プロダクション内での電子メール・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電子メール・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 電子メール受信アダプタの使用法 1
1.1 全般的な動作 1
1.2 電子メール受信アダプタを使用するビジネス・サービスの作成 3
1.3 OnProcessInput() メソッドの実装 3
1.4 電子メール・メッセージの使用法 4
1.4.1 メッセージ・コンテンツ 5
1.4.2 メッセージ・ヘッダ 5
1.4.3 その他のメッセージ情報 5
1.5 ビジネス・サービスの追加と構成 6
1.5.1 POP3 サーバへのログイン方法の指定 6
1.5.2 取得するメッセージの指定 7

2 電子メール送信アダプタの使用法 9
2.1 全般的な動作 9
2.2 アダプタを使用するビジネス・オペレーションの作成 9
2.3 メッセージ・ハンドラ・メソッドの作成 11
2.3.1 使用可能なメソッド 11
2.3.2 例 12
2.4 電子メール・メッセージの作成 13
2.4.1 電子メール・メッセージの作成 13
2.4.2 メッセージ・パートのコンテンツおよびヘッダの指定 14
2.4.3 メッセージ・ヘッダの指定 14
2.4.4 アタッチメントの追加 16
2.4.5 例 16
2.5 例 17
2.6 ビジネス・オペレーションの追加と構成 18
2.6.1 SMTP サーバへのログイン方法の指定 19
2.6.2 追加の電子メール・アドレスの指定 19

電子メール・アダプタの設定 21
電子メール受信アダプタに関する設定 22
電子メール送信アダプタに関する設定 25

1

電子メール受信アダプタの使用法

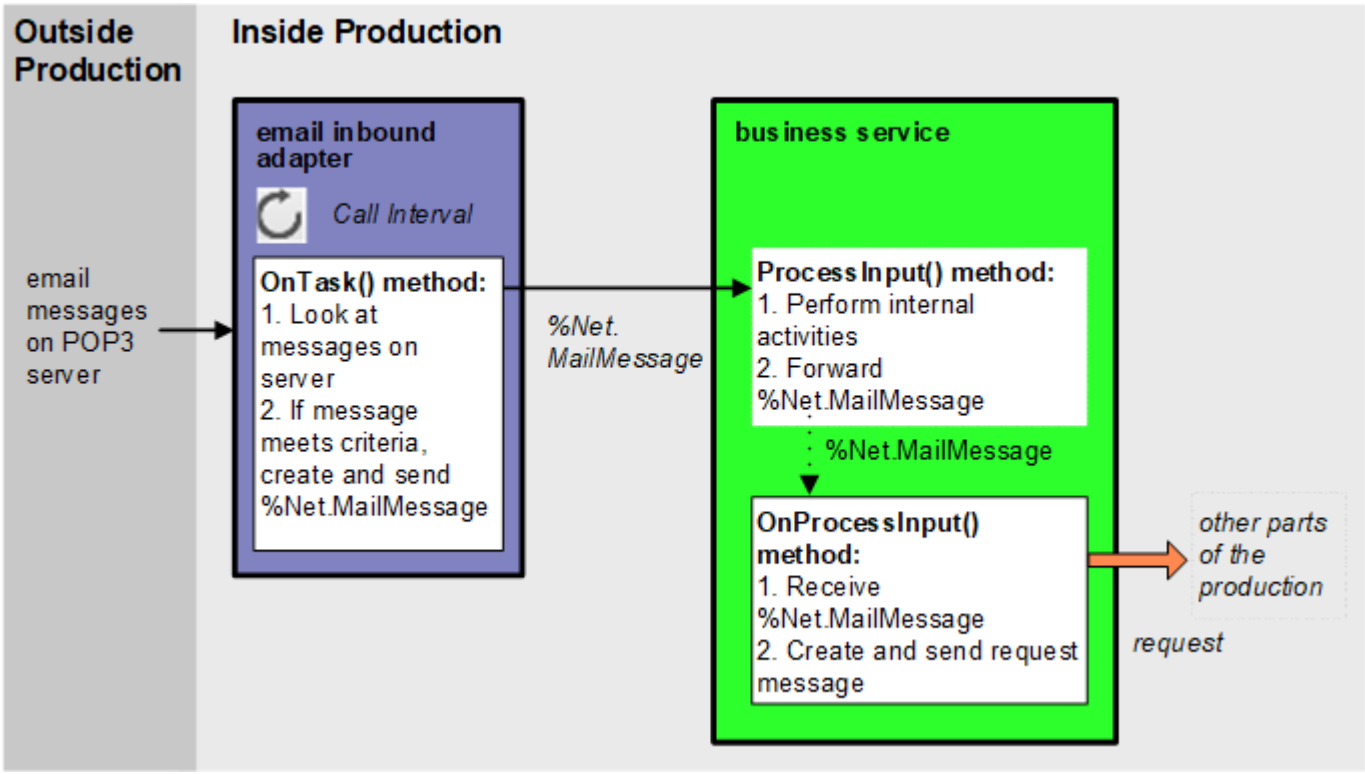
ここでは、プロダクションでの電子メールの送受信を可能にするための電子メール受信アダプタ (EnsLib.EMail.InboundAdapter) の既定の動作とプロダクションでのこのアダプタの使用方法について説明します。使用する SMTP サーバまたは POP3 サーバの要件および制限事項を理解している必要があります。

1.1 全般的な動作

まず、アダプタに対して指定する詳細について理解しておくと役立ちます。EnsLib.EMail.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。

- 使用する POP3 サーバ、およびメッセージが格納されるメールボックスに対するログイン情報
- 対象のメッセージを指定する一致条件
- アダプタが新規入力をチェックする頻度を制御する、ポーリング間隔

