Artículo
Pierre-Yves Duq... · 16 jun, 2020 Lectura de 26 min

Implementar Integraciones IRIS con .NET o Java usando PEX

Introducción

InterSystems IRIS 2020.1 incluye PEX (Production EXtension Framework), para facilitar el desarrollo de producciones de Interoperabilidad de IRIS con componentes escritos en Java o .NET.

Gracias a PEX, un desarrollador de integraciones con conocimientos Java o .NET puede beneficiarse de la potencia, escalabilidad y robustez del framework de Interoperabilidad de InterSystems IRIS, y ser productivo en muy poco tiempo.

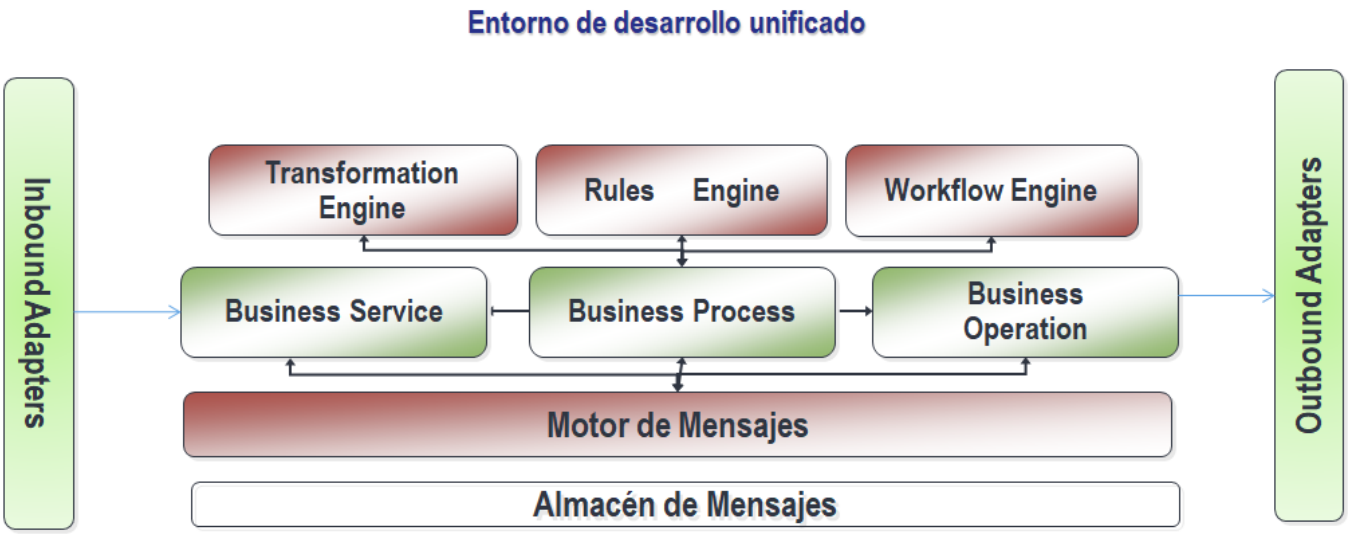
Para un experto en el framework de Interoperabilidad IRIS, PEX aporta facilidad para enriquecer las integraciones con componentes externos pre-existentes en lenguajes Java o .NET.

Este tutorial propone explorar PEX y guiar el desarrollador .NET en sus primeros pasos con PEX. El código esta disponible en https://github.com/es-comunidad-intersystems/webinar-PE

El tutorial complementa la documentación de PEX disponible en linea en https://docs.intersystems.com/irislatest/csp/docbook/DocBook.UI.Page.cls?KEY=EPEX

Arquitectura de Interoperabilidad InterSystems IRIS

PEX permite escoger el lenguaje de programación (actualmente Java o .NET, más adelante también python) para implementar cualquier de los distintos componentes (Business Service, Business Process, Business Operation, InboundAdapter, OutboundAdapter, Mensajes) que forman una producción de Interoperabilidad de IRIS.



Los 3 componentes principales de interoperabilidad de IRIS son:

- Business Services: Componentes que ofrecen un servicio que se puede invocar desde fuera. Ejemplo de Business Services son un Servicio REST, un servicio SOAP Web Service, pero también un Service que lee y procesa los nuevos ficheros que han sido escritos en un directorio, un servicio que lee y procesa filas de una tabla de BBDD, un servicio que lee ficheros por FTP, etc. El Business Service puede tener un Inbound Adapter asociado, que se encarga de los detalles de implementación de un protocolo especifico (SQLInboundAdapter para leer tablas SQL, FileInboundAdapter para leer ficheros, etc. El Business Service se encarga de procesar la información, copiarla a un mensaje de Interoperabilidad de IRIS, y enviar el mensaje a un Business Process o Business Operations y posiblemente esperar una respuesta si hay.
- Business Operation: Componente que recibiendo un mensaje de interoperabilidad de IRIS, actúa sobre un sistema externo, posiblemente con la ayuda de un OutboundAdapter que se encarga de los detalles de implementación de protocolos (TCP, REST, SOAP,Ficheros, SQL, etc). El Business Operation puede o no devolver una respuesta a quien le haya llamado.
- Business Process: proceso de orquestación, que, recibiendo un mensaje de Interoperabilidad de IRIS, puede hacer procesamientos y llamadas a uno o varios otros business processes o business operations para realizar operaciones complejas y/o devolver un mensaje con información agregada al Business Service que le he enviado el mensaje.

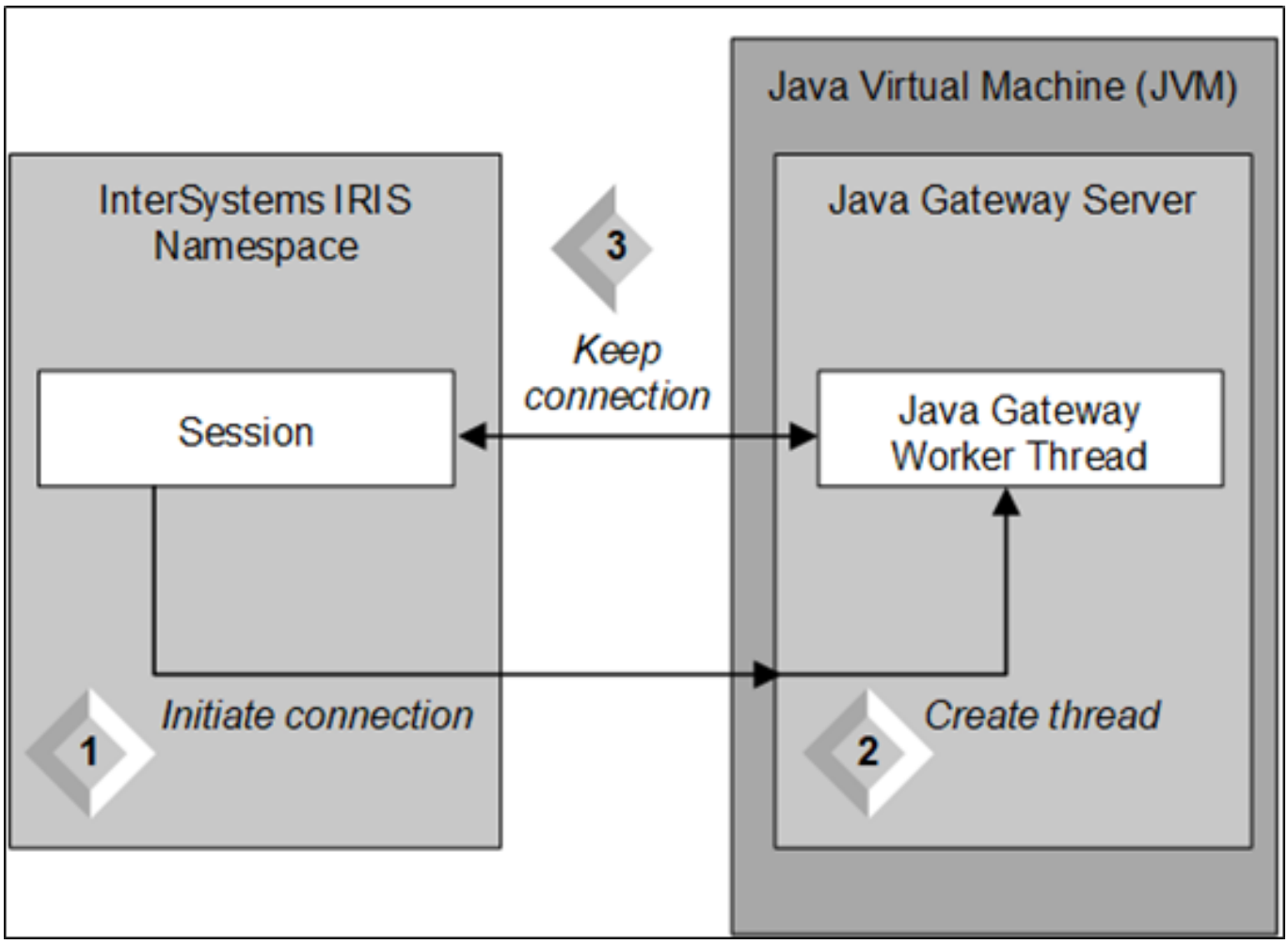
Toda la configuración de los componentes se agrupa en una "Producción"; la producción es una clase de InterSystems IRIS que contiene la definición de todas las integraciones que se arrancan juntas al iniciar IRIS, y la configuración de cada componente de estas integraciones. Se puede editar como una clase de IRIS, o modificar desde el portal de gestión.

Arquitectura de PEX

Para ejecutar el código .NET o Java, InterSystems IRIS usa el Object Gateway correspondiente. Este Gateway se puede instanciar localmente o en un servidor remoto.

Los Java y .Net Gateways

El Gateway es un componente nativo de .NET o Java, que escucha en un puerto TCP/IP concreto y recibe peticiones desde el proceso IRIS, las ejecuta y devuelve los resultados.



Los Componentes PEX pre-construidos

Para usar PEX en una producción de IRIS, es necesario añadir a la Producción un componente pre-construido para cada componente desarrollado en Java o .NET. El componente pre-construido sirve de envoltorio (wrapper) al elemento externo que referencia, y permite definir sus propiedades y configuración dentro de la producción.

Los componentes Java o .NET que se ejecutan desde el Gateway correspondiente deben heredar (ser subclase) de un componente de PEX pre-existente. A continuación se ofrece una recopilación de estos elementos:

Componentes de IRIS

Para .NET y para Java

Función	Clase	Comentario
Business Service	EnsLib.PEX.BusinessService	Adaptador configurado: Ens.InboundAdapter
Business Process	EnsLib.PEX.BusinessProcess.	
Business Operation	EnsLib.PEX.BusinessOperation	
Inbound Adapter	EnsLib.PEX.InboundAdapter	
Outbound Adapter	EnsLib.PEX.OutboundAdapter	
PEX Message	EnsLib.PEX.Message	Mensaje para enviar a un componente PEX
IRIS Message	Se Mapea al Mensaje IRIS	Mensaje de componente PEX a componente IRIS

Componentes Java o .NET

Estos componentes están disponibles en librerías que hay que añadir al proyecto java o .NET. La librería mínima a referenciar en un proyecto PEX es la del Gateway. Adicionalmente, si se quiere usar mensajes IRISObject para llamar a componentes IRIS desde PEX, es necesario referenciar la librería IRISClient:

Lenguaje	Librerías
.NET	<installdir> \dev \dotnet \bin \v4.5 \InterSystems.Data.Gateway64.exe <installdir> \dev \dotnet \bin \v4.5 \InterSystems.Data.IRISClient.dll
Java	<installdir> \dev \java \lib \JDK18 \intersystems-gateway-3.1.0.jar <installdir> \dev \java \lib \JDK18 \intersystems-jdbc-3.1.0.jar <installdir> \dev \java \lib \gson \gson-2.8.5.jar

SuperClase a usar para componentes .NET

Función	Clase .NET
Business Service	InterSystems.EnsLib.PEX.BusinessService
Business Process	InterSystems.EnsLib.PEX.BusinessProcess
Business Operation	InterSystems.EnsLib.PEX.BusinessOperation
Inbound Adapter	InterSystems.EnsLib.PEX.InboundAdapter
Outbound Adapter	InterSystems.EnsLib.PEX.OutboundAdapter
Mensaje PEX	InterSystems.EnsLib.PEX.Message
Mensaje IRIS	InterSystems.Data.IRISClient.ADO.IRISObject

Superclase a usar para componentes en Java

Función	Clase Java
Business Service	com.intersystems.enslib.BusinessService
Business Process	com.intersystems.enslib.BusinessProcess
Business Operation	com.intersystems.enslib.BusinessOperation.
Inbound Adapter	com.intersystems.enslib.pex.InboundAdapter
Outbound Adapter	com.intersystems.enslib.pex.OutboundAdapter
Mensaje PEX	com.intersystems.enslib.pex.Message
Mensaje IRIS	<...>.IRISObject

Funciones de Utilidades

En los componentes implementados en ObjectScript, InterSystems IRIS ofrece unos comandos en forma de MACRO para añadir información al log de Eventos de IRIS. Estos métodos están disponibles en JAVA y .NET en

la forma siguiente:

Método	Descripción
LOGINFO(mensaje)	Añadir mensaje al Log de Eventos con estado "info"
LOGALERT(mensaje)	Añadir mensaje al Log de Eventos con estado "alert"
LOGWARNING(mensaje)	Añadir mensaje al Log de Eventos con estado "warning"
LOGERROR(mensaje)	Añadir mensaje al Log de Eventos con estado "error"
LOGASSERT(mensaje)	Añadir mensaje al log de Eventos con estado "assert"

Interoperabilidad entre componentes nativos y componentes PEX

Es posible combinar componentes en lenguaje nativo ObjectScript de InterSystems IRIS con componentes PEX desarrollados en Java o .NET.

