SOLUCIONES

Best Technology for Best Business

Engage Business Solution 5
Integración de servicios web con Engage Integration Services 5.4

Tucumán 540 Piso 9 H · C1049AAL · Buenos Aires · Argentina
www.soluciones-ar.com.ar · resur-info@soluciones-ar.com.ar
Tel/Fax: + 54 (11) 4394 8575

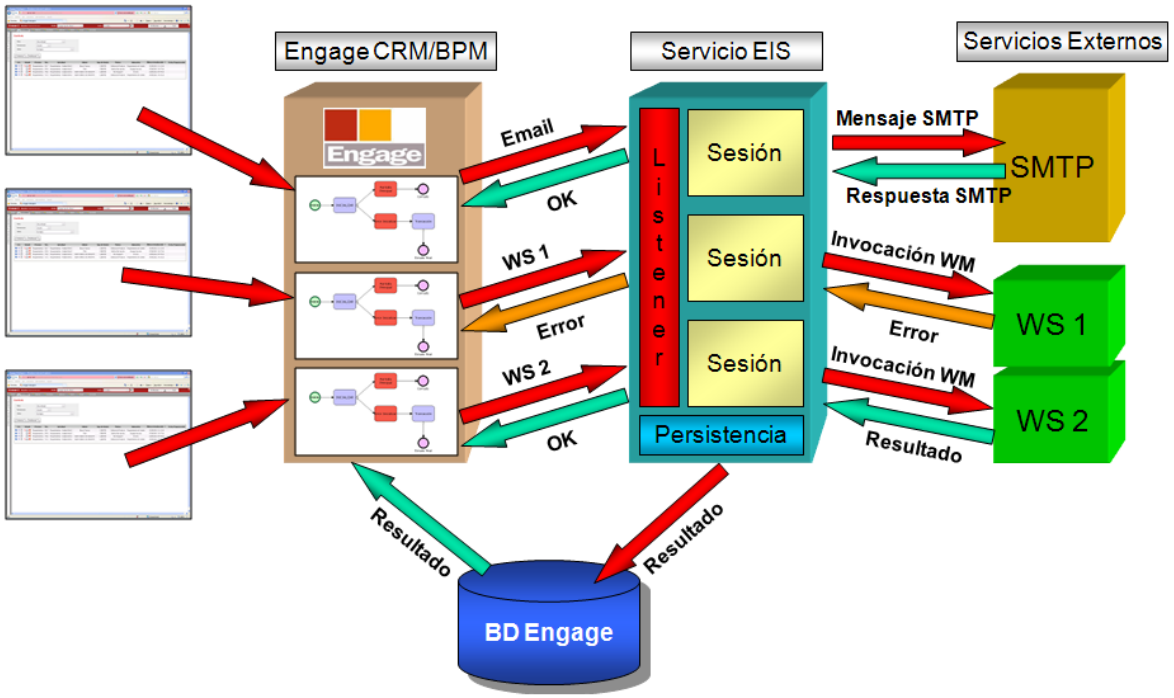
Contenido

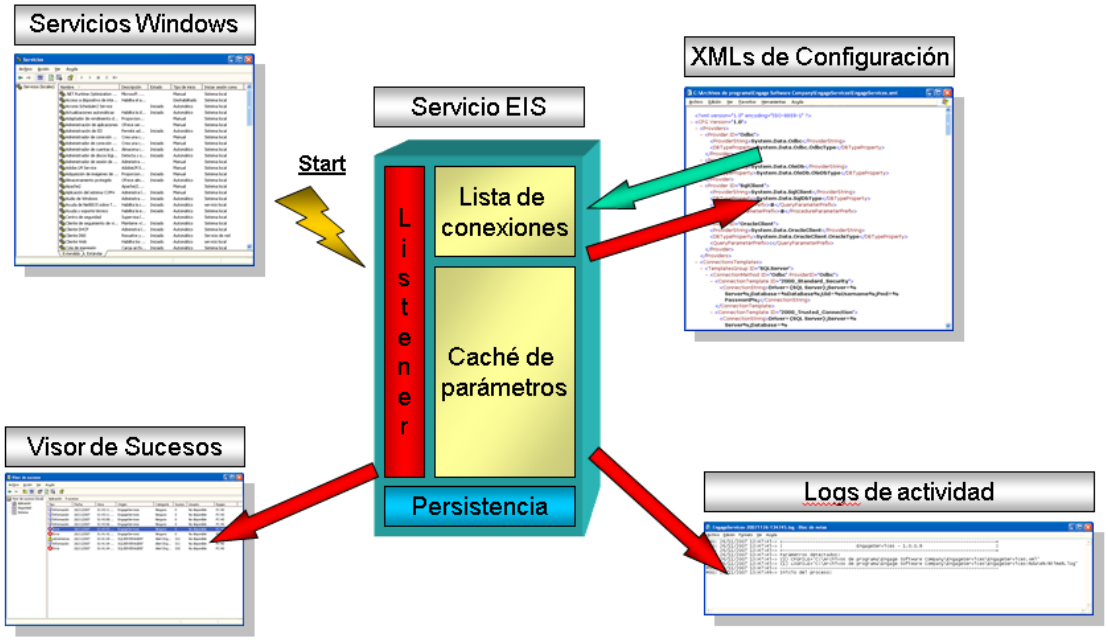
¿Qué es Engage Integration Services?

- Definición conceptual:
    - Es una capa de servicios que están disponibles para ser consumidos o invocados desde Engage. Entre esos servicios se encuentra el que permite la integración con servicios web.
    - Engage Integration Services está preparado para trabajar en forma independiente del motor de Engage.
- Definición técnica:
    - Es un servicio Windows que escucha en un puerto TCP, recibe peticiones de los clientes, ejecuta el servicio solicitado y devuelve una respuesta con el estado de la ejecución.
    - Eventualmente, permite la persistencia de la información devuelta en tablas asociadas al modelo de datos de Engage.

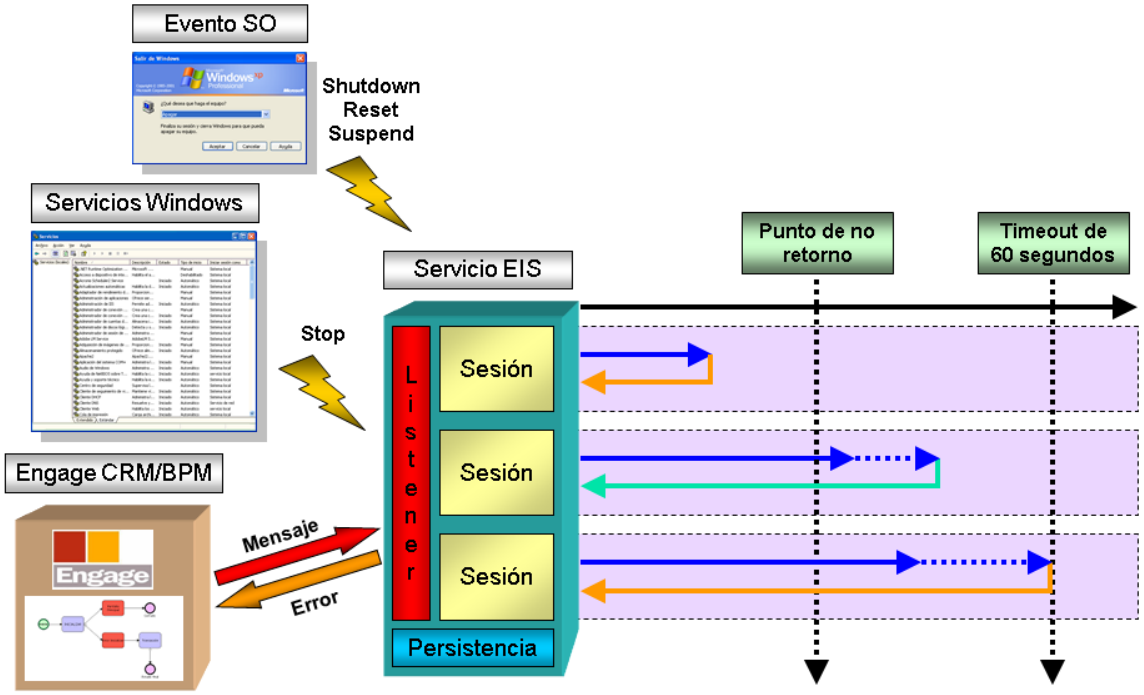
Características técnicas

- Desarrollado en .Net, Framework 4.0.
- Arquitectura integrada a Engage 5.x.
- Es un servicio de Windows nativo.
- Se configura a través de archivos XML.
- La actividad se registra en archivos de LOG.
- Acepta múltiples conexiones concurrentes.
- Accede a la BD de Engage en forma directa.





Tucumán 540 Piso 9 H · C1049AAL · Buenos Aires · Argentina
www.soluciones-ar.com.ar · resur-info@soluciones-ar.com.ar
Tel/Fax: + 54 (11) 4394 8575



¿Qué es un servicio web?

- Es una colección de protocolos (WSPS) y estándares que interactúan entre si para facilitar el intercambio de datos entre aplicaciones desarrolladas en lenguajes diferentes y ejecutadas sobre cualquier plataforma.
- Protocolos involucrados:
    - XML (datos)
    - SOAP (intercambio)
    - HTTP (transferencia)
    - WSDL (descripción)

¿Qué es una clase proxy?

- Es código escrito en un lenguaje de programación, que mapea a un contrato WSDL y que sirve para facilitar la invocación de un servicio web desde una aplicación.

- Aportan dinamismo ante la necesidad de incorporar nuevos servicios web o de actualizar los existentes.

- Evitan la necesidad de tener que interpretar los protocolos en forma directa, ya sea para consumir un servicio web como para obtener su descripción.

Cómo crear una librería proxy

- Para crear una librería proxy se utilizan archivos de comandos .bat, desde los cuales se invocan varios utilitarios de terceros:

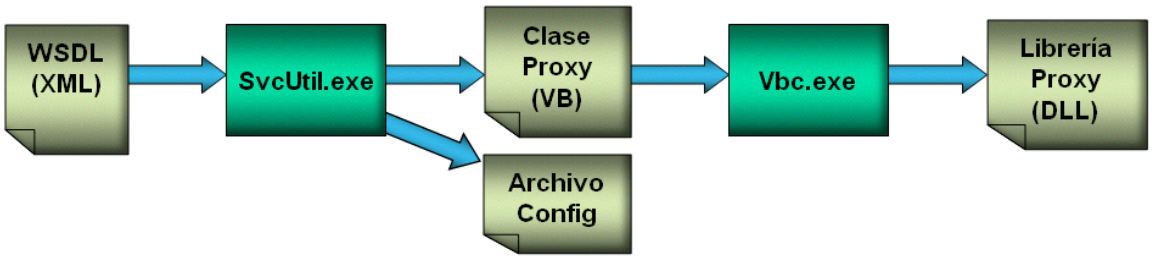
    - *SvcUtil.exe:* Utiliza el WSDL como entrada y genera como resultado el código fuente de la clase proxy y de otras clases que representan las estructuras de datos que van y vienen. El código fuente está escrito en Visual Basic .Net. También crea un archivo .config de ejemplo, que contiene la configuración del canal de comunicación con el servicio web.

