Guía de desarrollo, Anexo 22.05

Guía de pruebas de servicios con SoapUI

Arquitectura de Sistemas

GERENCIA INFORMÁTICA
JOSEFA VALCÁRCEL, 44
28027-MADRID

Índice General

Índice de Ilustraciones y Tablas

1 Introducción

1.1 Objetivo

El objetivo de este documento es ilustrar mediante un ejemplo la creación de proyectos y baterías de pruebas mediante SoapUI.

1.2 Audiencia

Equipos de desarrollo con proyecto bajo la versión V3 de la guía de desarrollo que deben entregar proyectos SoapUI en sus despliegues.

1.3 Glosario

Los términos y acrónimos que se utilizan en este documento y en el resto de documentos de la guía se encuentran recogidos por orden alfabético en el Anexo 30. Glosario con el objetivo de facilitar su lectura y comprensión.

1.4 Estructura del documento

Este documento está distribuido en 4 capítulos, con los siguientes contenidos:

- Capítulo 1: Introducción, contiene información relativa al propio documento
- Capítulo 2: Creación de un proyecto SOAP y su correspondiente batería de pruebas.
- Capítulo 3: Importación y exportación de proyectos SoapUI
- Capítulo 4: Anexos: Contiene minitutoriales para casuísticas comunes en DGT.

2 Proyecto SoapUI para servicio SOAP

El primer paso será crear una carpeta donde guardaremos todos los archivos del proyecto.

Una vez creada la carpeta podemos utilizar un workspace a parte o usar uno existente. Para mayor claridad en este tutorial utilizaremos un workspace a parte de forma que solo trabajemos con los proyectos relacionados con este tutorial. Para crear un nuevo workspace seleccionaremos “new workspace” en el menú “File” de SoapUI.

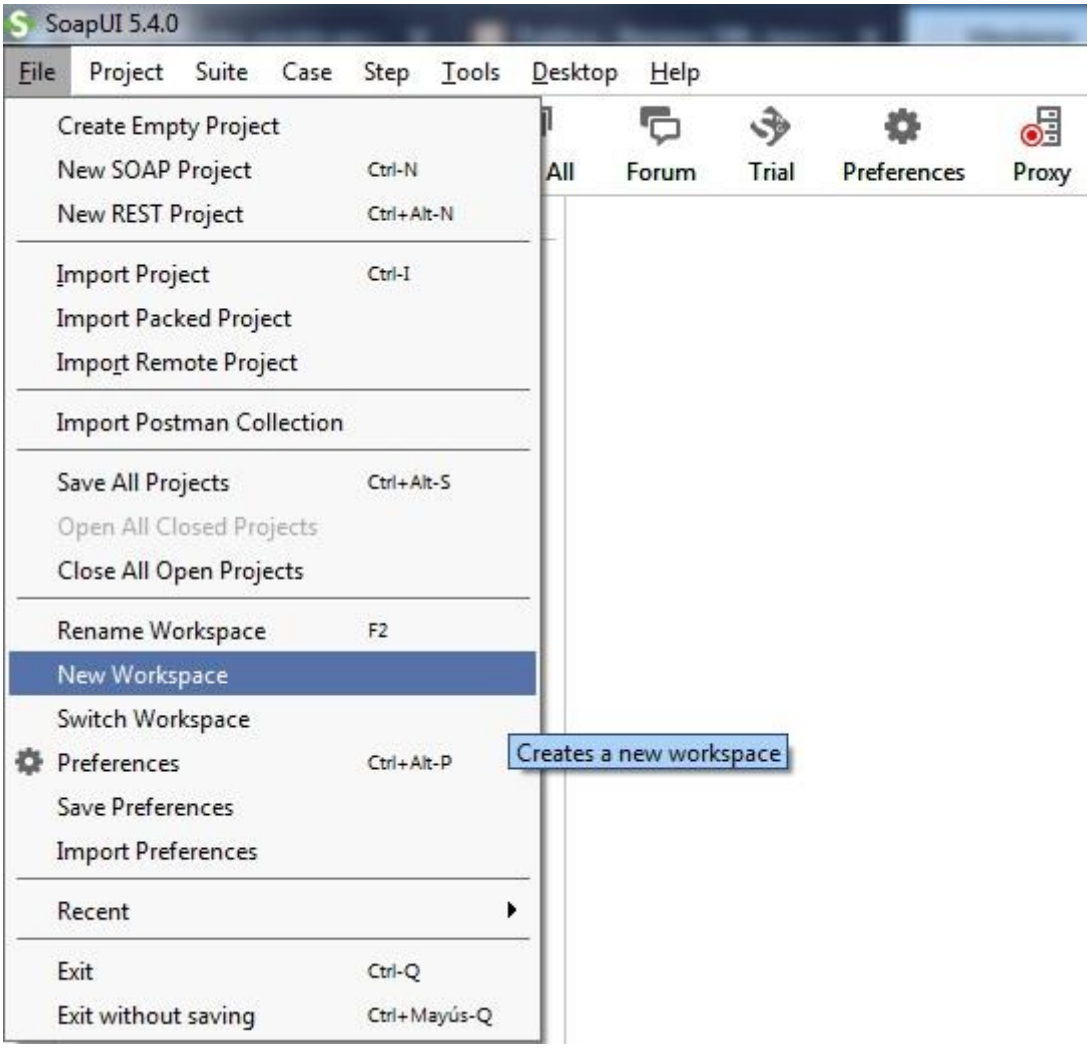

Ilustración 1: Creación de un Workspace

Después de indicar el nombre del workspace seleccionaremos la carpeta creada a tal efecto para guardarlo ahí. En este caso el workspace se llamará SFIR1 ya que es la aplicación que vamos a utilizar de ejemplo. Es importante que el nombre identifique la aplicación a la que pertenece.

Una vez creado el workspace pasaremos a añadir los servicios de SFIR1.

2.1 Creando el proyecto

Para crear el proyecto hacemos click derecho en el nombre del workspace (en este caso SFIR1) y seleccionamos “New SOAP Project”.

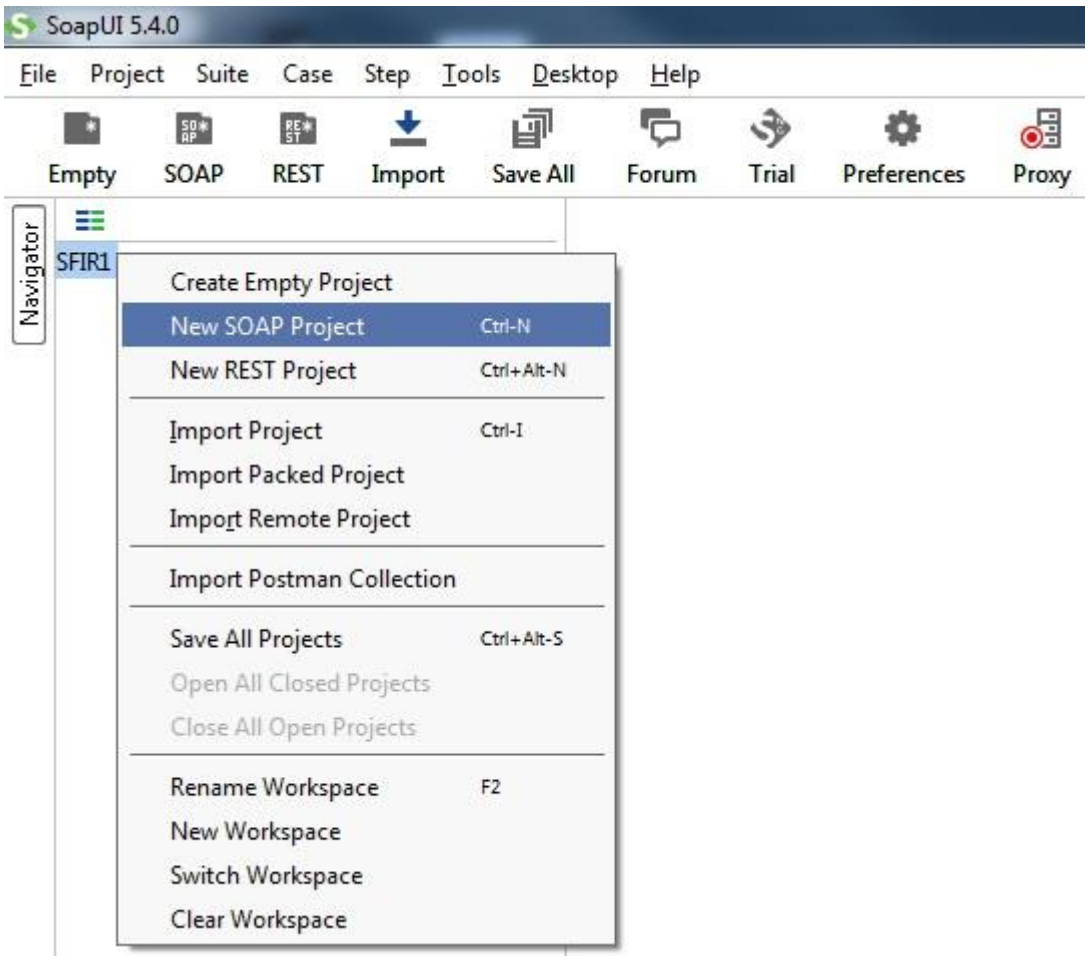

Ilustración 2: Creación de un proyecto SOAP en SoapUI

En la ventana emergente introduciremos el nombre del proyecto. El nombre deberá indicar el acrónimo del proyecto, subsistema y etiqueta. También deberemos indicar donde se encuentra el wsdl del servicio. Esto se puede hacer seleccionado un fichero o utilizando la URL de publicación del servicio. En este caso como el servicio ya está publicado usaremos la URL de publicación, añadiendo la cadena “?wsdl” a la URL de publicación, en este caso:

http://pr-pre-apl.trafico.es:80/WS_SFIR_ADAPTADOR_VALIDACION/services/DataPowerSOAP?wsdl

Dejaremos seleccionado “Create simple request for all operation” para que nos cree una petición vacía de ejemplo para cada operación.

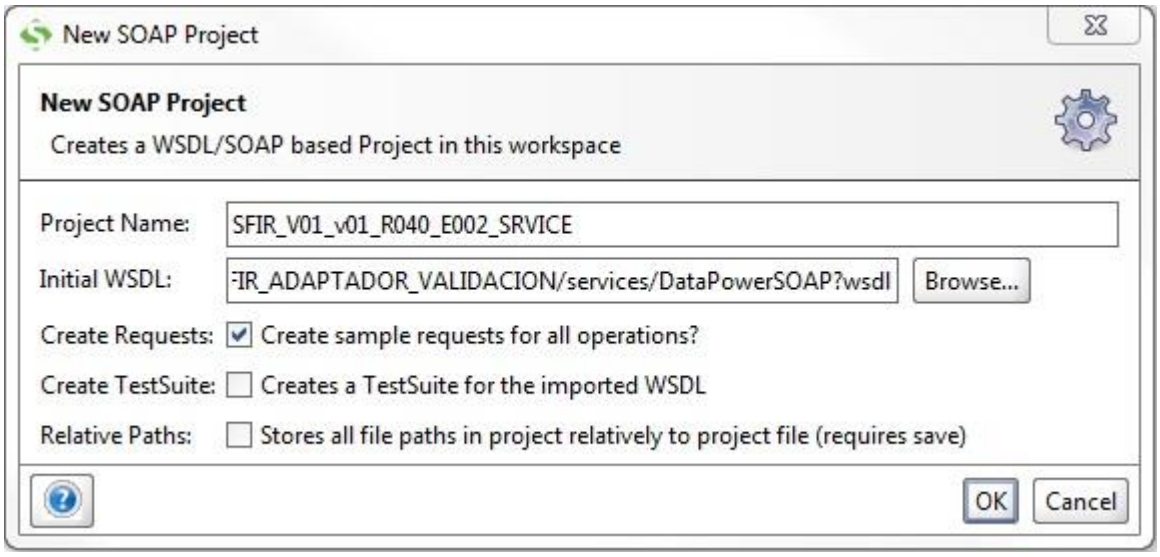

Ilustración 3: Nombre y WSDL del nuevo proyecto

Una vez cargado, tendremos una nueva carpeta en nuestro workspace con un nuevo proyecto, y dentro del proyecto, para cada operación, una llamada de ejemplo:

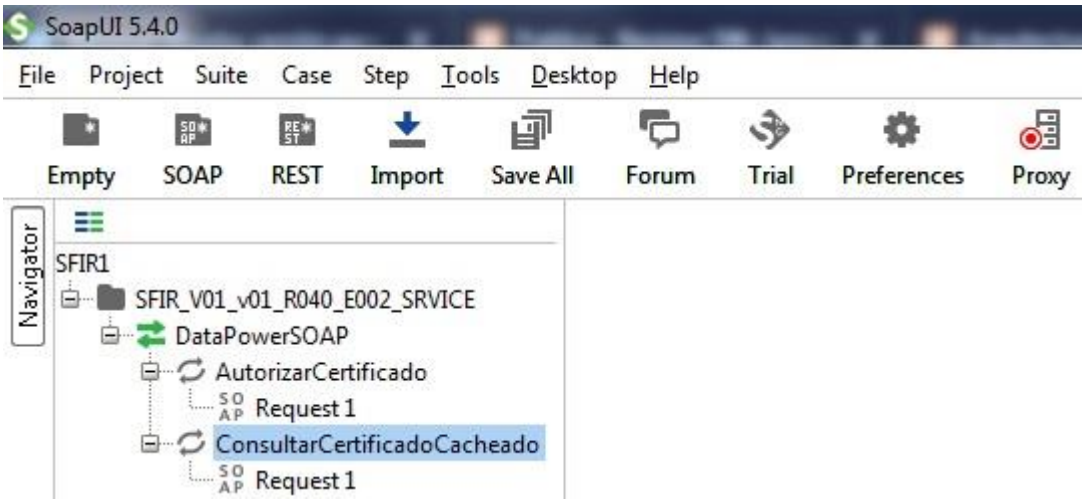

Ilustración 4: Estructura del proyecto

En este caso el servicio tiene dos operaciones “AutorizarCertificado” y “ConsultarCertificadoCacheado”. La operación “AutorizarCertificado” permite validar un certificado

y la operación “ConsultarCertificadoCacheado” extraer datos ampliados de un certificado validado anteriormente.