通常、受信電子メール・アダプタ (EnsLib.EMail.InboundAdapter) は定期的にメールボックスをチェックし、条件に一致するものを検索し、(%Net.MailMessage のインスタンスとして) 関連するビジネス・サービスにメッセージを送信し、電子メール・サーバからそのメッセージを削除します。ユーザが作成および構成するビジネス・サービスでは、このメッセージ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。



さらに具体的に説明します。

1. アダプタは定期的に OnTask() メソッドを実行します。このメソッドは、特定のユーザ名とパスワードを使用して、POP3 サーバに接続してログオンします。ポーリング間隔は CallInterval 設定によって決定されます。
2. アダプタはこのメールボックス内のすべてのメッセージを調べ、一致条件と比較します。
3. アダプタは、条件に一致するメッセージを見つけると、以下を実行します。
   a. アダプタは %Net.MailMessage クラスのインスタンスを作成し、電子メール・データをインスタンスに入れます。
   b. アダプタは、関連するビジネス・サービス・クラスの内部 ProcessInput() メソッドを呼び出し、%Net.MailMessage インスタンスを入力として渡します。
   c. アダプタは、サーバからメール・メッセージを削除します。
4. 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。
5. ProcessInput() メソッドがカスタムの OnProcessInput() メソッドを呼び出し、%Net.MailMessage インスタンス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。

応答メッセージは、同じパスを逆向きにたどります。

1.2 電子メール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、そのサービス・クラス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、該当するメッセージ・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.EMail.InboundAdapter である必要があります。
- 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。
- 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、必要となる一般的な構造を示しています。

Class Definition

```
Class EEMA.EmailService Extends Ens.BusinessService
{
Parameter ADAPTER = "EnsLib.EMail.InboundAdapter";

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

1.3 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() メソッドは以下のシグニチャを持つ必要があります。

```
Method OnProcessInput(pInput As %Net.MailMessage,
                      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電子メール・メッセージ・オブジェクトです。これは %Net.MailMessage のインスタンスです。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。

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. 電子メール・メッセージを調べて、それをどのように使用するかを決定します。

   電子メール・メッセージの構造はさまざまであり、それぞれ異なる処理が必要です。まず、メッセージの構造が期待どおりであることを確認したい場合があります。つまり、マルチパート・メッセージであるかどうか、また、各パートがバイナリかどうかを確認します。詳細は、"電子メール・メッセージの使用法" を参照してください。

2. メッセージの構造が確認できたら、%Net.MailMessage の別のプロパティを使用してメッセージ本文またはヘッダにアクセスします。"電子メール・メッセージの使用法" を参照してください。

3. ビジネス・サービスから送信され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。

4. 要求メッセージに対し、電子メール・メッセージの値を使用して適切にプロパティを設定します。
5. 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。
6. 必ず出力引数 (pOutput) を設定します。通常、受信した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。
7. 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 %Net.MailMessage,
pOutput As %RegisteredObject) As %Status
{
    //Check if mail message has multiple parts
    Set multi=pInput.IsMultiPart
    If multi
        {$$$TRACE("This message has multiple parts; not expected")
        Quit $$$ERROR($$$GeneralError,"Message has multiple parts")
        }

    //Check if mail message is binary
    Set bin=pInput.IsBinary
    If bin
        {$$$TRACE("This message is binary; not expected")
        Quit $$$ERROR($$$GeneralError,"Message is binary")
        }

    //Check if mail message is HTML
    Set html=pInput.IsHTML
    If html
        {$$$TRACE("This message is HTML not expected")
        Quit $$$ERROR($$$GeneralError,"Message is HTML")
        }

    //now safe to get text of message
    Set pReq=##class(EEMA.EmailContents).%New()
    Set pReq.MessageText=pInput.TextData

    Set tSc=..SendRequestSync("EEMA.EmailProcessor",pReq,.pResp)
    Set pOutput=pResp