- Cuando los 2 componentes son de tipo BusinessService - InboundAdapter o BusinessOperation - OutboundAdapter el desarrollador puede escoger el tipo de datos/tipo de objeto a usar en las llamadas: La llamadas se realizan entre IRIS y el Java/.NET gateway siguiendo las reglas de conversión de tipos de datos que determina el Gateway.
- Cuando los 2 componentes que intercambian información son Business Hosts (BS,BP, BO), el componente que debe recibir el mensaje (de petición o de respuesta) impone el tipo de objetos a usar para el mensaje:
    - Un Componente PEX siempre recibe un mensaje EnsLib.PEX.Message
    - Un Componente IRIS recibe un mensaje IRISObject

EnsLib.PEX.Message

El EnsLib.PEX.Message permite una representación en IRIS ObjectScript de un mensaje cuya estructura has sido definida en .NET o en Java. IRIS usa la funcionalidad de DynamicObject para manipular las propiedades. Internamente, IRIS usa JSON como medio de transporte entre IRIS y .NET/Java. Al crear un mensaje PEX en IRIS, es necesario informar en %classname el nombre de la clase de NET/Java que se debe instanciar para usar el mensaje.

IRISObject

Para poder llamar a un componente pre-construido de InterSystems IRIS desde un componente PEX, es necesario definir un mensaje que sea de tipo IRISObject (en .NET la clase completa es InterSystems.Data.IRISClient.ADO.IRISObject).

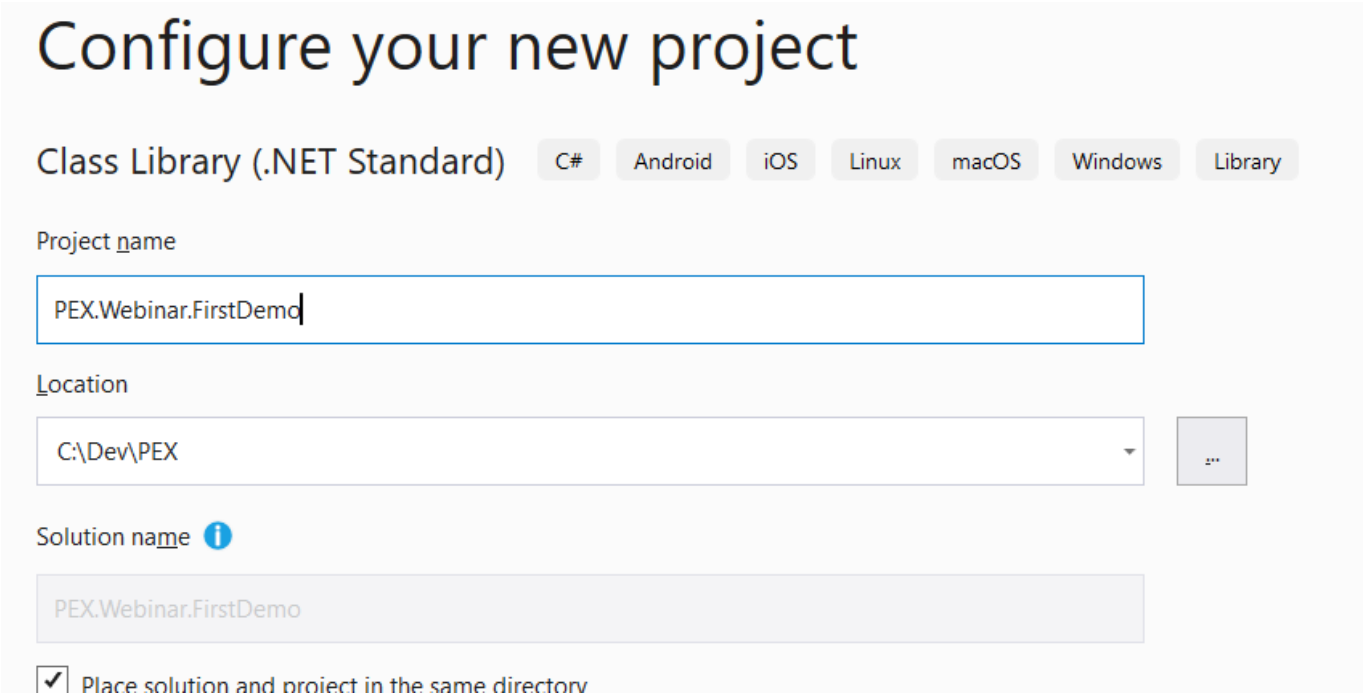
Primeros Pasos con PEX y .NET

En estos primeros pasos se procede a:

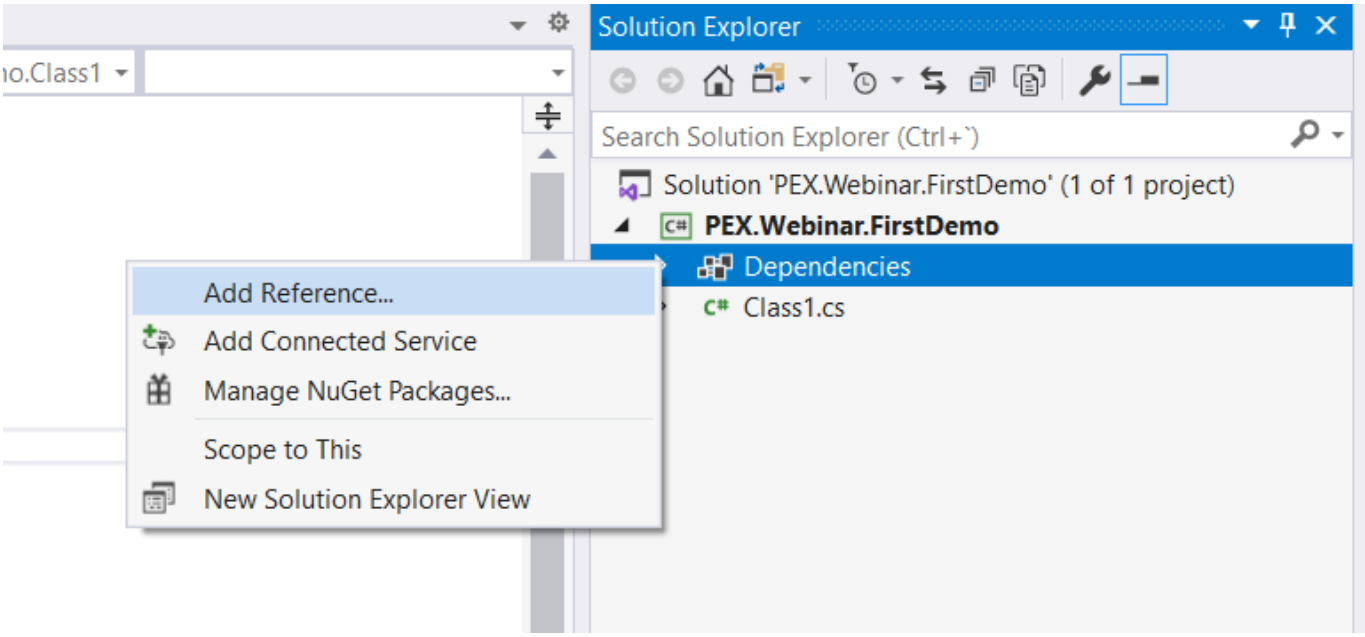
- crear un proyecto .NET con las librerias PEX necesarias, de la version 4.5 de .NET framework
- Añadir un BusinessOperation y un mensaje Simple al proyecto .NET
- Configurar el .NET Gateway de Intersystems IRIS
- Crear una Producción de Interoperabilidad
- Añadir un BO preconstruido

Creación de un Proyecto .NET con Visual Studio 2019

En Visual Studio, se crea un nuevo proyecto de tipo "Class Library .Net Standard in C#". Se escoge el nombre "PEX.Webinar.FirstDemo":



Para poder trabajar con PEX, se añaden las dependencias necesarias, desde el "Solution Explorer", menú contextual "Add Reference":



y con el botón "Browse", se añaden las 2 librerías (compatibles con .NET Framework 4.5!) ubicadas en un subdirectorio de la instalación de InterSystems IRIS:

```
<installdir>\dev\dotnet\bin\v4.5\InterSystems.Data.Gateway64.exe

<installdir>\dev\dotnet\bin\v4.5\InterSystems.Data.IRISClient.dll
```

A continuación, se renombra desde el solution Explorer Class1.cs para llamarlo "FirstOperation.cs", y se modifica la clase para que herede de la clase PEX de BusinessOperation (InterSystems.EnsLib.PEX.BusinessOperation). Se sobrescriben los 3 métodos de PEX.BusinessOperation:

Metodo	Descripción
OnInit	Se ejecuta 1 vez cuando se arranca este componente en la producción. Permite inicializar librerías, conexiones, variables...
OnTearDown	Se ejecuta 1 vez cuando se para este componente o la producción. Permite liberar recursos que se hayan usado en el ciclo de vida del componente
OnMessage	Se ejecuta para cada Mensaje recibido. Permite tratar el mensaje y devolver respuesta

De momento, no se ha definido ningún mensaje, y ninguna tarea que realizar. Por lo cual, se añade simplemente funciones LOGINFO en los 3 métodos. Por otra parte, no hace falta llamar al método de la superclase , por lo cual se puede eliminar las llamadas a la clase base (base.OnInit(), base.OnTearDown(), base.OnMessage). Dejamos la implementación como sigue:

Metodo	Implementación Inicial

OnInit	**OnInit**
	`public override void OnInit()`
	`{`
	`LOGINFO("PEX.Webinar.FirstDemo.FirstOperation:OnInit()");`
	`}`

OnTearDown	**OnTearDown**
	public override void OnTearDown()
	{
	LOGINFO("PEX.Webinar.FirstDemo.FirstOperation:OnTearDown()");
	}

OnMessage	**OnMessage** public override object OnMessage(object request) { LOGINFO("PEX.Webinar.FirstDemo.FirstOperation:OnMessage()"); return request;

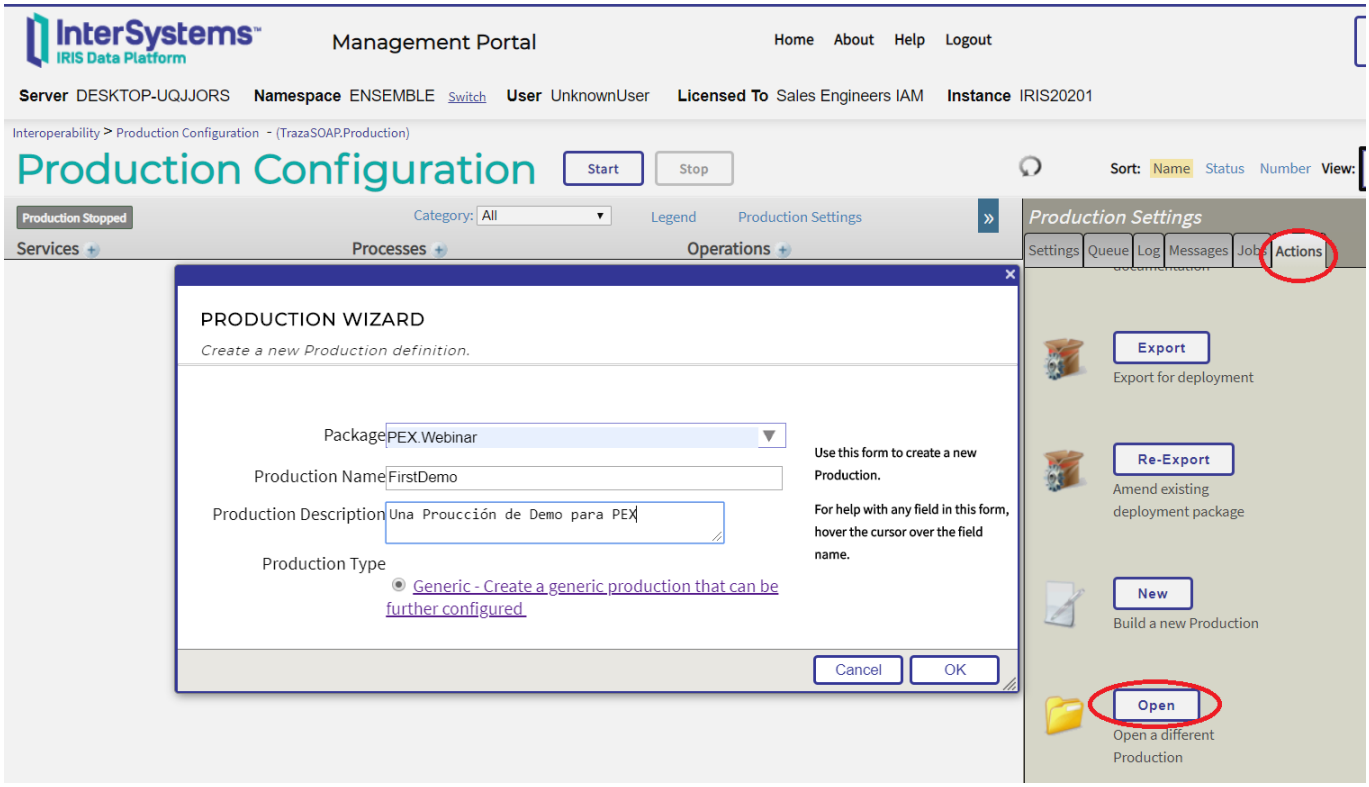
Metodo	Implementación Inicial
	}

Con esto, se puede compilar esta versión Inicial del proyecto .NET, con el Menú "Build Solution". Esto genera el fichero siguiente, que se utiliza a continuación desde IRIS:

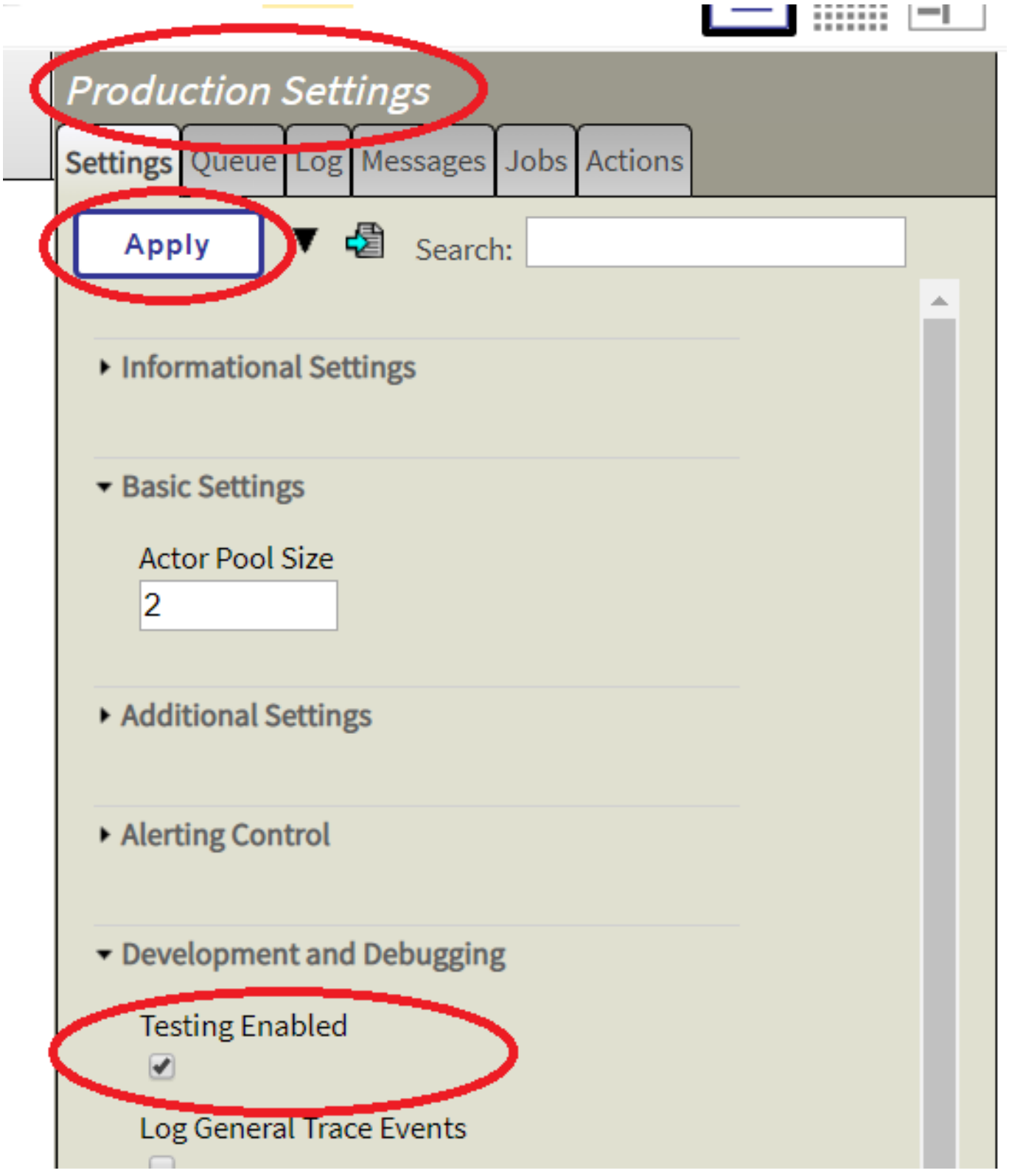
```
1>------ Build started: Project: PEX.Webinar.FirstDemo, Configuration: Debug Any CPU ------
1>PEX.Webinar.FirstDemo ->
C:\Dev\PEX\PEX.Webinar.FirstDemo\bin\Debug\netstandard2.0\PEX.Webinar.FirstDemo.dll
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Creación de una Producción de Interoperabilidad InterSystems IRIS

Ahora, desde el Portal de Gestión de IRIS, se puede escoger el menú "Interoperability", el namespace "Ensemble", y los menus "Configure", "Production". La pestaña "Actions", y el botón "New" permiten definir una nueva producción de Integración de IRIS.



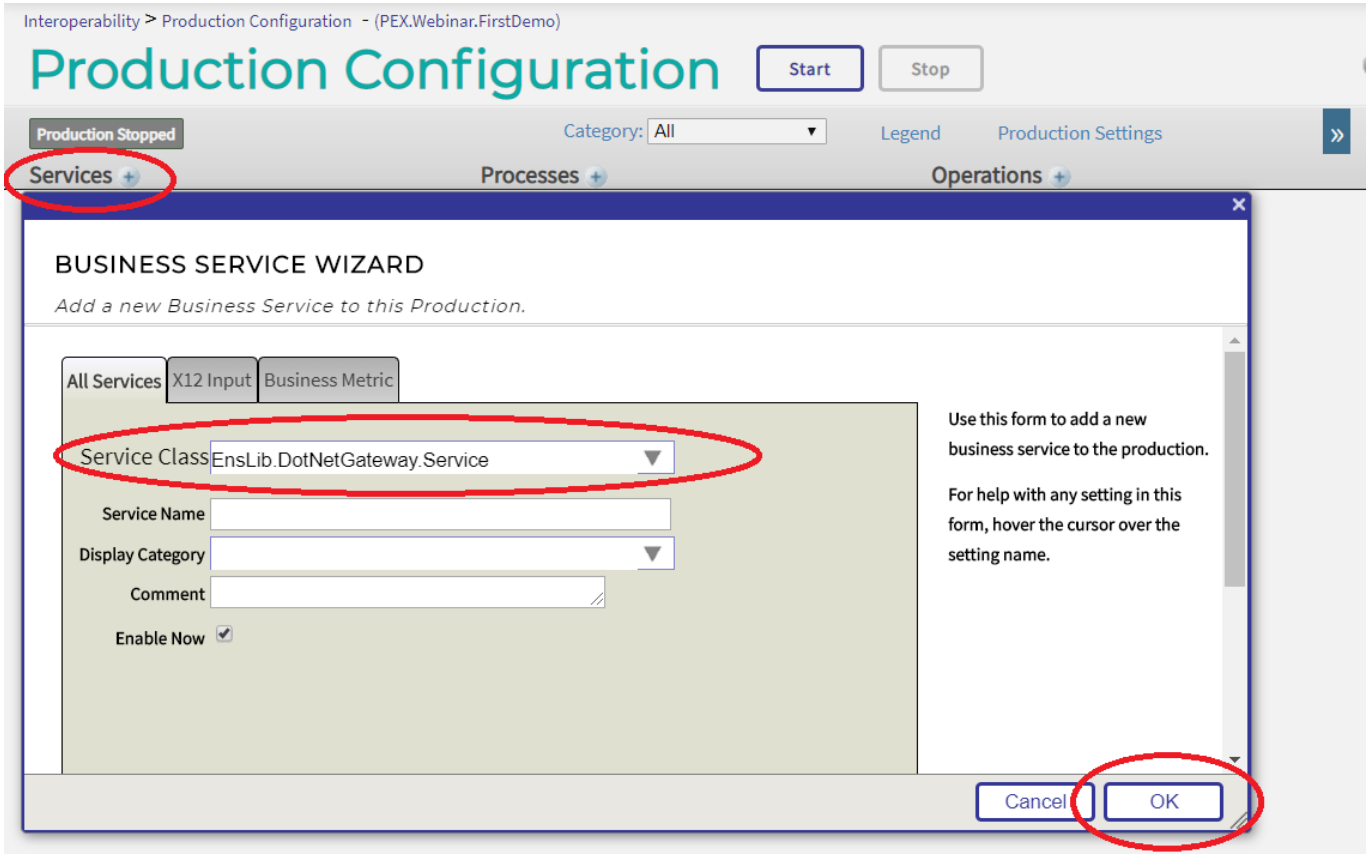
Para activar el Servicio de Pruebas, se selecciona la pestaña "Settings" y en el último apartado de la lista ("Development and Debugging") se habilita "Testing Enabled" y se hace click sobre el botón Apply:



Añadir el NET Gateway a la Producción

Para poder trabajar con PEX, se debe arrancar el Gateway de Java o .NET correspondiente. InterSystems recomienda configurar y arrancar estos gateway desde el menú "System Administration / Configuration / Connectivity / Object Gateways" par entornos de producción (live). Para este entorno de desarrollo, se puede añadir directamente un componente a la producción para arrancar el Gateway. Esto permite arrancar y parar el gateway al mismo tiempo que la producción en un solo click, y así liberar el acceso a la .DLL de proyecto de .NET cuando se quiere re- compilarla (Visual Studio no la puede recompilar si el .NET gateway está arrancado).

Para añadir el componente ".Net Gateway", se añade un Business Service a la producción con el botón "+" al lado de "Services". El nombre de clase del componente (Service Class) a añadir es "EnsLib.DotNetGateway.Service", y se selecciona "Enable now" para habilitar el componente.

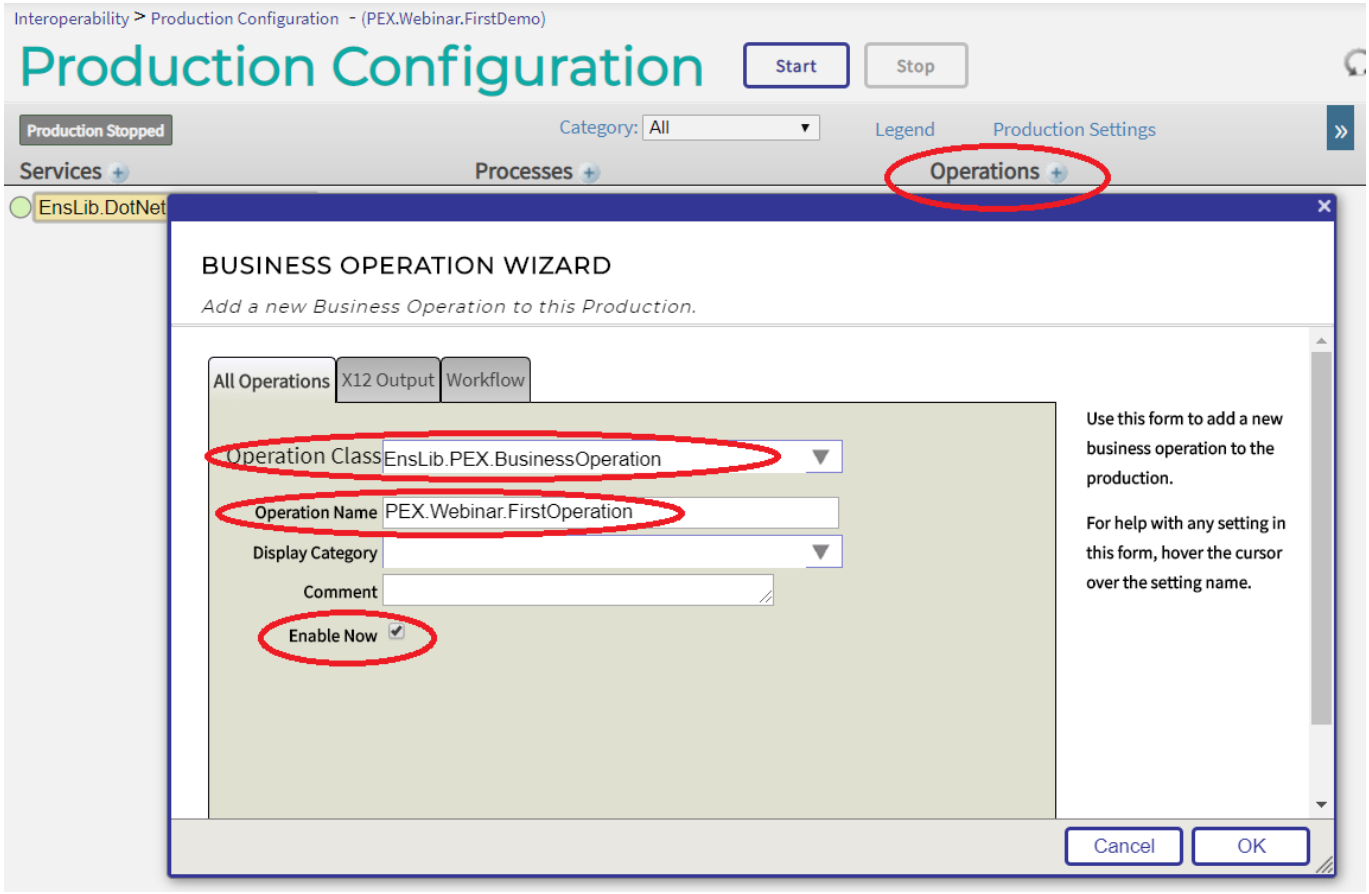


Se editan los parámetros de configuración del componente añadido, haciendo click sobre este, y rellenando a continuación los valores en la pestaña "Settings", seleccionado finalmente el botón "Apply":

Parametro	Valor	Descripción
Port	44444	Se puede cambiar al puerto si el puerto 55000 por defecto ya esta ocupado por otro proceso
FilePath	<installdir>\dev\dotnet\bin\v4.5\	Indica la ubicación donde encontrar el ejecutable del Gateway (InterSystems.Data.Gateway64.exe) <InstallDir> es el directorio de instalación de IRIS: Por ejemplo: C:\InterSystems\IRIS20201\dev\dotnet\bin\v4.5\
Exec64	true	Seleccionar la version 64Bits del gateway
.NET Version	4.5	La versión de .Net debe ser 4.5

Añadir el Business Operation creado en .NET

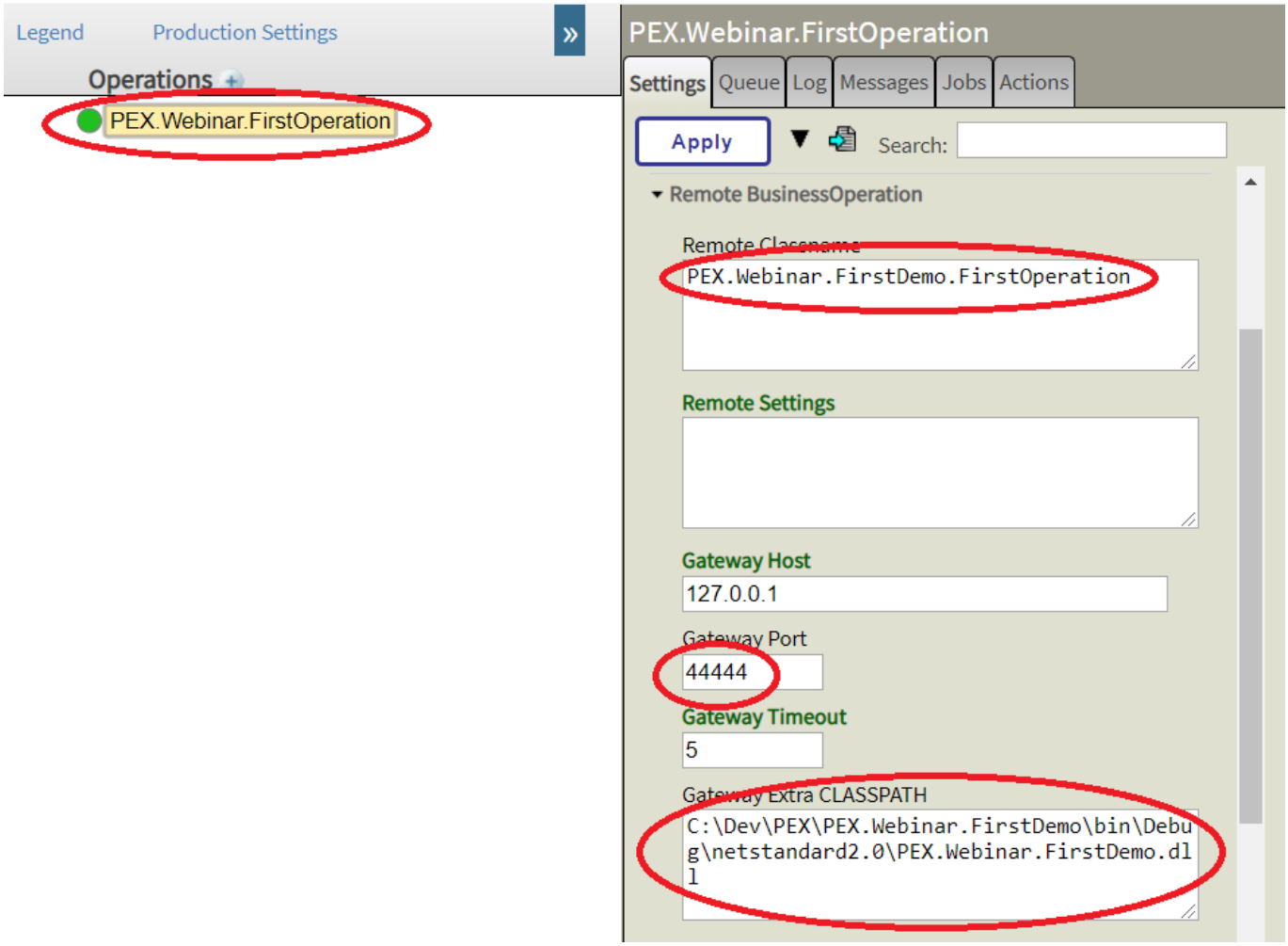
Se añade ahora un Business Operation de PEX para referenciar el código .NET creado anteriormente, con el botón "+" al lado de "Operations". El tipo de clase es "EnsLib.PEX.BusinessOperation", se nombra el componente (opcional) "PEX.Webinar.FirstOperation", se habilita con "Enable Now".