En cada una de las operaciones, SoapUI ha generado un ejemplo con la estructura de la llamada, siempre hay que revisar que el endpoint (extraído del wsdl) es el correcto:

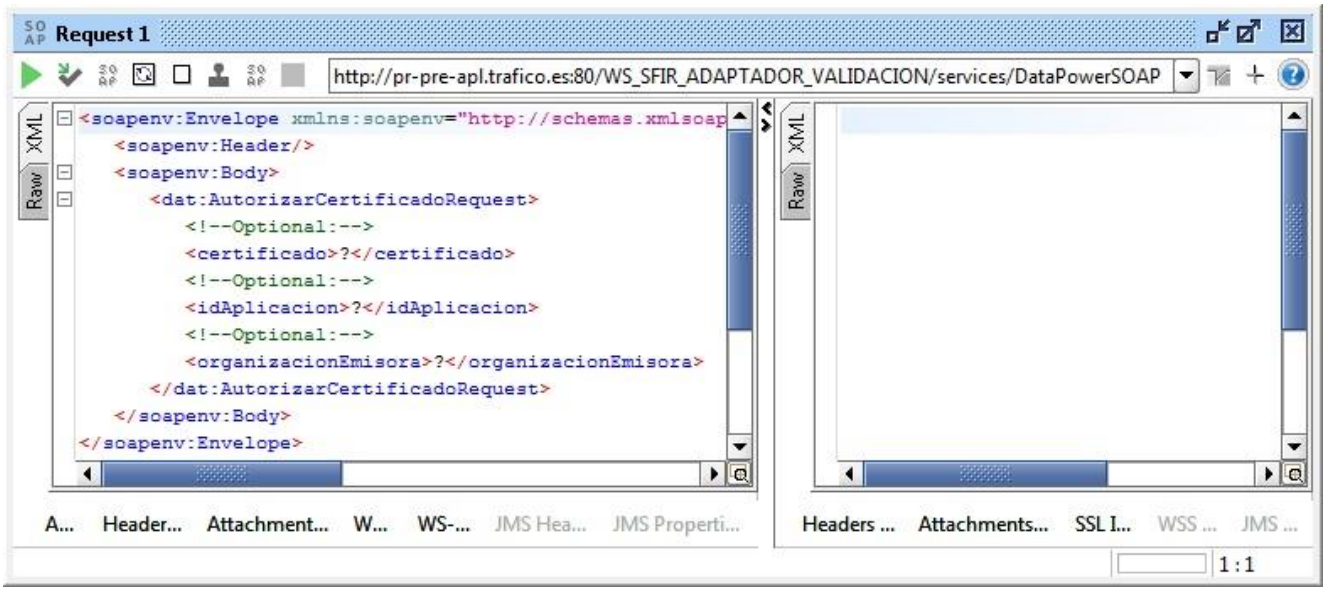

Ilustración 5: Llamadas de ejemplo

Como se puede ver los datos están sustituidos por un símbolo de interrogación “?”. El siguiente paso por tanto es cumplimentar estos datos, sustituyendo los interrogantes de la petición o sustituyendo la petición por una llamada real al servicio.

2.1.1 Creando llamadas de prueba

En este caso la llamada a la operación “AutorizarCertificado” tiene tres parámetros:

Certificado: Certificado a validar codificado en base64.
idAplicación: Usaremos DPOW
organizacionEmisora: Usaremos DATAPOWER.

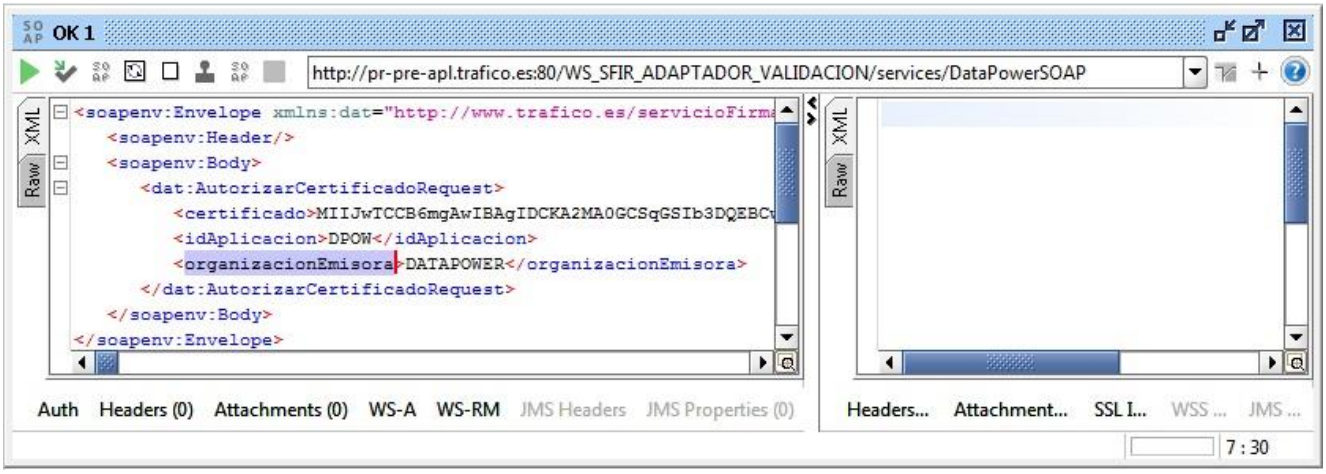

Ilustración 6: Petición para SFIR1

Una vez tenemos la llamada cumplimentada podemos pulsar el símbolo de play para lanzar la petición. En la parte de la derecha aparecerá la respuesta del servicio, en este caso los datos del certificado.

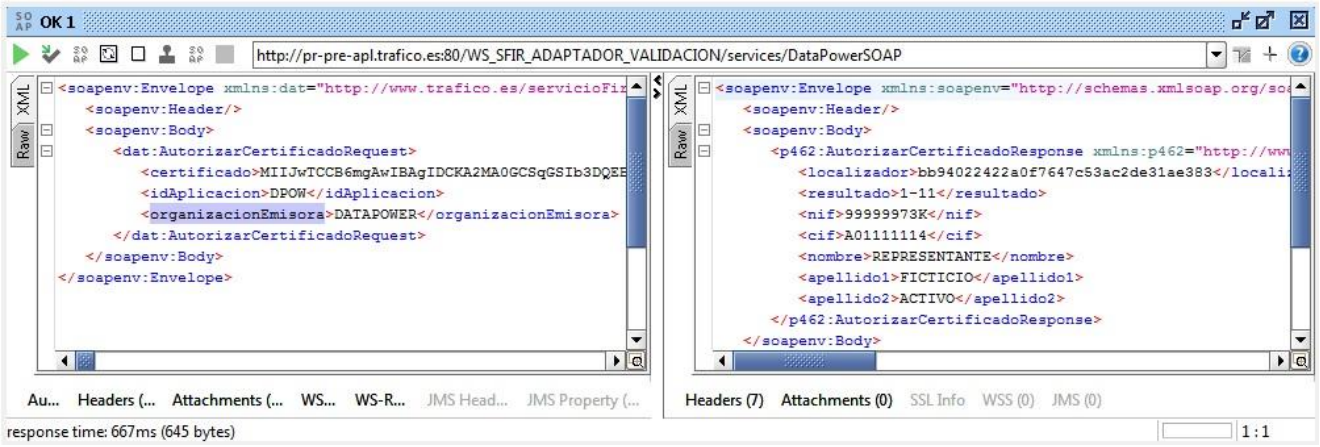

Ilustración 7: Primera llamada a SFIR1

Para identifica mejor las peticiones sustituiremos el nombre de la petición por defecto “Request 1” por algo más informativo como por ejemplo en este caso “01_RepresentanteFicticion OK”. De esa forma sabremos que esa petición valida el certificado de Representante Ficticio y que debería de funcionar. Para ello seleccionaremos la petición y haremos click derecho y seleccionaremos “Rename”.

Nota: Esta operación también se puede realizar seleccionando el nombre de la petición y pulsando la tecla F pero es muy importante asegurarse de selecciona bien la petición porque de lo contrario podemos cambiarle el nombre al workspace o a cualquier otra cosa.

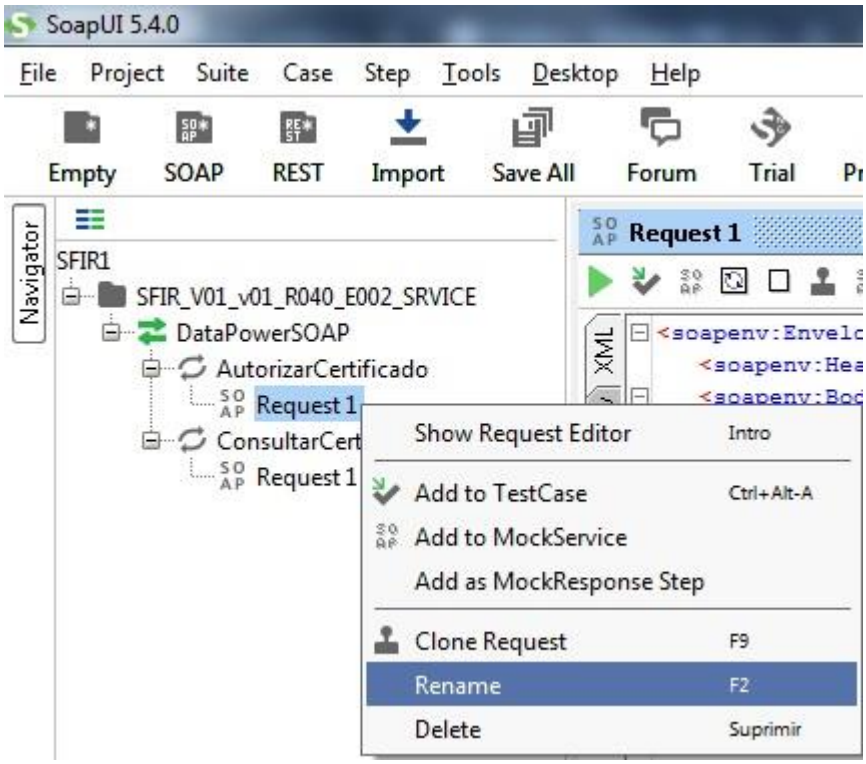

Ilustración 8: Renombrado de una petición

Clonando esa petición podemos ir añadiendo diferentes peticiones para probar lo mejor posible la aplicación, incluyendo peticiones con fallos. En este caso incluiremos una petición con un certificado caducado. Para ello pulsamos click derecho en la petición y seleccionamos “Clone Request”.

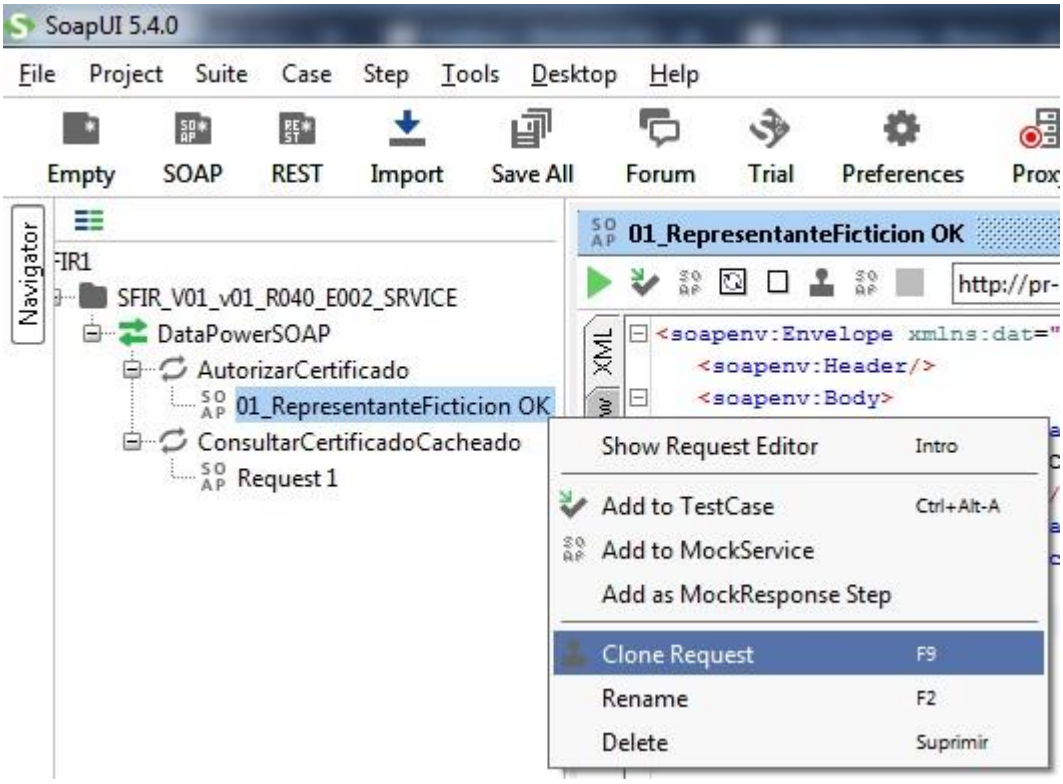

Ilustración 9: Clonado de una petición

Como en esta petición usaremos el certificado caducado del usuario Ciudadano Ficticio Actico de la CA (autoridad certificadora) de IZEMPE, llamaremos a la petición 02_CiudadanoFicticioIzempe KO. Una vez clonada la petición, sustituiremos el contenido de la etiqueta “Certificado” de la petición por el certificado de izempe. Pulsando el botón “play” podemos comprobar el resultado de la aplicación al enviar este certificado caducado:

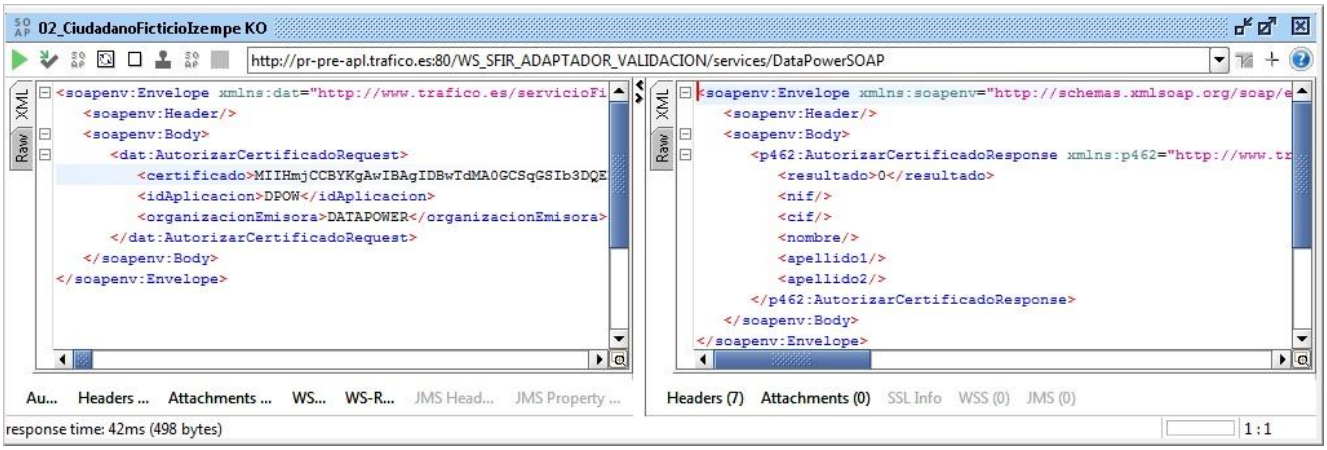

Ilustración 10: Llamada a la aplicación

Comparemos las dos respuestas:

```
<soapenv:Envelope>
   <soapenv:Header/>
   <soapenv:Body>
      <p462:AutorizarCertificadoResponse>
         <localizador>bb94022422a0f7647c53ac2de31ae383</localizador>
         <resultado>1-11</resultado>
         <nif>99999973K</nif>
         <cif>A01111114</cif>
         <nombre>REPRESENTANTE</nombre>
         <apellido1>FICTICIO</apellido1>
         <apellido2>ACTIVO</apellido2>
      </p462:AutorizarCertificadoResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Tabla 1. Respuesta de SFIR1 para un certificado válido

```
<soapenv>
   <soapenv:Header/>
   <soapenv:Body>
      <p462:AutorizarCertificadoResponse>
         <resultado>0</resultado>
         <nif/>
         <cif/>
         <nombre/>
         <apellido1/>
         <apellido2/>
      </p462:AutorizarCertificadoResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Tabla 2. Respuesta de SFIR1 para un certificado inválido

La especificación de SFIR1 indica que si el certificado es correcto, SFIR1 devolverá en el campo resultado un 1, un guion, y el tipo de certificado. En este caso, como el certificado es de representante (tipo 11) devuelve como resultado “1-11”. En caso de que el certificado no sea válido, SFIR devuelve el campo resultado a 0 y todos los datos del certificado en blanco. Conociendo esto y utilizando diferentes llamadas veremos cómo se puede comprobar que el comportamiento de la aplicación es el esperado.