    Quit tSc
}
```

1.4 電子メール・メッセージの使用法

前述したとおり、電子メール・メッセージ (%Net.MailMessage) を取得した後、通常はメッセージの種類とメッセージを読む方法を確認します。つまり、マルチパート・メッセージかどうか、および各パートがバイナリかどうかを判断します。

この手順では ContentType プロパティを使用できます。このプロパティは、Content-Type プロパティと同じです。または、IsBinary、IsHTML、および IsMultiPart プロパティを使用できます。これらのプロパティは、ContentType と同じ情報を間接的に提供します。

メッセージがマルチパート・メッセージの場合、各パートは %Net.MailMessagePart のインスタンスです。

メッセージにはメッセージ・ヘッダがあります。メッセージの各パートにメッセージ・ヘッダがある場合もあります。%Net.MailMessage や %Net.MailMessagePart のさまざまなプロパティを介してヘッダにアクセスできます。

1.4.1 メッセージ・コンテンツ

全般的なメッセージ構造がわかったら、以下の方法でコンテンツを取得します。

- マルチパート・メッセージの場合は、Parts プロパティを使用します。これはパートの配列です。Count() メソッドを使用してパート数を取得します。GetAt() メソッドを使用して指定されたパートを取得します。各パートのキーは整数で、最初のパートは 1 から始まります。
- バイナリ・メッセージ (またはメッセージ・パート) には、BinaryData プロパティを使用します。
- テキスト・メッセージ (またはメッセージ・パート) には、TextData プロパティを使用します。
  - IsHTML が 0 の場合、TextData プロパティは通常のテキスト文字列です。
  - IsHTML が 1 の場合、TextData プロパティは HTML テキスト文字列です。

メッセージを送信する電子メール・クライアントが、メッセージ内のラッピングを判断します。メール・サーバも InterSystems IRIS も、これを制御できません。

1.4.2 メッセージ・ヘッダ

メッセージ自体およびメッセージの各パートには、一連のヘッダがあります。

%Net.MailMessage クラスは、最も一般的に使用されるヘッダ用のプロパティを提供します。

- To ― このメッセージの送信先の電子メール・アドレスのリスト。このプロパティは、標準の InterSystems IRIS リストです。これを使用するには、標準のリスト・メソッドである Insert()、GetAt()、RemoveAt()、Count()、および Clear() を使用します。
- From ― このメッセージの送信元の電子メール・アドレス。
- Date ― このメッセージの日付。
- Subject ― このメッセージの件名を含む文字列。
- Sender ― メッセージの実際の送信者。
- Cc ― このメッセージの送信先のカーボン・コピー・アドレスのリスト。
- Bcc ― このメッセージの送信先のブラインド・カーボン・コピー・アドレスのリスト。メッセージの受信者がブラインド・カーボン・コピー・リストに記載されている場合、このプロパティにはその受信者のアドレスが含まれます。それ以外の場合は NULL です。

%Net.MailMessage および %Net.MailMessagePart は共に、以下のプロパティを提供します。これらのプロパティは、その他のヘッダを提供します。

- ContentType ― メッセージまたはメッセージ・パートの `Content-Type` ヘッダ。
- ContentTransferEncoding ― メッセージまたはメッセージ・パートの `Content-Transfer-Encoding` ヘッダ。
- Headers ― 任意の追加ヘッダへのアクセスを提供する、多次元配列。

また、GetAttribute() メソッドも使用できます。ヘッダ名と属性を指定すると、このメソッドはその属性の値を返します。

1.4.3 その他のメッセージ情報

以下のプロパティおよびメソッドは、メッセージに関する追加情報を提供します。

- MessageSize プロパティは、付加された電子メール・メッセージを除く、メッセージの全長を示します。

- GetLocalDateTime() メソッドは、メッセージが取得された日時を返します。$HOROLOG 形式で現地時刻に変換されます。
- GetUTCDateTime() メソッドは、メッセージが取得された日時を返します。$HOROLOG 形式で UTC に変換されます。
- GetUTCSeconds() メソッドは、メッセージが取得された日時を返します。1840 年 12 月 31 日からの経過秒数で表されます。
- HToSeconds() クラス・メソッドは、$HOROLOG 形式の日時を 1840 年 12 月 31 日からの経過秒数に変換します。
- SecondsToH() クラス・メソッドは、1840 年 12 月 31 日からの経過秒数を $HOROLOG 形式の日時に変換します。

1.5 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. ビジネス・サービスを有効化します。
3. POP3 メール・サーバにアクセスしてメッセージをダウンロードするアダプタを構成します。具体的には、以下を行います。
   - メールボックスにアクセスするために必要な、POP3 メール・サーバ情報およびログイン情報を指定します。
   - ダウンロードするメッセージを決定するための条件を指定します。
   - Call Interv¥al を使用してアダプタが電子メールを検索する頻度を指定します。
4. プロダクションを実行します。

1.5.1 POP3 サーバへのログイン方法の指定

以下の設定に対して値を指定し、ログオンする POP3 サーバ、およびメールボックスにアクセスするためのセキュリティ情報を指定します。

- POP3サーバ
- POP3ポート
- Credentials
- SSL構成

例えば、以下のような値を使用できます。

設定	値
POP3サーバ	`pop.hotpop.com`
POP3ポート	110
Credentials	`hotpop`

この例にある `hotpop` は、ユーザ名 `isctest@hotpop.com` とこれに対応するパスワードから成るプロダクション認証情報 の ID です。

1.5.2 取得するメッセージの指定

以下の設定に対して値を指定し、取得するメッセージを制御します。指定された条件にすべて一致するメッセージのみが使用されます。条件の照合では大文字と小文字が区別されます。

- Match From
- Match To
- Match Subject

これらの条件を変更すると、アダプタは、以前は条件に一致しなかったメッセージを調べ、必要に応じて処理します。

2

電子メール送信アダプタの使用法

ここでは、電子メール送信アダプタ (EnsLib.EMail.OutboundAdapter) 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。

このドキュメントに記載されていない設定は、"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。

2.1 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。

電子メール送信アダプタ (EnsLib.EMail.OutboundAdapter) には、以下のような項目の指定に使用する設定が用意されています。

- 接続する SMTP サーバ、およびそのサーバに対して必要なログイン情報 (ある場合)
- 電子メール送信元のデフォルト・アドレス (`From:` ヘッダ)
- ハードコードされた受信者の他に、アダプタが送信するメッセージに追加する受信者

このアダプタは、以下の 3 つのアクションを行うメソッドを提供します。

- `To:` リストへの受信者の追加
- `Cc:` リストへの受信者の追加
- メッセージの送信

2.2 アダプタを使用するビジネス・オペレーションの作成

EnsLib.EMail.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、そのクラス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、該当するメッセージ・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・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.EMail.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。
- 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、これらのメソッドは、ビジネス・オペレーションの Adapter プロパティのプロパティおよびメソッドを参照します。
- 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 EEMA.NewOperation1 Extends Ens.BusinessOperation
{
Parameter ADAPTER = "EnsLib.EMail.OutboundAdapter";

Parameter INVOCATION = "Queue";

Method SampleCall(pRequest As Ens.Request,
                  Output pResponse As Ens.Response) As %Status
{
  Quit $$$ERROR($$$NotImplemented)
}

XData MessageMap
{
<MapItems>
  <MapItem MessageType="Ens.Request">
    <Method>SampleCall</Method>
  </MapItem>
</MapItems>
}
}
```

