessInput() メソッドのシグニチャは、アダプタが使用している文字セットによって異なります。

- アダプタが非バイナリの文字セットを使用している場合、シグニチャは以下のようになります。

  ```
  Method OnProcessInput(pInput As %GlobalCharacterStream, pOutput As %RegisteredObject) As %Status
  ```

- アダプタがバイナリの文字セットを使用している場合、シグニチャは以下のようになります。

  ```
  Method OnProcessInput(pInput As %GlobalBinaryStream, pOutput As %RegisteredObject) As %Status
  ```

ここで、pInput は、アダプタがこのビジネス・サービスに送信する入力です。また、pOutput は、メソッド・シグニチャに必要な汎用出力引数です。

%GlobalCharacterStream および %GlobalBinaryStream は非推奨になっていますが、このような使用法は引き続きサポートされています。このユース・ケースで別のストリーム・クラスに置き換えることはお勧めできません。

OnProcessInput() メソッドは、以下の一部またはすべてを実行する必要があります。

1. 入力オブジェクトを調べて、そこから必要なデータを抽出します。"HTTP 要求の使用法" を参照してください。
2. ビジネス・サービスから送信されることになる要求メッセージのインスタンスを作成します。

   メッセージ・クラスの作成方法は、"メッセージの定義" を参照してください。
3. 要求メッセージに対し、入力ストリームから取得した値を使用して適切にプロパティを設定します。

   Attributes プロパティに Content-Type 属性が指定されている場合、デフォルトでは、その属性が要求メッセージの Content-Type として使用されます。Attributes プロパティに Content-Type 属性が指定されていない場合、デフォルトでは、要求メッセージの Content-Type は "text/html" に設定されます。これらのデフォルト値に該当しない場合は、この属性を必ず設定します。例えば、次のコードでは、入力ストリームの Content-Type 属性の値をチェックし、この値がなければ "text/xml" を使用します。

   ObjectScript

   ```
   Set outputContentType=$GET(pInput.Attributes("Content-Type"),"text/xml")
   ```

4. 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、"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。
5. 必要に応じて、以前のアクションのステータスを確認し、そのステータスに基づいて対応します。
6. 必要に応じて、ビジネス・サービスが受信した応答メッセージを調査し、それに基づいて対応します。
7. 必ず出力引数 (pOutput) を設定します。この手順は必須です。

8. 適切なステータスを返します。この手順は必須です。

簡単な例を以下に示します。

Class Member

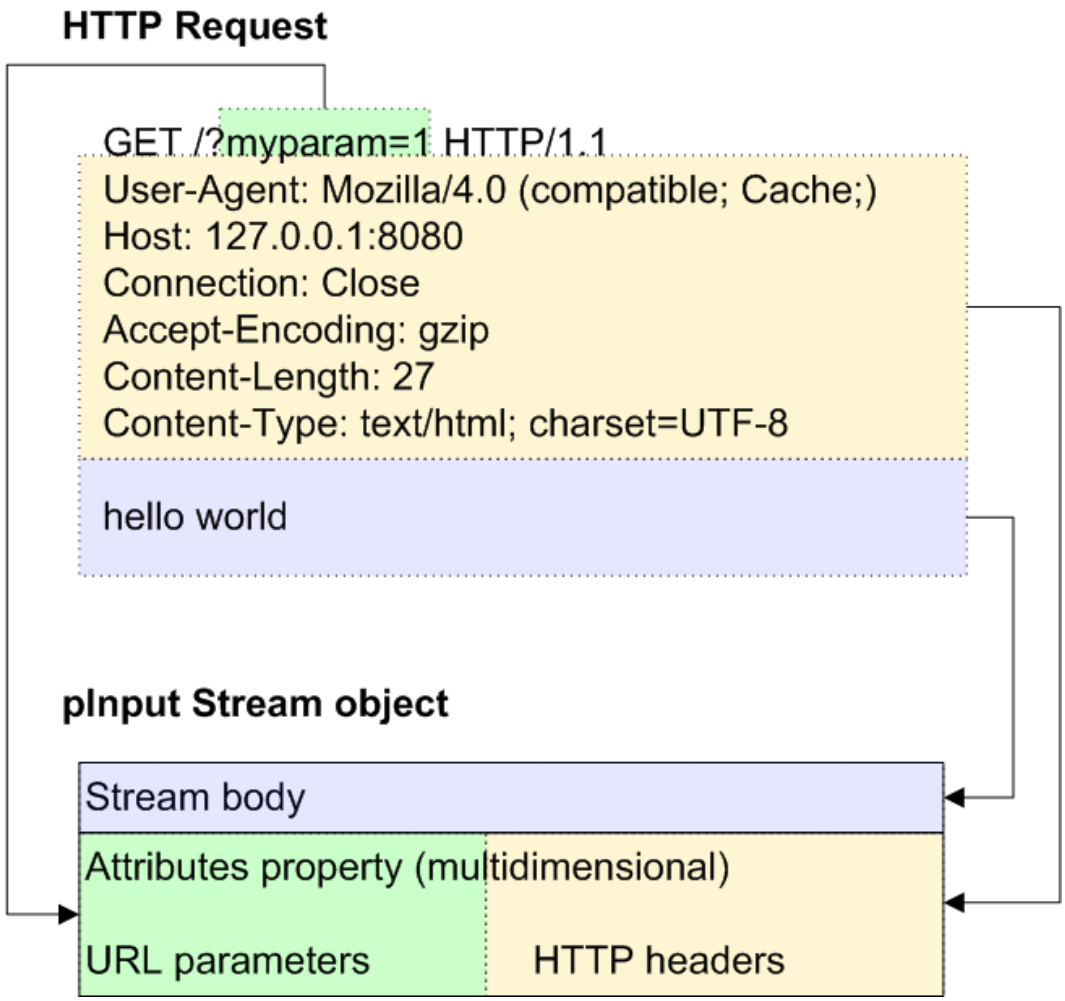
```
Method OnProcessInput(pInput As %GlobalCharacterStream, Output pOutput As %RegisteredObject) As %Status
{
    //get contents of inbound stream
    //in this case, the stream contains a single value: a patient ID
    Set id=pInput.Read(,.tSC)

    //make sure Read went OK
    If $$$ISERR(tSC) do $System.Status.DisplayError(tSC)

    //create request object to send
   Set tRequest=##class(EHTTP.Request.Patient).%New()
    Set tRequest.patientID=id

   //send to lookup process
   Set tSC=..SendRequestSync("EHTTP.LookupProcess",tRequest,.tResponse)

    //define output for OnProcessInput
    Set pOutput=tResponse

    Quit tSC
}
```