Nota: A la hora de probar una aplicación lo ideal es utilizar tantas peticiones como casos de uso tenga la aplicación, en este caso por ejemplo lo ideal sería utilizar una llamada por cada tipo de certificado soportado. De la misma forma se deben utilizar todas las llamadas necesarias para probar los errores de la aplicación, en este caso por ejemplo además de certificados caducados, certificados

mal formados, etc. También puede ser interesante probar las dependencias de la aplicación para en caso de fallo poder distinguir si se debe a la aplicación o al fallo de sistemas externos. Para ilustrar este ejemplo, sin embargo utilizaremos únicamente estas dos llamadas.

Para la operación "ConsultarCertificadoCacheado" realizamos el mismo procedimiento que con la operación "AutorizarCertificado", cumplimentando los datos que faltan y realizando una llamada de prueba comprobando la respuesta. El parámetro de entrada "localizador" es el localizador devuelto por SFIR1 al llamar a la operación "AutorizarCertificado" y para un mismo certificado es siempre el mismo. Como la segunda petición que hemos utilizado es errónea solo disponemos de la localización del certificado de Representante Ficticio Activo. Para saber que el localizado pertenece a ese certificado llamamos a la nueva llamada igual que en la otra operación.

Ilustración 11: Llamada a ConsultarCertificadoCacheado

Al realizar la llamada podemos comprobar que SFIR1 responde ampliando la información devuelta en la primera llamada, tal y como se espera:

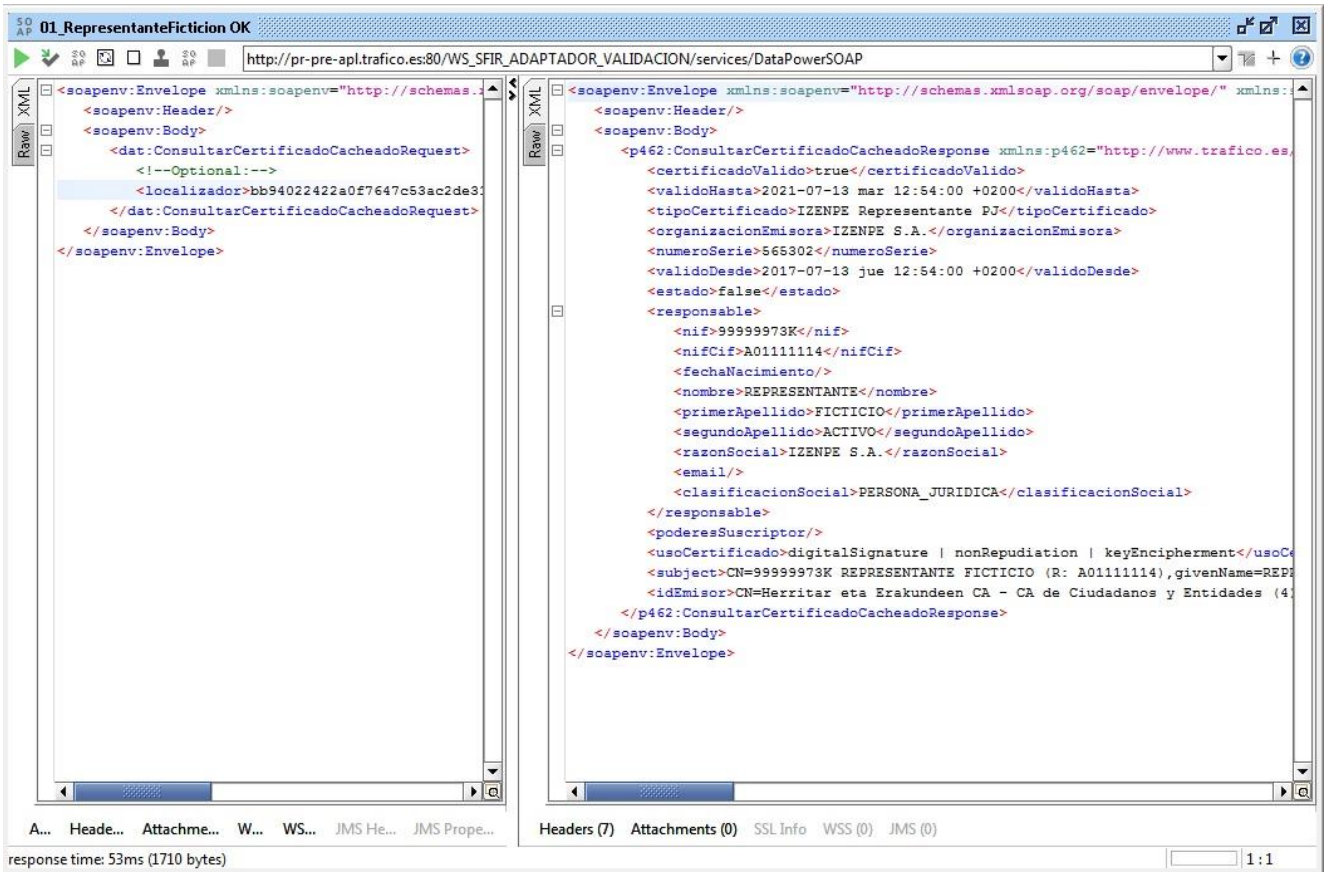

Ilustración 12: Llamada a ConsultarCertificadoCacheado

Con esto ya tendríamos llamadas de prueba para todas las operaciones del servicio. En el siguiente apartado veremos cómo se pueden usar estás llamadas para probar la aplicación. Antes de nada, es recomendable pulsar el botón “Save All” y guardar el proyecto en la carpeta creada para tal efecto.

2.2 Creando la batería de pruebas

Una vez se dispone de llamadas para los casos de uso de la aplicación se puede usar SoapUI para hacer una batería de pruebas. Para ello seleccionamos el nombre del servicio, en este caso “DataPowerSOAP” y hacemos click derecho seleccionando la opción “Generate TestSuite”.

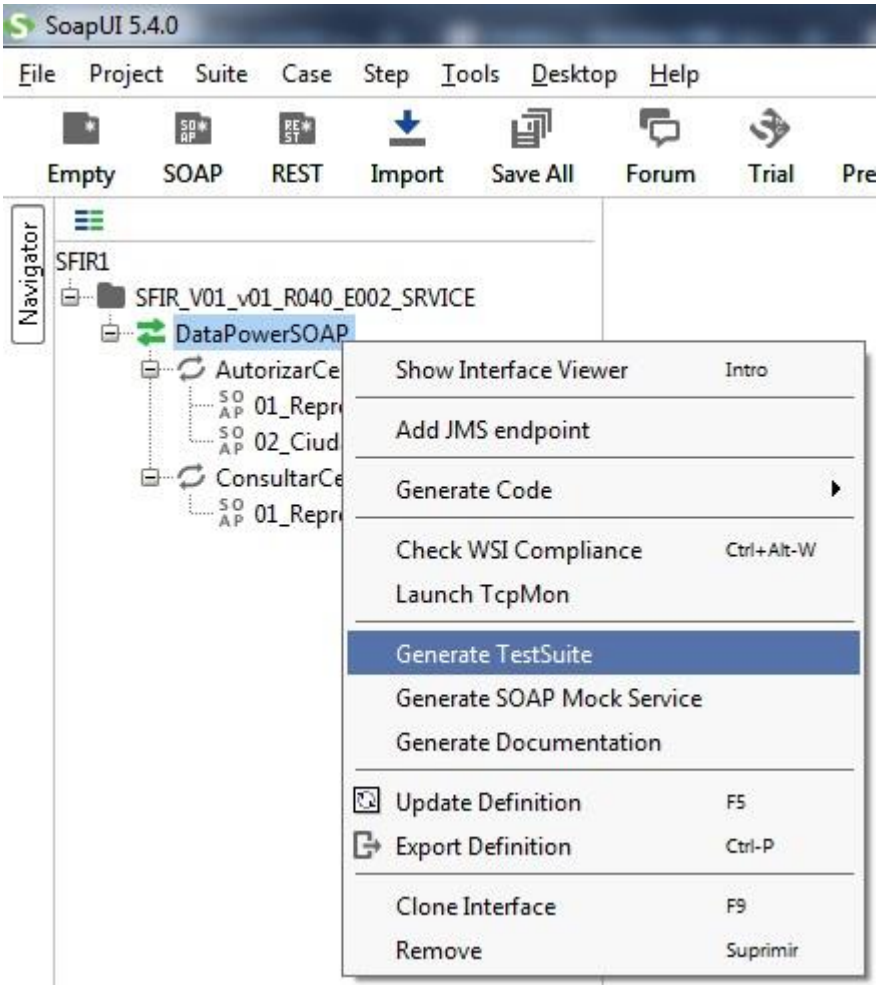

Ilustración 13: Creación de TestSuite

En la siguiente ventana aparecerán las opciones para crear la batería de pruebas. Una batería de pruebas “TestSuite” puede incluir varios casos de prueba “TestCase” y estos a su vez varias peticiones “TestStep”. En este caso crearemos un “TestCase” por cada operación, en nuestro caso “ConsultarCertificadoCacheado” y “AutorizarCertificado” y le pediremos al SoapUI que incluya automáticamente las peticiones que construimos en el paso anterior. Para ello dejaremos la pantalla de opciones tal cual está salvo que seleccionaremos la opción “Use existing Request in Interface”.

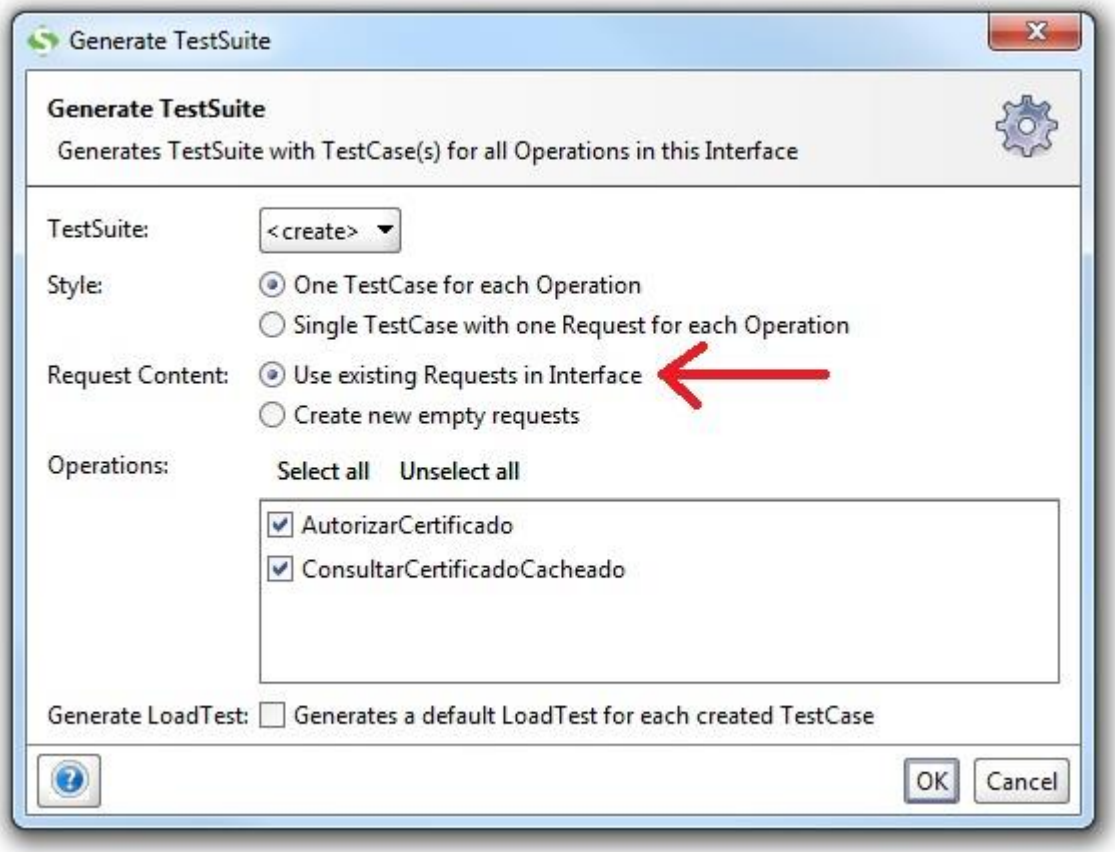

Ilustración 14: Configuración de TestSuite

Al pulsa OK se solicitará un nombre para la batería de pruebas, este debe ser lo más informativo posible. En este caso lo llamaremos "Prueba Completa". En la siguiente ventana solicitará un nombre, ya que por defecto cada llamada se llama como su operación. Como en la operación AutorizarCertificado hay dos llamadas no las puede llamar igual, de ahí el aviso. Podemos pulsar aceptar y cambiar los nombre después para que se vea claramente que es cada petición.