2.3 メッセージ・ハンドラ・メソッドの作成

EnsLib.EMail.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 SMTP サーバを介して電子メール・メッセージを送信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. 送信する電子メール・メッセージを作成します。詳細は、"電子メール・メッセージの作成" を参照してください。
3. 任意で、ビジネス・オペレーションの Adapter プロパティの AddRecipients()、AddCcRecipients()、および AddBccRecipients() メソッドを呼び出します。これらのメソッドは、電子メール・メッセージの送信時に、To: ヘッダ、Cc: ヘッダ、および Bcc: ヘッダに電子メール・アドレスを追加します。これらのメソッドについては、後で説明します。
4. ビジネス・オペレーションの Adapter プロパティの SendMail() メソッドを呼び出します。

   ```
   Set tSc=..Adapter.SendMail(email,.pf)
   ```

   このメソッドについては、後で説明します。
5. 応答を調べます。
6. 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・クラスの定義方法は、"メッセージの定義" を参照してください。
7. 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。
8. 適切なステータスを返します。この手順は必須です。

2.3.1 使用可能なメソッド

このアダプタは、以下のメソッドを提供します。

SendMail

```
Method SendMail(pMailMessage As %Net.MailMessage,
                Output pFailedRecipients As %ListOfDataTypes) As %Status
```

電子メール・メッセージを指定すると、このメソッドは、構成された SMTP サーバを使用して電子メール・メッセージを送信します。SMTP サーバによって失敗した受信者のリストが返されると、このメソッドは、出力として失敗した受信者のリストを返します。

AddRecipients

```
Method AddRecipients(pMailMessage As %Net.MailMessage,
                     pRecipients As %String)
```

電子メール・メッセージを指定すると、このメソッドは、メッセージの To: ヘッダに、リストされた電子メール・アドレスを追加します。

AddCcRecipients

```
Method AddCcRecipients(pMailMessage As %Net.MailMessage,
                       pRecipients As %String)
```

電子メール・メッセージを指定すると、このメソッドは、メッセージの Cc: ヘッダに、リストされた電子メール・アドレスを追加します。

AddBccRecipients

```
Method AddBccRecipients(pMailMessage As %Net.MailMessage,
                        pRecipients As %String)
```

電子メール・メッセージを指定すると、このメソッドは、メッセージの Bcc: ヘッダに、リストされた電子メール・アドレスを追加します。電子メールを送信する際は、To: ヘッダまたは Cc: ヘッダに少なくとも 1 つのアドレスを指定する必要があります。

ContinueAfterBadSendSet

```
Method ContinueAfterBadSendSet(%val As %Integer) as %Status
```

%val が真の場合は、1 人以上の受信者のアドレスが無効なときに、アダプタはメッセージの送信を続行します。デフォルトは真です。

2.3.2 例

メソッドは以下のようになります。

Class Member

```
Method SendMultipartEmailMessage(pRequest As EEMA.MultipartEmailMsg,
Output pResponse As Ens.Response) As %Status
{
    Set part1=##class(%Net.MailMessage).%New()
    Do part1.TextData.Write(pRequest.Message1)
    Set part2=##class(%Net.MailMessage).%New()
    Do part2.TextData.Write(pRequest.Message2)
    Set part3=##class(%Net.MailMessage).%New()
    Do part3.TextData.Write(pRequest.Message3)

    Set email=##class(%Net.MailMessage).%New()
    Set email.Subject=pRequest.Subject
    Set email.IsMultiPart=1
    Do email.Parts.SetAt(part1,1)
    Do email.Parts.SetAt(part2,2)
    Do email.Parts.SetAt(part3,3)

    Set tSc=..Adapter.SendMail(email,.pf)
    Set pResponse=##class(EEMA.EmailFailedRecipients).%New()
    Set pResponse.FailedRecipients=pf

    if pf.Count()'=""
    {
        set count=pf.Count()
        for i=1:1:count
        {
            $$$TRACE("Failed recipient:"_pf.GetAt(i))
            }
        }

    Quit tSc
}
```

電子メール・メッセージの作成については、次の節を参照してください。

2.4 電子メール・メッセージの作成

メッセージには 1 つまたは複数のパートを含めることができます。各パートにはコンテンツが含まれ、ヘッダを含めることができます。メッセージ自体にもヘッダがあります。以下の節で、次のトピックについて説明します。

- 一般的な電子メール・メッセージの作成方法
- 指定されたメッセージ・パートのコンテンツおよびヘッダを指定する方法
- メッセージおよびそのパートのヘッダを指定する方法
- アタッチメントの追加方法
- 簡単な例

注釈　使用している SMTP サーバの要件を理解しておく必要があります。例えば、一部の SMTP サーバでは `Subject:` ヘッダを含める必要があります。同様に、一部の SMTP サーバでは任意の `From:` ヘッダを許可しない場合があります。