2.4 OnErrorStream() メソッドの実装

ビジネス・サービスの ProcessInput() または OnProcessInput() メソッドからエラーが返された場合、プロダクションはビジネス・サービスの OnErrorStream() メソッドを呼び出します。このメソッドは、必要なエラー処理を含めて実装できます。このメソッドは、ステータス・コードを入力として受け入れ、必要な出力ストリームを返す必要があります。

2.5 HTTP 要求の使用法

OnProcessInput() の内部では、HTTP 要求を pInput として利用できます。これは、実装に応じて %GlobalCharacterStream または %GlobalBinaryStream のインスタンスになります。次の図は、このインスタンスで HTTP 要求がどのように表されるかを示しています。



要求の本文は、pInput ストリームに書き込まれます。ストリームの操作の詳細は、"ストリームを使用した作業" を参照してください。

次の項で説明するように、pInput ストリームの Attributes プロパティには、追加のデータが保持されています。これには、HTTP ヘッダが含まれます。要求が GET 要求だった場合は、フォーム変数 (URL パラメータ) もこのプロパティに含まれています。

要求が POST 要求であった場合、フォーム変数は本文にあります。

2.5.1 属性の配列について

pInput ストリームの Attributes プロパティは、以下のデータを保持する多次元配列です。

ノード	コンテンツ
Attributes(http_header_name) (http_header_name は "content-length" のような小文字のヘッダ名)	指定された HTTP ヘッダの値
Attributes("Params",form_variable_name,n)	指定された URL フォーム変数の n 番目のインスタンスの値 (HTTP 要求が GET 要求だった場合)
Attributes("URL")	HTTP 要求の完全な URL

したがって、ヘッダ値を取得するには、次のように指定します。

ObjectScript

```
 set contentlength = pInput.Attributes("content-length")
```

または、変数から URL を取得するには (GET 要求の場合)、次のように指定します。

ObjectScript

```
 set pResponse.MessageRequestTimeStamp = pInput.Attributes("Params","REQUESTTIMESTAMP",1)
```

2.5.2 POST 要求からのフォーム変数の取得

HTTP 要求が POST 要求であった場合、フォーム変数は pInput ストリームにあります。フォーム変数は、& を区切り文字として連結したキーと値のペアです。以下に例を挙げます。

```
Q1=Answer1&Q2=Answer2&Q3=Answer3
```

ObjectScript 文字列関数を使用してフォーム変数を取得します。この方法の例を以下に示します。

Class Member

```
Method OnProcessInput(pInput As %GlobalCharacterStream, pOutput As %RegisteredObject) As %Status
{
    Set tData=pInput.Read(,.tSC)

    If $$$ISERR(tSC) {
      Do $System.Status.DisplayError(tSC)
    }

    Set tRequest=##class(EHTTP.Request.Patient).%New()

    //use a delimiter to separate the form variables
    Set list=$LISTFROMSTRING(tData,"&")

    //examine each element and extract the relevant data
    Set ptr=0
    While $LISTNEXT(list,ptr,key) {
      If $PIECE(key,"=") = "Q1" {
        Set tRequest.Q1 = $PIECE(key,"=",2)
      }
      Elseif $PIECE(key,"=") = "Q2" {
        SET tRequest.Q2 = $PIECE(key,"=",2)
      }
      Elseif $PIECE(key,"=") = "Q3" {
        Set tRequest.Q3 = $PIECE(key,"=",2)
      }
    }

    Set tSC=..SendRequestSync("EHTTP.LookupProcess",tRequest,.tResponse)

    Set pOutput=tResponse

    Quit tSC
}
```

OnInit()を使用して POST 要求のフォーム変数を解析することもできます。

2.6 ビジネス・サービスの追加と構成

ビジネス・サービス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・サービス・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. 入力を受信できるようにアダプタを構成します。具体的には、以下を行います。
   - アダプタがリッスンするポートを指定します。そのためには、[ポート] 設定を指定します。
   - アダプタが通信するソースを制限したい場合は、アダプタが入力を受け入れる IP アドレスを指定します (オプション)。
   - 入力データの文字セットをオプションで指定します。
3. ビジネス・サービスを有効化します。
4. プロダクションを実行します。

2.6.1 HTTP 要求のソースの指定

次の 2 つの方法のいずれかで HTTP 要求のソースを認識するように受信 HTTP アダプタを構成できます。

- 任意のサーバからの HTTP 要求を許可できます。これがデフォルトです。
- 指定のサーバ (必要に応じてポートも指定できます) のリストにあるサーバからの HTTP 要求を許可できます。

  そのためには、[許可IPアドレス] 設定を指定します。"HTTP アダプタ設定" を参照してください。

2.6.2 使用する文字セットの指定

EnsLib.HTTP.InboundAdapter は、入力を受信すると、その入力内の文字を変換テーブルに従って変換します。使用する変換テーブルを決定するために、アダプタはまず入力に使用されている文字セットを判断します。

通常、入力の HTTP `Content-Type` ヘッダに、要求が使用している文字セットが示されています。デフォルトでは、アダプタはこの文字セットを使用します。

ただし、以下の実行時設定を使用して、アダプタが使用する文字セットを制御できます。

- 文字セット強制
- 文字セット

"HTTP アダプタ設定" を参照してください。

文字セットおよび変換テーブルの詳細は、"変換テーブル" を参照してください。

3

HTTP 送信アダプタの使用法

ここでは、HTTP 送信アダプタ (EnsLib.HTTP.OutboundAdapter) の全般的な動作、およびプロダクションでのこのアダプタの使用法について説明します。

Tipヒント InterSystems IRIS® データ・プラットフォームでは、このアダプタ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。"接続オプション" を参照してください。

3.1 全般的な動作

プロダクション内で、送信アダプタは、ユーザが作成および構成するビジネス・オペレーションに関連付けられます。このビジネス・オペレーションはプロダクション内からメッセージを受信し、メッセージ・タイプを調べ、適切なメソッドを実行します。このメソッドは、通常、関連するアダプタのメソッドを実行します。

EnsLib.HTTP.OutboundAdapter は、HTTP 要求をプロダクションの外部に送信し HTTP 応答を受信するアダプタです。このアダプタは、以下のような項目を制御する設定を提供します。

- アダプタが HTTP 要求を送信するサーバおよびポート
- 指定されたサーバおよびポートで、要求するリソースの URL パス
- サーバへの接続に使用するオプションの TLS 構成
- アダプタが要求をルーティングするプロキシ・サーバを指定するためのオプション情報

HTTP 要求には、ローカル InterSystems IRIS サーバのデフォルトの文字エンコードが使用されます。異なる文字エンコードを指定するには、カスタム HTTP 要求を作成して送信します。これについては、"カスタム HTTP 要求の作成" を参照してください。