Ilustración 15: Renombrado de TestStep

Al pulsar aceptar aparecerá una ventana con la batería de pruebas “Prueba Completa” y en ella los dos casos de pruebas, uno por operación “AutorizarCertificado TestCase” y “ConsultarCertificadoCacheado TestCase”.

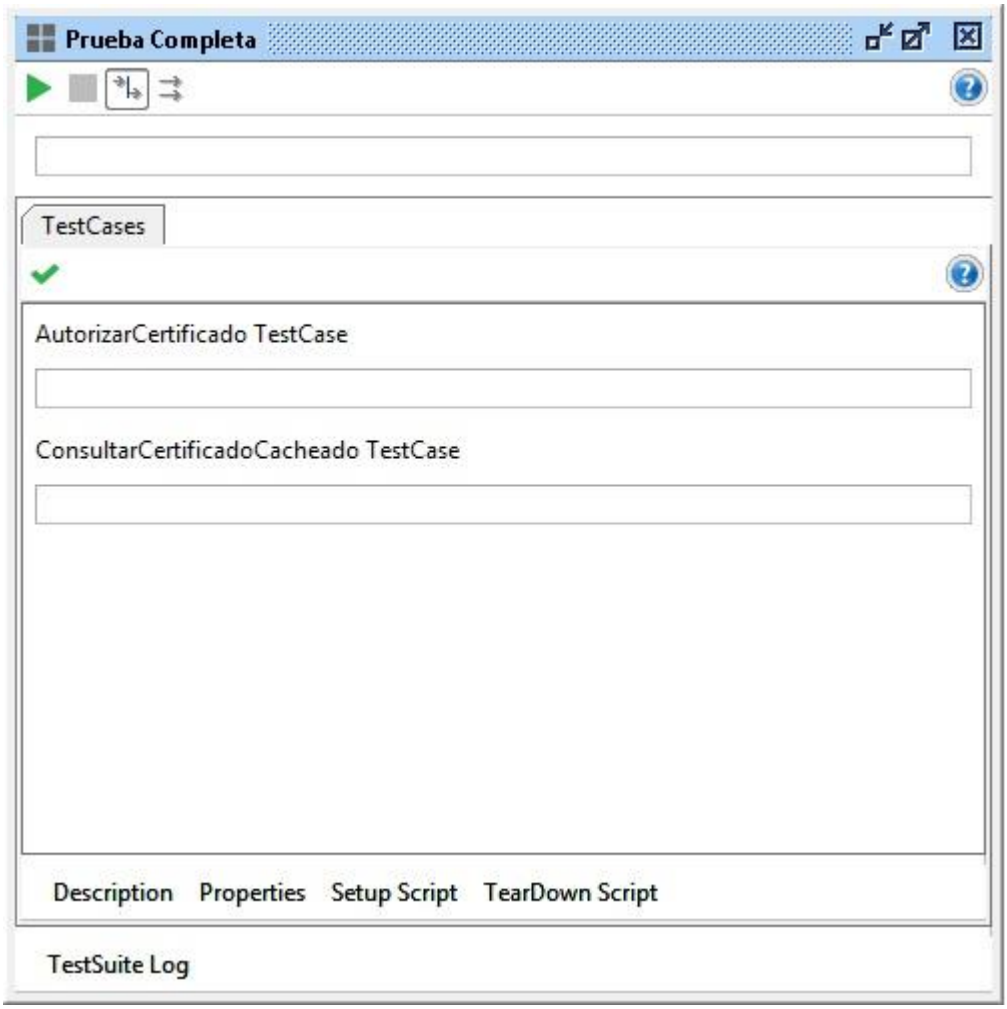

Ilustración 16: TestSuite

Si pulsamos cualquiera de los dos o expandimos el árbol de la derecha veremos los “TestSteps” que en este caso no son otra cosa que las llamadas incluidas en cada TestCase.

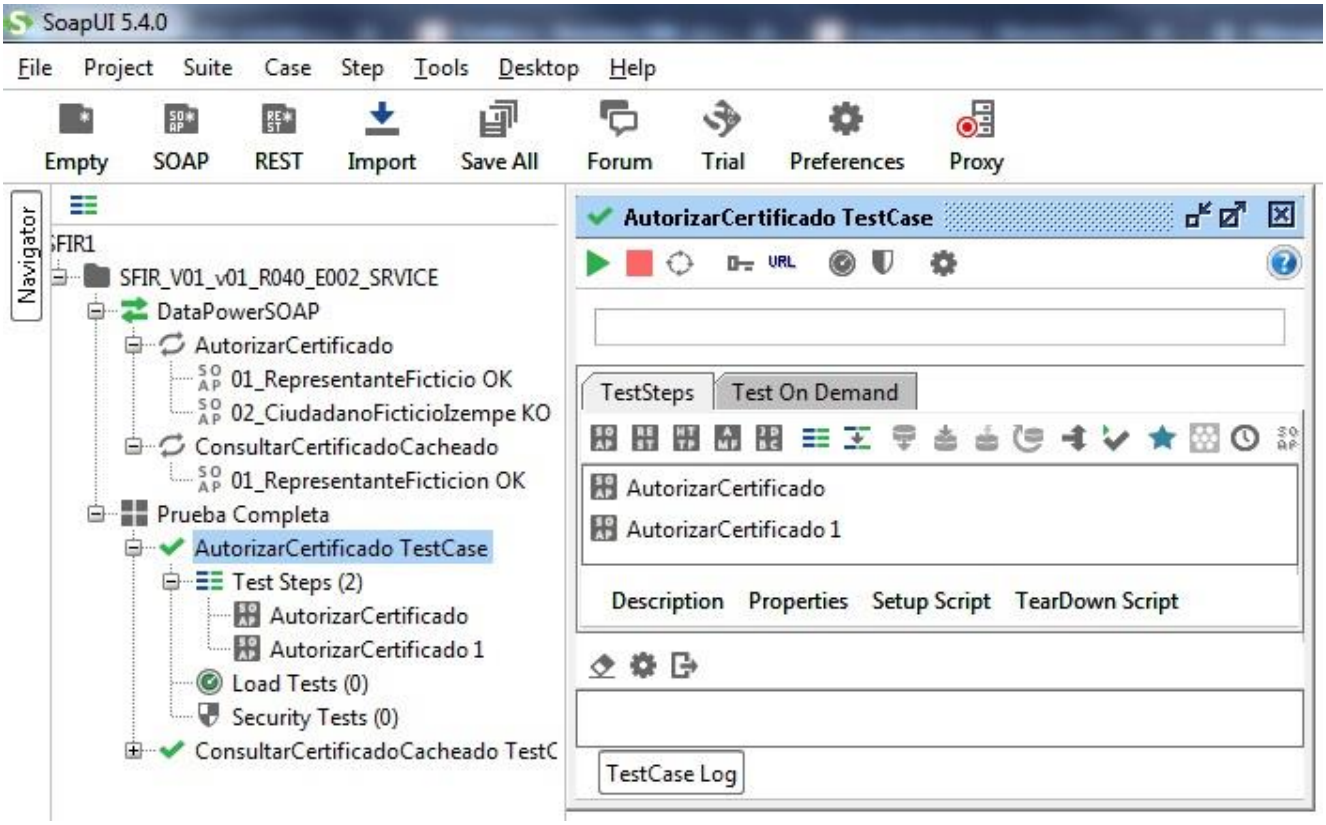

Ilustración 17: Inspección de los TestCase

Como comentábamos anteriormente los nombres de los “TestSteps” son genéricos, para saber que realiza cada uno los renombraremos con el nombre de la petición a partir de la cual se han generado. El renombrado se hace igual que el renombrado de las peticiones visto anteriormente. En este caso quedaría así:

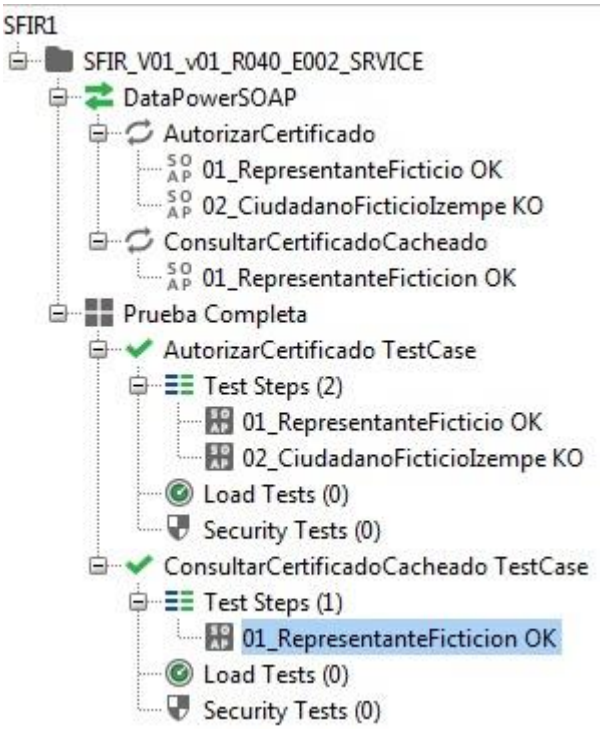

Ilustración 18: TestSteps con nombres definitivos

Si seleccionamos cualquiera de ellas aparecerá una ventana desde la que podremos lanzar la petición de la misma forma que en el apartado 2.1. Sin embargo, en la parte de abajo incluye una pestaña llamada “Assertions”.

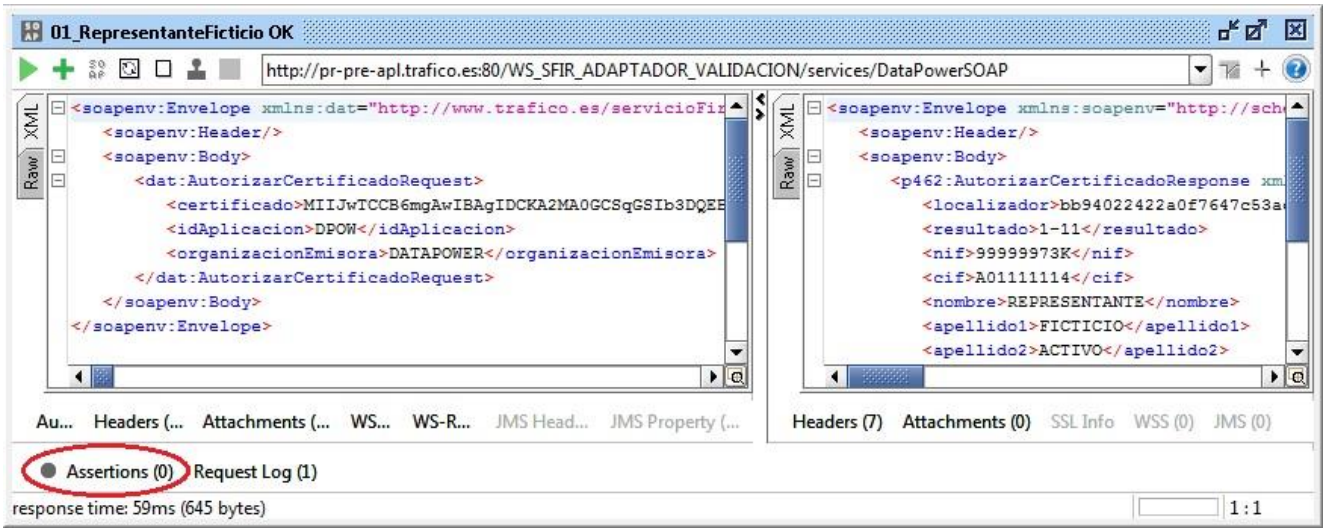

Ilustración 19: Pestaña "Assertions"

2.2.1 Aserciones para una petición correcta

Las aserciones son la característica más potente que tiene SoapUI para realizar baterías de pruebas. Normalmente cuando se lanza una petición, la respuesta se analiza a mano, como en el apartado 2.1. Mediante las aserciones se puede indicar a SoapUI que la respuesta de un servicio debe cumplir con determinados criterios. De esta forma SoapUI marcará en verde las llamadas que cumplan con los criterios definidos en las aserciones y en rojo las que no lo hagan. Si no se indica ninguna aserción las llamadas permanecen en gris.

Nota: Las aserciones de SoapUI permiten realizar comprobaciones sobre las respuestas de un servicio. Estas comprobaciones deben adaptarse a cada servicio, en este tutorial se mostrará algunos ejemplos pero es el propietario de cada servicio, gracias a su conocimiento de la aplicación el que debe adaptar las aserciones para comprobar un servicio con el nivel de detalle necesario.

En el caso de la operación AutorizarCertificado hemos visto que la respuesta a la petición “01_RepresentanteFicticio OK” debe ser una respuesta soap y que la etiqueta resultado debe contener el valor “1-11”. Para parametrizar esto en SoapUI pulsamos en el botón “assertions”. Esto abre un menú en la aparte de abajo donde irán apareciendo las aserciones que vayamos añadiendo. Para añadir una aserción pulsamos el símbolo +.

