Receta electrónica

Manual Técnico

[21/10/2009]

Introducción

Receta Electrónica se trata de un proyecto en constante evolución con el objetivo de ajustarse a las necesidades de todos los actores implicados en el proyecto.

Esta situación genera nuevas versiones del software. En este apartado se listan todas las versiones que se han generado históricamente en Receta Electrónica, incluyendo una breve explicación de las mejoras y correcciones que aporta el software con respecto a la versión inmediatamente anterior.

Cada una de las versiones del software que se indican a continuación son el conjunto de dll's que se encargan de permitir la comunicación entre la farmacia y los servidores centrales de Receta Electrónica y facilitar así la obtención de prescripciones y haciendo posible la emisión de dispensaciones. Además, dichas dll's hay que integrarlas dentro de las aplicaciones de los distribuidores de farmacia para su correcto funcionamiento.

Versiones

- Versión 2.1
  - Permite recuperar las prescripciones de tarjetas de 4kb
  - Están preparadas para realizar hoja de cupones
  - Preparadas para visado electrónico

- Versión 2.0
  - Permite leer prescripciones de la banda magnética de la tarjeta.
  - Borra el certificado de usuario del almacén de certificados una vez se han leído los datos del certificado de la tarjeta.
- Versión 1.6
  - Soluciona el problema de obtener más de 21 prescripciones.
  - Soluciona un problema de sincronismo a la hora de dispensar.
- Versión 1.5
  - Permite conectarse al servidor atravesando un proxy.
- Versión 1.4
  - Soluciona un error en la devolución del campo visado en la recuperación de prescripciones pendientes.
- Versión 1.3
  - El envío de mensajes mediante SOAP se realiza de forma asíncrona.
  - Permite leer certificados digitales almacenados en el equipo.

- Incluye un archivo para la configuración de pruebas en la farmacia.
- Permite realizar dispensaciones de prescripciones no registradas.

Este manual además, pretende exponer la parte técnica de las dll's que se integran en las aplicaciones de los distribuidores de farmacia enumerando a modo de índice los puntos que se explican en este manual:

1. Introducción
2. Simple Object Access Protocol (S.O.A.P.)
3. Equipos de desarrollo y prueba
4. Descripción del Interface Farmacia Receta Electrónica
5. Funcionamiento Interno
6. Como utilizar los componentes del Interface
7. Arquitectura

SIMPLE OBJECT ACCESS PROTOCOL (S.O.A.P.)

Versiones

- Versión 2.0 Service Pack 2.
- Versión 3.0 (No tratada).

Requerimientos

Los objetos para el servidor SOAP requieren los siguientes sistemas operativos de Microsoft:

- ✓ Windows 2000 (Todas las versiones)
- ✓ Windows NT 4 Service Pack 6

Los objetos para el cliente SOAP se ejecutan en los siguientes sistemas operativos de Microsoft:

- ✓ Microsoft Windows 98
- ✓ Microsoft Windows Me
- ✓ Microsoft Windows NT 4 Service Pack 6
- ✓ Microsoft Windows 2000 Service Pack 1

Notas

Es necesario tener instalado Microsoft Internet Explorer 5.0 o superior.

Es necesario tener instalado Microsoft Visual Basic Runtime para SOAP Messaging Objects (SMO) en el cliente corriendo bajo Windows NT 4 Service Pack 6, Windows 98, Windows Me.

La instalación del SOAP Toolkit 2.0 instala también el Microsoft XML Parser (MSXML 3.0).

Equipos de desarrollo y prueba

Para el desarrollo de los componentes de la Interface Farmacia Receta Electrónica se ha utilizado Microsoft Visual Basic 6.0 Service Pack 5, corriendo sobre un equipo con Microsoft Windows 2000 professional Service Pack3 en castellano y Microsoft Internet Explorer 6.0.2800.1106.

Para las pruebas de simulación real se utilizó un equipo con sistema operativo Microsoft Windows 98 SE en castellano y Microsoft Internet Explorer 5.00.2614.3500.

Descripción del Interface Farmacia Receta Electrónica

El interface se encarga de componer mensajes SOAP/XML para conectarse con el servidor y devolver también mediante SOAP/XML la respuesta que generó el servicio.

Actualmente se envían dos tipos de mensajes, uno para recuperar las prescripciones pendientes de un paciente y el otro para realizar la dispensación de una receta.

Para recuperar las prescripciones pendientes es necesario rellenar los siguientes campos de la clase CArgument (De **Initiator.dll**):

.Usuario
.Password

El campo TIS también es obligatorio, se presupone que se leerá de la tarjeta, pero mientras para pruebas se lee desde el archivo de configuración Interface.ini.

El mensaje devuelto contiene las prescripciones pendientes del paciente e información de comprobación por si se produce un error en el proceso, todos estos valores son almacenados en un objeto de la clase CPrescripciones.

Para realizar la dispensación de una receta prescrita es necesario rellenar los siguientes campos de la clase CPeticionDispensacion (Del **Interface.dll**):

.IdUsuario
.ClaveUsuario
.Idioma
.IdPrescripcion
.FechaDispensacion

.IdFarmacia
.dFarmaceutico
.EnvasesDispensados
.Importe
.ImporteAportacion
.CodigoDispensado
.IdProvincia

Si por el contrario se quiere realizar la dispensación de una receta que no ha sido prescrita previamente se deberán rellenar los campos que aparecen arriba además de estos otros, y establecer a blanco el campo IdPrescripcion ya que al no estar la receta prescrita no tiene código:

.CodigoColegiadoPrescriptor
.FechaPrescripcion
.TIS
.RegimenReceta
.FormulaMagistral
.TipoPrescripcion

El mensaje devuelto contiene los avisos y los errores que ha generado el servicio web, para poder determinar si se realizó la dispensación correctamente, los valores son almacenados en un objeto de la clase CRespuestaPeticion.

El envío y recepción de mensajes mediante SOAP/XML es asíncrono.