    - *Vbc.exe:* Compila la clase proxy generada por SvcUtil.exe y la incluye en un ensamblado de .Net (dll) que llamaremos "librería proxy".

El siguiente es un ejemplo de un archivo de comandos .bat que se utiliza para crear una librería proxy:

```
@echo off

cls

set WSDLPATH=http://localhost/WSExamples20/WSExampleSimple.asmx?wsdl
set DLLNAME=ClientWSExampleSimple

"..\VsSdkTools\svcutil.exe" /config:..\..\%DLLNAME%.config /l:VB /tcv:Version35
/o:.\%DLLNAME%.vb %WSDLPATH%

echo.

"..\VsSdkTools\vbc.exe" /t:library /out:..\..\%DLLNAME%.dll %DLLNAME%.vb

echo.

del %DLLNAME%.vb > nul
```

```
pause
```



¿Qué es una transformación?

- Son scripts escritos en XML, que se aplican sobre la respuesta de la ejecución de un método web.
- Se usan para inspeccionar la respuesta, seleccionar los datos que se necesitan y construir sentencias SQL que se ejecutarán en la base de datos para persistir la información obtenida.
- Se crean en base a las clases proxy, explorándolas con la ayuda del utilitario EISProxyManager.
- Se guardan en un archivo de transformaciones y se identifican unívocamente mediante un ID.
- En cada transformación se especifica lo siguiente:
    - En qué tablas se guardarán los datos de la respuesta y a qué campos de esas tablas se asignarán dichos datos.
    - Si se necesitan vaciar primero las tablas destino, qué datos de la respuesta formarán parte del filtro de eliminación de los registros existentes.
    - En qué dato de la respuesta vienen informados los errores de lógica y qué valores de ese dato deben ser tratados como errores.
    - Cual es el nombre de la clase de serialización que se utilizará para pre-procesar la respuesta, en caso que el valor devuelto sea un XML

informado como texto plano.

Mensajes de Engage Integration Services

Como regla general, los mensajes de invocación y las respuestas de Engage Integration Services son cadenas de texto delimitadas. Ya sea para los mensajes de entrada como para las respuestas, el delimitador que se utilizará siempre es el mismo, puede tener más de un caracter y debe especificase en el archivo de configuración EngageIntegrationServices.xml

Formatos del mensaje de invocación

Estándar:

```
WS|ProxyLibraryName|ClassName|
MethodName|CustPkey|JobPkey|UserId|
[TransformationId]|[MethodParameters]
```

Extendido:

```
WSX|ProxyLibraryName|ClassName|
MethodName|CustPkey|JobPkey|UserId|
[TransformationId]|[MethodParameters]
[EOM]
```

Engage Integration Services contempla dos modos del mensaje de invocación:

1- *Modo estándar:* Engage Integration Services asume que todo el mensaje de invocación viene en un solo paquete TCP, con lo cual, luego de recibir cada paquete iniciará el parseo y la validación del mensaje de invocación.

2- *Modo extendido:* Si el mensaje de invocación es muy largo, Engage Integration Services puede recibirlo particionado en más de un paquete TCP, con lo cual se

producirán errores en el parseo del mensaje o en la invocación del método web. Si estamos ante esta posibilidad, entonces debemos utilizar el modo extendido. Para ello, debemos agregar una X al final del tipo de servicio y la cadena de texto [EOM] al final del mensaje. De esta manera, Engage Integration Services concatenará todos los paquetes TCP recibidos hasta que en uno de ellos venga el delilmitador de fin de mensaje [EOM] y luego realiza el parseo y la validación del mensaje.

Nota: Se recomienda utilizar siempre el modo extendido.

Parámetros del mensaje de invocación

- **WSX**: Nombre del servicio (WS, HS, HTTP, MQ, EMAIL, etc)
- **ProxyLibraryName**: Nombre de la dll proxy, sin la extensión .dll
- **ClassName**: Nombre de la clase que representa al servicio web
- **MethodName**: Nombre del método web que se va a ejecutar
- **CustPkey**: Pkey del cliente con que se ejecuta la transacción
- **JobPkey**: Pkey del proceso con que ejecuta la transacción
- **UserId**: Id del usuario Engage que ejecuta la transacción
- **TransformationId (opcional)**: Id de la transformación que se aplicará sobre la respuesta del método web
- **MethodParameters (opcional)**: Parámetros del método web
- **[EOM]**: Terminador del mensaje

Parámetros del método web

La lista de parámetros informada en MethodParameters debe coincidir exactamente con la cantidad, el orden y los tipos de datos de los parámetros del método web. Engage Integration Services contempla parámetros por valor y por referencia. El formato genérico de la lista de parámetros del método web es la siguiente:

TipoDatoSimple1|TipoDatoSimple2|TipoDatoComplejo1|TipoDatoSimple3| ...

Donde: TipoDatoSimple son tipos de datos básicos (Numéricos, Boolean, String, Enumeraciones, DateTime, etc) y TipoDatoComplejo son clases o arrays.

El tipo de dato complejo, según sus características, puede informarse de dos formas distintas:

1- En caso de tratarse de una clase que contenga sólamente atributos de tipos de datos simples, puede utilizarse la siguiente nomenclatura plana:

   NombreAtributo1=TipoDatoSimple;NombreAtributo2=TipoDatoSimple; … ;
   NombreAtributoN=TipoDatoSimple

   Donde NombreAtributo es el nombre del atributo de la clase que quiere poblarse y TipoDatoSimple es el valor que se le asignará al mismo, según la definición del punto anterior.

2- En caso de tratarse de un array o de clases que contengan atributos que sean arrays o clases anidadas, debe utilizarse un xml de serialización, cuyo formato puede inferirse mediante la herramienta EISProxyManager.

   Nota: Esta manera de informar los parámetros puede utilizarse también para las clases del punto anterior.

Los siguientes son ejemplos de las distintas formas de informar los parámetros del método web dentro del mensaje de invocación:

1- Método web con un solo parámetro de tipo de dato simple (en rojo):

   WSX|ClientWSExampleSimple|WSExampleSimpleSoapClient|ObtenerDigesto|CustPkey|JobPkey|UserId|OBTENCION_DIGESTO|**Su saldo actual es de $12.575**[EOM]

2- Método web con dos parámetros de tipos de datos simples (en rojo):

   WSX|ClientWSExampleSimple|WSExampleSimpleSoapClient|ValidarDigesto|CustPkey|JobPkey|UserId|VALIDAR_DIGESTO|**Su saldo actual es de $12.575**|**72834E38C38D075954B6F99EC671EEEA56AA0EAE**[EOM]

3- Método web con dos parámetros de entrada, uno de tipo de dato complejo (en azul) y otro de tipo de dato simple (en rojo). El de tipo de dato complejo está informado con la nomenclatura plana:

   WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|AgregarCliente|CustPkey|JobPkey|UserId||**Cust_Primary_Id=123;Situacion=1;Deuda=5000.00;Customer_Name=PEREZ, JUAN JOSE**|**ControlarDuplicados**[EOM]

4- Mismo método web del punto anterior, pero con el parámetro de tipo de dato complejo (en azul) informado con un xml de serialización:

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|AgregarCliente|CustPkey|JobPkey|UserId||**<?xml version="1.0" encoding="utf-8"?><clsResults xmlns:xsd=http://www.w3.org/2001/XMLSchema xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance xmlns:t="http://tempuri.org/"><t:Cust_Primary_Id>123</t:Cust_Primary_Id><t:Situacion>1</t:Situacion><t:Deuda>5000</t:Deuda><t:Customer_Name>PEREZ, JUAN JOSE</t:Customer_Name><t:Contactos><t:clsContacto><t:Telefonos><t:string>1234-5678</t:string><t:string>2345-6789</t:string></t:Telefonos><t:Emails><t:string>email1@gmail.com</t:string><t:string>email2@gmail.com</t:string></t:Emails></t:clsContacto><t:clsContacto><t:Telefonos><t:string>15-1617-1819</t:string></t:Telefonos><t:Emails><t:string>email3@yahoo.com</t:string><t:string>email4@yahoo.com</t:string></t:Emails></t:clsContacto></t:Contactos></clsResults>|ControlarDuplicados**[EOM]

Formato del mensaje de respuesta

Formato:

Code [|Message]

Parámetros:

- **Code**: Código del tipo de respuesta (OK, VALUE, ERROR)
- **Message (opcional)**: Mensaje de respuesta, en caso que el código de tipo de respuesta sea VALUE o ERROR.

Pasos para integrar servicios web

Los siguientes son los pasos a seguir para hacer todas las definiciones que permitirán consumir servicios web desde Engage, utilizando el servicio de Engage Integration Services:

1- Como primera medida, debe obtenerse el archivo wsdl del servicio web al que se quiere acceder, como así, todas las plantillas xsd a las que se hace referencia.

2- Generar la librería proxy del servicio web a partir del archivo wsdl, mediante un archivo de comandos .bat. Pueden tomarse como ejemplo los que se encuentran dentro de la carpeta .\AssembliesGeneration\ClientGeneration.

3- Copiar la dll generada en el paso anterior dentro de la carpeta en la que se encuentra el ejecutable de Engage Inetgration Services.

4- Editar los archivos .config de Engage Integration Services y de EISProxyManager, para agregar los parámetros que se encuentran dentro del archivo .config correspondiente a la librería proxy.

5- Abrir la dll con el utilitario EISProxyManager, para inspeccionar los métodos web que contiene el servicio, como así los parámetros y las estructuras de salida de cada método web. Por otro lado, el utilitario permite la invocación de los métodos web para probarlos.

6- Si el método web contiene un parámetro que sea de tipo de dato complejo, se debe crear un xml de serialización de ejemplo con EISProxyManager.

7- Si el método web devuelve alguna respuesta, ya sea a través de un valor de retorno o por medio de parámetros por referencia, y si se necesita recuperar el valor de esa respuesta, habrá que agregar la transformación correspondiente en el archivo de transformaciones.

8- En el proceso de Engage donde se encuentra la actividad desde la cual se quiere consumir un servicio web, crear una transacción de tipo socket, especificando la dirección IP del servidor donde está instalado Engage Integration Services y el puerto donde escucha.

9- Definir los parámetros de entrada/salida de la transacción. La forma de definir los parámetros está directamente relacionada con el tipo de mensaje que se envia a Engage Integration Services y el tipo de respuesta que se devuelve.

10- Incluir la transacción de socket en el grafo de la estructura de conversación de la actividad desde la cual se quiere consumir un servicio web. Es una buena práctica contemplar siempre las salidas por Error y TimeOut de socket, como así los posibles valores del código de la respuesta de Engage Integration Services.

Nuevas Funciones

1- Capacidad de guardar un array de bytes como adjunto en las transformaciones aplicadas a las respuestas de los web services (CartaSur – Veraz).

Se agregó a EIS la posibilidad de hacer un insert en la tabla de adjuntos de Engage, cuando se ejecuta una transformación aplicada a la respuesta de un método web. Hasta el momento, sólo se podían guardar tipos de datos sin estructura en tablas de las entidades (strings, numéricos, lógicos, etc). Ahora se podrán procesar los arrays de bytes que devuelvan los métodos web y guardarlos como adjuntos relacionados al proceso o al cliente. Por ejemplo, en el caso que un método web devuelva una imagen como array de bytes. También se pueden guardar textos largos que vengan como strings o imágenes codificadas base 64.

