GUIA desarrolladores

Contents

[INTRODUCCIÓN 1](#_Toc436042602)

[FLUJOS 1](#_Toc436042603)

[Arranque de la aplicación. 1](#_Toc436042604)

[Recepción de un mensaje. 1](#_Toc436042605)

[COMPONENTES 2](#_Toc436042606)

[Bootstraper 2](#_Toc436042607)

[Listener 2](#_Toc436042608)

[Mediator 2](#_Toc436042609)

[Forwarder 2](#_Toc436042610)

[Quota 2](#_Toc436042611)

[Logger 3](#_Toc436042612)

[Configuración 3](#_Toc436042613)

[TESTS 3](#_Toc436042614)

# INTRODUCCIÓN

Davalor.MomProxy es una aplicación que permite delegar el envío de mensajes al Message oriented middleware, de ahora en adelante MOM, y la optimización del envío de los mismos configurando por topic/feed la política de reenvío.

Es importante señalar que el servicio levanta un puerto de escucha, para ello hay que lanzarlo con un usuario administrador o dar permisos al usuario que corre la aplicación mediante el comando [NETSH](https://technet.microsoft.com/es-es/library/cc754516(v=ws.10).aspx).

# FLUJOS

La aplicación comprende únicamente dos flujos de operación: Arranque de la aplicación, Recepción de un nuevo mensaje.

## Arranque de la aplicación.

1. Se cargan en memoria todos los mensajes existentes en el sistema de ficheros y se pasan a la cola correspondiente.
2. Una vez todos los mensajes pendientes de envío están cargados en memoria se levanta la interfaz web que queda a la escucha de nuevos mensajes.

## Recepción de un mensaje.

1. El servicio recibe una petición <http://[Maquina]:[puerto]/API/Message/NewMessage>
2. El servicio persiste el mensaje en disco y retorna al cliente confirmación de la recepción del mensaje.
3. En otro hilo el mensaje recibido se incorpora a una cola en memoria. Todos los mensajes recibidos han de tener una propiedad topic que determina la cola en la que se incorporan, si se recibe un mensaje sin la propiedad topic este no se gestiona y se retorna un http 400 al cliente.
4. Cuando el mensaje se incorpora a la cola, dependiendo de la cuota configurada para el topic, este se envía al MOM o se deja en la cola hasta que se cumple con la cuota configurada, en cuyo momento se cogen todos los mensajes de la cola y se envían en bloque al MOM.
5. Cuando se confirma el envío de un mensaje al MOM este se quita de la cola en memoria y se elimina de disco.

# COMPONENTES

## Bootstraper

Se encarga de gestionar el flujo de arranque de la aplicación, y es la clase Davalor.MomProxy.ConsoleHost.HostService

## Listener

Se encarga de escuchar nuevos mensajes que se quiere enviar al MOM está distribuido en dos clases:

* Davalor.MomProxy.ConsoleHost.WebAPISelfHost. Levanta el servicio WEB.API
* Davalor.MomProxy.ConsoleHost.MessageController. Gestiona la entrada de nuevos mensajes.

## Mediator

El servicio se gestiona mediante mensajes que se publican en el mediador. Cualquier componente del servicio puede potencialmente escuchar o emitir cualquiera de los mensajes definidos en el mediador. Los mensajes definidos a día de la redacción de este documento son:

* **ReceivedMessageSequence**. Ocurre cada vez que se ha recibido un mensaje válido.
* **ReceivedInvalidMessageSequence**. Ocurre cada vez que se ha recibido un mensaje inválido. Las razones por las que un mensaje es o no invalido están definidas en el método **IsValid** de la clase Davalor.Base.Messaging.Contracts.BaseEvents.
* **SavedIncommingMessage**. Ocurre cada vez que un mensaje recibido se guarda en el sistema de ficheros.
* **SentIncommingMessage**. Ocurre cada vez que se ha recibido la confirmación de envío de un mensaje al Mom.
* **DeletedIncommingMessage**. Ocurre cada vez que se ha eliminado un mensaje del sistema de ficheros.

El mediador es la clase Davalor.MomProxy. ServiceEvents

## Forwarder

Se encarga de gestionar el envío al Mom de los mensajes que llegan está dividido en dos clases:

1. Davalor.MomProxy.Services.MessageForwardingService. Se encarga de:
   * Recibir los mensajes que llegan
   * Enrutar los mensajes al MessageForwarder adecuado.
   * Parar el envío de mensajes al Mom
   * Iniciar el envío de mensajes al Mom
2. Davalor.MomProxy.Services.MessageForwarder. Se encarga de almacenar los mensajes que le va enviando el MessageForwardingService hasta que la cuota asociada indica que los mensajes se pueden enviar al Mom en cuyo momento los intenta enviar en bloque al Mom.

## Quota

Una cuota define si los mensajes de un topic se pueden o no enviar al Mom en un momento determinado. Hay Cuatro tipos de cuota:

1. **TransparentQuota.** Los mensajes se reenvían según se reenvían. Ante la ausencia de configuración esta es la cuota por defecto.
2. **NumberOfElementsQuota.** Los mensajes se reenvían cuando se han acumulado al menos el número de elementos definido.
3. **ElapsedTimeQuota.** Los mensajes se reenvían cuando ha pasado el tiempo configurado, en minutos, desde que se recibió el primer mensaje o desde que se realizó el último envío.
4. **TimeRangeQuota.** Los mensajes se reenvían siempre y cuando se esté dentro del rango de tiempo establecido. Los rango se establecen en horas(0-23) y en minutos (0-59)

Todos los topics tienen una cuota asociada y esta se establece por la ausencia o existencia de configuración asociada al topic. Si un topic no tiene configurada ninguna cuota en particular se establece la cuota por defecto es decir TransparentQuota.

Las cuotas existentes se encuentran implementadas en las clases:

* Davalor.MomProxy.Domain.Quota.ElapsedTimeQuota
* Davalor.MomProxy.Domain.Quota.NumberOfElementsQuota
* Davalor.MomProxy.Domain.Quota.TimeRangeQuota
* Davalor.MomProxy.Domain.Quota.TransparentQuota

## Logger

La aplicación usa “Event Tracing for Windows”, de ahora en adelante ETW, para trazar la actividad de la misma La instrumentación se hace generando eventos de ETW por cada evento detectado en el mediador. La instrumentación la lleva acabo la clase Davalor.MomProxy.Domain.MomProxyEventTracing

## Configuración

Para ver en detalle como configurar el servicio consultar el documento manual de explotación MomProxy.

# TESTS

### Tests Unitarios

Todos los proyectos que componen la solución, a excepción de Davalor.MomProxy.ConsoleHost , tienen un proyecto paralelo que se llama igual que el proyecto testeado pero acabado en “.unitTests” La estructura de estos proyectos mimetiza la del proyecto testeado con la excepción de fakes y clases de ayuda y que las clases testeadas acaban en “spec”.

A día de redacción de este documento todos los tests pasan correctamente y la cobertura de código es:

* Davalor.MomProxy.Domain 🡺 92,61 %
* Davalor.MomProxy.Host 🡺 60,24 %
* Davalor.MomProxy.Repository 🡺 100 %
* Davalor.MomProxy.Services 🡺 100 %

### Tests de integración

Con la solución se incluye un proyecto con tests básicos de integración Davalor.MomProxy.IntegrationTests. Para su correcta ejecución es necesario:

* Tener permisos para levantar una escucha en el puerto de red configurado en el fichero “HostConfiguration.json” definido en el proyecto de tests de integración.
* Tener acceso a las uris (Kafka) definidas en el fichero de configuración “KafkaConfiguration.json”