El motor de la Interface Farmacia-Receta Electrónica consta de los siguientes archivos:

1. Interface.dll, librería para la dispensación
2. Initiator.dll, librería para recuperar prescripciones pendientes
3. Executor.exe, lanzador de procesos asíncronos
4. Defrespo.xml, respuesta xml devuelta por defecto
5. Interface.ini, archivo de configuración del interface con los siguientes campos:

Archivo: Interface.ini Este archivo contiene toda la información de inicialización del motor del interface.	
[Configuracion]	
Servicio	URL del WSDL del Servicio Web de Dispensaciones
URL del servicio al que se utiliza para la dispensación y recuperación de prescripciones pendientes	
EsquemaXML	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
Información de esquemas utilizados por XML para enviar y recibir mensajes mediante SOAP	
GuardarPeticionSOAP	0
Obsoleto, no utilizar, debe ser siempre 0	
GenerarLog	1

Su valor puede ser 0 o 1 en este último caso genera un archivo con la información que genera el motor	
GenerarTrace	0
Su valor puede ser 0 o 1 en este último caso genera un archivo con las trazas de las funciones por las que pasa	
TamanioMaximoLog	2097152
Valor máximo en bytes del archivo de log, si se supera este tamaño se elimina el archivo de forma automática	
CodigoDispensacionCorrecta	P70AvisoDispensacionCorrecta
Código interno que indica que la dispensación se realizó correctamente	
[MensajeSOAP]	
Enviados	X
Número de mensajes SOAP enviados	
Recibidos	X
Número de mensajes SOAP recibidos	
[Proxy]	
UtilizarProxy	0
Su valor puede ser 0 o 1 en este último caso se utilizaran los valores del proxy	
Servidor	xxxxxxx
El host del servidor del proxy	
Puerto	xxxxxx
El puerto del proxy	
UsarProxy	0
Su valor puede ser 0 o 1 en este último caso se utilizaran los valores del proxy	
Usuario	XXXXXXX
El usuario para poder utilizar el Proxy	
Password	XXXXXXX
El password para poder utilizar el Proxy	
TimeOut	3000
El tiempo de espera en caso de no respuesta por parte del servidor	
[Asincrono]	
PeticionTimeOut	30
Tiempo de espera para recibir la respuesta SOAP/XML del servicio Web	
IntervaloComprobacion	2

Intervalo de comprobación de la existencia del archivo de respuesta en la carpeta Recibidos	
AutoRespuesta	0
Obsoleto, no utiliizar	
EliminarXMLPeticion	0
Su valor puede ser 0 o 1 en este último caso elimina de la carpeta Enviados el archivo XML con la petición SOAP	
EliminarXMLRespuesta	0
Su valor puede ser 0 o 1 en este último caso elimina de la carpeta Recibidos el archivo XML con la respuesta SOAP	
[Tarjeta]	
InsercionTarjetaTimeOut	50
Máximo tiempo de espera para leer el certificado de la la tarjeta	
IntervaloComprobacion	2
Intervalo de comprobación de datos	
QuitarCeros	0
Su valor puede ser 0 o 1 en este último caso los ceros a ala izquierda son eliminados	
CertIssuerSimpleName	Alias de la tarjeta de 2kb
Indica que cuando se lee la tarjeta solo se considerarán las que el CertIssuerName contenga el texto especificado	
CertIssuerSimpleName4	Alias de la tarjeta de 4kb
Indica que cuando se lee la tarjeta solo se considerarán las que el CertIssuerName4 contenga el texto especificado	
IdentificadorTIS	TIS
Token para identificar el TIS dentro de los valores del certificado	
GrupoCertificado	dnQualifier
Grupo del certificado	
[Seguimiento]	
Auditar	1
Su valor puede ser 0 o 1 en este último caso guarda en la carpeta Auditoria información relativa al envío y recepción de mensajes SOAP	
ArchivoDiario	1
Su valor puede ser 0 o 1 en este último caso se genera un archivo diario en la carpeta Auditoria, en otro caso se utiliza siempre el mismo archivo por defecto	
[RespuestaDefecto]	
NumAvisos	1

Debe ser siempre 1	
CodigoAviso	AvisoDefecto
Código del aviso	
TextoAviso	Aviso leído desde el archivo.ini
Descripción del aviso	
NumMotivosRechazo	1
Debe ser siempre 1	
CodigoMotivosRechazo	ErrorDefecto
Código del rechazo	
TextoMotivosRechazo	Motivo Rechazo leído desde el archivo.ini
Descripción del rechazo	
ResultResp	1
En caso de que no exista defrespo.xml se usarán las siguientes variables	
DescResp	Motivo rechazo leido desde el archivo.ini
Descripción de la respuesta	
CodResp	99999
Código de la respuesta	
TipResp	1
Tipo de respuesta de error	
CodDisp	99999
Codigo Dispensación	
[Initiator]	
GuardarPeticionSOAP	0
Su valor puede ser 0 o 1 en este último caso guarda en disco el archivo XML/SOAP generado por Initiator.dll para solicitar las prescripciones pendientes	
[Pruebas]	
TIS	10
TIS para pruebas por defecto, se utiliza para enviar la petición de prescripciones pendientes de dispensar	

Nota: Como el proceso de dispensación es asíncrono, puede ocurrir que se pase el tiempo de espera y no se haya recibido respuesta por parte del servidor. Si esto

ocurre se le proporciona al cliente de Interface.dll una respuesta por defecto que es la leída del archivo defrespo.xml. Si por algún motivo este archivo no existiese o se hubiera eliminado se le enviará la respuesta que se leerá desde el archivo de inicialización inteface.ini.

Cuando llegue la respuesta del servidor puede que el cliente ya haya recibido una respuesta por defecto, así que el mensaje recibido se guardará en la carpeta Retrasados. El aspecto que tendría la respuesta recibida para una dispensación sería el siguiente:

```
- <RespuestaPeticion xmlns:xsd="http://www.w3.org/1999/XMLSchema"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance">
    <NMotivosRechazo>1</NMotivosRechazo>

    <NAvisos>1</NAvisos>

  - <textoAviso xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
      xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:xsd="http://www.w3.org/1999/XMLSchema" xsi:type="SOAP-
      ENC:Array" SOAP-ENC:arrayType="xsd:string[1]">

      <SOAP-ENC:string>Aviso leido desde el archivo defrespo.xml</SOAP-
        ENC:string>

    </textoAviso>

  - <textoMotivosRechazo xmlns:xsi="http://www.w3.org/1999/XMLSchema-
      instance" xmlns:SOAP-
      ENC="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:xsd="http://www.w3.org/1999/XMLSchema" xsi:type="SOAP-
      ENC:Array" SOAP-ENC:arrayType="xsd:string[1]">

      <SOAP-ENC:string>MOtivo Rechazo leido desde el archivo
        defrespo.xml</SOAP-ENC:string>

    </textoMotivosRechazo>

  - <codigoMotivosRechazo xmlns:xsi="http://www.w3.org/1999/XMLSchema-
      instance" xmlns:SOAP-
      ENC="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:xsd="http://www.w3.org/1999/XMLSchema" xsi:type="SOAP-
      ENC:Array" SOAP-ENC:arrayType="xsd:string[1]">

      <SOAP-ENC:string>errorDefecto</SOAP-ENC:string>

    </codigoMotivosRechazo>

  - <codigoAviso xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
      xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:xsd="http://www.w3.org/1999/XMLSchema" xsi:type="SOAP-
      ENC:Array" SOAP-ENC:arrayType="xsd:string[1]">
```

```
<SOAP-ENC:string>AvisoDefecto</SOAP-ENC:string>
</codigoAviso>
</RespuestaPeticion>
```