Se decidió utilizar la tabla de adjuntos de Engage y no cualquier entidad por dos motivos:

1- No se pueden definir campos BLOB en una entidad de Engage con el Designer. Y, aunque se pudiera, el motor no los utilizaría.
2- Por un tema de practicidad, dado que todo el soporte de adjuntos de Engage utiliza la tabla PHYSICAL_ATTACHED_DOCUMENT.

Para poder utilizar esta característica, simplemente hay que escribir una transformación que tenga como destino a la tabla PHYSICAL_ATTACHED_DOCUMENT. De esta manera, EIS le dará un tratamiento especial a la transformación. A diferencia de las transformaciones normales, esta transformación no necesita que los sources de los assignments se escriban con sintaxis SQL. De hecho, hay que escribir el datos de texto sin apóstrofes y no se pueden incluir funciones de SQL, como por ejemplo GETDATE(). Dada la restricción anterior, los campos TS_BEGIN y TS_END no hace falta informarlos, porque los completa automáticamente EIS.

IES procesará los inserts de entidades normales y de adjuntos en forma separada, y esto se verá reflejado en el log. Por ejemplo, si al aplicar una transformación no se generan inserts de adjuntos, aparecerá un mensaje que dice: "La ejecución <MessageReplacement> no generó sentencias insert para adjuntos.", por más que sí se hayan realizado inserts en tablas de entidades.

Los siguientes son ejemplos de transformaciones que insertan un registro dentro de PHYSICAL_ATTACHED_DOCUMENT:

```
<Transformation ID="OBTENCION_IMAGEN">
  <InPut />
  <OutPut>
    <InsertCommands>
```

```
      <Destination Table="PHYSICAL_ATTACHED_DOCUMENT">
        <Assignments>
          <Assignment Destination="ATTACHED_MESSAGE" Source="Adjunto:
%Result.Text%" />
          <Assignment Destination="FILE_NAME" Source="img001.jpg" />
          <Assignment Destination="ATTACHED_DOC" Source=D"%Result.Image%" />
          <Assignment Destination="PKEY"
Source="#GetPkey(PHYSICAL_ATTACHED_DOCUMENT)#" />
          <Assignment Destination="PAR_KEY" Source="%Message.JobPkey%" />
          <Assignment Destination="TS_USER_ID" Source="%Message.UserId%" />
        </Assignments>
      </Destination>
    </InsertCommands>
  </OutPut>
 </Transformation>

 <Transformation ID="OBTENCION_IMAGEN_CODIFICADA">
  <InPut />
  <OutPut>
    <InsertCommands>
      <Destination Table="PHYSICAL_ATTACHED_DOCUMENT">
        <Assignments>
          <Assignment Destination="ATTACHED_MESSAGE" Source="Adjunto:
%Parameter.Text%" />
          <Assignment Destination="FILE_NAME" Source="img002.jpg" />
          <Assignment Destination="ATTACHED_DOC" Source="%.%" />
          <Assignment Destination="PKEY"
Source="#GetPkey(PHYSICAL_ATTACHED_DOCUMENT)#" />
          <Assignment Destination="PAR_KEY" Source="%Message.JobPkey%" />
          <Assignment Destination="TS_USER_ID" Source="%Message.UserId%" />
        </Assignments>
      </Destination>
    </InsertCommands>
  </OutPut>
 </Transformation>

 <Transformation ID="ADJUNTAR_XML">
  <InPut />
  <OutPut>
    <InsertCommands>
      <Destination Table="PHYSICAL_ATTACHED_DOCUMENT">
        <Assignments>
          <Assignment Destination="ATTACHED_MESSAGE" Source="Adjunto:
%Parameter.FileName%" />
          <Assignment Destination="FILE_NAME" Source="%Parameter.FileName%" />
```

```
          <Assignment Destination="ATTACHED_DOC" Source="%.%" />
          <Assignment Destination="PKEY"
Source="#GetPkey(PHYSICAL_ATTACHED_DOCUMENT)#" />
          <Assignment Destination="PAR_KEY" Source="%Message.JobPkey%" />
          <Assignment Destination="TS_USER_ID" Source="%Message.UserId%" />
        </Assignments>
      </Destination>
    </InsertCommands>
  </OutPut>
 </Transformation>

 <Transformation ID="ADJUNTAR_HTML">
  <InPut />
  <OutPut>
    <XMLSerializations>
      <XMLSerialization Source=".">
        <LibraryName>ResponseVeraz</LibraryName>
        <ClassType>mensaje</ClassType>
      </XMLSerialization>
    </XMLSerializations>
    <CustomErrors SourceData="Result.estado">
      <SourceErrorNumber SourceData="Result.codigoError" />
      <SourceErrorMessage SourceData="Result.mensajeError" />
      <SuccessReturnCodes CodesList="0" />
    </CustomErrors>
    <InsertCommands>
      <Destination Table="PHYSICAL_ATTACHED_DOCUMENT"
SourceData="Result.respuesta">
        <Assignments>
          <Assignment Destination="ATTACHED_MESSAGE" Source="Adjunto:
%Result.integrante(0).documento%" />
          <Assignment Destination="FILE_NAME"
Source="%Result.integrante(0).documento%.html" />
          <Assignment Destination="ATTACHED_DOC"
Source="%Result.informe.html.risc_html.Value%" />
          <Assignment Destination="PKEY"
Source="#GetPkey(PHYSICAL_ATTACHED_DOCUMENT)#" />
          <Assignment Destination="PAR_KEY" Source="%Message.JobPkey%" />
          <Assignment Destination="TS_USER_ID" Source="%Message.UserId%" />
        </Assignments>
      </Destination>
    </InsertCommands>
  </OutPut>
 </Transformation>
```

2- **Posibilidad de utilizar imágenes background en el html del cuerpo de los emails que van a ser abiertos con un browser (Itaú)**

Hasta ahora, no estaban soportados en EIS los atributos "background" en el html del cuerpo del email, porque no todos los clientes de email lo interpretan correctamente. De hecho, al abrir el email con el OutLook las imágenes background no se ven y aparece un mensaje sugiriendo que, si hay problemas para visualizar el contenido del email, entonces hay que tratar de abrirlo con un navegador. Si al mismo archivo .eml se lo abre con un navegador, entonces las imágenes se ven correctamente. Lo que se hizo fue modificar EIS para que soporte las imágenes background en el atributo Style de cualquier elemento html. No se soporta el atributo background diréctamente, ni dentro de una plantilla de estilo .css (ni externa ni embebida dentro del html). Para poder ver las imágenes tendrán que incluirlas explícitamente en el atributo Style de cada elemento html de las siguientes formas:

1- **<ELEMENTO style="background:url(imagen.jpg) no-repeat;"/>**
2- **<ELEMENTO style="background-image:url(imagen.jpg); background-repeat: no-repeat;"/>.**

No se puede hacer lo siguiente:

<ELEMENTO background="imagen.jpg"/>

Por una restricción del parser de HTML de .Net, tampoco puede haber un espacio entre "url" y el "(".

Debe ser **"url(imagen.jpg)"**
Y no **"url (imagen.jpg)"**

A pesar de esta modificación, seguirá vigente la restricción que comento al principio. Es decir, las imágenes background las van a poder ver con un botón de vista previa de email que utilice el explorador, para luego imprimir el email. Si quieren enviar el email a través de un servidor SMTP, el destinatario es probable que no vea las imágenes.

Los siguientes son ejemplos de la utilización de imágenes background:

<td height="1" valign="bottom" style="background-image:url(linea.jpg);"><img src="grey-line.jpg" width="242" height="1"></td>

<table width=450 border=0 align=center cellPadding=0 cellSpacing=2 style="type:text/css; background:url(tarjeta.jpg) no-repeat; height:284px;" class=txt1>

<table width=450 border=0 align=center cellPadding=0 cellSpacing=2

style="type:text/css; background-image:url(tarjeta.jpg); background-repeat:no-repeat; height:284px;" class=txt1>

3- Nuevo servicio EKS, para poder iniciar un proceso Engage desde una transacción de socket de otro proceso (Galicia).

Se agregó un nuevo servicio a EIS, que se llama EKS y que se une a otros dos existentes que son del mismo tipo: EPS y ECS. Ninguno de los tres servicios necesitan licencia para ser utilizados y tampoco dejan rastros de su invocación en el archivo de log, a menos que den error.

El servicio de EKS se usa para invocar primitivas de Kernel a través de una transacción de socket (de la misma manera que EPS invoca a Passport y ECS al servicio de campañas). Por ahora, este nuevo servicio posee un único comando, pero es posible que en el futuro se agreguen otros. Dicho comando es STARTACTIVITY y el formato es el siguiente:

EKSX|STARTACTIVITY(Ticket, JobTypeCode, CustPkey [, ParentJobPkey] [, SuccessStateCode] [, MethodParameters])[EOM]

Donde:

1- **EKSX**: Nombre del servicio de EIS en modo extendido.
2- **STARTACTIVITY**: Nombre del comando (el único disponible por el momento)
3- **Ticket**: Id de sesión válida.
4- **JobTypeCode**: Código del tipo de proceso que se quiere iniciar.
5- **CustPkey**: Pkey de Customer que se quiere asociar al proceso.
6- **ParentJobPkey** (opcional): Pkey del Job padre.
7- **SuccessStateCod e** (opcional): Código del estado de cierre que se tomará como salida válida del proceso.
8- **MethodParameters** (opcional): Parámetros de inicio del proceso, con el formato: **NombrePrm1=ValorPrm1;NombrePrm2=ValorPrm2; ... ;NombrePrmN=ValorPrmN**
9- **[EOM]**: Terminador de mensaje extendido.

Por ejemplo:

EPSX|STARTACTIVITY(00001:c792fe14-7ad0-40c0-80a9-dd78bdba32ed, TEST_FORM, 0003916516_77, 6271fc29-1b1a-4242-81c6-0363f7705cf0, OK, CODIGO=123;FECHA=2013-01-31)[EOM]

Si se informa el parámetro "MethodParameters", todos los valores "ValorPrmN" se guardarán en los campos de la tabla relacionada al proceso que tengan los nombres "NombrePrmN". Para el ejemplo de arriba se espera que la tabla posea como mínimo

los campos: CODIGO y FECHA. No hace falta nombrar todos los campos de la tabla, sinó sólo los que se quieren poblar antes que se ejecute el primer paso del proceso.

SuccessStateCode se usa para comprobar que el proceso termina en un estado de cierre que se considera correcto. Por ejemplo, si un proceso tiene los estados de cierre OK, ERROR e INCOMPLETO y en el comando se informa OK, entonces el proceso se considerará válido sólo si termina en el estado OK. Si termina en cualquier otro estado, EIS devolverá el error: "El estado de salida '<código_estado>' del proceso no es válido."