3.2 アダプタを使用するビジネス・オペレーションの作成

EnsLib.HTTP.OutboundAdapter を使用するビジネス・オペレーションを作成するために、新しいビジネス・オペレーション・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。

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.HTTP.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、このメッセージが処理されると、別のバックグラウンド・ジョブがこ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。
- クラスでは、少なくとも 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>
}
```

- クラスでは、メッセージ・マップ内で名前が付けられたすべてのメソッドを定義します。これらのメソッドは、メッセージ・ハンドラと呼ばれます。各メッセージ・ハンドラは、以下のシグニチャを持っている必要があります。

```
Method Sample(pReq As RequestClass, Output pResp As ResponseClass) As %Status
```

  ここで、Sample はメソッド名、RequestClass は要求メッセージ・クラスの名前、ResponseClass は応答メッセージ・クラスの名前です。通常、メソッド・コードは、ビジネス・オペレーションのプロパティおよび Adapter プロパティのメソッドを参照します。

  メッセージ・クラスの定義方法は、"メッセージの定義" を参照してください。

  メッセージ・ハンドラ・メソッドの定義方法は、"メッセージ・ハンドラ・メソッドの作成" を参照してください。
- その他のオプションと一般情報は、"ビジネス・オペレーション・クラスの定義" を参照してください。

以下の例は、必要となる一般的な構造を示しています。

Class Definition

```
Class EHTTP.NewOperation1 Extends Ens.BusinessOperation
{
Parameter ADAPTER = "EnsLib.HTTP.OutboundAdapter";

Parameter INVOCATION = "Queue";

Method Sample(pReq As RequestClass, Output pResp As ResponseClass) As %Status
{
  Quit $$$ERROR($$$NotImplemented)
}

XData MessageMap
{
<MapItems>
  <MapItem MessageType="RequestClass">
    <Method>Sample</Method>
  </MapItem>
</MapItems>
}
}
```

3.3 メッセージ・ハンドラ・メソッドの作成

EnsLib.HTTP.OutboundAdapter で使用するビジネス・オペレーション・クラスを作成する場合の最大のタスクは、通常、このアダプタで使用するメッセージ・ハンドラ、つまり、プロダクション・メッセージを受信してさまざまな HTTP 操作を実行するメソッドの記述です。

各メッセージ・ハンドラ・メソッドは、以下のシグニチャを持っている必要があります。

```
Method Sample(pReq As RequestClass, Output pResp As ResponseClass) As %Status
```

ここで、Sample はメソッド名、RequestClass は要求メッセージ・クラスの名前、ResponseClass は応答メッセージ・クラスの名前です。

通常、このメソッドは以下の操作を実行します。

1. 受信要求メッセージを調べます。
2. 受信要求の情報を使用して、ビジネス・オペレーションの Adapter プロパティのメソッドを呼び出します。例えば、HTTP GET コマンドを送信する Get メソッドを呼び出します。

   ```
   set status=..Adapter.Get(.pHttpResponse,"Name",pRequest.Name)
   ```

   この例は GET 要求を送信し、pRequest.Name という値を持つ Name と呼ばれるパラメータで渡します。

   使用可能なメソッドについては、次の節で説明します。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。

   これらのメソッドは、出力として HTTP 応答も返します。上記の例では、応答は pHTTPResponse に配置されています。応答は %Net.HttpResponse のインスタンスです。このオブジェクトの使用法は、"HTTP 応答の使用法" を参照してください。
3. HTTP 応答を調べます。
4. HTTP 応答内の情報を使用して、応答メッセージ（Ens.Response またはサブクラスのインスタンス）を作成します。メソッドは出力としてこのメッセージを返します。

   メッセージ・クラスの定義方法に関する基本情報は、"メッセージの定義" を参照してください。
5. 必ず出力引数（pOutput）を設定します。通常、応答メッセージと同じように設定します。この手順は必須です。
6. 適切なステータスを返します。この手順は必須です。

以下に例を示します。

Class Member

```
Method PostMessage(pRequest As EHTTP.Request.OutboundPost,
Output pResponse As EHTTP.Response.OutboundPost) As %Status
{
  Set tSC=$$$OK
  Set input=pRequest.MessageStream
  Set tResponse = ##class(%Net.HttpResponse).%New()
  Set tSC = ..Adapter.Post(.tResponse,,input)

  If $$$ISOK(tSC){
      Set pResponse = ##class(EHTTP.Response.OutboundPost).%New()
      Set len = tResponse.Data.SizeGet()
      While (tResponse.Data.AtEnd = 0) {
        Do pResponse.MessageStream.Write(tResponse.Data.Read())
      }
  }
  Return tSC
}
```

3.4 HTTP コマンドの呼び出し

このアダプタは、HTTP、POST、GET、PATCH、PUT、DELETE の各要求を送信するメソッドを提供します。

- メイン・メソッドは HTTP アクションの後で指定されます。例えば、Post() メソッドは POST 要求を扱います。これらのメイン・メソッドのそれぞれで、応答オブジェクトの出力引数、フォーム変数名のカンマ区切りリスト、およびフォーム変数引数の変数数値 (1 つの変数数値がカンマ区切りリスト内の各名前に対応) を使用できます。1 つのフォーム変数に複数の値を設定する場合は、リスト内で同じ名前を複数回使用できます。もちろん、フラットなスカラ・コンテンツ (通常の Web ページなど) を要求するために、フォーム変数なしでこれらのメソッドを使用することもできます。
- フォーム変数の複雑なセットがある状況では、変数引数のリストの代わりに多次元配列を受け入れる代替メソッドを使用できます。HTTP アクションごとに、これらの代替メソッドのいずれかを指定します。例えば、Post() メソッドの代わりに PostFormDataArray() メソッドを使用できます。多次元配列は、名前リスト内で複数のエントリを使用することなく、所定のフォーム変数の複数の値を配列のサブノードとして提供できるため、情報の組織化に役立ちます。また、位置ではなくフォーム変数名によって配列にインデックスを割り当てることができます。
- その他の特別な HTTP 要求を使用する必要がある場合や、フォーム変数や Cookie 以外の HTTP 要求の面でカスタマイズをする必要がある場合には、低レベルのワーカ・メソッドである SendFormDataArray() を使用できます。

EnsLib.HTTP.OutboundAdapter のメソッドを使用して HTTP コマンドを送信する方法の詳細は、以下の各トピックを参照してください。

- フォーム・データの送信方法
- 要求本文の送信方法
- アダプタ・メソッドについての参照情報
- HTTP 応答についての基本情報

3.4.1 フォーム・データの送信

アダプタのどのメソッドを使用しても HTTP フォーム・データを送信できます。各メソッドは (出力として) HTTP 応答 (%Net.HttpResponse のインスタンス) を返します。このオブジェクトの使用法の詳細は、"HTTP 応答の使用法" を参照してください。

詳細はメソッドによって異なりますが、一部を例に示します。