En la carpeta Auditoria se almacena toda la información referente a la fecha de envío de la petición, si se recibió respuesta del servidor y el texto de la respuesta que se le proporcionó al cliente de interface.dll. El aspecto del archivo de auditoria es el siguiente:

```
[2800000000245]
FECHA_ENVIO=10:53:20
PETICION_GENERADA=Si
MENSAJE_ENVIADO=Si
MENSAJE_RECIBIDO=Si
RESPUESTA_GENERADA=srv
TEXTO_RESPUESTA=Se realizo la dispensación

[2800000000222]
FECHA_ENVIO=10:53:25
PETICION_GENERADA=Si
MENSAJE_ENVIADO=Si
MENSAJE_RECIBIDO=Si
RESPUESTA_GENERADA=srv
TEXTO_RESPUESTA=No se puede interpretar la respuesta recibida
```

NOTA: Los valores que puede contener el campo RESPUESTA GENERADA son los siguientes:

srv, Indica que la respuesta la proporcionó el servicio web
xml, Indica que la respuesta la proporcionó el archivo defrespo.xml
ini, Indica que la respuesta la proporcionó el archivo interface.ini
int, indica que la respuesta la proporcionó interface.dll internamente
sin valor, Indica que no se ha producido respuesta

Funcionamiento Interno

El proceso completo de funcionamiento del Interface Farmacia-Receta Electrónica consta de varias etapas :

- Realizar una llamada al método GetCardData de la clase CCard de la librería de enlace dinámico initiator.dll, para verificar si ha introducido la tarjeta chip y recuperar los datos.

Si estos son recuperados correctamente se deberá llamar al método RecuperarPrescripciones de la clase CInitiator de la librería de enlace dinámico initiator.dll que automáticamente generará un mensaje SOAP/XML que se enviará al servicio web para que devuelva todas las prescripciones pendientes de un paciente.

Si ocurriese algún error tanto accediendo al lector de tarjetas como en el propio proceso de recuperación de prescripciones se informará a la aplicación cliente mediante la lectura de la propiedad *Errores* que contiene los objetos de las clases *CInitiator y CInterface*.

Si el proceso terminó sin incidencias se pueden recuperar los datos de las prescripciones del objeto de la clase CPrescripciones que contiene una colección de objetos CPrescripcion con información relativa a una prescripción individual.

- La aplicación de la farmacia deberá gestionar la lista de prescripciones pendientes del paciente como considere oportuno.

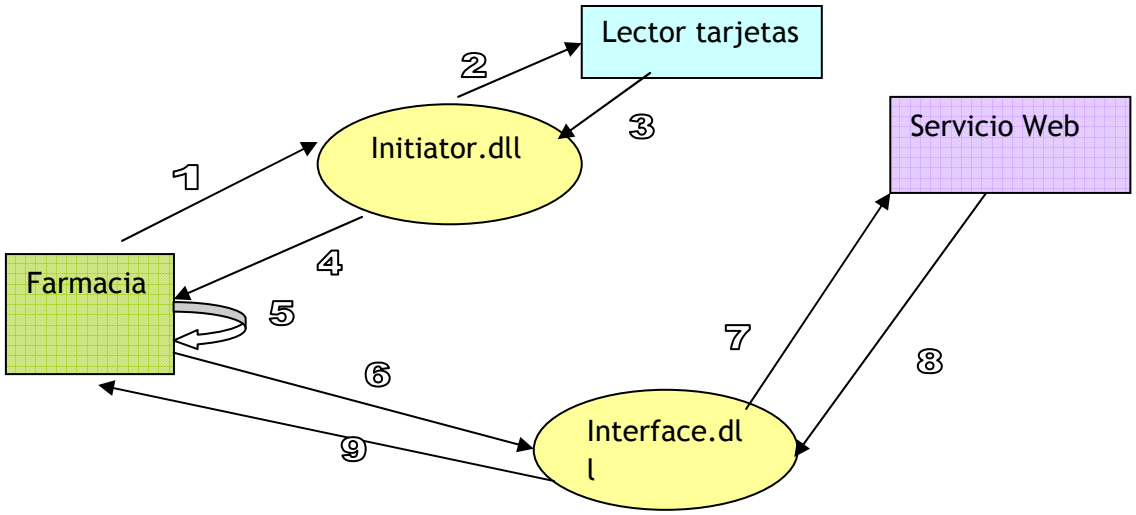
- La dispensación se deberá realizar llamando al método *SolicitudDispensacion* de la clase *CInterface* de la librería de enlace dinámico Interface.dll. Para ello hay que rellenar los datos necesarios para la dispensación como el identificador de la prescripción, el importe, etc..,

  Se generará un mensaje SOAP/XML que se enviará al servicio web para que realice la dispensación.

  Si ocurriese algún error en el proceso se podría consultar la propiedad Errores de la clase *CInterface para* recuperar el error ocurrido.

  Si el proceso terminó sin incidencias se pueden recuperar la información que devolvió el servicio consultando el objeto de la clase *CRespuestaPeticion*.