A continuación, se configura el componente (haciendo click sobre el componente añadido) y seleccionando la pestaña "Settings" a la derecha:

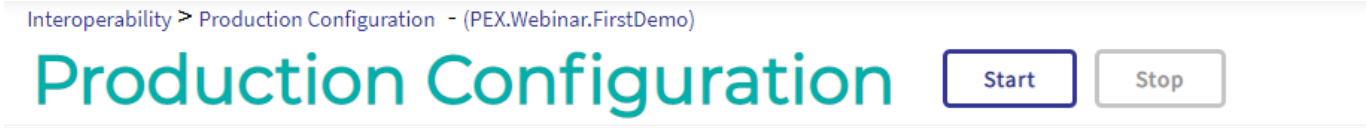
Parámetro	Valor	Descripción
Remote Classname	PEX.Webinar.FirstDemo.FirstOperation	El Nombre de la clase .NET generada
Gateway Port	44444	El puerto TCP en el cual esta configurado el .NET Gateway
Gateway Extra CLASSPATH	C:\Dev\PEX\PEX.Webinar.FirstDemo\bin\Debug\ netstandard2.0\PEX.Webinar.FirstDemo.dll	Ubicación de las .DLL a incluir. Es la ubicación donde la compilación de Visual Studio ha generado la DLL

Para aplicar los cambio, se hace click sobre el botón "Apply":

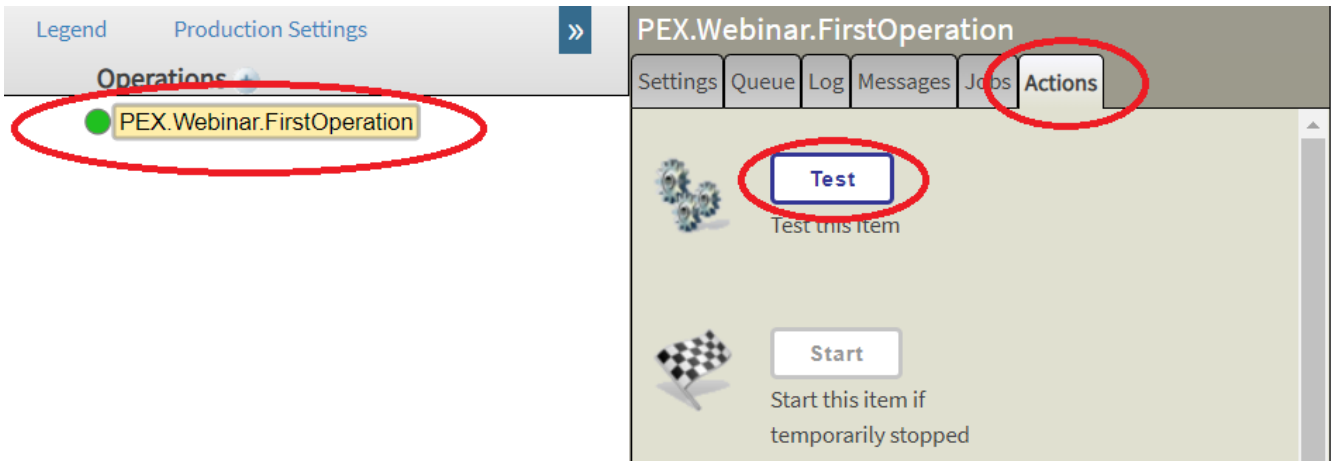


Arrancar la Producción y Probar

El botón "Start" permite arrancar la producción y todos sus componentes:

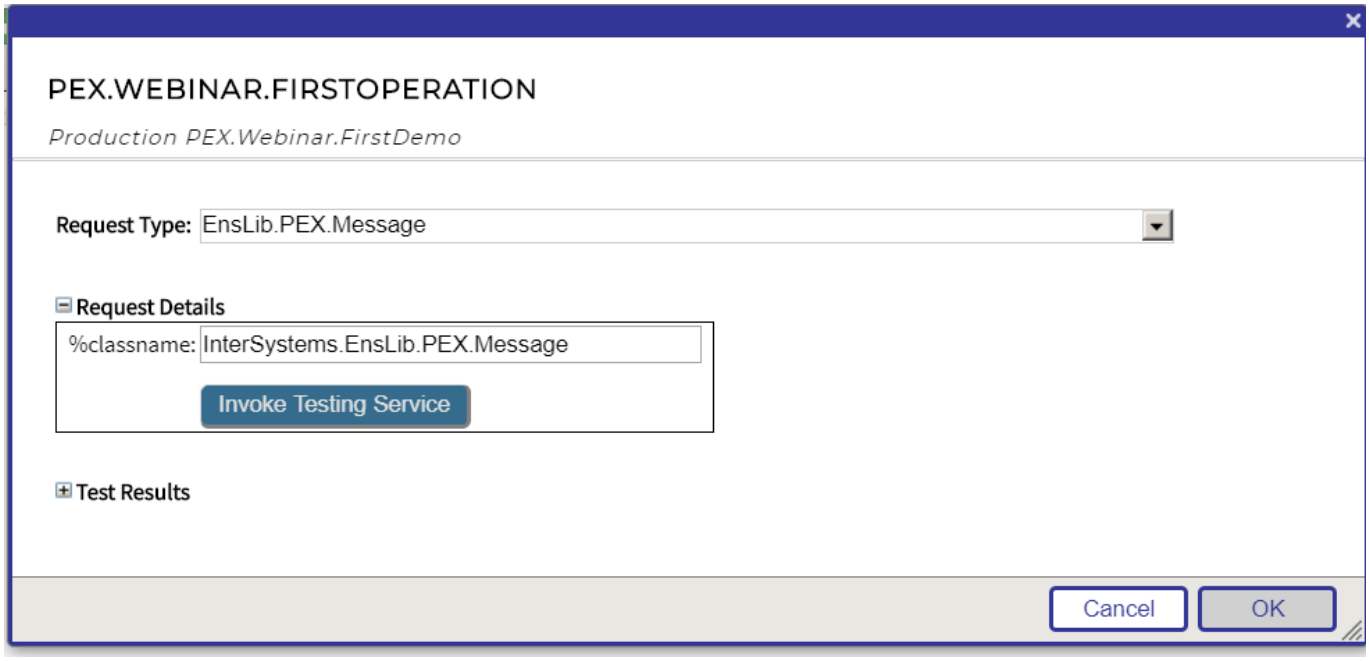


InterSystems IRIS permite probar de manera simple y aislada un componente. Se hace click para seleccionar el "Business Operation" llamado "PEX.Webinar.FirstOperation", y se selecciona el botón "Test" de la pestaña Actions:

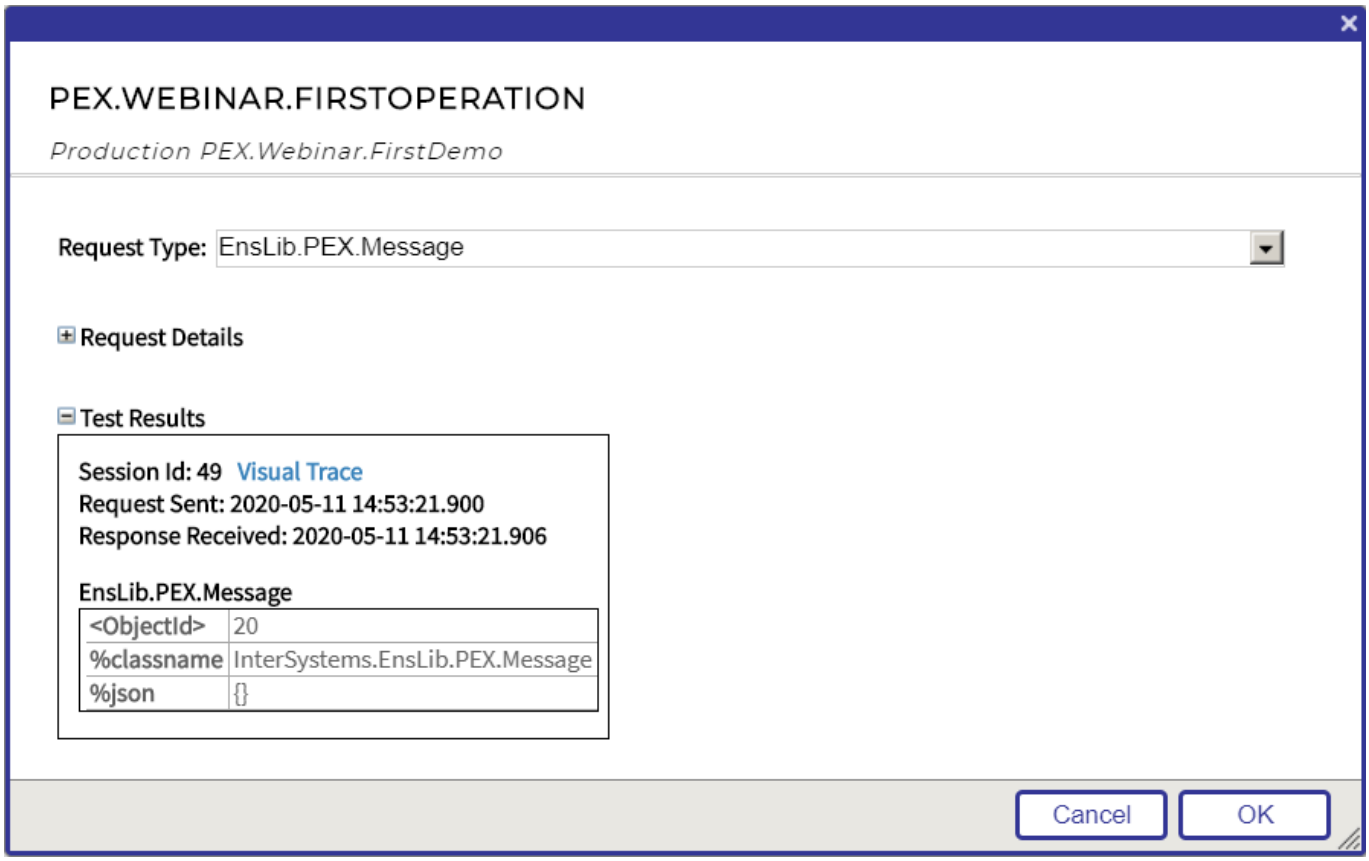


En la pantalla siguiente, se rellenan los datos como sigue:

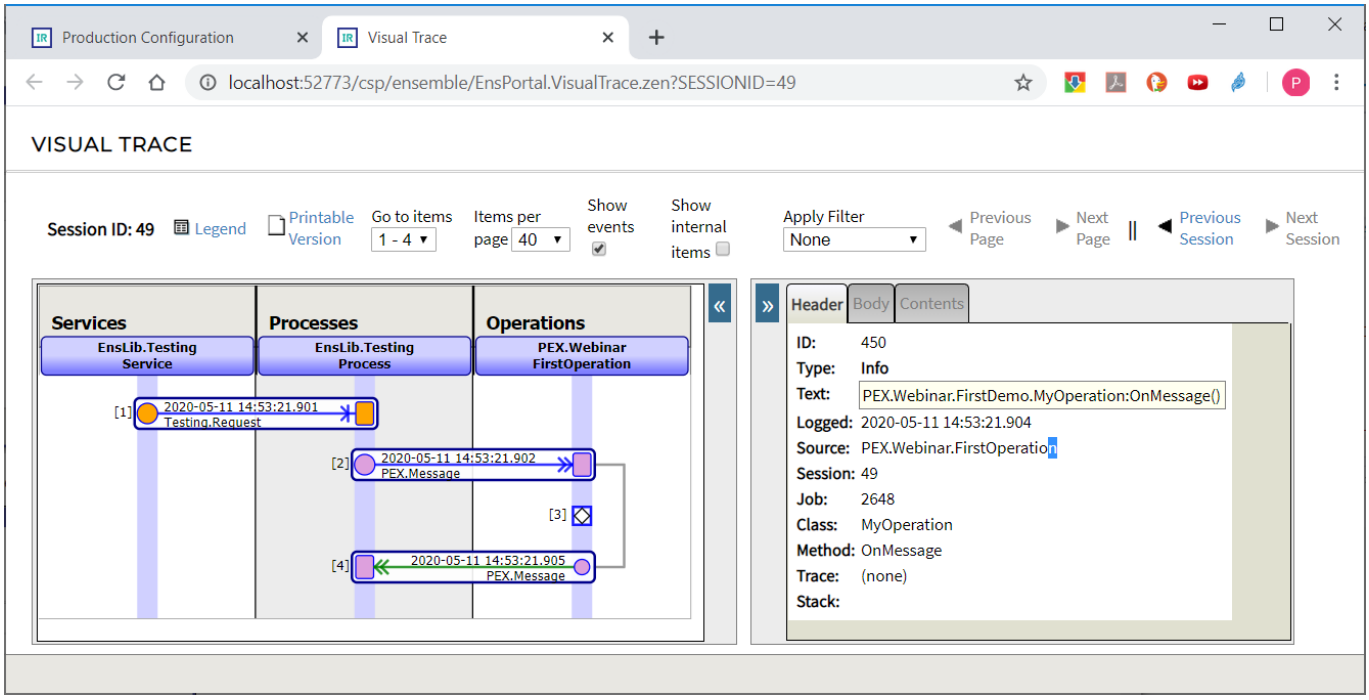
Parámetro	Valor	Descripción
Request Type	EnsLib.PEX.Message	El tipo de mensaje de IRIS que se envía. Siempre es un EnsLib.PEX.Message. Permite añadir propiedades dinámicas para traspasar valores al componente PEX Java o .NET
%classname	InterSystems.EnsLib.PEX.Message	Es el Nombre de la clase del mensaje definido en Java o .NET como petición (request) del componente. Para .NET tiene que ser InterSystems.EnsLib.PEX.Message o una subclase



Cuando se hace click en "Invoke Testing Service", el framework de Interoperabilidad de IRIS envía el mensaje al componente y el código .NET se ejecuta.



Haciendo click sobre el "Visual Trace", se pueden ver los detalles. El punto blanco numerado "3" muestra la traza "LOGINFO" implementada en .NET.



El Log de Eventos de IRIS contiene un recopilatorio de todos los mensajes LOGINFO() generados, incluidos en OnInit() y OnTearDown().

Siguientes Pasos con PEX: Completando la producción

A continuación se crean componentes de otros tipos en .NET:

- Mensajes .NET
- Business Services .NET
- Business Process .NET

Creación y uso de un mensaje PEX con datos

Para pasar información desde Ensemble a componentes PEX, se define una clase en .NET como subclase de InterSystems.EnsLib.PEX.Message, con las propiedades para la información que se quiere pasar. Siguiendo con un ejemplo simplificado, añadimos una propiedad "value" de tipo string al mensaje. El Business Operation PEX devolverá el mismo tipo de mensaje, con el contenido transformado a majusculas.

Clase de mensaje en .NET

Se añade un nuevo fichero "FirstMessage.cs" as proyecto .NET, con la siguiente definición:

FirstMessage