2.4.1 電子メール・メッセージの作成

電子メール・メッセージを作成するには、以下の操作を行います。

1. %Net.MailMessage のインスタンスを作成します。

   Tipヒント　%New() への引数として文字セットを指定できます。文字セットを指定した場合は、メッセージに Charset プロパティが設定されます。

   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。

2. インスタンスの To プロパティ、From プロパティ、および Subject プロパティを設定します。これらのプロパティは必須です。
   - To – このメッセージの送信先となる電子メール・アドレスのリスト。このプロパティは、標準の InterSystems IRIS リストです。これを使用するには、標準のリスト・メソッドである Insert()、GetAt()、RemoveAt()、Count()、および Clear() を使用します。
   - From – このメッセージの送信元の電子メール・アドレス。
   - Subject – メッセージの件名。
3. 必要に応じて、Date、Cc、Bcc およびその他のプロパティを設定します。詳細は、"基本的なヘッダ" を参照してください。
4. シングルパート・メッセージを作成する場合、"メッセージ・パートのコンテンツおよびヘッダの指定" で説明されているようにコンテンツおよびヘッダを指定します。
5. マルチパート・メッセージを作成する場合は、以下の操作を行います。
   a. IsMultiPart プロパティを 1 に設定します。
   b. MultiPartType プロパティを、`"related"`、`"alternative"`、または `"mixed"` のいずれかに設定します。
   c. メッセージに含まれる各パートに対して、%Net.MailMessagePart のインスタンスを作成します。次に、"メッセージ・パートのコンテンツおよびヘッダの指定" で説明されているようにコンテンツおよびヘッダを指定します。
   d. 親電子メール・メッセージに対して Parts プロパティを設定します。これは配列です。この配列に、各子メッセージ・パートを挿入します。

2.4.2 メッセージ・パートのコンテンツおよびヘッダの指定

1. メッセージがプレーン・テキストでない場合、以下のプロパティを設定して、作成するメッセージの種類を指定します。
   - これが HTML メッセージの場合は、IsHTML プロパティを 1 に設定します。
   - これがバイナリ・メッセージの場合は、IsBinary プロパティを 1 に設定します。
2. メッセージとそのヘッダの文字セットを指定する場合は、必要に応じて、Charset プロパティを設定します。

   重要　　メッセージのコンテンツを追加する前に文字セットを指定することが重要です。

   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. メッセージのコンテンツを追加します。
   - プレーン・テキストまたは HTML の場合は、TextData プロパティを使用します。これは %FileCharacterStream のインスタンスです。このストリームの TranslateTable プロパティを指定する必要はありません。電子メール・メッセージの文字セットを指定すると自動的に発生します。

     システムは、自動的に、文字エンコーディングを以前のステップで指定されたエンコーディングに変換します。
   - バイナリ・データの場合は、BinaryData プロパティを使用します。これは %FileBinaryStream のインスタンスです。

   %FileCharacterStream および %FileBinaryStream 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。

   Tipヒント　ストリームの Filename プロパティを指定するときは、ユーザが書き込み権限を持っているディレクトリを使用してください。

   これらのプロパティを操作するには、標準ストリーム・メソッドの Write()、WriteLine()、Read()、ReadLine()、Rewind()、MoveToEnd()、および Clear() を使用します。また、ストリームの Size プロパティを使用して、メッセージ・コンテンツのサイズを指定することもできます。

2.4.3 メッセージ・ヘッダの指定

前述したとおり、メッセージ自体およびメッセージの各パートの両方には、一連のヘッダがあります。

%Net.MailMessage クラスおよび %Net.MailMessagePart クラスは、最も一般的に使用されるヘッダに簡単にアクセスできるようにするプロパティを提供しますが、必要なヘッダを追加することができます。この節では、すべてのヘッダに関する情報を提供すると共に、カスタム・ヘッダの作成方法についても説明します。

注釈　　指定されたメッセージ・パートのヘッダは、そのパートの Charset プロパティで指定された文字セットで記述されます。

2.4.3.1 基本的なヘッダ

以下のプロパティ (%Net.MailMessage のみ) を設定し、メッセージ自体の最も一般的に使用されるヘッダを指定します。

- To ― (必須) このメッセージの送信先となる電子メール・アドレスのリスト。このプロパティは、標準の InterSystems IRIS リストです。これを使用するには、標準のリスト・メソッドである Insert()、GetAt()、RemoveAt()、Count()、および Clear() を使用します。
- From ― (必須) このメッセージの送信元の電子メール・アドレス。
- Date ― このメッセージの日付。

- Subject ― このメッセージの件名を含む文字列。
- Sender ― メッセージの実際の送信者。
- Cc ― このメッセージの送信先となるカーボン・コピー・アドレスのリスト。
- Bcc ― このメッセージの送信先となるブラインド・カーボン・コピー・アドレスのリスト。

注釈　使用しているSMTPサーバの要件を理解しておく必要があります。例えば、一部のSMTPサーバでは`Subject:`ヘッダを含める必要があります。同様に、一部のSMTPサーバでは任意の`From:`ヘッダを許可しない場合があります。

2.4.3.2 Content-Type ヘッダ

メッセージを送信する場合、メッセージおよび各メッセージ・パートの`Content-Type`ヘッダは、自動的に以下のように設定されます。

- メッセージがプレーン・テキストの場合 (IsHTML が 0 で IsBinary が 0 の場合)、`Content-Type`ヘッダは`"text/plain"`に設定されます。
- メッセージがHTMLの場合 (IsHTML が 1 で IsBinary が 0 の場合)、`Content-Type`ヘッダは`"text/html"`に設定されます。
- メッセージがバイナリの場合 (IsBinary が 1 の場合)、`Content-Type`ヘッダは`"application/octet-stream"`に設定されます。
- メッセージがマルチパートの場合、`Content-Type`ヘッダはMultiPartTypeプロパティの値に対して適切に設定されます。

%Net.MailMessage および %Net.MailMessagePart は共に、ContentTypeプロパティを提供します。このプロパティは、`Content-Type`ヘッダへのアクセスを提供します。

2.4.3.3 Content-Transfer-Encoding ヘッダ

%Net.MailMessage および %Net.MailMessagePart は共に、ContentTransferEncodingプロパティを提供します。このプロパティは、メッセージまたはメッセージ・パートの`Content-Transfer-Encoding`ヘッダを簡単に指定する方法を提供します。