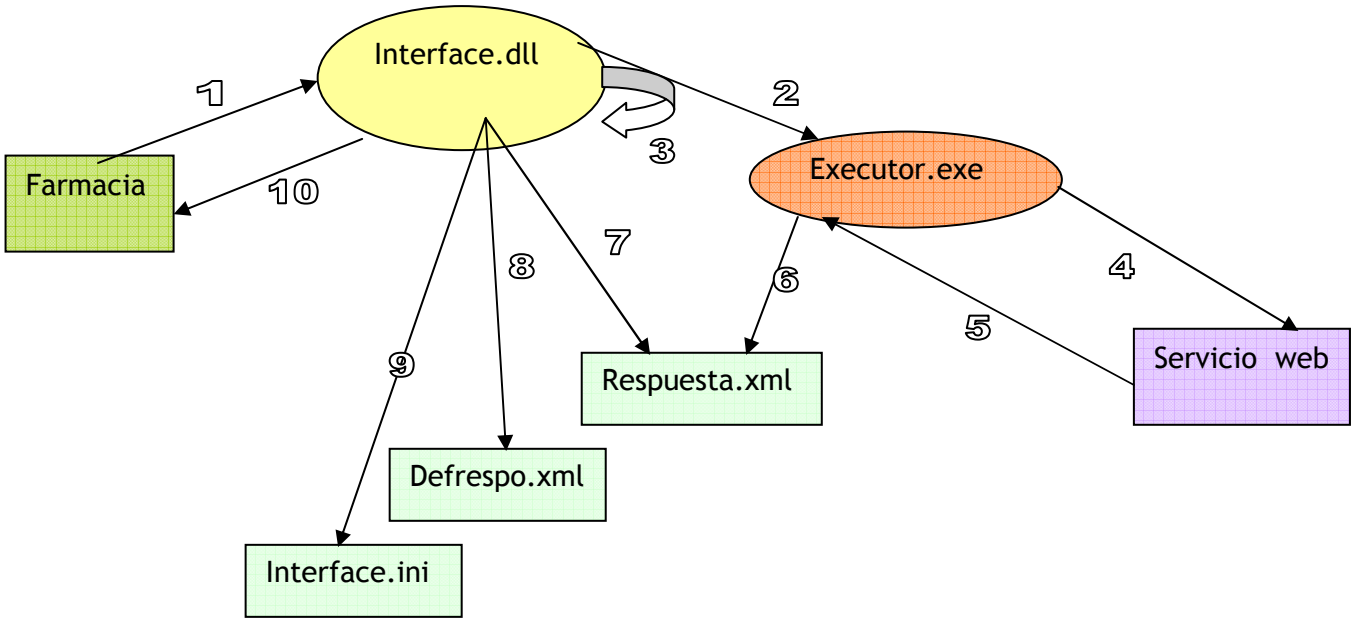
El siguiente gráfico ilustra el proceso completo a un nivel alto de abstracción:



Proceso completo e iteractivo entre la farmacia los componentes el lector de tarjetas y los servicios web.

1. Solicitar a Initiator la recuperación de los datos de la tarjeta chip
2. Initiator.dll se comunica con el lector de tarjetas para recuperar la información
3. El lector devuelve la información a Initiator.dll
4. Initiator.dll devuelve a la farmacia los datos de la tarjeta o en su defecto los errores producidos
5. La farmacia gestiona los datos recibidos
6. La farmacia inicia una solicitud de dispensación llamando a Interface.dll
7. Interface.dll se comunica con el servicio web para realizar la dispensación
8. El servicio devuelve la información a Interface.dll
9. Interface.dll devuelve la información a la farmacia

El siguiente gráfico ilustra el proceso asíncrono de la dispensación:



Proceso asíncrono de la dispensación.

1. Solicitud de dispensación de una prescripción pendiente
2. Creación de una nueva instancia de Executor.exe para iniciar proceso asíncrono
3. Bucle que comprueba si ha llegado la respuesta del servicio web
4. Envío al servicio web de los datos para realizar la dispensación
5. El servicio web devuelve información sobre la dispensación
6. Executor.exe guarda en disco la respuesta del servicio web
7. Interface.dll comprueba si existe la respuesta del servicio web
8. SI no hay respuesta del servicio se lee la respuesta por defecto del archivo defrespo.xml
9. SI no existe el archivo defrespo.xml se lee los datos de la respuesta por defecto del archivo de inicialización
10. Interface.dll devuelve la respuesta a la farmacia

Mensajes SOAP/XML

Los mensajes SOAP/XML enviados y recibidos son totalmente transparentes para el usuario final ya que estos se limitan al manejo directo de las clases.

Mensaje soap enviado para la recuperación de prescripciones pendientes

```
- <p70PeticionConsultaPNDPorTIS>
    <usuario>xxxxx</usuario>
    <passwd>xxxxx</passwd>
    <idioma>x</idioma>
    <TIS>xxx</TIS>
    <fecha>yyyymmdd</fecha>
  </p70PeticionConsultaPNDPorTIS>
```

Mensaje soap enviado para la dispensación

```
- <PeticionDispensacion xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance" xmlns:SOAP-
    ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <importe>xx</importe>
    <importeAportacion>xx</importeAportacion>
    <codFarmacia>xxxx</codFarmacia>
    <fechaDispensacion> yyyymmdd </fechaDispensacion>
    <idPrescripcion>xxxxxx</idPrescripcion>
    <idioma>x</idioma>
    <idUsuario>xxxx</idUsuario>
    <idFarmaceutico>xxxx</idFarmaceutico>
    <claveUsuario>xxxx</claveUsuario>
    <envasesDispensados>x</envasesDispensados>
    <numeroReceta />
    <codProvincia>xx</codProvincia>
    <codigoDispensado />
```

```
</PeticionDispensacion>
```

Mensaje soap recibido con la información de la dispensación

```
- <RespuestaPeticion xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <NMotivosRechazo>1</NMotivosRechazo>

    <NAvisos>0</NAvisos>

  - <textoAviso xmlns:xsi="http://www.w3.org/2001/XMLSchema-
      instance" xmlns:SOAP-
      ENC="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="xsd:string[0]">

    </textoAviso>

  - <textoMotivosRechazo
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="xsd:string[1]">

      <SOAP-ENC:string>Se ha producido un error al dispensar
        </SOAP-ENC:string>

    </textoMotivosRechazo>

  - <codigoMotivosRechazo
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="xsd:string[1]">

      <SOAP-ENC:string>errorDispensar</SOAP-ENC:string>

    </codigoMotivosRechazo>

  - <codigoAviso xmlns:xsi="http://www.w3.org/2001/XMLSchema-
      instance" xmlns:SOAP-
      ENC="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="xsd:string[0]">

    </codigoAviso>

  </RespuestaPeticion>
```

Cómo utilizar los componentes del Interface

Se ha dejado bastante simplificado el uso de los componentes tanto para dispensar como para recuperar las prescripciones pendientes.

Es necesario incluir una referencia a la librería (dll) necesaria en el proyecto o en su defecto utilizar la sentencia ***CreateObject*** para poder instanciar los objetos.

En los ejemplos inferiores se muestra como se deben de utilizar los componentes para el correcto funcionamiento del sistema.