Las posibles respuestas del servicio EKS son las siguientes:

1- **VALUE|<tipo_respuesta>**
2- **ERROR|<mensaje_error>**

EIS devolverá VALUE cuando el proceso haya terminado sin errores. Los valores posibles de <tipo_respuesta> son:

1- **SCRIPT**: El proceso está devolviendo una pantalla de interacción con el usuario
2- **CANCEL**: El proceso está devolviendo una pantalla de cancelación
3- **CLOSE_JOB**: El proceso está devolviendo una pantalla de cierre
4- **DERIVATE_SINGLE**: El proceso está devolviendo una pantalla de derivación simple
5- **DERIVATE_MULTIPLE**: El proceso está devolviendo una pantalla de derivación múltiple
6- **FINALIZE**: El proceso finalizó en un estado de cierre válido y distinto de los mencionados en 7, 8, 9, 10 y 11
7- **LOGOUT**: El proceso finalizó en un estado de cierre que se llama LOGOUT
8- **GOTOACTIVITIES**: El proceso finalizó en un estado de cierre que se llama GOTOACTIVITIES
9- **GOTOCUSTOMERSEARCH**: El proceso finalizó en un estado de cierre que se llama GOTOCUSTOMERSEARCH
10- **GOTOHOME**: El proceso finalizó en un estado de cierre que se llama GOTOHOME
11- **REFRESHALL**: El proceso finalizó en un estado de cierre que se llama REFRESHALL

EIS devolverá ERROR cuando:

1- Se haya producido un error al ejecutar el proceso
2- Se informó SuccessStateCode y el valor del mismo no coincide con el estado de cierre del proceso
3- El proceso devuelve una pantalla de warning (las que aparecen en un popup cuando se utiliza el agente web).

Se recomienda que el proceso Engage que se inicie con el servicio EKS quede siempre cerrado y que no necesite la interacción con el usuario. Esto es por un tema de prolijidad y para asegurarse que no aparezcan procesos pendientes en la inbox del usuario. De todas maneras, nada impide que el proceso termine de otra forma. Puede haber casos en los que sea válido que otro proceso quede iniciado en la inbox.

La ejecución del proceso Engage invocado por el servicio EKS es siempre sincrónica. Es decir, que el timeout de la transacción de socket debería contemplar el tiempo que tarda el proceso en ejecutarse. Si se necesita ejecutar en forma asincrónica, entonces se puede poner un 1 en el timeout de la transacción y contemplar el flujo de TimeOut como válido. La desventaja de esto es que no se va a poder saber cual fue el estado de finalización del proceso.

Apéndice

Práctica: Servicio Web Simple

Introducción:

Hay dos herramientas que se utilizan para probar la ejecución de un servicio web con Engage Integration Services.

a- *TCPTester*: Esta herramienta se desarrolló para uso interno, pero muchas veces es útil en una implementación para probar el armado de los mensajes de invocación, sin necesidad de crear una transacción en Engage y de un proceso que la contenga.

b- *EISProxyManager*: Se utiliza para explorar las clases proxy de una forma más amigable, sin necesidad de tener que interpretar el wsdl para conocer los métodos que contiene el servicio web o las estructuras de datos que van y vienen. Es útil también para armar las transformaciones.

Pasos para la práctica:

1- Entrar con IE a la url:

http://localhost/WSExamples20/WSExampleSimple.asmx

Explicación: Es un servicio web simple programado en VB.Net. Si lo quieren ver, el código fuente está en C:\inetpub\wwwroot\WSExamples20\App_Code, en el archivo WSExampleSimple.vb. Microsoft da la posibilidad de contar con una página automática que se utiliza para probar los servicios web. Ésta solo se presenta si se ejecuta en el servidor local al IIS. La url muestra dicha página, en la cual se muestran todos los métodos que contiene el servicio web. Si se hace click en alguno de ellos, y si todos los parámetros son por valor y de tipos de datos simples, se pueden ejecutar.

2- Entrar con IE a la url:

http://localhost/WSExamples20/WSExampleSimple.asmx?wsdl

Explicación: Esto mostrará el wsdl del servicio web.

3- Dentro de la carpeta ClientGeneration hay varios archivos .bat. Hay que duplicar el archivo GenerateSimpleClient.bat, asignándole el nombre GenerateWSE20SimpleClient.bat

4- Editar el archivo GenerateWSE20SimpleClient.bat y modificar la siguiente línea:

"..\VsSdkTools\svcutil.exe" /config:..\..\%DLLNAME%.config /l:VB /tcv:Version35 /o:.\%DLLNAME%.vb %WSDLPATH%

Para que quede así:

"..\VsSdkTools\svcutil.exe" /namespace:*,WSE20 /config:..\..\%DLLNAME%.config /l:VB /tcv:Version35 /o:.\%DLLNAME%.vb %WSDLPATH%

Explicación: Esa línea del archivo .bat es la que invoca al utilitario svcutil.exe, para generar el código fuente de la clase proxy que mapea al wsdl del servicio web. Se le está agregando el parámetro /namespace, para que la clase pertenezca al namespace WSE20.

5- Sin salir del editor, modificar la siguiente línea:

set DLLNAME=ClientWSExampleSimple

Para que quede así:

set DLLNAME=ClientWSExampleSimpleWSE20

Explicación: Se está cambiando el nombre que se le asignará a la dll.

6- Sin salir del editor, modificar la siguiente línea:

del %DLLNAME%.vb > nul

Para que quede así:

REM del %DLLNAME%.vb > nul

Explicación: Se está comentando el comando delete, que elimina el archivo .vb con el código fuente de la clase proxy. De ésta manera, se podrá inspeccionar el código fuente.

7- Salvar el archivo .bat modificado, salir del editor y ejecutarlo.

Explicación: Luego de ejecutar el .bat quedarán creados: el archivo .vb con el código fuente de la clase proxy, la dll que contiene la clase proxy compilada y el archivo .config correspondientes al servicio web. La dll y el .config se crearan en la carpeta raíz de EIS, con los nombres: ClientWSExampleSimpleWSE20.dll y ClientWSExampleSimpleWSE20.config. El archivo .vb se creará en la misma carpeta donde reside el .bat, con el nombre: ClientWSExampleSimpleWSE20.vb.

8- Editar los archivos de configuración de EIS y de EISProxyManager, para agregar los parámetros que se encuentran en el archivo .config de la dll proxy generada. Esto debe hacerse respetando la estructura. Los parámetros que hay que agregar son los siguientes:

   a- <binding name="WSExampleSimpleSoap12">…</binding>

      Que se encuentra dentro de:

      <configuration><system.serviceModel><bindings><customBinding>

   b- <endpoint … name="WSExampleSimpleSoap12" />

      Que se encuentra dentro de:

      <configuration><system.serviceModel><client>

      Explicación: Los archivos .config no se levantan automáticamente para las dlls. Sí para los .exe. Con lo cual, cualquier configuración particular de una dll debe agregarse al archivo .config del .exe que la utiliza.

9- Una vez que está creada la dll, se puede cargar con EISProxyManager y explorarla, abriendo los nodos del árbol de la derecha y haciendo click sobre los elementos que contienen.

   Explicación: Se van a ver las estructuras de datos que van y vienen y la clase que representa al servicio web. Esta última es la clase proxy propiamente dicha y se puede ver diferenciada con un pequeño mundo en el ícono. También podrá verse que tanto las estructuras de datos como la clase proxy pertenecen al namespace WSE20. Esto último se hace para evitar conflictos con otras clases proxy que se llamen igual.

10- Instanciar la clase que representa al servicio web.

   Explicación: haciendo click en el nodo correspondiente a la clase proxy, del lado derecho aparecerá una interface con el botón "Nueva instancia". Hacer click en el mismo y asignarle un nombre cualquiera.

11- Abrir el nodo "Métodos", dentro de la clase recién instanciada.

   Explicación: Esto mostrará todos los métodos disponibles, los parámetros de entrada y el tipo de dato de la respuesta

12- Hacer click en el método ObtenerDigesto

Explicación: Aparecerá un formulario con todos los parámetros de entrada y un botón de "Ejecutar método". Este método web devolverá el hash correspondiente al dato ingresado en el parámetro Data.

13- Ejecutar el método web, ingresando un dato cualquiera en el parámetro Data y haciendo click en el botón "Ejecutar método".

14- Editar el archivo EngageIntegrationServices.xml y al parámetro UsePersistence ponerlo en false

Explicación: Esto hará que EIS genere las sentencias delete e insert contra la base de datos, que las escriba en el log, pero que no las ejecute.

15- Editar el archivo .config de EIS, y poner el valor "Information" al parámetro ServiceTraceSourceSwitch

Explicación: Esto aumenta el nivel de detalle del log de servicio

16- Reiniciar el servicio de EIS

Explicación: Esto es para que tome los nuevos parámetros

17- Ejecutar TCPTester desde c:\archivos de programa\tcptester

Explicación: A diferencia de EISProxyManager, TCPTester utiliza el servicio de EIS para ejecutar métodos web. TCPTester no se incluye como parte de Engage Integration Services.

18- Conectarse a EIS con TCPTester haciendo click en el botón "Start".

Explicación: Se establece una nueva conexión TCP con EIS.

19- Del combo de mensajes, seleccionar al mensaje que invoca al método ObtenerDigesto y hacer click en copy

Explicación: Este procedimiento, copia un mensaje de ejemplo del archivo TCPTester.txt (que se puede editar) en la caja de texto superior, lo cual habilita el botón send. El contenido del mensaje es el siguiente:

WSX|ClientWSExampleSimpleWSE20|WSE20.WSExampleSimpleSoapClient|Obtener Digesto|CustPkey|JobPkey|UserId|OBTENCION_DIGESTO|Su saldo actual es de $12.575[EOM]

20- Ejecutar el método web con el botón send

Explicación: Se envía el mensaje e EIS para que ejecute el método web correspondiente. Notar que la respuesta es OK

21- Editar el archivo de log de EIS

Explicación: Notar que se escribió una sentencia insert, generada a través de la transformación OBTENCION_DIGESTO. También se escribe el mensaje que se envió desde TCPTester.

22- Ver la transformación aplicada sobre el resultado del servicio web.

Explicación: Abrir el archivo Transformations.xml y buscar el id OBTENCION_DIGESTO. Notar que se informa una tabla y los nombres de los campos de la misma a los cuales irán a parar los atributos devueltos por el servicio web.

Práctica: Servicio Web Complejo

Pasos para la práctica:

1- Dentro de la carpeta ClientGeneration hay varios archivos .bat. Hay que ejecutar GenerateComplexClient.bat para crear la clase proxy de este servicio web.

Explicación: La dll se creará en la carpeta raíz de EIS, con el nombre: ClientWSExampleComplex.dll. La clase proxy creada utiliza WCF para consumir el servicio web, por ello también se crea un archivo .config relacionado. Este archivo .config, debe editarse y agregarse la configuración que contiene al archivo .config de EIS. Los archivos .config no se levantan automáticamente para las dlls. Sí para los .exe. Con lo cual, cualquier configuración particular de una dll debe agregarse al archivo .config del .exe que la utiliza. Esto no hay que hacerlo para este ejemplo, dado que EIS sale de fábrica con las configuraciones incluidas.

2- Una vez que está creada la dll, se puede cargar con EISProxyManager y explorarla, abriendo los nodos del árbol de la derecha y haciendo click sobre los elementos que contienen.

