Diseño de un modelo de integración de sistemas mediante arquitectura ESB.

Trabajo Fin de Máster
Máster Universitario En Investigación En Ingeniería De Software Y Sistemas Informáticos.

Fecha última Modificación	11/06/20188
Versión del documento	1.0
Alumno	José Cebrián Torrejón

Diseño de un modelo de integración de sistemas mediante arquitectura ESB.

Máster Universitario En Investigación En Ingeniería De Software Y Sistemas Informáticos. Ingeniería Del Software.
Código 31105128

Tipo B

AUTORIZACIÓN

Autorizo a la Universidad Nacional de Educación a Distancia a difundir y utilizar, con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

José Cebrián Torrejón

Firma del Autor

RESUMEN

La mayoría de las organizaciones disponen de diversos sistemas y aplicaciones que se integran y que deben estar sincronizados. Un Enterprise Service Bus (ESB) es una infraestructura de software que funciona como capa intermedia (middleware), proporcionando servicios de integración de las distintas aplicaciones a través de mensajería basada en estándares y servicios de sincronización. Aunque un ESB no implementa por sí mismo una arquitectura orientada a servicios (SOA), proporciona características para su implementación.
En definitiva, un ESB debe ser capaz de reemplazar todo el contacto directo entre aplicaciones, consiguiendo que todas ellas se comuniquen a través del bus.
Los ESB transmiten y reciben mensajes basados en estándares, pero deben ser capaces de transformar mensajes a formatos que sean reconocidos por las distintas aplicaciones en el caso de que sea necesario, lo que se realiza a través de adaptadores.
Además, el intercambio de mensajes debe ser independiente de la plataforma. Esto permite al ESB integrar aplicaciones que se ejecutan en diversos sistemas operativos o mainframes.

PALABRAS CLAVE

ESB, SOA, EMS, JMS, XML, Integración, Middleware, Modelado de procesos de integración, Arquitectura de integración, Framework

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

1 INTRODUCCIÓN

1.1 OBJETIVO DEL PROYECTO

El objetivo de este PFM es la realización del análisis y diseño de una arquitectura ESB.

Con este objetivo se desea exponer los beneficios que aporta la implantación de un ESB en una empresa.

Durante el desarrollo de este proyecto se pondrá solución a estos problemas de integración mediante la definición de un framework ESB que facilitará la integración entre todos los sistemas y a su vez ofrecerá una capa de monitorización y alarmado de los mismos.

Este Framework será completamente estándar, de modo que podría ser implantado en cualquier empresa sin ningún tipo de adaptación.

Para finalizar el trabajo se definirá el caso de uso ficticio para una empresa ficticia que contará con una serie de sistemas informáticos. Esta empresa tiene serios problemas para poder integrar sus sistemas debido a que tienen que convivir sistemas antiguos con nuevas tecnologías.

Como apoyo a la memoria con el análisis y definición del framework ESB, se desarrollará dicho framework para dar por probado el correcto funcionamiento del diseño.

Los desarrollos serán realizados mediante la suite de herramientas de TIBCO. Más concretamente TIBCO Business Works y TIBCO EMS.

Este proyecto contará por consiguiente con esta memoria donde se expondrá el análisis y diseño de la solución, junto con un repositorio con el código desarrollado.

1.2 CONTEXTO

1.2.1 Cuál es el papel de la integración en la empresa

En la actualidad, las medianas y grandes empresas tienen un gran número de sistemas informáticos para cubrir todas las necesidades del negocio de las mismas (RPM, CRM, Mainframe, Aplicaciones Web, Apps móviles etc…). Esto hace que la integración entre ellos sea liosa y compleja, llegando a duplicar integraciones entre ellas y realizar complejos desarrollos con el fin de comunicarlas. Debido a esta necesidad surge el paradigma del ESB (Enterprise Service Bus) que tendrá como principal función focalizar las integraciones de una empresa desde un único "bus de información". Esto favorece la reutilización de código mediante una serie de adaptadores para cada sistema y facilita la securización y monitorización de todos los flujos funcionales que pasen por él.

A continuación se muestra una representación gráfica simplificada de una integración entre sistemas con, y sin ESB.

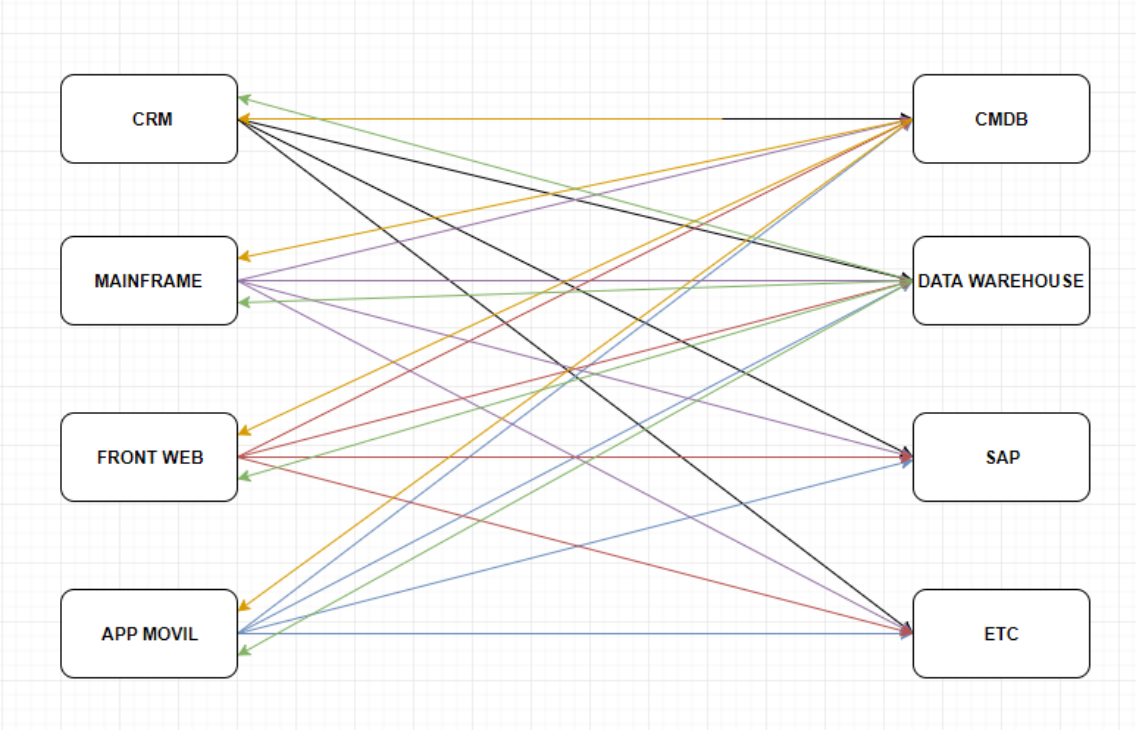


Imagen 1 - Integración sin ESB

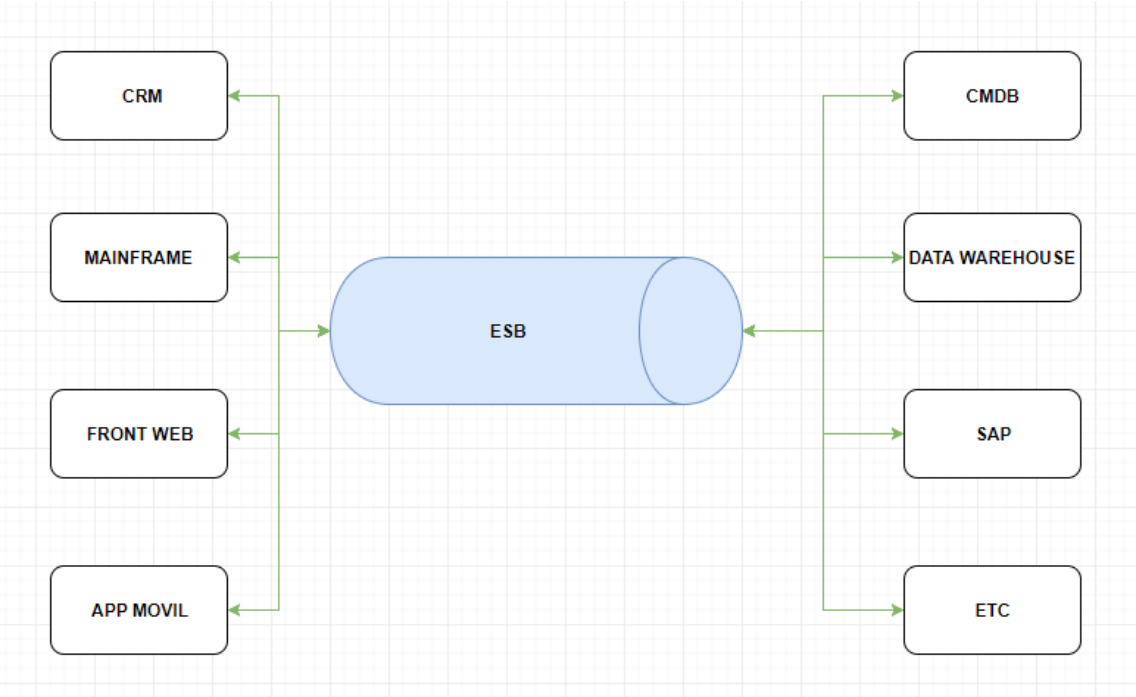


Imagen 2 - Integración con ESB

1.2.2 Que es un framework

Un framework es un espacio de trabajo que contiene todas las herramientas necesarias para el desarrollo de un producto concreto. Para ello, cuenta con patrones predefinidos y plantillas de desarrollo. También contiene todas las partes comunes de la herramienta. Las principales ventajas del uso de frameworks a la hora de desarrollar son:

- Agilidad:
  - Debido a que el framework consta de plantillas y partes comunes a todos los procesos desarrolladas, simplifica el desarrollo y lo agiliza, reduciendo así el número de horas invertidas en la construcción.

- Homogeneidad:
  - Debido a que el framework cuenta con una serie de patrones pre-desarrollados y con los elementos comunes ya desarrollados, asegura que los desarrollos se generen de forma homogénea independientemente del número de desarrolladores que intervengan en el mismo.
- Reusabilidad:
  - Al poder contener las partes de la aplicación comunes a todos los desarrollos (trazabilidad, auditoría, gestión de errores, adaptadores de comunicación), el uso de frameworks fomenta la reutilización de código consiguiendo su objetivo de ahorrar horas de desarrollo.
- Disminución del número de errores:

- Al contar con patrones y plantillas, los desarrollos son completamente guiados, por lo cual se reduce considerablemente el número de errores de programación.

- Disminuye la necesidad de expertos en tecnología.
  - El framework tiene como tarea facilitar el desarrollo de un sistema concreto. Por esta razón las partes que requieren un conocimiento avanzado de la herramienta están incluidas en el. Facilitando así el desarrollo sobre él y minimizando la necesidad de expertos de dicha aplicación en la plantilla.

1.2.3 Que es un ESB

ESB, Enterprise Service Bus, en sus siglas en ingles, es un componente diseñado para comunicar los diferentes sistemas de una empresa. El objetivo principal de este producto es canalizar las integraciones de una compañía en un único punto con el fin de mejorar su organización, la integración entre sistemas de cualquier tipo y reutilizar al máximo las comunicaciones entre sistemas. [01]

Para ello el ESB deberá contar con una serie de adaptadores para cada sistema, que serán los encargados de recibir o enviar la información. Estos adaptadores enviarán la información a elementos intermedios que realizarán transformaciones sobre los datos u orquestarán los diferentes flujos de negocio.

Al ser un componente intermedio entre todos los sistemas, se convierte en el escenario ideal para monitorizar los flujos de negocio sin tener que impactar a todos los sistemas. Ya que es posible monitorizar las trazas del ESB para obtener cualquier información del negocio que pase por el mismo.

En definitiva el ESB es un middleware que nace con el objetivo de facilitar la integración de una compañía dotándola a su vez de una monitorización y alarmado.

1.2.4 Que es SOA

SOA, Service Oriented Architecture, en sus siglas en ingles, es una arquitectura que tiene como paradigma la organización de las funcionalidades de una empresa en servicios.
Su idea original es atomizar todas estas funcionalidades dotando a la organización de menor acoplamiento entre ellas y facilitando la reutilización de las mismas.

Esta arquitectura está muy relacionada con el ESB, ya que es una forma muy práctica de establecer las comunicaciones entre los diferentes sistemas. De forma que el sistema maestro de la información puede exponer sus funcionalidades por medio de un servicio para que el resto de servicios de la compañía los pueda consumir.

1.2.5 En qué consiste un Framework para el desarrollo de un ESB

La idea original de un framework para los distintos desarrollos en un ESB es el disminuir de forma sustanciosa el volumen de desarrollo.
Hay que tener en cuenta que dentro de un ESB hay muchas tareas que se van a repetir en todos los flujos como pueden ser la gestión de errores o de trazas, a estas tareas se les denomina tareas “cross”.
También hay que valorar que, en un ESB, existen 3 tipos de integraciones:

- Síncrona
- Asíncrona
- Batch

Estos 3 tipos de integraciones se pueden cubrir con un número limitado de adaptadores (entendiendo como adaptador una pieza que facilita la comunicación con un sistema). De este modo es bastante factible añadir en este framework una serie de plantillas pre desarrolladas para cada adaptador.

Estas plantillas estarán formadas por flujos ya desarrollados que se encargarán de todas las tareas que no tienen dependencias con el negocio al que se va a orientar el desarrollo dentro del ESB.
De este modo, el desarrollador tan solo deberá preocuparse de generar las piezas con dependencias del negocio. Durante la ejecución, estas piezas serán llamadas desde los componentes ya incluidos en el framework.

Dicho esto, se muestra a continuación un mapa con todos los artefactos que formarán el framework del ESB que se va a detallar en este documento.

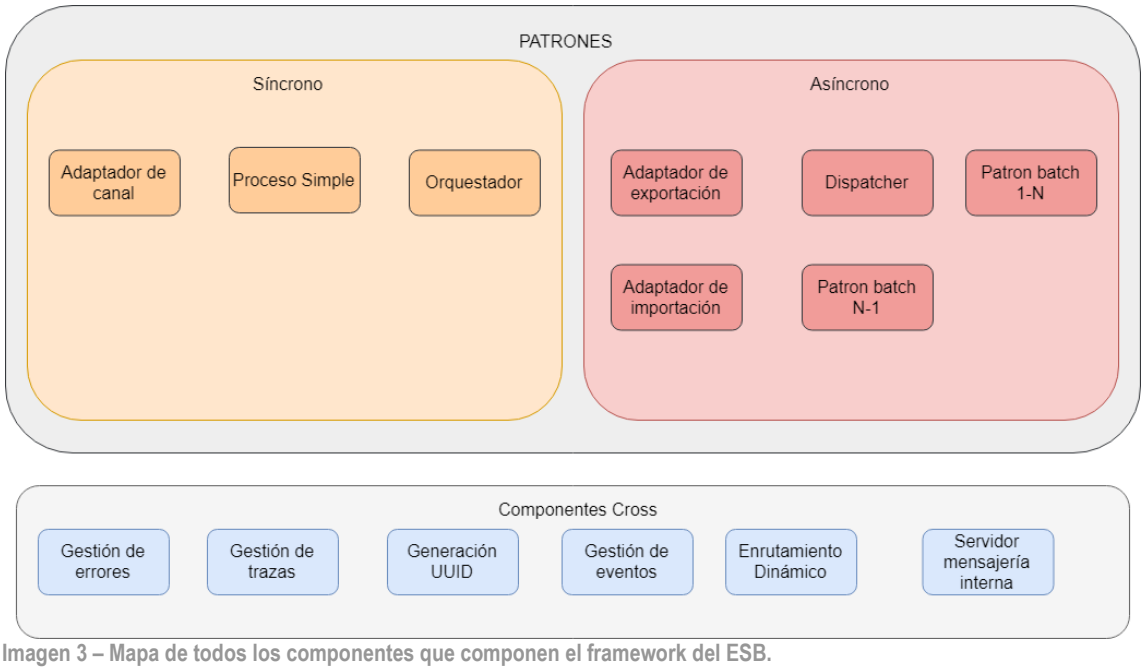


Imagen 3 – Mapa de todos los componentes que componen el framework del ESB.

1.2.6 Que herramientas ofrece el mercado

En este punto se analizarán las distintas herramientas que ofrece el mercado para el desarrollo de un ESB.
Como se comentó en el punto introductorio, para llevar a cabo este proyecto se ha decidido utilizar las herramientas que ofrece la suite de la empresa TIBCO. Más concretamente:

- Tibco ActiveMatrix Business Works 5.13.
- TIBCO EMS 8.3

La decisión se ha tomado con base a las siguientes razones:

- Es la herramienta con mayor experiencia en el mercado.
- Ofrece un desarrollo gráfico, el cual te permite llevar a cabo construcciones con tan solo nociones básicas de Java y XPath.
- Es la herramienta mejor valorada dentro del cuadrante Gartner.
- Se contaba previamente con mucha experiencia en el uso de la suite de TIBCO.

Imagen 4 – Cuadrante Mágico Gartner 2014

Pese a que la elección de la herramienta de desarrollo ha sido la correspondiente a la Suite de TIBCO, el diseño de este framework es totalmente independiente a la herramienta. Por lo que puede ser desarrollado en cualquier otra. Algunas de las más destacadas son:

MuleSoft:
MuleSoft, Inc. es una compañía de software con sede en San Francisco, California, que ofrece software de integración para conectar aplicaciones, datos y dispositivos. Originalmente, la compañía proporcionó middleware y mensajería, y más tarde se expandió para proporcionar un enfoque de plataforma de integración como servicio (iPaaS) para las empresas. MuleSoft desarrolla Mule ESB, una plataforma de integración para conectar aplicaciones empresariales locales y a la nube, diseñada para eliminar la necesidad de un código personalizado de integración punto a punto. [02]

Apache Camel:
Apache Camel es un motor de enrutamiento y mediación basado en reglas que provee una implementación basada en objetos Java de los patrones propuestos en Enterprise Integration Patterns,1 ya sea empleando una API o bien un lenguaje específico del dominio declarativo expresado en Java, para configurar las reglas de ruteo y mediación. El uso de un lenguaje específico del dominio significa que Apache Camel es capaz de soportar un completamiento automático de las reglas de ruteo en un entorno de desarrollo integrado usando código Java corriente sin gran cantidad de archivos de configuración XML, aunque también se soporta la configuración en XML en los marcos de Spring.
Camel muchas veces se emplea en conjunto con Apache ServiceMix, Open ESB, Apache ActiveMQ y Apache CXF en proyectos de infraestructura orientados a servicios. [03]

SoftwareAG:
Fundada en 1969, Software AG es una empresa de software empresarial con más de 10.000 clientes empresariales en más de 70 países. La compañía es el segundo mayor proveedor de software en Alemania, y el séptimo más grande en Europa. Software AG se comercializa en la Bolsa de Frankfurt bajo el símbolo "SOW" y parte del índice tecnológico TecDAX. [04]

IBM:
IBM Integration Bus (anteriormente conocido como WebSphere Message Broker) es un intermediario de integración de IBM de la familia de productos WebSphere que permite que la información empresarial fluya entre aplicaciones dispares en múltiples plataformas de hardware y software. Las reglas se pueden aplicar a los datos que fluyen a través del intermediario de mensajes para enrutar y transformar la información. El producto es un Enterprise Service Bus que suministra un canal de comunicación entre aplicaciones y servicios en una arquitectura orientada a servicios. [05]

2 ANALISIS DE LA SOLUCIÓN

2.1 INTRODUCCIÓN

Durante este punto del proyecto se analizarán todas las necesidades de integración que pueden surgir y como se pueden solventar con la solución que se propone.

La integración entre sistemas se puede resumir como la comunicación entre sistemas, y esta comunicación no es más que el intercambio de información entre dichos sistemas. Esta comunicación se puede llevar a cabo de varias formas. En este proyecto estas formas de comunicación entre sistemas se van a dividir en 3 escenarios principales:

- Integraciones Asíncronas.
- Integraciones Síncronas.
- Integraciones Batch.

En los siguientes puntos se analizarán los tres escenarios.

2.2 INTEGRACIONES ASÍNCRONAS

Se habla de integración asíncrona a la comunicación entre sistemas que no requiere de una respuesta puramente “online”. Esto significa que la petición y la respuesta dentro la comunicación no estará dentro de la misma ejecución. Las peculiaridades de este tipo de integración son las siguientes:

- El sistema origen no espera la respuesta.
    - No tiene que mantener la ejecución de la respuesta, liberando así al sistema de tener hilos de ejecución esperando.
- Se suele utilizar para inserción de información o modificación.
- Puede tener otro hilo de ejecución destinado a recibir la respuesta.
    - De este modo se entiende que la integración asíncrona puede constar de dos flujos de información totalmente independientes. Uno de ida con la petición y otro de vuelta con la respuesta.

2.2.1 Diagrama de flujo

En el siguiente punto se podrá observar de forma gráfica el flujo de intercambio de información de una integración asíncrona.

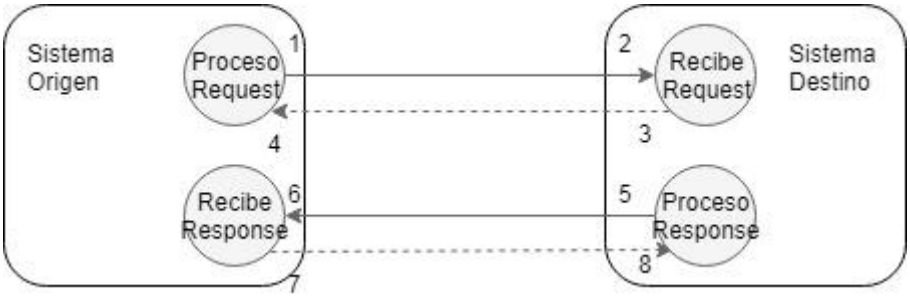


Imagen 5 – Diagrama de flujo Integración Asíncrona

1. El sistema origen envía la petición al sistema destino
2. El sistema destino recibe la petición
3. El sistema destino envía un ACK y empieza a procesar la petición
4. El sistema origen recibe el ack y termina la ejecución
5. El sistema destino una vez procesada la petición envía una respuesta.
6. El sistema origen recibe la respuesta en un nuevo proceso.
7. El sistema origen manda un ack de confirmación y empieza a procesar la respuesta.
8. El sistema destino recibe el ack y termina la ejecución del proceso.

2.2.2 Solución

Para este tipo de integraciones asíncronas se ha definido una solución basada en adaptadores de exportación e importación.

El adaptador de exportación será el encargado de cubrir el flujo entre el sistema origen y el ESB, mientras que el adaptador de importación será el encargado de la segunda parte de la integración entre el ESB y el sistema destino. De este modo se abstrae a los sistemas de la comunicación pudiendo finalizarla una vez enviado el mensaje al ESB.

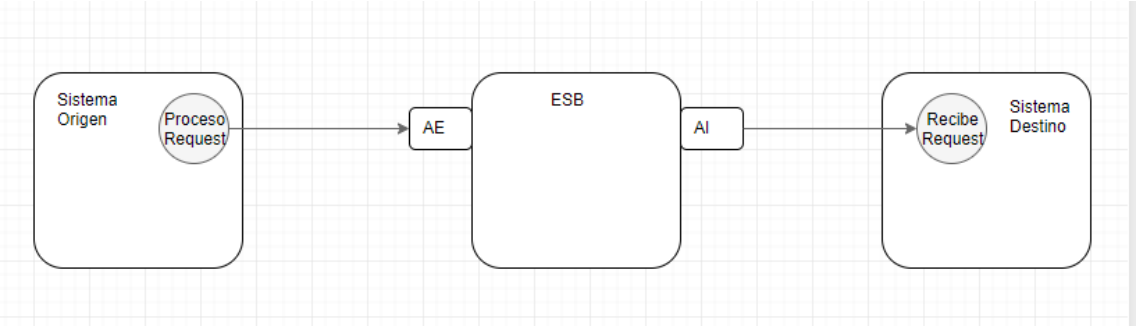


Imagen 6 - Diagrama de flujo de comunicación asincrona (Ida)

A su vez en el caso de necesitar una respuesta asíncrona se invertirá este flujo, siendo el sistema destino o origen de la información quien envíe un mensaje al adaptador de exportación y el sistema origen quien reciba dicho mensaje del adaptador de importación del ESB.

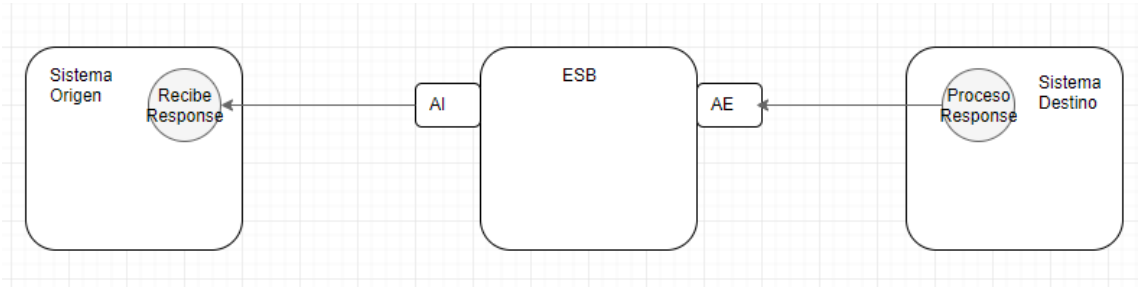


Imagen 7 - Diagrama de flujo de comunicación asíncrona (Vuelta)

2.3 INTEGRACIONES SÍNCRONAS

Se habla de integración síncrona a la comunicación entre sistemas que requiere de una respuesta puramente "online". Esto significa que la petición y la respuesta dentro la comunicación estará dentro de la misma ejecución. Las peculiaridades de este tipo de integración son las siguientes:

- El sistema origen necesita de la respuesta para continuar la ejecución.
- Se suele utilizar para consultas o para flujos transaccionales.
- El sistema origen depende del tiempo de respuesta del sistema destino, pudiendo verse afectado el rendimiento de este.

2.3.1 Diagrama de flujo

En el siguiente punto se podrá observar de forma gráfica el flujo de intercambio de información de una integración asíncrona.

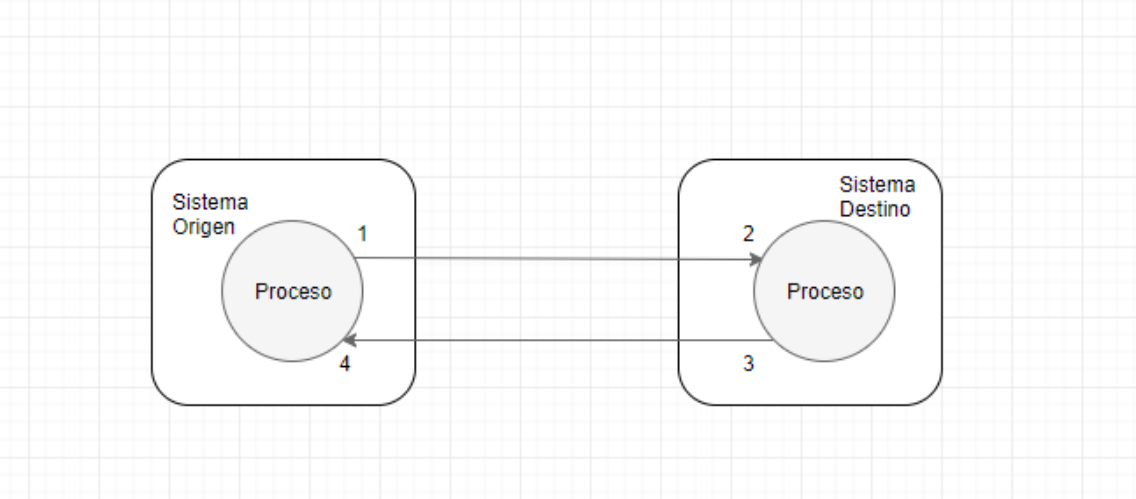


Imagen 8 - Diagrama de flujo Integración Síncrona

1. El sistema origen envía la petición al sistema destino
2. El sistema destino recibe la petición
3. El sistema destino una vez procesada la petición envía una respuesta.
4. El sistema origen recibe la respuesta en el mismo proceso

2.3.2 Solución

Para este tipo de integraciones síncronas se ha definido una solución basada en adaptadores de canal y procesos simple.

El adaptador de canal ofrece la función de interfaz de cara al sistema origen. Este suele estar representado técnicamente mediante un Web Service.

El Proceso Simple, es un tipo de adaptador que recoge una petición y la envía a un sistema destino. Este adaptador espera la respuesta del sistema destino en el mismo hilo de ejecución para devolvérselo al adaptador de canal.

La razón por la que existen al menos dos piezas en el patrón síncrono, es debido a que se pretende aislar la parte del sistema origen con la parte del sistema destino, de modo que el sistema origen tan solo tenga dependencias del adaptador de canal y el sistema destino del proceso simple. Con esta solución se gana en reutilización ya que el proceso simple puede ser invocado por otros adaptadores de canal de otros sistemas origen.

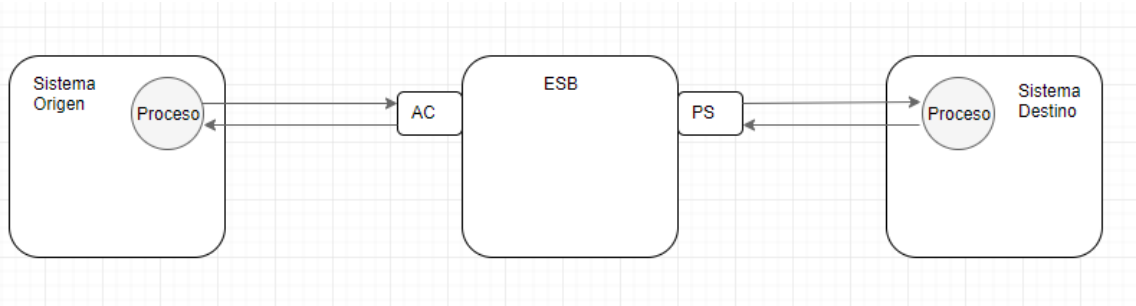


Imagen 9 - Diagrama de flujo de comunicación síncrona (Ida y vuelta)

A este patrón se le podría añadir un adaptador orquestador. Este adaptador tendría la función de orquestar, en el caso de que fueran necesarias, varias llamadas a diferentes sistemas destino. Por ejemplo en una inserción de una entidad que tenga que darse de alta en n sistemas.

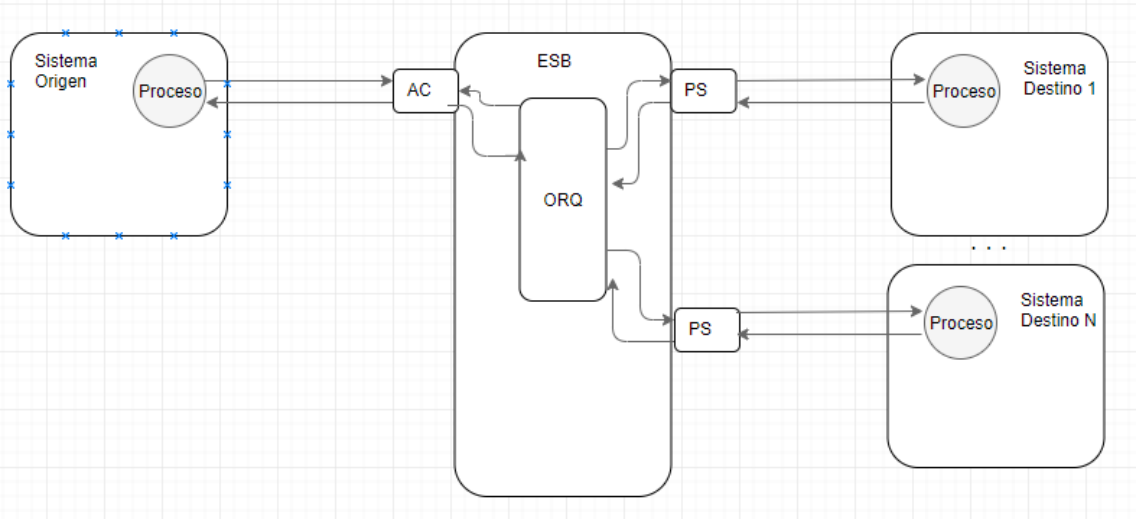


Imagen 10 - Diagrama de flujo de comunicación síncrona orquestada (Ida y vuelta)

2.4 INTEGRACIÓN BATCH 1-N

Se habla de integración batch a la comunicación entre sistemas que no requiere de un tratamiento online. En este tipo de integraciones prima la optimización de recursos a la simultaneidad del envío. Las peculiaridades de este tipo de integración son las siguientes:

- Las peticiones enviadas no son requeridas por el sistema destino en el momento en el que se procesan.
- Pueden almacenarse de forma temporal y ser enviadas un conjunto de ellas cada x tiempo.
- La forma en la que se transfiere la información en este tipo de integraciones es por fichero.

El patrón de integración N-1 que se va a revisar en este punto consiste en que el sistema origen envía un mensaje de forma asíncrona al ESB mediante el protocolo que se defina (http, jms, fichero…). El esb almacena esa petición y todas las siguientes hasta pasado n segundos. En el momento que pasan n segundos se forma un fichero con todos los mensajes recibidos y se envía al sistema destino. Una vez enviado el ESB vuelve a almacenar mensajes para repetir el procedimiento.

2.4.1 Diagrama de flujo

En el siguiente punto se podrá observar de forma gráfica el flujo de intercambio de información de una integración batch 1-N.

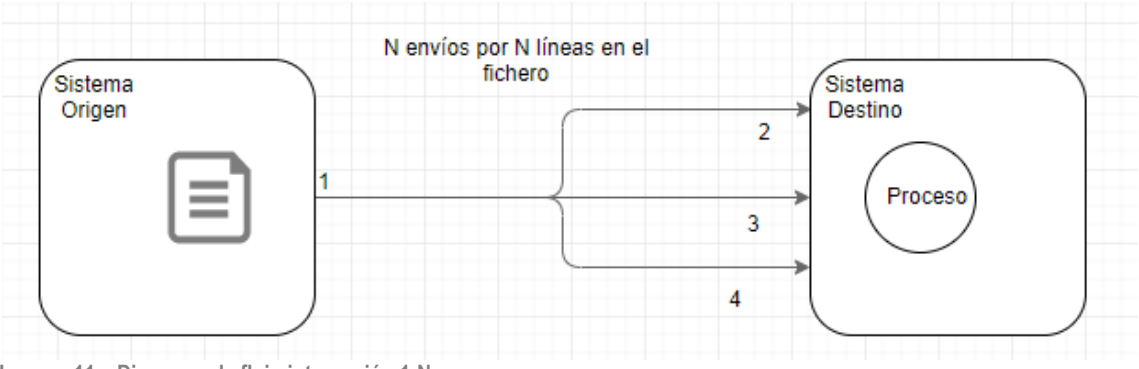


Imagen 11 – Diagrama de flujo integración 1-N

1. El sistema origen envía un fichero con n líneas.
2. Se envía una petición con la 1ª línea.
3. Se envía una petición con la 2ª línea.
4. Se envía una petición con la 3ª línea.

2.4.2 Solución

Para este tipo de integraciones batch se ha definido una solución basada en adaptador 1-N y un adaptador de importación.

El adaptador 1-N Recibe un fichero de un sistema origen vía FTP. Parsea este fichero en n peticiones por las n líneas que tenga el fichero y envía la información a un adaptador de importación. El adaptador de importación de forma asíncrona envía las peticiones al sistema destino. La comunicación entre los dos adaptadores se realiza bajo el servidor de mensajería interno.

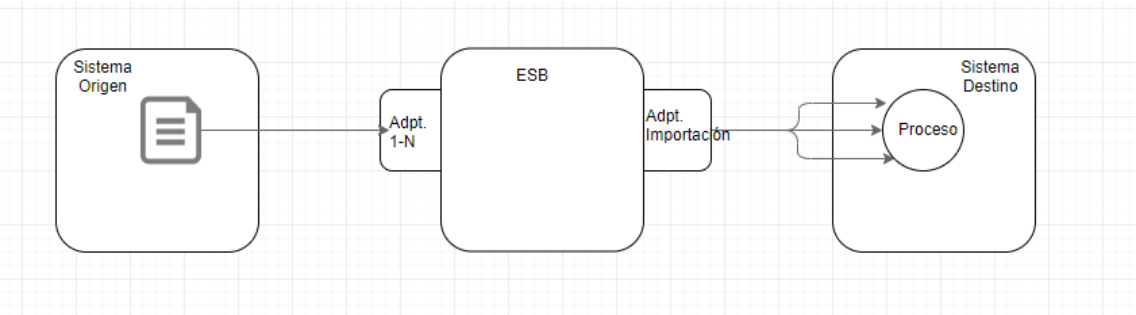


Imagen 12 - Diagrama de flujo de comunicación batch 1-N

2.5 INTEGRACIÓN BATCH N-1

Se habla de integración batch a la comunicación entre sistemas que no requiere de un tratamiento online. En este tipo de integraciones prima la optimización de recursos a la simultaneidad del envío. Las peculiaridades de este tipo de integración son las siguientes:

- Las peticiones enviadas no son requeridas por el sistema destino en el momento en el que se procesan.
- Pueden almacenarse de forma temporal y ser enviadas un conjunto de ellas cada x tiempo.
- La forma en la que se transfiere la información en este tipo de integraciones es por fichero.