```
set tFormVar="USER,ROLE,PERIOD"

set tUserID=pRequest.UserID
set tRole=pRequest.UserRole
set tED=pRequest.EffectiveDate

set tSC=..Adapter.Get(.tResponse,tFormVar,tUserID,tRole,tED)
```

この例では、要求メッセージに UserID、UserRole、および EffectiveDate のプロパティがあると仮定します。

3.4.2 要求本文の送信

アダプタのどのメソッドを使用しても要求の本文を送信できます。この場合、要求本文を引数として渡すので、フォーム・データ引数は空のままにしておきます。要求本文は、必要に応じて、ストリームまたは文字列のいずれかになります。ストリームとしての作成と書き込みについては、ドキュメントを参照してください。例えば、"クラスの定義と使用" の “ストリームを使用した作業” の章を参照してください。

これらの各メソッドは (出力として) HTTP 応答 (%Net.HttpResponse のインスタンス) を返します。このオブジェクトの使用法の詳細は、"HTTP 応答の使用法" を参照してください。

詳細はメソッドによって異なりますが、一部を例に示します。

```
set tsc = ..Adapter.Post(.tResponse,,pRequest.MessageStream)
```

この例では、要求メッセージに MessageStream プロパティがあると仮定します。

また、EnsLib.HTTP.OutboundAdapter を使用して、形式を text/xml としたデータを POST 送信するには、HTTPRequest オブジェクト (データの POST 送信に使用するクラス・メソッドへの入力とするオブジェクト) の EntityBody プロパティに、目的のデータを置く必要があります。

3.4.2.1 HTTP ヘッダとしてのストリーム属性

ストリームを作成して要求本文として送信する場合、HTTP 送信アダプタが、HTTP ヘッダとして一部またはすべてのストリーム属性を含むようにすることができます。そのためには、SkipBodyAttrs プロパティを、HTTP ヘッダとして使用されない属性のリストと同じに設定する必要があります。デフォルト値は "*" です。これは、ストリーム属性はデフォルトでは無視される (そしてヘッダとして使用されない) ことを意味します。このプロパティは、実行時設定として使用できません。

ストリーム属性の詳細は、%Library.AbstractStream のクラス参照を参照してください。

3.4.3 HTTP メソッドについての参照情報

この節では、HTTP コマンドの呼び出しに使用できるメソッドについての参照情報を提供します。

Post()

```
Method Post(Output pHttpResponse As %Net.HttpResponse,
            pFormVarNames As %String = "",
            pData...) As %Status
```

構成された宛先に HTTP POST コマンドを送信します ("宛先の指定" を参照)。フォーム・データまたは要求本文を送信します。

HTTP 応答は、最初の引数で出力として返されます。これは %Net.HttpResponse のインスタンスです。このオブジェクトの使用法は、"HTTP 応答の使用法" を参照してください。

このメソッドを使用して、以下のいずれかを行います。

- 指定されたフォーム変数にフォーム・データを送信するには、pFormVarNames 引数および pData 引数を必要に応じて指定します。pFormVarNames は、使用するフォーム変数名のカンマ区切りリストです。リスト内の各名前に対して、pData 引数を提供します。

  指定されたフォーム変数に複数の値を渡すには、pFormVarNames リストにその変数名を複数回含めます。

  追加の pData 引数を提供すると、リストの最後のフォーム変数に割り当てられます。

- フォーム変数ではなく要求本文を送信するには、pFormVarNames を空のままにしておき、pData 引数として要求本文を渡します。

本文テキストとして渡されるデータ値は、文字列またはストリームです。フォーム変数として渡されるデータ値は、文字列である必要があります。

PostFormDataArray()

```
Method PostFormDataArray(Output pHttpResponse As %Net.HttpResponse,
                         pFormVarNames As %String = "",
                         ByRef pData) As %Status
```

構成された宛先に HTTP POST コマンドを送信します ("宛先の指定" を参照)。指定されたフォーム変数にフォーム・データを送信します。

pFormVarNames は、使用するフォーム変数名のカンマ区切りのリストです。リスト内の各名前に対して、pData 引数を提供します。

pData 引数は配列です。配列の最上位ノードは使用されません。各サブノードは、pFormVarNames リストにある対応するフォーム変数のインデックスによって、添え字が付けられます。指定された添え字の値は、次のように指定します。

- 単一の値を持つフォーム変数 (varname) の場合、pData("varname") の値には、送信するフォーム・データ値を指定します。サブノードはありません。
- 複数の値を持つフォーム変数 (varname) の場合、pData("varname") の値には、値の数を指定します。このフォーム変数の各値はサブノードで使用され、ノード内での位置によって添え字が付けられます。

PostURL()

```
method PostURL(pURL As %String,
               Output pHttpResponse As %Net.HttpResponse,
               pFormVarNames As %String = "",
               pData...) as %Status
```

指定された URL (pURL) に HTTP POST コマンドを送信します。指定されたフォーム変数にフォーム・データを送信します。これにより、アダプタの URL プロパティをオーバーライドできます。Post() も参照してください。

Get()

```
Method Get(Output pHttpResponse As %Net.HttpResponse,
           pFormVarNames As %String = "",
           pData...) As %Status
```

構成された宛先に HTTP GET コマンドを送信します ("宛先の指定" を参照)。フォーム・データまたは要求本文を送信します。"Post()" も参照してください。

GetFormDataArray()

```
Method GetFormDataArray(Output pHttpResponse As %Net.HttpResponse,
                        pFormVarNames As %String = "",
                        ByRef pData) As %Status
```

構成された宛先に HTTP GET コマンドを送信します ("宛先の指定" を参照)。指定されたフォーム変数にフォーム・データを送信します。PostFormDataArray() も参照してください。

GetURL()

```
method GetURL(pURL As %String,
              Output pHttpResponse As %Net.HttpResponse,
              pFormVarNames As %String = "",
              pData...) as %Status
```

指定された URL (pURL) に HTTP GET コマンドを送信します。指定されたフォーム変数にフォーム・データを送信します。これにより、アダプタの URL プロパティをオーバーライドできます。Post() も参照してください。

Put()

```
Method Put(Output pHttpResponse As %Net.HttpResponse,
           pFormVarNames As %String = "",
           pData...) As %Status
```

構成された宛先に HTTP PUT コマンドを送信します ("宛先の指定" を参照)。フォーム・データまたは要求本文を送信します。"Post()" も参照してください。

PutFormDataArray()