Explicación: Se van a ver las estructuras de datos que van y vienen y la clase que representa al servicio web. Esta última es la clase proxy propiamente dicha y se puede ver diferenciada con un pequeño mundo en el ícono. También podrá verse que tanto las estructuras de datos como la clase proxy pertenecen al namespace WSE20. Esto último se hace para evitar conflictos con otras clases proxy que se llamen igual.

3- Instanciar la clase que representa al servicio web.

Explicación: haciendo click en el nodo correspondiente a la clase proxy, del lado derecho aparecerá una interface con el botón "Nueva instancia". Hacer click en el mismo y asignarle un nombre cualquiera.

4- Abrir el nodo "Métodos", dentro de la clase recién instanciada.

Explicación: Esto mostrará todos los métodos disponibles, los parámetros de entrada y el tipo de dato de la respuesta

5- Hacer click en el método ConsultarClientes

Explicación: Aparecerá un formulario con todos los parámetros de entrada y un botón de "Ejecutar método". Este método web, si no se ingresa un valor en el parámetro, devuelve una lista de todos los registros. Si se ingresa un valor, devuelve sólo el registro que corresponda

6- Ejecutar el método web de dos formas:

1- Sin informar un valor en el parámetro
2- Informando 123.

Explicación: El servicio web apunta al archivo c:\inetpub\wwwroot\WSExamples20\App_Data\WSExample.txt.

7- Explorar los resultados, haciendo click en los botones "..." para expandir información jerárquica

Explicación: Los métodos web pueden devolver estructuras de datos, tipos de datos simples, arrays, o nada. Este método web, particularmente, devuelve un array de la estructura clsResults

8- Editar el archivo EngageIntegrationServices.xml y al parámetro UsePersistence ponerlo en false

Explicación: Esto hará que EIS genere las sentencias delete e insert contra la base de datos, que las escriba en el log, pero que no las ejecute.

9- Editar el archivo .config de EIS, y poner el valor "Information" a los parámetros: ServiceTraceSourceSwitch y PersistenceTraceSourceSwitch.

Explicación: Esto aumenta el nivel de detalle del log de servicio y del log de persistencia.

10- Reiniciar el servicio de EIS

Explicación: Esto es para que tome los nuevos parámetros

11- Ejecutar TCPTester desde c:\archivos de programa\tcptester

Explicación: A diferencia de EISProxyManager, TCPTester utiliza el servicio de EIS para ejecutar métodos web.

12- Conectarse a EIS con TCPTester haciendo click en el botón "Start".

Explicación: Se establece una nueva conexión TCP con EIS.

13- Del combo de mensajes, seleccionar al mensaje que invoca al método ConsultarClientes y hacer click en copy

Explicación: Este procedimiento, copia un mensaje de ejemplo del archivo TCPTester.txt (que se puede editar) en la caja de texto superior, lo cual habilita el botón send. El contenido del mensaje es el siguiente:

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|ConsultarClientes|CustPkey|JobPkey|UserId|CONSULTA_DE_CLIENTES|[EOM]

14- Ejecutar el método web con el botón send

Explicación: Se envia el mensaje e EIS para que ejecute el método web correspondiente. Notar que la respuesta es OK

15- Editar el archivo de log de EIS

Explicación: Notar que se escribieron una sentencia delete y varias insert por cada registro recuperado del servicio web. También se escribe el mensaje que envió TCPTester.

16- Con Designer, crear un nuevo proceso global con:

Código="TMT_CONSULTA_CLIENTE"
Descripción="Consulta de clientes"
Unidad="Desarrollo"

El resto de los parámetros hay que dejarlos como están.

17- Crear un estado del proceso que se llame "FIN".

18- Crear una entidad del proceso con:

Descripción="Datos de clientes"
Tabla Relacionada="DATOS_CLIENTE"
Cardinalidad="1 a Muchos"

La entidad debe contener los siguientes campos:

CUST_PRIMARY_ID=TEXTO (Longitud 100, Atributo de acceso)
SITUACION=NUMERO (Longitud 2, Decimales 0)
DEUDA=NUMERO (Longitud 10, Decimales 2)
CUSTOMER_NAME=TEXTO (Longitud 200)
TELEFONO=TEXTO (Longitud 100)
EMAIL=TEXTO (Longitud 100)

19- Validar la entidad creada

Explicación: Este paso valida que no hayan errores en la definición y además crea la tabla física.

20- Crear una transacción de socket global con:

Código="TRX_CONSULTA_CLI"
Descripción="Consultar clientes"
TimeOut(seg)="5"
Número de Puerto="7000"
Ubicación de Host="Local al Server"

21- Definir los siguientes parámetros de la transacción:

INPUT:

1- (Fijo) WSX|ClientWSExampleComplex|
2- (Fijo) WSE20.WSExampleComplexSoapClient|ConsultarClientes|
3- (Variable) Atributo fijo (Cliente), Clave del Cliente
4- (Fijo) |
5- (Variable) Atributo fijo (Proceso), Clave del Proceso
6- (Fijo) |
7- (Variable) Atributo fijo (Actividad), Agente
8- (Fijo) |CONSULTA_DE_CLIENTES|[EOM]

OUTPUT:

1- CODIGO (Largo 5)
2- SEPARADOR (Largo 1)
3- DESCRIPCION (Largo 1000)

Explicación: Los parámetros de input que sean variables se reemplazarán por el valor correspondiente durante la ejecución del proceso. Luego, todos los parámetros se concatenarán en un sólo string final. Por el contrario, los parámetros de output se poblarán luego de parsear el mensaje de vuelta, tratándolo como un string de ancho fijo.

22- Crear un procedimiento almacenado global con:

Código="SP_CONSULTAR_CLIENTE"
Referencia a="SP_CONSULTAR_CLIENTES"
Nombre="Consultar clientes", TimeOut(seg)="5".

23- Definir un parámetro de entrada del procedimiento almacenado con: Nombre="USERID", Descripción="User ID", Tipo de dato="VarChar", Largo="30", Tipo="(Variable) Atributo fijo (Actividad), Agente"

24- Crear el procedimiento almacenado en la base de datos. Deberá ejecutar un select * sobre la entidad DATOS_CLIENTE con where TS_USER_ID=@USERID

El script para crear el SP quedaría así:

```
CREATE PROCEDURE SP_CONSULTAR_CLIENTES (@USERID VARCHAR(50))
AS
BEGIN
    SELECT * FROM DATOS_CLIENTE WHERE TS_USER_ID = @USERID
END
```

25- Crear una actividad del proceso con:

Código="ACT_CONSULTA_CLIENTE"
Descripción="Consulta de clientes"
Inbound/Outbound="AMBOS"

El resto de los parámetros hay que dejarlos como están.

26- Crear un estado cerrado de la actividad con: Resultado="FIN", Descripción="Fin"

27- Crear una nueva estructura de la actividad con:

Nombre="EST_CONSULTA_CLIENTE".

28- Editar el grafo asociado a la estructura, para integrar la ejecución de la transacción de socket y asociarla con tres pantallas de Engage. Deben contemplarse los flujos de salida de la transacción por error y timeout, pero no el de ok. En su lugar, hay que crear dos flujos que evalúen las siguientes fórmulas:

1- Fórmula: TRX_CONSULTA_CLI.CODIGO = 'OK'
   Descripción: OK

2- Fórmula: NOT TRX_CONSULTA_CLI.CODIGO = 'OK'
   Descripción: ERROR

Los flujos de error y timeout deben conectarse a una pantalla de Engage que advierta que se produjo un error de socket.

El flujo de la primer fórmula debe conectarse a una pantalla Engage que mostrará los datos devueltos por EIS.

El flujo de la segunda fórmula debe conectarse a una pantalla Engage que muestre el error de ejecución de EIS.

29- Dentro de la pantalla Engage que recibe los flujos de error y timeout, agregar un label con la descripción del error y un botón de continuar.

30- Dentro de la pantalla Engage que recibe el flujo OK, agregar una tabla que ejecute el procedimiento almacenado y un botón de continuar.

31- Dentro de la pantalla Engage que recibe el flujo ERROR, agregar un parámetro de transacción asociado a la descripción del error y un botón de continuar.

32- Editar el diseño del proceso, para asociar la actividad y el estado de cierre de la actividad.

33- Validar el proceso para que quede vigente y se pueda ejecutar.

34- Activar la persistencia en el archivo EngageIntegrationServices.xml

Explicación: Poner en true el parámetro UsePersistence

35- Reiniciar EIS

Explicación: Asegurarse que TCPTester está desconectado, de lo contrario el reinicio del servicio se demorará por lo menos un minuto

36- Con Security Administrator, dar permisos a la unidad Desarrollo sobre el proceso, la actividad y la entidad del proceso.

37- Ejecutar el proceso Engage.

Práctica: Final

Introducción:

Estamos en un proyecto en el que se necesita construir una integración con un servicio web. El proyecto está en una etapa en la que se necesita avanzar con la integración, pero el servicio web todavía no está desarrollado.

La única información con la que contamos es que el servicio web tendrá un sólo método web, y ese método web se utilizará para invocar a todas las funciones posibles del servicio web. Además, el método web tendrá un solo parámetro de entrada de tipo string y devolverá también un string como respuesta. Ambos strings, tanto el de ida como el de vuelta, son datos con formato XML.

Cuando nosotros les solicitamos mayores detalles a los responsables del servicio web, nos dicen que lo único que tienen resuelto hasta el momento y que nos pueden facilitar es un XML de ejemplo con la respuesta. De esta manera, nos envían el archivo RespuestaAsConsulta.xml.

Por otro lado, nuestro Líder de Proyecto nos dice que todavía no está definida la información que se necesita extractar de la respuesta del método web, con lo cual, nos pide que volquemos absolutamente todos los datos en tablas. Esas tablas todavía no existen, y por ello no tenemos idea de cuales serán los nombres de las tablas ni de los campos. Tampoco sabemos cuales serán los tipos de dato de cada campo, pero nuestro Líder de Proyecto nos pide que por ahora tratemos todo como string.

Pasos para la práctica:

1- Obtener el XSD del XML de ejemplo.

2- Corregir el XSD de serialización, controlando que los elementos definidos como arrays lo sean en realidad.

3- Crear la dll de serialización a partir del XSD corregido.

4- Comenzar a construir la transformación, utilizando nombres tentativos para las tablas y los campos.

5- La transformación estará asociada al proceso.

6- Simular el método web en un servicio web de prueba, para poder probar la transformación, mientras vemos que las sentencias insert se crean correctamente.

7- Probar las variantes del mensaje de respuesta que se reciban con o sin errores.

