

Creación de un enrutador de WCF

Michele Leroux Bustamante

Hospedar y consumir un servicio de Windows® Communication Foundation (WCF) a menudo requiere algunos pasos fundamentales: implementar el servicio, configurar los extremos para llegar al servicio, hospedar el servicio, generar un archivo de Lenguaje de descripción de servicios web (WSDL) o habilitar el intercambio de metadatos para que los clientes puedan generar un proxy para llamar al servicio, escribir código para crear una instancia del proxy con su configuración asociada y empezar a llamar a las operaciones de servicio. Rara vez necesitará tratar el tema en profundidad, pero, incluso en el caso más sencillo, los canales de cliente y de servicio dependen de una configuración compatible para controlar la semántica de direccionamiento y el filtrado de mensajes para asegurar la invocación de la operación correcta.

A veces, resulta útil introducir un intermediario o un servicio de enrutador entre un cliente y un servicio de destino para recibir los mensajes que fluyen entre ambos y realizar actividades adicionales, como el registro, el enrutamiento de la prioridad, el enrutamiento con o sin conexión, el equilibrio de carga o la introducción de un límite de seguridad. Al introducir un servicio intermedio, es necesario ajustar algunos comportamientos de filtrado de direccionamiento y de mensajes.

Examinemos más detenidamente cómo puede trabajar con servicios intermedios, a los que llamaré colectivamente enrutadores. En este número, explicaré los conceptos de filtrado de direccionamiento y de mensajes de WCF mientras me centro en el escenario del enrutador, además de algunas de las opciones para la configuración de enrutamiento junto con la configuración pertinente. En la segunda parte de esta serie, le mostraré cómo aprovechar estos cimientos para conseguir implementaciones prácticas de enrutamiento más avanzadas.

Semántica de direccionamiento predeterminada

En el artículo [Estación de servicio de junio de 2007 \(msdn.microsoft.com/msdnmag/issues/07/06/ServiceStation\)](http://msdn.microsoft.com/msdnmag/issues/07/06/ServiceStation), Aaron Skonnard explicó cómo controla WCF los filtrados lógicos y físicos de direccionamiento de extremos, de encabezados de direccionamiento y de mensajes. En esta sección, repasaré algunas de las características de direccionamiento más importantes y cómo influyen en los escenarios de enrutamiento (el artículo de Aaron también le resultará muy útil si desea obtener información adicional acerca de estas características de WCF).

Normalmente, los clientes envían mensajes directamente al servicio de destino mediante un proxy generado a partir de la descripción del servicio. Para hacer que el cliente y el servicio sean compatibles, comparten configuraciones de contrato y de extremo equivalentes. Si tiene en cuenta el contrato y la configuración del servicio que aparecen en la [Figura 1](#), puede obtener diversos requisitos de direccionamiento importantes para el servicio.

En primer lugar, el encabezado de direccionamiento Action para enviar una solicitud a la operación SendMessage:

<http://www.thatindigogirl.com/samples/2008/01/IMessageManagerService/SendMessage>

Ya que OperationContractAttribute no especifica un valor Action, se deriva del espacio de nombres del contrato de servicio, del nombre del contrato (cuyo nombre predeterminado es el nombre de la interfaz) y del nombre de la operación (cuyo nombre predeterminado es el nombre del método).

En segundo lugar, el encabezado Action esperado para una respuesta de SendMessage es:

<http://www.thatindigogirl.com/samples/2008/01/IMessageManagerService/SendMessageResponse>

Ya que OperationContractAttribute no especifica un valor ReplyAction, éste se deriva de la misma forma que la propiedad Action, con el sufijo "Response" anexo.

Por último, el encabezado To para los mensajes dirigidos al extremo del servicio es:

<http://localhost:8000/MessageManagerService>

Este valor se obtiene a partir del atributo de dirección del elemento de extremo, que se considera la dirección lógica del extremo. Aunque se puede especificar lo contrario, la dirección física del extremo suele coincidir con la dirección lógica. Esto implica que los clientes a menudo envían mensajes a direcciones físicas que coinciden con el encabezado To.