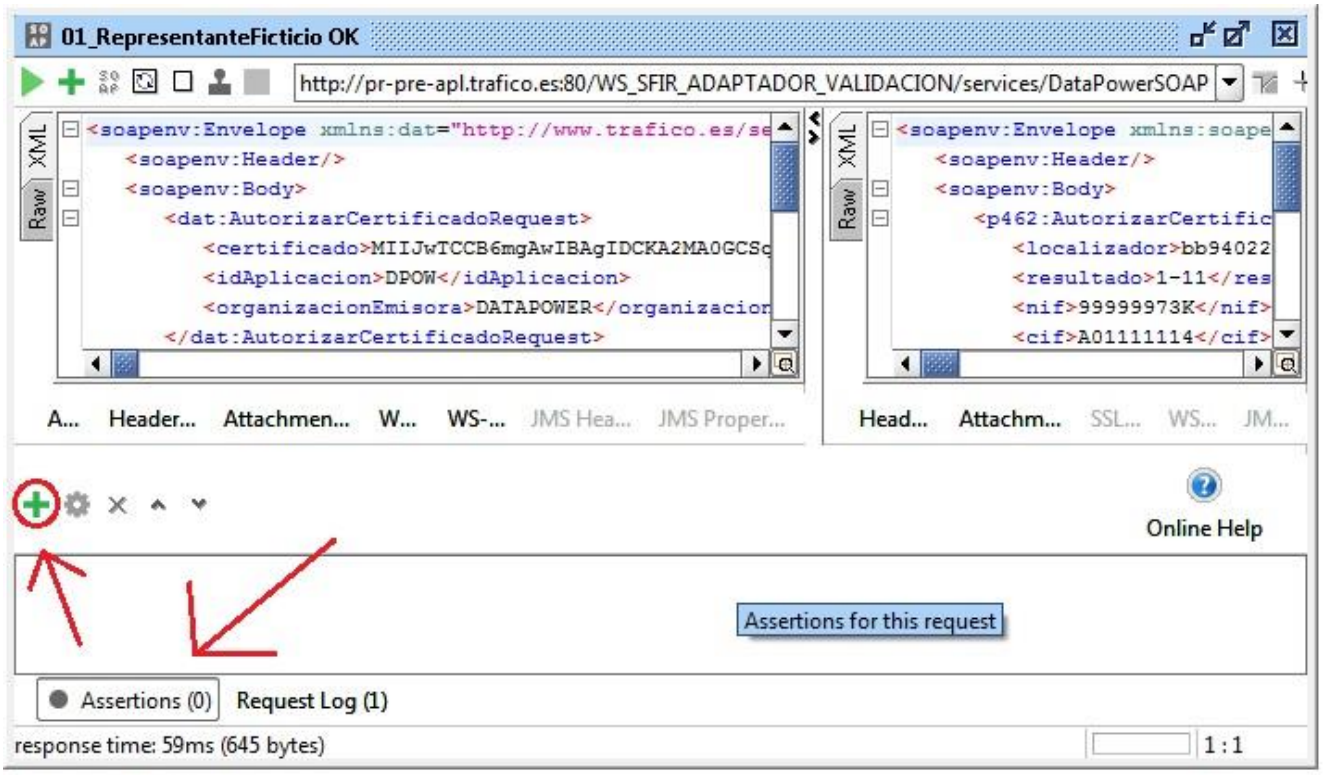

Ilustración 20: Añadiendo aserciones

En la ventana emergente aparecen clasificadas por categorías (izquierda) las aserciones que podemos usar (derecha). Como primera aserción vamos a indicar que el código http de respuesta debe ser un 200. Para ello seleccionamos en la parte izquierda “Compliance, Status and Standards” y seleccionamos la aserción (derecha) “Valid HTTP Status Code”.

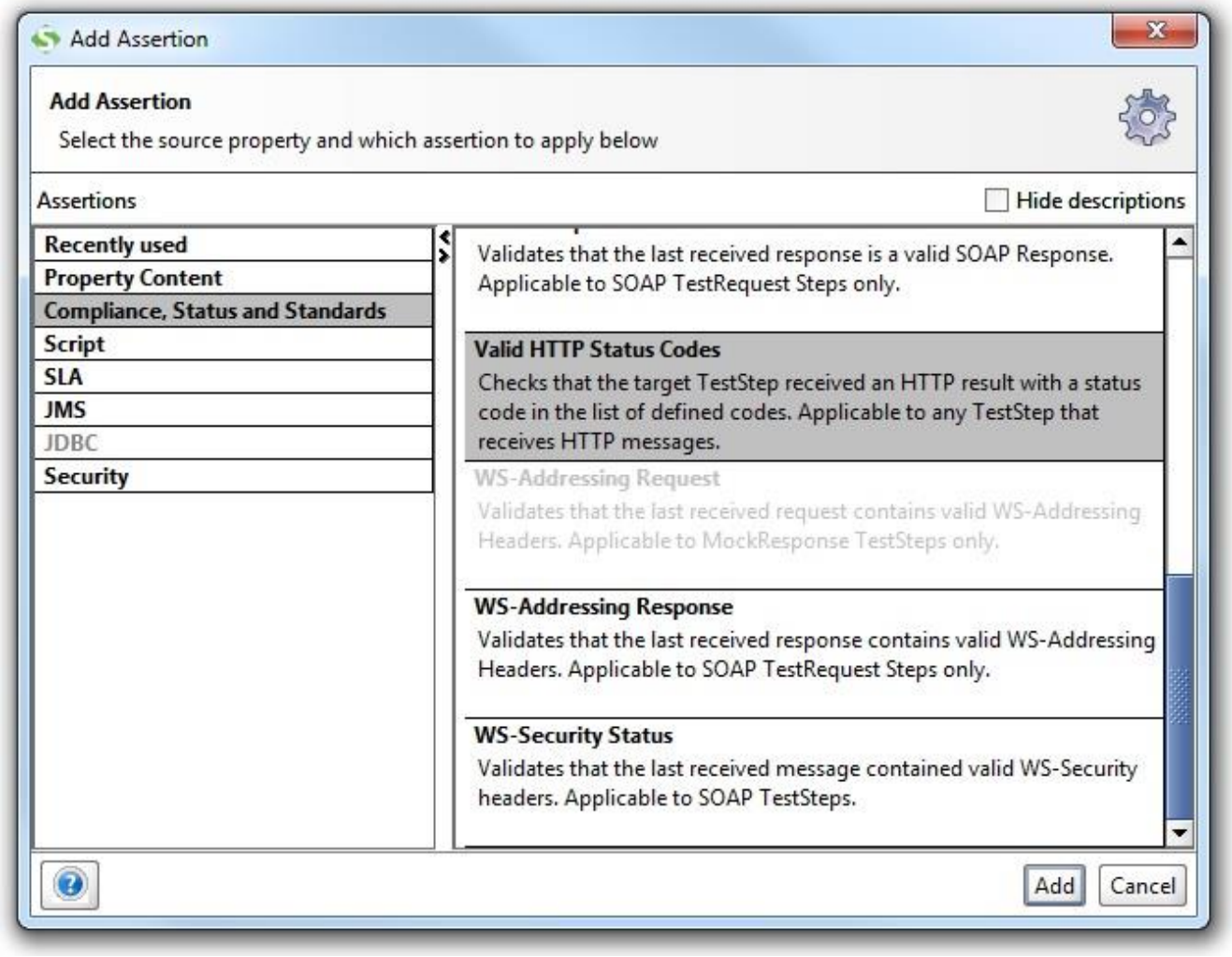

Ilustración 21: Aserción "Valid HTTP Status Codes"

Al pulsar el botón “Add” aparecerá una ventana preguntando el código HTTP válido (se pueden especificar varios)

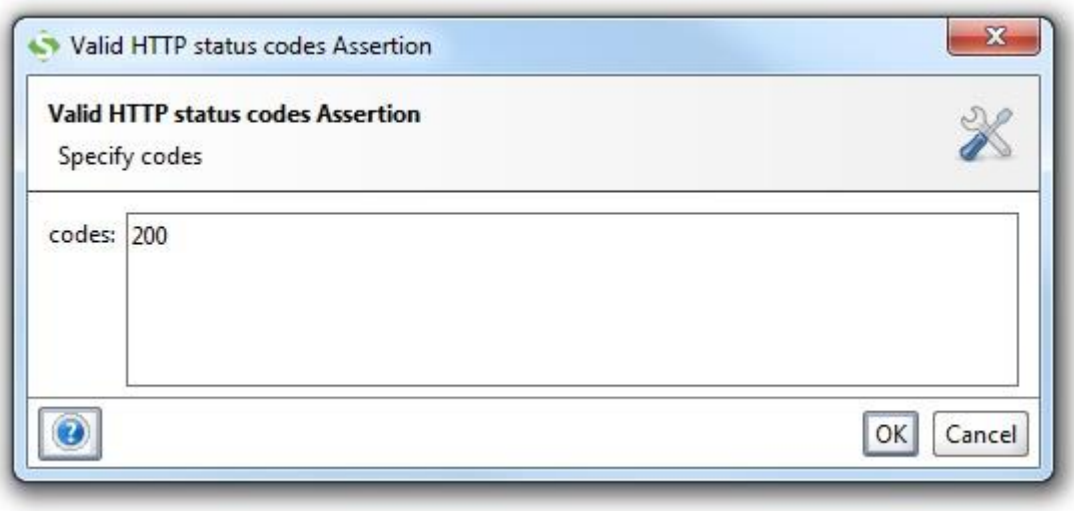

Ilustración 22: Especificando código HTTP

Este tipo de aserciones es muy útil, para detectar fallos. Si la aplicación devuelve un código de error 500, la aserción fallaría he indicaría un mensaje de error. En este caso, como la respuesta es correcta la aserción de cumple. Esto se puede ver ya que el SopaUI marca en verde el "TestStep" en varios puntos de la interfaz.

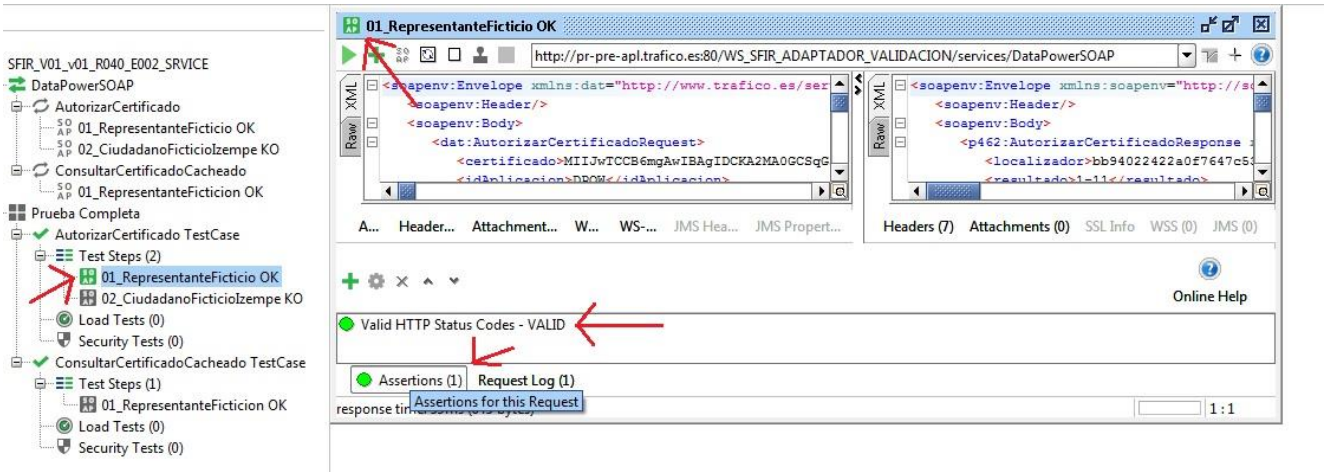

Ilustración 23: Validando TestStep

Siguiendo el mismo procedimiento se pueden añadir más aserciones, estás se irán comprobando en orden por lo que suelen ordenar de las más sencillas a las más complejas.

- Compliance, Status and Standards -> SOAP Response.
- Compliance, Status and Standards -> Not SOAP Fault.
- Compliance, Status and Standards -> Schema Compliance (solicita el WSDL)

Con estas 4 aserciones nos aseguramos de que la respuesta del servicio sea una respuesta SOAP correcta. Sin embargo, hemos visto en el apartado 2.1 que en el caso de que el certificado esté caducado la respuesta del servicio también es una respuesta SOAP correcta. No se devuelve ni un error 500 ni un Soap Fault por ejemplo así que estas 4 aserciones se cumplirían incluso si el certificado a validar no fuera correcto. La diferencia es que en un caso se devuelve en la etiqueta resultado el valor "1-11" y en caso contrario se devuelve un "0". Para realizar esta comprobación usaremos una potente aserción "Property Content" -> "Xpath Match".

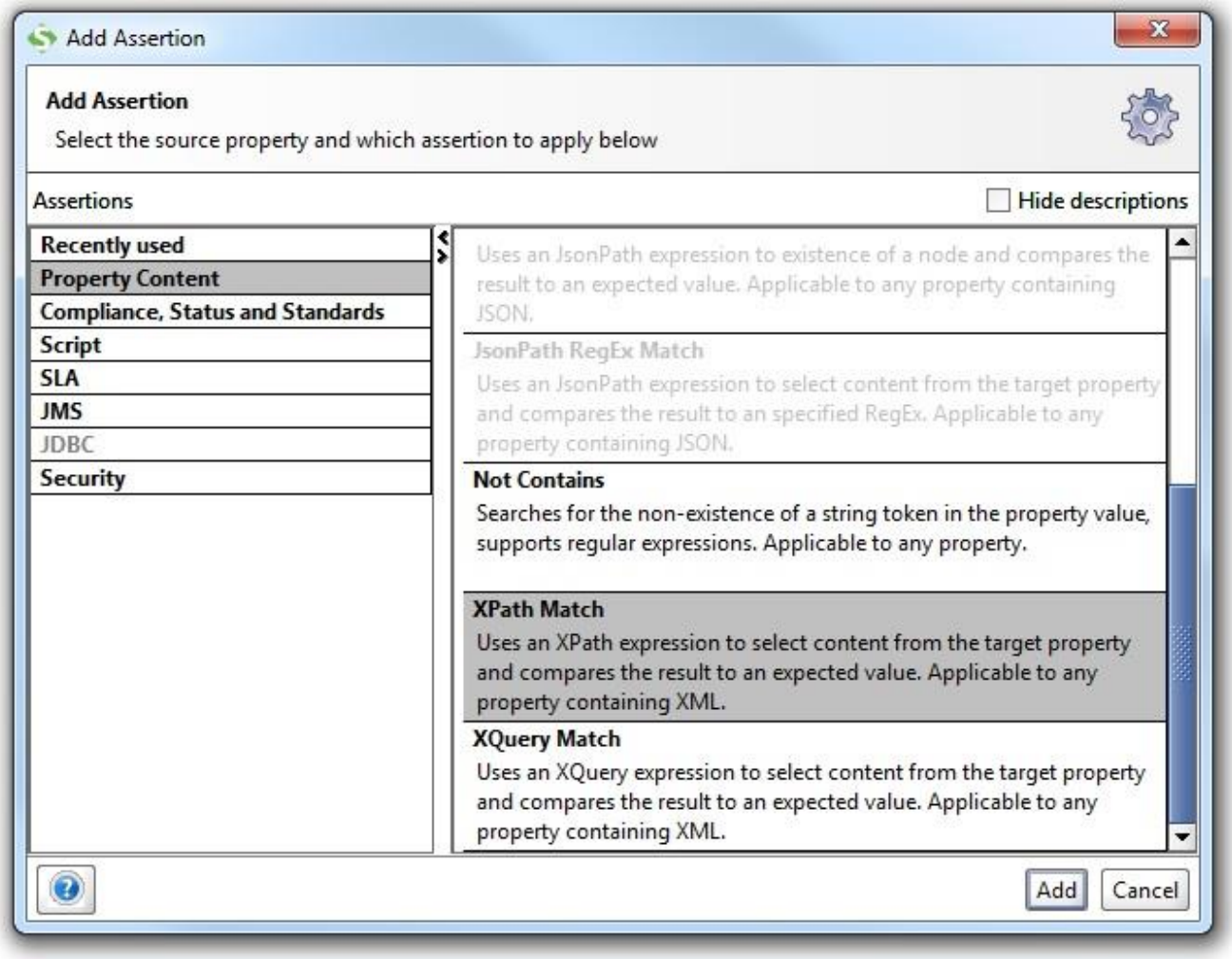

Ilustración 24: La aserción "XPath Match"