```
using System;

using System.Collections.Generic;

using System.Text;

namespace PEX.Webinar.FirstDemo

{

   class FirstMessage : InterSystems.EnsLib.PEX.Message

   {

      public string value;

   }

}
```

Uso del mensaje desde .NET

En el Business Operation FirstOperation, el tipo de Datos de OnMessage está definido como Object. Se tiene que hacer el cast a la clase "FirstMesssage" para usarlo:

OnMessageV2

```
public override object OnMessage(object request)

     {

        LOGINFO("PEX.Webinar.FirstDemo.FirstOperation:OnMessage()");
```

```
        ///se Instancia el mensaje de respuesta
        FirstMessage response = new FirstMessage();
        //Se copia el value en "uppercase" de la peticion
        response.value = ((FirstMessage)request).value.ToUpper();
        //Se devuelve la respuesta
        return response;
    }
```

Es necesario parar la producción de IRIS (con el botón "Stop"), o como mínimo deshabilitar el .NET Gateway (doble-click sobre el componente), y recompilar el proyecto .NET

Uso de la clase desde InterSystems IRIS

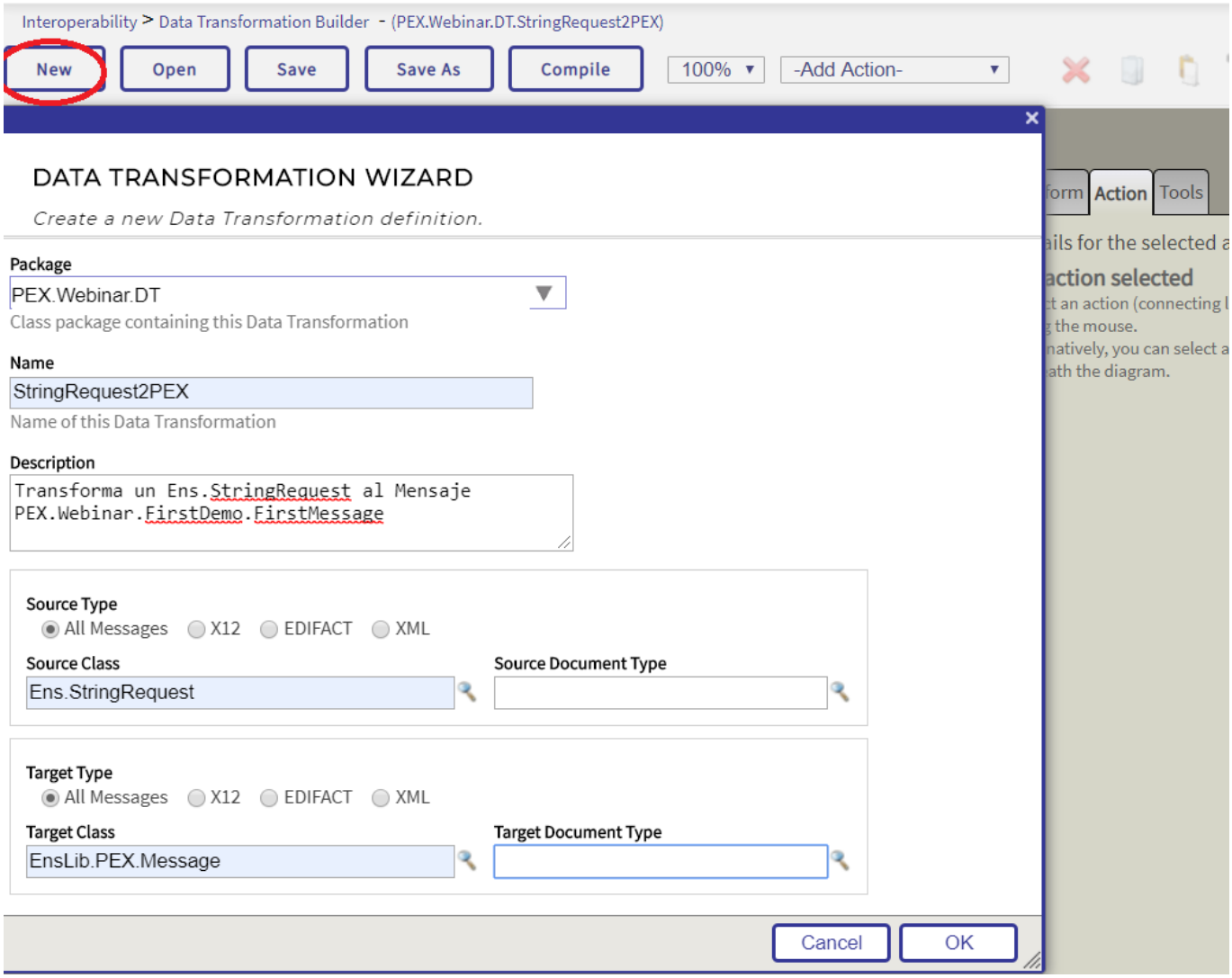
El botón "Test" (Testing Service) no permite rellenar las propiedades dinámicas de la clase EnsLib.PEX.Message. Sin embargo, se puede usar el Testing Service con un mensaje de tipo "Ens.StringRequest" nativo de IRIS, y transformar (Con una Transformación de Datos) este mensaje en un mensaje PEX desde una transformaciçon de datos. La transformación se invocar desde un Business Process de tipo Enrutador ya pre-construido.

Transformación de Datos

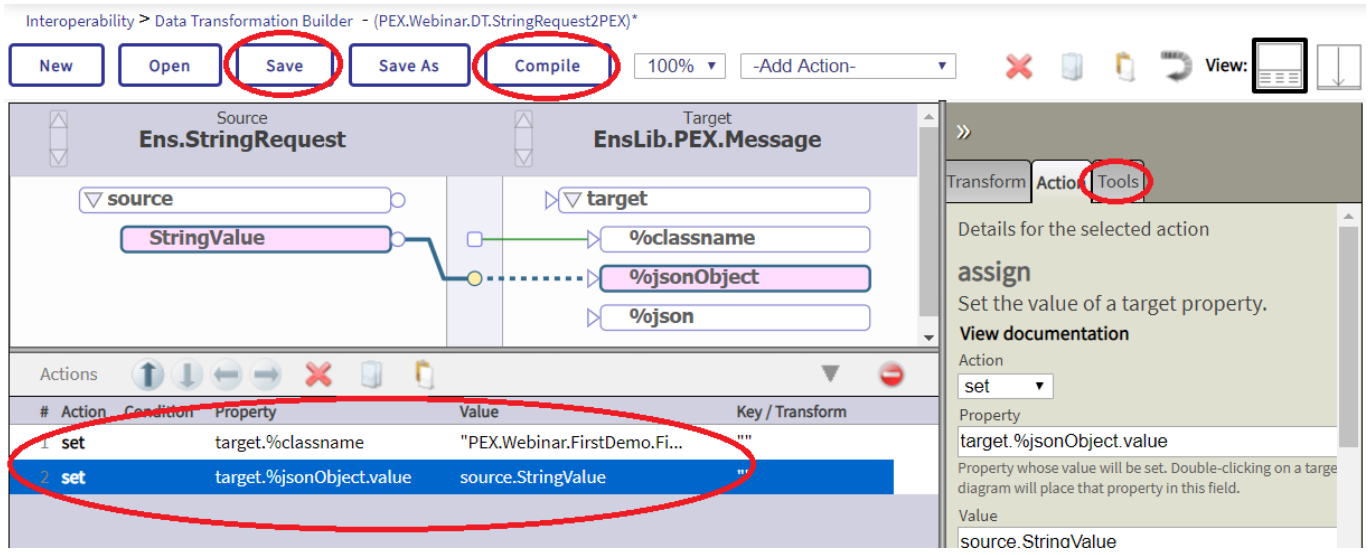
Se crea una Transformación de Datos para copiar los datos del mensaje nativo ensemble:

	source	target	Comentario
Clase	EnsLib.StringRequest	EnsLib.PEX.Message	Las clases de origen y de destino de la transformación
%classname	n/a	PEX.Webinar.FirstDemo.FirstMessage	Para un mensaje PEX (target), el %classname define el nombre de la clase .NET/Java a usar.
property	source.StringValue	target.%jsonObject.value	El en mensaje PEX de destino, las propiedades dinámicas esta ubicadas dentro del %jsonObject.

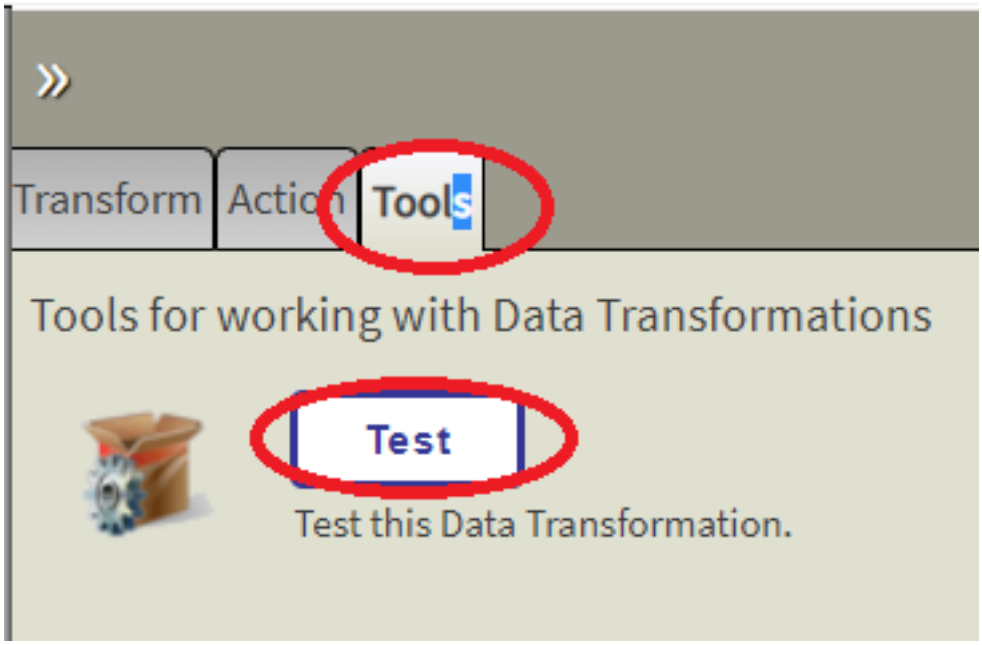
La Transformación de Datos se puede crear desde el portal de gestión "Interoperability" / "Build" / "Data Transformations", con el botón "New":



A continuación se definen las transformaciones del mensaje "source" al "target" como descrito en la tabla anterior, usando la herramienta gráfica y completando el texto en la pestaña "Action" a la derecha, para obtener las 2 líneas de texto mostradas abajo. Después, se puede guardar (botón "Save"), Compilar (botó "Compile") y probar con la pestaña "Tools".



Ahora, abriendo la pestaña tools, se puede probar esta Transformación de Datos:



Se abre la Ventana de Pruebas, donde se puede especificar valores para el mensaje de origen (en StringValue) y verificar el resultado:

Como se puede ver, el mensaje PEX usa una representación interna en json para traspasar los valores entre IRIS y Java o .NET.

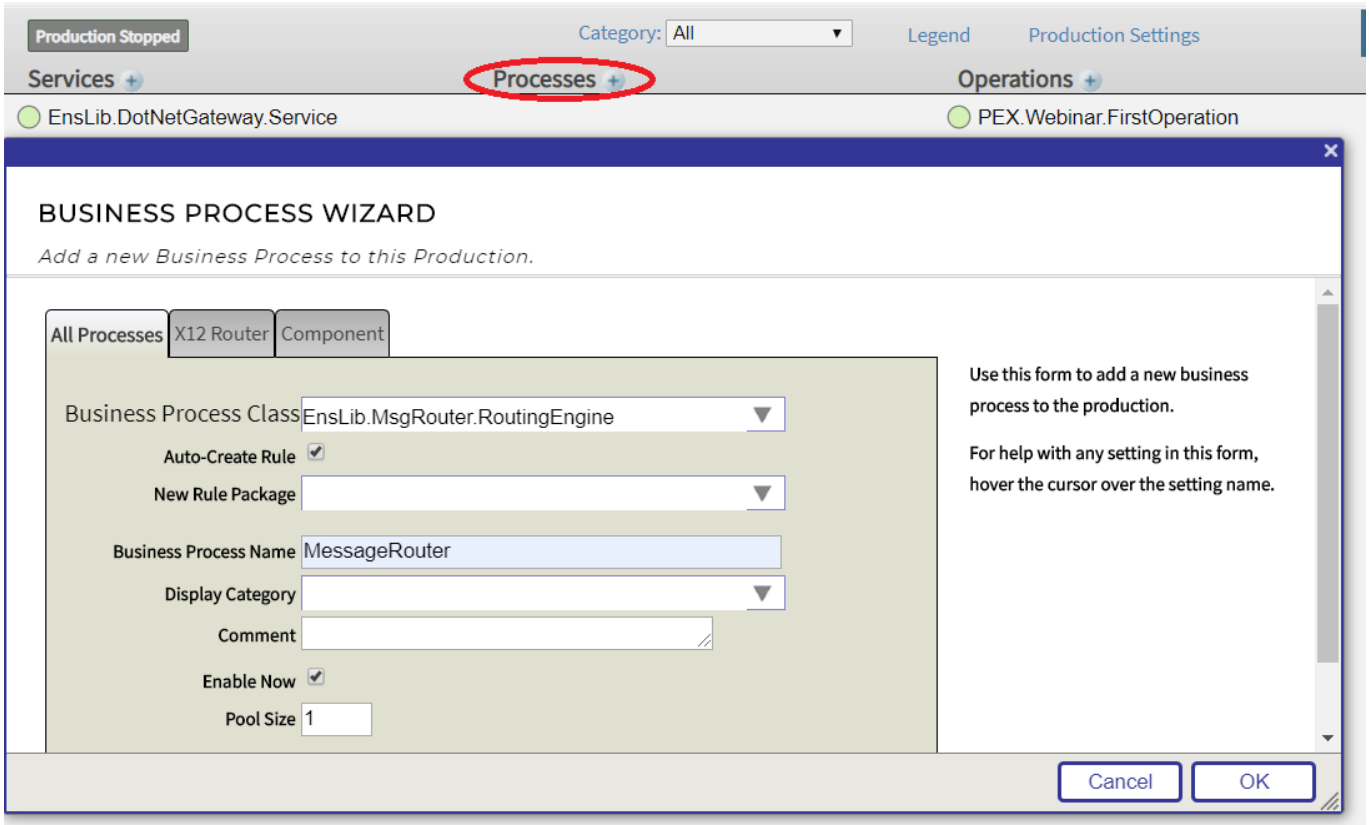
Creación de un Business Process de tipo Enrutador

Ahora es posible añadir un Business Process de tipo Enrutador a la Producción, y definir una regla de

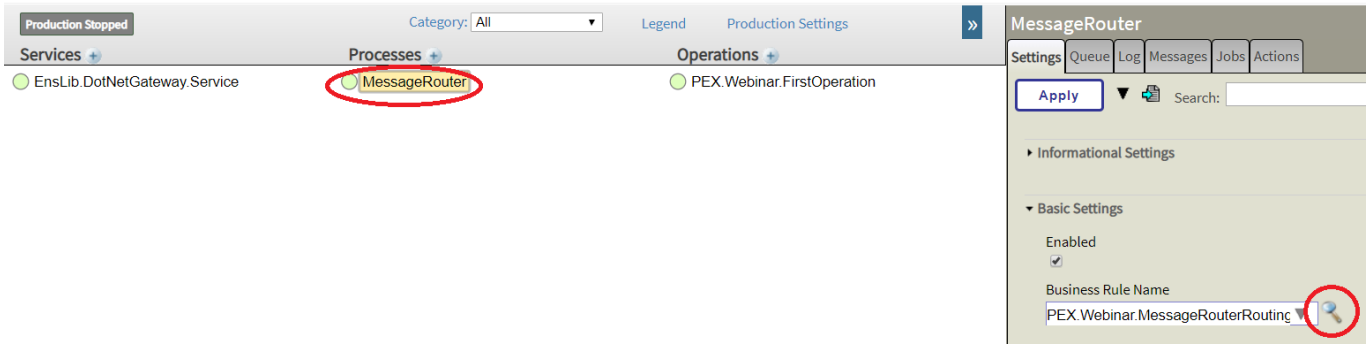
Enrutamiento que llame a la Transformación de Datos creada e envie en mensaje al Business Operation existente.