El patrón de integración N-1 que se va a revisar en este punto consiste en que el sistema origen envía un mensaje de forma asíncrona al ESB mediante el protocolo que se defina (http, jms, fichero…). El esb almacena esa petición y todas las siguientes hasta pasado n segundos. En el momento que pasan n segundos se forma un fichero con todos los mensajes recibidos y se envía al sistema destino. Una vez enviado el ESB vuelve a almacenar mensajes para repetir el procedimiento.

2.5.1 Diagrama de flujo

En el siguiente punto se podrá observar de forma gráfica el flujo de intercambio de información de una integración batch N-1.

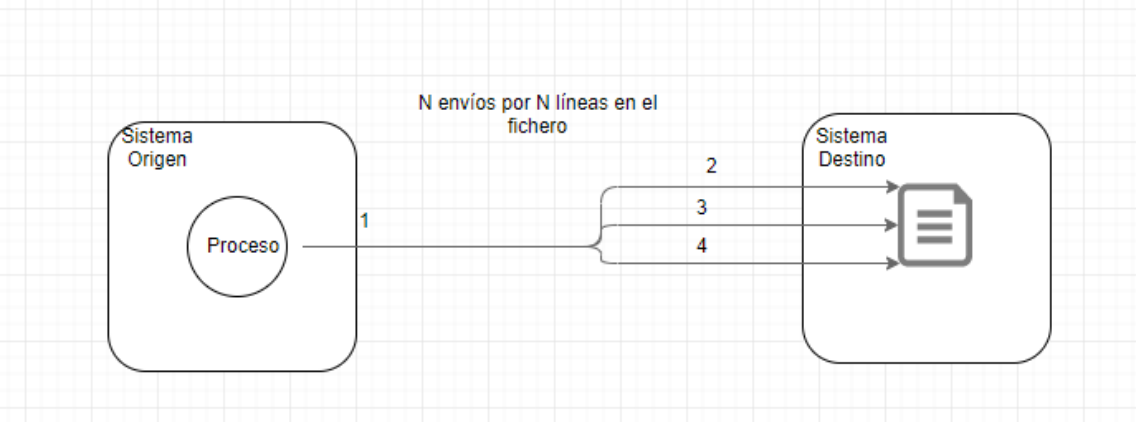


Imagen 13 - Diagrama de flujo integración N-1

1. El sistema origen envía n peticiones
2. La petición se integra en un único fichero para enviarse al sistema destino.
3. La petición se integra en un único fichero para enviarse al sistema destino.
4. La petición se integra en un único fichero para enviarse al sistema destino.

2.5.2 Solución

Para este tipo de integraciones batch se ha definido una solución basada en adaptador N-1 y un adaptador de exportación.

El sistema origen envía n peticiones a un adaptador de exportación (vía HTTP/JMS…).
El adaptador de exportación realiza una serie de tareas dependiendo del requerimiento y envía las peticiones al adaptador N-1 que, agrupa las peticiones recibidas y envía un fichero con ellas al sistema destino pasado un tiempo definido La comunicación entre los dos adaptadores se realiza bajo el servidor de mensajería interno.

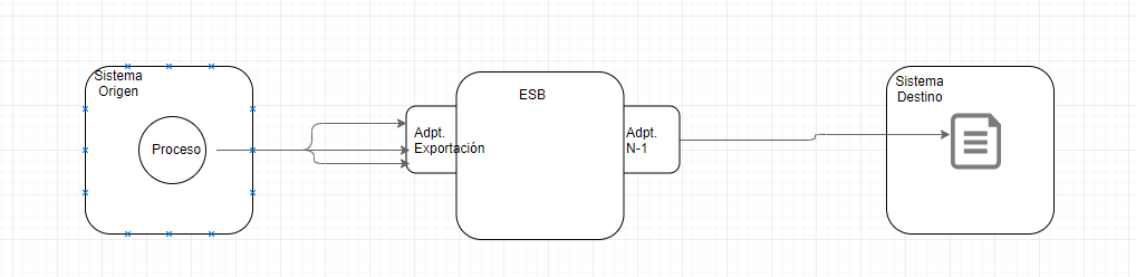


Imagen 14 - Diagrama de flujo de comunicación batch N-1

3 DISEÑO DE LA SOLUCIÓN

3.1 INTRODUCCIÓN

Durante este punto se analizarán en detalle los diferentes diseños de todos los adaptadores de la solución. En este diseño se estudiarán todas las piezas que forman cada adaptador. Estas piezas irán añadiendo funcionalidades distintas durante el trascurso del flujo de cada integración.

3.1.1 Mapa de patrones y adaptadores

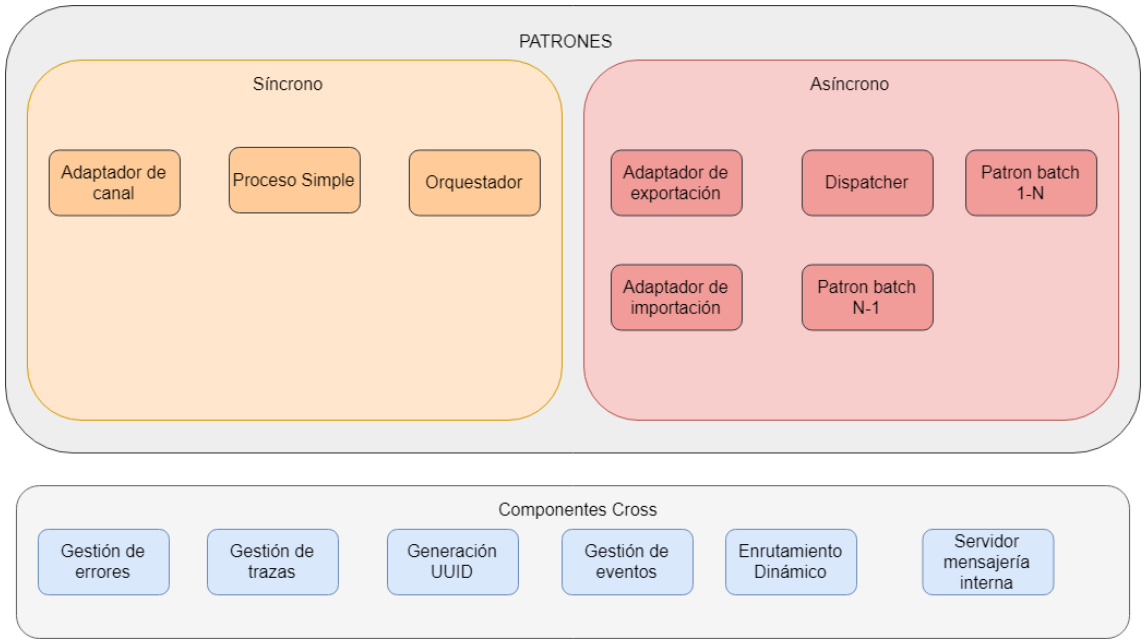


Imagen 15 - Mapa de patrones y adaptadores

3.1.2 Tipos de acciones

Gestión de Errores

Acción destinada al tratamiento de los errores. Los errores en el ESB van a ser divididos en dos grupos principales:

- Errores técnicos.
  - Estos errores son debidos a errores técnicos del sistema, como pueden ser:
    - Timeout
    - Errores de indisponibilidad
    - Errores de conexión
    - Errores de código
- Errores Funcionales.
  - Estos errores están relacionados con errores del ámbito de negocio, como pueden ser:
    - Errores relacionados a reglas de negocio incorrectas.
    - Errores de validación.
    - Errores de formato.

Gestión de trazas

Acción destinada al tratamiento de los logs dentro del ESB. En estos logs se trazarán una serie de eventos definidos en los patrones. Estos eventos serían los siguientes:

Evento	Desc	AE	AI	AC	ORQ	PS	N1	1N
PROCIN	Evento de entrada al adaptador	X	X	X	X	X	X	X
REQUEST	Evento de envío petición a Backend			X		X		
RESPONSE	Evento de respuesta de Backend			X		X		
PROCOUT	Evento de fin del adaptador	X	X	X	X	X	X	X
ENVIADESTINOS	Evento de envío asíncrono al Backend		X					
ERROR	Evento de error	X	X	X	X	X	X	X

Tabla 1 - Eventos de auditoría

Generación UUID

Se conoce por UUID (universally unique identifier) a un identificador único en un scope definido. Este formato de identificadores se encuentra dentro del estándar ISO/IEC 11578:1996 [06]. Este identificador será generado al inicio de la ejecución de cada adaptador. De este modo podrán ser identificados de forma unívoca.
Debido a que las integraciones entre sistemas dependerán de varios de estos adaptadores, se definirá que, el UUID del adaptador que origina el flujo, será a su vez el E2E id.
El E2E id es el identificador de inicio a fin de flujo. Este es utilizado para poder correlacionar todos los adaptadores de un mismo flujo. Pudiendo trazar de forma unívoca cada ejecución de integración de origen a destino.

La versión que se utilizará de UUID es la versión 1.
La versión 1 concatena e48 bits de la dirección MAC del "nodo" (es decir, el equipo que genera el UUID), con una marca de tiempo de 60 bits, siendo el número de 100-nanosegundos intervalos desde la medianoche 15 de octubre 1582 Tiempo Universal Coordinado (UTC), la fecha en que el calendario gregoriano fue adoptado por primera vez. RFC 4122 establece que el valor de tiempo se da la vuelta alrededor de 3400 AD, en función del algoritmo utilizado, lo que implica que la marca de tiempo de 60 bits es una cantidad firmado. Sin embargo, algunos programas, como la biblioteca libuuid, trata a la marca de tiempo como no firmada, poniendo el momento de vuelco en 5236 AD. [07]

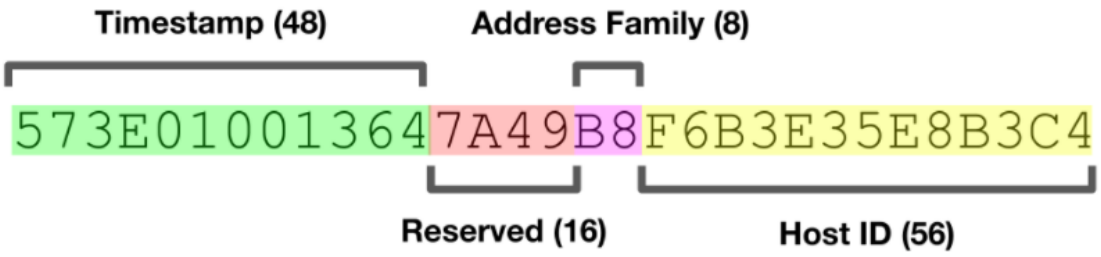


Imagen 16 - Formato UUID

Gestión de eventos

Se conoce como gestión de eventos en el ESB, al procedimiento de generación y envío de los diferentes eventos. Estos eventos son generados durante la ejecución del flujo y pueden ser remitidos a un 3er sistema para su tratamiento (Almacenamiento en BBDD, Monitorización de los procesos en un sistema de tratamiento en streaming etc…) Los eventos con los que cuenta el framework del ESB por defecto son los siguientes:

Evento	Desc	AE	AI	AC	ORQ	PS	N1	1N
PROCIN	Evento de entrada al adaptador	X	X	X	X	X	X	X
REQUEST	Evento de envío petición a Backend			X		X		
RESPONSE	Evento de respuesta de Backend			X		X		
PROCOUT	Evento de fin del adaptador	X	X	X	X	X	X	X
ENVIADESTINOS	Evento de envío asíncrono al Backend		X					
ERROR	Evento de error	X	X	X	X	X	X	X

Tabla 3 - Eventos de auditoría

Se puede comprobar cómo los eventos de auditoría son los mismos que los eventos que se registran en la gestión de trazas.

Enrutamiento Dinámico

El enrutamiento dinámico en el ESB consiste en la función de poder enviar a diferentes destinos dependiendo de un factor determinado en tiempo de ejecución. Esta función es muy útil, ya que proporciona un grado de reutilización bastante alto al código desarrollado en el ESB. Esto es debido a que con un solo proceso se puede despachar peticiones a diferentes sistemas dependiendo de una casuística determinada. De modo que pueden añadirse nuevos receptores con tan solo modificar el fichero de configuración donde se definen estas fuentes.

El funcionamiento a alto nivel del enrutamiento dinámico es el siguiente:

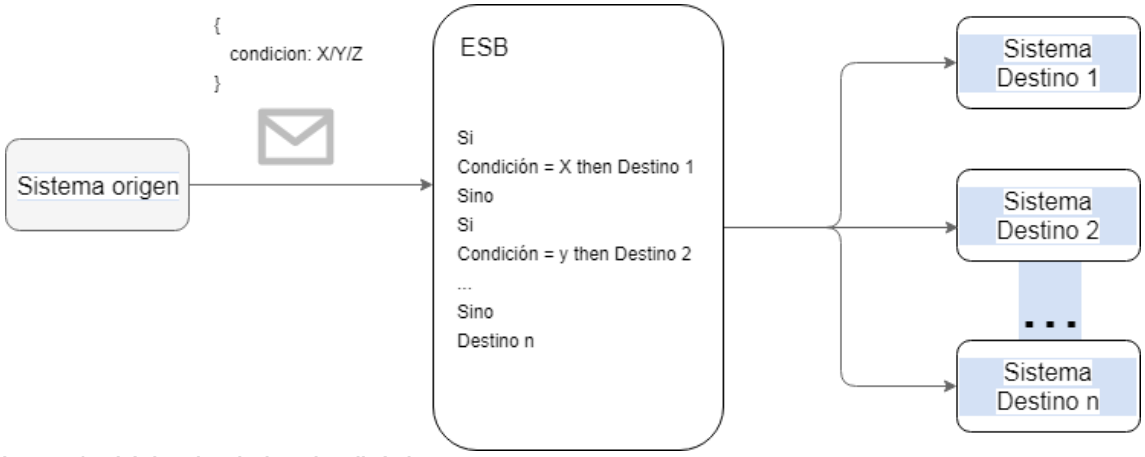


Imagen 17 - Lógica alto nivel routing dinámico

Mensajería interna

Una de los puntos fuertes del ESB es el balanceo de carga a un nivel atómico. Esto lo consigue mediante la mensajería interna. El Framework del ESB trabaja de una forma en que cada uno de los adaptadores que forman el flujo de integración comunica con el resto mediante comunicación JMS. Esto hace posible que en cada uno de estos pasos entre adaptadores la carga pueda repartirse de forma equitativa entre todos los nodos que forman el cluster del ESB. La forma de realizar esto es mediante la creación de colas dentro del servidor EMS. Estas colas son el canal de comunicación entre los adaptadores. De modo que todas las instancias del adaptador que va a recibir los mensajes están escuchando en esa cola. Esto hace que, estos mensajes se repartan entre todas las instancias/nodos que están escuchando en esa cola.

A continuación se detalla el proceso mediante un gráfico explicativo:

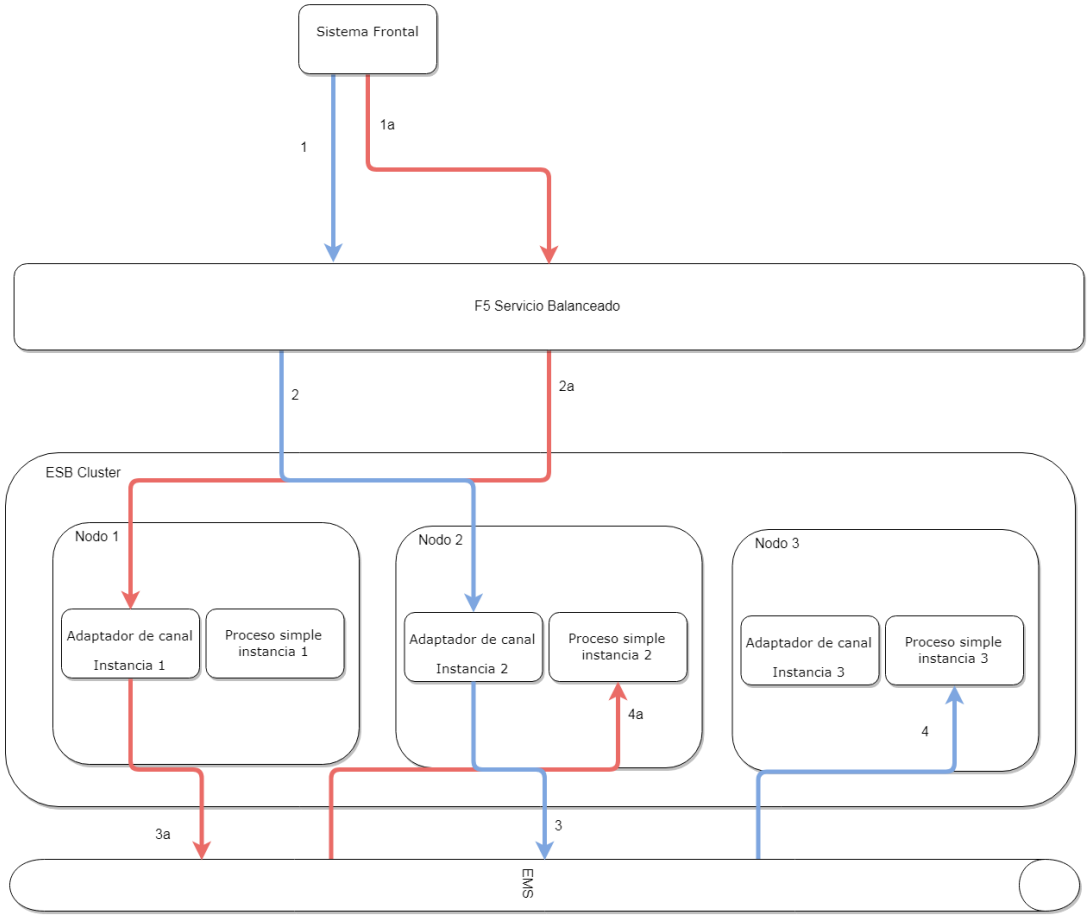


Imagen 18 - Diagrama de funcionamiento mensajería interna

1/1ª El sistema frontal realiza una llamada HTTP al servicio Balanceado.

2/2ª El servicio balanceado envía la petición a uno de los adaptadores de canal de cualquiera de los nodos.

3/3ª El adaptador de canal, después de realizar las acciones correspondientes, envía la petición a la cola del EMS en la que están escuchando todas las instancias del proceso simple.

4/4ª Una de las instancias del proceso simple recibe la petición y la procesa.

3.1.1 Tipos de adaptadores

Durante este apartado se definirá el diseño de los diferentes adaptadores que formarán el ESB. Se dividirán en 3 subconjuntos principales:

- Asincrono
- Síncrono
- Batch

Como aclaración antes de comenzar este punto, comentar que esta parte se hablará de procesos y subprocesos, entendiendo cada uno de ellos de la siguiente forma:

Procesos: Unidad lógica que abarca una funcionalidad propia del Framework que se definirá. En dicho framework se podrán observar procesos de:

- Starter
- Map
- Routing
- Error

Subproceso: Unidad organizativa lógica, que será utilizada para facilitar la reutilización y el entendimiento de los procesos nombrados anteriormente. De modo que ciertas partes del código de cada proceso serán encapsulados en dichos subprocesos a modo de funcionalidades atómicas.

3.1.1.1 Adaptadores Asíncronos

En este apartado se definirá el diseño de los adaptadores para integraciones asíncronas. Como se ha comentado a lo largo del documento, las integraciones asíncronas son aquellas que no precisan de retorno. En este tipo de integraciones están pensadas para acciones que no requieren del ACK (acknowledgement) de dicha acción, como pueden ser inserción de entidades en sistemas legacy, borrado de entidades en operaciones no transaccionales etc.

Este tipo de patrones están formados principalmente por dos adaptadores, adaptador de exportación y adaptador de importación.

El adaptador de exportación es el encargado de la comunicación entre el sistema origen de la integración y el ESB.

El adaptador de importación está orientado a la comunicación entre el ESB y el sistema destino.

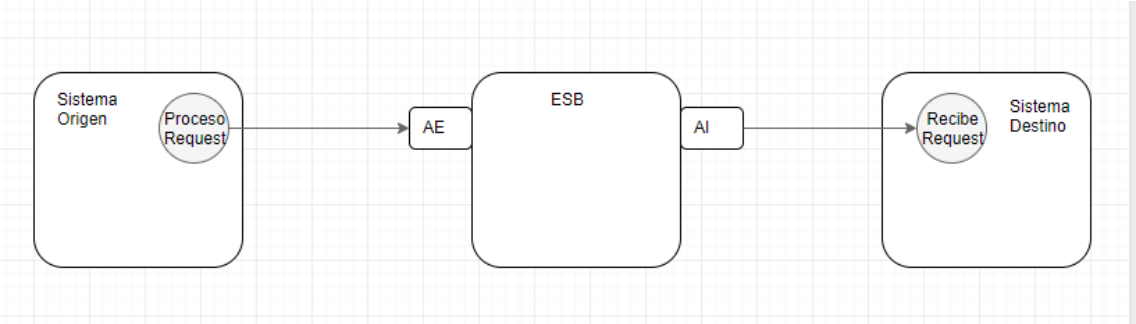


Imagen 19 - Diagrama de flujo de comunicación asíncrona

3.1.1.2 Adaptadores Batch

En este apartado se definirá el diseño de los adaptadores para integraciones batch. Las integraciones batch son aquellas que no se ejecutan en el momento, es decir, que almacenan las peticiones de entrada hasta la llegada de un evento que desencadene su posterior envío (Normalmente un timer).

Este tipo de integraciones son muy útiles para integrar sistemas online (aplicaciones web, sistemas CRM modernos…) con los sistemas legacy, que suelen preferir la carga de información mediante ficheros batch.

Este tipo de patrones están formados por dos adaptadores, adaptador N-1 y adaptador de 1-N. Se pueden entender estos adaptadores en función del sentido del flujo:

Adaptador N-1
Adaptador que recibe N peticiones y las concatena en un fichero hasta que llegue un evento que desencadene el cierre del mismo y el posterior envío vía FTP.

Adaptador 1-N
Este adaptador cubre exactamente el flujo contrario al anterior. Recibe un fichero vía FTP y lo traduce a N peticiones al sistema destino.

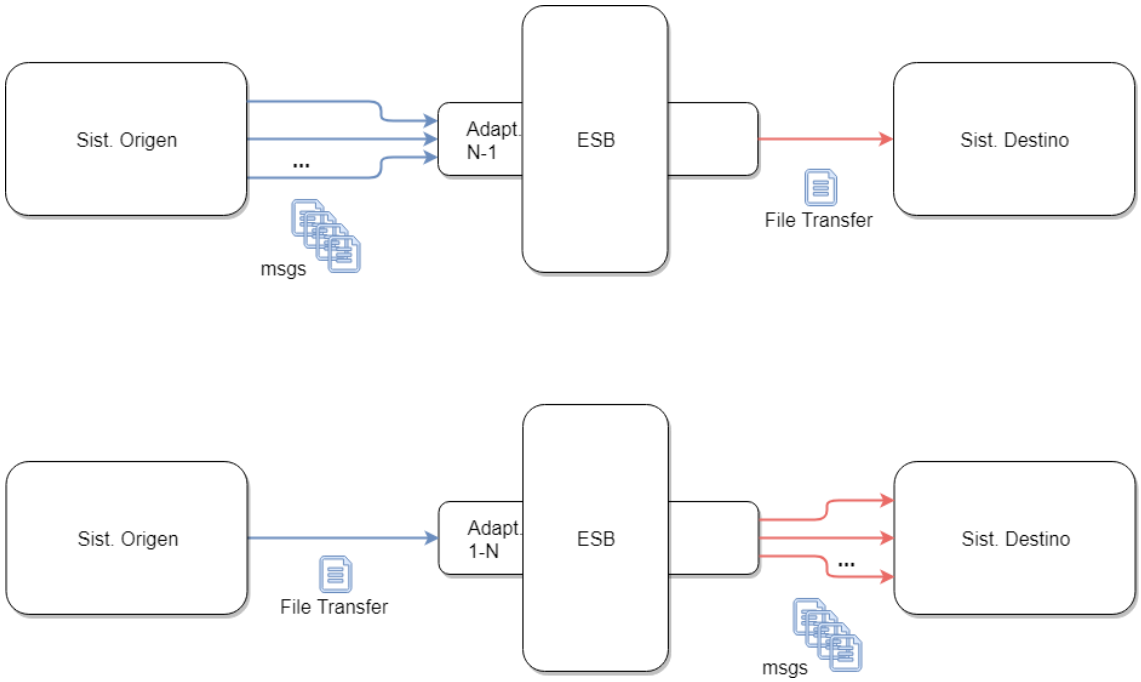


Imagen 20 - Diagrama de flujo de comunicación N-1/1-N

3.1.1.3 Adaptadores Síncronos

En este apartado se definirán el último grupo de adaptadores que se definirán en este trabajo. Estos son los adaptadores síncronos. Este tipo de adaptadores tienen como principal característica la gestión de una respuesta en el mismo flujo de ejecución de la petición entrante.
Esta es la parte SOA del framework, ya que la idea principal de estos adaptadores es exponerlos como web services.

Un web service es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios Web. Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares. Es una máquina que atiende las peticiones de los clientes web y les envía los recursos solicitados. [08]

Estos son los adaptadores síncronos que se van a tratar:

Adaptador de canal:
Adaptador que se expone al sistema frontal. Este adaptador sirve como pasarela entre el frontal y el proceso simple o el orquestado. Si principal función es dotar de cierta independencia a los diferentes sistemas frontales que invocan a una misma funcionalidad. Teniendo tantos adaptadores de canal como sistemas frontales invoquen a un PS (Proceso simple) o un ORQ (Orquestado).

Adaptador Proceso Simple:
Adaptador, que una vez recibida una petición HTTP realiza una transformación del mensaje recibido y envía la petición a un sistema Backend. Este sistema backend devuelve una respuesta que el adaptador, transforma y la propaga hasta el sistema origen.

Adaptador Orquestado:
Adaptador, que una vez recibida una petición HTTP realiza una serie de llamadas orquestadas a diferentes Procesos simples, que a su vez invocan a diferentes backend. Finalmente con todas las respuestas obtenidas por los diferentes backend forma una respuesta y la devuelve al sistema origen.

A continuación se expone un diagrama a alto nivel con la integración de los 3 adaptadores:

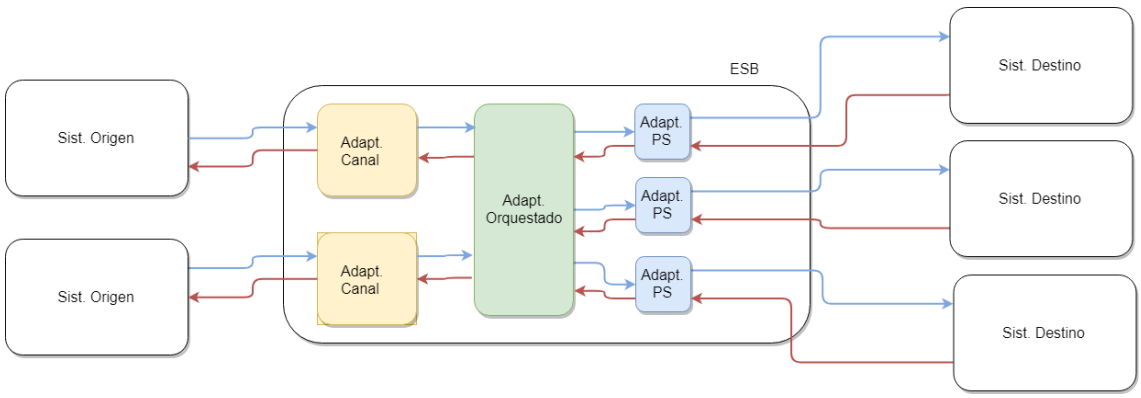


Imagen 21 – Diagrama flujo síncrono

En el diagrama se puede ver un escenario de un servicio web orquestado al que invocan dos sistemas frontales.

Estos dos sistemas frontales invocan cada uno a un adaptador de canal. Con esto se consigue independencia entre ambos sistemas, de modo que posibles cambios sobre el interfaz de uno no afectarían al otro. A su vez al no contener lógica de negocio, esta medida no afecta a la reutilización del código ya que ambos sistemas pasan por el mismo adaptador de orquestación y procesos simples.

Como se ha comentado, los adaptadores de canal llaman al adaptador de orquestación, que bajo una lógica de negocio realiza una serie de llamadas a diferentes adaptadores de procesos simples que a su vez realizan invocaciones a diferentes backends.

3.2 DISEÑO DE LOS ADAPTADORES

Durante este punto se expondrá el diseño detallado paso a paso de cada uno de los diferentes adaptadores que formarán el Framework del ESB.

3.2.1 Adaptador de exportación

Durante este punto se definirá el adaptador de exportación que forma parte del patrón asíncrono del FW del ESB.

Este adaptador está definido con la intención de recoger las peticiones entrantes dentro del flujo asíncrono.

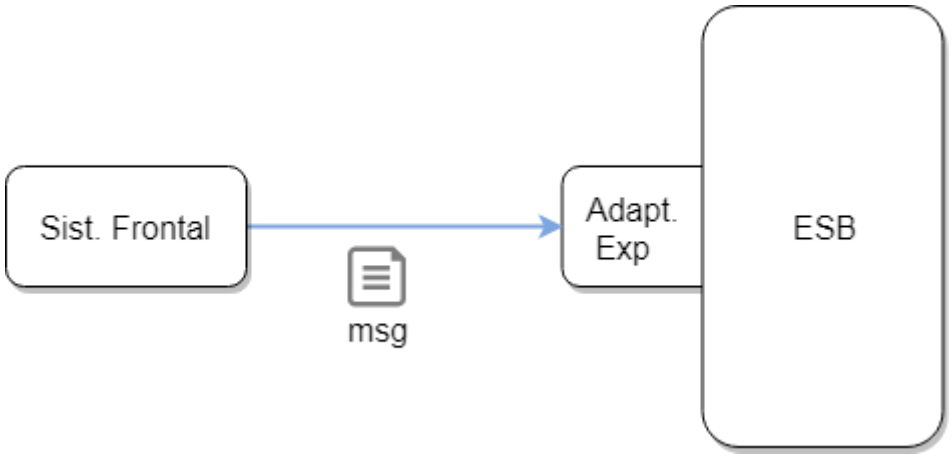


Imagen 22 - Flujo alto nivel adaptador exportación

3.2.1.1 Estructura del adaptador

Para poder adaptar los desarrollos que se vayan a realizar sobre el FW del ESB, se tendrá que seguir una serie de pautas necesarias para el acoplamiento entre estos.
Estas pautas son necesarias debido a que el propio FW cuenta con muchas configuraciones dinámicas (invocaciones a procesos de negocio) las cuales dependen de la estructura de las carpetas del proyecto entre otras cosas.

Por esta razón, los adaptadores de exportación que se desarrollen bajo el FW del ESB han de contar con la siguiente estructura de carpetas.

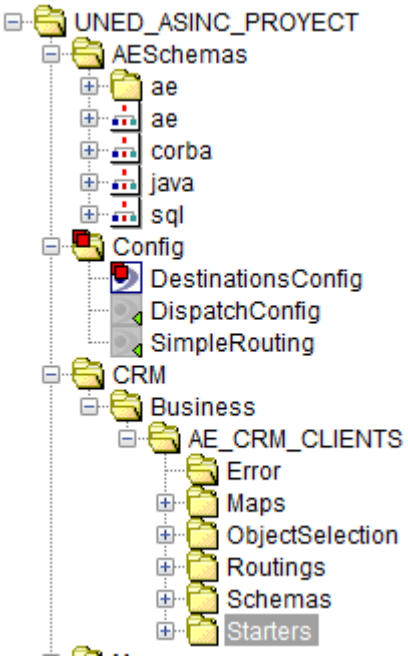


Imagen 23 - Estructura de carpetas de un proyecto de adaptador de exportación

Como se observa en la imagen 24 la estructura del proyecto del adaptador de exportación se forma por las siguientes carpetas:

Nombre Carpeta	Nivel	Descripción
NOMBRE_ASYNC_PROJECT	0	Carpeta root del proyecto, en ella está contenidos todos los procesos y funcionalidades.
Config	1	Carpeta destinada a los ficheros de configuración.
NOMBRE_PROYECTO	1	Carpeta con el nombre del proyecto, dentro de esta carpeta se encuentra todo el código del adaptador.
Business	2	Carpeta que contiene el código relativo al requisito del negocio.
AE_NOMBRE_ADAPTADOR	3	Carpeta que contiene todo lo relativo al adaptador.
Error	4	Carpeta donde se encuentran los procesos encargados de la gestión de errores.
Maps		Carpeta donde se encuentran los procesos encargados de realizar los mapeos.
ObjectSelection	4	Carpeta donde se encuentran los procesos encargados de la selección de la entidad
Routing	4	Carpeta donde se encuentran los procesos encargados de la lógica de

		enrutado
Schemas	4	Carpeta donde se almacenan los interfaces del proceso (XSD, WSDL, WADL, SWAGGER…)
Starters	4	Carpeta donde se encuentran los procesos de starter, es decir, los procesos que reciben la petición del sistema frontal…
SharedResources	2	Carpeta donde se encuentran las conexiones a base de datos, http, jms… así como cualquier elemento de uso común dentro del proyecto.
Deploy	1	Carpeta donde se encuentran los artefactos instalables.

Tabla 2 - Estructura del proyecto

3.2.1.2 Nomenclatura

Al igual que la estructura del proyecto, se necesita seguir una nomenclatura para que el framework funcione correctamente. La nomenclatura a seguir es la siguiente:

Tipo de elemento	Ejemplo Nomenclatura	Descripción
Proceso Starter	AE_CRM_CLIENTS_EXPORT_JMS_STARTER	AE: tipo de adaptador. CRM: Sistema origen. CLIENTS: Entidad EXPORT: Función. JMS: Protocolo STARTER: Tipo de proceso
Proceso Map	Map_Client	Client: Entidad

3.2.1.3 Tipos de procesos

Dentro de cada adaptador, se podrán diferenciar claramente dos tipos de procesos que serán los que hagan posible su ejecución:

- Procesos de negocio
- Procesos de framework

Procesos de negocio

Este tipo de procesos son los destinados a albergar la lógica de negocio del adaptador. Esta lógica está dividida en las siguientes funcionalidades:

Map: Proceso destinado a realizar las transformaciones. Se divide en dos tipos de mapper:

Request: Transformaciones dedicadas a la petición de entrada.
Response: Transformaciones dedicadas a la respuesta en la salida.
Error: Transformaciones dedicadas a los errores controlados.

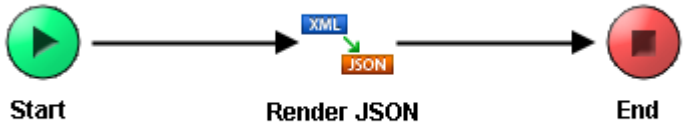


Imagen 24 – Ejemplo de transformación de XML a JSON.

Routing: Proceso dedicado a realizar la lógica de enrutado dependiendo de la petición recibida.

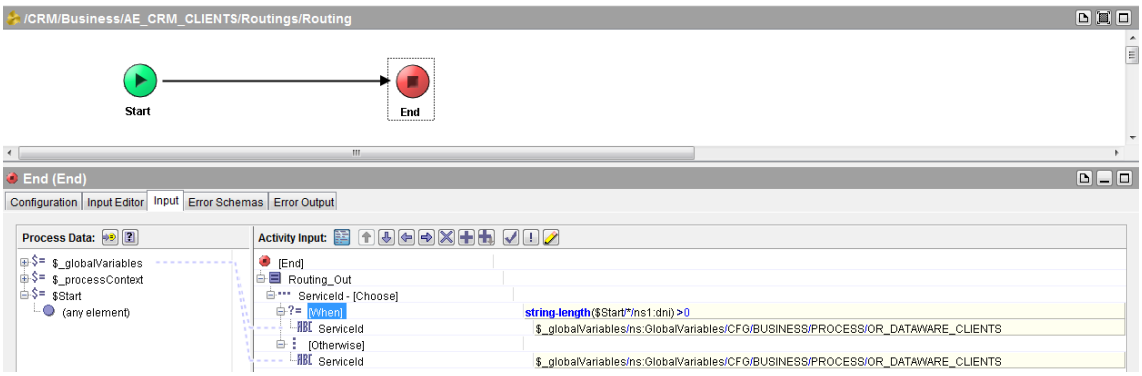


Imagen 25 – Ejemplo de routing mediante lógica en XPath

ObjectSelection: Proceso dedicado a realizar la lógica de la selección de la entidad sobre la que va a trabajar el flujo. Esta decisión se tomará de forma dinámica dependiendo de la petición entrante.