このプロパティは、`"base64"`、`"quoted-printable"`、`"7bit"`、`"8bit"`のいずれかです。

デフォルトは以下のとおりです。

- バイナリのメッセージまたはメッセージ・パートには、`"base64"`を使用します。
- テキストのメッセージまたはメッセージ・パートには、`"quoted-printable"`を使用します。

2.4.3.4 カスタム・ヘッダ

%Net.MailMessage および %Net.MailMessagePart の両方を使用して、Headersプロパティにアクセスすることにより、カスタム・ヘッダを設定または取得できます。このプロパティは、以下の構造を持つ配列です。

配列キー	配列の値
`"Priority"`など、ヘッダの名前	ヘッダの値

このプロパティを使用して、優先順位などの追加のヘッダを含めます。以下に例を示します。

```
Do msg.Headers.SetAt("Urgent","Priority")
```

2.4.4 アタッチメントの追加

電子メールのメッセージまたはメッセージ・パートにアタッチメントを追加できます。そのためには、%Net.MailMessagePart または %Net.MailMessage の以下のメソッドを使用します。

これらの各メソッドは、元のメッセージ (またはメッセージ・パート) の Parts 配列にアタッチメントを追加し、自動的に IsMultiPart プロパティを 1 に設定します。

AttachFile

```
method AttachFile(Dir As %String,
                  File As %String,
                  isBinary As %Boolean = 1,
                  charset As %String = "",
                  ByRef count As %Integer) as %Status
```

指定されたファイルを電子メール・メッセージに添付します。デフォルトでは、ファイルはバイナリ・アタッチメントとして送信されますが、代わりにテキストとして指定することができます。また、このファイルがテキストの場合、ファイルが使用する文字セットを指定することもできます。(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。)

具体的には、このメソッドは %Net.MailMessagePart のインスタンスを作成し、BinaryData または TextData プロパティ内のファイルのコンテンツを適切に配置し、Charset プロパティおよび TextData.TranslateTable プロパティを必要に応じて設定します。このメソッドは、Parts 配列内のこの新しいメッセージ・パートの位置を示す整数を、参照渡しで返します。

このメソッドは、メッセージまたはメッセージ・パートの Dir プロパティおよび FileName プロパティも設定します。

AttachNewMessage

```
method AttachNewMessage() as %Net.MailMessagePart
```

%Net.MailMessage の新しいインスタンスを作成し、メッセージに追加して、新しく修正された親のメッセージまたはメッセージ・パートを返します。

AttachEmail

```
method AttachEmail(mailmsg As %Net.MailMessage)
```

電子メール・メッセージ (%Net.MailMessage のインスタンス) を指定すると、このメソッドは、それをメッセージに追加します。

注釈　このメソッドは、ContentType を "message/rfc822" に設定します。この場合、他のアタッチメントを追加することはできません。

例については、%Net.MailMessagePart のクラス参照を参照してください。

2.4.5 例

以下に例を示します。

Class Member

```
ClassMethod CreateTextMessage() As %Net.MailMessage
{
    Set msg = ##class(%Net.MailMessage).%New()
    set msg.From = "test@test.com"
    Do msg.To.Insert("xxx@xxx.com")
    Do msg.Cc.Insert("yyy@yyy.com")
    Do msg.Bcc.Insert("zzz@zzz.com")
    Set msg.Subject="subject line here"
    Set msg.IsBinary=0
    Set msg.IsHTML=0
    Do msg.TextData.Write("This is the message.")

    Quit msg
}
```

2.5 例

この節では、文字列コンテンツを含むシングルパートまたはマルチパートの電子メール・メッセージを送信する例を示します。プロダクション・システムでは、文字列の代わりにストリームを使用する場合もあります。ただし、この例では文字列を使用するので、プロダクション・テスト・ページを使用して簡単にテストできます。

まず、シングルパート・メッセージの要求メッセージ・クラスは、以下のようになります。

Class Definition

```
Class EEMA.SimpleMsg Extends Ens.Request
{

Property Subject As %String(MAXLEN = 100);

Property Message As %String(MAXLEN = 5000);

}
```

次は、マルチパート・メッセージの要求メッセージ・クラスです。このクラスには、3 パートのメッセージを含めることができます。

Class Definition

```
Class EEMA.MultipartMsg Extends Ens.Request
{

Property Subject As %String(MAXLEN = 100);

Property Part1 As %String(MAXLEN = 5000);

Property Part2 As %String(MAXLEN = 5000);

Property Part3 As %String(MAXLEN = 5000);

}
```

以下のビジネス・オペレーションは、これらのメッセージのいずれかを受け取り、適切な電子メール・メッセージを作成して、構成された SMTP サーバを介して送信します。