Recuperar las prescripciones pendientes

Para recuperar las prescripciones pendientes para un paciente hay que utilizar la librería ***Initiator.dll***, crear una serie de objetos y llamar al método *GetCardData* de la clase *CCard* y a *RecuperarPrescripciones* de la clase *CInitiator*

Es necesario rellenar los campos Usuario y Password del objeto objArgumentos, porque serán los que se utilicen para identificar a la farmacia cuando se conecte con el servicio web.

*Aparece marcado en color marrón

Al finalizar el procedimiento o función es importante asegurarse de destruir todos los objetos para no ir dejando referencias sueltas en memoria.

Si el proceso finaliza correctamente, podemos recuperar los datos de las prescripciones utilizando el objeto *objPrescripciones.* Este contiene una colección de objetos de la clase *CPrescripcion* llamada *Registros*.

*Aparece marcado en color azul

Si por el contrario se producen errores durante el proceso ya sea debidos al lector de tarjetas si existiese, a la conexión, servicio web etc.., se puede recuperar la información del error producido de dos formas distintas:

- Si se produce un error al llamar al método *GetCardData* que recupera la información de la tarjeta el objeto objError declarado como instancia de la clase CErrorItem tiene una serie de propiedades como el número de error, descripción del error, etc.., que se pueden consultar directamente para determinar que ocurrió.
- Si se produce un error/es al llamar al método *RecuperarPrescripciones* que recupera las prescripciones pendientes para un paciente, lo primero que hay que hacer es comprobar el valor de la propiedad *DatosValidos* de la clase *objPrescripciones*, si el valor devuelto es **False** indica que se han producido errores y para ello tendremos que recorrer la colección de errores que contiene el objeto *objInitiator* que a su vez son objetos de la clase CErrorItem, por lo tanto podríamos recuperar la información del error/es producidos de la misma manera que en el caso anterior.

*Aparece marcado en color rojo

```
Private Sub Command1_Click()
    Dim objArgumentos As New CArgument
    Dim objError As New CErrorItem
    Dim objCard As New CCard
    Dim objInitiator As New CInitiator
    Dim objPrescripciones As New CPrescripciones
    Dim vntElement As Variant

    'Establecer el nombre de usuario y la contraseña para conectarse al servicio, el idioma es
opcional
    objArgumentos.Usuario = "farma"
    objArgumentos.Password = "farma"

    'Recuperar los datos de la tarjeta
    If objCard.GetCardData(objArgumentos, objError) Then
        Set objPrescripciones = objInitiator.RecuperarPrescripciones(objArgumentos)
        If objPrescripciones.DatosValidos Then
            'Pasar los datos del objeto Prescripciones al cliente
            Msgbox "Proceso terminado correctamente", vbInformation
            Msgbox objPrescripciones.Registros.Item(1). m_strIdPrescripcion
        Else
            'Se han producido errores
            For Each vntElement In objInitiator.Errores
                Msgbox  vntElement.Description
            Next
        End If
    Else
        Msgbox  "Error Al recuperar la información de la tarjeta " & objError.Description
    End If

    'Destruir los objetos
    Set objInitiator = Nothing
    Set objPrescripciones = Nothing
    Set objCard = Nothing
    Set objArgumentos = Nothing
    Set objError = Nothing
End Sub
```

Dispensar las recetas

Para realizar una dispensación hay que utilizar la librería ***Interface.dll***, crear un objeto CInterface, CRespuestaPeticion y CPeticionDispensacion y llamar al método *SolicitudDispensacion* de la clase CInterface.

Los objetos de las tres clases anteriores se pueden declarar como **WithEvents,** para recoger los eventos que lanzan, como por ejemplo AlIniciarSolicitudDispensacion, AlConectar, etc..

Es necesario rellenar un mínimo de campos para poder realizar la dispensación, además se muestran todos los campos a rellenar opcionalmente por si en futuro se decide a su utilización.

*Aparece marcado en color marrón

Al finalizar el procedimiento o función es importante asegurarse de destruir todos los objetos para no ir dejando referencias sueltas en memoria.

Si el proceso finaliza correctamente podemos recuperar la respuesta a la solicitud de dispensación que nos devolvió el servicio web accediendo a las propiedades del objeto de la clase CRespuestaPeticion.

*Aparece marcado en color azul

Si se produce un error/es al llamar al método *SolicitudDispensacion* que realiza la dispensación, lo primero que hay que hacer es comprobar el valor de la propiedad *DatosValidos* de la clase *CRespuestaPeticion*, si el valor devuelto es **False** indica que se han producido errores y para ello tendremos que recorrer la colección de errores que contiene el objeto de la clase *CInterface* que a su vez son objetos de la clase *CErrorItem.*

*Aparece marcado en color rojo

```
Private Sub Command1_Click()
    DIm WithEvents objInterfaz As CInterface
    Dim WithEvents objPeticion As CPeticionDispensacion
    Dim WithEvents objRespuesta As CRespuestaPeticion
    Dim vntElement As Object
    Dim i As Integer
    Dim lngInicio As Long
    Dim lngFin As Long

    Set objInterfaz = New CInterface
    Set objPeticion = New CPeticionDispensacion
    Set objRespuesta = New CRespuestaPeticion


    With objPeticion
        .IdUsuario = "farma"                        'IdUsuario
        .ClaveUsuario = "farma"                     'ClaveUsuario
        .Idioma = "0"                               'Idioma
        .IdPrescripcion = "'2800000000118"          'IdPrescripcion
        .FechaDispensacion = Me.txtFecha            'FechaDispensacion Formato UTC
yyyy/mm/dd
        .IdFarmacia = "farma1"                      'IdFarmacia
        .IdFarmaceutico = "farma"                   'IdFarmaceutico
        .EnvasesDispensados = "3"                   'EnvasesDispensados
        .Importe = "111"                            'Importe
        .ImporteAportacion = "22"                   'ImporteAportacion
        .CodigoDispensado = ""                      'Código del producto dispensado
        .IdProvincia="23"                           'Provincia de la farmacia
    End With

    Set objRespuesta = objInterfaz.SolicitudDispensacion(objPeticion)
```

```
 If objRespuesta.DatosValidos Then
If  objRespuesta. DispensacionRealizada then
                    'Realizar la acción pertinente
                    'Msgbox "Proceso terminado correctamente", vbInformation
End if
Else
    'Se han producido errores
    For Each vntElement In objInterfaz.Errores
       Msgbox  vntElement.Description
    Next
 End If
End Function
```

Arquitectura

Descripcion

La aplicación esta compuesta por dos módulos principales, uno para realizar la dispensación de recetas (interface.dll) y otro para recuperar las prescripciones pendientes de un paciente (initiator.dll). Ambos son librerías de enlace dinámico ActiveX realizadas con Microsoft Visual Basic 6.0 SP5.