```
Method PutFormDataArray(Output pHttpResponse As %Net.HttpResponse,
                        pFormVarNames As %String = "",
                        ByRef pData) As %Status
```

構成された宛先に HTTP PUT コマンドを送信します (″宛先の指定″を参照)。指定されたフォーム変数にフォーム・データを送信します。PostFormDataArray() も参照してください。

PutURL()

```
method PutURL(pURL As %String,
              Output pHttpResponse As %Net.HttpResponse,
              pFormVarNames As %String = "",
              pData...) as %Status
```

指定された URL (pURL) に HTTP PUT コマンドを送信します。指定されたフォーム変数にフォーム・データを送信します。これにより、アダプタの URL プロパティをオーバーライドできます。Post() も参照してください。

Patch()

```
Method Patch(Output pHttpResponse As %Net.HttpResponse,
           pFormVarNames As %String = "",
           pData...) As %Status
```

構成された宛先に HTTP PATCH コマンドを送信します (“宛先の指定” を参照)。フォーム・データまたは要求本文を送信します。Post() も参照してください。

PatchFormDataArray()

```
Method PatchFormDataArray(Output pHttpResponse As %Net.HttpResponse,
                        pFormVarNames As %String = "",
                        ByRef pData) As %Status
```

構成された宛先に HTTP PATCH コマンドを送信します (“宛先の指定” を参照)。指定されたフォーム変数にフォーム・データを送信します。PostFormDataArray() も参照してください。

PatchURL()

```
method PatchURL(pURL As %String,
              Output pHttpResponse As %Net.HttpResponse,
              pFormVarNames As %String = "",
              pData...) as %Status
```

指定された URL (pURL) に HTTP PATCH コマンドを送信します。指定されたフォーム変数がフォーム・データに送信されます。これにより、アダプタの URL プロパティをオーバーライドできます。Post() も参照してください。

Delete()

```
method Delete(Output pHttpResponse As %Net.HttpResponse,
              pFormVarNames As %String = "",
              pData...) as %Status
```

構成された宛先に HTTP DELETE コマンドを送信します (″宛先の指定″ を参照)。フォーム・データまたは要求本文を送信します。″Post()″ も参照してください。

DeleteFormDataArray()

```
method DeleteFormDataArray(Output pHttpResponse As %Net.HttpResponse,
                           pFormVarNames As %String = "",
                           ByRef pData) as %Status
```

構成された宛先に HTTP DELETE コマンドを送信します ("宛先の指定" を参照)。指定されたフォーム変数にフォーム・データを送信します。PostFormDataArray() も参照してください。

DeleteURL()

```
method DeleteURL(pURL As %String,
                 Output pHttpResponse As %Net.HttpResponse,
                 pFormVarNames As %String = "",
                 pData...) as %Status
```

指定された URL (pURL) に HTTP DELETE コマンドを送信します。指定されたフォーム変数にフォーム・データを送信します。これにより、アダプタの URL プロパティをオーバーライドできます。Post() も参照してください。

SendFormDataArray()

```
Method SendFormDataArray(Output pHttpResponse As %Net.HttpResponse,
                         pOp As %String,
                         pHttpRequestIn As %Net.HttpRequest,
                         pFormVarNames As %String = "",
                         ByRef pData) As %Status
```

構成された宛先に HTTP 要求を送信します ("宛先の指定" を参照)。指定されたフォーム変数にフォーム・データを送信します。

pOp 引数は、実行する HTTP アクションを指定します。これは、"POST"、"GET"、"PUT" のいずれかです。

pFormVarNames は、使用するフォーム変数名のカンマ区切りのリストです。リスト内の各名前に対して、pData 引数を提供します。

pData 引数は配列です。GetFormDataArray() メソッドも参照してください。

特別なニーズに対しては、%Net.HTTPRequest またはサブクラスのインスタンスを作成し、そのプロパティを指定して、これを pHttpRequestIn 引数として使用します。これを行う場合、HTTP 要求はインスタンスのプロパティを使用して初期化されます。このメソッドを使用すると、送信アダプタの他のメソッドよりも詳細な制御が可能になります。例えば、このメソッドを使用すると、独自の HTTP ヘッダを要求に追加できます。

3.4.4 HTTP 応答の処理

これまでに説明したメソッドを使用する場合、出力として、HTTP 応答を受信します。このオブジェクトは %Net.HttpResponse のインスタンスです。詳細は、"HTTP 応答の使用法" を参照してください。

3.5 Cookie の管理

Cookie とは応答ヘッダ内のテキスト文字列です。サーバはクライアントに対して、この文字列を保存し、その後の HTTP 要求でヘッダ値として返すように要求できます。一部のサーバは、Cookie を使用してオープン・セッションを維持します。

EnsLib.HTTP.OutboundAdapter は、Cookie を管理するための以下のプロパティおよびメソッドも提供します。カスタム・メソッドでこれらを使用できます。

UseCookies プロパティ

このアダプタをインスタンス化するときに HTTP 応答で受信される Cookie を保存するどうか、また、その後の各 HTTP 要求に挿入するかどうかを指定します。UseCookies が真の場合、そのアダプタのインスタンスに関連するすべてのジョブで Cookie のコレクションを保持し、そのジョブが新しく要求を送信するたびに適切な Cookie を選択して一緒に送信します。UseCookies が偽 (0) の場合、Cookie は送信されません。この規則により、各ジョブは要求元の Web サーバと独自の永続セッションを維持できます。

デフォルトは偽です。このプロパティは、実行時設定としても使用できます。

%Cookies プロパティ

Cookie の配列を格納します。ドメイン/サーバによってインデックス化されます。各要素は $LB(name, domain, path, value, expires, secure) です。このプロパティは InterSystems IRIS 多次元配列です。

DeleteCookie() メソッド

```
Method DeleteCookie(pName As %String,
                    pPath As %String,
                    pDomain As %String) As %Status
```

特定の Cookie を削除します。

DeleteCookies() メソッド