Al seleccionar la aserción aparecerá una ventana dividida en dos partes. En la parte de arriba se definen nos namespaces y se indica la ruta Xpath del elemento que queremos comprobar, en este caso la etiqueta resultado. En la parte de abajo se indica el valor que debe cumplir la respuesta. En este caso

sabemos que la respuesta siempre va a ser “1-11” así que así lo indicamos. En resultado sería el siguiente:

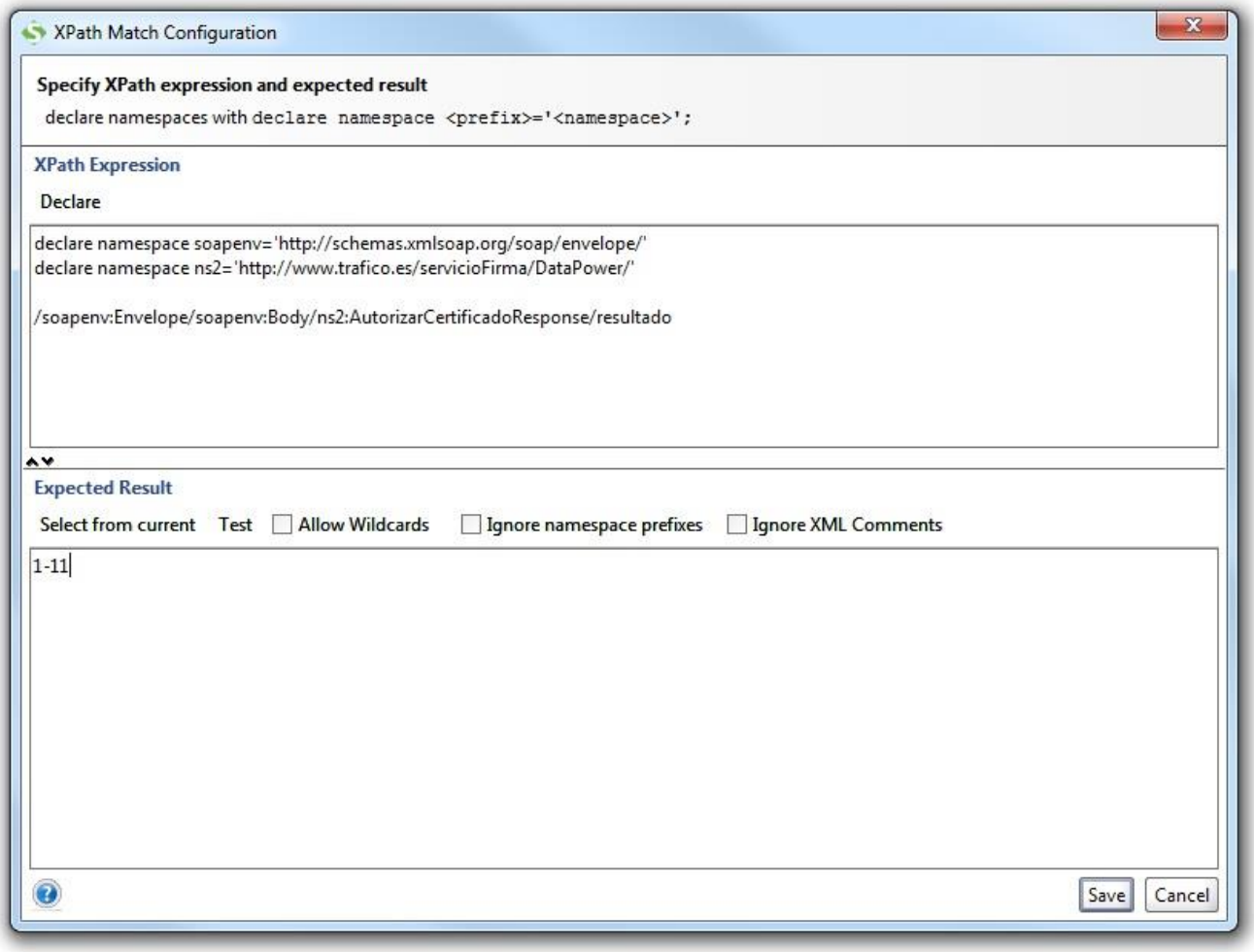

Ilustración 25: Construcción de una expresión XPath

La aserción permite el uso de comodines, por ejemplo, si se quisiera reusar esta misma aserción para diferentes llamadas (clonando el TestStep por ejemplo) se podría seleccionar “Allow wildcards” y establecer como resultado “1-*” sabiendo que independientemente del certificado el resultado debe empezar por un 1. Con estas aserciones se puede comprobar con detalle que la respuesta de la aplicación es correcta.

2.2.2 Aserciones para una petición incorrecta

En el apartado anterior hemos añadido aserciones a un teststep para comprobar que la respuesta de la aplicación es correcta. Igual de importante es probar que al recibir peticiones erróneas o mal

formadas la aplicación responde correctamente. En el caso del TestStep “02_CiudadanoFicticioIzempe KO” se envía un certificado caducado, por este motivo la aplicación responde con un código de resultado 0. Las aserciones para esta petición serían similares a las de la petición correcta, solo cambia el código de resultado. En resumen, las 5 aserciones serían:

- “Compliance, Status and Standards” -> “Valid HTTP Status Code”. Indicando como válido el código 200.
- Compliance, Status and Standards -> SOAP Response.
- Compliance, Status and Standards -> Not SOAP Fault.
- Compliance, Status and Standards -> Schema Compliance (solicita el WSDL)
- “Property Content” -> Xpath Match. Indicando que la etiqueta “resultado” debe valer 0.

Llegados a este punto, tenemos dos TestStep, uno por cada llamada con sus correspondientes aserciones. Ambos TestStep los hemos generado a partir de llamadas del proyecto, pero se pueden crear de 0. Para ver cómo se puede crear un Teststep de cero vamos a crear un TestStep con un certificado corrupto de forma que se genere un error no controlado. Así también tendremos un ejemplo de una respuesta de error no controlado. Es importante comprobar que las aplicaciones resisten llamadas incorrectas sin bloquearse o caerse. En estos casos las aserciones son diferentes pero la mecánica es la misma.

Para añadir un TestStep de cero hacemos click derecho en la lista de Teststeps y seleccionamos “Add TestStep” -> “SOAP Request”.

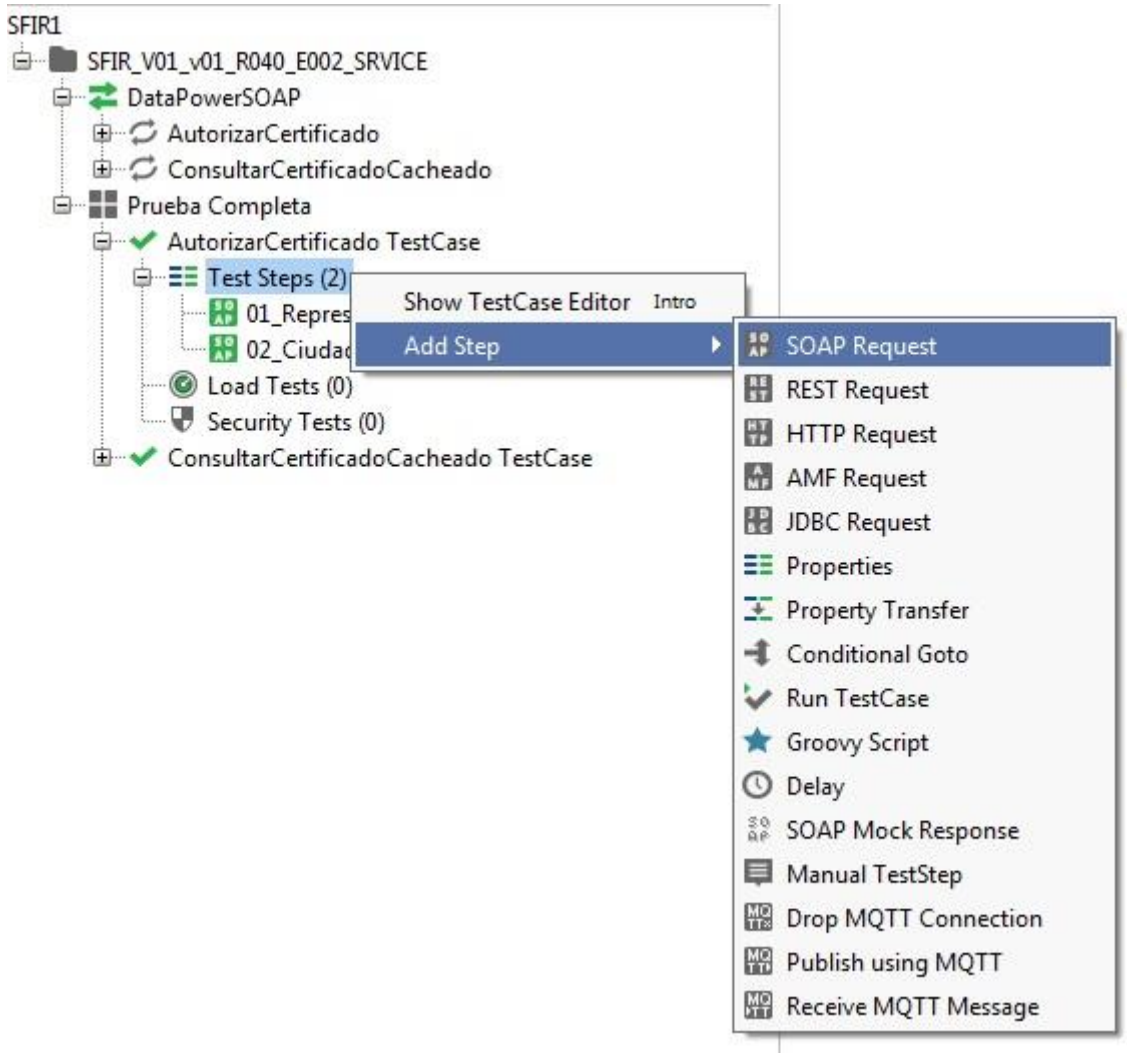

Ilustración 26: Menú para añadir un TestStep a un TestCase

Nos solicitara un nombre, en este caso llamaremos al TestStep “03_CertificadoCorrupto KO”.

Ilustración 27: Nombrado de un TestStep

A continuación nos preguntará en que operación añadir el TestStep, en este caso la operaciones “AutorizarCertificado”

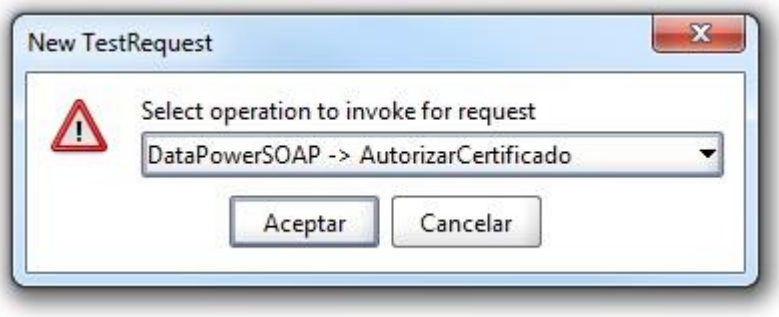

Ilustración 28: Operación asociada a un TestStep

Y para finalizar aparecerá una ventana donde nos volverá a solicitar el nombre (aunque ya debería estar bien cumplimentado) y nos preguntará si queremos añadir algunas aserciones genéricas. Como las añadiremos a mano más adelante dejamos todo sin marcar.

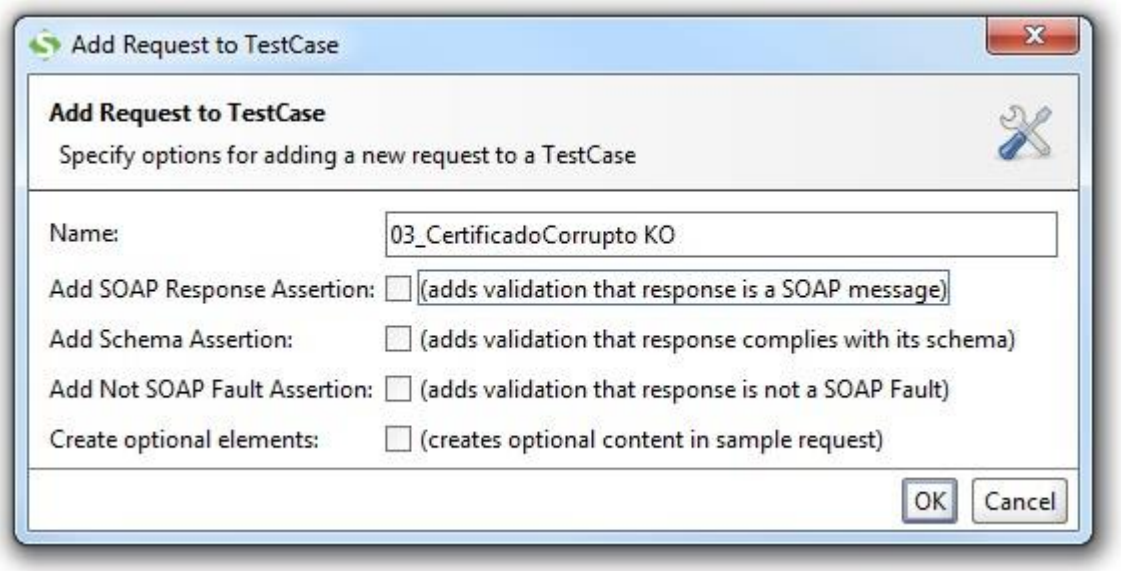

Ilustración 29: Opciones al añadir un TestStep

Al pulsar OK ya se habrá creado un TestStep con una llamada genérica.