Así como un archivo ejecutable que se encarga de instanciar objetos en hilos separados de las librerías para permitir el proceso asíncrono.

Las clases que componen cada componente pueden verse a continuación.

Vista externa de las clases

La librería Initiator.dll encargada de recuperar las prescripciones pendientes de un paciente contiene las siguientes clases:

Clase: CArgument Esta clase contiene los datos del paciente como el TIS, el login y password para acceder al servicio web. Se utiliza para almacenar los datos de la tarjeta chip o de la banda magnética y para pasar información al servicio web que devuelve las prescripciones pendientes de dispensar.			
Propiedades:			
Usuario	String	LE	Identificador del usuario
Password	String	LE	Password del usuario
Idioma	Integer	LE	Identificador de idioma para los mensajes de error
TIS	String	LE	TIS del paciente
TitularTarjeta	String	LE	Titular de la tarjeta

Lector	String	LE	Indica de donde se han leído los datos del paciente. Puede tener los siguientes valores: 'C' Chip de la tarjeta 'B' Banda magnética de la tarjeta '' TIS introducido manualmente
Fecha	String	L	Fecha de la petición, se utiliza la fecha del sistema por defecto

Clase: CCard Esta clase se encarga de comprobar el estado del lector de tarjetas, recuperar los datos que contiene la tarjeta y que son necesarios para realizar la consulta al servicio web que devuelve las prescripciones pendientes.	
Funciones:	
GetCardData	**(ByRef objArgument As CArgument, ByRef objError As CErrorItem) As Boolean**
	Recupera los datos de la tarjeta chip y los almacena en el objeto CArgument

Clase: CCardBM Esta clase se encarga de comprobar el estado del lector de tarjetas, recuperar los datos que contiene la tarjeta y que son necesarios para realizar la consulta al servicio web que devuelve las prescripciones pendientes.	
Funciones:	
GetCardDataBM	**(ByVal strLecturaBM As String, ByRef objArgument As CArgument, ByRef objError As CErrorItem) As Boolean**
	Recupera los datos de la banda magnética de la tarjeta

Clase: CErrorItem Esta clase contiene toda la información relativa a un error.		
Atributos:		
Description	String	Descripción completa del error
HelpContext	String	Contexto del error
HelpFile	String	Nombre del archivo de ayuda asociado al error
Number	Long	Código numérico del error
Source	String	Nombre de la clase y/o función en la que se produjo el error

Clase: CError

Esta clase contiene una colección de objetos CErrorItem con todos los errores producidos.

Propiedades:

Errores	Collection	L	Colección de objetos CErrorItem con todos los errores producidos

Funciones:

AddError	**(ByRef objError As ErrObject)**
	Transforma un objeto objError en un objeto CErrorItem y lo añade a la colección de errores
AddError2	**(ByRef objError As ErrObject)**
	Añade un objeto CErrorItem a la colección de errores
ResetErrorsCollection	**()**
	Elimina todos los elementos de la colección de errores

Clase: CInitiator

Esta clase es la encargada de conectarse con el servicio web y de recuperar las prescripciones pendientes de un paciente.

Propiedades:

Errores	Collection	L	Colección de objetos CErrorItem con todos los errores producidos

Funciones:

RecuperarPrescripciones	**(ByRef objArg As CArgument, Optional ByVal blnCronico As Boolean) As CPrescripciones**
	Recupera todas las prescripciones pendientes que tiene un paciente

Clase: CPrescripciones

Esta clase contiene toda la información devuelta por el servicio web.

Propiedades:

DatosValidos	Boolean	L	Indica si los datos que contiene de la propiedad *m_colPrescripciones* son válidos
Registros	Collection	L	Colección de objetos CPrescripcion, con las prescripciones pendientes

CodigoError	Long	L	Código de error generado por la respuesta del servicio web
DescripcionError	String	LE	Descripción del error generada por el servicio web

Clase: CPrescripcion Esta clase contiene los atributos particulares de una prescripción devuelta por el servicio web.			
Propiedades:			
IDPrescripcion	String	L	Identificador de la prescripción
FechaPrescripcion	String	L	Fecha de realización de la prescripción
CodigoProducto	String	L	Código del producto prescrito
EnvasesPrescritos	Integer	L	Número de envases prescritos
RegimenReceta	RegimenRecetaEnum	L	Régimen de la receta 0 Activos 1Pensionistas 2 Otros
RequiereVisado	RequiereVisadoEnum	L	Puede tener los siguientes valores: 0 No requiere visado 1 Pendiente de visado 2 Ya visado 3 Visado rechazado

Clase: CLog Esta clase se utiliza para la generación del log o de trazas.	
Funciones:	
WriteIn	**(ByVal strData As String, Optional ByVal blnAddDate As Boolean = True)**
	Escribe una línea en el archivo de log
WriteTrace	**(ByRef objClass As Object, ByVal blnIsIn As Boolean, ByVal strMethod As String, Optional ByVal strMoreData As String = "", Optional ByVal blnAddDate As Boolean = True)**
	Escribe una línea en el archivo de trazas

Interfaz: IObject	
Esta clase se encarga de la administración de archivos, toda creación, eliminación cambio de emplazamiento se realiza llamando a las funciones de proporciona la clase.	
Funciones:	
FromXML	**(ByRef objIXMLDOMNodeList As Object) As Boolean**
	Lectura de los datos de un archivo XML
RequiredFields	**() As Boolean**
	Comprobación de datos requeridos para el mensaje SOAP
ToXML	**() As Object**
	Escritura de los datos a un archivo XML

La librería Interface.dll encargada de realizar la dispensación de las prescripciones pendientes de un paciente contiene las siguientes clases:

Clase: CAuditor			
Esta clase se utiliza para la auditoría de mensajes.			
Propiedades:			
DiaryStore	Boolean	LE	Indica si los mensajes se guardan diariamente.
Enable	Boolean	LE	Indica si el objeto esta activo
OutputFileName	String	L	Path completo del archivo de log
Funciones:			
Save			**(ByVal strIdPrescripcion As String, ByVal intAction As AuditTypeEnum, Optional ByVal intResponse As ReceivedTypeEnum = rcvNone, Optional ByVal strTextResponse As String = "") As Boolean**
			Audita sobre generación y envío de mensajes