Mensajes de invocación de ejemplo para el servicio web simple:

WSX|ClientWSExampleSimple|WSExampleSimpleSoapClient|FechaActual|CustPkey|JobPkey|UserId|OBTENCION_FECHA_ACTUAL[EOM]

WSX|ClientWSExampleSimple|WSExampleSimpleSoapClient|DiaSemana|CustPkey|JobPkey|UserId|OBTENCION_DIA_SEMANA[EOM]

WSX|ClientWSExampleSimple|WSExampleSimpleSoapClient|ComponerFecha|CustPkey|JobPkey|UserId|COMPOSICION_DE_FECHA|06|08|1970[EOM]

WSX|ClientWSExampleSimple|WSExampleSimpleSoapClient|TextoFecha|CustPkey|JobPkey|UserId|OBTENCION_TEXTO_FECHA|1970-08-06[EOM]

WSX|ClientWSExampleSimple|WSExampleSimpleSoapClient|ObtenerDigesto|CustPkey|JobPkey|UserId|OBTENCION_DIGESTO|Su saldo actual es de $12.575[EOM]

WSX|ClientWSExampleSimple|WSExampleSimpleSoapClient|ValidarDigesto|CustPkey|JobPkey|UserId|VALIDAR_DIGESTO|Su saldo actual es de $12.575|72834E38C38D075954B6F99EC671EEEA56AA0EAE[EOM]

WSX|ClientWSExampleSimple|WSExampleSimpleSoapClient|ObtenerIMC|CustPkey|JobPkey|UserId|OBTENCION_IMC|97.5|1.87[EOM]

WSX|ClientWSExampleSimple|WSExampleSimpleSoapClient|EvaluarIMC|CustPkey|JobPkey|UserId|EVALUAR_IMC|27.881838199548167[EOM]

WSX|ClientWSExampleSimple|WSExampleSimpleSoapClient|ObtenerPGC|CustPkey|JobPkey|UserId|OBTENCION_PGC|27.881838199548167|42|Masculino[EOM]

WSX|ClientWSExampleSimple|WSExampleSimpleSoapClient|EvaluarPGC|CustPkey|JobPkey|UserId|EVALUAR_PGC|26.918205839457805|Masculino[EOM]

Mensajes de invocación de ejemplo para el servicio web complejo:

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|BuscarNombre|CustPkey|JobPkey|UserId|BUSQUEDA_DE_NOMBRE|123[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|PaginarNombres|CustPkey|JobPkey|UserId|PAGINACION_DE_NOMBRES_SIMPLE|2|5[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|PaginarNombres|CustPkey|JobPkey|UserId|PAGINACION_DE_NOMBRES_SIMPLE|0|5[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|PaginarNombres|CustPkey|JobPkey|UserId|PAGINACION_DE_NOMBRES_COMPLETA|2|5[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|PaginarNombres|CustPkey|JobPkey|UserId|PAGINACION_DE_NOMBRES_COMPLETA|0|5[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|BuscarCliente|CustPkey|JobPkey|UserId|CONSULTA_DE_CLIENTES|[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|BuscarCliente|CustPkey|JobPkey|UserId|CONSULTA_DE_CLIENTES|123[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|ConsultarClientes|CustPkey|JobPkey|UserId|CONSULTA_DE_CLIENTES|[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|ConsultarClientes|CustPkey|JobPkey|UserId|CONSULTA_DE_CLIENTES|123[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|ListarClientes|CustPkey|JobPkey|UserId|LISTADO_DE_CLIENTES|2|5[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|PaginarClientes|CustPkey|JobPkey|UserId|PAGINACION_DE_CLIENTES|2|5[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|PaginarClientes|CustPkey|JobPkey|UserId|PAGINACION_DE_CLIENTES|0|5[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|PaginarClientes|CustPkey|JobPkey|UserId|ESTADO_EJECUCION_PAGINACION|2|5[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|PaginarClientes|CustPkey|JobPkey|UserId|ESTADO_EJECUCION_PAGINACION|0|5[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|ConsultarClientes|CustPkey|JobPkey|UserId|CONSULTA_DE_CONTACTOS|[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|ConsultarClientes|CustPkey|JobPkey|UserId|CONSULTA_DE_CONTACTOS|123[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|ExtraerClientes|CustPkey|JobPkey|UserId|EXTRACCION_DE_CLIENTES|[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|ExtraerClientes|CustPkey|JobPkey|UserId|EXTRACCION_DE_CLIENTES|123[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|ConsultarClientesXML|CustPkey|JobPkey|UserId|CONSULTA_DE_CLIENTES_XML|[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|ConsultarClientesXML|CustPkey|JobPkey|UserId|CONSULTA_DE_CLIENTES_XML|123[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|AgregarCliente|CustPkey|JobPkey|UserId||Cust_Primary_Id=123;Situacion=1;Deuda=5000.00;Customer_Name=PEREZ, JUAN JOSE|ControlarDuplicados[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|InsertarCliente|CustPkey|JobPkey|UserId||Cust_Primary_Id=123;Situacion=1;Deuda=5000.00;Customer_Name=PEREZ, JUAN JOSE|ControlarDuplicados[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|AgregarCliente|CustPkey|JobPkey|UserId||<?xml version="1.0" encoding="utf-8"?><clsResults xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:t="http://tempuri.org/"><t:Cust_Primary_Id>123</t:Cust_Primary_Id><t:Situacion>1</t:Situacion><t:Deuda>5000</t:Deuda><t:Customer_Name>PEREZ, JUAN JOSE</t:Customer_Name><t:Contactos><t:clsContacto><t:Telefonos><t:string>1234-5678</t:string><t:string>2345-6789</t:string></t:Telefonos><t:Emails><t:string>email1@gmail.com</t:string><t:string>email2@gmail.com</t:string></t:Emails></t:clsContacto><t:clsContacto><t:Telefonos><t:string>15-1617-819</t:string></t:Telefonos><t:Emails><t:string>email3@yahoo.com</t:string><t:string>email4@yahoo.com</t:string></t:Emails></t:clsContacto></t:Contactos></clsResults>|ControlarDuplicados[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|GetXmlFileAsString|CustPkey|JobPkey|UserId|CONSULTA_VERAZ_DEBUG|ResponseVeraz.xml[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|GetXmlFileAsString|CustPkey|JobPkey|UserId|CONSULTA_VERAZ_CUST|ResponseVeraz.xml[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|GetXmlFileAsString|CustPkey|JobPkey|UserId|CONSULTA_VERAZ_JOB|ResponseVeraz.xml[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|GetXmlFileAsString|CustPkey|JobPkey|UserId|CONSULTA_MENSAJE_210|RespuestaMensaje0210.xml[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|GetXmlFileAsString|CustPkey|JobPkey|UserId|CONSULTA_MENSAJE_210|RespuestaErrorMensaje0210.xml[EOM]

WSX|ClientWSExampleComplex|WSE20.WSExampleComplexSoapClient|GetXmlFileAsString|CustPkey|JobPkey|UserId|CONSULTA_MENSAJE_210|ResponseVeraz.xml[EOM]

Transformaciones de ejemplo para el servicio web simple:

```
<Transformation ID="OBTENCION_FECHA_ACTUAL">
  <InPut />
  <OutPut>
    <InsertCommands>
      <Destination Table="DATOS_FECHA">
        <Assignments>
          <Assignment Destination="FECHA_ACTUAL"
                      Source="CAST('%.%' AS DATETIME)" />
          <Assignment Destination="PKEY" Source="'#GetPkey(DATOS_FECHA)#'" />
          <Assignment Destination="PAR_KEY" Source="'%Message.CustPkey%'" />
          <Assignment Destination="TS_BEGIN" Source="Getdate()" />
          <Assignment Destination="TS_END" Source="Getdate()" />
          <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
        </Assignments>
      </Destination>
    </InsertCommands>
  </OutPut>
</Transformation>

<Transformation ID="OBTENCION_DIA_SEMANA">
  <InPut />
  <OutPut>
    <InsertCommands>
      <Destination Table="DATOS_FECHA">
        <Assignments>
          <Assignment Destination="DIA_SEMANA" Source="'%.%'" />
          <Assignment Destination="PKEY" Source="'#GetPkey(DATOS_FECHA)#'" />
          <Assignment Destination="PAR_KEY" Source="'%Message.CustPkey%'" />
          <Assignment Destination="TS_BEGIN" Source="Getdate()" />
          <Assignment Destination="TS_END" Source="Getdate()" />
          <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
        </Assignments>
      </Destination>
    </InsertCommands>
  </OutPut>
</Transformation>

<Transformation ID="COMPOSICION_DE_FECHA">
  <InPut />
  <OutPut>
```

```
    <InsertCommands>
      <Destination Table="DATOS_FECHA">
        <Assignments>
          <Assignment Destination="FECHA" Source="CAST('%.%' AS DATETIME)" />
          <Assignment Destination="PKEY" Source="'#GetPkey(DATOS_FECHA)#'" />
          <Assignment Destination="PAR_KEY" Source="'%Message.CustPkey%'" />
          <Assignment Destination="TS_BEGIN" Source="Getdate()" />
          <Assignment Destination="TS_END" Source="Getdate()" />
          <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
        </Assignments>
      </Destination>
    </InsertCommands>
  </OutPut>
</Transformation>

<Transformation ID="OBTENCION_TEXTO_FECHA">
  <InPut />
  <OutPut>
    <InsertCommands>
      <Destination Table="DATOS_FECHA">
        <Assignments>
          <Assignment Destination="TEXTO_FECHA" Source="'%.%'" />
          <Assignment Destination="PKEY" Source="'#GetPkey(DATOS_FECHA)#'" />
          <Assignment Destination="PAR_KEY" Source="'%Message.CustPkey%'" />
          <Assignment Destination="TS_BEGIN" Source="Getdate()" />
          <Assignment Destination="TS_END" Source="Getdate()" />
          <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
        </Assignments>
      </Destination>
    </InsertCommands>
  </OutPut>
</Transformation>

<Transformation ID="OBTENCION_DIGESTO">
  <InPut />
  <OutPut>
    <DeleteCommands>
      <Destination Table="DATOS_DIGESTO">
        <DeleteFilters>
          <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
        </DeleteFilters>
      </Destination>
    </DeleteCommands>
    <InsertCommands>
      <Destination Table="DATOS_DIGESTO">
```

```
      <Assignments>
        <Assignment Destination="DATO" Source="'%Parameter.Data%'" />
        <Assignment Destination="DIGESTO" Source="'%.%'" />
        <Assignment Destination="PKEY"
                    Source="'#GetPkey(DATOS_DIGESTO)#'" />
        <Assignment Destination="PAR_KEY" Source="'%Message.JobPkey%'" />
        <Assignment Destination="TS_BEGIN" Source="Getdate()" />
        <Assignment Destination="TS_END" Source="Getdate()" />
        <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
      </Assignments>
    </Destination>
  </InsertCommands>
 </OutPut>
</Transformation>

<Transformation ID="VALIDAR_DIGESTO">
 <InPut />
 <OutPut>
  <DeleteCommands>
   <Destination Table="CONTROL_DIGESTO">
    <DeleteFilters>
     <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
    </DeleteFilters>
   </Destination>
  </DeleteCommands>
  <InsertCommands>
   <Destination Table="CONTROL_DIGESTO">
    <Assignments>
     <Assignment Destination="DATO" Source="'%Parameter.Data%'" />
     <Assignment Destination="DIGESTO" Source="'%Parameter.Hash%'" />
     <Assignment Destination="ES_VALIDO" Source="%.%" />
     <Assignment Destination="PKEY"
                 Source="'#GetPkey(CONTROL_DIGESTO)#'" />
     <Assignment Destination="PAR_KEY" Source="'%Message.JobPkey%'" />
     <Assignment Destination="TS_BEGIN" Source="Getdate()" />
     <Assignment Destination="TS_END" Source="Getdate()" />
     <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
    </Assignments>
   </Destination>
  </InsertCommands>
 </OutPut>
</Transformation>

<Transformation ID="OBTENCION_IMC">
 <InPut />
```

```
  <OutPut>
    <DeleteCommands>
      <Destination Table="DATOS_IMC">
        <DeleteFilters>
          <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
        </DeleteFilters>
      </Destination>
    </DeleteCommands>
    <InsertCommands>
      <Destination Table="DATOS_IMC">
        <Assignments>
          <Assignment Destination="PESO"
                      Source="%Parameter.Peso%" Default="0.0" />
          <Assignment Destination="ALTURA"
                      Source="%Parameter.Altura%" Default="0.0" />
          <Assignment Destination="IMC" Source="%.%" Default="0.0" />
          <Assignment Destination="PKEY" Source="'#GetPkey(DATOS_IMC)#'" />
          <Assignment Destination="PAR_KEY" Source="'%Message.JobPkey%'" />
          <Assignment Destination="TS_BEGIN" Source="Getdate()" />
          <Assignment Destination="TS_END" Source="Getdate()" />
          <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
        </Assignments>
      </Destination>
    </InsertCommands>
  </OutPut>
</Transformation>

<Transformation ID="EVALUAR_IMC">
  <InPut />
  <OutPut>
    <DeleteCommands>
      <Destination Table="CONTROL_IMC">
        <DeleteFilters>
          <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
        </DeleteFilters>
      </Destination>
    </DeleteCommands>
    <InsertCommands>
      <Destination Table="CONTROL_IMC">
        <Assignments>
          <Assignment Destination="IMC"
                      Source="%Parameter.IMC%" Default="0.0" />
          <Assignment Destination="ESTADO_IMC" Source="'%.%'" />
          <Assignment Destination="PKEY" Source="'#GetPkey(CONTROL_IMC)#'" />
          <Assignment Destination="PAR_KEY" Source="'%Message.JobPkey%'" />
```

```
        <Assignment Destination="TS_BEGIN" Source="Getdate()" />
        <Assignment Destination="TS_END" Source="Getdate()" />
        <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
      </Assignments>
    </Destination>
  </InsertCommands>
 </OutPut>
</Transformation>

<Transformation ID="OBTENCION_PGC">
 <InPut />
 <OutPut>
  <DeleteCommands>
   <Destination Table="DATOS_PGC">
    <DeleteFilters>
     <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
    </DeleteFilters>
   </Destination>
  </DeleteCommands>
  <InsertCommands>
   <Destination Table="DATOS_PGC">
    <Assignments>
     <Assignment Destination="IMC"
                 Source="%Parameter.IMC%" Default="0.0" />
     <Assignment Destination="EDAD"
                 Source="%Parameter.Edad%" Default="0" />
     <Assignment Destination="SEXO" Source="'%Parameter.Sexo%'" />
     <Assignment Destination="PGC" Source="%.%" Default="0.0" />
     <Assignment Destination="PKEY" Source="'#GetPkey(DATOS_PGC)#'" />
     <Assignment Destination="PAR_KEY" Source="'%Message.JobPkey%'" />
     <Assignment Destination="TS_BEGIN" Source="Getdate()" />
     <Assignment Destination="TS_END" Source="Getdate()" />
     <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
    </Assignments>
   </Destination>
  </InsertCommands>
 </OutPut>
</Transformation>

<Transformation ID="EVALUAR_PGC">
 <InPut />
 <OutPut>
  <DeleteCommands>
   <Destination Table="CONTROL_PGC">
    <DeleteFilters>
```

```
        <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
      </DeleteFilters>
    </Destination>
  </DeleteCommands>
  <InsertCommands>
    <Destination Table="CONTROL_PGC">
      <Assignments>
        <Assignment Destination="PGC"
                    Source="%Parameter.PGC%" Default="0.0" />
        <Assignment Destination="SEXO" Source="'%Parameter.Sexo%'" />
        <Assignment Destination="ESTADO_PGC" Source="'%.%'" />
        <Assignment Destination="PKEY" Source="'#GetPkey(CONTROL_PGC)#'" />
        <Assignment Destination="PAR_KEY" Source="'%Message.JobPkey%'" />
        <Assignment Destination="TS_BEGIN" Source="Getdate()" />
        <Assignment Destination="TS_END" Source="Getdate()" />
        <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
      </Assignments>
    </Destination>
  </InsertCommands>
 </OutPut>
</Transformation>
```