Class Definition

```
Class EEMA.EmailOperation Extends Ens.BusinessOperation
{

Parameter ADAPTER = "EnsLib.EMail.OutboundAdapter";

Parameter INVOCATION = "Queue";

Method SendSimpleMessage(pRequest As EEMA.SimpleMsg,
Output pResponse As Ens.Response) As %Status
{
```

```
    Set email=##class(%Net.MailMessage).%New()

    //Get info from pReq
    Do email.TextData.Write(pRequest.Message)
    Set email.Subject=pRequest.Subject

    //simple case--do not check for failed recipients
    Set tSc=..Adapter.SendMail(email)

    //send an empty response message after message is sent
    Set pResponse=##class(Ens.Response).%New()

    ;Quit tSc
}

Method SendMultipartEmailMessage(pRequest As EEMA.MultipartMsg,
Output pResponse As Ens.Response) As %Status
{
    Set part1=##class(%Net.MailMessage).%New()
    Do part1.TextData.Write(pRequest.Part1)
    Set part2=##class(%Net.MailMessage).%New()
    Do part2.TextData.Write(pRequest.Part2)
    Set part3=##class(%Net.MailMessage).%New()
    Do part3.TextData.Write(pRequest.Part3)

    Set email=##class(%Net.MailMessage).%New()
    Set email.Subject=pRequest.Subject
    Set email.IsMultiPart=1
    Do email.Parts.SetAt(part1,1)
    Do email.Parts.SetAt(part2,2)
    Do email.Parts.SetAt(part3,3)

    //simple case--do not check for failed recipients
    Set tSc=..Adapter.SendMail(email)

    //send an empty response message after message is sent
    Set pResponse=##class(Ens.Response).%New()

    Quit tSc
}

XData MessageMap
{
<MapItems>
    <MapItem MessageType="EEMA.SimpleMsg">
        <Method>SendSimpleMessage</Method>
    </MapItem>
    <MapItem MessageType="EEMA.MultipartMsg">
        <Method>SendMultipartEmailMessage</Method>
    </MapItem>
</MapItems>
}

}
```

2.6 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. ビジネス・オペレーションを有効化します。
3. SMTP メール・サーバとそれにアクセスするために必要な認証情報を指定します。
4. 任意で、電子メール・メッセージの追加のアドレスを指定します。
5. プロダクションを実行します。

2.6.1 SMTP サーバへのログイン方法の指定

使用する SMTP サーバおよびそれに関連するログイン資格情報を指定するには、EnsLib.EMail.OutboundAdapter の以下の設定に対して値を指定します。

- SMTPサーバ
- SMTPポート
- Credentials
- SSL構成

例えば、以下のような値を使用できます。

設定	値
SMTPサーバ	`smtp.hotpop.com`
SMTPポート	25
Credentials	`hotpop`

この例にある `hotpop` は、ユーザ名 `isctest@hotpop.com` とこれに対応するパスワードから成るプロダクション認証情報 の ID です。

2.6.2 追加の電子メール・アドレスの指定

EnsLib.EMail.OutboundAdapter の以下の設定を使用して、このアダプタから送信される電子メール・メッセージの電子メール・アドレスを指定できます。

- Recipient
- Cc
- From

ここに記載されていない設定については、"プロダクションの構成" を参照してください。

電子メール・アダプタの設定

ここでは、電子メール・アダプタの設定に関する参照情報を提供します。

電子メール受信アダプタに関する設定

電子メール受信アダプタの設定に関する参照情報を提供します。

概要

受信電子メール・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	[POP3サーバ]、[POP3ポート]、[認証情報]、[呼び出し間隔]
接続設定	[SSL構成]、[SSL チェック・サーバ ID]
OAuth2	[OAuth2 認可プロパティ]、[OAuth2 認可ワークフローロール]、[OAuth2 コールバックハンドラ]、[OAuth2 クライアントアプリケーション名]、[OAuth2 グラントタイプ]、[OAuth2 スコープ]
OAuth2 グラント固有	[OAuth2 JWT Subject]
追加設定	[Match From]、[Match To]、[Match Subject]

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。

呼び出し間隔

アダプタが新しい電子メールをもう一度チェックするまで待機する秒数です。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。

デフォルト Call Interval は 5 秒です。最小値は 0.1 秒です。

Credentials

指定された POP3 サーバに存在する有効なメールボックスの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。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。

Match From

セミコロン (;) で区切られた、受信電子メール・メッセージの `From:` フィールド内で検索する文字列のリストです。この設定が NULL の場合、取得するメッセージの選択時に `From:` フィールドは無視されます。詳細は、"取得するメッセージの指定" を参照してください。

Match Subject

セミコロン (;) で区切られた、電子メール・メッセージの `Subject:` フィールド内で検索する文字列のリストです。この設定が NULL の場合、取得するメッセージの選択時に `Subject:` フィールドは無視されます。詳細は、"取得するメッセージの指定" を参照してください。

Match To

セミコロン (;) で区切られた、電子メール・メッセージの `To:` フィールド内で検索する文字列のリストです。この設定が NULL の場合、取得するメッセージの選択時に `To:` フィールドは無視されます。詳細は、"取得するメッセージの指定" を参照してください。

OAuth2 認可プロパティ

付与フローの承認プロセスの追加プロパティ。

`Access_type=offline,prompt=consent` のような、コンマ区切りのキーと値のペアです。

OAuth2 認可ワークフローロール

付与タイプ・フローに応じて、承認要求の送信先となるワークフロー・ロール。ワークフロー・ロールの詳細は、"ワークフロー・ロールとワークフロー・ユーザ" を参照してください。

OAuth2 コールバックハンドラ

[OAuth2 クライアントアプリケーション名] が指定されている場合、このクラスを使用して、アクセス・トークンの取得を処理します。既定値は `Ens.Util.XOAuth2.Handler` で、アクセス・トークンの取得をカスタマイズするため、サブクラス化することもできます。

OAuth2 クライアントアプリケーション名

使用する OAuth2 クライアント構成アプリケーション名。これは、InterSystems IRIS® OAuth 2.0 設定で作成されるクライアント構成です。指定されている場合、サブクラスは、これを使用して OAuth 2.0 が使用されることを示すことができ、承認およびアクセス・トークンの取得プロセスではその名前が使用されます。

OAuth2 グラントタイプ

これは、OAuth2 コールバック・ハンドラが従う付与タイプ・フローです。付与フロー・タイプに従うことができるかどうかは、OAuth2 コールバック・ハンドラの実装、InterSystems IRIS® および外部 OAuth2 サーバでの付与タイプ・フローのサポートによって異なります。