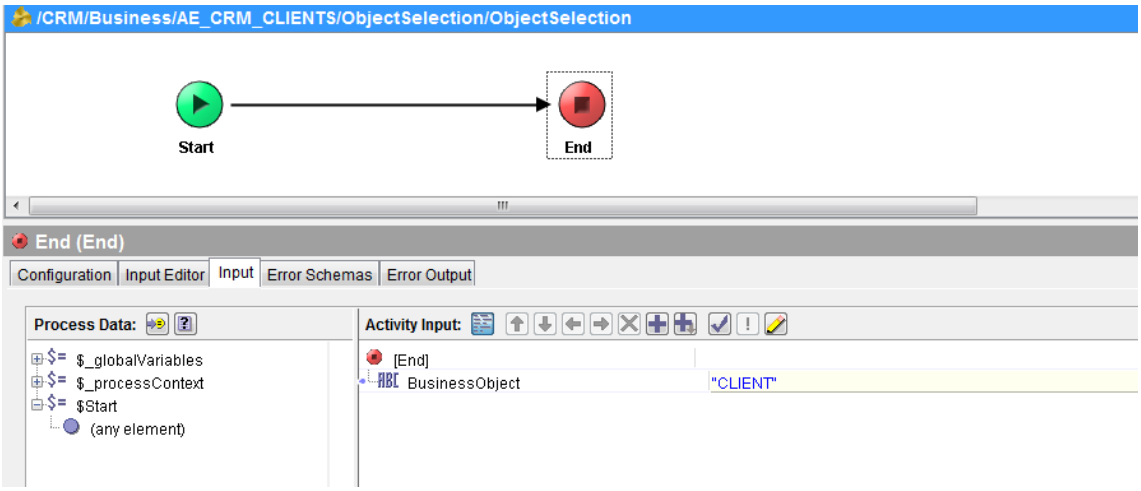


Imagen 26 – Ejemplo de selección de entidades mediante XPath

Procesos de framework

Este tipo de procesos son completamente estándar. No es necesario modificarlos para el desarrollo de ningún adaptador. Son una serie de plantillas cerradas que de forma dinámica irán trazando los mensajes e invocando a los procesos de negocio. Este tipo de procesos son lo que se denomina "Framework" y facilitan la labor del desarrollador abstrayéndolo únicamente a la lógica de negocio. Además gracias a que no se pueden modificar dotan a los adaptadores de una homogeneidad imprescindible a la hora de llevar un soporte adecuado de la herramienta.
Los procesos del framework para el adaptador que tratamos en este punto son los siguientes:

Export_Pattern: Proceso principal del adaptador, actúa a modo de orquestado dirigiendo el flujo por el resto de procesos que conforman el framework para el adaptador que estamos tratando.

PreActions: Proceso que contiene una serie de subprocesos que llevan a cabo las tareas necesarias en el inicio del flujo.

LoadTargetService: Proceso que mediante el resultado obtenido del proceso dinámico de routing. Obtiene las configuraciones necesarias para realizar el envío en el proceso de PostActions.

PostActions: Proceso que se encarga del envío del mensaje al destinatario y de auditar el fin del flujo del adaptador.

3.2.1.4 Diseño por pasos

A continuación se mostrará el diseño paso a paso de dicho adaptador, desde la recepción del mensaje al envío a su adaptador de importación correspondiente.

PASO 1 Recepción del mensaje

El primer paso de todos es la recepción del mensaje, este puede recepcionarse bajo cualquiera de estos protocolos:

- HTTP
- JMS
- FTP

Para ello se definirá un proceso de entrada que será el encargado de recepcionarlo.

Este proceso será distinto dependiendo del protocolo de comunicación establecido, quedando de esta forma:

HTTP

Se definirá un proceso que se encargará de levantar un socket http para escuchar en un puerto determinado las peticiones.
Una vez recibida la petición la enviará a un proceso del framework que se encargará de orquestar las llamadas dinámicas al resto de procesos de negocio.
Al ser un adaptador de flujo asíncrono no es necesario mapear la response.

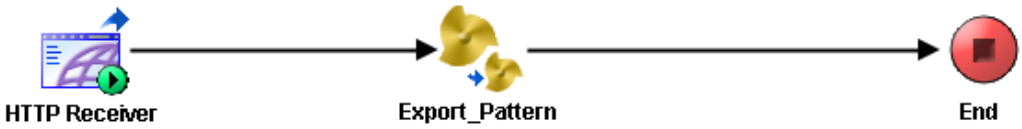


Imagen 27 - Proceso Starter HTTP

En esta imagen se describe el proceso de Starter:

1. HTTP Receiver: Tarea que dada una URL y un Puerto. Levanta un socket HTTP para recepcionar peticiones.
2. Export_Pattern: Una vez recabada la petición se manda de forma interna a un proceso del framework, que comienza a realizar una serie de acciones e invocaciones dinámicas.

3. End: Una vez el proceso del framework realiza todas las acciones el proceso termina su ejecución.

PASO 2 Gestión de la petición por el Framework.

En este paso el proceso genérico del framework (Export_Pattern) realiza una serie de acciones denominadas preacciones y postacciones, para procesar la petición.
Dentro de estas acciones se encuentran las llamadas dinámicas a los procesos de negocio. Este proceso es un proceso que forma parte del Framework, es decir no es modificable por el desarrollador, con la información proporcionada en la entrada y la estructura/nomenclatura definidas es capaz de invocar y trazar todos los procesos que contienen la lógica de negocio.

El flujo de este proceso se detalla a continuación.

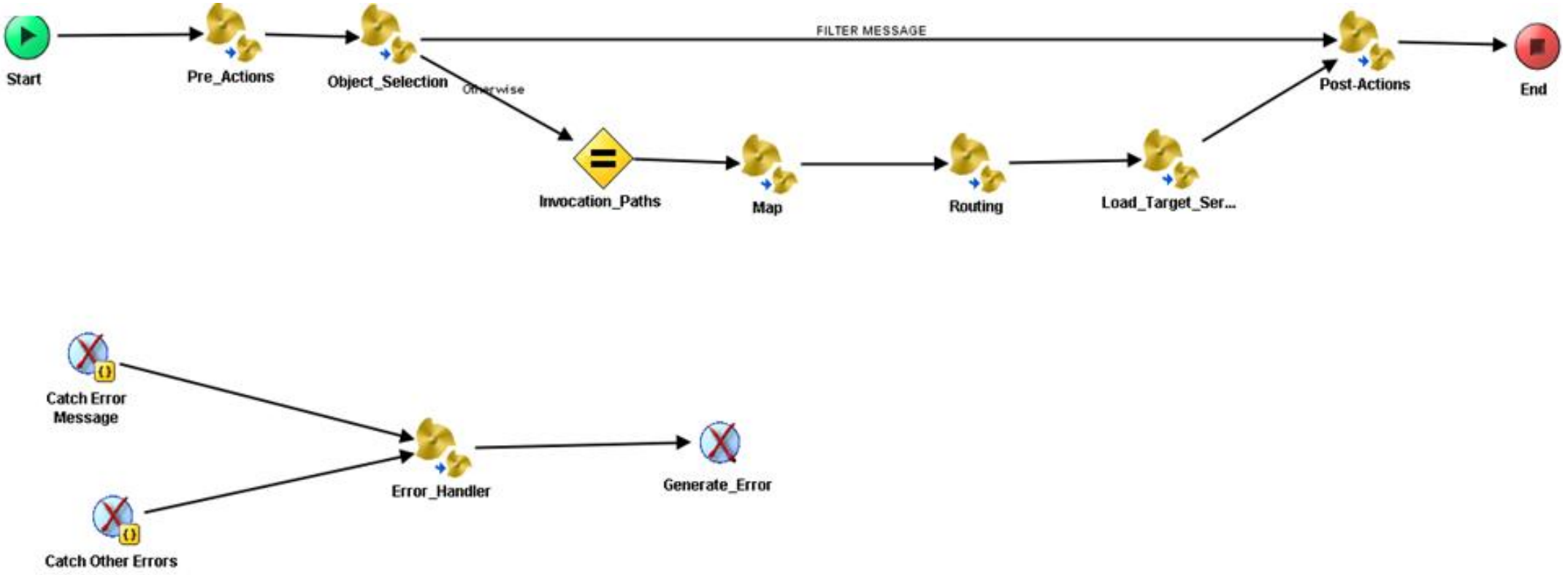


Imagen 28 – Flujo Proceso principal Adaptador_Exportación FW

En la imagen anterior se describe el proceso de Export-Pattern. Estos son los pasos que realiza:

1. Al proceso le llega la petición del Starter con la siguiente información:

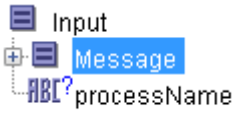


Imagen 29 – Esquema de entrada Export-Pattern

2. A continuación se ejecutan las preacciones que están formadas por las siguientes acciones:

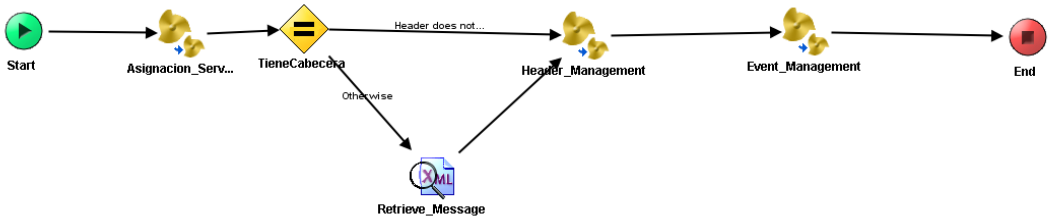


Imagen 30 – Flujo con la ejecución de las distintas preacciones.

   a. La primera es la asignación de un identificador único de la ejecución, mediante el algoritmo de generación de UUID.
   b. Después se genera la cabecera de arquitectura en el caso de que no esté ya generada. Esta cabecera estará generada siempre y cuando el adaptador que se está ejecutando no es el primero del flujo, ya que se va propagando. Esta cabecera consta con los siguientes campos:

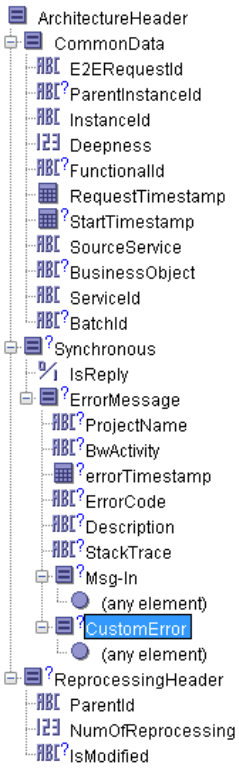


Imagen 31 – Parámetros de la cabecera de arquitectura o gestión.

c. Por último se gestiona la creación de eventos, dependiendo del nivel de trazabilidad establecido. Esta gestión consistirá en el envío del evento a un 3er sistema y la escritura en un log de ejecución.

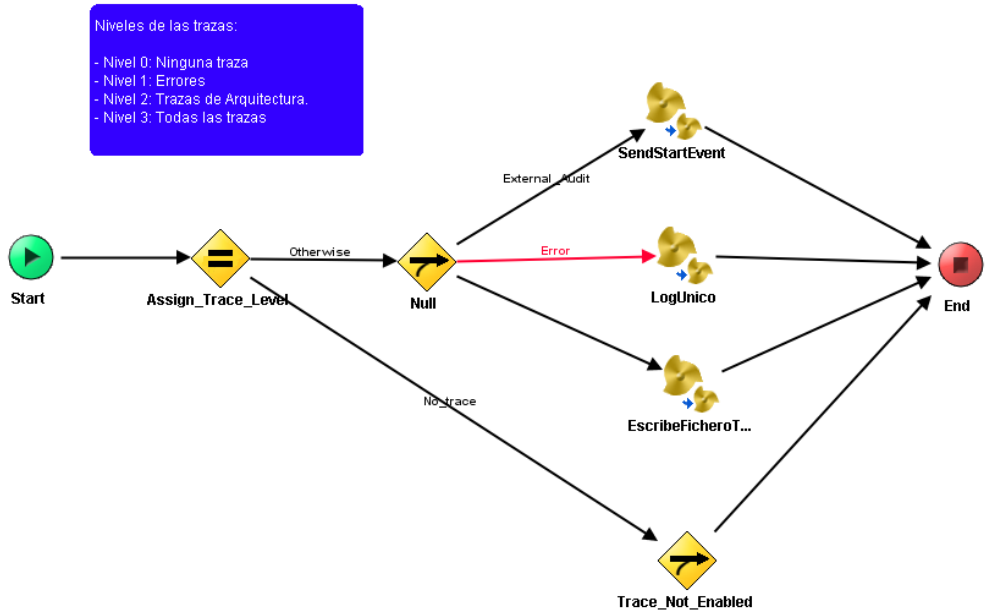


Imagen 32 – Flujo de gestión de eventos.

3. El tercer paso consiste en la selección del objeto. Este paso se encarga de comprobar si se han definido una o varias entidades para este flujo. En caso de ser así se realizará una llamada dinámica al proceso de negocio “ObjectSelection” para discernir de que entidad se trata.

4. A continuación con la entidad resultante del paso anterior se formarán las rutas dinámicas para invocar a los procesos de negocio de mapeos y routing correspondientes con esa entidad.

5. El siguiente paso es la invocación dinámica al proceso de negocio de mapeos. En este paso se invocará con las rutas generadas dinámicamente en el punto anterior al proceso que contiene las transformaciones del mensaje de entrada según los requisitos del negocio.

6. Una vez realizadas las transformaciones necesarias según los requisitos de negocio, se ejecutará de la misma forma dinámica el proceso de Routing. Este proceso tendrá como resultado el identificador del destino del flujo del adaptador de exportación.

7. Con el identificador del destino recién obtenido, mediante el proceso de cargar destino obtendremos el protocolo de transporte y el endpoint del destino. Esto se realiza mediante una búsqueda en un fichero de configuración xml con el identificador del destino.

8. Por último se llevan a cabo las postacciones del flujo del adaptador de exportación que contiene las siguientes funcionalidades:

Imagen 33 – Flujo con las post acciones

a. La primera de las acciones es la del envío del mensaje al destino. Este envío se realiza con las configuraciones obtenidas de forma dinámica en el paso anterior.

b. El último paso es el envío del evento de fin tanto a un 3er sistema como al componente de escritura en logs. Este proceso es el mismo que el que se invoca en las preacciones.

3.2.2 Adaptador de importación

Durante este punto se definirá el adaptador de importación que forma parte del patrón asíncrono del FW del ESB.

Este adaptador está definido con la intención de recoger las peticiones recibidas por el adaptador de exportación y las envía al sistema destino dentro del flujo asíncrono.

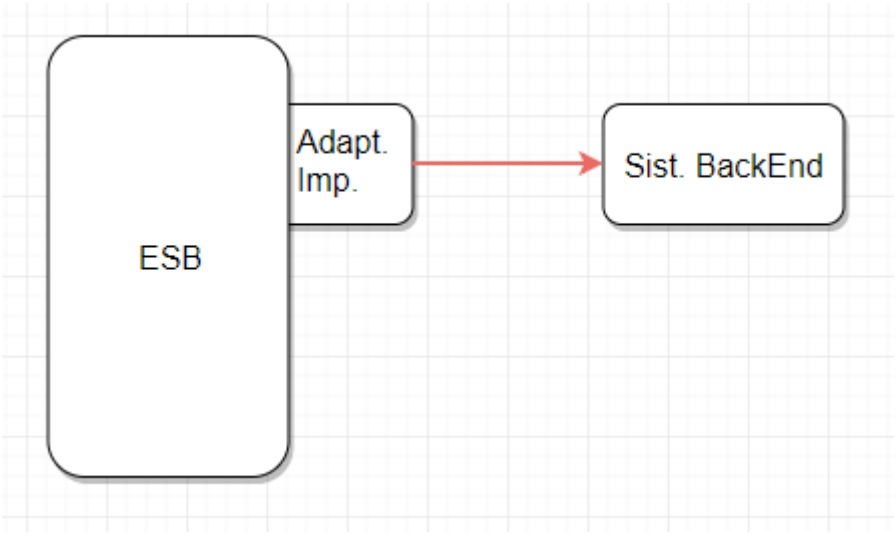


Imagen 34 - Flujo alto nivel adaptador importación

3.2.2.1 Estructura del adaptador

Para poder adaptar los desarrollos que se vayan a realizar sobre el FW del ESB, se tendrá que seguir una serie de pautas necesarias para el acoplamiento entre estos.
Estas pautas son necesarias debido a que el propio FW cuenta con muchas configuraciones dinámicas (invocaciones a procesos de negocio) las cuales dependen de la estructura de las carpetas del proyecto entre otras cosas.

Por esta razón, los adaptadores de importación que se desarrollen bajo el FW del ESB han de contar con la siguiente estructura de carpetas.

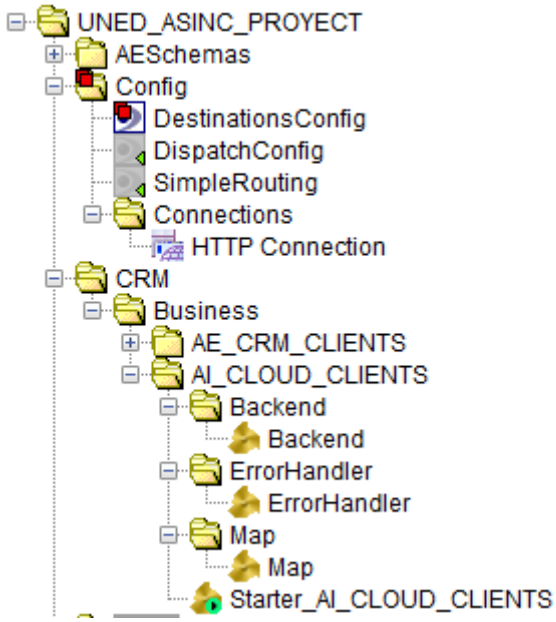


Imagen 35 - Estructura de carpetas de un proyecto de adaptador de exportación

Como se observa en la imagen 24 la estructura del proyecto del adaptador de importación se forma por las siguientes carpetas:

Nombre Carpeta	Nivel	Descripción
NOMBRE_ASYNC_PROJECT	0	Carpeta root del proyecto, en ella está contenidos todos los procesos y funcionalidades.
Config	1	Carpeta destinada a los ficheros de configuración.
NOMBRE_PROYECTO	1	Carpeta con el nombre del proyecto, dentro de esta carpeta se encuentra todo el código del adaptador.
Business	2	Carpeta que contiene el código relativo al requisito del negocio.
AI_NOMBRE_ADAPTADOR	3	Carpeta que contiene todo lo relativo al adaptador.
ErrorHandler	4	Carpeta donde se encuentran los procesos encargados de la gestión de errores.
Map		Carpeta donde se encuentran los procesos encargados de realizar los mapeos.
BackEnd	4	Carpeta donde se encuentran los procesos encargados del envío al sistema BackEnd
Schemas	4	Carpeta donde se almacenan los interfaces del proceso (XSD, WSDL, WADL, SWAGGER…)

Starters	4	Carpeta donde se encuentran los procesos de starter, es decir, los procesos que reciben la petición del sistema frontal…
SharedResources	2	Carpeta donde se encuentran las conexiones a base de datos, http, jms… así como cualquier elemento de uso común dentro del proyecto.
Deploy	1	Carpeta donde se encuentran los artefactos instalables.

Tabla 3 - Estructura del proyecto

3.2.2.2 Nomenclatura

Al igual que la estructura del proyecto, se necesita seguir una nomenclatura para que el framework funcione correctamente. La nomenclatura a seguir es la siguiente:

Tipo de elemento	Ejemplo Nomenclatura	Descripción
Proceso Starter	AE_CLOUD_CLIENTS_IMPORT_JMS_STARTER	AE: tipo de adaptador. CLOUD: Sistema origen. CLIENTS: Entidad IMPORT: Función. JMS: Protocolo STARTER: Tipo de proceso
Proceso Map	Map	

Tabla 4 – Nomenclatura de procesos

3.2.2.3 Tipos de procesos

Dentro de cada adaptador, se podrán diferenciar claramente dos tipos de procesos que serán los que hagan posible su ejecución:

- Procesos de negocio
- Procesos de framework

Procesos de negocio

Este tipo de procesos son los destinados a albergar la lógica de negocio del adaptador. Esta lógica está dividida en las siguientes funcionalidades:

Backend: Proceso destinado a realizar el envío al sistema destino del flujo. Dependiendo del protocolo de envío que se acuerde (HTTP/JMS/FTP…), el desarrollo del proceso variará.

Imagen 36 – Flujo del proceso Backend mediante protocolo HTTP.

ErrorHandler: Proceso destinado a gestionar el tratamiento de error. Al tratarse de un proceso asíncrono el cual no tiene respuesta el tratamiento del error residirá en trazar el error.

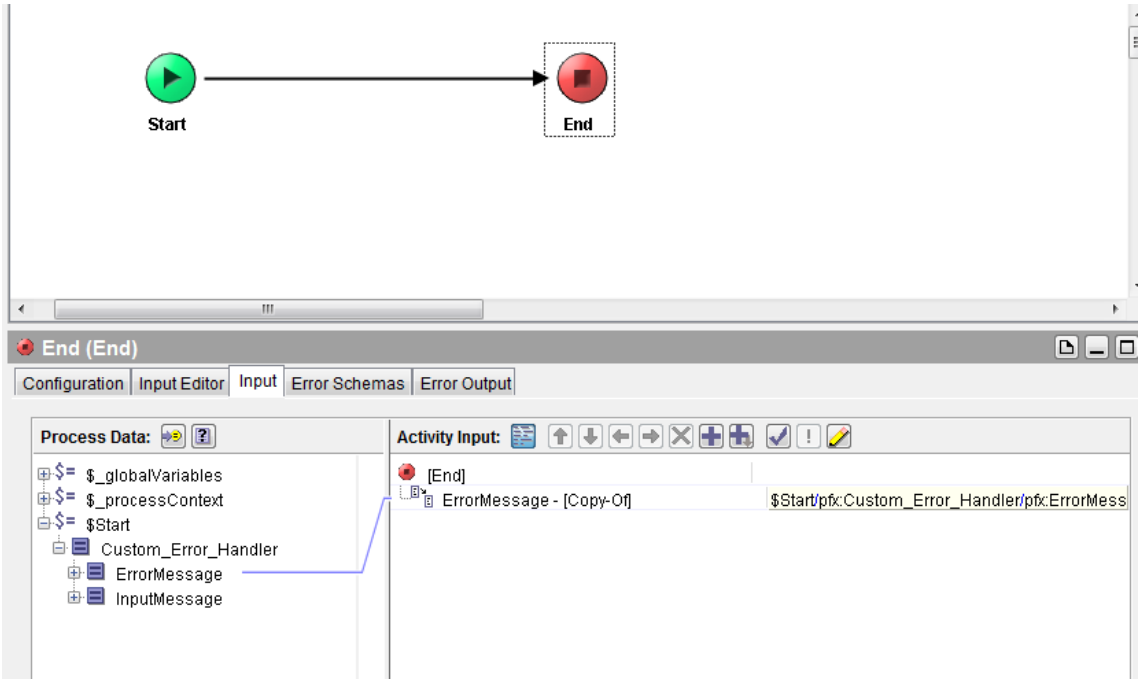


Imagen 37 – Proceso de tratamiento de error asíncrono.

Map: Proceso destinado a realizar las transformaciones. El tipo de mapeo que se llevará a cabo en el adaptador de importación es el destinado a realizar las transformaciones para el envío de la información al Backend: Transformaciones dedicadas a la petición de entrada.

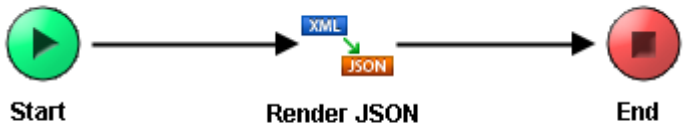


Imagen 38 – Ejemplo de transformación de XML a JSON.

Procesos de framework

Este tipo de procesos son completamente estándar. No es necesario modificarlos para el desarrollo de ningún adaptador. Son una serie de plantillas cerradas que de forma dinámica irán trazando los mensajes e invocando a los procesos de negocio. Este tipo de procesos son lo que se denomina “Framework” y facilitan la labor del desarrollador abstrayéndolo únicamente a la lógica de negocio. Además gracias a que no se pueden modificar dotan a los adaptadores de una homogeneidad imprescindible a la hora de llevar un soporte adecuado de la herramienta.
Los procesos del framework para el adaptador que tratamos en este punto son los siguientes:

Import_Pattern: Proceso principal del adaptador, actúa a modo de orquestado dirigiendo el flujo por el resto de procesos que conforman el framework para el adaptador que estamos tratando.

PreActions: Proceso que contiene una serie de subprocesos que llevan a cabo las tareas necesarias en el inicio del flujo.

ServiceInvocation_Event: Proceso que traza el envío que se va a realizar al BackEnd justo antes de llevarlo a cabo.

PostActions: Proceso que se encarga del envío del mensaje al destinatario y de auditar el fin del flujo del adaptador.

3.2.2.4 Diseño por pasos

A continuación se mostrará el diseño paso a paso de dicho adaptador, desde la recepción del mensaje al envío al backEnd correspondiente.

PASO 1 Recepción del mensaje

El primer paso de todos es la recepción del mensaje, este se recepcionará bajo el protocolo de mensajería interna JMS

- JMS

Para ello se definirá un proceso de entrada que será el encargado de recepcionarlo.

Se definirá un proceso que se encargará de levantar un receiver JMS para escuchar en una queue definida la petición enviada desde el Adaptador de exportación.
Una vez recibida la petición la enviará a un proceso del framework que se encargará de orquestar las llamadas dinámicas al resto de procesos de negocio.
Al ser un adaptador de flujo asíncrono no es necesario mapear la response.

Imagen 39 – Proceso starter JMS

En esta imagen se describe el proceso de Starter:

4. JMS Queue Receiver: Tarea que dada un servidor JMS y una queue, levanta un lisener JMS para recepcionar peticiones.
5. Import_Pattern: Una vez recabada la petición se manda de forma interna a un proceso del framework, que comienza a realizar una serie de acciones e invocaciones dinámicas.
6. End: Una vez el proceso del framework realiza todas las acciones el proceso termina su ejecución.

PASO 2 Gestión de la petición por el Framework.

En este paso el proceso genérico del framework (Import_Pattern) realiza una serie de acciones denominadas preacciones y postacciones, para procesar la petición.

Dentro de estas acciones se encuentran las llamadas dinámicas a los procesos de negocio. Este proceso es un proceso que forma parte del Framework, es decir no es modificable por el desarrollador, con la información proporcionada en la entrada y la estructura/nomenclatura definidas es capaz de invocar y trazar todos los procesos que contienen la lógica de negocio.

El flujo de este proceso se detalla a continuación.

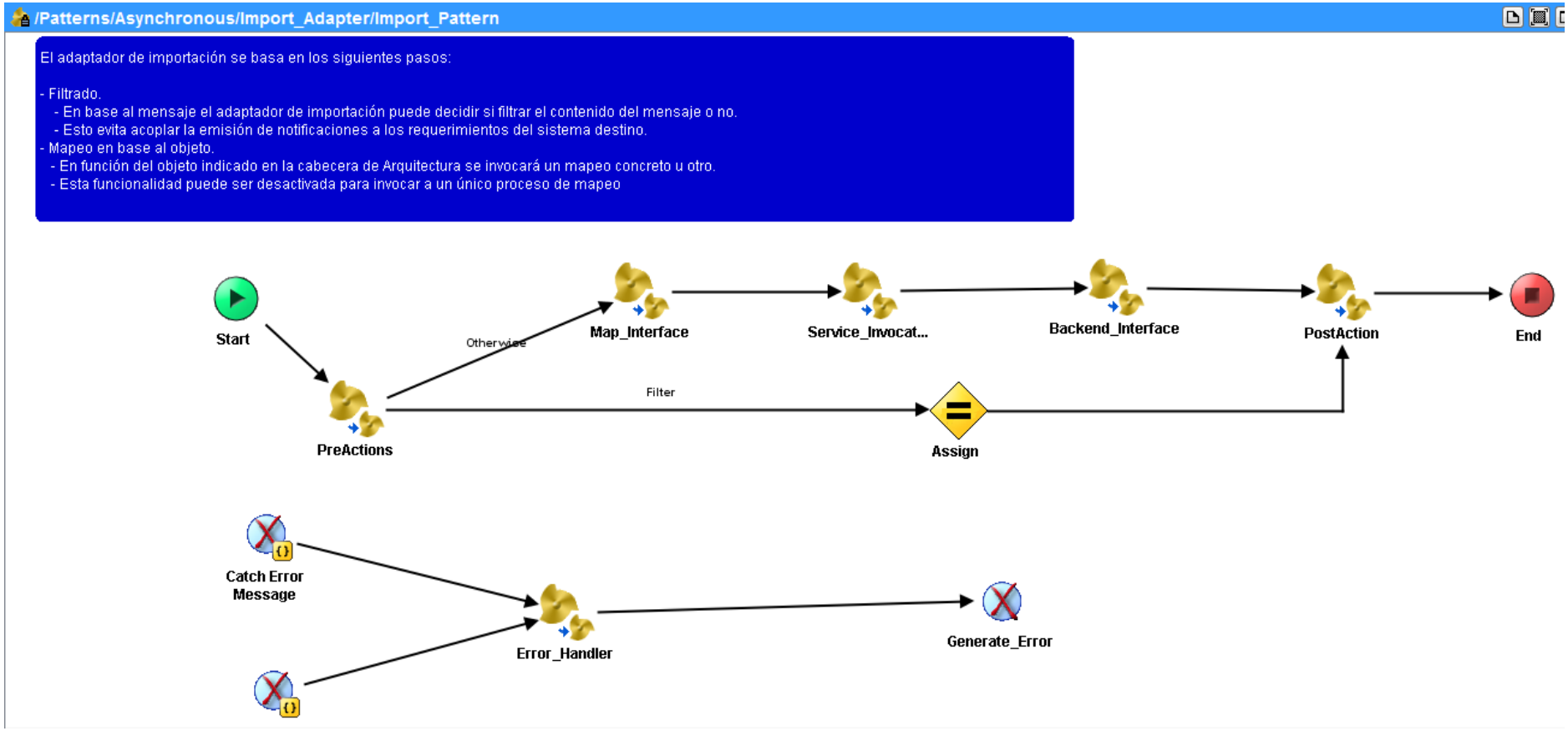


Imagen 40 – Flujo Proceso principal Adaptador_Importación FW

En la imagen anterior se describe el proceso de Import-Pattern. Estos son los pasos que realiza:

1. Al proceso le llega la petición del Starter con la siguiente información:

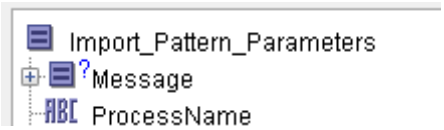


Imagen 41 – Esquema de entrada Import-Pattern

En ella se detalla el mensaje que se ha recibido del adaptador de exportación y el nombre del proceso.

2. A continuación se ejecutan las pre acciones que están formadas por las siguientes acciones:

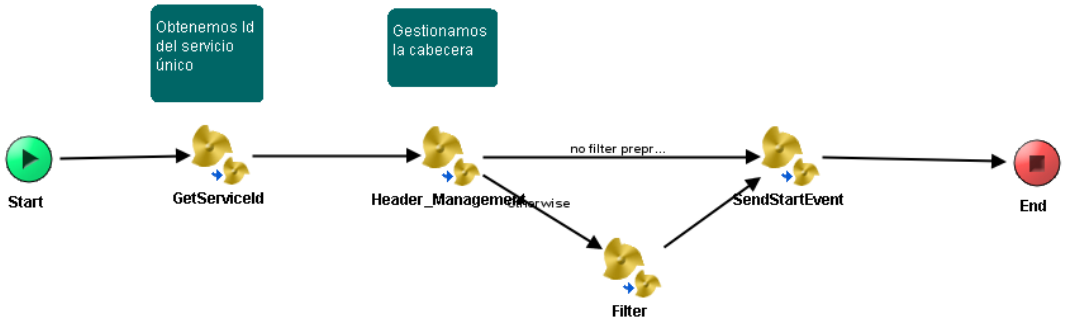


Imagen 42 – Flujo con la ejecución de las distintas preacciones.

   a. La primera es la asignación de un identificador único de la ejecución, mediante el algoritmo de generación de UUID.
   b. Después se genera la cabecera de arquitectura en el caso de que no esté ya generada. Esta cabecera estará generada siempre y cuando el adaptador que se está ejecutando no es el primero del flujo, ya que se va propagando. Esta cabecera consta con los siguientes campos:

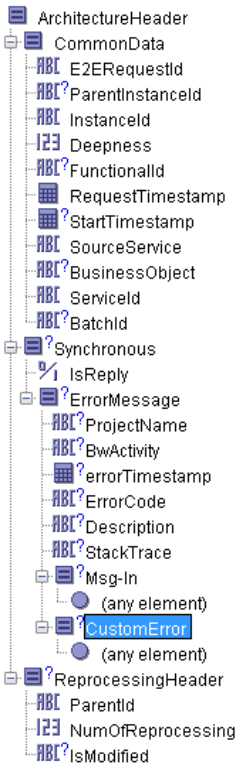


Imagen 43 – Parámetros de la cabecera de arquitectura o gestión.

c. Por último se gestiona la creación de eventos, dependiendo del nivel de trazabilidad establecido. Esta gestión consistirá en el envío del evento a un 3er sistema y la escritura en un log de ejecución.

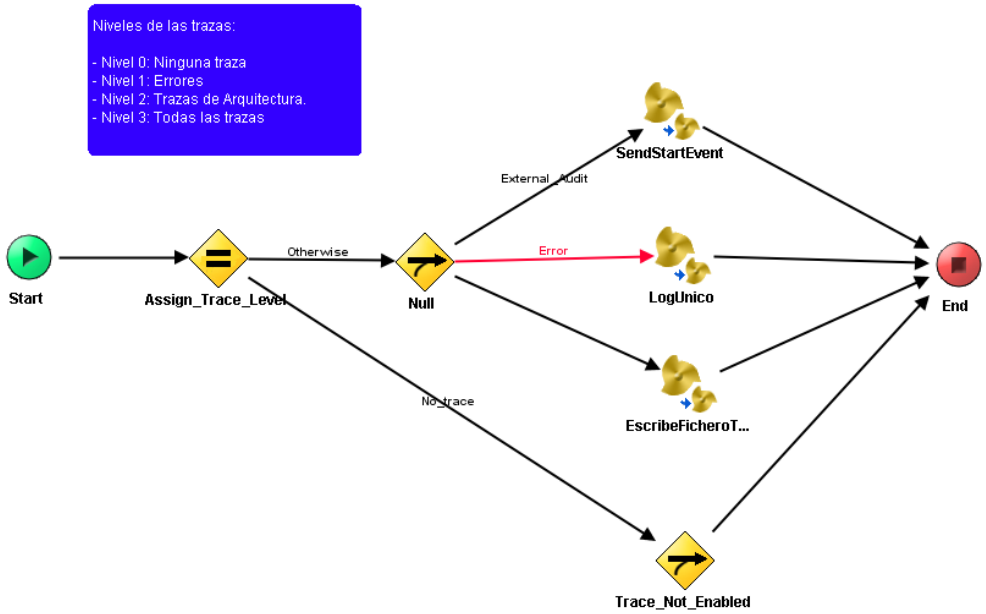


Imagen 44 – Flujo de gestión de eventos.

3. El siguiente paso es la invocación dinámica al proceso de negocio de mapeos. En este paso se invocará con las rutas generadas dinámicamente en el punto anterior al proceso que contiene las transformaciones del mensaje de entrada según los requisitos del negocio.

4. Una vez realizadas las transformaciones necesarias según los requisitos de negocio, se trazará el mensaje resultante en los logs del ESB y se enviará a un tercer sistema en el caso de requerirlo.

5. El paso número 5 es el envío del mensaje al backend, esto se realiza mediante la llamada dinámica al proceso de negocio “BackEnd”.

6. Por último se llevan a cabo las postacciones del flujo del adaptador de exportación que contiene las siguientes funcionalidades:

Imagen 45 – Flujo con las post acciones

   a. La primera y última acción de las postacciones del adaptador de importación es el envío del evento de fin de ejecución con el resultado obtenido y la escritura de las trazas correspondientes.

3.2.3 Adaptador N-1

Durante este punto se definirá el adaptador N-1 que forma parte del patrón batch del FW del ESB.

Este adaptador está definido con la intención de recoger las peticiones recibidas de forma online por un sistema online y enviarlas tras la llegada de un evento de tiempo que desencadena el cierre y posterior envío del fichero.