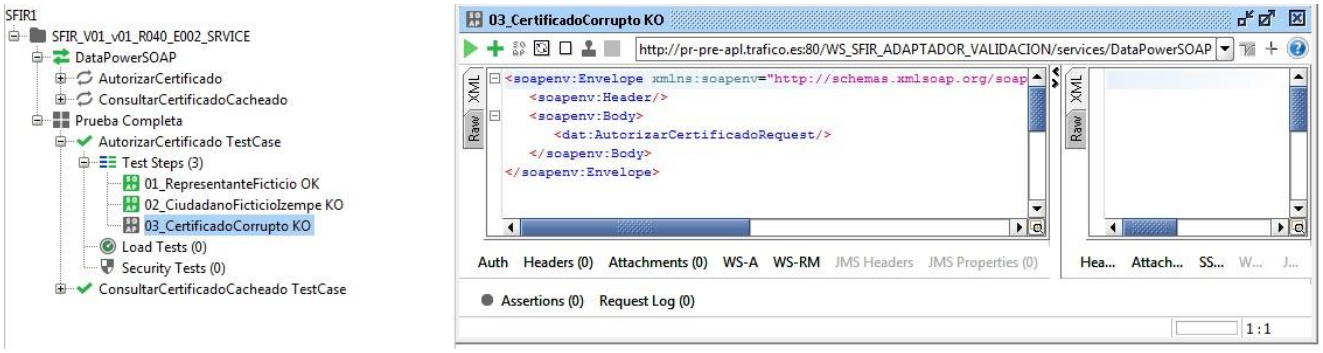

Ilustración 30: El TestStep 03_CertificadoCorrupto KO

Para continuar seleccionamos el TestStep "02_CiudadanoFicticioIzempe KO" y copiamos el contenido de la llamada. Con la llamada en el portapapeles seleccionamos el TestStep "03_CertificadoCorrupto KO" y pegamos la petición. Para simular un certificado corrupto sustituiremos el contenido de la etiqueta certificado por la cadena "tutorial" y lanzaremos la llamada.

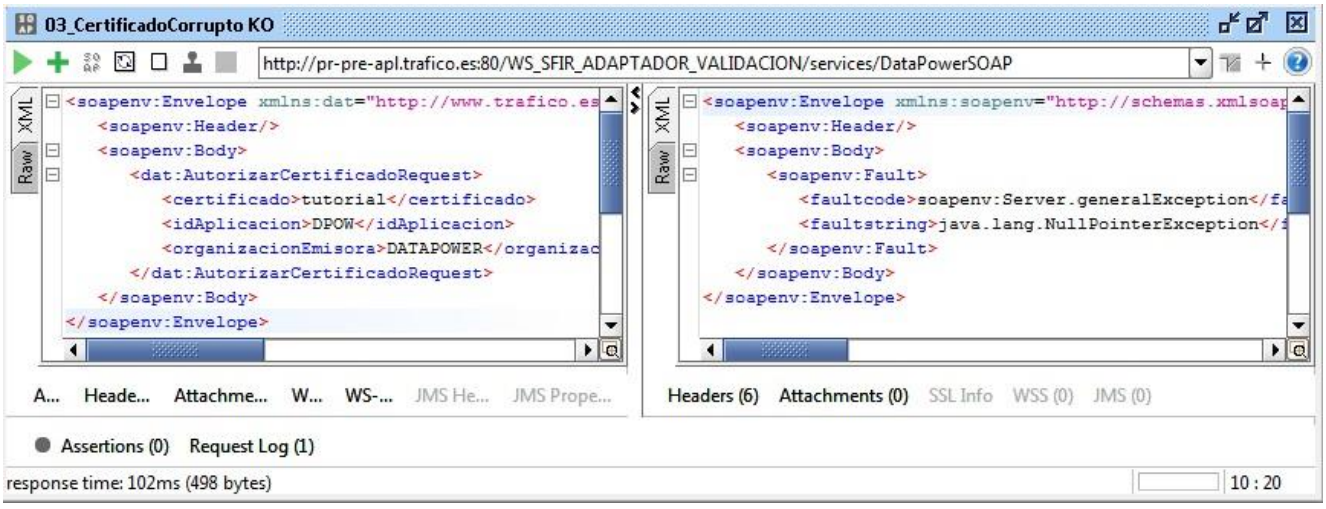

Ilustración 31: Añadiendo contenido de la petición al TestStep

Podemos ver que en este caso la respuesta del servicio es un error no controlado:

```
<soapenv:Envelope>
   <soapenv:Header/>
   <soapenv:Body>
      <soapenv:Fault>
         <faultcode>soapenv:Server.generalException</faultcode>
         <faultstring>java.lang.NullPointerException</faultstring>
      </soapenv:Fault>
   </soapenv:Body>
</soapenv:Envelope>
```

Tabla 3. Respuesta errónea de SFIR1

Si además seleccionamos la pestaña “Raw” podemos ver que el código http de respuesta es un 500 (Internal server error)

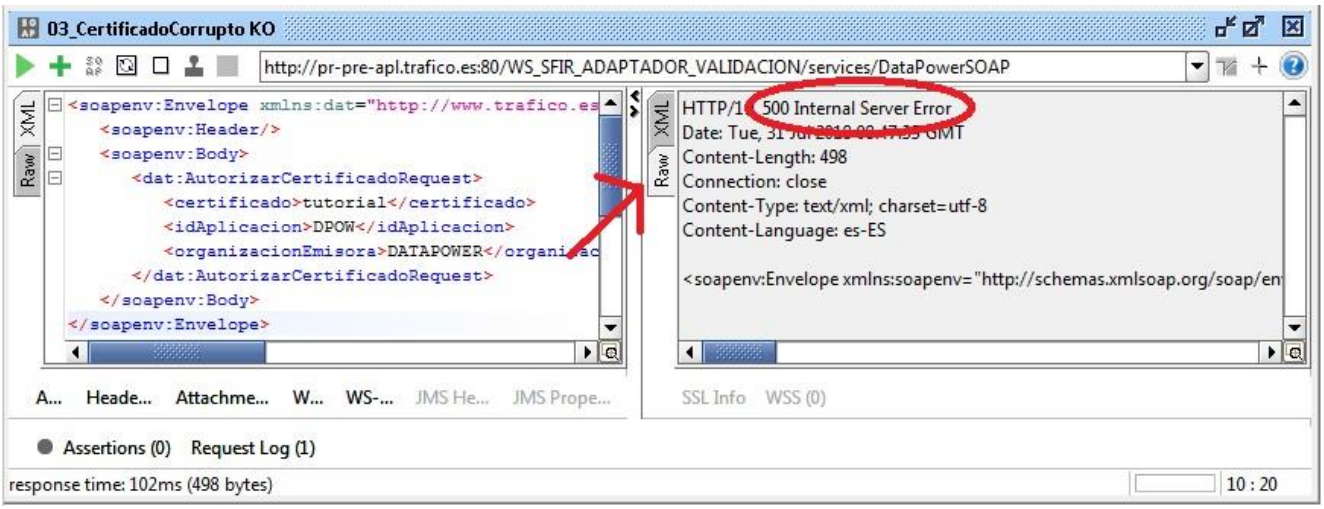

Ilustración 32: Código HTTP de respuesta de SFIR1

En este caso, como lo que esperamos es un error no controlado de la aplicación las aserciones son ligeramente distintas:

- “Compliance, Status and Standards” -> “Valid HTTP Status Code”. Indicando como válido el código 500.
- Compliance, Status and Standards -> SOAP Response.
- Compliance, Status and Standards -> SOAP Fault (en contraposición a la aserción “Not Soap Fault” usada hasta ahora).
- Compliance, Status and Standards -> Schema Compliance (solicita el WSDL)
- “Property Content” -> “Xpath Match”. Indicando que la etiqueta “faultstring” debe valer “java.lang.NullPointerException”.

Aquí se muestra como se construiría la expresión xpath:

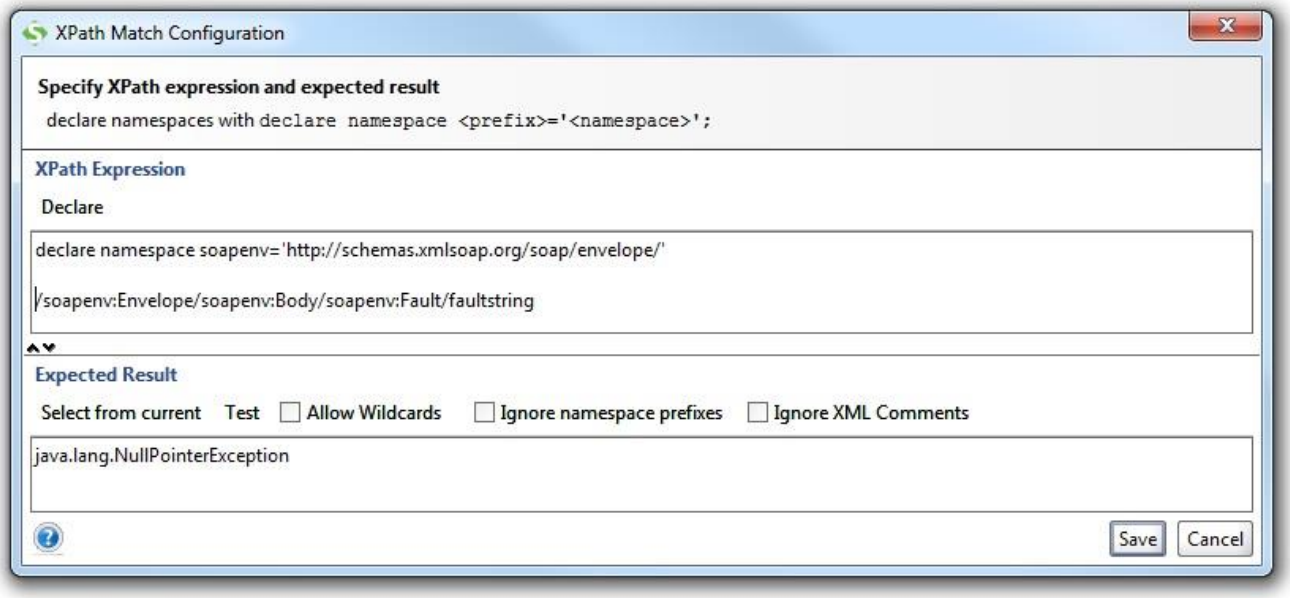

Ilustración 33: Expresión XPath para un SoapFault

Llegados a este punto ya tenemos tres TestStep para la operación "Autorizarcertificado". Nos faltaría añadir las aserciones para la operación "ConsultarCertificadoCacheado" y completar con llamadas erróneas como hemos hecho con la operación "AutorizarCertificado". El procedimiento es el mismo que el seguido con la operación "Autorizarcertificado". En este caso tendremos dos llamadas o TestStep, "01_RepresentanteFicticio OK" y "02_RepresentanteFicticio KO". Para la llamada "01_RepresentanteFicticio OK" usaremos las aserciones:

- "Compliance, Status and Standards" -> "Valid HTTP Status Code". Indicando como válido el código 200.
- Compliance, Status and Standards -> SOAP Response.
- Compliance, Status and Standards -> Not SOAP Fault.
- Compliance, Status and Standards -> Schema Compliance (solicita el WSDL)
- "Property Content" -> Xpath Match. Indicando que la etiqueta "certificadoValido" debe valer "true".

La construcción del XPath sería la siguiente:

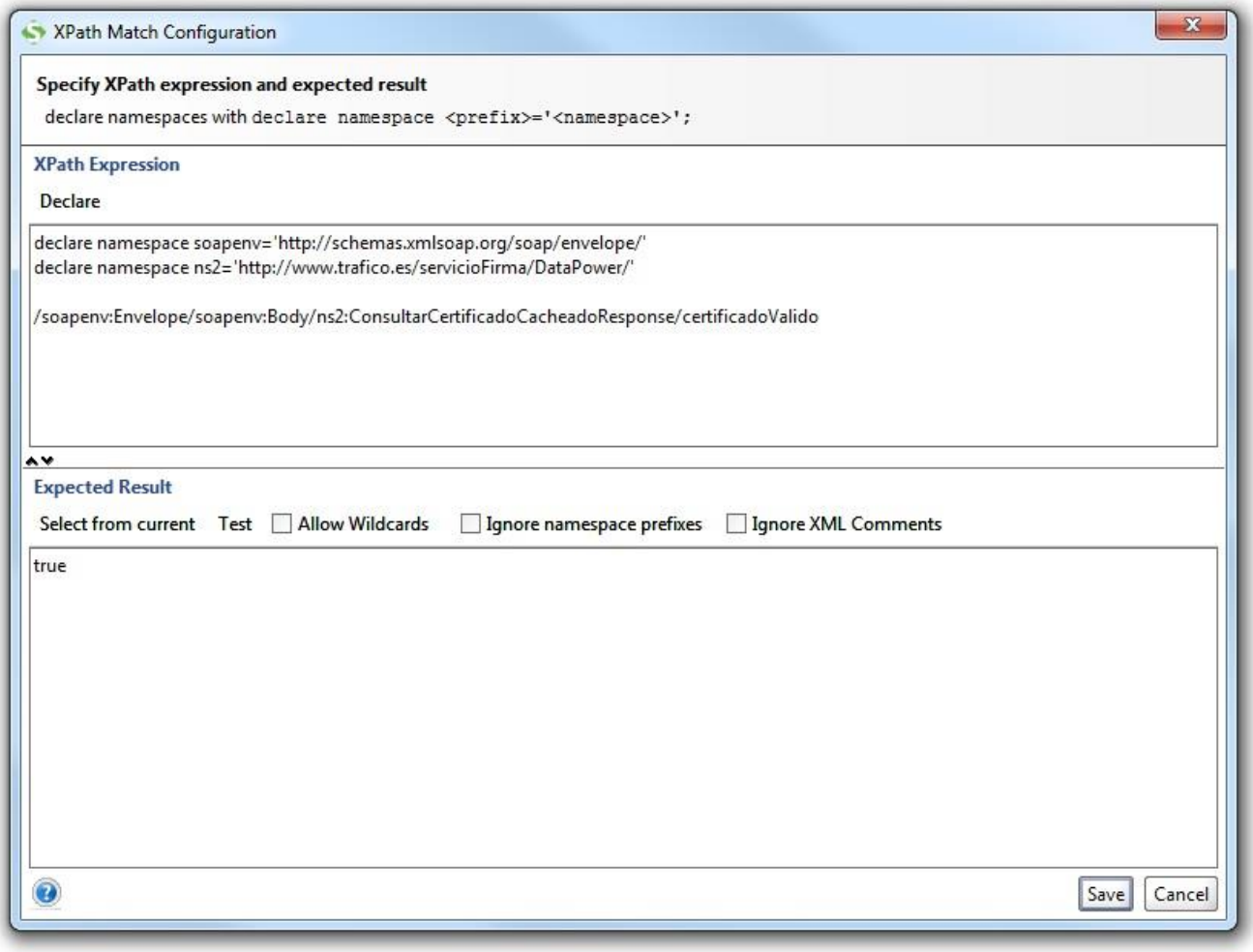

Ilustración 34: Expresión XPath para la operación ConsultarCertificadoCacheado

Las aserciones del TestStep "02_RepresentanteFicticio KO" de la operación ConsultarCertificadocacheado son exactamente las mismas que para la petición "03_CertificadoCorrupto KO" de la operación AutorizarCertificado.