Los metadatos del servicio describen estos requisitos de forma que los clientes puedan generar un proxy y una configuración compatibles. El contrato de servicio generado para el cliente refleja la misma configuración para las configuraciones de Action y ReplyAction del servicio, y la configuración de enlaces del cliente refleja un extremo con las direcciones lógica y física adecuadas. Por ejemplo, el siguiente extremo del cliente es compatible con el servicio de la [Figura 1](#):

```
<client>  
  <endpoint
```

```

address="http://localhost:8000/MessageManagerService"
binding="basicHttpBinding"
contract="localhost.IMessageManagerService"
name="basicHttp" />
</client>

```

El cliente proxy usa la propiedad de la dirección del elemento de extremo del cliente para las direcciones lógicas y física. Como resultado, se envían mensajes a una dirección física que coincida con el encabezado To, como ya indiqué. Cuando el proxy llama a la operación SendMessage, se envía un mensaje con los encabezados To y Action que aparecen en la [Figura 2](#).

La combinación de los encabezados To y Action, respectivamente, indican qué distribuidor de canal debería procesar el mensaje y qué operación debería invocar el distribuidor al modelo del servicio. De forma predeterminada, debe haber un extremo con una dirección lógica que coincida con el encabezado To y una operación que coincida con el encabezado Action. La [Figura 3](#) ilustra este flujo.



Figura 3 Direccionamiento típico sin enrutador (Hacer clic en la imagen para ampliarla)

En las próximas secciones, explicaré las implicaciones de las direcciones lógicas y física, los encabezados To y Action y las reglas de filtrado de mensajes cuando entra en juego un enrutador.

Arquitectura de enrutamiento

Aunque existen algunas variaciones a la hora de diseñar un enrutador, la mayoría de los enrutadores deben ser capaces de dirigirse a cualquier servicio y de reenviar el mensaje original al servicio de destino adecuado. Existen dos enfoques fundamentales para el diseño de un enrutador: un enrutador de paso a través o un enrutador de procesamiento.

Un enrutador de paso a través es transparente para el cliente. La relación del cliente se lleva a cabo a través de los servicios de flujo descendente, pero los mensajes fluyen a través del enrutador. El cliente debe enviar el mensaje al enrutador mediante un protocolo de transporte y un codificador de mensajes compatibles y dicho mensaje debe satisfacer todos los protocolos de seguridad, de sesiones confiables, de sesiones de aplicación o de otro tipo de mensaje que requiera el canal de servicios. El enrutador puede mirar los encabezados del

mensaje o, incluso, insertarlos, pero los elementos del mensaje original se reenvían al servicio sin modificar. En la Figura 4 se ilustra este paso.

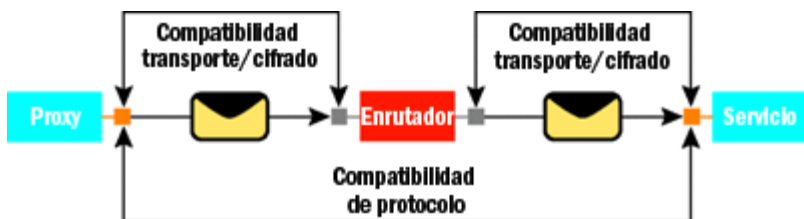


Figura 4 Operación de un enrutador de paso a través (Hacer clic en la imagen para ampliarla)

Un enrutador de procesamiento desempeña un papel más activo a la hora de procesar mensajes para la aplicación. De este modo, la relación del cliente se lleva a cabo con el enrutador en lo que se refiere al transporte, a la codificación y a la compatibilidad de protocolos, aunque el cliente debe seguir siendo capaz de enviar mensajes que sean compatibles con los servicios de flujo descendente. Los mensajes fluyen a través del enrutador a los servicios y el cuerpo del mensaje (y cualquier encabezado necesario para el servicio) permanece intacto.

A menudo, el enrutador controla la seguridad, las sesiones confiables y los encabezados o mensajes relacionados con otros protocolos de comunicación. Éste crea un mensaje nuevo conforme a los protocolos de comunicación adecuados para los servicios de flujo descendente. En la Figura 5 se ilustra la compatibilidad para un enrutador de procesamiento.

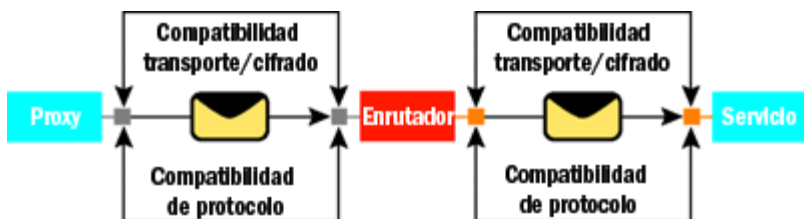


Figura 5 Operación de un enrutador de procesamiento (Hacer clic en la imagen para ampliarla)

Cada configuración de enrutamiento tiene implementaciones prácticas. También se puede implementar una solución híbrida en algún punto entre los dos.