OAuth2 JWT Subject

これは、JWT 承認の付与タイプ・フローを使用する場合の JWT サブジェクトです。JWT の詳細は、"Introduction to JSON Web Tokens" を参照してください。

OAuth2 スコープ

これは、承認要求に含まれる範囲を指定します。指定されない場合は、[OAuth2 クライアントアプリケーション名] で指定される既定の範囲が使用されます。

POP3ポート

メールを取得する POP3 電子メール・サーバの TCP ポート。デフォルト値は 110 です。

POP3サーバ

メールを取得する POP3 電子メール・サーバのアドレス。

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。この文字列は [SSL構成] の設定の値として使用します。

SSL チェック・サーバ ID

TLS を使用して POP3 サーバまたは SMTP サーバに接続する場合、証明書のサーバ名が、サーバへの接続に使用される DNS 名と一致している必要があります。この一致は、RFC 2818 のセクション 3.1 に規定されているルールに基づきます。

電子メール送信アダプタに関する設定

送信アダプタの設定に関する参照情報を提供します。

概要

送信電子メール・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	[SMTPサーバ]、[SMTPポート]、[認証情報]
接続設定	[SSL構成]、[SSL チェック・サーバ ID]
OAuth2	[OAuth2 認可プロパティ]、[OAuth2 認可ワークフローロール]、[OAuth2 コールバックハンドラ]、[OAuth2 クライアントアプリケーション名]、[OAuth2 グラントタイプ]、[OAuth2 スコープ]
OAuth2 グラント固有	OAuth2 JWT Subject
追加設定	Recipient、Cc、Bcc、From、ContinueIfInvalidRecipient

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。

Bcc

送信される各メール・メッセージの `Bcc:` リストに追加する、電子メール・アドレスのカンマ区切りのリストを指定します。

Cc

送信される各メール・メッセージの `Cc:` リストに追加する、電子メール・アドレスのカンマ区切りのリストを指定します。

ContinueIfInvalidRecipient

選択すると、1 人以上の受信者のアドレスが無効なときに、アダプタはメッセージの送信を続行します。

Credentials

指定されたサーバ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。

From

メール・メッセージに入力される、デフォルトの `From:` アドレス。ビジネス・オペレーション実装コードによって上書きできます。

OAuth2 認可プロパティ

付与フローの承認プロセスの追加プロパティ。

`Access_type=offline,prompt=consent` のような、コンマ区切りのキーと値のペアです。

OAuth2 認可ワークフローロール

付与タイプ・フローに応じて、承認要求の送信先となるワークフロー・ロール。ワークフロー・ロールの詳細は、"ワークフロー・ロールとワークフロー・ユーザ" を参照してください。

OAuth2 コールバックハンドラ

[OAuth2 クライアントアプリケーション名] が指定されている場合、このクラスを使用して、アクセス・トークンの取得を処理します。既定値は `Ens.Util.XOAuth2.Handler` で、アクセス・トークンの取得をカスタマイズするため、サブクラス化することもできます。

OAuth2 クライアントアプリケーション名

使用する OAuth2 クライアント構成アプリケーション名。これは、InterSystems IRIS® OAuth 2.0 設定で作成されるクライアント構成です。指定されている場合、サブクラスは、これを使用して OAuth 2.0 が使用されることを示すことができ、承認およびアクセス・トークンの取得プロセスではその名前が使用されます。

OAuth2 グラントタイプ

これは、OAuth2 コールバック・ハンドラが従う付与タイプ・フローです。付与フロー・タイプに従うことができるかどうかは、OAuth2 コールバック・ハンドラの実装、InterSystems IRIS® および外部 OAuth2 サーバでの付与タイプ・フローのサポートによって異なります。

OAuth2 JWT Subject

これは、JWT 承認の付与タイプ・フローを使用する場合の JWT サブジェクトです。JWT の詳細は、″Introduction to JSON Web Tokens″ を参照してください。

OAuth2 スコープ

これは、承認要求に含まれる範囲を指定します。指定されない場合は、[OAuth2 クライアントアプリケーション名] で指定される既定の範囲が使用されます。

Recipient

送信される各メール・メッセージの `To:` リストに追加する、電子メール・アドレスのカンマ区切りのリストを指定します。

SMTPポート

メールの送信先となる SMTP サーバのポート。デフォルト値は 25 です。

SMTPサーバ

メールの送信先となる SMTP サーバの IP アドレス (注意 : 接続およびメール送信のタイムアウトは、10 分以上に設定することができます)。

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 構成の編集] フォームの最初のフィールドは [構成名] です。この文字列は [SSL構成] の設定の値として使用します。

通常、この設定に値を指定すると、送信電子メールは既定のポート 465 のソケットを開き、SMTP over TLS を使用します。ただし、[SSL 構成] 設定では、RFC3207 標準で説明されているサーバとのやり取りもサポートされています。具体的には、システムが標準の TCP ソケットを開いてから、同じポートで TLS 接続に切り替えることで、接続を開始できるように、設定できます。システムでは、`STARTTLS` コマンドを発行することによってこの切り替えを実行します。この特別なタイプの接続を有効にするには、[SSL 構成] の値の末尾にアスタリスクを付けます (例 : `MySSLItem*`)。この場合の既定の SMTP ポートは 25 です。

さらに詳しい情報は、"Class Reference" の EnsLib.EMail.OutboundAdapter のエントリにある SSLConfig プロパティの説明を参照してください。

SSL チェック・サーバ ID

TLS を使用して POP3 サーバまたは SMTP サーバに接続する場合、証明書のサーバ名が、サーバへの接続に使用される DNS 名と一致している必要があります。この一致は、RFC 2818 のセクション 3.1 に規定されているルールに基づきます。