```
Method DeleteCookies(pDomain As %String = "",
                     pPath As %String = "") As %Status
```

指定されたドメインやパスからすべての Cookie を削除します。

注釈　Cookie はある HTTP サーバに固有のものです。Cookie を挿入する場合は、特定のサーバへの接続を使用しており、他のサーバではその Cookie を使用できません。

3.6 カスタム HTTP 要求の作成

HTTP 送信アダプタの一般的なメソッド (Get など) を使用すると、アダプタは自動的に HTTP 要求を作成および送信します。この HTTP 要求にはフォーム・データまたは要求本文を含めることができます。特別な場合には、カスタム HTTP 要求を作成して、プロキシ認証や異なる文字エンコードなどの詳細を指定することもできます。

カスタム要求を送信するには、HTTP 送信アダプタの SendFormDataArray() メソッドを使用します。その方法については、"HTTP コマンドの呼び出し" を参照してください。その後、"インターネット・ユーティリティの使用法" の "HTTP 要求の送信と HTTP 応答の読み取り" の章を参照してください。

3.7 HTTP 応答の使用法

送信アダプタを使用して HTTP 要求を送信した後、応答オブジェクト (%Net.HttpResponse) を受信します。具体的には、HTTP.OutboundAdapter の主要メソッドによって HTTP 要求を送信できます。これらのすべてのメソッドで、返信として送信される HTTP 応答にアクセスできます。

- PostFormData()、PostFormDataArray()、GetFormData()、または GetFormDataArray メソッドを使用した場合は、メソッド呼び出しの最初の引数の出力として HTTP 応答が返されます。この引数は %Net.HttpResponse のインスタンスです。
- SendFormDataArray() メソッドを使用した場合は、要求の HttpResponse プロパティが更新されます。このプロパティは %Net.HttpResponse のインスタンスです。

これらのメソッドの詳細は、"HTTP コマンドの呼び出し" を参照してください。

HTTP 応答の使用法の詳細は、"HTTP 要求の送信と HTTP 応答の読み取り" を参照してください。

3.8 例

この節では、2 つの例を示します。

3.8.1 Post の使用例

以下の例は、HTTP 送信アダプタを使用して、構成された宛先に XML メッセージを送信します。まず、ビジネス・オペレーション・クラスは以下のとおりです。

Class Definition

```
Class EHTTP.PostOperation Extends Ens.BusinessOperation
{

Parameter ADAPTER = "EnsLib.HTTP.OutboundAdapter";

Parameter INVOCATION = "Queue";

Method PostMessage(pRequest As EHTTP.Request.OutboundPost,
Output pResponse As EHTTP.Response.OutboundPost) As %Status
{
  Set tSC=$$$OK

  Set tResponse = ##class(%Net.HttpResponse).%New()
  Set tSC = ..Adapter.Post(.tResponse,,pRequest.MessageStream)
  Set stream = ""

  If $$$ISOK(tSC){
      Set pResponse = ##class(EHTTP.Response.OutboundPost).%New()
      Set len = tResponse.Data.SizeGet()
      While (tResponse.Data.AtEnd = 0) {
        Do pResponse.MessageStream.Write(tResponse.Data.Read())
      }
  }
  Return tSC
}

XData MessageMap
{
<MapItems>
  <MapItem MessageType="EHTTP.Request.OutboundPost">
    <Method>PostMessage</Method>
  </MapItem>
</MapItems>
}

}
```

要求メッセージ・クラスは以下のとおりです。

Class Definition

```
Class EHTTP.Request.OutboundPost Extends Ens.Request
{

/// MessageStream contains the complete SOAP Message to post
Property MessageStream As %GlobalCharacterStream(CONTENT = "MIXED");

}
```

応答メッセージ・クラスは以下のとおりです。

Class Definition

```
Class EHTTP.Response.OutboundPost Extends Ens.Response
{

/// MessageStream contains the Response to the SOAP Message post
Property MessageStream As %GlobalCharacterStream(CONTENT = "MIXED");

}
```

3.8.2 Get の使用例

以下の例では、Get() メソッドを使用しています。

```
Method GetEvent(pRequest As EHTTP.Request.GetEvent,
ByRef pResponse As EHTTP.Response.GetEvent) As %Status
{
  Set tSC=$$$OK

  Set pResponse=##class(EHTTP.Response.GetEvent).%New()

  Set tFormVar="Name,Country,City"

  Set tName=pRequest.EventName
  Set tCountry=pRequest.EventCountry
  Set tCity=pRequest.EventCity

  Set tSC=..Adapter.Get(.tResponse,tFormVar,tName,tCountry,tCity)

  If '$IsObject(tResponse) {Quit
  }
  ;
  ;now parse the XML stream
  ;
  Set reader=(%XML.Reader).%New()
  Do tResponse.Data.Rewind()
  Set tSC=reader.OpenStream(tResponse.Data)

  If $$$ISERR(tSC) {
     $$$LOGWARNING("Unable to open the XML Stream")
     $$$TRACE("XML request probably failed")
     Do tResponse.Data.Rewind()
     Set traceline = tResponse.Data.Read(1000)
     $$$TRACE(traceline)
     Set tSC=$$$OK
     Quit
  }
  ;
  ;Associate a class name with the XML element Name
  ;
  Do reader.Correlate("EventResult","EHTTP.Event.EventResult")

  If reader.Next(.tResults,.tSC)
  {
     Set pResponse.EventValues=tResults.EventValues
  }
  Return tSC
}
```

この例は、対象のサーバから XML ファイルを取得して、そのファイルを解析し、特定のデータを取得します。

3.9 ビジネス・オペレーションの追加と構成

ビジネス・オペレーション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・クラスのインスタンスをプロダクションに追加します。
2. 特定の外部データ・ソースと通信を行うためのアダプタを構成します。具体的には、以下を行います。
   - HTTP 要求の宛先の指定
   - オプションのプロキシ・サーバの指定
3. ビジネス・オペレーションを有効化します。
4. プロダクションを実行します。

3.9.1 宛先サーバと URL パスの指定

以下の実行時設定を使用して、HTTP 要求を送信する宛先を指定します。

- HTTPサーバ
- HTTPポート
- 認証情報
- URL

3.9.1.1 例

例えば、送信先の URL が `http://122.67.980.43/HTTPReceive.aspx` であるとします。

この場合、以下の設定を使用します。

設定	値
HTTPServer	`122.67.980.43`
URL	`HTTPReceive.aspx`

3.9.2 プロキシ・サーバの指定

必要な場合は、以下の実行時設定を使用して、プロキシ・サーバを介して HTTP 要求をルーティングします。

- プロキシ・サーバ
- プロキシ・ポート
- HTTPSプロキシ

4

組み込み HTTP コンポーネント

InterSystems IRIS® データ・プラットフォームには、HTTP アダプタを使用する組み込みビジネス・ホストが用意されています。これを使用すると、カスタムのビジネス・サービスやビジネス・オペレーションを作成せずに、プロダクションに HTTP サポートを追加できます。HTTP 受信アダプタを使用するビジネス・サービスが必要な場合は、プロダクションに EnsLib.HTTP.GenericService を追加します。同様に、HTTP 送信アダプタの活用を必要とするプロダクションでは、EnsLib.HTTP.GenericOperation を使用できます。

InterSystems IRIS には、プロダクション全体に HTTP 要求または HTTP 応答を伝達するように設計されたメッセージ・クラスも用意されています。このメッセージ・クラス EnsLib.HTTP.GenericMessage には、HTTP 要求のヘッダと本文が記述されています。まったくの白紙から EnsLib.HTTP.GenericMessage を構築し、EnsLib.HTTP.GenericOperation を使用してそれを送信する方が有利な場合もあります。このプロセスでは、本文を作成し、ヘッダを記述してから新しいメッセージを作成します。以下に例を示します。