Contrato de enrutador

Los enrutadores reciben mensajes diseñados para los servicios de flujo descendente y son los responsables de reenviar dichos mensajes al servicio adecuado. También se encargan de recibir respuestas del servicio y devolvérselas al cliente. Un contrato de enrutador típico expone una única operación que puede

procesar cualquier solicitud o respuesta de mensaje. En el siguiente ejemplo, esa operación recibe el nombre `ProcessMessage`:

```
[ServiceContract(Namespace =  
"http://www.thatindigogirl.com/samples/2008/01")]  
public interface IRouterService {  
    [OperationContract(Action = "*", ReplyAction = "*")]  
    Message ProcessMessage(Message requestMessage);  
}
```

En una situación normal, las propiedades `Action` y `ReplyAction` de `OperationContractAttribute` se obtienen de la forma que ya expliqué anteriormente en este artículo (del espacio de nombres del contrato de servicio). De forma predeterminada, cuando llega un mensaje, el distribuidor de canal debe encontrar una operación que coincida totalmente con el encabezado `Action`. No obstante, si `Action` y `ReplyAction` están establecidos en "*", el distribuidor de canal enviará todos los mensajes que no estén asignados a una operación de catch-all con independencia del valor del encabezado `Action`. Para evitar la ambigüedad, sólo una operación puede especificar "*" para la propiedad `Action` o `ReplyAction`.

Un enrutador típico ofrece una única operación como `ProcessMessage` que puede procesar todos los mensajes que le lleguen. Aunque `Action` y `ReplyAction` se encargarán de que el distribuidor de canal asigne el mensaje a `ProcessMessage`, la firma de la operación también debe ser capaz de procesar cualquier mensaje.

Con el fin de solucionar esto, `ProcessMessage` recibe y devuelve mensajes sin tipo con la forma del tipo `Mensaje`. El enrutador puede tener acceso al encabezado en la colección y al cuerpo del mensaje mediante este tipo, pero no tiene lugar ninguna serialización automática más allá de los encabezados de direccionamiento comunes, que están deserializados y disponibles a través de las propiedades de establecimiento inflexible de tipos.

Cualquier otro procesamiento de los mensajes depende de la implementación del enrutador. Un enrutador básico simplemente recibirá el mensaje sin tipo y lo reenviará tal cual a los servicios de flujo descendente a la espera de una respuesta. De forma similar, se reenvía la respuesta al cliente que llama con el mismo formato sin procesar.

Reenvío de mensajes

Una vez que el enrutador recibe un mensaje y lo procesa según sus propios requisitos, lo reenvía al servicio de flujo descendente adecuado para su posterior procesamiento. En la [Figura 6](#), se muestra una implementación del enrutador

sencilla para el contrato tratado anteriormente. `ProcessMessage` construye un canal de cliente (o proxy) con `ChannelFactory<T>` y usa este proxy para reenviar los mensajes a un extremo de servicio particular, a la vez que devuelve todas las respuestas.

Los proxy suelen tener un establecimiento de tipos inflexible, pero, en este caso, el proxy debería ser capaz de reenviar todos los mensajes y de recibir cualquier respuesta (algo que facilita el contrato del enrutador). En este sencillo ejemplo, el enrutador reenvía el mensaje original al servicio de destino y devuelve todas las respuestas. Si la operación en el servicio de destino es unidireccional, no se envía ninguna respuesta.

Ya que el contrato actúa con los mensajes sin tipo, se reenvía el mismo mensaje al servicio, tal y como se muestra en la Figura 7. No obstante, debe ser consciente de que se ha implementado al menos un cambio en el mensaje que quizás no esperaba: se ha alterado el encabezado `To` antes de enviar el mensaje al servicio.

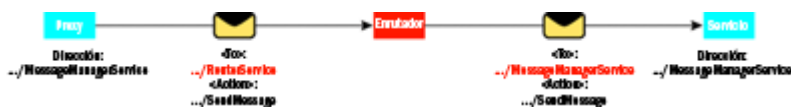


Figura 7 Semántica de direccionamiento mediante un enrutamiento sencillo (Hacer clic en la imagen para ampliarla)

Recuerde que el proxy usará de forma predeterminada la dirección lógica de su configuración de extremo para establecer el encabezado `To` para los mensajes salientes (incluso si se devuelve una instancia de `Message` que ya cuenta con un encabezado `To`). Aunque esto pueda parecer algo positivo (ya que todos los servicios necesitan que el encabezado `To` coincida con la dirección lógica de uno de los extremos de servicio), puede producir otros efectos secundarios. Por ejemplo, si el encabezado `To` no está firmado y el servicio tiene seguridad activada, se rechazará el mensaje.