Transformaciones de ejemplo para el servicio web complejo:

```
<Transformation ID="BUSQUEDA_DE_NOMBRE">
 <InPut />
 <OutPut>
  <DeleteCommands>
   <Destination Table="DATOS_CLIENTE">
    <DeleteFilters>
     <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
    </DeleteFilters>
   </Destination>
  </DeleteCommands>
  <InsertCommands>
   <Destination Table="DATOS_CLIENTE">
    <Assignments>
     <Assignment Destination="CUSTOMER_NAME" Source="'%.%'" />
     <Assignment Destination="PKEY" Source="'#GetPkey(DATOS_CLIENTE)#'" />
     <Assignment Destination="PAR_KEY" Source="'%Message.CustPkey%'" />
     <Assignment Destination="TS_BEGIN" Source="Getdate()" />
     <Assignment Destination="TS_END" Source="Getdate()" />
     <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
    </Assignments>
   </Destination>
```

```
    </InsertCommands>
  </OutPut>
</Transformation>

<Transformation ID="PAGINACION_DE_NOMBRES_SIMPLE">
  <InPut />
  <OutPut>
    <DeleteCommands>
      <Destination Table="DATOS_CLIENTE">
        <DeleteFilters>
          <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
        </DeleteFilters>
      </Destination>
    </DeleteCommands>
    <InsertCommands>
      <Destination Table="DATOS_CLIENTE">
        <Assignments>
          <Assignment Destination="CUSTOMER_NAME" Source="'%.%'" />
          <Assignment Destination="PKEY" Source="'#GetPkey(DATOS_CLIENTE)#'" />
          <Assignment Destination="PAR_KEY" Source="'%Message.CustPkey%'" />
          <Assignment Destination="TS_BEGIN" Source="Getdate()" />
          <Assignment Destination="TS_END" Source="Getdate()" />
          <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
        </Assignments>
      </Destination>
    </InsertCommands>
  </OutPut>
</Transformation>

<Transformation ID="PAGINACION_DE_NOMBRES_COMPLETA">
  <InPut />
  <OutPut>
    <CustomErrors SourceData="Result(0)">
      <SourceErrorNumber SourceData="." />
      <SourceErrorMessage SourceData="." />
      <ErrorReturnCodes CodesList="Error%" />
    </CustomErrors>
    <DeleteCommands>
      <Destination Table="DATOS_PAGINACION">
        <DeleteFilters>
          <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
        </DeleteFilters>
      </Destination>
      <Destination Table="DATOS_CLIENTE">
        <DeleteFilters>
```

```
        <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
      </DeleteFilters>
    </Destination>
   </DeleteCommands>
   <InsertCommands>
    <Destination Table="DATOS_PAGINACION" SourceData="Result(0)">
     <Assignments>
      <Assignment Destination="DESDE_PAGINA"
                  Source="%Parameter.DesdePagina%" Default="0" />
      <Assignment Destination="CANTIDAD"
                  Source="%Parameter.Cantidad%" Default="0" />
      <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
     </Assignments>
    </Destination>
    <Destination Table="DATOS_CLIENTE">
     <Assignments>
      <Assignment Destination="CUSTOMER_NAME" Source="'%.%'" />
      <Assignment Destination="PKEY" Source="'#GetPkey(DATOS_CLIENTE)#'" />
      <Assignment Destination="PAR_KEY" Source="'%Message.CustPkey%'" />
      <Assignment Destination="TS_BEGIN" Source="Getdate()" />
      <Assignment Destination="TS_END" Source="Getdate()" />
      <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
     </Assignments>
    </Destination>
   </InsertCommands>
  </OutPut>
</Transformation>

<Transformation ID="CONSULTA_DE_CLIENTES">
  <InPut />
  <OutPut>
   <DeleteCommands>
    <Destination Table="DATOS_CLIENTE">
     <DeleteFilters>
      <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
     </DeleteFilters>
    </Destination>
   </DeleteCommands>
   <InsertCommands>
    <Destination Table="DATOS_CLIENTE">
     <Assignments>
      <Assignment Destination="CUST_PRIMARY_ID"
                  Source="'%Result.Cust_Primary_Id%'" />
      <Assignment Destination="SITUACION"
                  Source="%Result.Situacion%" Default="0" />
```

```
        <Assignment Destination="DEUDA"
                    Source="%Result.Deuda%" Default="0.0" />
        <Assignment Destination="CUSTOMER_NAME"
                    Source="'%Result.Customer_Name%'" />
        <Assignment Destination="TELEFONO"
                    Source="'%Result.Contactos(0).Telefonos(0)%'" />
        <Assignment Destination="EMAIL"
                    Source="'%Result.Contactos(0).Emails(0)%'" />
        <Assignment Destination="PKEY" Source="'#GetPkey(DATOS_CLIENTE)#'" />
        <Assignment Destination="PAR_KEY" Source="'%Message.CustPkey%'" />
        <Assignment Destination="TS_BEGIN" Source="Getdate()" />
        <Assignment Destination="TS_END" Source="Getdate()" />
        <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
      </Assignments>
    </Destination>
  </InsertCommands>
 </OutPut>
</Transformation>

<Transformation ID="LISTADO_DE_CLIENTES">
 <InPut />
 <OutPut>
  <DeleteCommands>
   <Destination Table="DATOS_PAGINACION">
    <DeleteFilters>
     <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
    </DeleteFilters>
   </Destination>
   <Destination Table="DATOS_CLIENTE">
    <DeleteFilters>
     <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
    </DeleteFilters>
   </Destination>
  </DeleteCommands>
  <InsertCommands>
   <Destination Table="DATOS_PAGINACION" SourceData="Result(0)">
    <Assignments>
     <Assignment Destination="DESDE_PAGINA"
                 Source="%Parameter.DesdePagina%" Default="0" />
     <Assignment Destination="CANTIDAD"
                 Source="%Parameter.Cantidad%" Default="0" />
     <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
    </Assignments>
   </Destination>
   <Destination Table="DATOS_CLIENTE">
```

```
      <Assignments>
        <Assignment Destination="CUST_PRIMARY_ID"
                    Source="'%Result.Cust_Primary_Id%'" />
        <Assignment Destination="SITUACION"
                    Source="%Result.Situacion%" Default="0" />
        <Assignment Destination="DEUDA"
                    Source="%Result.Deuda%" Default="0.0" />
        <Assignment Destination="CUSTOMER_NAME"
                    Source="'%Result.Customer_Name%'" />
        <Assignment Destination="TELEFONO"
                    Source="'%Result.Contactos(0).Telefonos(0)%'" />
        <Assignment Destination="EMAIL"
                    Source="'%Result.Contactos(0).Emails(0)%'" />
        <Assignment Destination="PKEY" Source="'#GetPkey(DATOS_CLIENTE)#'" />
        <Assignment Destination="PAR_KEY" Source="'%Message.CustPkey%'" />
        <Assignment Destination="TS_BEGIN" Source="Getdate()" />
        <Assignment Destination="TS_END" Source="Getdate()" />
        <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
      </Assignments>
    </Destination>
  </InsertCommands>
 </OutPut>
</Transformation>

<Transformation ID="PAGINACION_DE_CLIENTES">
 <InPut />
 <OutPut>
  <CustomErrors SourceData="Result.Error">
   <SourceErrorNumber SourceData="Result.ErrorCode" />
   <SourceErrorMessage SourceData="Result.ErrorDescription" />
   <SuccessReturnCodes CodesList=" " />
  </CustomErrors>
  <DeleteCommands>
   <Destination Table="DATOS_PAGINACION">
    <DeleteFilters>
     <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
    </DeleteFilters>
   </Destination>
   <Destination Table="DATOS_CLIENTE">
    <DeleteFilters>
     <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
    </DeleteFilters>
   </Destination>
   <Destination Table="TELEFONOS_CLIENTE">
    <DeleteFilters>
```

```
      <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
    </DeleteFilters>
  </Destination>
  <Destination Table="EMAILS_CLIENTE">
    <DeleteFilters>
      <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
    </DeleteFilters>
  </Destination>
</DeleteCommands>
<InsertCommands>
  <Destination Table="DATOS_PAGINACION">
    <Assignments>
      <Assignment Destination="DESDE_PAGINA"
                  Source="%Parameter.DesdePagina%" Default="0" />
      <Assignment Destination="CANTIDAD"
                  Source="%Parameter.Cantidad%" Default="0" />
      <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
    </Assignments>
  </Destination>
  <Destination Table="DATOS_CLIENTE" SourceData="Result.Data">
    <Declarations>
      <Declaration Variable="PKEY"
                  Source="#GetPkey(DATOS_CLIENTE)#" AutoRefresh="True" />
    </Declarations>
    <Assignments>
      <Assignment Destination="CUST_PRIMARY_ID"
                  Source="'%Result.Cust_Primary_Id%'" />
      <Assignment Destination="SITUACION"
                  Source="%Result.Situacion%" Default="0" />
      <Assignment Destination="DEUDA"
                  Source="%Result.Deuda%" Default="0.0" />