```
// Build the header
#dim tRESTHTTPHeaders
Set tRESTHTTPHeaders("HttpRequest")="POST"
Set tRESTHTTPHeaders("HTTPVersion")="1.1"

// Build the body
#dim tPOSTStream = ##class(%Stream.GlobalCharacter).%New()
#dim tPOSTJSON = {}
Set tPOSTJSON.projection = []
Do tPOSTJSON.projection.%Push("%Doc")
Do tPOSTJSON.%ToJSON(.tPOSTStream)

// Add more headers
Set tRESTHTTPHeaders("content-length") = tPOSTStream.Size
Set tRESTHTTPHeaders("content-type") = "application/json"

// Create message
Return ##class(EnsLib.HTTP.GenericMessage).%New(tPOSTStream,,.tRESTHTTPHeaders)
```

4.1 メッセージ・ヘッダの詳細

InterSystems IRIS で作成した EnsLib.HTTP.GenericMessage のヘッダの例を以下に示します。

```
<HTTPHeaders>
  <HTTPHeadersItem HTTPHeadersKey="CharEncoding" xsi:nil="true"></HTTPHeadersItem>
  <HTTPHeadersItem HTTPHeadersKey="EnsConfigName">ForDocker</HTTPHeadersItem>
  <HTTPHeadersItem HTTPHeadersKey="HTTPVersion">1.1</HTTPHeadersItem>
  <HTTPHeadersItem HTTPHeadersKey="HttpRequest">POST</HTTPHeadersItem>
  <HTTPHeadersItem HTTPHeadersKey="IParams">0</HTTPHeadersItem>
  <HTTPHeadersItem HTTPHeadersKey="RawParams" xsi:nil="true"></HTTPHeadersItem>
  <HTTPHeadersItem HTTPHeadersKey="TranslationTable">RAW</HTTPHeadersItem>
  <HTTPHeadersItem HTTPHeadersKey="URL">/api/atelier/v4/TEST/action/query</HTTPHeadersItem>
  <HTTPHeadersItem HTTPHeadersKey="accept">application/json, text/plain, */*</HTTPHeadersItem>
  <HTTPHeadersItem HTTPHeadersKey="connection">close</HTTPHeadersItem>
  <HTTPHeadersItem HTTPHeadersKey="content-length">127</HTTPHeadersItem>
  <HTTPHeadersItem HTTPHeadersKey="content-type">application/json</HTTPHeadersItem>
  <HTTPHeadersItem
HTTPHeadersKey="cookie">CSPSESSIONID-SP-42773-UP-api-atelier-=001000000000t7LT7VX9SXUNIM7S8yKxK44S$uBbUtzNHVjIAk</HTTPHeadersItem>

  <HTTPHeadersItem HTTPHeadersKey="host">127.0.0.1:55773</HTTPHeadersItem>
  <HTTPHeadersItem HTTPHeadersKey="user-agent">axios/0.21.1</HTTPHeadersItem>
</HTTPHeaders>
```

EnsLib.HTTP.GenericMessage を白紙から作成する場合は、これらのヘッダがすべて必要なわけではありません。少なくとも、HttpRequest、HTTPVersion、content-type の各ヘッダを定義する必要があります。また、一般的には content-length ヘッダの定義も必要です。

EnsLib.HTTP.GenericOperation を使用しているプロダクションでは、送信アダプタの URL プロパティと共にメッセージの URL ヘッダが処理され、要求の URL パスが判断されます。アダプタの他のプロパティ同様に、URL プロパティは管理ポータルを使用して設定し、プロダクションのビジネス・オペレーションの URL 設定を定義します。URL ヘッダと URL プロパティの動作の詳細は、アダプタの URL プロパティを参照してください。

EnsLib.HTTP.GenericMessage.%New() を使用して HTTP メッセージを作成する場合は、ヘッダを渡すためのオプションをいくつか使用できます。一般的に、ヘッダの引数は配列で渡します。ただし、%New メソッドでは、ObjectScript 配列参照、自身の属性のコピー元を伴う %AbstractStream、自身のヘッダのコピー元を伴う %Net.HttpResponse オブジェクト、または 'a=1,b=2,c=3' 形式の文字列としたヘッダも使用できます。

HTTP アダプタ設定

ここでは、HTTP アダプタの参照情報を提供します。

"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。

HTTP 受信アダプタの設定

HTTP 受信アダプタ EnsLib.HTTP.InboundAdapter の設定に関する参照情報を提供します。

概要

受信 HTTP 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	[呼び出し間隔]、[ポート]
接続設定	[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・インタフェース]、[標準リクエスト有効]
追加設定	[文字セット]、[文字セット強制]、[GenerateSuperSessionID]

残りの設定はすべてのビジネス・サービスに共通のものです。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。

許可IPアドレス

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。

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。

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。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。

"HTTP 要求のソースの指定"も参照してください。

呼び出し間隔

このアダプタはポーリングを使用しません。接続をリッスンし、接続が確立されると、その接続上でメッセージをリッスンします (また、可能な場合は直ちに応答します)。ただし、アダプタのシャットダウン要求または休止要求があったかどうかを定期的に確認します。この設定は、EnsLib.HTTP.InboundAdapter がこの確認を行う間隔を秒単位で指定します。

デフォルト値は 5 秒です。最小値は 0.1 秒です。

文字セット

受信データの文字セットを指定します。InterSystems IRIS® は、自動的にこのエンコードから変換します。この設定値は大文字と小文字が区別されません。`Binary` は、バイナリ・ファイル、新規行文字と改行文字が異なるデータ、または、HL7 バージョン 2 や EDI メッセージのように変更しないまま残す必要のあるデータに対して使用します。テキスト・ドキュメントを転送するときは、他の設定が便利な場合があります。選択肢は以下のとおりです。

- `Auto` – 受信 HTTP ヘッダ `Content-Type` フィールドで宣言されたエンコード方法を使用します。これがデフォルトです。

- `AutoXML` ― 受信 XML の本文コンテンツ (存在する場合) の XML ヘッダで宣言されているエンコード方法を使用します。
- `Binary` ― 文字のエンコード変換を行わず、本文の生のバイトを読み取ります。
- `RawBytes` ― 文字のエンコード変換を行わず、本文の生のバイトを読み取ります。
- `Default` ― ローカル InterSystems IRIS サーバのデフォルトの文字エンコードを使用します。
- `Latin1` ― ISO Latin1 8 ビット・エンコード
- `ISO-8859-1` ― ISO Latin1 8 ビット・エンコード
- `UTF-8` ― Unicode 8 ビット・エンコード
- `UCS2` ― Unicode 16 ビット・エンコード
- `UCS2-BE` ― Unicode 16 ビット・エンコード (ビッグ・エンディアン形式)
- InterSystems IRIS に NLS (各国言語サポート) をインストールするための、国際文字エンコード規格に基づくその他のエイリアス。

文字セットおよび変換テーブルの詳細は、"変換テーブル" を参照してください。

文字セット強制

この設定が真の場合、アダプタは、受信 HTTP ヘッダ `Content-Type` フィールドで宣言された文字セットではなく、Charset 設定を使用します。デフォルトは偽です。

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、アダプタは各接続に対する新しいジョブを生成しません。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。

標準リクエスト有効

アダプタに関連付けられているビジネス・サービスが、Web ゲートウェイ・ポートと指定したカスタム・ポートの両方から要求を受信できることを指定します。

この設定は、HTTP ビジネス・サービスが、EnsLib.HTTP.Service クラスを拡張するカスタム・クラスに基づいている場合にのみ適用されます。詳細は、"HTTP 受信アダプタを使用するビジネス・サービスの作成" を参照してください。

HL7 HTTP ビジネス・サービスのデフォルト値は `false` です。その他すべての HTTP ビジネス・サービスのデフォルト値は `true` です。

この設定を有効にする場合、Web ゲートウェイがビジネス・サービスと確実に通信できるようにするために、以下のような追加の手順が必要になります。

- カスタム・クラスに基づいているビジネス・サービスが 1 つのみの場合は、そのビジネス・サービスの構成名をカスタム・クラスの名前に設定します。次に、ビジネス・サービスを呼び出す URL にカスタム・クラスの名前を含めます。以下に例を示します。