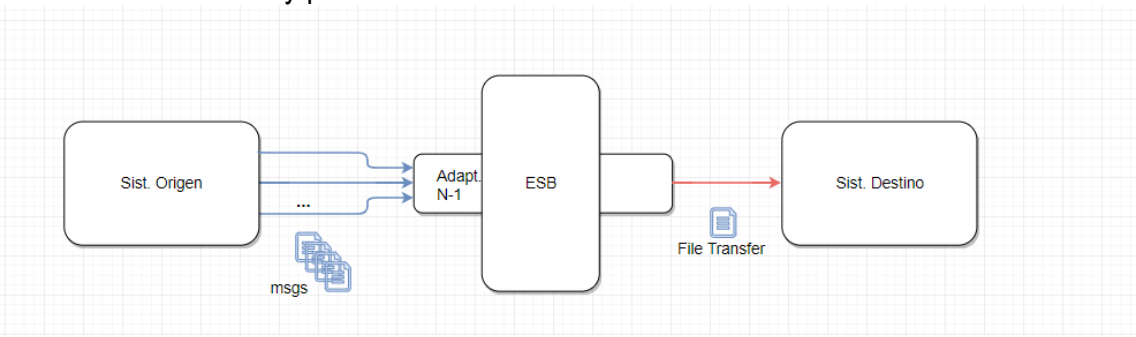


Imagen 46 - Flujo alto nivel adaptador N-1

Debido a que durante el cierre y envío del fichero, pueden seguir llegando peticiones, es imprescindible tener un adaptador de exportación delante del adaptador N-1. Este adaptador recibirá las peticiones en el protocolo acordado con el sistema origen, y las enviará a una queue donde el adaptador N-1 estará suscrito. De este modo durante el cierre y envío del fichero, se desactivará temporalmente el suscriptor de la queue y las peticiones que lleguen en ese momento se encolarán. Pudiendo ser añadidas a un nuevo fichero una vez se haya enviado el anterior. Dicho esto el flujo completo será de la siguiente forma:

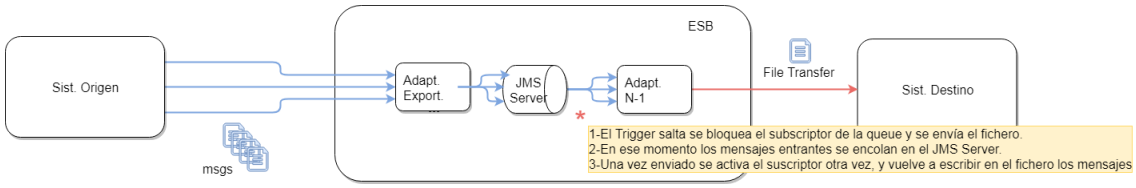


Imagen 47 – Flujo completo adaptador N-1

3.2.3.1 Estructura del adaptador

Para poder adaptar los desarrollos que se vayan a realizar sobre el FW del ESB, se tendrá que seguir una serie de pautas necesarias para el acoplamiento entre estos.
Estas pautas son necesarias debido a que el propio FW cuenta con muchas configuraciones dinámicas (invocaciones a procesos de negocio) las cuales dependen de la estructura de las carpetas del proyecto entre otras cosas.

Por esta razón, los adaptadores de importación que se desarrollen bajo el FW del ESB han de contar con la siguiente estructura de carpetas.

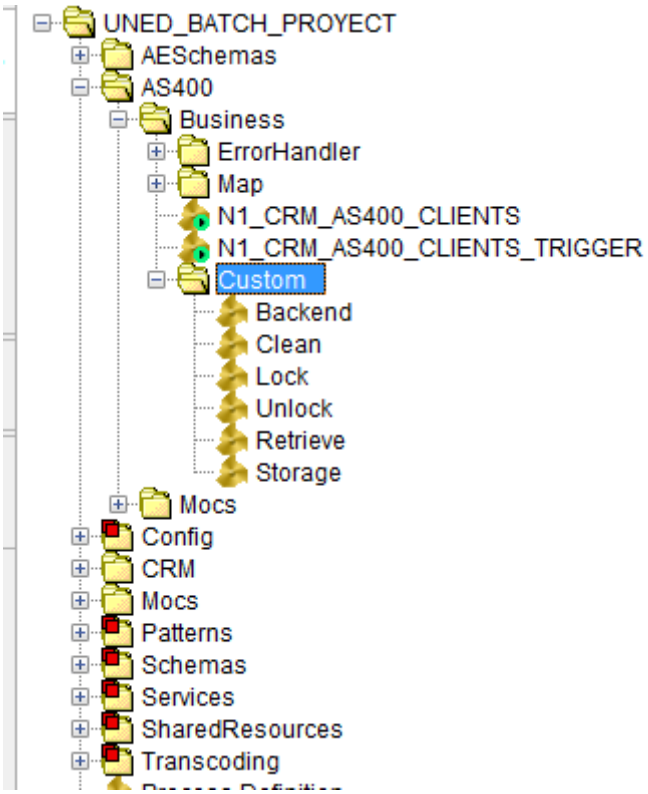


Imagen 48 - Estructura de carpetas de un proyecto de adaptador batch N-1

Como se observa en la imagen 49 la estructura del proyecto del adaptador de N-1 se forma por las siguientes carpetas:

Nombre Carpeta	Nivel	Descripción
NOMBRE_BATCH_PROJECT	0	Carpeta root del proyecto, en ella está contenidos todos los procesos y funcionalidades.
Config	1	Carpeta destinada a los ficheros de configuración.
NOMBRE_PROYECTO	1	Carpeta con el nombre del proyecto, dentro de esta carpeta se encuentra todo el código del adaptador.
Business	2	Carpeta que contiene el código relativo al requisito del negocio.

AI_NOMBRE_ADAPTADOR	3	Carpeta que contiene todo lo relativo al adaptador.
ErrorHandler	4	Carpeta donde se encuentran los procesos encargados de la gestión de errores.
Map	4	Carpeta donde se encuentran los procesos encargados de realizar los mapeos.
Custom	4	Carpeta donde se encuentran los procesos de negocio propios para tratar el fichero.
Schemas	4	Carpeta donde se almacenan los interfaces del proceso (XSD, WSDL, WADL, SWAGGER…)
SharedResources	2	Carpeta donde se encuentran las conexiones a base de datos, http, jms… así como cualquier elemento de uso común dentro del proyecto.
Deploy	1	Carpeta donde se encuentran los artefactos instalables.

Tabla 5 - Estructura del proyecto

3.2.3.2 Nomenclatura

Al igual que la estructura del proyecto, se necesita seguir una nomenclatura para que el framework funcione correctamente. La nomenclatura a seguir es la siguiente:

Tipo de elemento	Ejemplo Nomenclatura	Descripción
Proceso Starter	N1_CRM_AS400_CLIENTS	N1: tipo de adaptador. CRM: Sistema origen. CLIENTS: Entidad AS400: Sistema Destino.
Proceso Map	Map	
Proceso BackEnd	Backend	
Proceso Clean	Clean	
Proceso Lock	Lock	
Proceso Unlock	Unlock	
Proceso Storage	Storage	

Proceso Retrieve	Retrieve	

Tabla 6 – Nomenclatura de procesos

3.2.3.3 Tipos de procesos

Dentro de cada adaptador, se podrán diferenciar claramente dos tipos de procesos que serán los que hagan posible su ejecución:

- Procesos de negocio
- Procesos de framework

Procesos de negocio

Este tipo de procesos son los destinados a albergar la lógica de negocio del adaptador. Esta lógica está dividida en las siguientes funcionalidades:

ErrorHandler: Proceso destinado a gestionar el tratamiento de error. Al tratarse de un proceso asíncrono el cual no tiene respuesta el tratamiento del error residirá en trazar el error.

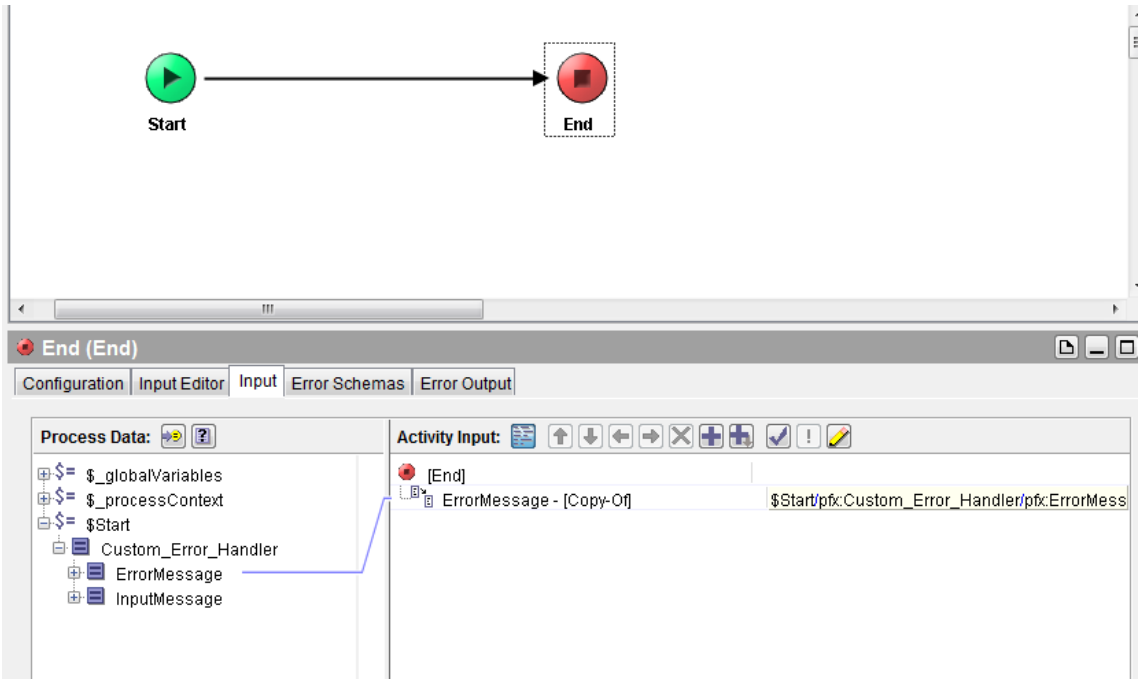


Imagen 49 – Proceso de tratamiento de error asíncrono.

Map: Proceso destinado a realizar las transformaciones. El tipo de mapeo que se llevará a cabo en el adaptador de N-1 es el destinado a realizar las transformaciones para el envío de la información al Backend: Transformaciones dedicadas al fichero.

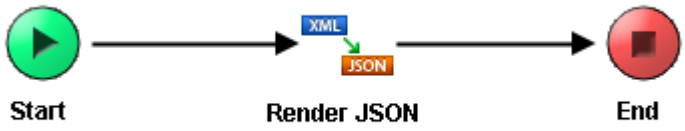


Imagen 50 – Ejemplo de transformación de XML a JSON.

Backend: Proceso destinado a realizar el envío de fichero a la ruta especificada.

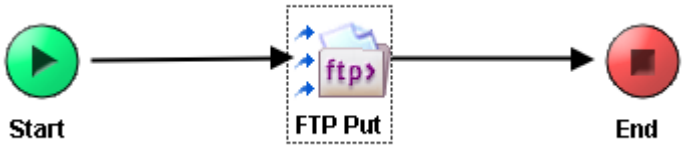


Imagen 53 – Ejemplo de envío de fichero.

Lock: Proceso destinado a realizar el bloqueo del proceso. Esto se realiza durante el envío del fichero. Para que las peticiones entrantes durante ese momento se queden encoladas en el servidor JMS y se escriban una vez se haya enviado, en un nuevo fichero.

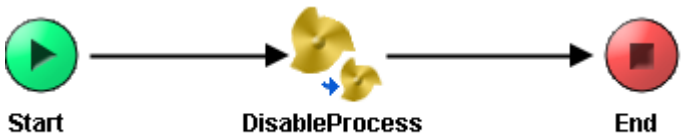


Imagen 54 – Ejemplo de bloqueo del proceso.

Unlock: Proceso destinado a realizar el desbloqueo del proceso. Una vez realizado el envío del fichero a la ruta o localización especificada. Se desbloquea el proceso para que vuelva a consumir los mensajes de la queue.

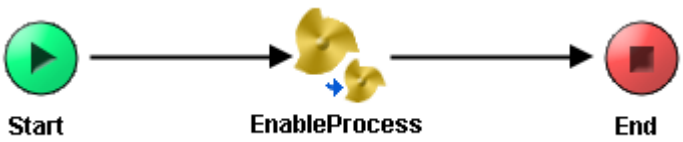


Imagen 55 – Ejemplo de desbloqueo del proceso.

Storage: Proceso donde se genera el fichero temporal antes de ser enviado.

Imagen 51 – Ejemplo de creación del fichero temporal.

Clean: Proceso destinado a eliminar el fichero temporal una vez se ha enviado.

Imagen 52 – Ejemplo de eliminación del fichero temporal.

Procesos de framework

Este tipo de procesos son completamente estándar. No es necesario modificarlos para el desarrollo de ningún adaptador. Son una serie de plantillas cerradas que de forma dinámica irán trazando los mensajes e invocando a los procesos de negocio. Este tipo de procesos son lo que se denomina "Framework" y facilitan la labor del desarrollador abstrayéndolo únicamente a la lógica de negocio. Además gracias a que no se pueden modificar dotan a los adaptadores de una homogeneidad imprescindible a la hora de llevar un soporte adecuado de la herramienta.
Los procesos del framework para el adaptador que tratamos en este punto son los siguientes:

Batch_N_1_Pattern: Proceso principal del adaptador, actúa a modo de orquestado dirigiendo el flujo por el resto de procesos que conforman el framework para el adaptador que estamos tratando.

PreActions: Proceso que contiene una serie de subprocesos que llevan a cabo las tareas necesarias en el inicio del flujo.

Batch_storage: Proceso que se encarga de ir formando el fichero para su posterior envío.

3.2.3.4 Diseño por pasos

Como se ha comentado anteriormente, este adaptador tiene dos flujos diferenciados dependiendo de si se pretende seguir agrupando mensajes en un fichero o se pretende cerrar y enviar el fichero. A continuación se detallarán los dos flujos:

Flujo inserción Mensaje en Fichero

A continuación se mostrará el diseño paso a paso de dicho adaptador en modo escritura de mensajes en fichero.

PASO 1 Recepción del mensaje

El primer paso de todos es la recepción del mensaje, este se recepcionará bajo el protocolo de mensajería interna JMS

- JMS

Para ello se definirá un proceso de entrada que será el encargado de recepcionarlo.

Se definirá un proceso que se encargará de levantar un receiver JMS para escuchar en una queue definida la petición enviada desde el Adaptador de exportación.
Una vez recibida la petición la enviará a un proceso del framework que se encargará de orquestar las llamadas dinámicas al resto de procesos de negocio.
Al ser un adaptador de flujo asíncrono no es necesario mapear la response.

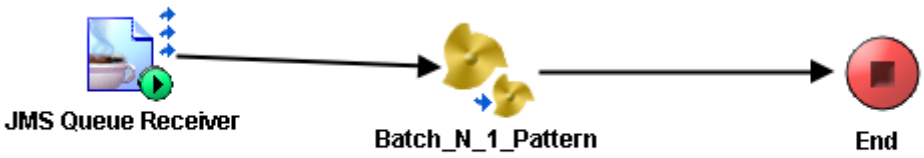


Imagen 53 – Proceso starter JMS

En esta imagen se describe el proceso de Starter:

1. JMS Queue Receiver: Tarea que dada un servidor JMS y una queue, levanta un lisener JMS para recepcionar peticiones.
2. Batch_N_1_Pattern: Una vez recabada la petición se manda de forma interna a un proceso del framework, que comienza a realizar una serie de acciones e invocaciones dinámicas.
3. End: Una vez el proceso del framework realiza todas las acciones el proceso termina su ejecución.

PASO 2 Gestión de la petición por el Framework.

En este paso el proceso genérico del framework (Batch_N_1_Pattern) realiza una serie de acciones denominadas preacciones y postacciones, para procesar la petición. Dentro de estas acciones se encuentran las llamadas dinámicas a los procesos de negocio. Este proceso es un proceso que forma parte del Framework, es decir no es modificable por el desarrollador, con la información proporcionada en la entrada y la estructura/nomenclatura definidas es capaz de invocar y trazar todos los procesos que contienen la lógica de negocio.

El flujo de este proceso se detalla a continuación.

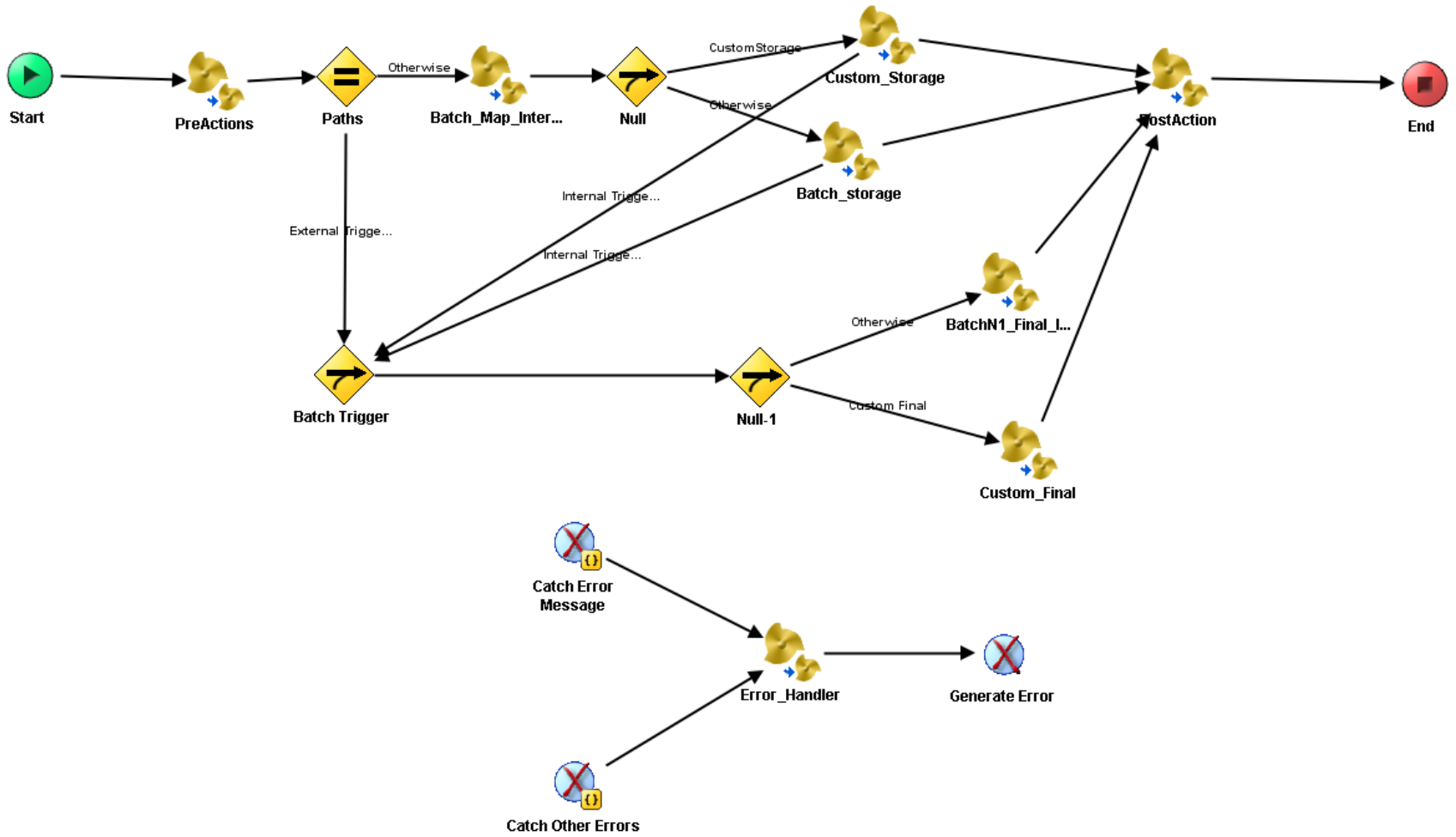


Imagen 54 – Flujo del componente del FW BATCH N-1

En la imagen anterior se describe el proceso de Batch_N_1_Pattern.

Estos son los pasos que realiza:

1. Al proceso le llega la petición del Starter con la siguiente información:

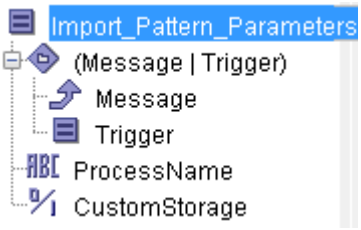


Imagen 55 – Esquema de entrada Batch_N_1_Pattern

En ella se detalla la siguiente información:

Campo	Nivel	Tipo	Descripción
Import_Pattern_Parameters	0	Complex Type	Campo padre que alberga la estructura
Message	1	Complex Type	Campo choice donde se encuentra el mensaje introducido
Trigger	1	Any type	Campo que indica que en vez de insertar un nuevo mensaje, hay que cerrar el fichero y enviarlo.
ProcessName	1	String	Nombre del proceso
CustomStorage	1	Boolean	Flag que indica si se quiere realizar un tratamiento customizado en el cierre y envío del fichero o se quiere usar el definido por defecto en el FW.

Tabla 7 – Tabla de campos de entrada en el patrón batch N-1

2. A continuación se ejecutan las pre acciones que están formadas por las siguientes acciones:

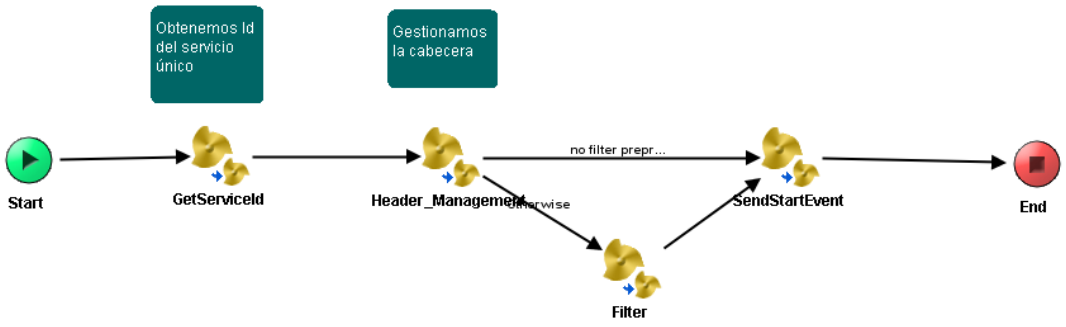


Imagen 56 – Flujo con la ejecución de las distintas pre acciones.

a. La primera es la asignación de un identificador único de la ejecución, mediante el algoritmo de generación de UUID.
b. Después se genera la cabecera de arquitectura en el caso de que no esté ya generada. Esta cabecera estará generada siempre y cuando el

adaptador que se está ejecutando no es el primero del flujo, ya que se va propagando. Esta cabecera consta con los siguientes campos:

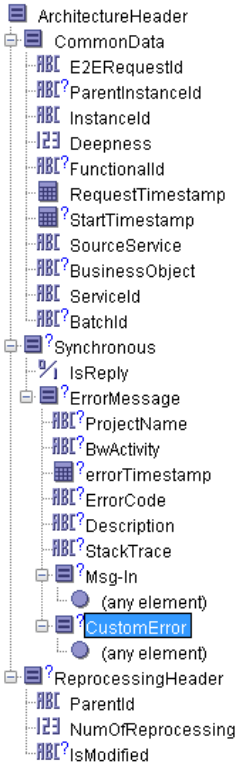


Imagen 57 – Parámetros de la cabecera de arquitectura o gestión.

c. Por último se gestiona la creación de eventos, dependiendo del nivel de trazabilidad establecido. Esta gestión consistirá en el envío del evento a un 3er sistema y la escritura en un log de ejecución.

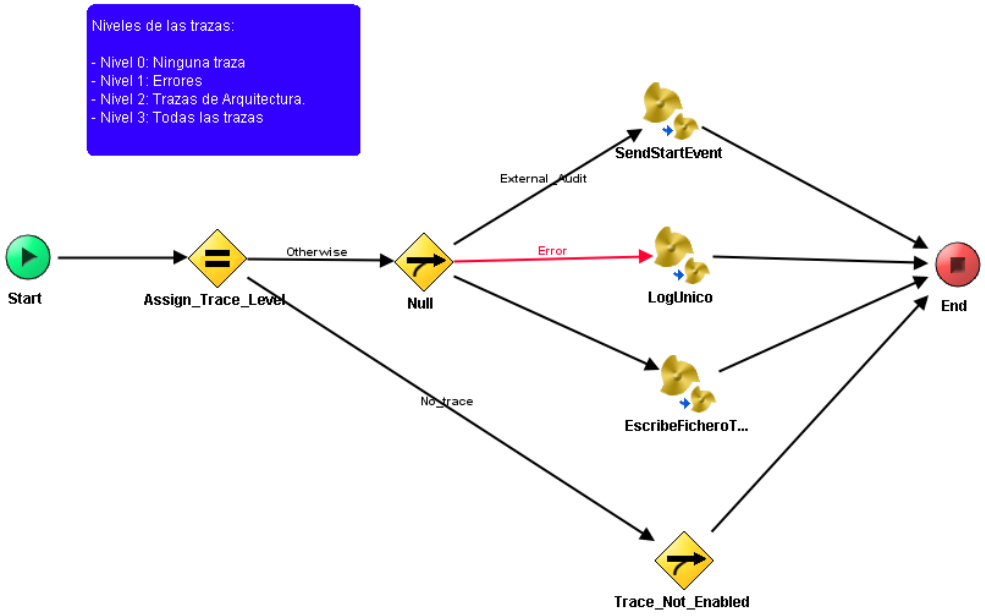
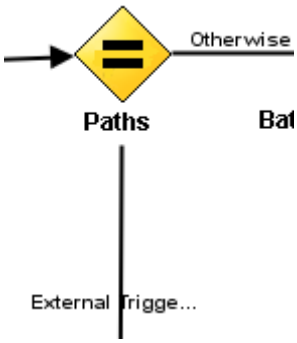


Imagen 58 – Flujo de gestión de eventos.

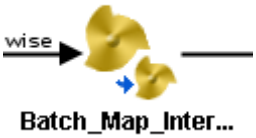
3. El siguiente paso es una comprobación de si la petición se ha realizado para insertar un nuevo mensaje en el fichero, o por el contrario, se pretende activar el

trigger para cerrar el fichero y enviarlo. Esta bifurcación del flujo se realiza comprobando que elemento ha llegado al proceso (Mensaje/Trigger).

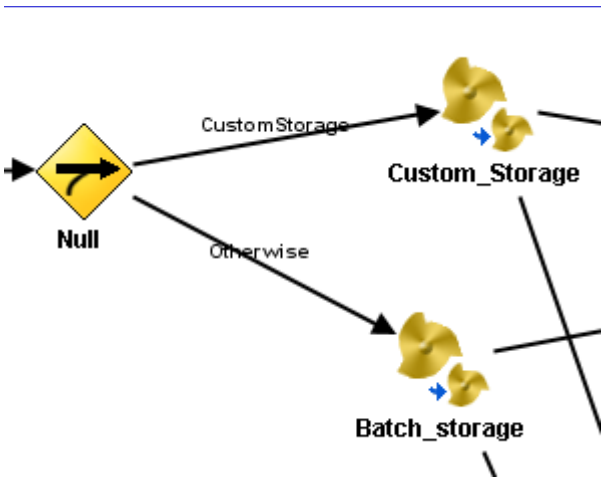


Como se está exponiendo primero el flujo de insertar un mensaje en el fichero se continuará por esta rama.

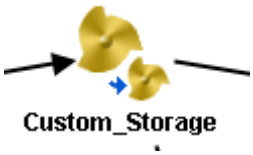
4. El siguiente paso es la invocación dinámica al proceso de negocio de mapeos. En este paso se invocará con las rutas generadas dinámicamente en el punto anterior al proceso que contiene las transformaciones del mensaje de entrada según los requisitos del negocio.



5. A continuación se comprobará si se ha decidido realizar un almacenaje customizado o se ha elegido utilizar el que ofrece el framework por defecto. En este punto se supondrá que se decide usar el customizado para poder observar el flujo más completo:



6. A continuación se realiza una llamada dinámica al proceso de almacenamiento en fichero. Que introducirá el mensaje en un fichero temporal:



7. Por último se llevan a cabo las postacciones del flujo del adaptador de exportación que contiene las siguientes funcionalidades:

Imagen 59 – Flujo con las post acciones

a. La primera y última acción de las postacciones del adaptador de importación es el envío del evento de fin de ejecución con el resultado obtenido y la escritura de las trazas correspondientes.

Este proceso se podrá repetir tantas veces como mensajes lleguen durante el periodo de tiempo que se defina la ejecución del trigger para cerrar y enviar el fichero. Cada mensaje seguirá este flujo quedando guardado en el fichero.

Una vez pasado un tiempo predefinido, se ejecutará el proceso trigger. Este flujo se detalla a continuación.

Flujo trigger cierre y envío de fichero

A continuación se mostrará el diseño paso a paso de dicho adaptador en modo trigger para el cierre y posterior envío del fichero.

PASO 1 Ejecución programada del evento de cierre y envío del fichero

El primer paso de todos es la ejecución programada del envío del evento de cierre y envío de fichero.

Mediante un evento de tipo Timer se iniciará la ejecución del flujo

Se definirá un proceso con un evento de tipo Timer que iniciará el flujo de cierre y envío del fichero. Al ser un adaptador de flujo asíncrono no es necesario mapear la response.

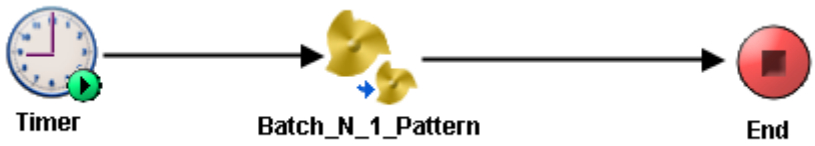


Imagen 60 – Proceso starter Timer

En esta imagen se describe el proceso de Starter:

1. El timer enviará un evento cada X tiempo.
2. Este evento iniciará el flujo del framework para realizar todas las acciones correspondientes para el cierre y posterior envío del fichero.

PASO 2 Gestión de la petición por el Framework.

En este paso el proceso genérico del framework (Batch_N_1_Pattern) realiza una serie de acciones denominadas preacciones y postacciones, para procesar la petición.
Dentro de estas acciones se encuentran las llamadas dinámicas a los procesos de negocio. Este proceso forma parte del Framework, es decir no es modificable por el desarrollador, con la información proporcionada en la entrada y la estructura/nomenclatura definidas es capaz de invocar y trazar todos los procesos que contienen la lógica de negocio.

El flujo de este proceso se detalla a continuación.

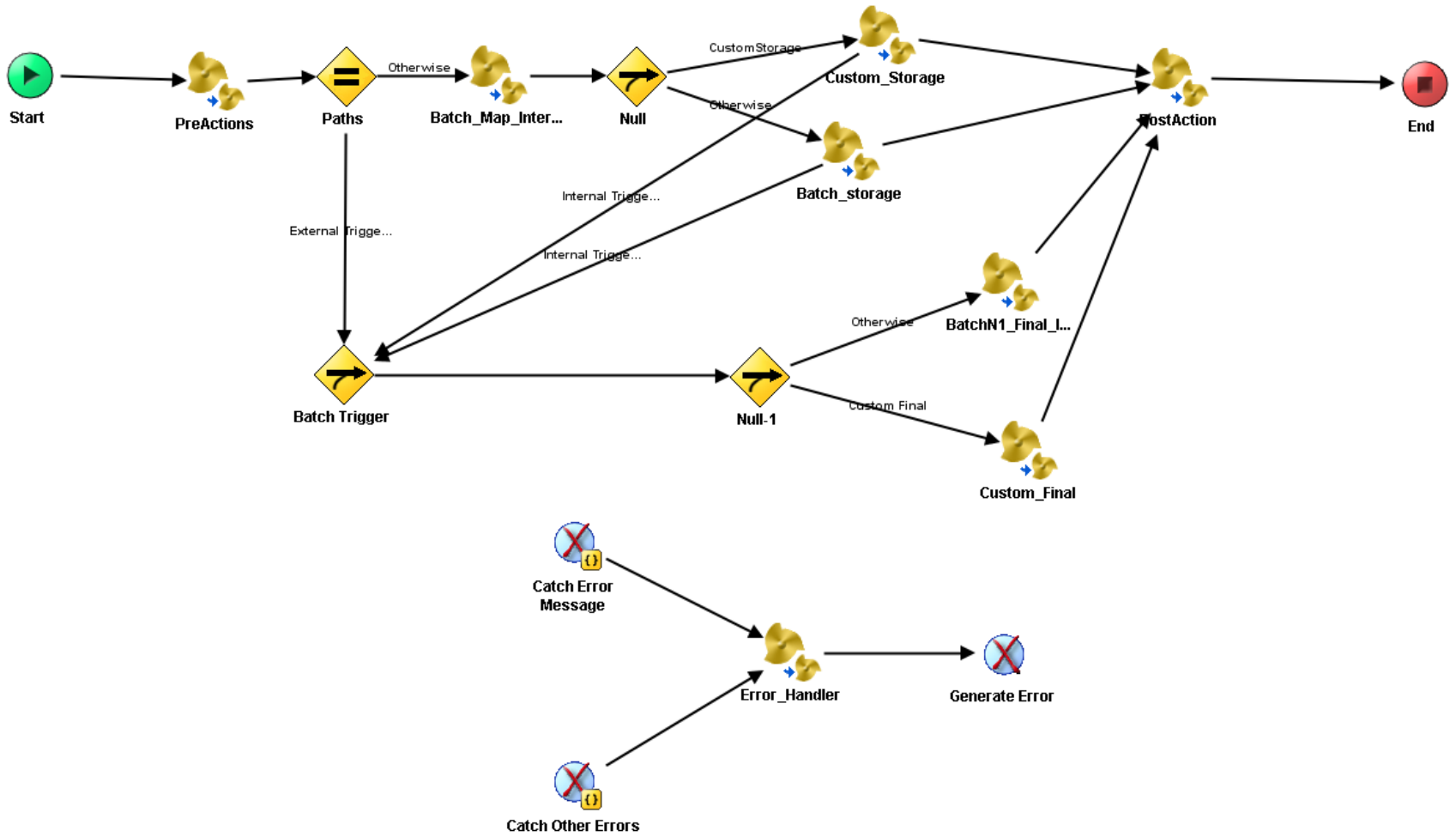


Imagen 61 – Flujo del componente del FW BATCH N-1

En la imagen anterior se describe el proceso de Batch_N_1_Pattern.

Estos son los pasos que realiza:

1. Al proceso le llega la petición del Starter con la siguiente información:

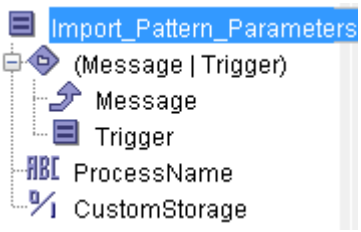


Imagen 62 – Esquema de entrada Batch_N_1_Pattern

En ella se detalla la siguiente información:

Campo	Nivel	Tipo	Descripción
Import_Pattern_Parameters	0	Complex Type	Campo padre que alberga la estructura
Message	1	Complex Type	Campo choice donde se encuentra el mensaje introducido
Trigger	1	Any type	Campo que indica que en vez de insertar un nuevo mensaje, hay que cerrar el fichero y enviarlo.
ProcessName	1	String	Nombre del proceso
CustomStorage	1	Boolean	Flag que indica si se quiere realizar un tratamiento customizado en el cierre y envío del fichero o se quiere usar el definido por defecto en el FW.

Tabla 8 – Tabla de campos de entrada en el patrón batch N-1

2. A continuación se ejecutan las pre acciones que están formadas por las siguientes acciones:

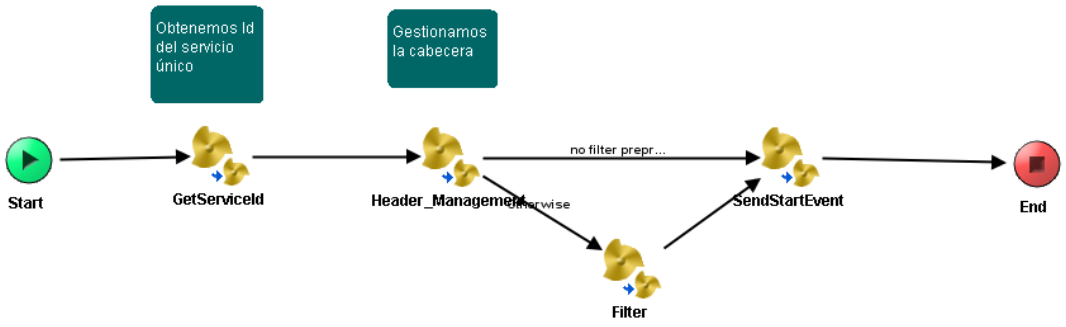


Imagen 63 – Flujo con la ejecución de las distintas pre acciones.

   a. La primera es la asignación de un identificador único de la ejecución, mediante el algoritmo de generación de UUID.
   b. Después se genera la cabecera de arquitectura en el caso de que no esté ya generada. Esta cabecera estará generada siempre y cuando el

adaptador que se está ejecutando no es el primero del flujo, ya que se va propagando. Esta cabecera consta con los siguientes campos:

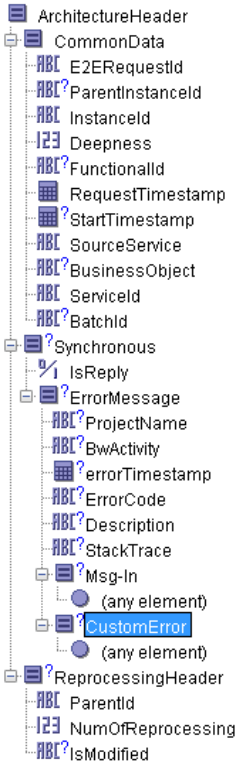


Imagen 64 – Parámetros de la cabecera de arquitectura o gestión.

c. Por último se gestiona la creación de eventos, dependiendo del nivel de trazabilidad establecido. Esta gestión consistirá en el envío del evento a un 3er sistema y la escritura en un log de ejecución.