Clase: CConfig			
Esta clase se utiliza para la gestión de configuración de la aplicación			
Propiedades:			
EsquemaXML	String	L	Información de esquemas utilizados por XML para enviar y recibir mensajes mediante SOAP

CodigoDispensacionOk	String	L	Código interno que indica que la dispensación se realizó correctamente

Clase: CErrorItem Esta clase contiene toda la información relativa a un error.		
Atributos:		
Description	String	Descripción completa del error
HelpContext	String	Contexto del error
HelpFile	String	Nombre del archivo de ayuda asociado al error
Number	Long	Código numérico del error
Source	String	Nombre de la clase y/o función en la que se produjo el error

Clase: CError Esta clase contiene una colección de objetos CErrorItem con todos los errores producidos.			
Propiedades:			
Errores	Collection	L	Colección de objetos CErrorItem con todos los errores producidos
Funciones:			
AddError			**(ByRef objError As ErrObject)**
			Transforma un objeto objError en un objeto CErrorItem y lo añade a la colección de errores
AddError2			**(ByRef objError As ErrObject)**
			Añade un objeto CErrorItem a la colección de errores
ResetErrorsCollection			**()**
			Elimina todos los elementos de la colección de errores

Clase: CExecutor Esta clase es la encargada de realizar la dispensación, es la única que se puede conectar con el servicio web.
Propiedades:

NombreServicio	String	L	URL del servicio al que se está conectado
Conectado	Boolean	L	Indica si se ha conectado con el servicio web
DatosEnviados	Boolean	L	Indica si la petición ha sido enviada al servidor
Enabled	Boolean	L	Indica si la interfaz esta habilitada
Errores	Collection	L	Colección de items con todos los errores que se han producido en la dispensación, tanto por parte del Interface como por parte del servicio web, etc.
NombreLog	String	L	Path completo del archivo de log
Opciones	Long	LE	Opciones de configuración adicionales (Son leídas desde el archivo de configuración)
OpcionesLog	Long	LE	Opciones de configuración adicionales para el archivo de log
IDPrescripcion	String	LE	Identificador de la prescripción a dispensar

Funciones:	
ConectarServicio	**() As Boolean**
	Conecta con el servicio web
SolicitudDispensacion	**(ByVal strIdPrescripcion As String) As Boolean**
	Realiza la dispensación conectando con el servicio web. Devuelve un boolean que indica si ocurrió algún error
SolicitudDispensaciones	**(ByVal strIdPrescripcion As String) As Boolean**
	Realiza la dispensación masiva conectando con el servicio web. Devuelve un boolean que indica si ocurrió algún error

Eventos:	
AlConectar	Se lanza una vez se ha realizado la conexión con el servicio web
AlIniciarSolicitudDispensacion	Se lanza al iniciar la solicitud de dispensación
AlTerminarLeerIni	Se lanza al terminar de leer la configuración del archivo .ini
AlTerminarSolicitudDispensacion	Se lanza al terminar la solicitud de dispensación

Clase: CFileAdmin Esta clase se encarga de la administración de archivos, toda creación, eliminación cambio de emplazamiento se realiza llamando a las funciones de proporciona la clase.			
Propiedades:			
DefaultResponseFile	String	LE	Archivo de respuesta por defecto
InternalResponseFile	String	L	Archivo de respuesta interno
Funciones:			
DeleteFromReceived			**(ByVal strIdPrescripcion As String) As Boolean** Elimina un archivo de la carpeta Recibidos
DeleteFromSended			**(ByVal strIdPrescripcion As String) As Boolean** Elimina un archivo de la carpeta Enviados
DeleteLogFileIfExcededSize			**()** Elimina el archivo de log si este supera el tamaño máximo en bytes que esta definido en el archivo de inicialización
ExistDefaultResponseFile			**() As Boolean** Indica si existe el archivo de respuesta por defecto
ExistExecutorFile			**() As Boolean** Indica si existe el archivo executor.exe
ExistIniFile			**() As Boolean** Indica si existe el archivo de inicialización
ExistResponseFile			**(ByVal strIdPrescripcion As String, Optional ByVal ResponseDir As FolderEnum = fldReceived) As Boolean** Indica si existe el archivo de respuesta
GetAuditorPath			**() As String** Retorna el path de la carpeta Auditoria
GetDelayedPath			**() As String**

	Retorna el path de la carpeta Retrasados
GetDllPath	**() As String**
	Retorna el path donde se encuentra la dll
GetReceivedPath	**() As String**
	Retorna el path de la carpeta Recibidos
GetSendedPath	**() As String**
	Retorna el path de la carpeta Enviados
MakeAuditor	**() As Boolean**
	Crea la carpeta Auditoria
MakeDelayed	**() As Boolean**
	Crea la carpeta Retrasados
MakeReceived	**() As Boolean**
	Crea la carpeta Recibidos
MoveToDelayed	**(ByVal strFolder As String, ByVal strFileTitle As String) As Boolean**
	Mueve un archivo de una carpeta a la capeta Retrasados
MakeSended	**() As Boolean**
	Crea la carpeta Enviados

Clase: CInterface

Esta clase es la administradora del motor de la interface para realizar la dispensación.

Propiedades:

Enabled	Boolean	L	Indica si el objeto esta activo
Errores	Collection	L	Colección de objectos CerrorItem con todos los errores que se han producido en la dispensación, tanto por parte del Interface como por parte del servicio web, etc.
NombreLog	String	L	Path completo del archivo de log
NombreServicio	String	L	URL del servicio al que se está conectado

Opciones	Long	LE	Opciones de configuración adicionales (Son leídas desde el archivo de configuración)
OpcionesLog	Long	LE	Opciones de configuración adicionales para el archivo de log

Funciones:	
SolicitudDispensacion	**(objPeticionDispensacion As CPeticionDispensacion) As CRespuestaPeticion**
	Realiza la dispensación iniciando un subproceso asíncrono. Devuelve un objeto CRespuestaPeticion
SolicitudDispensaciones	**(ByRef objListaDispensaciones As CListadoDispensaciones) As CRespDispensaciones**
	Realiza la dispensación masiva iniciando un subproceso asíncrono. Devuelve un objeto CRespDispensaciones
Eventos:	
AlConectar	Se lanza una vez se ha realizado la conexión con el servicio web
AlIniciarSolicitudDispensacion	Se lanza al iniciar la solicitud de dispensación
AlTerminarLeerIni	Se lanza al terminar de leer la configuración del archivo .ini
AlTerminarSolicitudDispensacion	Se lanza al terminar la solicitud de dispensación

Clase: CListadoDispensaciones

Esta clase se utiliza para la dispensación en grupo de prescripciones.