En el portal de getión, se vuelve a la pantalla de Configuración de Producción ("Interoperability" / "Configure" / "Production"), y se añade el Business Process de tipo "EnsLib.Message.Router", haciendo "click" sobre el "+" al lado de "Processes", y rellenando como sigue:

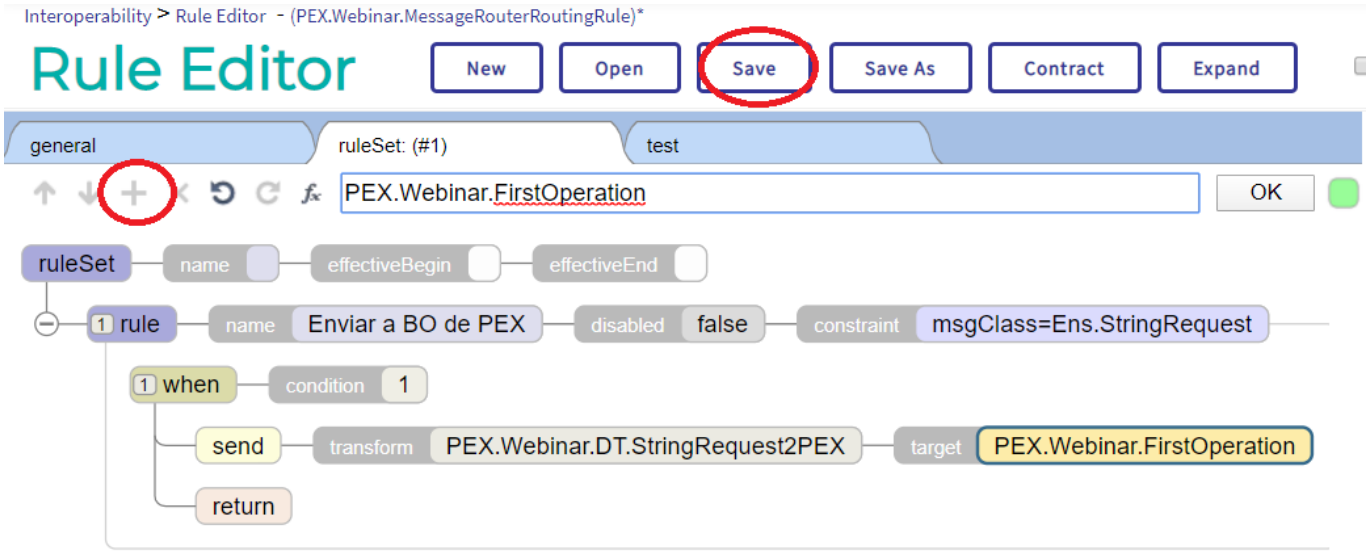
Parametro	valor	descripcio
Business Process Class	EnsLib.MsgRouter.RoutingEngine	Un Business Process para enrutamiento de mensajes basado en reglas
Auto-Create Rule	SI	Definir el esqueleto de una regla de Enrutamiento
Business Process Name	MessageRouter	Una nombre para nombrar este componente en la producción
Enable Now	SI	Activar este componente de inmediato
Pool Size	1	Escalabilidad. Permite definir el número de procesos activos en parallelo.



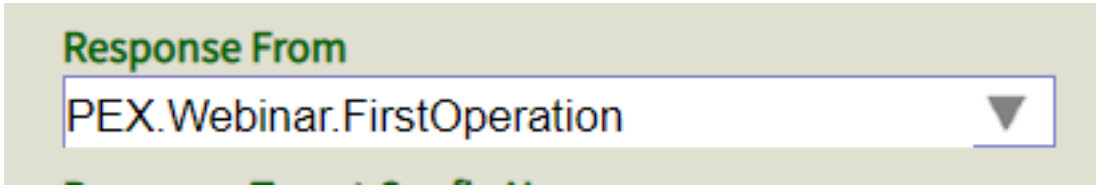
Queda por Editar la Regla de Enrutamiento de este Componente. Par esto se selecciona el componente "MessageRouter" en la producción, y se hace "click" sobre la lupa al lado de la Regla de Enrutamiento (Business Rule Name):



El editor de Reglas permite editar la Regla para enviar todos los mensajes de tipo "Ens.StringRequest" al Business Operation llamado "PEX.Webinar.FirstOperation" después de aplicarles la transformación. El botón "+" permite añadir elementos a las reglas, y se puede editar un elemento seleccionando-lo previamente. La regla a guardar debe estar como sigue:

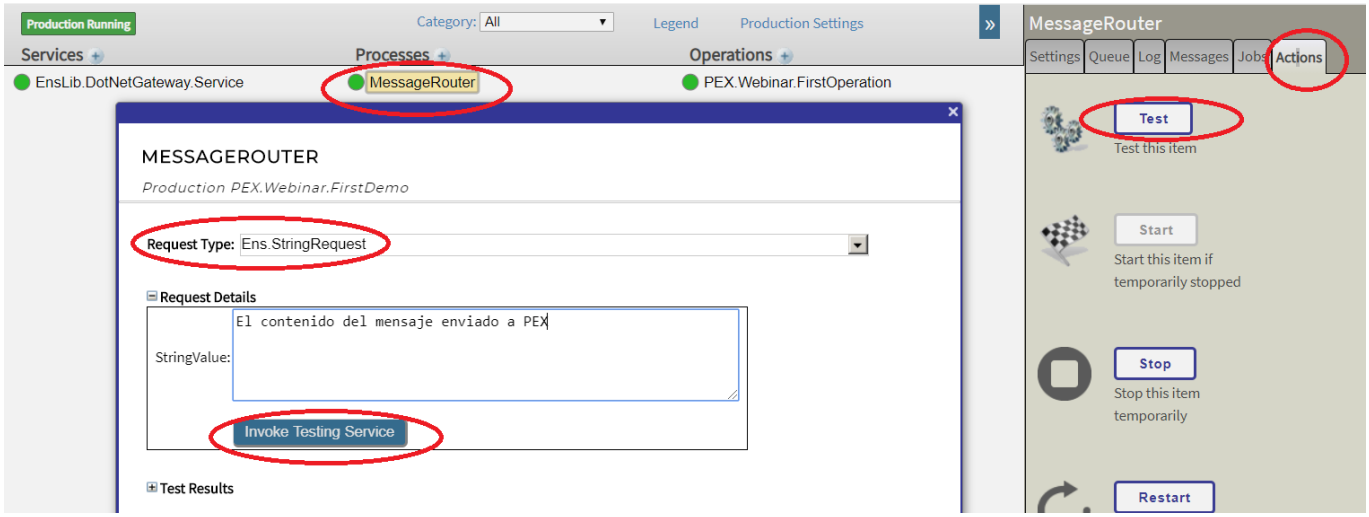


Por defecto este componente no espera respuesta. Se cambia su configuración en la pestaña "Settings" del componente, para recibir la respuesta del Business Operation:



Prueba del Enrutador con el Servicio de Pruebas

De vuelta a la página de la producción, se puede usar el botón "Start" para arrancar la producción, seleccionar el componente "MessageRouter", y con la pestaña "Actions", hacer click sobre el botón "Test", para enviar un mensaje de tipo "Ens.StringRequest":



El mensaje de Respuesta de la prueba se puede ver en "Test Results", y la traza de mensaje completa muestra todos los detalles de la ejecución en los distintos componentes de la producción de Interoperabilidad de IRIS:

Respuesta:

MESSAGEROUTER

Production PEX.Webinar.FirstDemo

Request Type: Ens.StringRequest

Request Details

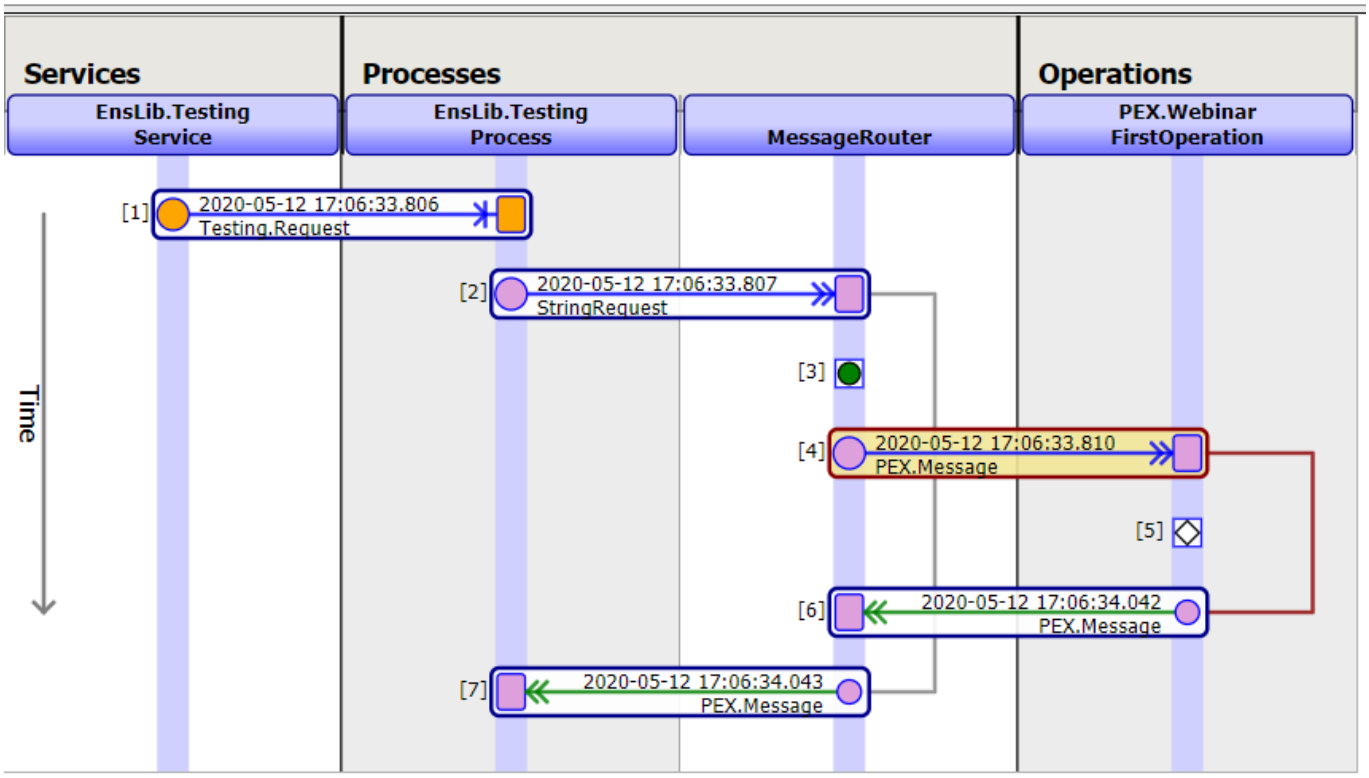
Test Results

Session Id: 65 Visual Trace
Request Sent: 2020-05-12 17:06:33.804
Response Received: 2020-05-12 17:06:34.044

EnsLib.PEX.Message

<ObjectId>	30
%classname	PEX.Webinar.FirstDemo.MyMessage
%json	{"value":"EL CONTENIDO DEL MENSAJE ENVIADO A PEX"}

Traza de los mensajes:



Business Service en .NET

Implementación en .NET

Se vuelve a Visual Studio para crear un Business Service en .NET. En este ejemplo Inicial, este Business Service no tiene "Inbound Adapter" especifico asociado; Se usa el "EnsLib.InboundAdapter", un componente que se ejecuta a intervalos regulares (según el valor de CallInterval) y hace una llamada al Business Service cada CallInterval.

En Visual Studio, se genera un nuevo fichero "FirstService.cs", para la clase "FirstService". El procesamiento de cada evento detectado por el Inbound Adapter se hace en el método "OnProcessInput". El objeto pasado en parámetro depende de la implementación del InboundAdapter, es este caso no se usa:

FirstService:OnProcessInput