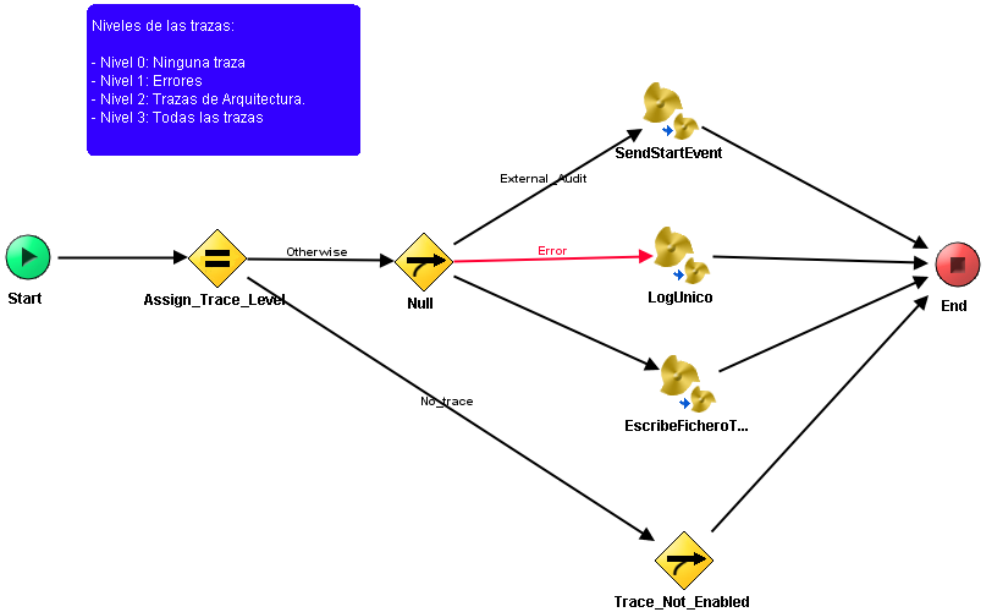
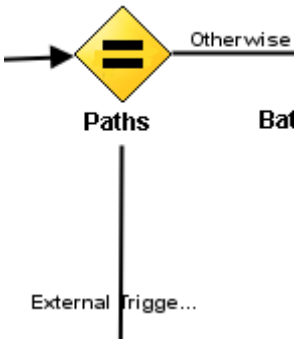


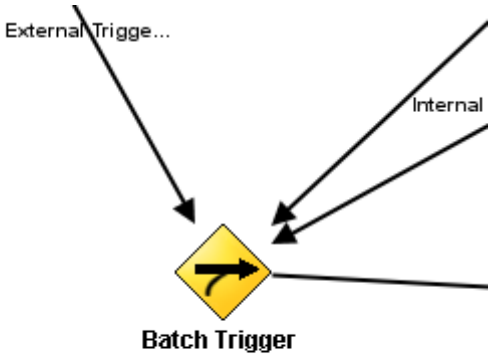
Imagen 65 – Flujo de gestión de eventos.

3. El siguiente paso es una comprobación de si la petición se ha realizado para insertar un nuevo mensaje en el fichero, o por el contrario, se pretende activar el

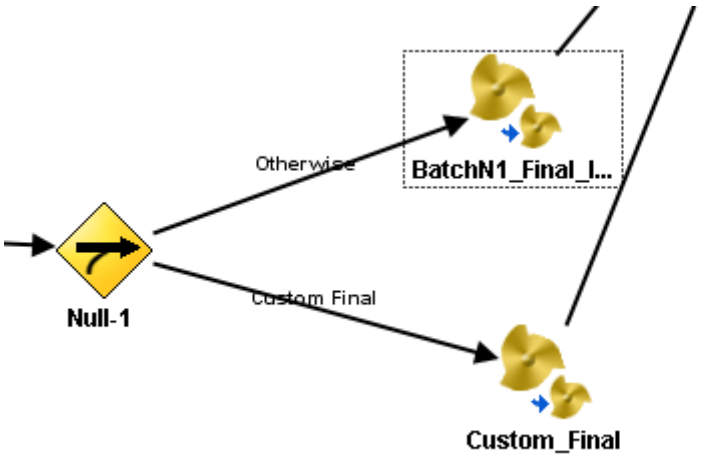
trigger para cerrar el fichero y enviarlo. Esta bifurcación del flujo se realiza comprobando que elemento ha llegado al proceso (Mensaje/Trigger).



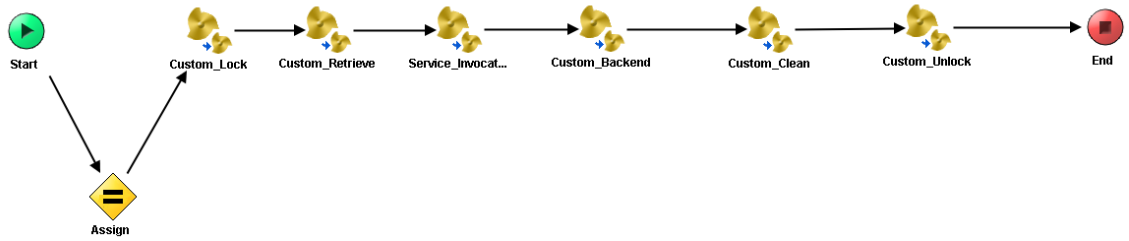
Como se está exponiendo el flujo de cerrar y enviar el fichero, se continuará por esta rama.



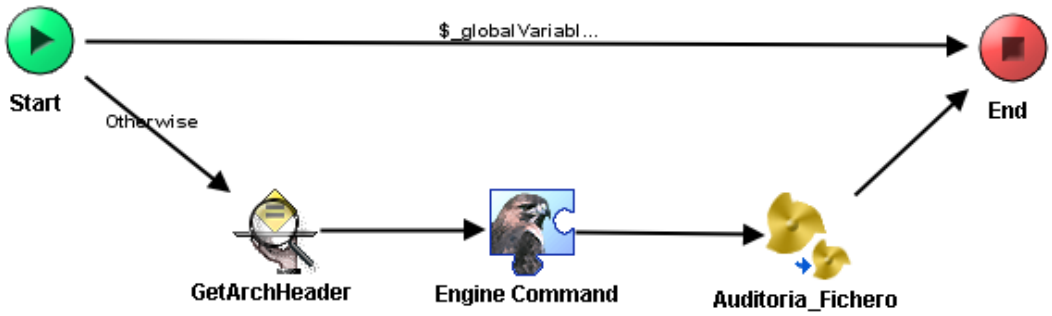
4. El siguiente paso decide si se tiene que ir por el proceso customizado de cierre y envío o por el contrario se opta por el proceso predefinido en el framework. Esta decisión la toma dependiendo del valor que el desarrollador haya dado al flag custom que se comenta en la tabla de campos del principio de este punto.
   En este caso, se optará por la rama custom ya que es más completa:



5. A continuación se ejecutará el proceso Custom_Final, que orquestará todos los procesos customizados a nivel de negocio para llevar a cabo el envío del fichero:

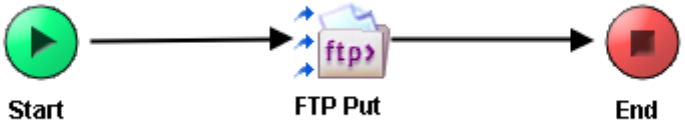


a. El primero de estos pasos es el Custom_Lock, que tiene como objetivo bloquear el proceso de suscripción a la queue de mensajes:



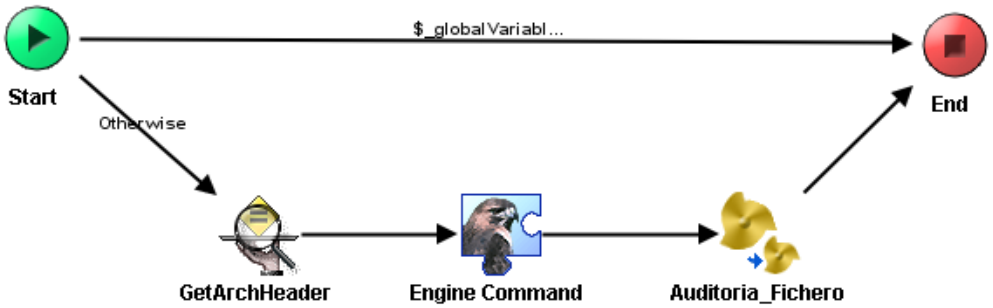
Este proceso desactiva el suscriptor durante el tiempo que dure el procedimiento de envío del fichero, para que una vez terminado se vuelva a activar.

b. El siguiente proceso de negocio customizado sería el envío del fichero via FTP mediante el Custom_Backend:



c. El siguiente paso sería la ejecución del proceso customizado de limpieza. Para esto se utilizará el proceso de Custom_Clean que eliminará el fichero temporal creado para almacenar los mensajes durante el periodo de recepción.

d. Por último para cerrar con este proceso Custom, se desbloqueará de nuevo el suscriptor de la queue para que vuelva a almacenar mensajes en un nuevo fichero.



6. Por último se llevan a cabo las postacciones del flujo del adaptador de exportación que contiene las siguientes funcionalidades:

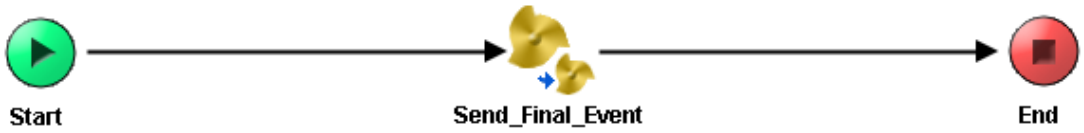


Imagen 66 – Flujo con las post acciones

a. La primera y última acción de las post-acciones del adaptador de importación es el envío del evento de fin de ejecución con el resultado obtenido y la escritura de las trazas correspondientes.

3.2.4 Adaptador 1-N

Durante este punto se definirá el adaptador 1-N que forma parte del patrón batch del FW del ESB.

Este adaptador está definido con la intención de recoger una única petición en forma de fichero o mediante mensaje jms, y realizar un Split sobre el mismo para transformarlo a N peticiones a un sistema BackEnd.

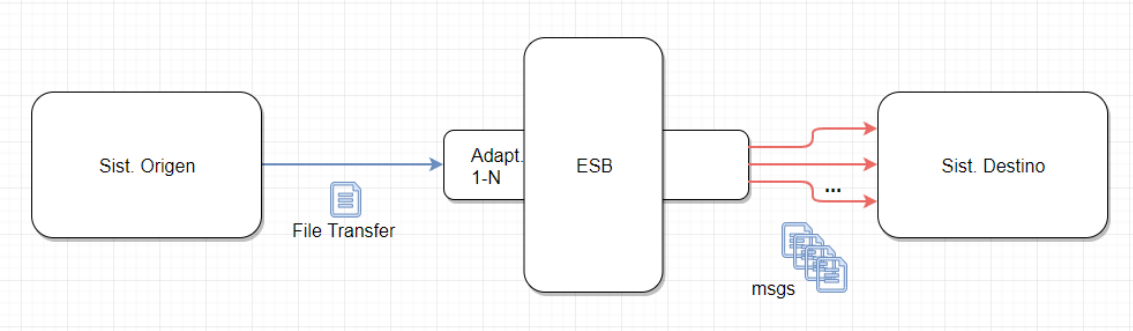


Imagen 67 - Flujo alto nivel adaptador 1-N

Debido a que la idea original del adaptador es generar N peticiones independientes de un único mensaje. Este adaptador se enriquece con un adaptador de importación en la 2ª parte del flujo de modo el adaptador 1-N enviará estas peticiones vía JMS al adaptador de importación y este enviará dichas peticiones en ejecuciones independientes. Pudiendo controlar los posibles errores en cada una de ellas. La arquitectura final de este adaptador se muestra a continuación:

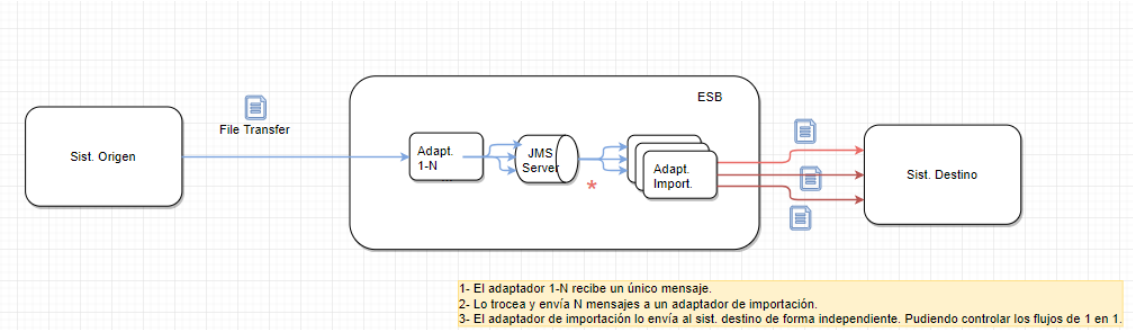


Imagen 68 – Flujo completo adaptador 1-N

3.2.4.5 Estructura del adaptador

Para poder adaptar los desarrollos que se vayan a realizar sobre el FW del ESB, se tendrá que seguir una serie de pautas necesarias para el acoplamiento entre estos.
Estas pautas son necesarias debido a que el propio FW cuenta con muchas configuraciones dinámicas (invocaciones a procesos de negocio) las cuales dependen de la estructura de las carpetas del proyecto entre otras cosas.

Por esta razón, los adaptadores de importación que se desarrollen bajo el FW del ESB han de contar con la siguiente estructura de carpetas.

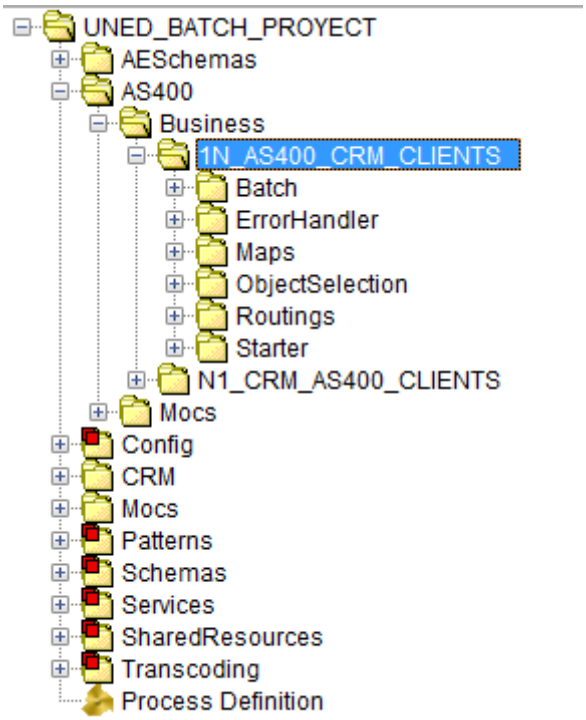


Imagen 69 - Estructura de carpetas de un proyecto de adaptador batch 1-N

Como se observa en la imagen 72 la estructura del proyecto del adaptador de N-1 se forma por las siguientes carpetas:

Nombre Carpeta	Nivel	Descripción
NOMBRE_BATCH_PROJECT	0	Carpeta root del proyecto, en ella está contenidos todos los procesos y funcionalidades.
Config	1	Carpeta destinada a los ficheros de configuración.
NOMBRE_PROYECTO	1	Carpeta con el nombre del proyecto, dentro de esta carpeta se encuentra todo el código del adaptador.
Business	2	Carpeta que contiene el código relativo al requisito del negocio.
AI_NOMBRE_ADAPTADOR	3	Carpeta que contiene todo lo relativo al

		adaptador.
ErrorHandler	4	Carpeta donde se encuentran los procesos encargados de la gestión de errores.
Map	4	Carpeta donde se encuentran los procesos encargados de realizar los mapeos.
Routings	4	Carpeta donde se encuentran los procesos de negocio propios para tratar el fichero.
ObjectSelection	4	Carpeta donde se encuentran los procesos encargados de seleccionar la entidad sobre la que se ejecutará el flujo
Batch	4	Carpeta donde se encuentran los procesos de negocio relacionados con el tratamiento del fichero de entrada
Schemas	4	Carpeta donde se almacenan los interfaces del proceso (XSD, WSDL, WADL, SWAGGER…)
SharedResources	2	Carpeta donde se encuentran las conexiones a base de datos, http, jms… así como cualquier elemento de uso común dentro del proyecto.
Deploy	1	Carpeta donde se encuentran los artefactos instalables.

Tabla 9 - Estructura del proyecto

3.2.4.6 Nomenclatura

Al igual que la estructura del proyecto, se necesita seguir una nomenclatura para que el framework funcione correctamente. La nomenclatura a seguir es la siguiente:

Tipo de elemento	Ejemplo Nomenclatura	Descripción
Proceso Starter	1N_AS400_CRM_CLIENTS	1N: tipo de adaptador. AS400: Sistema Origen. CRM: Sistema Destino. CLIENTS: Entidad
Proceso Map	Map	
Proceso	Routing	

Routing		
Proceso map	Map	
Proceso ObjectSelection	Objectselection	
Proceso Split	Splitter	

Tabla 10 – Nomenclatura de procesos

3.2.4.7 Tipos de procesos

Dentro de cada adaptador, se podrán diferenciar claramente dos tipos de procesos que serán los que hagan posible su ejecución:

- Procesos de negocio
- Procesos de framework

Procesos de negocio

Este tipo de procesos son los destinados a albergar la lógica de negocio del adaptador. Esta lógica está dividida en las siguientes funcionalidades:

ErrorHandler: Proceso destinado a gestionar el tratamiento de error. Al tratarse de un proceso asíncrono el cual no tiene respuesta el tratamiento del error residirá en trazar el error.

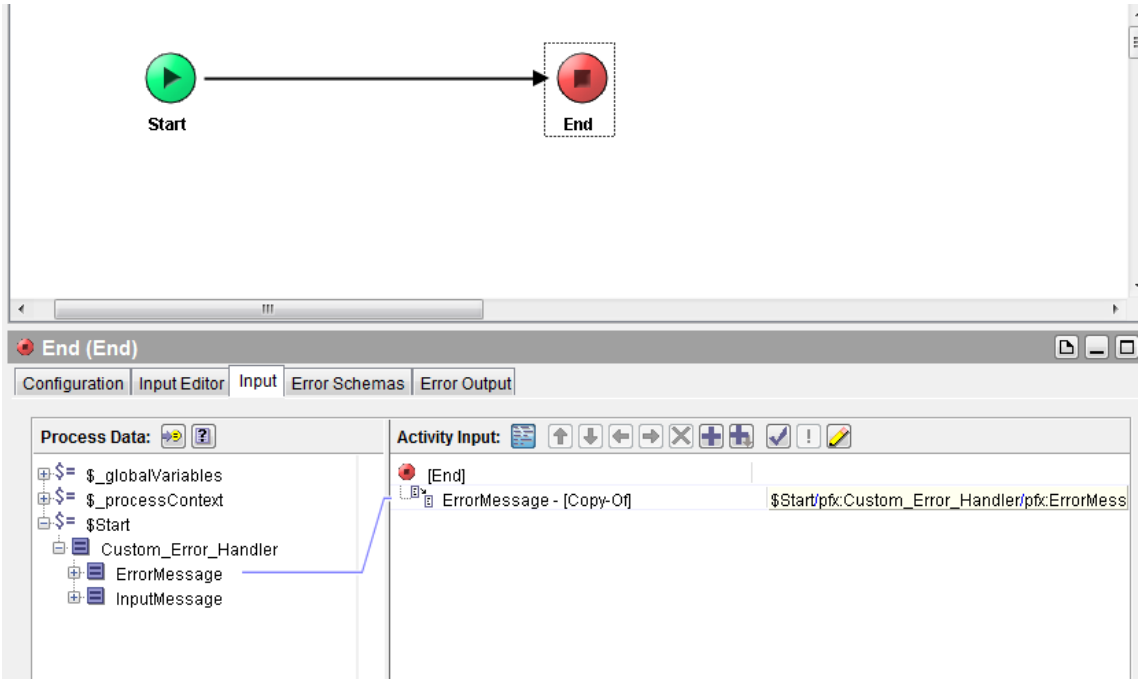


Imagen 70 – Proceso de tratamiento de error asíncrono.

Map: Proceso destinado a realizar las transformaciones. El tipo de mapeo que se llevará a cabo en el adaptador de 1-N es el destinado a realizar las transformaciones del mensaje original para el envío de la información al Backend: Transformaciones dedicadas al fichero.

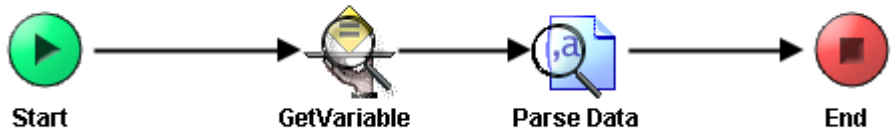


Imagen 71 – Ejemplo de transformación de parseando la información contenida en un fichero

Splitter: Proceso destinado a realizar el troceado del mensaje original en N Mensajes.

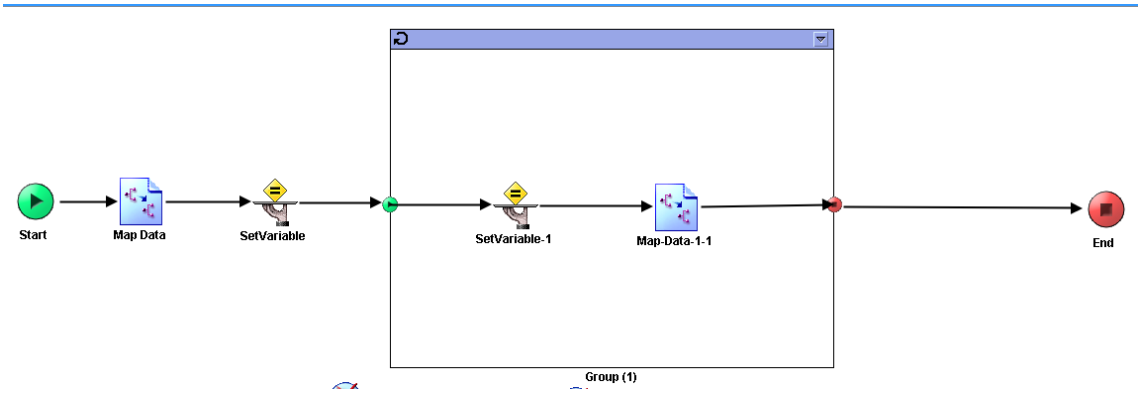


Imagen 53 – Ejemplo de split de fichero.

Routing: Proceso dedicado a realizar la lógica de enrutado dependiendo de la petición recibida.

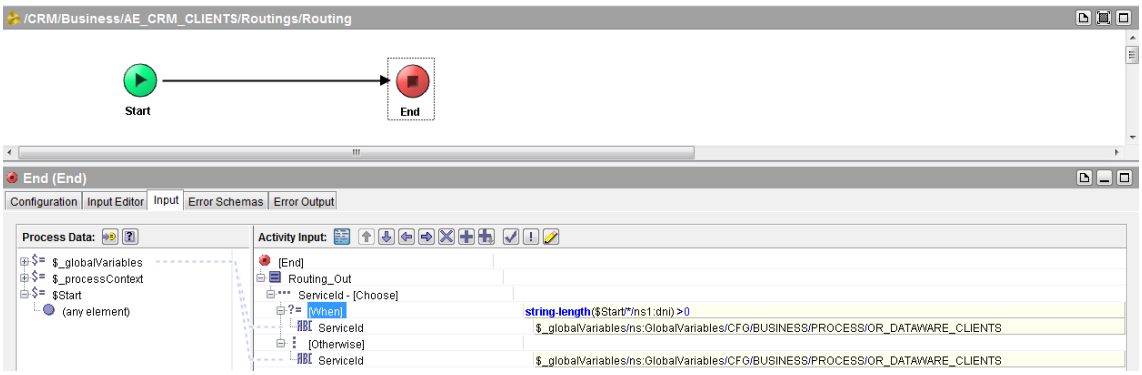


Imagen 72 – Ejemplo de routing mediante lógica en XPath

ObjectSelection: Proceso dedicado a realizar la lógica de la selección de la entidad sobre la que va a trabajar el flujo. Esta decisión se tomará de forma dinámica dependiendo de la petición entrante.

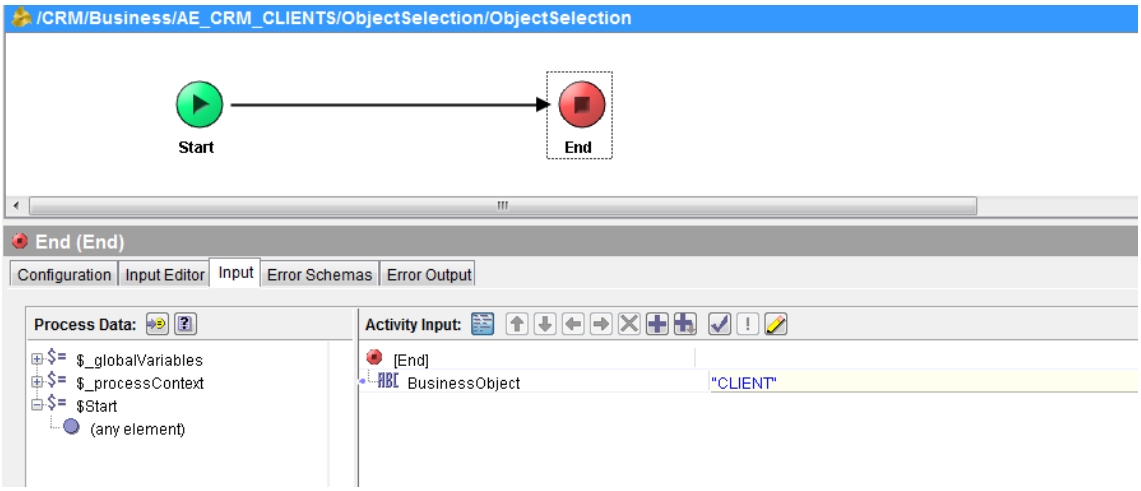


Imagen 73 – Ejemplo de selección de entidades mediante XPath

ErrorHandler: Proceso destinado a gestionar el tratamiento de error. Al tratarse de un proceso asíncrono el cual no tiene respuesta el tratamiento del error residirá en trazar el error.

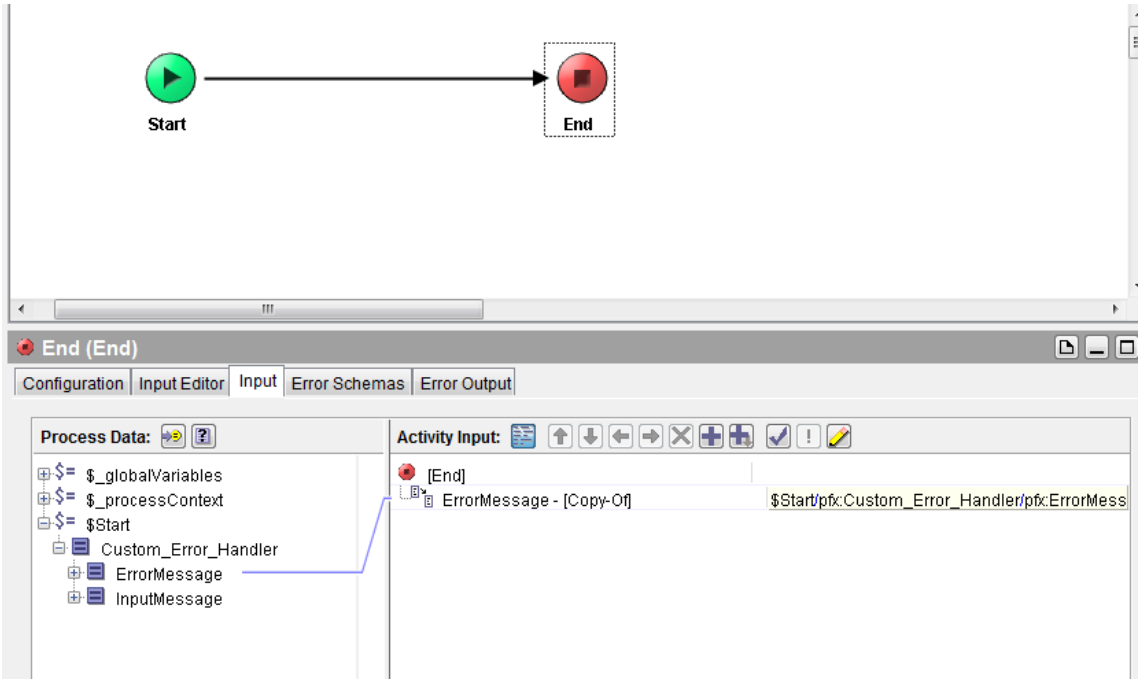


Imagen 74 – Proceso de tratamiento de error asíncrono.

Procesos de framework

Este tipo de procesos son completamente estándar. No es necesario modificarlos para el desarrollo de ningún adaptador. Son una serie de plantillas cerradas que de forma dinámica irán trazando los mensajes e invocando a los procesos de negocio. Este tipo de procesos son lo que se denomina "Framework" y facilitan la labor del desarrollador abstrayéndolo únicamente a la lógica de negocio. Además gracias a que no se pueden modificar dotan a los adaptadores de una homogeneidad imprescindible a la hora de llevar un soporte adecuado de la herramienta.
Los procesos del framework para el adaptador que tratamos en este punto son los siguientes:

Batch_1_N_Pattern: Proceso principal del adaptador, actúa a modo de orquestado dirigiendo el flujo por el resto de procesos que conforman el framework para el adaptador que estamos tratando.

PreActions: Proceso que contiene una serie de subprocesos que llevan a cabo las tareas necesarias en el inicio del flujo.

PostActions: Proceso que contiene una serie de subprocesos que llevan a cabo las tareas necesarias en el fin del flujo.

3.2.4.8 Diseño por pasos

A continuación se definirá el diseño del flujo que seguirá este adaptador paso a paso. De modo que se pueda observar que pasos se llevan a cabo y el porqué de cada uno de ellos.

PASO 1 Recepción del mensaje

El primer paso de todos es la recepción del fichero. Este proceso se ejecutará con un evento lanzado por un demonio que está comprobando en una ruta determinada si han entrado ficheros nuevos. Como resultado de esto el proceso se iniciará con el contenido de dicho fichero. Por lo tanto se puede decir que el protocolo de envío en el inicio del flujo es:

- FTP

Para ello se definirá un proceso de entrada que será el encargado de recepcionarlo.

Se definirá un proceso que se encargará de levantar un demonio que está comprobando en una ruta determinada si han entrado ficheros nuevos. Cuando entra un fichero nuevo, genera un evento con el contenido del fichero.

Una vez recibida la petición la enviará a un proceso del framework que se encargará de orquestar las llamadas dinámicas al resto de procesos de negocio.
Al ser un adaptador de flujo asíncrono no es necesario mapear la response.

Imagen 75 – Proceso starter File Poller

En esta imagen se describe el proceso de Starter:

1. File Poller: Demonio que está comprobando en una ruta determinada si han entrado ficheros nuevos. Cuando entra un fichero nuevo, genera un evento con el contenido del fichero.
2. Batch_1-N_Pattern: Una vez recabada la petición se manda de forma interna a un proceso del framework, que comienza a realizar una serie de acciones e invocaciones dinámicas.
3. End: Una vez el proceso del framework realiza todas las acciones el proceso termina su ejecución.

PASO 2 Gestión de la petición por el Framework.

En este paso el proceso genérico del framework (Batch_1-N_Pattern) realiza una serie de acciones denominadas preacciones y postacciones, para procesar la petición.
Dentro de estas acciones se encuentran las llamadas dinámicas a los procesos de negocio. Este proceso es un proceso que forma parte del Framework, es decir no es modificable por el desarrollador, con la información proporcionada en la entrada y la estructura/nomenclatura definidas es capaz de invocar y trazar todos los procesos que contienen la lógica de negocio.

El flujo de este proceso se detalla a continuación.

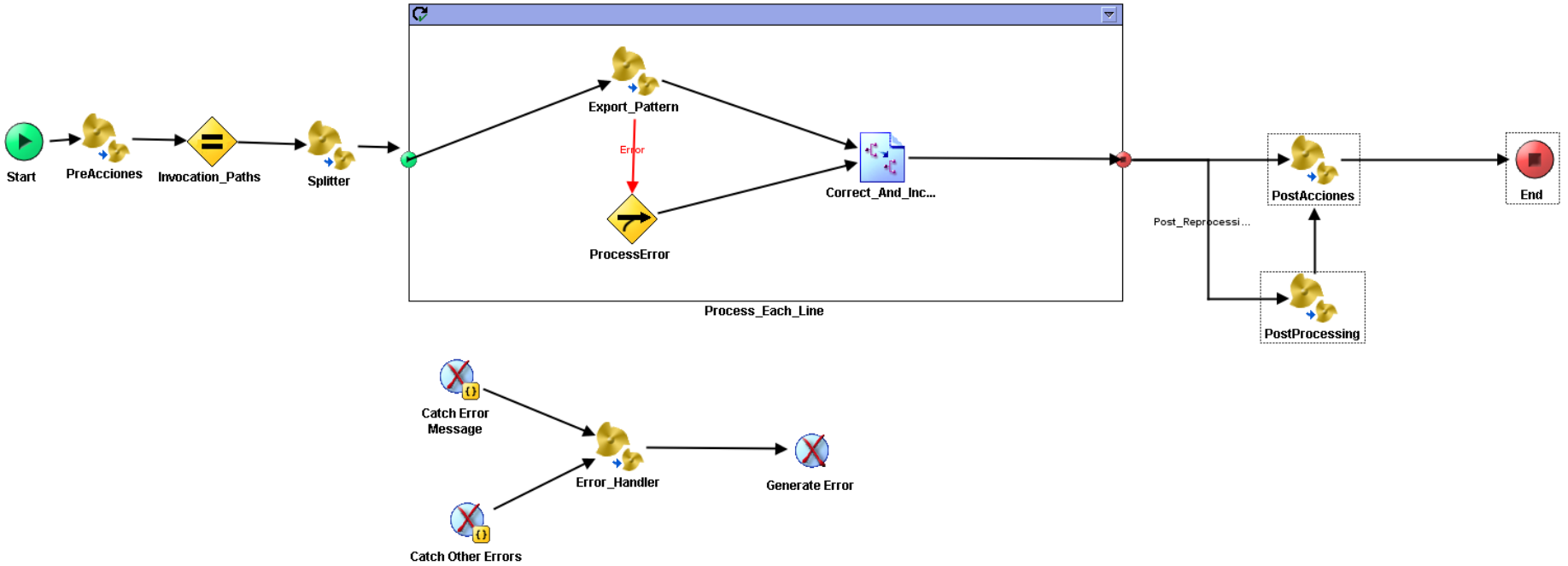


Imagen 76 – Flujo del componente del FW BATCH 1-N

En la imagen anterior se describe el proceso de Batch_1_N_Pattern.

Estos son los pasos que realiza:

4. Al proceso le llega la petición del Starter con la siguiente información:

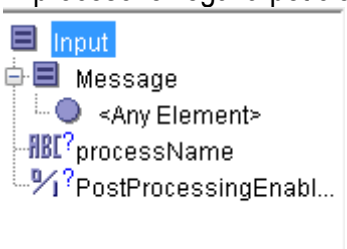


Imagen 77 – Esquema de entrada Batch_N_1_Pattern

En ella se detalla la siguiente información:

Campo	Nivel	Tipo	Descripción
Input	0	Complex Type	Campo padre que alberga la estructura
Message	1	Complex Type	Campo donde se encuentra el mensaje introducido
Any Element	1	Any type	Campo con el contenido del fichero
ProcessName	1	String	Nombre del proceso
PostProcessingEnable	1	Boolean	Flag que indica si se quiere realizar un tratamiento customizado al finalizar el envío de los n mensajes resultantes.