      <Assignment Destination="CUSTOMER_NAME"
                  Source="'%Result.Customer_Name%'" />
      <Assignment Destination="TELEFONO"
                  Source="'%Result.Contactos(0).Telefonos(0)%'" />
      <Assignment Destination="EMAIL"
                  Source="'%Result.Contactos(0).Emails(0)%'" />
      <Assignment Destination="PKEY" Source="'%Variable.PKEY%'" />
      <Assignment Destination="PAR_KEY" Source="'%Message.CustPkey%'" />
      <Assignment Destination="TS_BEGIN" Source="Getdate()" />
      <Assignment Destination="TS_END" Source="Getdate()" />
      <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
    </Assignments>
    <RelatedDestinations>
      <Destination SourceData="Result.Contactos">
```

```
        <RelatedDestinations>
          <Destination Table="TELEFONOS_CLIENTE"
                       SourceData="Result.Telefonos">
            <Assignments>
              <Assignment Destination="TELEFONO" Source="'%.%'" />
              <Assignment Destination="PKEY"
                          Source="'#GetPkey(TELEFONOS_CLIENTE)#'" />
              <Assignment Destination="PAR_KEY" Source="'%Variable.PKEY%'" />
              <Assignment Destination="TS_BEGIN" Source="Getdate()" />
              <Assignment Destination="TS_END" Source="Getdate()" />
              <Assignment Destination="TS_USER_ID"
                          Source="'%Message.UserId%'" />
            </Assignments>
          </Destination>
          <Destination Table="EMAILS_CLIENTE" SourceData="Result.Emails">
            <Assignments>
              <Assignment Destination="EMAIL" Source="'%.%'" />
              <Assignment Destination="PKEY"
                          Source="'#GetPkey(EMAIL_CLIENTE)#'" />
              <Assignment Destination="PAR_KEY" Source="'%Variable.PKEY%'" />
              <Assignment Destination="TS_BEGIN" Source="Getdate()" />
              <Assignment Destination="TS_END" Source="Getdate()" />
              <Assignment Destination="TS_USER_ID"
                          Source="'%Message.UserId%'" />
            </Assignments>
          </Destination>
        </RelatedDestinations>
      </Destination>
    </RelatedDestinations>
  </Destination>
 </InsertCommands>
</OutPut>
</Transformation>

<Transformation ID="ESTADO_EJECUCION_PAGINACION">
 <InPut />
 <OutPut SourceData="Result.Error">
  <CustomErrors>
   <SourceErrorNumber SourceData="Result.ErrorCode" />
   <SourceErrorMessage SourceData="Result.ErrorDescription" />
   <SuccessReturnCodes CodesList=" " />
  </CustomErrors>
 </OutPut>
</Transformation>
```

```
<Transformation ID="CONSULTA_DE_CONTACTOS">
  <InPut />
  <OutPut>
    <DeleteCommands>
      <Destination Table="CONTACTOS_CLIENTE">
        <DeleteFilters>
          <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
        </DeleteFilters>
      </Destination>
    </DeleteCommands>
    <InsertCommands>
      <Destination>
        <Declarations>
          <Declaration Variable="CUST_PRIMARY_ID"
                       SourceData="Result.Cust_Primary_Id" Source="%.%"
                       AutoRefresh="True" />
        </Declarations>
        <RelatedDestinations>
          <Destination SourceData="Result.Contactos">
            <Declarations>
              <Declaration Variable="CONTACTO_ID"
                           Source="#GetPkey(CONTACTOS_CLIENTE)#"
                           AutoRefresh="True" />
            </Declarations>
            <RelatedDestinations>
              <Destination Table="CONTACTOS_CLIENTE"
                           SourceData="Result.Telefonos">
                <Assignments>
                  <Assignment Destination="CUST_PRIMARY_ID"
                              Source="'%Variable.CUST_PRIMARY_ID%'" />
                  <Assignment Destination="CONTACTO_ID"
                              Source="'%Variable.CONTACTO_ID%'" />
                  <Assignment Destination="TIPO" Source="'TELEFONO'" />
                  <Assignment Destination="VALOR" Source="'%.%'" />
                  <Assignment Destination="PKEY"
                              Source="'#GetPkey(CONTACTOS_CLIENTE)#'" />
                  <Assignment Destination="PAR_KEY"
                              Source="'%Message.JobPkey%'" />
                  <Assignment Destination="TS_BEGIN" Source="Getdate()" />
                  <Assignment Destination="TS_END" Source="Getdate()" />
                  <Assignment Destination="TS_USER_ID"
                              Source="'%Message.UserId%'" />
                </Assignments>
              </Destination>
              <Destination Table="CONTACTOS_CLIENTE" SourceData="Result.Emails">
```

```
            <Assignments>
              <Assignment Destination="CUST_PRIMARY_ID"
                          Source="'%Variable.CUST_PRIMARY_ID%'" />
              <Assignment Destination="CONTACTO_ID"
                          Source="'%Variable.CONTACTO_ID%'" />
              <Assignment Destination="TIPO" Source="'EMAIL'" />
              <Assignment Destination="VALOR" Source="'%.%'" />
              <Assignment Destination="PKEY"
                          Source="'#GetPkey(CONTACTOS_CLIENTE)#'" />
              <Assignment Destination="PAR_KEY"
                          Source="'%Message.JobPkey%'" />
              <Assignment Destination="TS_BEGIN" Source="Getdate()" />
              <Assignment Destination="TS_END" Source="Getdate()" />
              <Assignment Destination="TS_USER_ID"
                          Source="'%Message.UserId%'" />
            </Assignments>
          </Destination>
        </RelatedDestinations>
      </Destination>
    </RelatedDestinations>
  </Destination>
 </InsertCommands>
</OutPut>
</Transformation>

<Transformation ID="EXTRACCION_DE_CLIENTES">
 <InPut />
 <OutPut>
  <DeleteCommands>
   <Destination Table="DATOS_EXTRACCION">
    <DeleteFilters>
     <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
    </DeleteFilters>
   </Destination>
   <Destination Table="DATOS_CLIENTE">
    <DeleteFilters>
     <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
    </DeleteFilters>
   </Destination>
  </DeleteCommands>
  <InsertCommands>
   <Destination Table="DATOS_EXTRACCION" RecordSetFilter="2">
    <Assignments>
     <Assignment Destination="CANTIDAD"
                 Source="%Result.Cantidad%" Default="0" />
```

```
        <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
      </Assignments>
    </Destination>
    <Destination Table="DATOS_CLIENTE" RecordSetFilter="1, 3">
      <Assignments>
        <Assignment Destination="CUSTOMER_NAME" Source="'%Result.NOM%'" />
        <Assignment Destination="PKEY" Source="'#GetPkey(DATOS_CLIENTE)#'" />
        <Assignment Destination="PAR_KEY" Source="'%Message.CustPkey%'" />
        <Assignment Destination="TS_BEGIN" Source="Getdate()" />
        <Assignment Destination="TS_END" Source="Getdate()" />
        <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
      </Assignments>
    </Destination>
  </InsertCommands>
 </OutPut>
</Transformation>

<Transformation ID="CONSULTA_DE_CLIENTES_XML">
 <InPut />
 <OutPut>
  <XMLSerializations>
   <XMLSerialization Source=".">
    <LibraryName>ConsultarClientesXML</LibraryName>
    <ClassType>ConsultarClientesResponse</ClassType>
   </XMLSerialization>
  </XMLSerializations>
  <DeleteCommands>
   <Destination Table="DATOS_CLIENTE">
    <DeleteFilters>
     <DeleteFilter Destination="TS_USER_ID" Source="'%Message.UserId%'" />
    </DeleteFilters>
   </Destination>
  </DeleteCommands>
  <InsertCommands>
   <Destination Table="DATOS_CLIENTE"
                SourceData="Result.ConsultarClientesResult">
    <Assignments>
     <Assignment Destination="CUST_PRIMARY_ID"
                 Source="'%Result.Cust_Primary_Id%'" />
     <Assignment Destination="SITUACION"
                 Source="%Result.Situacion%" Default="0" />
     <Assignment Destination="DEUDA"
                 Source="%Result.Deuda%" Default="0.0" />
     <Assignment Destination="CUSTOMER_NAME"
                 Source="'%Result.Customer_Name%'" />
```

```
        <Assignment Destination="TELEFONO"
                    Source="'%Result.Contactos(0).Telefonos(0)%'" />
        <Assignment Destination="EMAIL"
                    Source="'%Result.Contactos(0).Emails(0)%'" />
        <Assignment Destination="PKEY" Source="'#GetPkey(DATOS_CLIENTE)#'" />
        <Assignment Destination="PAR_KEY" Source="'%Message.CustPkey%'" />
        <Assignment Destination="TS_BEGIN" Source="Getdate()" />
        <Assignment Destination="TS_END" Source="Getdate()" />
        <Assignment Destination="TS_USER_ID" Source="'%Message.UserId%'" />
      </Assignments>
    </Destination>
   </InsertCommands>
  </OutPut>
</Transformation>
```