Propiedades:			
Dispensaciones	Collection	L	Colección de peticiones de dispensación para el caso de la dispensación masiva

Funciones:	
aniadirDispensacion	**(ByRef objPeticionDispensacion As CPeticionDispensacion)**
	Añade una nueva petición de dispensación a la colección de peticiones de dispensación
ResetDispensacionesCol	**()**
	Elimina todos los elementos de la colección de peticiones de dispensación

Clase: CLog Esta clase se utiliza para la generación del log o de trazas.	
Funciones:	
WriteIn	**(ByVal strData As String, Optional ByVal blnAddDate As Boolean = True)**
	Escribe una línea en el archivo de log
WriteTrace	**(ByRef objClass As Object, ByVal blnIsIn As Boolean, ByVal strMethod As String, Optional ByVal strMoreData As String = "", Optional ByVal blnAddDate As Boolean = True)**
	Escribe una línea en el archivo de trazas

Clase: CPeticionDispensacion Esta clase se encarga de la administración de archivos, toda creación, eliminación cambio de emplazamiento se realiza llamando a las funciones de proporciona la clase.		
Atributos:		
ClaveUsuario	String	Clave de usuario
EnvasesDispensados	String	Número de envases dispensados
IdUsuario	String	Identificador de usuario
FechaDispensacion	String	FechaDispensacion Formato UTC yyyy/mm/dd
Idioma	String	Idioma
CodigoDispensado	String	Código del producto dispensado
Importe	String	Importe
ImporteAportacion	String	Importe de la aportación
IdFarmaceutico	String	Id del farmaceútico
IdFarmacia	String	Id de la farmacia
IdPrescripcion	String	Identificador de la prescripción
CodigoColegiadoPrescriptor	String	Código del colegiado que realiza la prescripción
FechaPrescripcion	String	Fecha de la prescripción, debería ser el día en el que se prescribe la receta
TIS	String	Código TIS del paciente
RegimenReceta	Integer	Puede ser 0 Trabajador/Activo 1 :Pensionista

FormulaMagistral	String	Nombre de la fórmula magistral en el caso que el tipo de prescripción sea fórmula magistral	
TipoPrescripcion	Integer	Tipo de la prescripción, puede tener los siguientes valores: 1 Especialidad farmacéutica 2 Principio activo 3 Formula magistral 5 Efectos y accesorios	
IdProvincia	String	Provincia de la farmacia	
NumeroReceta	String	Número de la receta	
Lector	String	Indica de donde se han leído los datos del paciente. Puede tener los siguientes valores: 'C' Chip de la tarjeta 'B' Banda magnética de la tarjeta '' TIS introducido manualmente	
Propiedades:			
SaveSOAPMsgToFile	Boolean	LE	Clave de usuario
Eventos:			
onBeforeTransformFromXML	Se lanza antes de transformar datos desde el archivo XML		
onBeforeTransformToXML	Se lanza antes de transformar datos a un archivo XML		
onEndTransformFromXML	Se lanza al terminar de transformar datos desde el archivo XML		
onEndTransformToXML	Se lanza al terminar de transformar datos a un archivo XML		

Clase: CRespDispensaciones

Esta clase contiene los datos devueltos por el mensaje para su análisis.

Propiedades:			
DatosValidos	Boolean	LE	Indica si los datos que contiene el objeto son válidos
DispensacionRealizada	Boolean	L	Indica si la dispensación se realizó correctamente
ResultadoDispensacion	Collection	L	Colección con los resultados de las dispensaciones

Funciones:	
addDispensationResult	**(ByRef objRespError As CRespError)**
	Añade un resultado a la colección de resultados de dispensaciones
Eventos:	
onBeforeTransformFromXML	Se lanza antes de transformar datos desde el archivo XML
onBeforeTransformToXML	Se lanza antes de transformar datos a un archivo XML
onEndTransformFromXML	Se lanza al terminar de transformar datos desde el archivo XML
onEndTransformToXML	Se lanza al terminar de transformar datos a un archivo XML

Clase: CRespError Clase para los errores de la dispensacion.		
Atributos:		
Codigo	String	Código del error
Description	String	Descripción del error
CodigoDispensado	String	Código del producto dispensado
TipoRespuesta	String	Tipo de repuesta
Funciones:		
createResp	**(ByRef objNode As Object) As CRespError**	
	Crea un objeto con la respuesta de error.	
ToString	**() As String**	
Eventos:		
onBeforeTransformFromXML	Se lanza antes de transformar datos desde el archivo XML	
onBeforeTransformToXML	Se lanza antes de transformar datos a un archivo XML	
onEndTransformFromXML	Se lanza al terminar de transformar datos desde el archivo XML	
onEndTransformToXML	Se lanza al terminar de transformar datos a un archivo XML	

Clase: CRespuestaPeticion			
Esta clase se encarga de la administración de archivos, toda creación, eliminación cambio de emplazamiento se realiza llamando a las funciones de proporciona la clase.			
Atributos:			
m_colCodigoAviso	Collection	Colección de códigos de avisos producidos	
m_colCodigoMotivosRechazo	Collection	Colección de códigos de motivos de rechazo	
m_colTextoAviso	Collection	Colección de descripciones de aviso	
m_colTextoMotivosRechazo	Collection	Colección de descripciones de motivos de rechazo	
m_intNAvisos	Integer	Número de avisos recibidos	
m_intNMotivosRechazo	Integer	Número de motivos de rechazo recibidos	
Propiedades:			
DatosValidos	Boolean	LE	Indica si los datos que contiene el objeto son válidos
DispensacionRealizada	Boolean	L	Indica si la dispensación se realizó correctamente
Eventos:			
onBeforeTransformFromXML			Se lanza antes de transformar datos desde el archivo XML
onBeforeTransformToXML			Se lanza antes de transformar datos a un archivo XML
onEndTransformFromXML			Se lanza al terminar de transformar datos desde el archivo XML
onEndTransformToXML			Se lanza al terminar de transformar datos a un archivo XML

Interfaz: IObject	
Esta clase se encarga de la administración de archivos, toda creación, eliminación cambio de emplazamiento se realiza llamando a las funciones de proporciona la clase.	
Funciones:	
FromXML	**(ByRef objIXMLDOMNodeList As Object) As Boolean**
	Lectura de los datos de un archivo XML

RequiredFields	**() As Boolean**
	Comprobación de datos requeridos para el mensaje SOAP
ToXML	**() As Object**
	Escritura de los datos a un archivo XML