  ```
  /csp/myapplication/mynamespace/myserviceclassname
  ```

- カスタム・クラスに基づいているビジネス・サービスが複数ある場合は、ビジネス・サービスを呼び出す URL に `?CfgItem=configitemname` パラメータを含めます。configitemname は、ビジネス・サービスの構成名です。以下に例を示します。

  ```
  /csp/myapplication/mynamespace/myserviceclassname?CfgItem=myserviceconfignameA
  ```

  ```
  /csp/myapplication/mynamespace/myserviceclassname?CfgItem=myserviceconfignameB
  ```

- さらに、ビジネス・サービスとの通信に CSP Web アプリケーションを使用する場合は、その Web アプリケーションの [**ディスパッチ・クラス**] をカスタム・クラスの名前に設定します。

[**標準リクエスト有効**] を選択して、プール・サイズを 0 に設定した場合、ビジネス・サービスは Web ゲートウェイ・ポートからのみデータを受信し、カスタム・ポートからは受信しません。

注釈　Web ゲートウェイ・ポートを使用する場合、メッセージの先入れ先出し処理はサポートされません。

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。

ポート

アダプタが HTTP 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・ポートの初期データを受信した後、次に続く受信読み取りを待機する秒数です。

接続を維持

要求の間に、TCP 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、各メッセージの受信後に直ちに接続を切断します。
- この値が正の場合、アイドル・タイムを秒単位で指定します。アダプタは、このアイドル・タイムの経過後に接続を切断します。
- 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。

HTTP 送信アダプタの設定

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

概要

送信 HTTP 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	[HTTPサーバ]、[HTTPポート]、[URL]、[認証情報]
接続設定	[SSL構成]、[SSL チェック・サーバ ID]、[プロキシ・サーバ]、[プロキシ・ポート]、[HTTPSプロキシ]、[HTTP プロキシ・トンネル]、[応答タイムアウト]、[ConnectTimeout]、[WriteTimeout]、[LocalInterface]
追加設定	Cookie の使用、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のものです。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。

接続タイムアウト

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。

認証情報

指定された宛先 URL への接続を承認できるプロダクション認証情報の ID を識別します。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ポート

HTTP 要求の送信先となるサーバの TCP ポート (デフォルトでは 80、SSLConfig が指定されている場合は 443 を使用)。80 以外の値は、送信する HTTP 要求の Host: ヘッダに含められます。詳細は、"宛先サーバと URL パスの指定" を参照してください。

HTTPサーバ

HTTP 要求の送信先となるサーバの IP アドレス。これは、送信する HTTP 要求の Host: ヘッダで使用されます。"宛先サーバと URL パスの指定" も参照してください。

ローカル・インタフェース

HTTP 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。

HTTPSプロキシ

プロキシ・サーバを使用する場合は、そのプロキシ・サーバがターゲット・システムとの通信に HTTPS を使用するかどうかを指定します。このオプションを有効にして [プロキシ・サーバ] プロパティに値を指定すると、プロキシ・サーバは HTTP ページではなく HTTPS ページの要求を発行します。また、このオプションを有効にすると、デフォルトの HTTP ポートが、HTTPS ポートである 443 に変更されます。

プロキシ・ポート

プロキシ・サーバを使用する場合、接続先とするそのプロキシ・サーバ上のポートを指定します。デフォルト値は 8080 です。

プロキシ・サーバ

プロキシ・サーバを使用する場合、プロキシ・サーバのホスト名を指定します。指定したプロキシ・サーバの、指定したプロキシ・ポートに対して、要求が発行されます。値を指定しない場合、プロキシ・サーバは使用されません。

HTTP プロキシ・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。

応答タイムアウト

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。

応答が受信されない場合、アダプタは、Failure Timeout を Retry Interval で割った回数だけ繰り返し再試行します。

SSL チェック・サーバ ID

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。

SSL構成

この接続の認証に使用する既存の TLS 構成の名前。アダプタ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。この文字列を [SSL構成] 設定の値として使用します。

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 HTTP 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、HTTP 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。

HTTP 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。

URL

サーバから要求するための URL パス (`http://` またはサーバ・アドレスを含みません)。

"宛先サーバと URL パスの指定" も参照してください。

Cookie の使用

このアダプタをインスタンス化するときに HTTP 応答で受信される Cookie を保存するどうか、また、その後の各 HTTP 要求に挿入するかどうかを指定します。

WriteTimeout

Web サーバへの書き込みのタイムアウト値を指定します。この設定が NULL の場合、タイムアウトはありません。