Tabla 11 – Tabla de campos de entrada en el patrón batch N-1

5. A continuación se ejecutan las pre acciones que están formadas por las siguientes acciones:

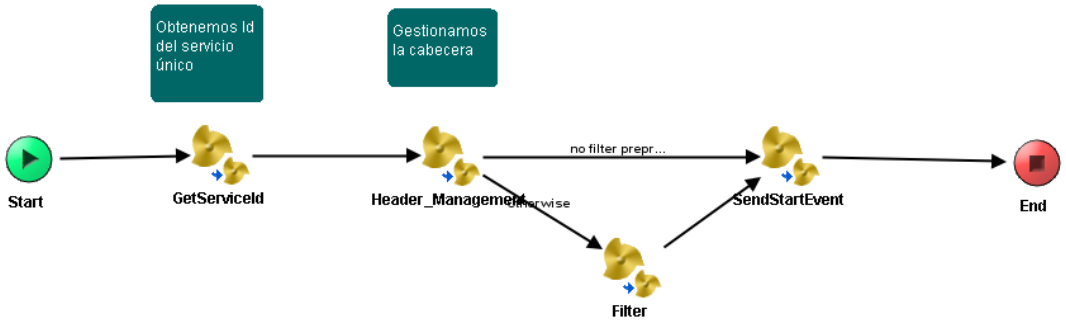


Imagen 78 – Flujo con la ejecución de las distintas pre acciones.

b. La primera es la asignación de un identificador único de la ejecución, mediante el algoritmo de generación de UUID.
c. Después se genera la cabecera de arquitectura en el caso de que no esté ya generada. Esta cabecera estará generada siempre y cuando el adaptador que se está ejecutando no es el primero del flujo, ya que se va propagando. Esta cabecera consta con los siguientes campos:

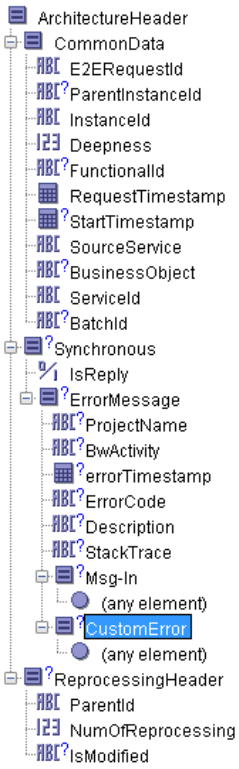


Imagen 79 – Parámetros de la cabecera de arquitectura o gestión.

d. Por último se gestiona la creación de eventos, dependiendo del nivel de trazabilidad establecido. Esta gestión consistirá en el envío del evento a un 3er sistema y la escritura en un log de ejecución.

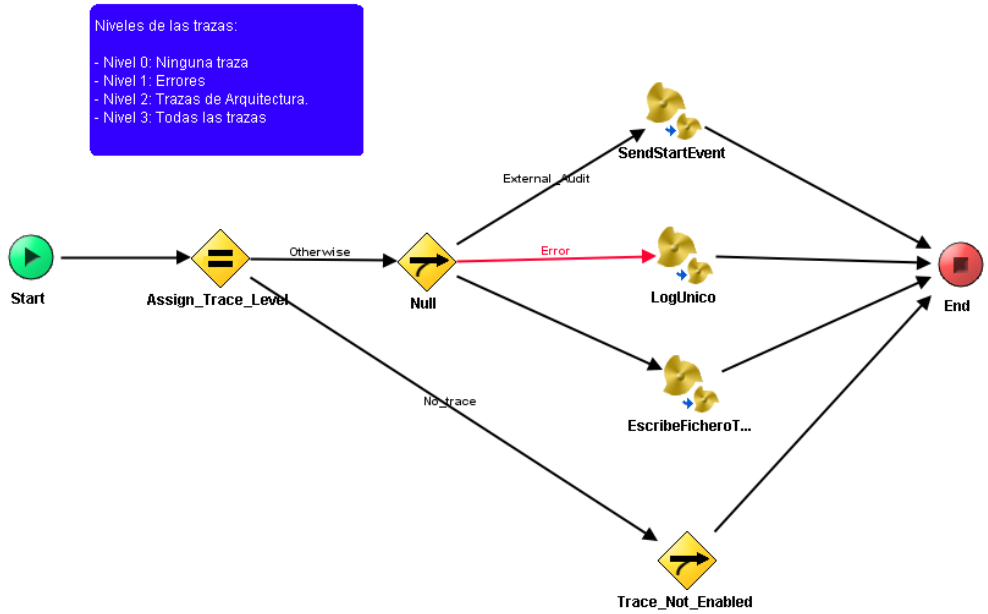
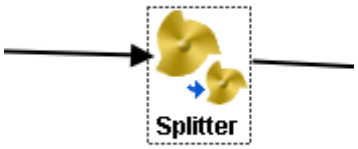
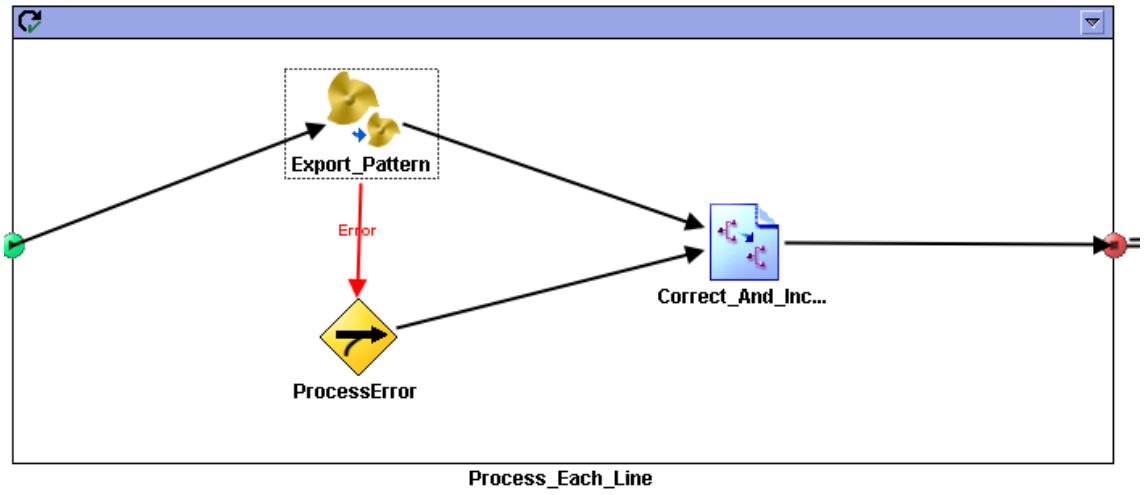


Imagen 80 – Flujo de gestión de eventos.

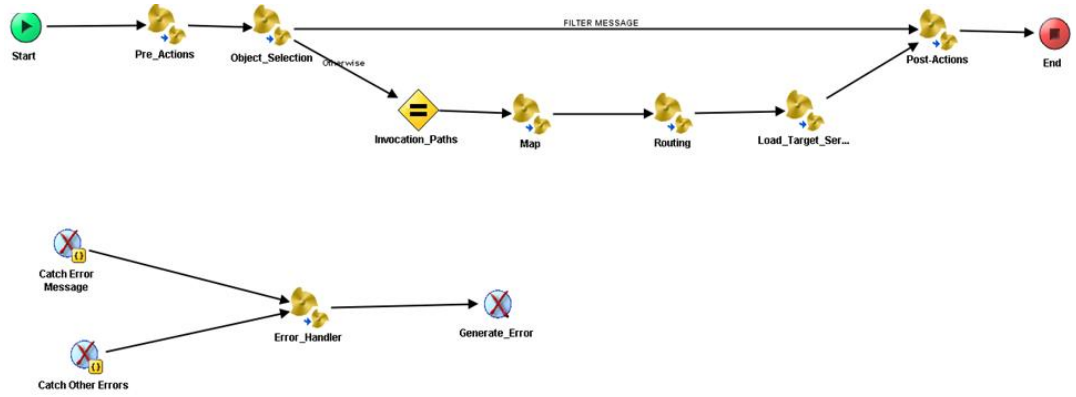
6. El siguiente paso es la llamada dinámica al Splitter. Que se encargará de trocear el mensaje entrante del fichero en N mensajes a enviar al sistema BackEnd



7. El siguiente paso consiste en recorrer el array con todos los mensajes resultantes del Splitter para tratarlos 1 a 1 como peticiones independientes. Al tratarse de un flujo asíncrono, se puede reutilizar el patrón asíncrono del adaptador de exportación. Por lo que se realiza una invocación al patrón de dicho adaptador dentro de un bucle para ejecutarlo por cada mensaje resultante.



8. A partir de este punto se llevan a cabo los pasos del adaptador de exportación anteriormente comentado.



9. Por último se llevan a cabo las postacciones del flujo del adaptador de exportación que contiene las siguientes funcionalidades:

Imagen 81 – Flujo con las post acciones

a. La primera y última acción de las postacciones del adaptador de importación es el envío del evento de fin de ejecución con el resultado obtenido y la escritura de las trazas correspondientes.

3.2.5 Adaptador de canal

Durante este punto se definirá el adaptador de canal que forma parte del patrón síncrono del FW del ESB.

Este adaptador está definido con la intención de recoger las peticiones entrantes dentro del flujo síncrono. Teniendo tantos adaptadores de canal como sistemas frontales invocantes.

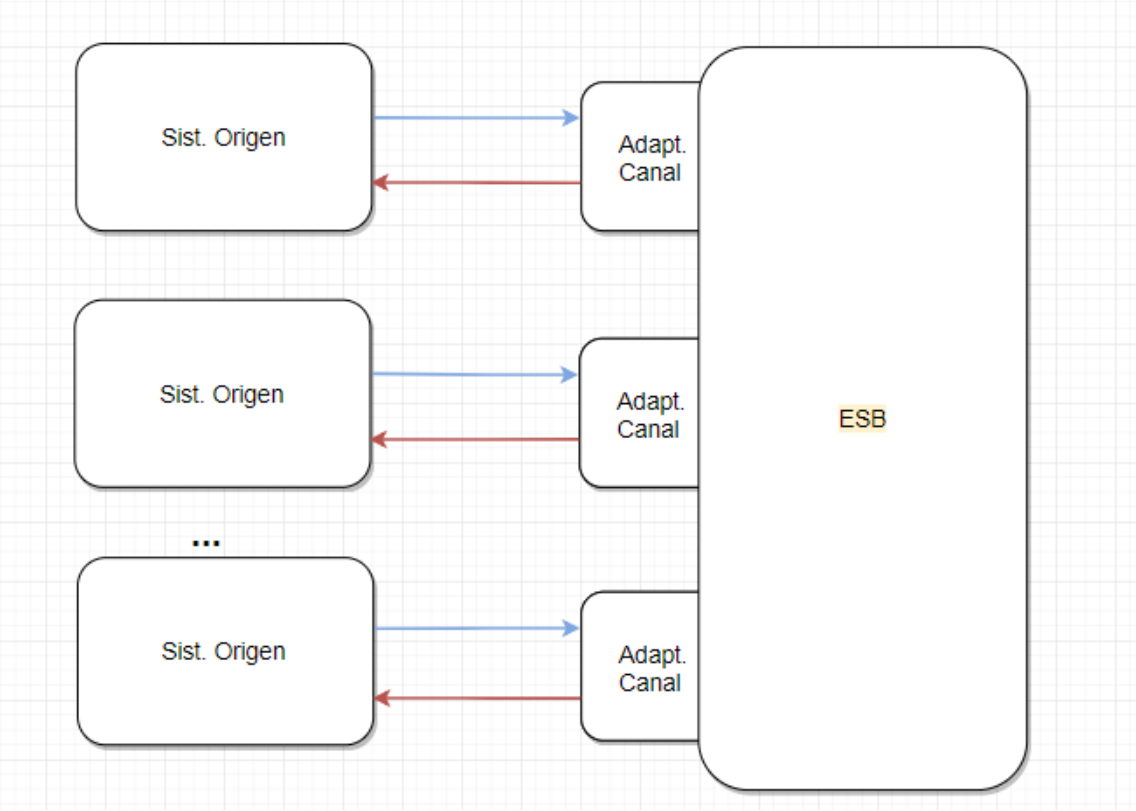


Imagen 82 - Flujo alto nivel adaptador canal

3.2.5.1 Estructura del adaptador

Para poder adaptar los desarrollos que se vayan a realizar sobre el FW del ESB, se tendrá que seguir una serie de pautas necesarias para el acoplamiento entre estos.
Estas pautas son necesarias debido a que el propio FW cuenta con muchas configuraciones dinámicas (invocaciones a procesos de negocio) las cuales dependen de la estructura de las carpetas del proyecto entre otras cosas.

Por esta razón, los adaptadores de canal que se desarrollen bajo el FW del ESB han de contar con la siguiente estructura de carpetas.

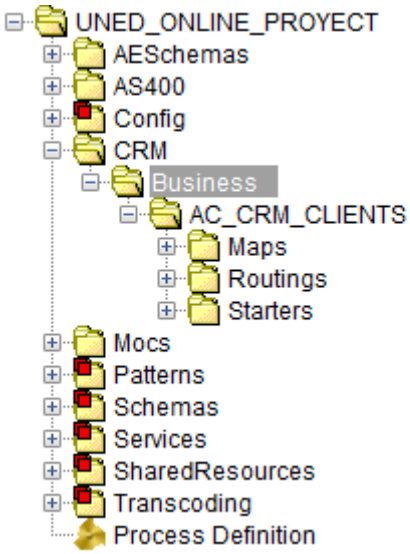


Imagen 83 - Estructura de carpetas de un proyecto de adaptador de canal

Como se observa en la imagen 87 la estructura del proyecto del adaptador de exportación se forma por las siguientes carpetas:

Nombre Carpeta	Nivel	Descripción
NOMBRE_ONLINE _PROJECT	0	Carpeta root del proyecto, en ella está contenidos todos los procesos y funcionalidades.
Config	1	Carpeta destinada a los ficheros de configuración.
NOMBRE_PROYECTO	1	Carpeta con el nombre del proyecto, dentro de esta carpeta se encuentra todo el código del adaptador.
Business	2	Carpeta que contiene el código relativo al requisito del negocio.
AC_NOMBRE_ADAPTADOR	3	Carpeta que contiene todo lo relativo al adaptador.
Error	4	Carpeta donde se encuentran los procesos encargados de la gestión de errores.
Maps		Carpeta donde se encuentran los procesos encargados de realizar los mapeos.
Starters	4	Carpeta donde se encuentran los procesos de starter, es decir, los procesos que reciben la petición del sistema frontal…
SharedResources	2	Carpeta donde se encuentran las conexiones a base de datos, http,jms … así como cualquier elemento de uso común dentro del proyecto.

Deploy	1	Carpeta donde se encuentran los artefactos instalables.

Tabla 12 - Estructura del proyecto

3.2.5.2 Nomenclatura

Al igual que la estructura del proyecto, se necesita seguir una nomenclatura para que el framework funcione correctamente. La nomenclatura a seguir es la siguiente:

Tipo de elemento	Ejemplo Nomenclatura	Descripción
Proceso Starter	AC_CRM_CLIENTS	AC: tipo de adaptador. CRM: Sistema origen. CLIENTS: Entidad
Proceso Map	Map_Request	
Proceso Map	Map_Response	
Proceso Map	Map_Error	
Proceso Routing	Routing	

Tabla 13 – Definición de procesos

3.2.5.3 Tipos de procesos

Dentro de cada adaptador, se podrán diferenciar claramente dos tipos de procesos que serán los que hagan posible su ejecución:

- Procesos de negocio
- Procesos de framework

Procesos de negocio

Este tipo de procesos son los destinados a albergar la lógica de negocio del adaptador. Esta lógica está dividida en las siguientes funcionalidades:

Map: Proceso destinado a realizar las transformaciones. Se divide en dos tipos de mapper:

Request: Transformaciones dedicadas a la petición de entrada.

Response: Transformaciones dedicadas a la respuesta en la salida.
Error: Transformaciones dedicadas a los errores controlados.

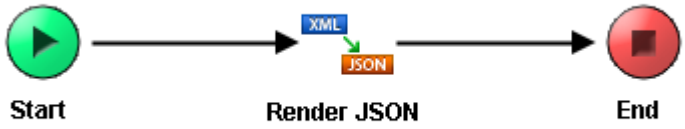


Imagen 84 – Ejemplo de transformación de XML a JSON.

Routing: Proceso dedicado a realizar la lógica de enrutado dependiendo de la petición recibida.

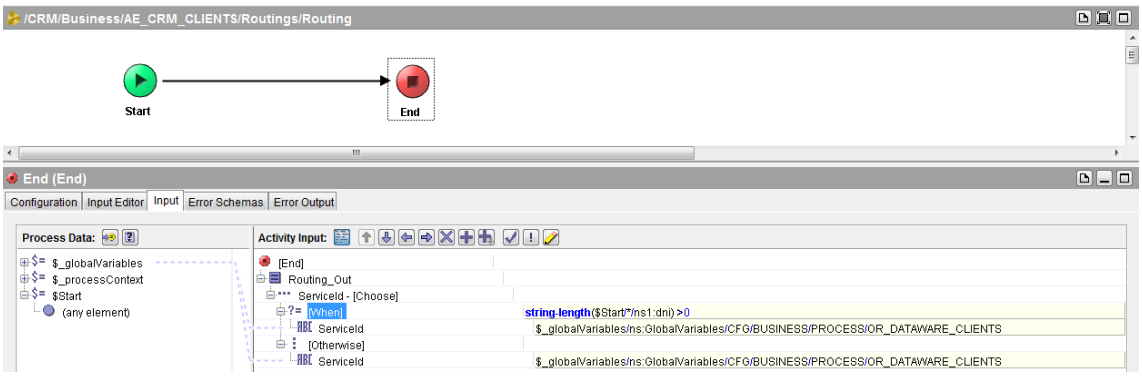


Imagen 85 – Ejemplo de routing mediante lógica en XPath

Procesos de framework

Este tipo de procesos son completamente estándar. No es necesario modificarlos para el desarrollo de ningún adaptador. Son una serie de plantillas cerradas que de forma dinámica irán trazando los mensajes e invocando a los procesos de negocio. Este tipo de procesos son lo que se denomina “Framework” y facilitan la labor del desarrollador abstrayéndolo únicamente a la lógica de negocio. Además gracias a que no se pueden modificar dotan a los adaptadores de una homogeneidad imprescindible a la hora de llevar un soporte adecuado de la herramienta.
Los procesos del framework para el adaptador que tratamos en este punto son los siguientes:

Channel_Adapter_Pattern: Proceso principal del adaptador, actúa a modo de orquestado dirigiendo el flujo por el resto de procesos que conforman el framework para el adaptador que estamos tratando.

PreActions_In: Proceso que contiene una serie de subprocesos que llevan a cabo las tareas necesarias en el inicio del flujo antes de invocar al Orquestador (ORQ) o al proceso simple (PS).

PostActions_In: Proceso que se encarga de auditar el mensaje que se va a enviar al destinatario (ORQ o PS).

Message_to_ESB: Proceso que se encarga de enviar el mensaje al ORQ o al PS mediante comunicación JMS.

PreActions_Out: Proceso que lleva a cabo la auditoría de la respuesta ofrecida por el ORQ o el PS.

PostActions_Out: Proceso que se encarga de auditar el final del adaptador de canal.

3.2.5.4 Diseño por pasos

A continuación se mostrará el diseño paso a paso de dicho adaptador, desde la recepción del mensaje, el envío al PS u ORQ y la recepción y posterior envío de la respuesta.

PASO 1 Recepción del mensaje

El primer paso de todos es la recepción del mensaje, este puede recepcionarse bajo cualquiera de estos protocolos:

- HTTP

Para ello se definirá un proceso de entrada que será el encargado de recepcionarlo.

Se definirá un proceso que ejerza de capa REST, para recibir las peticiones y enviar la correspondiente respuesta.

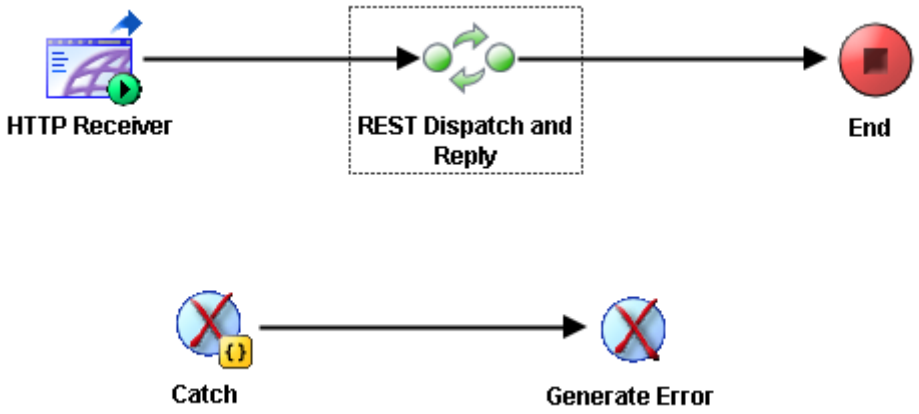


Imagen 86 - Proceso Starter REST

En esta imagen se describe el proceso de Starter:

1. HTTP Receiver: Tarea que dada una URL y un Puerto. Levanta un socket HTTP para recepcionar peticiones.
2. REST Dispatch and Reply: En la tecnología utilizada para ejemplificar este diseño, existe un adaptador REST que facilita el trabajo. Este adaptador se encarga de enrutar las operaciones y urls a los diferentes adaptadores de canal así como el envío de la respuesta. En este adaptador habrá que especificarle:
    a. Las operaciones del servicio REST
    b. Las urls
    c. El adaptador de canal que cubre cada url.
    d. Los interfaces de las Request y Response.

3. End: Una vez el proceso del framework realiza todas las acciones el proceso termina su ejecución.

PASO 2 Starter del Adaptador de Canal

Una vez la ejecución entra en el objeto REST el flujo ejecuta el starter del Adaptador de canal:

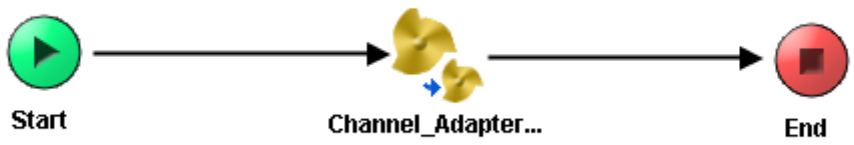


Imagen 87 – Proceso Starter del Adaptador de canal

4. Este proceso recibe el mensaje que ha llegado vía http mediante la tarea de Start.

5. El mensaje se envía al componente del framework “Channel_Adapter_Pattern” que se encarga de ejecutar los procesos del framework y los procesos de negocio mediante llamadas dinámicas.

PASO 3 Gestión de la petición por el Framework.

En este paso el proceso genérico del framework (Channel_Adapter_Pattern) realiza una serie de acciones denominadas preacciones y postacciones, para procesar la petición. Dentro de estas acciones se encuentran las llamadas dinámicas a los procesos de negocio. Este proceso es un proceso que forma parte del Framework, es decir no es modificable por el desarrollador, con la información proporcionada en la entrada y la estructura/nomenclatura definidas es capaz de invocar y trazar todos los procesos que contienen la lógica de negocio.

El flujo de este proceso se detalla a continuación.

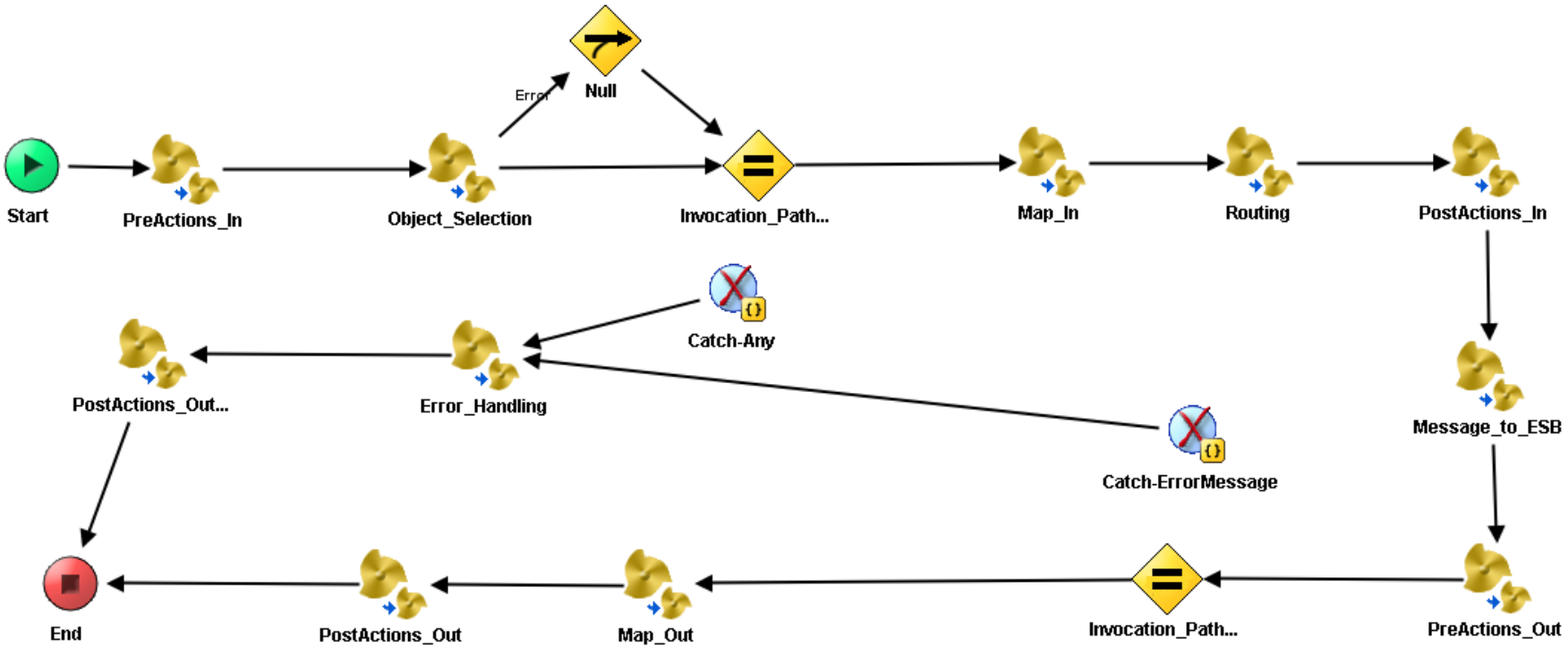


Imagen 88 - Flujo Proceso principal del patrón CA del FW

En la imagen anterior se describe el proceso de Channel_Adapter_Pattern. Estos son los pasos que realiza:

6. Al proceso le llega la petición del Starter con la siguiente información:

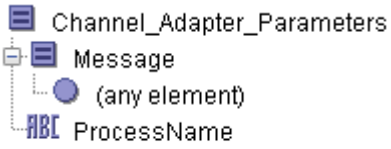


Imagen 89 – Esquema de entrada Channel_Adapter_Pattern

7. A continuación se ejecutan las preacciones de entrada que están formadas por las siguientes acciones:

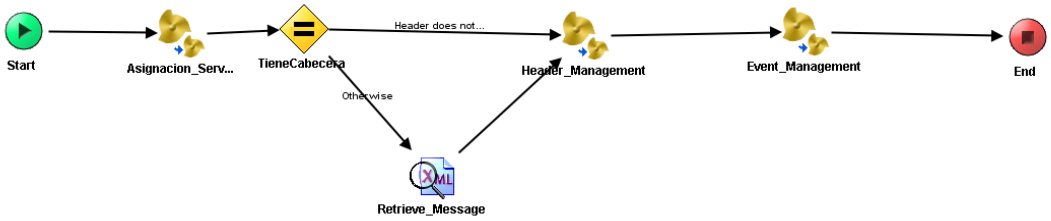


Imagen 90 – Flujo con la ejecución de las distintas preacciones.

a. La primera es la asignación de un identificador único de la ejecución, mediante el algoritmo de generación de UUID.
b. Después se genera la cabecera de arquitectura en el caso de que no esté ya generada. Esta cabecera estará generada siempre y cuando el adaptador que se está ejecutando no es el primero del flujo, ya que se va propagando. Esta cabecera consta con los siguientes campos:

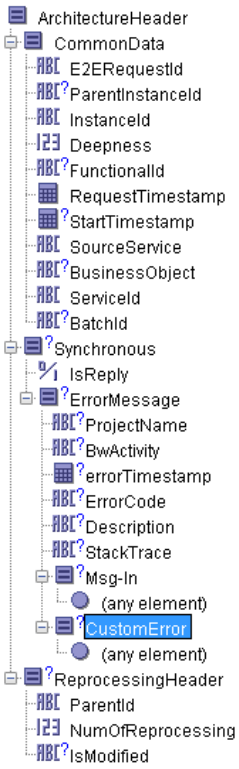


Imagen 91 – Parámetros de la cabecera de arquitectura o gestión.

c. Por último se gestiona la creación de eventos, dependiendo del nivel de trazabilidad establecido. Esta gestión consistirá en el envío del evento a un 3er sistema y la escritura en un log de ejecución.

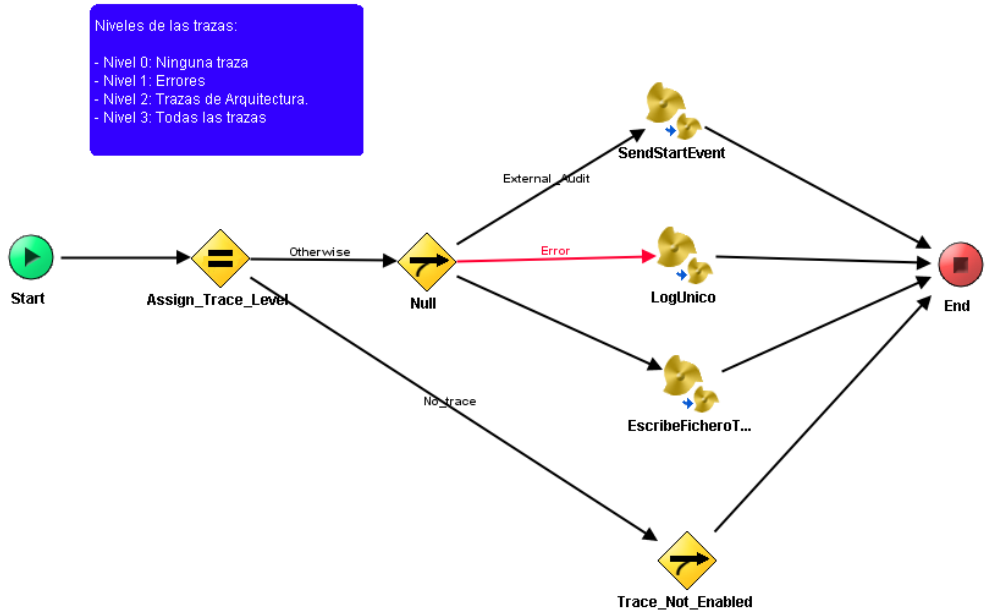
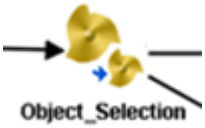


Imagen 92 – Flujo de gestión de eventos.

8. El tercer paso consiste en la selección del objeto. Este paso se encarga de comprobar si se han definido una o varias entidades para este flujo. En caso de ser así se realizará una llamada dinámica al proceso de negocio “ObjectSelection” para discernir de que entidad se trata.

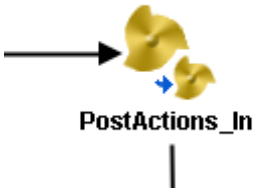


9. A continuación con la entidad resultante del paso anterior se formarán las rutas dinámicas para invocar a los procesos de negocio de mapeos y routing correspondientes con esa entidad.

10. El siguiente paso es la invocación dinámica al proceso de negocio de mapeos. En este paso se invocará con las rutas generadas dinámicamente en el punto anterior al proceso que contiene las transformaciones del mensaje de entrada según los requisitos del negocio.

11. Una vez realizadas las transformaciones necesarias según los requisitos de negocio, se ejecutará de la misma forma dinámica el proceso de Routing. Este proceso tendrá como resultado el identificador del destino del flujo del adaptador de exportación.

12. Con el identificador del destino recién obtenido, se inician las postacciones de entrada:

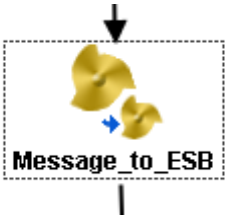


En este subproceso se encuentran las siguientes acciones a realizar:

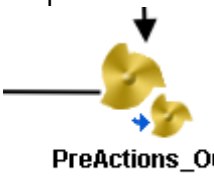
Imagen 93 – Postacciones de entrada.

a. Primero se genera el mensaje de auditoría que se va a enviar.
b. Después se envía dicho mensaje a los logs del ESB y a un 3r sistema si está configurado así.
c. Por último se obtiene la cola con el valor obtenido en el proceso de Routing anterior.

13. El siguiente paso es la invocación al subproceso del framework Message_to_ESB. Que mediante la Queue que se ha obtenido anteriormente se manda el mensaje al siguiente adaptador (PS/ORQ).



14. Una vez enviado el mensaje al siguiente adaptador, el adaptador de canal espera el mensaje con la respuesta de dicho adaptador. Una vez obtenida se comienza con las preacciones de la respuesta:



    a. Este subproceso tan solo envía un evento de auditoría con el resultado de la respuesta.

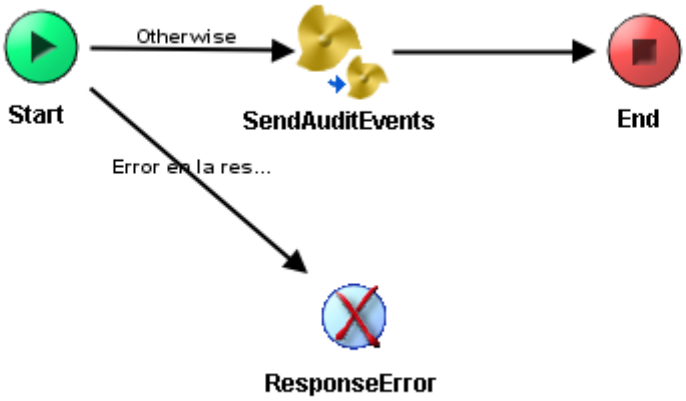
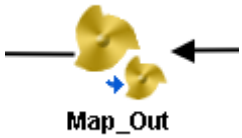


Imagen 94 - PreAcciones de respuesta.

15. El siguiente paso es realizar el mapeo de la respuesta. Este mapeo se encarga de llevar a cabo todas las transformaciones especificadas por el negocio, por lo tanto, se ejecuta de forma dinámica.



16. Por último para terminar con el flujo del adaptador de canal se ejecutan las postacciones de la respuesta.

a. Este subproceso tan solo envía un evento de auditoría con el resultado de la respuesta.

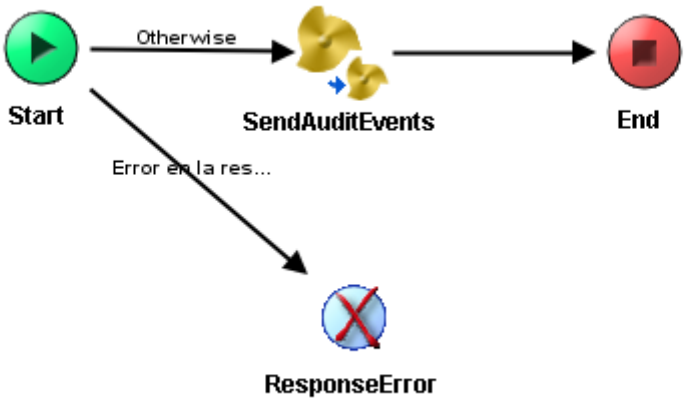


Imagen 95 - PostAcciones de respuesta.

17. Finalmente se devuelve la respuesta al sistema origen como final al flujo síncrono.

3.2.6 Adaptador Orquestado.

Durante este punto se definirá el adaptador de orquestación que forma parte del patrón síncrono del FW del ESB.

Este adaptador está definido con la intención de orquestar bajo una lógica funcional el flujo síncrono de una funcionalidad definida. Este adaptador se dedicará a orquestar una serie de Adaptadores de proceso simple, teniendo como entrada el adaptador de canal definido anteriormente.