En condiciones ideales, el cliente debería enviar un mensaje con un encabezado `To` que coincida con el servicio de destino, el enrutador debería aceptar el mensaje, a pesar de que no coincida, y el enrutador debería reenviar dicho mensaje al servicio sin alterar el encabezado `To`. Este aspecto se puede tratar mediante configuraciones de enlaces, que discutiré a continuación.

Direccionamiento lógico y físico

Al introducir un enrutador en la arquitectura de la aplicación, es mejor si el cliente puede enviar mensajes con el encabezado `To` adecuado para el servicio mientras que envía el mensaje al enrutador. Una forma de conseguirlo es a través de la configuración del cliente para que use `ClientViaBehavior`, como en la configuración

que aparece en la [Figura 8](#). Ello indica al cliente proxy que genere un mensaje con un encabezado To de acuerdo con la dirección lógica del extremo, pero para enviar a través de la dirección física del enrutador. El problema es que así el usuario se vincula a la existencia de un enrutador.

Otra forma de solucionar esto es hacer que el servicio configure el atributo listenUri para sus extremos, de forma que la dirección lógica del servicio pasa a ser la del enrutador, mientras que la dirección física es específica del servicio. Tenga en cuenta esta configuración de servicio:

```
<endpoint address="http://localhost:8010/RouterService"
  contract="MessageManager.IMessageManagerService"
  binding="wsHttpBinding"
  bindingConfiguration="wsHttpNoSecurity"
  listenUri="http://localhost:8000/MessageManagerService"/>
```

Los metadatos del servicio resultantes publican la dirección del enrutador a los clientes, de forma que los extremos de los clientes reflejan la dirección del enrutador. Personalmente, no me gusta esta solución, ya que vincula el servicio al enrutador y, en condiciones ideales, el servicio no lo necesita.

La alternativa es hacer que los servicios usen una dirección lógica que consista en un tipo de URI no vinculado ni al enrutador ni al servicio y, a continuación, comunicarle a los clientes de forma manual la dirección física a la que enviar los mensajes, ya que no forman parte de los metadatos. Aquí tiene un ejemplo de este tipo de configuración de extremo:

```
<endpoint address="urn:MessageManagerService"
  contract="MessageManager.IMessageManagerService"
  binding="wsHttpBinding"
  bindingConfiguration="wsHttpNoSecurity"
  listenUri="http://localhost:8000/MessageManagerService"/>
```

En cualquier caso, el servicio recibirá un encabezado To que coincida con la configuración de extremo y el enrutador recibirá el mensaje en primer lugar.

El enrutador debería soportar la carga de la configuración, de forma que los clientes y los servicios no dependan de su presencia. Así, el encabezado To nunca coincidirá con la dirección lógica del enrutador. De forma predeterminada, los servicios usan EndpointAddressMessageFilter para determinar si el encabezado To de un mensaje coincide con alguno de sus extremos configurados. Ya que un enrutador no puede esperar que esto funcione, debería instalar MatchAllMessageFilter.

ServiceBehaviorAttribute admite esto mediante la propiedad AddressFilterMode, que se puede establecer en una de las enumeraciones AddressFilterMode: exacto

(predeterminado), prefijo o cualquiera. Ya que el prefijo del enrutador no puede garantizar su coincidencia con todos los servicios para los que recibe mensajes, tiene sentido que permita que pasen todos los encabezados To, de la siguiente forma:

```
[ServiceBehavior(InstanceContextMode =
    InstanceContextMode.Single,
    ConcurrencyMode = ConcurrencyMode.Multiple,
    AddressFilterMode=AddressFilterMode.Any)]
public class RouterService : IRouterService
```

De forma predeterminada, el encabezado To se actualizará continuamente para que coincida con una dirección lógica de proxy, en función de su configuración de extremo (con independencia de si la dirección To ya está establecida con el valor correcto. Para suprimir este comportamiento y que el enrutador pueda reenviar los mensajes al servicio mediante el encabezado To original, el enrutador debe usar una configuración de enlaces con direccionamiento manual. Esto no es una propiedad que se pueda establecer en todos los enlaces estándares, por lo que debe usar enlaces personalizados para conseguirlo.