```
public override object OnProcessInput(object messageInput)
    {
        //crear un nuevo Objeto de Petición
        FirstMessage myRequest = new FirstMessage();
        myRequest.value = "La Hora de envio es: " + System.DateTime.Now.ToString();
        //Para Enviar Sin esperar una respuesta:
        //SendRequestAsync("PEX.Webinar.FirstOperation", myRequest);
        //Para Enviar y Esperar la respuesta con un timeout de 20 segundos:
        FirstMessage myResponse=(FirstMessage) SendRequestSync("PEX.Webinar.FirstOperation", myRequest,
20);
```

```
        return null;
    }
```

FirstService:OnProcessInput

```
public override object OnProcessInput(object messageInput)
    {
        //crear un nuevo Objeto de Petición
        FirstMessage myRequest = new FirstMessage();
        myRequest.value = "La Hora de envio es: " + System.DateTime.Now.ToString();
        //Para Enviar Sin esperar una respuesta:
        //SendRequestAsync("PEX.Webinar.FirstOperation", myRequest);
        //Para Enviar y Esperar la respuesta con un timeout de 20 segundos:
        FirstMessage myResponse=(FirstMessage) SendRequestSync("PEX.Webinar.FirstOperation", myRequest,
20);
        return null;
    }
```

Para poder enviar el mensaje a cualquier componente de la producción si tener que recompilar, se puede usar un parámetro que se configure desde el portal de gestión; los valores se especifican en formato de una cadena json dentro del parámetro de configuración "RemoteSettings".

Parameters

```
class FirstService : InterSystems.EnsLib.PEX.BusinessService
  {
    /// <summary>
    /// Un parametro que se puede cambiar desde el Portal
    /// </summary>
    public string TargetConfigName;
   (...)
```

Si el valor especificado parar "TargetConfigName" esta a null, el Business Service no podrá enviar el mensaje a

ningún destino. Para detectar este problema lo antes posible, una opción es validar el valor de TargetConfigName cuando se inicia el componente, en la llamada a OnInit():

FirstService:OnInit

```
public override void OnInit()

    {

      //Verificar que las propiedades esten correctamente informadas

      if (TargetConfigName==null )

       {

          LOGWARNING("Falta valor para TargetConfigName; es necsario asignarle un valor en
RemoteSettings");

       }else

       {

          LOGINFO("TargetConfigname=" + TargetConfigName);

       }

    }
```

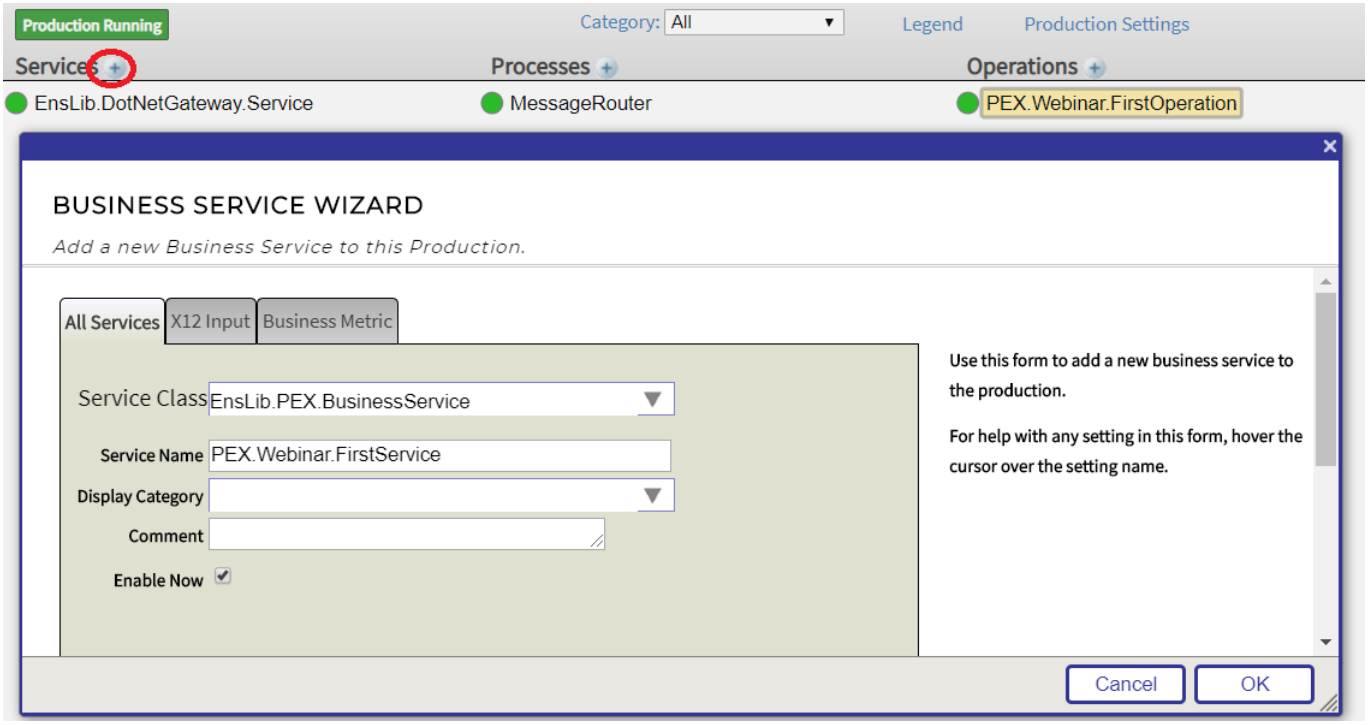
Ahora, se puede Cambiar el OnProcessInput para usar el valor de TargetConfigName:

OnProcessInput

```
//Para Enviar y Esperar la respuesta con un timeout de 20 segundos:

FirstMessage myResponse=(FirstMessage) SendRequestSync(TargetConfigName, myRequest, 20);
```

Business Service PEX en InterSystems IRIS

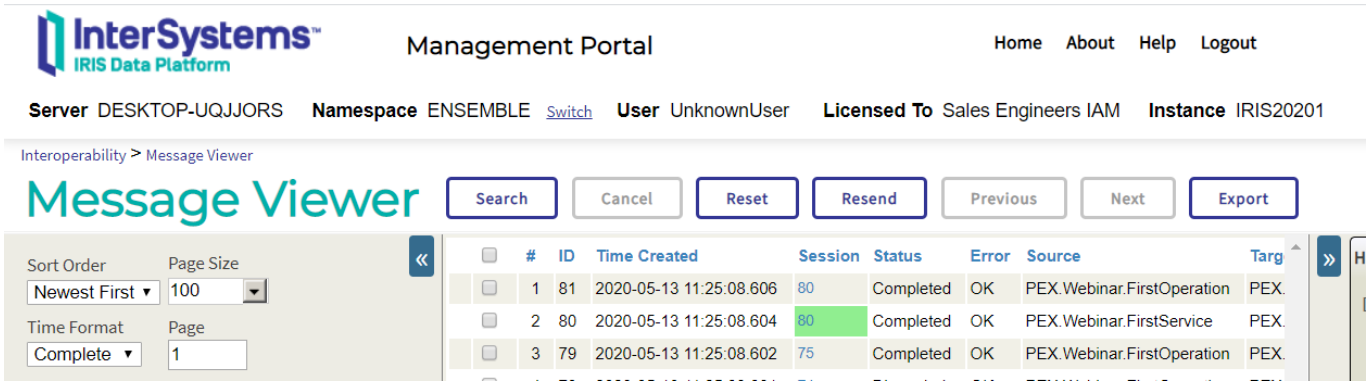
En el portal de Gestión, en la definición de la Producción se añade un componente de tipo Business Service, haciendo click sobre el "+" al lado de



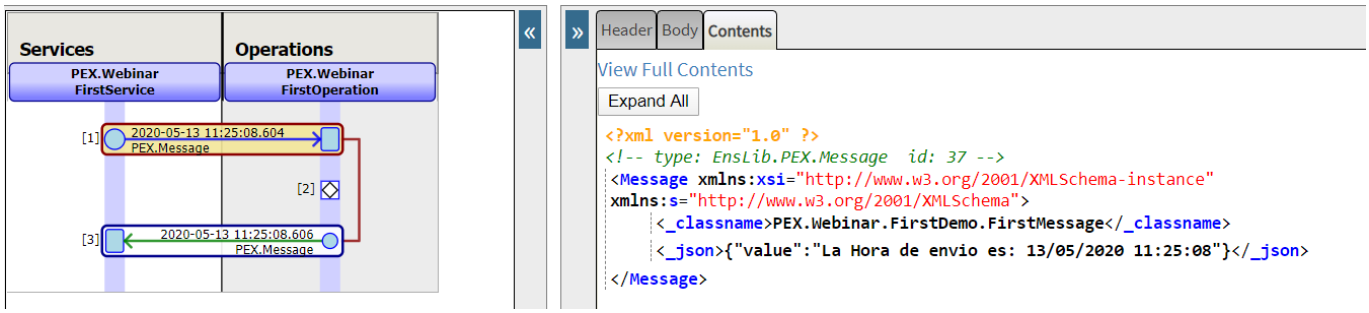
A continuación, se configura el componente (haciendo click sobre el componente añadido) y seleccionando la pestaña "Settings" a la derecha:

Parámetro	Valor	Descripción
CallInterval	20	El tiempo entre cada ejecución del OnTask() del Adaptador Associado, y de cada llamada a OnProcessInput()
Remote Classname	PEX.Webinar.FirstDemo.FirstService	El Nombre de la clase .NET generada
Remote Settings	TargetConfigName=PEX.Webinar.FirstOperation	Una Lista de parametros en formato param=value, separados por <newline>
Gateway Port	44444	El puerto TCP en el cual esta configurado el .NET Gateway
Gateway Extra CLASSPATH	C:\Dev\PEX\PEX.Webinar.FirstDemo\bin\Debug\ netstandard2.0\PEX.Webinar.FirstDemo.dll	Ubicación de las .DLL a incluir. Es la ubicación donde la compilación de Visual Studio ha generado la DLL

Al Activar este Business Service, cada CallIntervall el OnTask() del adaptador asociado (Ens.InboundAdapter) se ejecuta. Aquí, el OnTask simplemente llama FirstService.OnProcessInput(). Es Decir, cada callInterval, FirstService.OnProcessInput() genera un mensaje que envia al componente definido en "TargetConfigName". La Traza de Mensajes (Message Viewer) permite verificar esto:



Con los detalles de una llamada:



Business Process .NET

Implementación .NET

Como Cada Business Host, el Business Process tiene los Metodos de CallBack OnInit() y OnTearDown(). Los Metodos especificos de un Business Process son los siguientes:

Metodo	Descripción
OnRequest(mensaje)	Se ejecuta para cada mensaje que se envía al Business Process. Es el sitio donde implementar las acciones iniciales del Proceso, y realizar envíos asíncronos a otros componentes (Procesos o Operaciones)
OnResponse	Se ejecuta 1 vez para cada respuesta de las llamadas asíncronas realizadas. Permite gestionar las respuestas y guardar los resultados
OnComplete	Se ejecuta 1 vez al final de la ejecución del Business Process. Permite construir el mensaje de respuesta final del Proceso.

Es importante notar que un Business Process puede tener una ejecución que se alarga en el tiempo (horas o días, esperando una respuesta asíncrona). Para permitir esto, el Framework de IRIS puede interrumpir y reanudar la ejecución de un proceso. Todos los valores que se quieren mantener durante un proceso se tienen que poner en propiedades de la clase que IRIS guarda de manera persistente. Para esto, es necesario indicarlo con la anotación "[persistent]".

En este Ejemplo, se realiza una implementación minimalista, donde el Business Process enruta a un destino el mensaje PEX que recibe:

Se usan 2 variables que se pueden modificar desde el portal de Gestión:

FirstProcess

```
class FirstProcess: InterSystems.EnsLib.PEX.BusinessProcess

  {

    //Timeout para las Llamadas

    public string Timeout = "PT10S";

    public string TargetConfigName;

    public override void OnInit()

    {

      //Verificar que las propiedades esten correctamente informadas

      if (TargetConfigName == null)

      {

        LOGWARNING("Falta valor para TargetConfigName; es necesario asignarle un valor en
RemoteSettings");

      }

      else

      {

        LOGINFO("TargetConfigname=" + TargetConfigName);

      }

    }
```

Cuando recibe un mensaje se envia de manera asincrona al TargetConfigName:

FirstProces:OnRequest

```
public override object OnRequest(object request)

    {

      LOGINFO("OnRequest");

      SendRequestAsync(TargetConfigName,
(InterSystems.EnsLib.PEX.Message)request, true); //ResponseRequired=true

      SetTimer(Timeout, "HasTimedOut");

      return null;
```

```
    }
```

En OnResponse permite gestionar las respuestas de las llamadas:

FirstProcess:OnResponse

```
public override object OnResponse(object request, object response, object callRequest,

object callResponse, string completionKey)

    {

        LOGINFO("OnResponse, CompletionKey=" + completionKey);

        if (completionKey!= "HasTimedOut")

        {

            response = (FirstMessage)callResponse;

        }

        LOGINFO("Response:" + response.ToString());

        return response;

    }
```

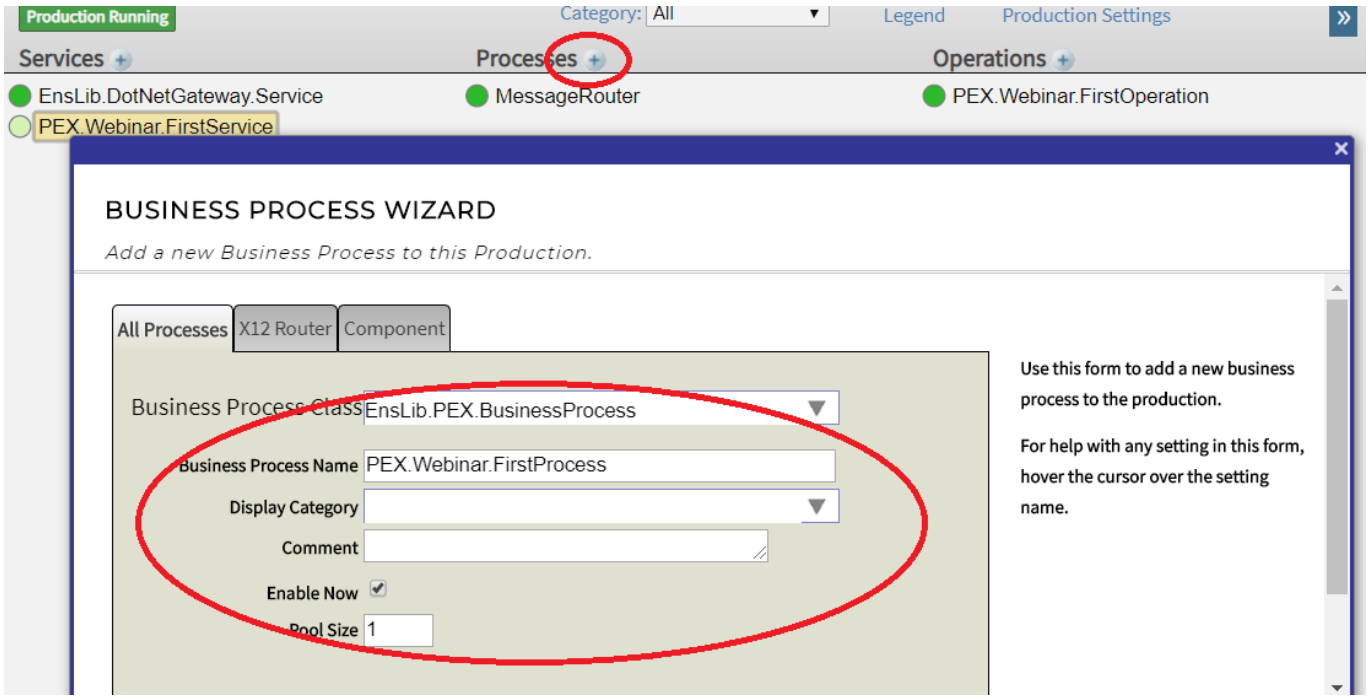
Y al finalizar el proceso, se devuelve la respuesta:

FirstProcess:OnComplete

```
public override object OnComplete(object request, object response)

    {

        LOGINFO("OnComplete");

        return response;

    }
```

Business Process PEX en Intersystems IRIS

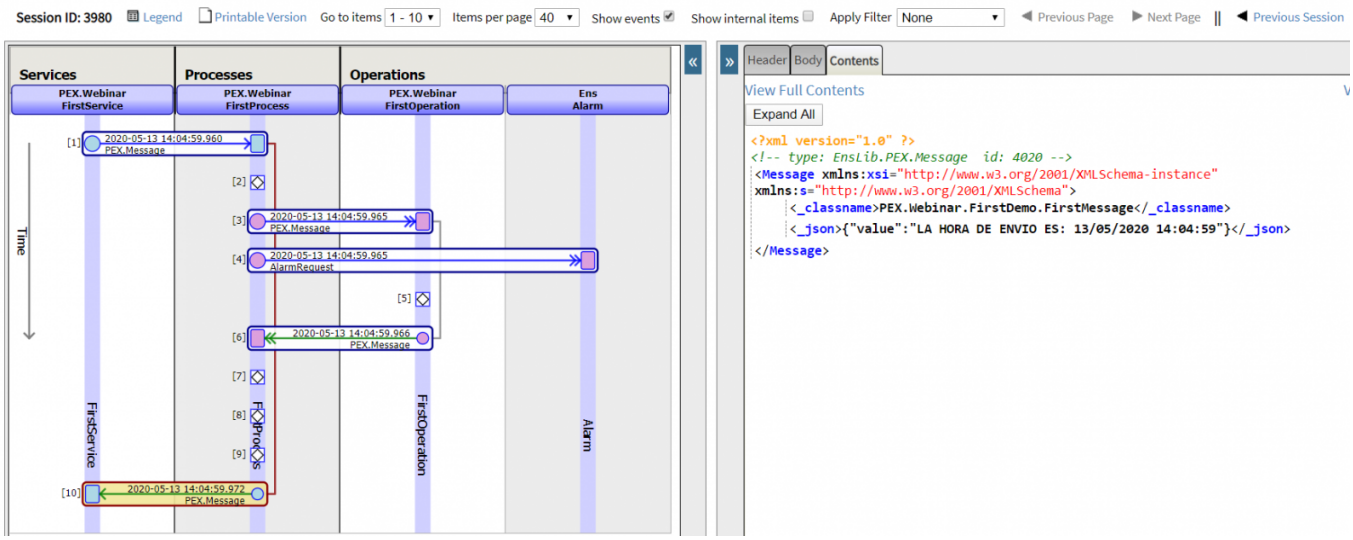
Se añade un Business Process de tipo EnsLib.PEX.BusinessProcess, se configura como sigue, y se cambia el TargetConfigName del Business Service "PEX.Webinar.FirstProcess" para enviar los mensajes al nuevo Processo en vez de enviarlos dicectamente al BO.



Los "Settings" del componente se definen así:

Parametro	valor
Remote Classname	PEX.Webinar.FirstDemo.FirstProcess
Remote Settings	Timeout=PT20S TargetConfigName=PEX.Webinar.FirstOperation
Gateway Port	44444
Gateway Extra Classpath	C:\Dev\PEX\PEX.Webinar.FirstDemo\bin\Debug\netstandard2.0\PEX.Webinar.FirstDemo.dll

El Resultado en la traza es este:



Conclusión

PEX permite implementar integraciones en .NET o Java de manera muy fluida y ofrece toda la potencia y robustez de la capa de interoperabilidad de InterSystems IRIS para desarrolladores con experiencia en estos lenguajes.

#.NET #Interoperabilidad #Java #Mejores prácticas #InterSystems IRIS

URL de fuente:https://es.community.intersystems.com/post/implementar-integraciones-iris-con-net-o-java-usando-pex