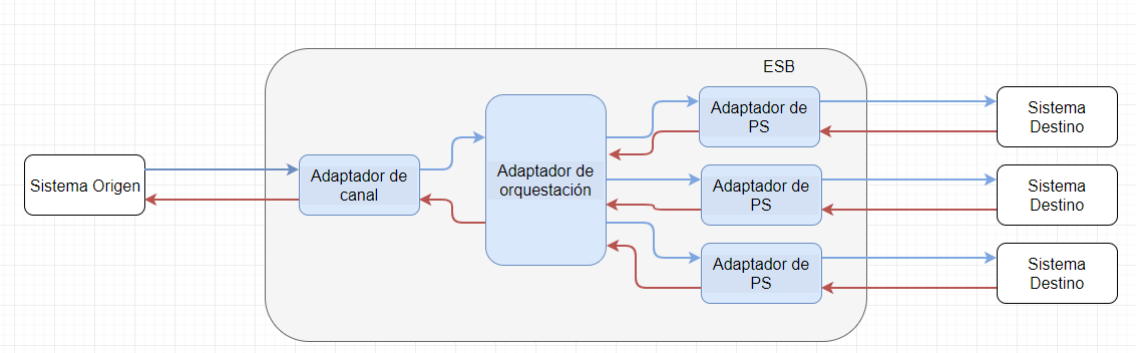


Imagen 96 - Flujo alto nivel adaptador de orquestación

Este adaptador recibirá una petición via JMS del adaptador de canal, que a su vez la ha recibido del sistema origen. Una vez recibida la petición, mediante la lógica de negocio definida, comenzará a enviar peticiones a los distintos procesos simples (PS) que invocan a los sistemas destino. Estas invocaciones pueden variar dependiendo de los resultados obtenidos, p.e. si la primera invocación devuelve como resultado una “X” se invoca al PS 2 y ante cualquier otra respuesta invoca al PS 3. Una vez terminadas todas las invocaciones al PS se devuelve la respuesta al adaptador de canal y este a su vez al sistema frontal.

3.2.6.1 Estructura del adaptador

Para poder adaptar los desarrollos que se vayan a realizar sobre el FW del ESB, se tendrá que seguir una serie de pautas necesarias para el acoplamiento entre estos.
Estas pautas son necesarias debido a que el propio FW cuenta con muchas configuraciones dinámicas (invocaciones a procesos de negocio) las cuales dependen de la estructura de las carpetas del proyecto entre otras cosas.

Por esta razón, los adaptadores de canal que se desarrollen bajo el FW del ESB han de contar con la siguiente estructura de carpetas.

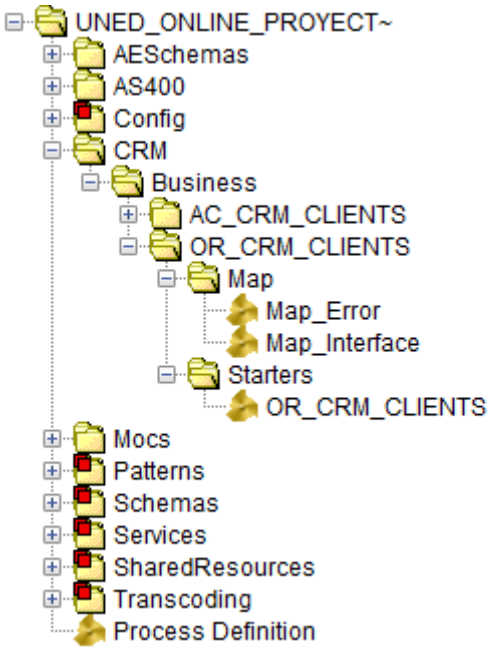


Imagen 97 - Estructura de carpetas de un proyecto de adaptador de orquestación

Como se observa en la imagen 87 la estructura del proyecto del adaptador de exportación se forma por las siguientes carpetas:

Nombre Carpeta	Nivel	Descripción
NOMBRE_ONLINE _PROJECT	0	Carpeta root del proyecto, en ella está contenidos todos los procesos y funcionalidades.
Config	1	Carpeta destinada a los ficheros de configuración.
NOMBRE_PROYECTO	1	Carpeta con el nombre del proyecto, dentro de esta carpeta se encuentra todo el código del adaptador.
Business	2	Carpeta que contiene el código relativo al requisito del negocio.
OR_NOMBRE_ADAPTADOR	3	Carpeta que contiene todo lo relativo al adaptador.
Maps		Carpeta donde se encuentran los procesos encargados de realizar los mapeos.
Starters	4	Carpeta donde se encuentran los procesos de starter, es decir, los procesos que reciben la petición del sistema frontal…
SharedResources	2	Carpeta donde se encuentran las conexiones a base de datos, http, jms… así como cualquier elemento de uso común dentro del proyecto.

Deploy	1	Carpeta donde se encuentran los artefactos instalables.

Tabla 14 - Estructura del proyecto

3.2.6.2 Nomenclatura

Al igual que la estructura del proyecto, se necesita seguir una nomenclatura para que el framework funcione correctamente. La nomenclatura a seguir es la siguiente:

Tipo de elemento	Ejemplo Nomenclatura	Descripción
Proceso Starter	OR_CRM_CLIENTS	OR: tipo de adaptador. CRM: Sistema origen. CLIENTS: Entidad
Proceso Map	Map_Interface	
Proceso Map	Map_Error	

Tabla 15 – Definición de procesos

3.2.6.3 Tipos de procesos

Dentro de cada adaptador, se podrán diferenciar claramente dos tipos de procesos que serán los que hagan posible su ejecución:

- Procesos de negocio
- Procesos de framework

Procesos de negocio

Este tipo de procesos son los destinados a albergar la lógica de negocio del adaptador. Esta lógica está dividida en las siguientes funcionalidades:

Map: Proceso destinado a realizar las transformaciones. Se divide en dos tipos de mapper:

Interface: Proceso donde se realiza la orquestación a los procesos Simples
Error: Mapeo de error en el caso de realizarse.
Error: Transformaciones dedicadas a los errores controlados.

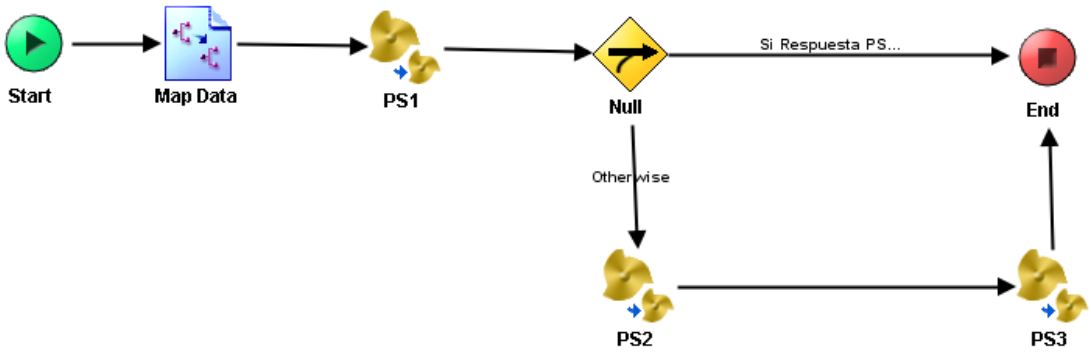


Imagen 98 – Ejemplo de orquestación (Map_Interface) de 3 PS.

Procesos de framework

Este tipo de procesos son completamente estándar. No es necesario modificarlos para el desarrollo de ningún adaptador. Son una serie de plantillas cerradas que de forma dinámica irán trazando los mensajes e invocando a los procesos de negocio. Este tipo de procesos son lo que se denomina "Framework" y facilitan la labor del desarrollador abstrayéndolo únicamente a la lógica de negocio. Además gracias a que no se pueden modificar dotan a los adaptadores de una homogeneidad imprescindible a la hora de llevar un soporte adecuado de la herramienta.
Los procesos del framework para el adaptador que tratamos en este punto son los siguientes:

Orchestrated_Pattern: Proceso principal del adaptador, actúa a modo de orquestado dirigiendo el flujo por el resto de procesos que conforman el framework para el adaptador que estamos tratando.

PreActions_In: Proceso que contiene una serie de subprocesos que llevan a cabo las tareas necesarias en el inicio del flujo, antes de iniciar la orquestación.

PostActions_Out: Proceso que se encarga de auditar el final del adaptador de orquestación.

3.2.6.4 Diseño por pasos

A continuación se mostrará el diseño paso a paso de dicho adaptador, desde la recepción del mensaje, el envío a los diferentes PS y la recepción y posterior envío de la respuesta.

PASO 1 Recepción del mensaje

El primer paso de todos es la recepción del mensaje, este puede recepcionarse bajo cualquiera de estos protocolos:

- JMS

Este mensaje JMS se recibirá del Adaptador de canal, que previamente lo ha recibido vía http de un sistema origen. Para ello se definirá un proceso de entrada que será el encargado de recepcionarlo.

Una vez recepcionado el mensaje y procesada la orquestación se enviará la respuesta mediante una cola temporal asociada a la cola de entrada.

Este mensaje llegara al adaptador de canal que lo devolverá en forma de respuesta HTTP al sistema Origen.

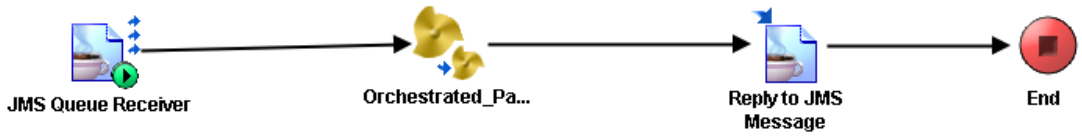


Imagen 99 - Proceso Starter JMS

En esta imagen se describe el proceso de Starter:

1. JMS Queue Receiver: Tarea que escucha de una queue definida recibiendo cada mensaje que deposita en ella el adaptador de canal correspondiente.
2. Orchestrated_Pattern: Tarea del framework que orquesta toda las tareas propias del framework y del negocio.
3. Reply to JMS Message: Tarea que una vez recibida la respuesta de los diferentes servicios de orquestación, devuelve una respuesta al adaptador de canal en una queue temporal asociada a la queue de origen.
4. End: Tarea que finaliza el flujo del proceso

PASO 2 Gestión de la petición por el Framework.

En este paso el proceso genérico del framework (Orchestrated _Pattern) realiza una serie de acciones denominadas preacciones y postacciones, para procesar la petición. Dentro de estas acciones se encuentran las llamadas dinámicas a los procesos de negocio. Este proceso es un proceso que forma parte del Framework, es decir no es modificable por el desarrollador, con la información proporcionada en la entrada y la estructura/nomenclatura definidas es capaz de invocar y trazar todos los procesos que contienen la lógica de negocio.

El flujo de este proceso se detalla a continuación.

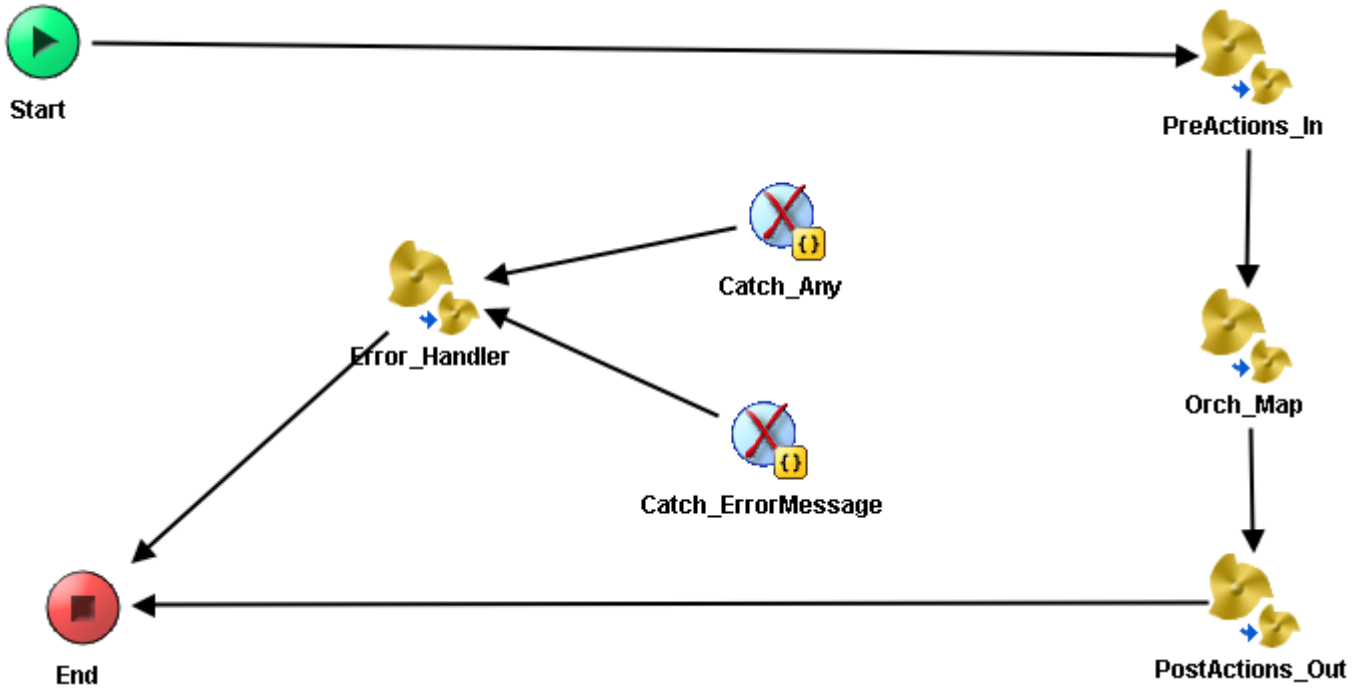


Imagen 100 - Flujo Proceso principal del patrón OR del FW

En la imagen anterior se describe el proceso de Orchestrated_Pattern. Estos son los pasos que realiza:

5. Al proceso le llega la petición del Starter con la siguiente información:

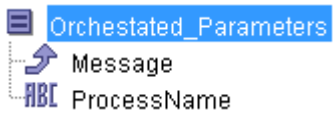


Imagen 101 – Esquema de entrada Orchestrated _Pattern

6. A continuación se ejecutan las preacciones de entrada que están formadas por las siguientes acciones:

Imagen 102 – Flujo con la ejecución de las distintas preacciones.

   a. La primera es la asignación de un identificador único de la ejecución, mediante el algoritmo de generación de UUID.
   b. Después se genera la cabecera de arquitectura en el caso de que no esté ya generada. Esta cabecera estará generada siempre y cuando el adaptador que se está ejecutando no es el primero del flujo, ya que se va propagando. Esta cabecera consta con los siguientes campos:

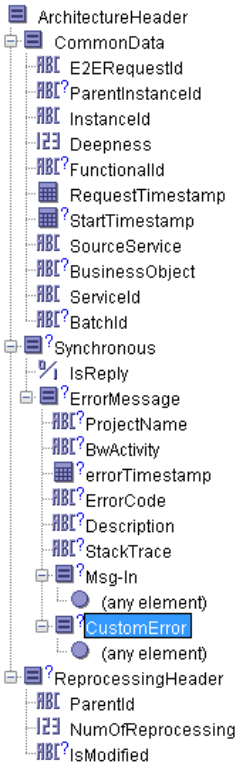


Imagen 103 – Parámetros de la cabecera de arquitectura o gestión.

c. Por último se gestiona la creación de eventos, dependiendo del nivel de trazabilidad establecido. Esta gestión consistirá en el envío del evento a un 3er sistema y la escritura en un log de ejecución.

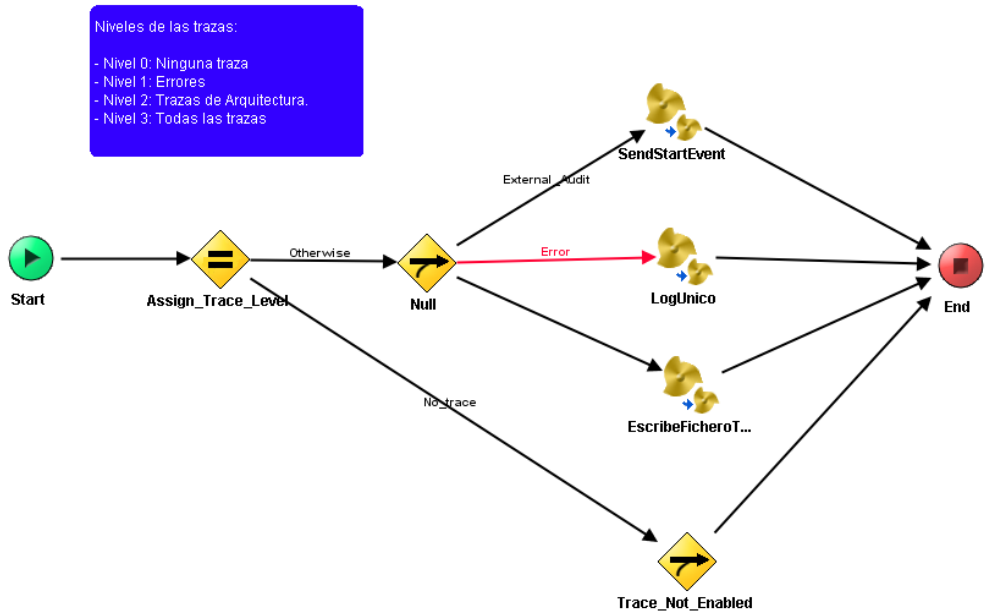


Imagen 104 – Flujo de gestión de eventos.

7. El siguiente paso es la ejecución del Map_Interface. Este es el proceso funcional encargado de realizar las orquestaciones de los diferentes PS.

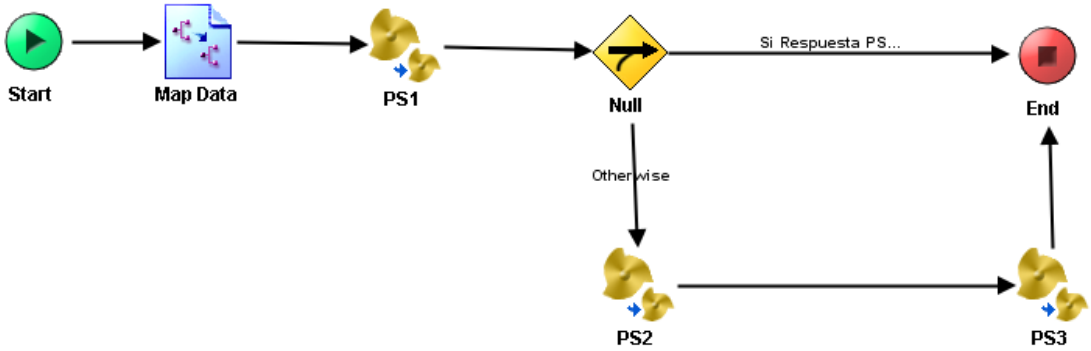
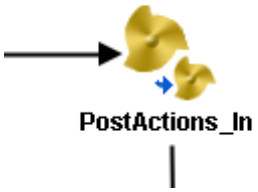


Imagen 105 – Flujo de orquestación del proceso Map_Interface funcional

8. Con el identificador del destino recién obtenido, se inician las postacciones de salida:



En este subproceso se encuentran las siguientes acciones a realizar:

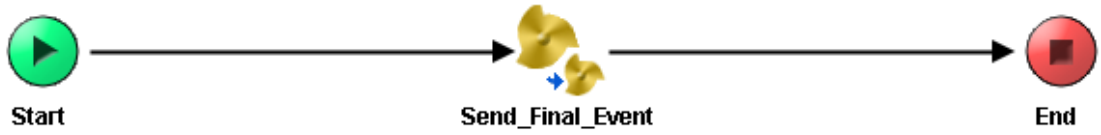


Imagen 106 – Proceso de Postacciones .

a. Este proceso tan solo genera el mensaje de auditoría relacionado con el fin del orquestador.

3.2.7 Adaptador de proceso simple

Durante este punto se definirá el adaptador de proceso simple que forma parte del patrón síncrono del FW del ESB.

Este adaptador está definido con la intención de realizar una comunicación síncrona con un sistema backend.

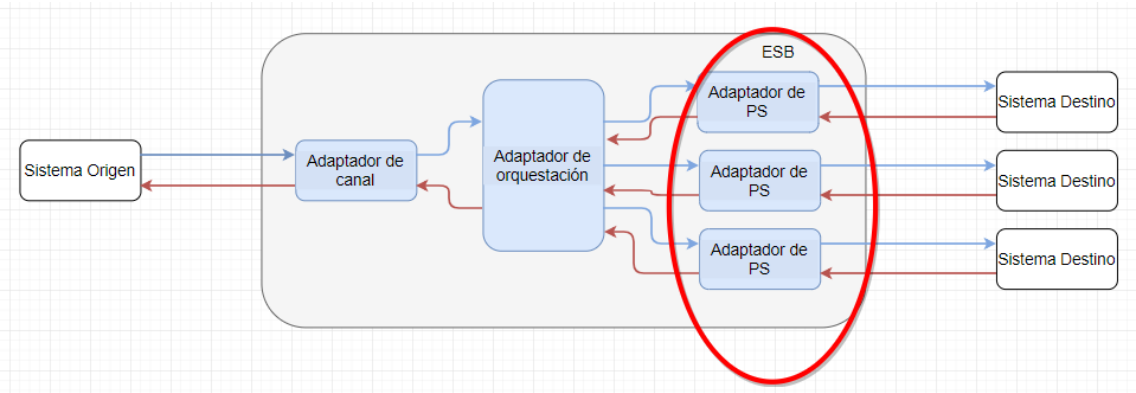


Imagen 107 - Flujo alto nivel adaptador de proceso simple

3.2.7.5 Estructura del adaptador

Para poder adaptar los desarrollos que se vayan a realizar sobre el FW del ESB, se tendrá que seguir una serie de pautas necesarias para el acoplamiento entre estos.
Estas pautas son necesarias debido a que el propio FW cuenta con muchas configuraciones dinámicas (invocaciones a procesos de negocio) las cuales dependen de la estructura de las carpetas del proyecto entre otras cosas.

Por esta razón, los adaptadores de exportación que se desarrollen bajo el FW del ESB han de contar con la siguiente estructura de carpetas.

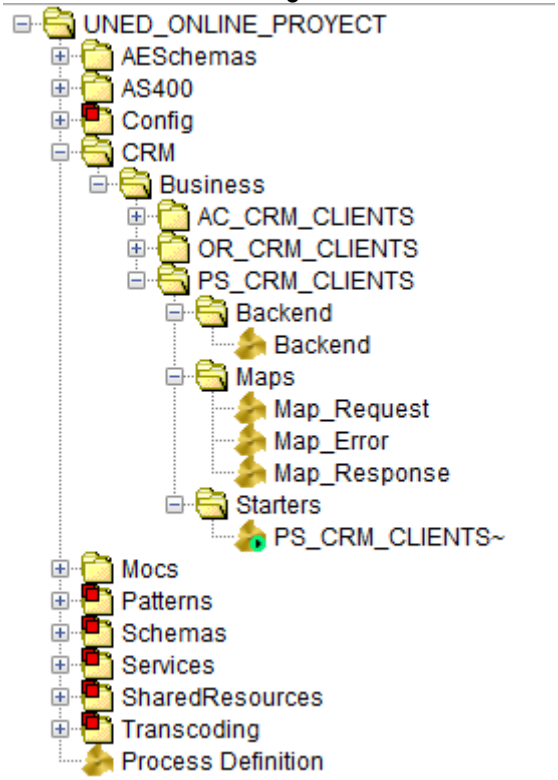


Imagen 108 - Estructura de carpetas de un proyecto de adaptador de proceso simple

Como se observa en la imagen 24 la estructura del proyecto del adaptador de exportación se forma por las siguientes carpetas:

Nombre Carpeta	Nivel	Descripción
NOMBRE_ONLINE_PROJECT	0	Carpeta root del proyecto, en ella está contenidos todos los procesos y funcionalidades.
Config	1	Carpeta destinada a los ficheros de configuración.
NOMBRE_PROYECTO	1	Carpeta con el nombre del proyecto, dentro de esta carpeta se encuentra todo el código del adaptador.
Business	2	Carpeta que contiene el código relativo al requisito del negocio.
PS_NOMBRE_ADAPTADOR	3	Carpeta que contiene todo lo relativo al adaptador.
Backend	4	Carpeta donde se encuentran los procesos encargados de la gestión de la invocación al backend.
Maps		Carpeta donde se encuentran los

		procesos encargados de realizar los mapeos.
Schemas	4	Carpeta donde se almacenan los interfaces del proceso(XSD,WSDL,WADL,SWAGGER…)
Starters	4	Carpeta donde se encuentran los procesos de starter, es decir, los procesos que reciben la petición del sistema frontal…
SharedResources	2	Carpeta donde se encuentran las conexiones a base de datos, http,jms … así como cualquier elemento de uso común dentro del proyecto.
Deploy	1	Carpeta donde se encuentran los artefactos instalables.

Tabla 16 - Estructura del proyecto

3.2.7.6 Nomenclatura

Al igual que la estructura del proyecto, se necesita seguir una nomenclatura para que el framework funcione correctamente. La nomenclatura a seguir es la siguiente:

Tipo de elemento	Ejemplo Nomenclatura	Descripción
Proceso Starter	PS_CRM_CLIENTS	PS: tipo de adaptador. CRM: Sistema origen. CLIENTS: Entidad
Proceso Map	Map_Request	Mapeo de la petición al backend
Proceso Map	Map_Response	Mapeo de la respuesta del backend
Proceso Map	Map_Error	Mapeo del posible error en el flujo
Proceso Backend	Backend	Proceso de envío de petición al Backend

Tabla 17 – Definición de procesos

3.2.7.7 Tipos de procesos

Dentro de cada adaptador, se podrán diferenciar claramente dos tipos de procesos que serán los que hagan posible su ejecución:

- Procesos de negocio
- Procesos de framework

Procesos de negocio

Este tipo de procesos son los destinados a albergar la lógica de negocio del adaptador. Esta lógica está dividida en las siguientes funcionalidades:

Map: Proceso destinado a realizar las transformaciones. Se divide en dos tipos de mapper:

Request: Transformaciones dedicadas a la petición de entrada.
Response: Transformaciones dedicadas a la respuesta en la salida.
Error: Transformaciones dedicadas a los errores controlados.

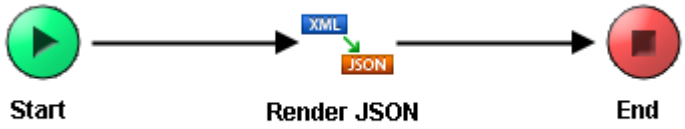


Imagen 109 – Ejemplo de transformación de XML a JSON.

Backend: Proceso dedicado a realizar la lógica de envío al backend y recepción de su respuesta

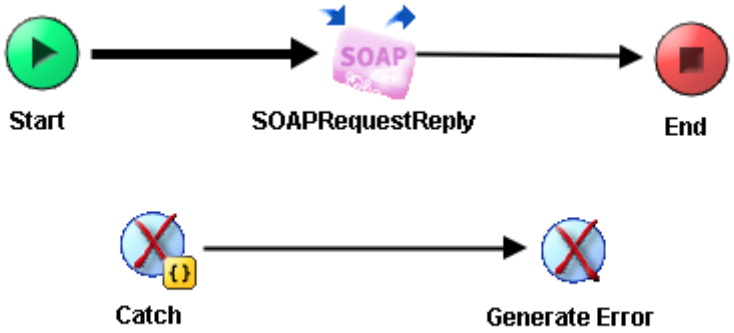


Imagen 110 – Ejemplo de selección envío de petición al backend bajo protocolo SOAP.

Procesos de framework

Este tipo de procesos son completamente estándar. No es necesario modificarlos para el desarrollo de ningún adaptador. Son una serie de plantillas cerradas que de forma dinámica irán trazando los mensajes e invocando a los procesos de negocio. Este tipo de procesos son lo que se denomina "Framework" y facilitan la labor del desarrollador abstrayéndolo únicamente a la lógica de negocio. Además gracias a que no se pueden modificar dotan a los adaptadores de una homogeneidad imprescindible a la hora de llevar un soporte adecuado de la herramienta.
Los procesos del framework para el adaptador que tratamos en este punto son los siguientes:

SimpleService_Pattern: Proceso principal del adaptador, actúa a modo de orquestado dirigiendo el flujo por el resto de procesos que conforman el framework para el adaptador que estamos tratando.

PreActions_In: Proceso que contiene una serie de subprocesos que llevan a cabo las tareas necesarias en el inicio del flujo antes de invocar al backend.

PostActions_In: Proceso que se encarga de auditar el mensaje que se va a enviar al destinatario.

Request_Response: Proceso que se encarga de enviar el mensaje al backend mediante un protocolo determinado.

PreActions_Out: Proceso que lleva a cabo la auditoría de la respuesta ofrecida por el backend.

PostActions_Out: Proceso que se encarga de auditar el final del adaptador de proceso simple.

3.2.7.8 Diseño por pasos

A continuación se mostrará el diseño paso a paso de dicho adaptador, desde la recepción del mensaje al envío a su adaptador de importación correspondiente.

PASO 1 Recepción del mensaje

El primer paso de todos es la recepción del mensaje, este puede recepcionarse bajo cualquiera de estos protocolos:

- JMS

Este mensaje JMS se recibirá del Adaptador de canal, que previamente lo ha recibido vía http de un sistema origen. Para ello se definirá un proceso de entrada que será el encargado de recepcionarlo.

Una vez recepcionado el mensaje y procesada la orquestación se enviará la respuesta mediante una cola temporal asociada a la cola de entrada.

Este mensaje llegara al adaptador de canal que lo devolverá en forma de respuesta HTTP al sistema Origen.

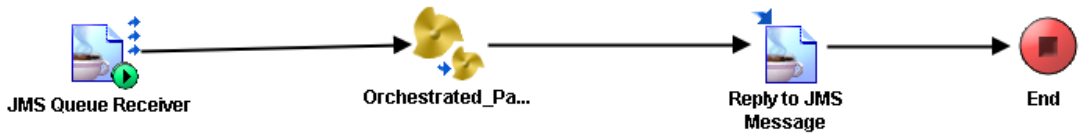


Imagen 111 - Proceso Starter JMS

En esta imagen se describe el proceso de Starter:

1. JMS Queue Receiver: Tarea que escucha de una queue definida recibiendo cada mensaje que deposita en ella el adaptador de canal correspondiente.
2. Orchestrated_Pattern: Tarea del framework que orquesta toda las tareas propias del framework y del negocio.
3. Reply to JMS Message: Tarea que una vez recibida la respuesta de los diferentes servicios de orquestación, devuelve una respuesta al adaptador de canal en una queue temporal asociada a la queue de origen.
4. End: Tarea que finaliza el flujo del proceso

PASO 2 Gestión de la petición por el Framework.

En este paso el proceso genérico del framework (SimpleService_Pattern) realiza una serie de acciones denominadas preacciones y postacciones, para procesar la petición. Dentro de estas acciones se encuentran las llamadas dinámicas a los procesos de negocio. Este proceso es un proceso que forma parte del Framework, es decir no es modificable por el desarrollador, con la información proporcionada en la entrada y la estructura/nomenclatura definidas es capaz de invocar y trazar todos los procesos que contienen la lógica de negocio.

El flujo de este proceso se detalla a continuación.

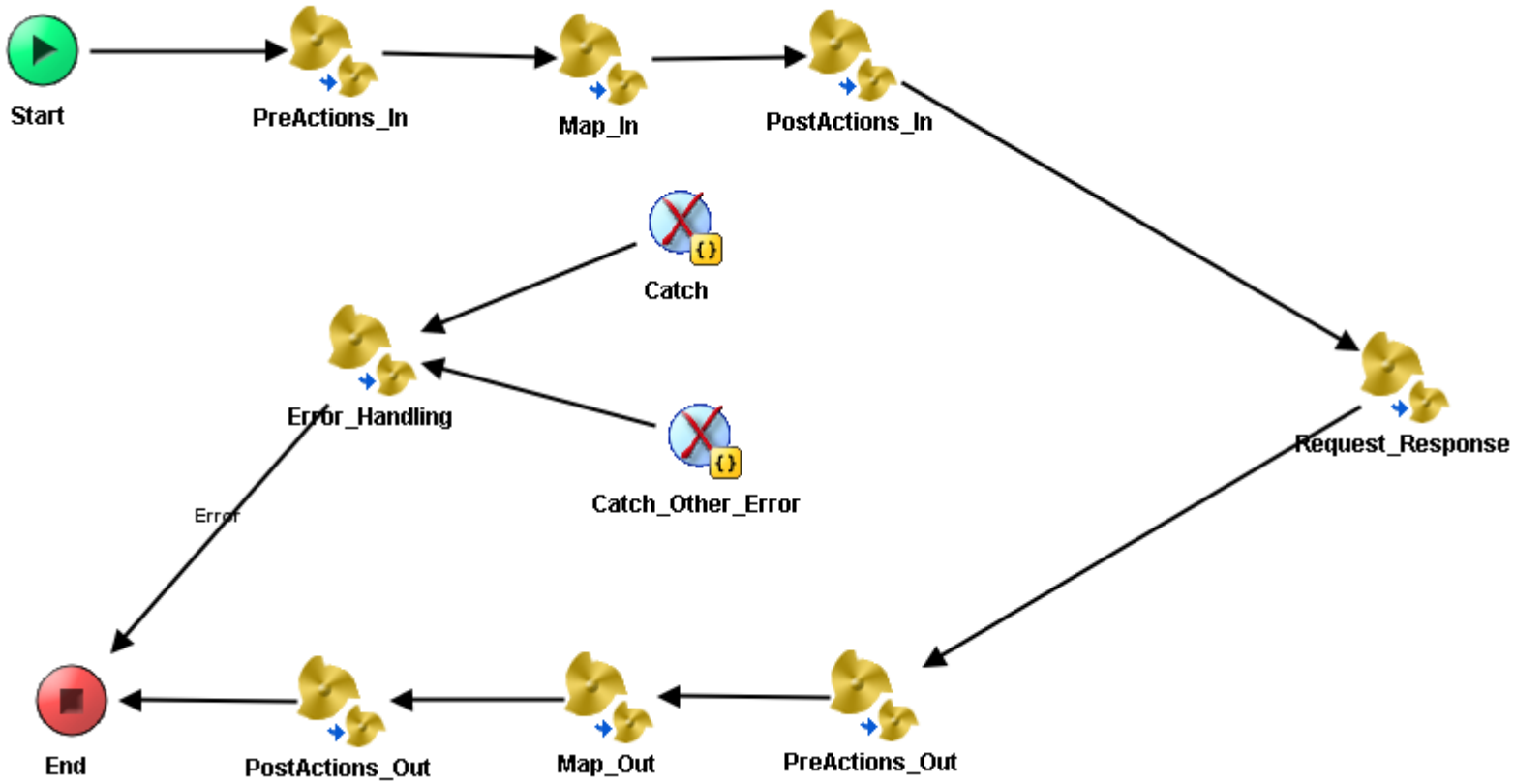


Imagen 112 – Flujo Proceso principal Adaptador de proceso simple FW

En la imagen anterior se describe el proceso de SimpleService-Pattern. Estos son los pasos que realiza:

5. Al proceso le llega la petición del Starter con la siguiente información:

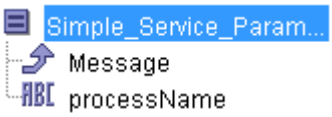


Imagen 113 – Esquema de entrada SimpleService-Pattern

6. A continuación se ejecutan las preacciones que están formadas por las siguientes acciones:

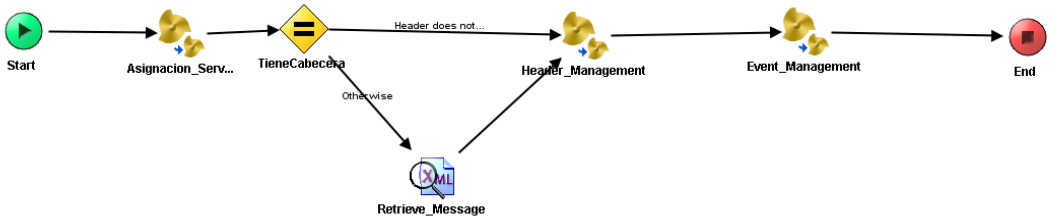


Imagen 114 – Flujo con la ejecución de las distintas preacciones.

   a. La primera es la asignación de un identificador único de la ejecución, mediante el algoritmo de generación de UUID.
   b. Después se genera la cabecera de arquitectura en el caso de que no esté ya generada. Esta cabecera estará generada siempre y cuando el adaptador que se está ejecutando no es el primero del flujo, ya que se va propagando. Esta cabecera consta con los siguientes campos:

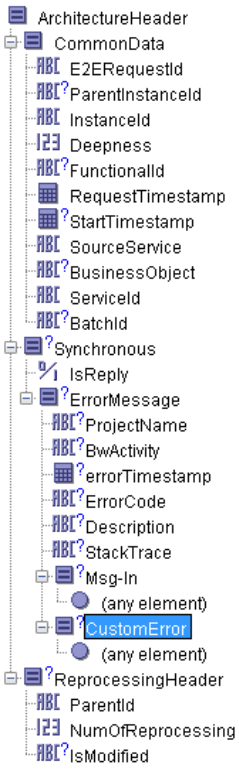


Imagen 115 – Parámetros de la cabecera de arquitectura o gestión.

c. Por último se gestiona la creación de eventos, dependiendo del nivel de trazabilidad establecido. Esta gestión consistirá en el envío del evento a un 3er sistema y la escritura en un log de ejecución.