El siguiente fragmento de código ilustra una sección de customBinding que establece esta característica para el canal de transporte HTTP:

```
<customBinding>
  <binding name="manualAddressing">
    <textMessageEncoding />
    <httpTransport manualAddressing="true"/>
  </binding>
</customBinding>
```

de esta forma, se facilita el flujo de direccionamiento que aparece en la Figura 9, donde no se modifican los encabezados.



Figura 9 Semántica de direccionamiento mediante un enrutador con direccionamiento manual (Hacer clic en la imagen para ampliarla)

Encabezados MustUnderstand

Hasta el momento, me he centrado en una implementación de enrutamiento sencilla para ilustrar las principales consideraciones para diseñar el enrutador que también tienen repercusiones en la configuración de direccionamiento, de filtrado y de enlaces. Esta sencilla solución de enrutamiento sólo funciona si el servicio no habilita la seguridad, las sesiones confiables o cualquier otro protocolo rico para

sus enlaces. En la Figura 10, se muestra una vista simplificada de los protocolos de enlace que he adoptado para la discusión hasta el momento.



Figura 10 Contrato de servicio y configuración de extremo (Hacer clic en la imagen para ampliarla)

En la Figura 11 se muestra la misma vista para una enrutador de paso a través cuando el servicio necesita sesiones confiables y seguridad. Habilitar estos protocolos implica que los servicios de cliente y de canales intercambiarán mensajes adicionales para establecer sesiones, solicitar tokens de seguridad y demás mensajería relacionada. Ya que el enrutador permite que pasen los mensajes, estos mensajes específicos de protocolo también pasarán al servicio (lo que es muy positivo).



Figura 11 Configuración de paso a través con sesiones confiables y seguridad (Hacer clic en la imagen para ampliarla)

No obstante, surge un problema cuando los mensajes dirigidos al servicio y procedentes de éste incluyen encabezados que el destinatario debe comprender. Ya que el enrutador de paso a través no tiene sesiones de confiabilidad o seguridad habilitadas, dichos canales no están presentes para procesar los encabezados de protocolo asociados.

Puede indicar al servicio de enrutador que ignore los encabezados MustUnderstand estableciendo la propiedad `ValidateMustUnderstand` de `ServiceBehaviorAttribute` en falso, como se muestra aquí:

```
[ServiceBehavior(InstanceContextMode =  
    InstanceContextMode.Single,  
    ConcurrencyMode = ConcurrencyMode.Multiple,  
    AddressFilterMode=AddressFilterMode.Any,  
    ValidateMustUnderstand=false)]  
public class RouterService : IRouterService
```

De esta forma, se dirigen los mensajes entrantes del cliente, pero no los mensajes devueltos de los servicios de flujo descendente.

Para solucionar esto, también debe modificar la implementación del enrutador para especificar este comportamiento al inicializar la fábrica de canales para llamar al servicio de flujo descendente, de la siguiente forma:

```
using (ChannelFactory<IRouterService> factory =
    new ChannelFactory<IRouterService>("serviceEndpoint"))
{
    factory.Endpoint.Behaviors.Add(new MustUnderstandBehavior(false));
    IRouterService proxy = factory.CreateChannel();

    // remaining code
}
```

Ahora, los mensajes de servicios y protocolos pueden fluir libremente entre el cliente y el servicio a través del enrutador (suponiendo que el protocolo usado sea HTTP).

Otro problema surge cuando se emplean protocolos dúplex como TCP o canalizaciones con nombre. Esto significa que el sistema puede iniciar los mensajes para el cliente, como al habilitar sesiones confiables. Existe una configuración de enrutador avanzada que puede usar para ocuparse de este caso especial, pero abordaré ese escenario y su sentido práctico en la segunda parte de esta serie.

Envíe sus preguntas y comentarios a sstation@microsoft.com.

Michele Leroux Bustamante es responsable de arquitectura de IDesign Inc., directora regional de Microsoft en San Diego y MVP de Microsoft para sistemas conectados. Su último libro es *Learning WCF*. Puede ponerse en contacto con ella en la dirección mlb@idesign.net o visitar design.net. Blogs de Michele en dasblonde.net.

© 2007 Microsoft Corporation and CMP Media, LLC. Reservados todos los derechos; queda prohibida la reproducción parcial o total sin previa autorización.