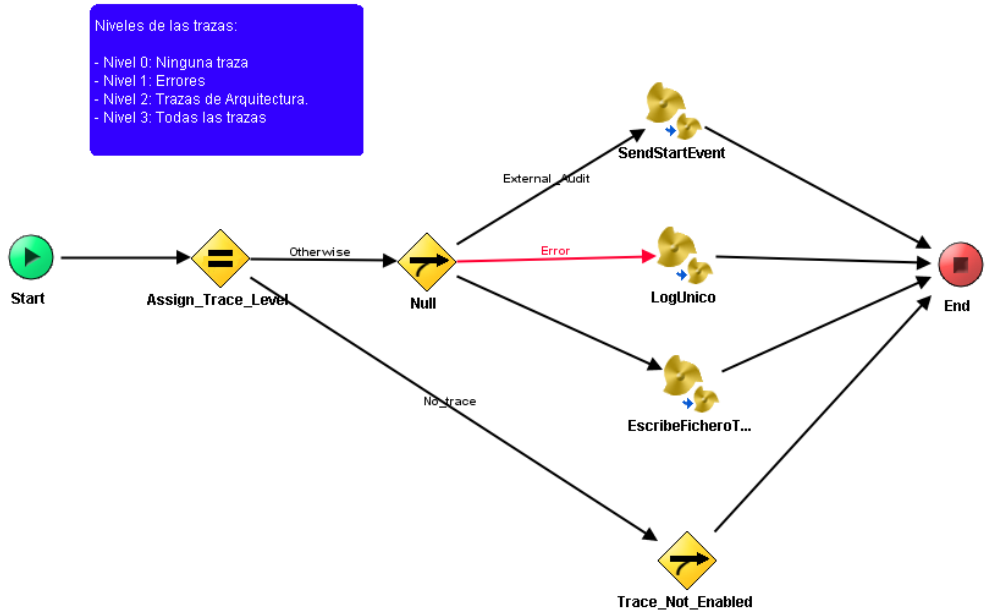
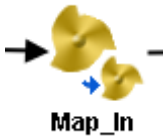


Imagen 116 – Flujo de gestión de eventos.

7. El siguiente paso es la invocación dinámica al proceso de negocio de mapeos. En este paso se invocará con las rutas generadas dinámicamente en el punto anterior al proceso que contiene las transformaciones del mensaje de entrada según los requisitos del negocio. Este mapeo es el relativo a la petición de entrada sobre el backend que se realizará en el siguiente paso.



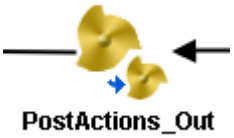
8. El paso siguiente consiste en el subproceso del fw de postacciones antes de la petición al backend. Este paso sirve básicamente para trazar y auditar el mensaje mapeado que se enviará al backend.

9. Una vez realizadas las transformaciones necesarias según los requisitos de negocio, se ejecutará de la misma forma dinámica el proceso de invocación al backend. Al ser una integración síncrona se lanzará una petición al sistema backend esperando la respuesta.

10. Una vez recibida la respuesta a la petición del backend. Se ejecutará el subproceso de postactions_out. El fin de este subproceso es trazar la respuesta del backend antes de mapearla conforme al interfaz definido en el PS.

11. El paso siguiente será el de realizar la llamada al proceso de Map_Response, que se realizará como todas las invocaciones a procesos de negocio, de forma dinámica. En este proceso se transformará la respuesta recibida del sistema backend en la respuesta definida en la interfaz del PS.

12. Por último se ejecutan las postacciones de después de invocar al backend. En este subproceso se trazará el mensaje final resultante así como el final de la ejecución del adaptador de proceso simple.



4 CASO DE USO FICTICIO

4.1 INTRODUCCIÓN

En este apartado se definirá un escenario de una empresa ficticia con el objetivo de mostrar cuales son las necesidades que pueden llevar a la implantación de un ESB. A su vez también se analizarán los beneficios que puede aportar y carencias que puede cubrir en los sistemas de la empresa.

4.2 DESCRIPCIÓN DE LA EMPRESA

La empresa sobre la que se va a desarrollar este análisis de implantación, es una empresa ficticia dedicada al sector Retail. La compañía está inmersa en un proceso de digitalización el cual le ha llevado a una modernización de sus sistemas. Centrando el foco actualmente en sentar las bases a nivel de infraestructura. Una de esas medidas es la implantación de un ESB para reorganizar las comunicaciones dentro de la compañía.

La idea principal es focalizar las integraciones en este nuevo componente de modo que se pueda adquirir un gobierno sobre las mismas y dotar a los sistemas de una trazabilidad que ayude a localizar los puntos de error del proceso de negocio a nivel técnico.

Durante el resto del apartado denominaremos a esta empresa RetailCo.

4.2.1 RetailCo

Empresa destinada a la venta de alimentos al público general. Consta de un número bastante amplio de supermercados en todo el área nacional y con una infraestructura para dar soporte a su eCommerce.

De este modo se podría dividir la compañía en dos líneas de negocio.

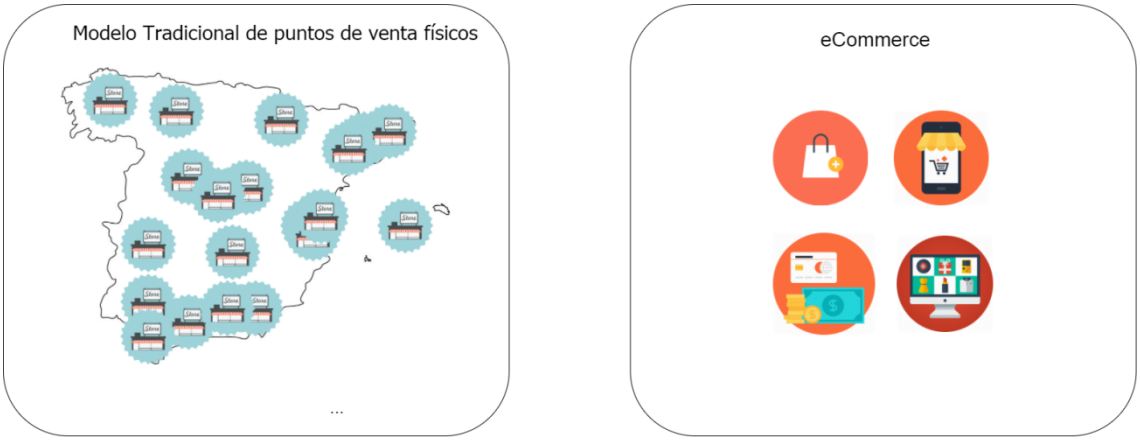


Imagen 117 – Modelo de negocio simplificado

4.3 ARQUITECTURA DE SISTEMAS

Durante este punto se definirá la arquitectura de RetailCo. Para simplificar este punto, ya que no es un punto fundamental del proyecto se ha simplificado su arquitectura a 4 sistemas, que son los suficientes para poder demostrar la usabilidad del producto ESB.

De este modo la arquitectura quedaría de la siguiente forma:

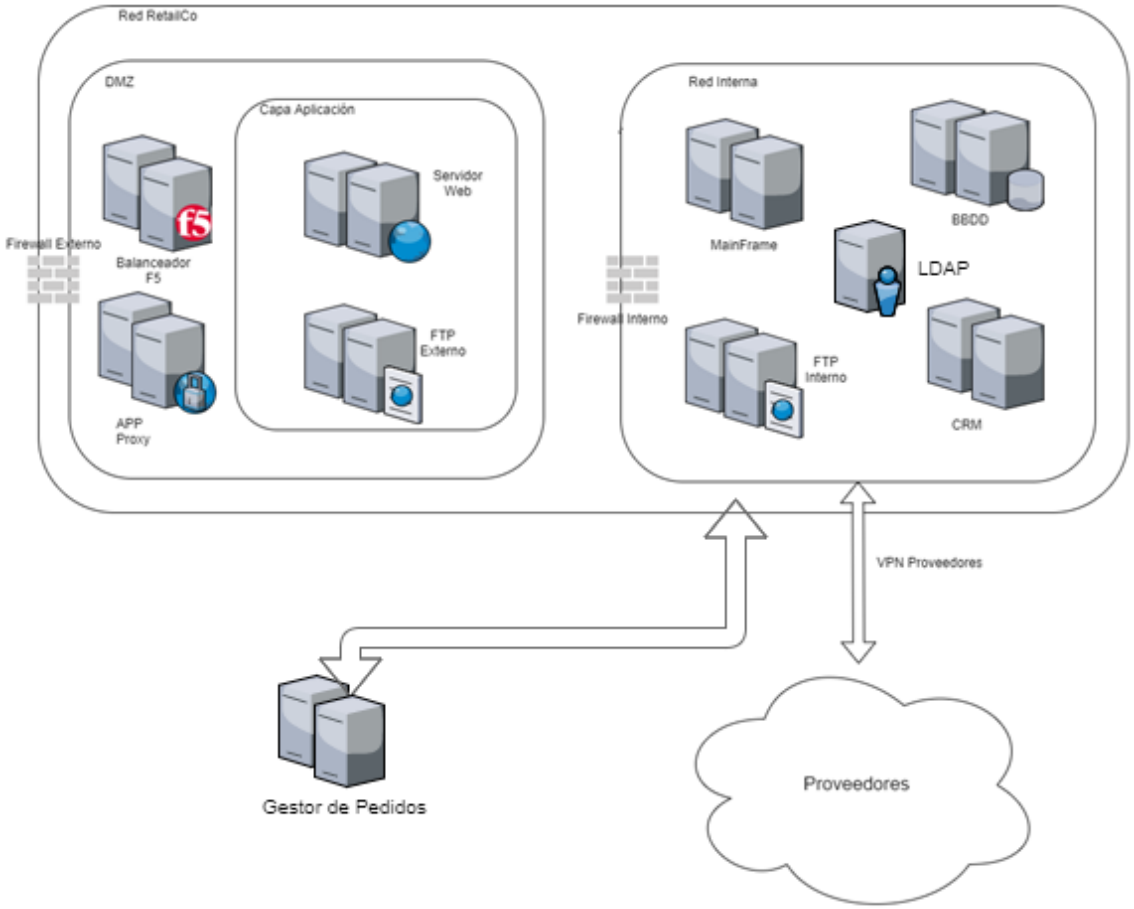


Imagen 118 – Mapa de Arquitectura de Sistemas

Sistema	Ubicación	Nº instancias	Función
Firewall Externo	DMZ	2	Contiene todas las reglas de acceso y salida a la red RetailCo.
APP Proxy	DMZ	2	Contiene las reglas
Balanceador	DMZ	2	Contiene los servicios balanceados del sistema para enviar a un nodo u otro en los sistemas que gozan de LB
Servidor Web	Capa Aplicación	2	Sistema que contiene las aplicaciones web de la empresa
FTP Externo	Capa Aplicación	2	Sistema para el traspaso de ficheros con los proveedores que no tienen VPN
Firewall Interno	Red Interna	2	Contiene todas las reglas de acceso a la red interna de RetailCo
FTP Interno	Red Interna	2	Sistema de traspaso de ficheros entre sistemas internos de RetailCO
MainFrame	Red Interna	2	Sistema Host de RetailCo, donde se encuentra toda la operative de la empresa
BBDD	Red Interna	2	Sistema DataWareHouse donde se persiste toda la información de la empresa
CRM	Red Interna	2	Sistema que gestiona todo el área de clientes de RetailCo
VPN proveedores	Comunicación con Red Interna	N/A	Sistema de comunicaciones con proveedores ajenos a RetailCo
LDAP	Red Interna	1	Directorio Activo con los usuarios de la compañía
Gestion de Pedidos	Externa	2	Sistema externo encargado de gestionar el ciclo de vida de los pedidos y su logística posterior.

Tabla 18 – Definición de sistemas

4.4 SITUACIÓN EN LA QUE SE ENCUENTRA

Tal y como se ha comentado en el punto 2.1, la empresa se encuentra en medio de un proceso de transformación digital. Se ha decidido apostar por las nuevas tecnologías, reforzando así su modelo de negocio eCommerce. Para ello, son conscientes que tienen que empezar por los cimientos de la compañía, es decir por dotar de disponibilidad y accesibilidad a sus sistemas Legacy.

Actualmente la empresa cuenta con una serie de sistemas monolíticos que no están desarrollados para la comunicación entre sí. Esto conlleva a la duplicidad de los datos en muchas ocasiones, debido a que es más sencillo y económico que estableces integraciones online entre ellos.

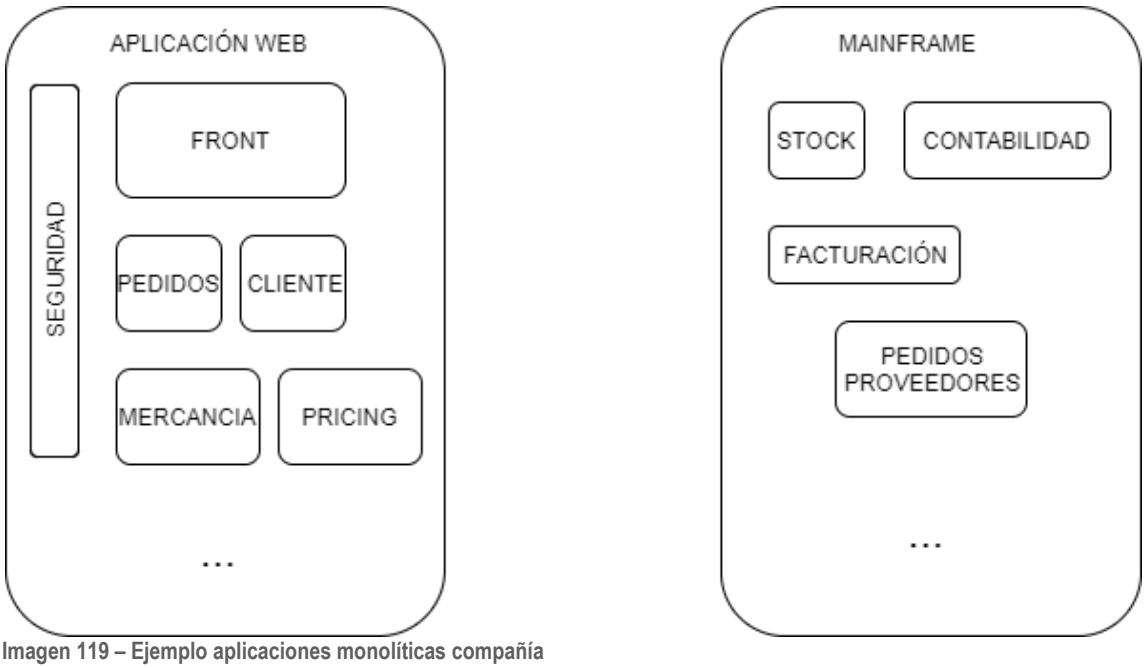


Imagen 119 – Ejemplo aplicaciones monolíticas compañía

La primera idea de la empresa fue la de sustituir todos estos sistemas monolíticos por sistemas desacoplados mediante nuevas tecnologías como puede ser una arquitectura orientada a microservicios.

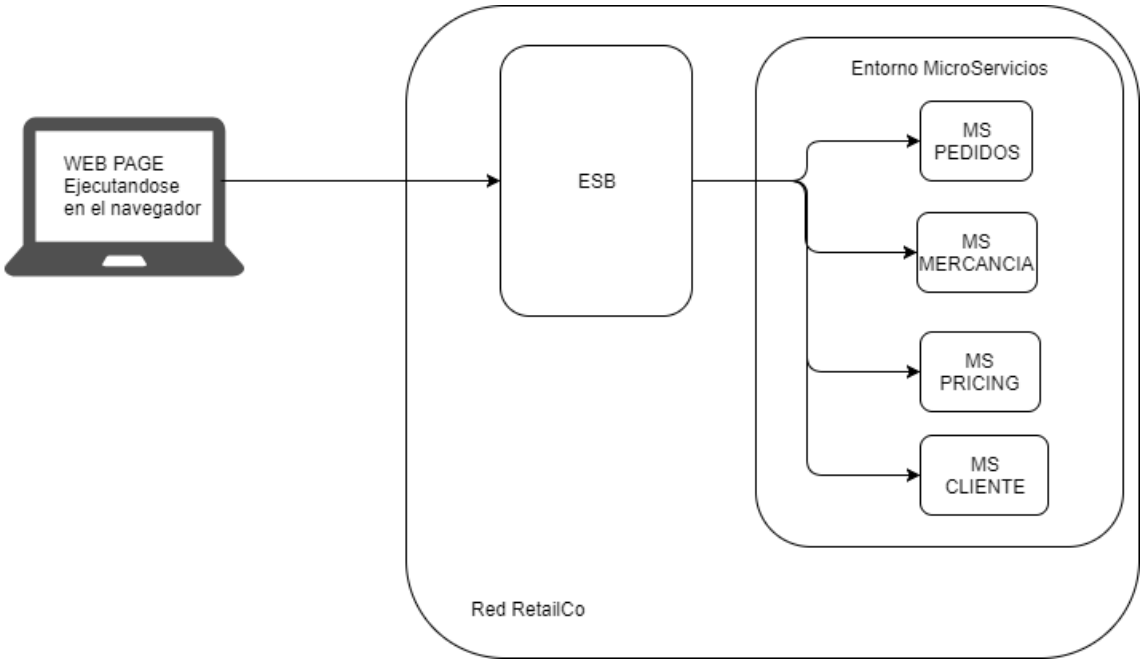


Imagen 120 – Ejemplo atomización aplicación web

El problema que esto conlleva, es que hay sistemas (Parte Mainframe) que no son sencillos de transformar. Estos sistemas tienen mucha lógica de negocio y llevan mucho tiempo implantados en la compañía. Además estos sistemas tienen un muy buen rendimiento en sus competencias.

Por lo que finalmente se decide apostar por un modelo hibrido, que se basa en sustituir aplicaciones monolíticas con poca funcionalidad de negocio en aplicaciones más desacopladas y mantener las aplicaciones con el core de negocio, eliminando las partes en las que la información se duplica añadiendo integraciones para estos datos.

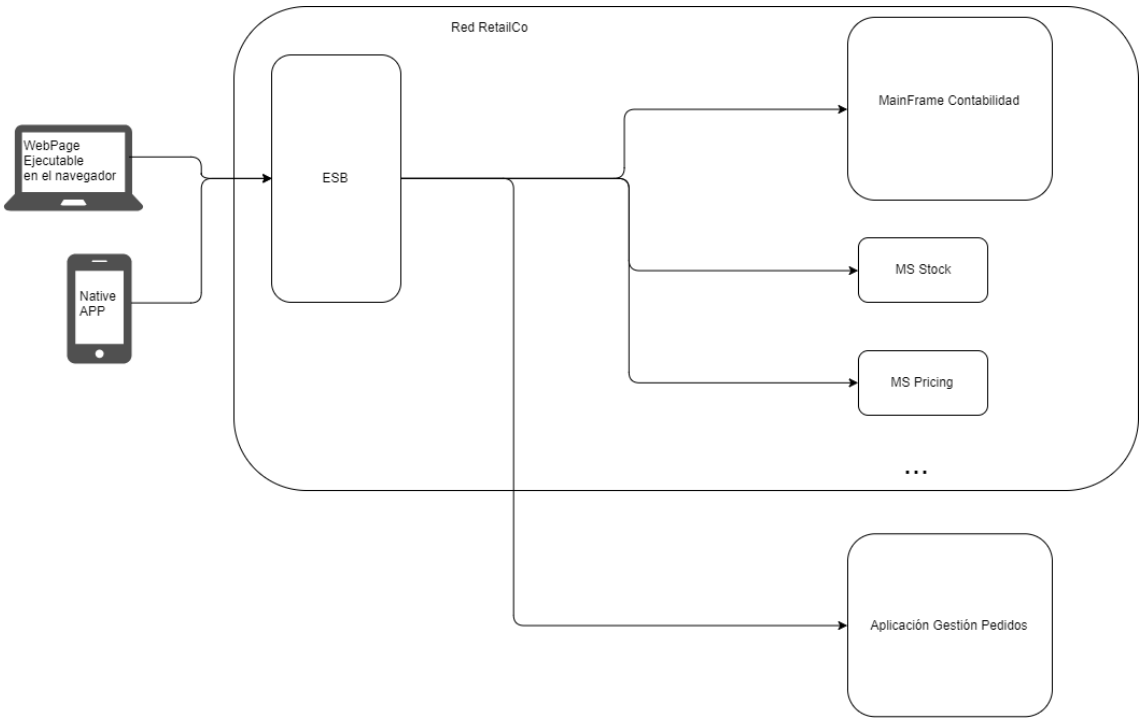


Imagen 121 - Ejemplo Modelo Hibrido de transformación

De este modo se decide implantar un ESB que se encargue de exponer interfaces de los diferentes datos de los que son maestros los diferentes sistemas.

4.5 ALCANCE

Como último punto de la contextualización del proyecto, durante este punto se definirá el alcance que tendrá el mismo.

4.5.1 Implantación de ESB

Este proyecto se va a focalizar en la implantación de un Bus de Servicio (ESB) dentro de la transformación digital de la empresa.
Este sistema se encargará de dirigir los flujos de negocio hacia los sistemas correspondientes, dotando a su vez a dichos flujos de una monitorización.

FRAMEWORK

Para ello se va a desarrollar un framework de trabajo en el que dejarán definidos una serie de patrones para llevar a cabo las integraciones.
Estos patrones serán una especie de plantillas sobre las que los equipos de desarrollo deberán trabajar en un futuro. Los patrones estarán formados a su vez por una serie de adaptadores que tendrán la función de conectar con los diferentes sistemas.

Los patrones iniciales que tendrán este framework son los siguientes.

PATRÓN SINCRONO
Patrón definido para integraciones puramente online, de consulta de entidades o de cualquier operación que necesite una respuesta en ese momento.

PATRÓN ASINCRONO
Patrón que no precisa de una respuesta inmediata, este tipo de patrones se usa para sistemas que no pueden depender del tiempo de respuesta de un tercer sistema, ya que son flujos críticos.

PATRÓN BATCH
Patrón pensado para carga de datos u operaciones que no tienen como clave el tiempo de procesamiento. Pensado para dejar ejecutándolo en un segundo plano o durante los periodos valle de los sistemas.

Los adaptadores que formarán estos patrones son los siguientes:

ADAPTADOR DE CANAL
Adaptador diseñado para la comunicación de entrada con los sistemas origen del flujo. Su diseño está pensado para recibir la petición de un sistema, procesarla y enviarla al siguiente adaptador del flujo.

ADAPTADOR DE IMPORTACIÓN
Adaptador de uso exclusivo en el patrón asíncrono. Está pensado para recibir una respuesta de forma no transaccional de una petición realizada en un flujo anterior.
ADAPTADOR DE ORQUESTACIÓN
Adaptador que tiene como función orquestar las diferentes llamadas a diferentes sistemas en el caso de procesos de negocio que lo requieran.

ADAPTADOR DE PROCESO SIMPLE
Adaptador diseñado para invocar a los sistemas destino del flujo. Este adaptador siempre es desencadenado por un adaptador de canal o un adaptador de orquestación.

ADAPTADOR 1-N
Adaptador que recibe un fichero a modo batch y lo convierte en n peticiones (por ejemplo por línea de fichero) al sistema destino.

ADAPTADOR N-1
Adaptador que recibe N peticiones y las agrupa en único fichero para mandarlo al sistema destino.

4.5.2 Suite TIBCO

Para llevar a cabo el ESB se va a utilizar una suite de herramientas del fabricante TIBCO.

TIBCO (The Information Bus Company) Software Inc. es una empresa estadounidense que ofrece software de integración, análisis y procesamiento de eventos para que las empresas lo utilicen en sus instalaciones o como parte de entornos de computación en la nube. El software maneja información, decisiones, procesos y aplicaciones para más de 10,000 clientes. Tiene sede en Palo Alto, California, y oficinas en América del Norte, Europa, Asia, Medio Oriente, África y América del Sur.
Entre sus clientes se incluyen AirFrance / KLM, Citi, Conway, ING, Marks y Spencer, Nielsen, Shell, Universidad de Medicina de Chicago, Western Union… [09]

Dentro de esta suite de herramientas de integración se han utilizado las siguientes:

TIBCO ACTIVEMATRIX BW 5.13:
TIBCO Business Works es una plataforma robusta que puede ser utilizada para desarrollar nuevos servicios, automatizar los procesos de negocio, y la integración de aplicaciones.

TIBCO EMS 8.3:
TIBCO Enterprise message Service Sistema de comunicación basado en JMS(Java Message Service), que se utiliza dentro del ecosistema TIBCO como medio de comunicación basado en queues y topics.

TIBCO TRA 5.10

TIBCO Runtime Agent es un paquete de software TIBCO y software de terceros (JAVA) que se necesita para ejecutar muchas aplicaciones TIBCO, como TIBCO ActiveMatrix BusinessWorks ™ y TIBCO Adapters. [10]

TIBCO ADMINISTRATOR

TIBCO Administrator es la herramienta de gestión y administración de los procesos TIBCO. Con esta herramienta se gestionará el despliegue, tuneado y organización de los procesos.

TIBCO RV 8

TIBCO Rendezvous es el sistema de mensajería nato de TIBCO, utilizándolo en todas sus comunicaciones internas.

TIBCO DESIGNER 5.10

TIBCO Designer es el IDE utilizado para programar los procesos. Este proporciona una interfaz que ayuda a realizar una programación intuitiva basada en una interfaz gráfica. Transformando las diferentes acciones en “paletas”. De este modo se pueden definir los flujos arrastrando las diferentes paletas y conectándolas entre ellas.

5 PASOS FUTUROS

En este apartado se comentarán los posibles futuros pasos o mejoras que podrían aplicarse al proyecto.

5.1 MONITORIZACIÓN TÉCNICA Y FUNCIONAL

Durante toda la memoria se ha comentado que el Framework genera evento que pueden ser enviados a 3os sistemas para tener toda la traza de los flujos de negocio, por ejemplo en un sistema de monitorización. Durante este apartado se analizará un posible proyecto futuro que podría completar el Framework del ESB definido.

5.1.1 Que aporta la monitorización

La monitorización es una pata clave dentro de una organización empresarial. Esta proporcionará información exacta del estado de los procesos y a su vez aportará información clave al negocio de la compañía. De este modo la monitorización en una compañía se puede dividir en dos secciones:

- Monitorización técnica:
  - Encargada de mostrar el estado de los procesos, los nodos y el cluster que está dando servicio a la compañía. Suele estar acompañada de un alarmado mediante el envío de eventos para que, en el caso de que algo no vaya según los parámetros establecidos de normalidad, el equipo responsable sea informado de forma online para poder proceder con su revisión de inmediato.

- Monitorización Funcional:
  - Esta monitorización estará definida por el negocio y les dotará de la información necesaria para poder realizar reportes de forma online sobre el estado de las funcionalidades de la empresa. Suele estar basada en los diferentes KPI's (key performance indicator) que establezca la compañía.

5.1.2 Suite ELK

Como parte del proyecto se pretende montar una infraestructura de monitorización sobre el ESB, de forma que sea capaz de enviar eventos del flujo a esta nueva infraestrucutura para poder realizar monitorizaciones tanto técnicas como funcionales. Para ellos se ha optado por la suite de ELK (ElasticSearch/LogStash/Kibana).

Esta suite está formada por tres componentes de la empresa *"Elastic"* y juntos forman el Elastic Stack. Cada uno de los compontes tiene una función específica que se describe a continuación:

ELASTICSEARCH:

Es una potente herramienta que nos permite indexar un gran volumen de datos y posteriormente hacer consultas sobre ellos soportando entre otras muchas cosas búsquedas aproximadas, facetas y resaltadas. Un uso puede ser hacer consultas de texto completo, al estar los datos indexados los resultados se obtienen de forma muy rápida. [11]

LOGSTASH

Logstash es una herramienta para la administración de logs. Esta herramienta se puede utilizar para recolectar, parsear y guardar los logs para futuras búsquedas.2 La aplicación se encuentra basada en jRuby y requiere de Java Virtual Machine para correr. Como corre en JVM puede ser ejecutada en cualquier Sistema Operativo que corra JVM (Linux, Mac OS X, Windows). [12]

KIBANA

Kibana es una herramienta open-source perteneciente a Elastic, que nos permite visualizar y explorar datos que se encuentran indexados en ElasticSearch, es decir, un plugin de ElasticSearch.

Imagen 122 – Ejemplo de monitorización en Kibana.

6 CONCLUSIONES

Durante este apartado se tratará de reflexionar acerca de las conclusiones extraídas en relación al trabajo de análisis y diseño del framework para el ESB.
Se irán enumerando y exponiendo a continuación 1 a 1.

6.1 AHORRO EN EL DESARROLLO

El ahorro en el desarrollo, es posiblemente la medida más llamativa de cara al mundo empresarial. Tras el análisis y el diseño realizado se ha podido observar que este framework reduce considerablemente las horas de desarrollo necesarias para llevar a cabo una integración.
Este ahorro es debido a que el desarrollador tan solo se debe de preocupar de implementar las piezas que contienen definiciones del negocio:

- Mapeos.
- Selección de entidades.
- Enrutamientos.
- Llamadas al Backend.

Esto se logra gracias a la creación de piezas genéricas para las partes del proceso puramente técnicas las cuales, son contenidas en una librería que deberá ser importada al inicio de cada proyecto. Esto es enriquecido a su vez con una serie de plantillas que dan la estructura hecha al desarrollador.

A continuación se mostrará una simulación de horas de desarrollo con y sin el framework.
Para realizar esta simulación se ha simplificado la estimación de horas de la siguiente forma:

Desarrollo de una pieza de integración= 2 horas.

El adaptador que se usará para este ejemplo es el adaptador del proceso simple.

Desarrollo sin framework:
Número de piezas a desarrollar: 4
Piezas a desarrollar:
Map_Request
Map_Response
Map_Error
Backend
Total de horas: 8h

Desarrollo con framework:
Número de piezas a desarrollar: 12
Piezas a desarrollar:

- Map_Request
- Map_Response
- Map_Error
- Backend
- Simple_Service_Pattern
- Preactions_IN
- Postactions_IN
- Preactions_OUT
- Postactions_IN
- GetServiceId
- HeaderManagement
- SendEvent

Horas de desarrollo: 24

Como se puede observer en un solo adaptador hay una diferencia de 16 horas.

Teniendo en cuenta que un flujo de integración al menos cuenta con dos adaptadores la diferencia sería de 32 horas por flujo.

Suponiendo que un proyecto de integración tiene unos 4 flujos la diferencia por proyecto de integración es de 128 horas.

Nivel	Horas sin FW	Horas con FW	Diferencia
Adaptador	24	8	16
Flujo	48	16	32
Proyecto	192	64	128

Tabla 19 - Diferencia de horas de programación

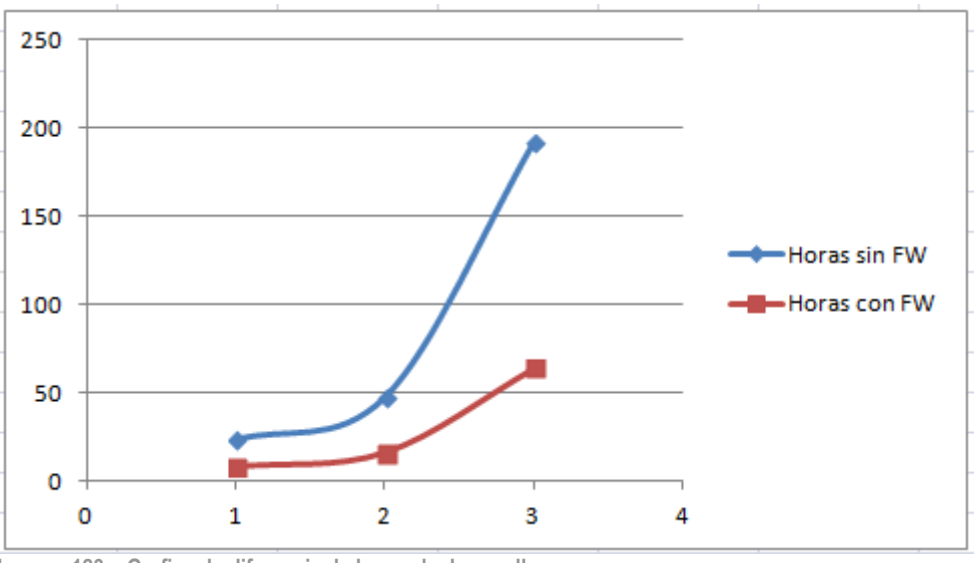


Imagen 123 - Grafica de diferencia de horas de desarrollo

Con este sencillo ejemplo se puede observar el crecimiento exponencial de las horas de desarrollo al no usar el framework.

6.2 HOMOGENEIDAD EN EL DESARROLLO

Otro aspecto importante que ha quedado demostrado tras el diseño y análisis del framework ESB es el apartado de la Homogeneidad.

Gracias al uso de este framework tan guiado, los desarrolladores tan solo tienen que utilizar los patrones definidos mediante las plantillas y librerías. Esto ayuda a que todos los proyectos de integración dentro del ESB sigan las mismas pautas, siendo mucho más fácil su mantenimiento y soporte.

Este punto es muy importante sobre todo en grandes empresas, en las que suelen haber varios equipos muchas veces de distintas empresas desarrollando procesos de integración. En este escenario, sin utilizar el framework definido, sería imposible obtener una homogeneidad, dando soluciones muy diferentes para casos de negocio idénticos y desarrollando distintas piezas. Haciendo muy costoso el soporte y mantenimiento de los proyectos.

Durante el punto de diseño de esta memoria, se ha hecho hincapié en la estructura de los proyectos, la nomenclatura y el uso correcto de los patrones con sus procesos de fw. Siguiendo estas directrices, dos procesos de negocio que son cubiertos con el mismo patrón son prácticamente idénticos, a excepción de los procesos de negocio.

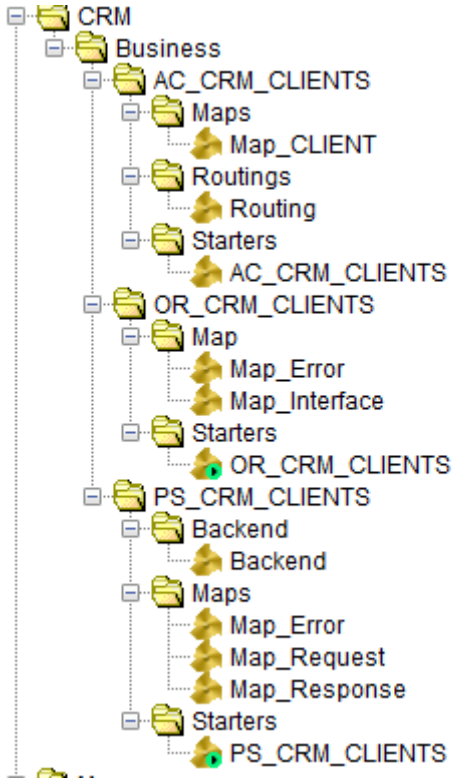


Imagen 124 – Ejemplo de estructuras de ejemplo de adaptadores correspondientes al patrón síncrono.

En la imagen se ve claramente la homogeneidad de los diferentes adaptadores.

6.3 REUTILIZACIÓN DEL CÓDIGO

Como se ha podido observar durante toda la memoria, la reutilización del código es una de las claves de este framework.
La creación de procesos que se usan en todos los adaptadores de todos los proyectos de integración es una de las claves del framework.
De hecho la gran parte de los adaptadores usan muchos procesos comunes como pueden ser:

- Generación del UUID.
- Gestión de la cabecera de arquitectura.
- Gestión de trazas.
- Envío de eventos.

Además gracias a la utilización de la librería y las plantillas, la utilización de estos componentes comunes es completamente transparente al desarrollador.
Esto hace que los desarrolladores no necesiten tener grandes conocimientos técnicos ya que tan solo se tienen que centrar en la parte funcional.

7 REFERENCIAS

Número de referencia	Fuente	Descripción
[01]	UST-GLOBAL. (s.f.)	http://www.tcpsi.com
[02]	WIKIPEDIA. (s.f.)	https://en.wikipedia.org/wiki/MuleSoft
[03]	WIKIPEDIA. (s.f.)	https://es.wikipedia.org/wiki/Apache_Camel
[04]	WIKIPEDIA. (s.f.)	https://en.wikipedia.org/wiki/Software_AG
[05]	WIKIPEDIA. (s.f.)	https://en.wikipedia.org/wiki/IBM_Integration_Bus
[06]	WIKIPEDIA. (s.f.)	https://es.wikipedia.org/wiki/Identificador_%C3%BAnico_universal
[07]	http://programacion.jias.es. (n.d.)	http://programacion.jias.es
[08]	WIKIPEDIA. (n.d.)	https://es.wikipedia.org/wiki/Servicio_web
[08]	WIKIPEDIA. (n.d.)	https://en.wikipedia.org/wiki/TIBCO_Software
[10]	TIBCO. (s.f.)	https://docs.tibco.com/products/tibco-runtime-agent-5-10-0
[11]	Elastic. (n.d.)	https://picodotdev.github.io/blog-bitix/2014/04/introduccion-a-elasticsearch/
[12]	WIKIPEDIA. (n.d.)	Retrieved from https://es.wikipedia.org/wiki/Logstash