2.3 Realizando las Pruebas

Una vez ya tenemos la batería de pruebas, lanzarla es muy sencillo. Se puede seleccionar la batería completa (TestSuite) haciendo doble click sobre ella en el árbol de la parte izquierda de la pantalla.

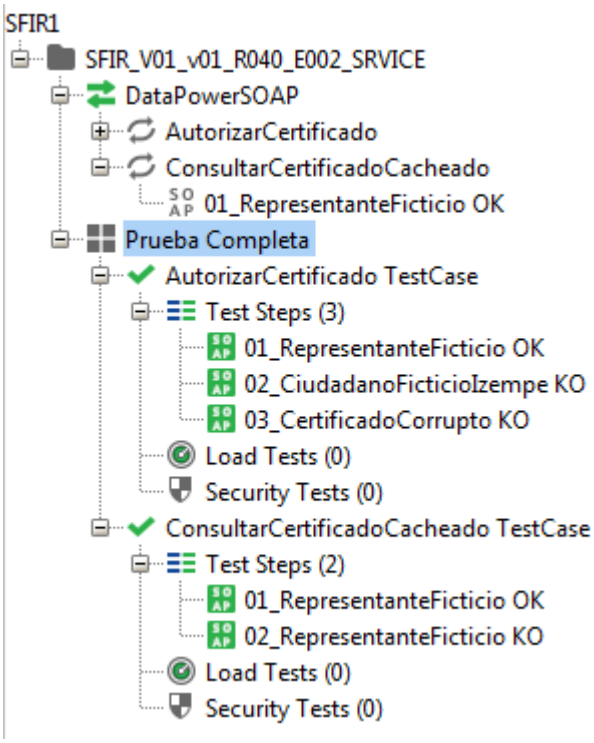

Ilustración 35: Selección de una batería de pruebas (TestSuite)

Al hacerlo aparecerá en la parte derecha de la pantalla una ventana con todos los TestCase, en este caso “AutorizarCertificado” y “ConsultarCertificadoCacheado”

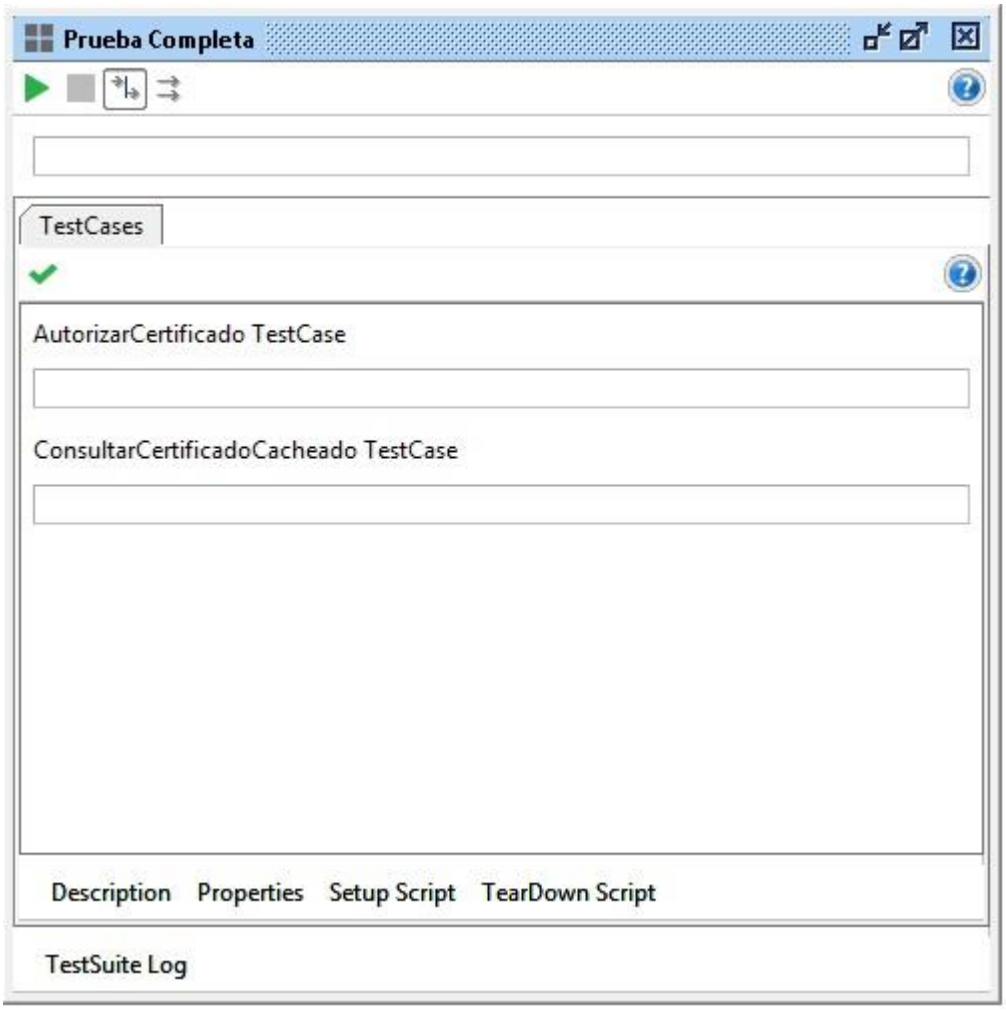

Ilustración 36: Ventana de una batería de pruebas (TestSuite)

Si pulsamos el botón “Play” de la parte superior derecha, SoapUI lanzará por orden todas las peticiones (TestSteps) configuradas en ambos TestCase e indicará con una barra de progreso en verde o en rojo si las pruebas han finalizado correctamente o no. En este caso como las pruebas son correctas nos aparecerán dos barras verdes en la parte inferior indicando que ambos TestCase han terminado correctamente y una barra verde en la superior indicando, que en general todo ha ido bien.

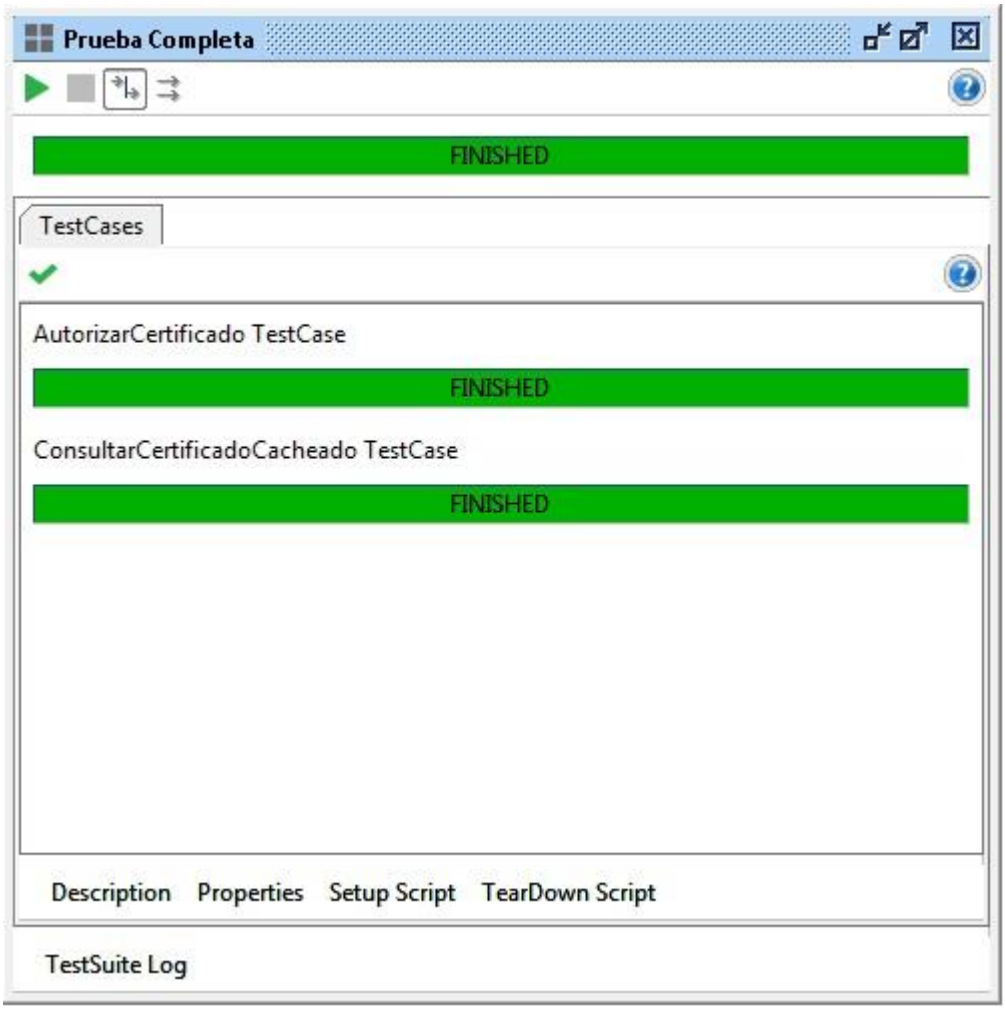

Ilustración 37: Prueba con éxito

En el caso en el que alguna llamada no cumpliese sus aserciones, las barras de progreso se volverían rojas, si fallase una llamada del método “ConsultarCertificadoCacheado” el aspecto sería el siguiente:

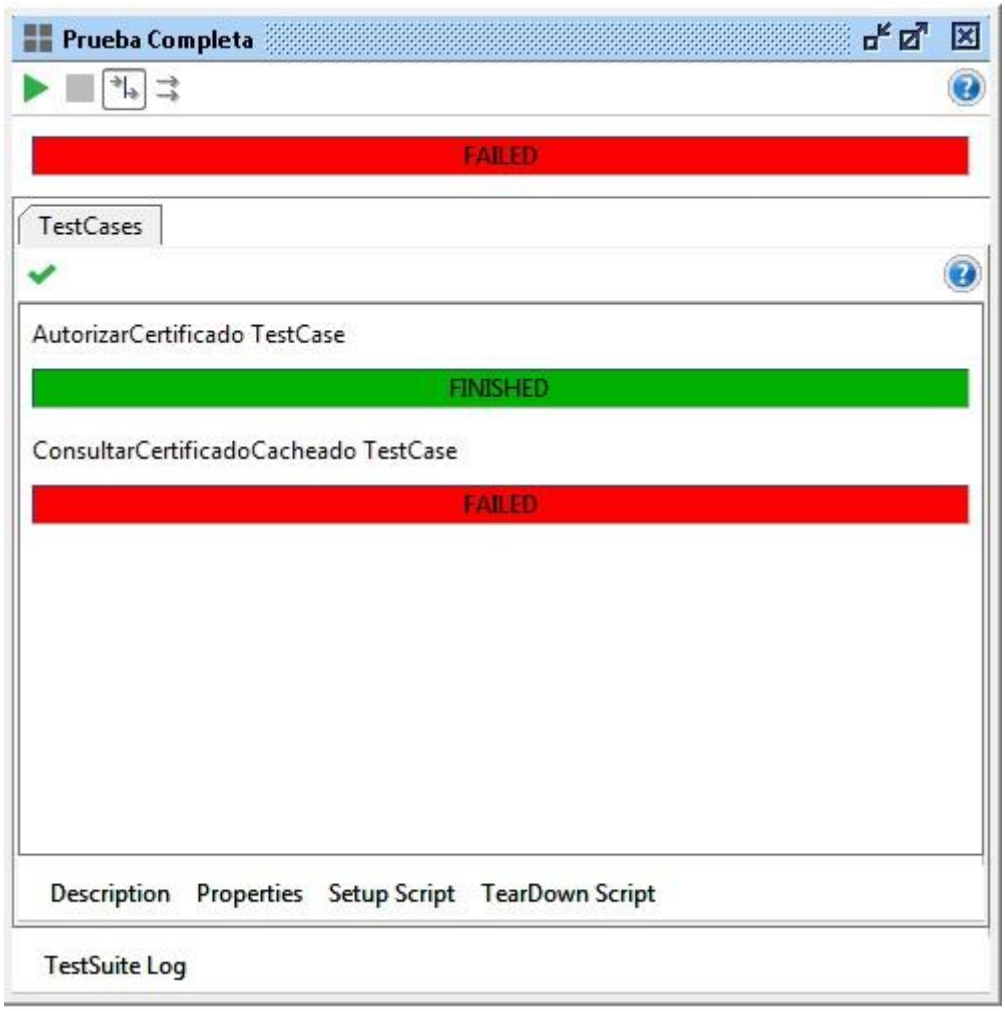

Ilustración 38: Prueba con errores

Haciendo doble click en el TestCase se puede ver exactamente que llamada falló coloreando su icono en rojo:

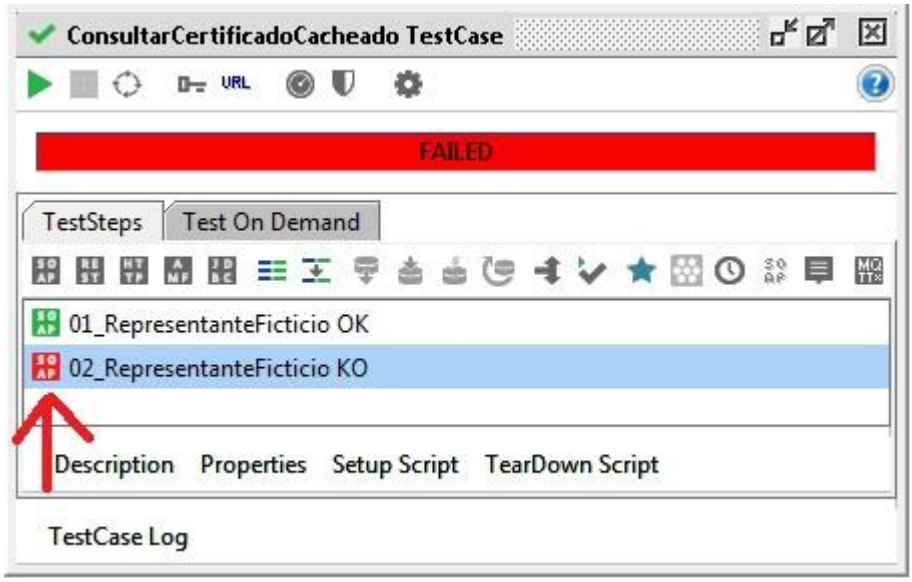

Ilustración 39: TestStep erróneo

Si hacemos doble click en la petición se puede ver la respuesta y exactamente que aserción no se ha cumplido, en este caso el Xpath Match (modificado expresamente para que fallara).

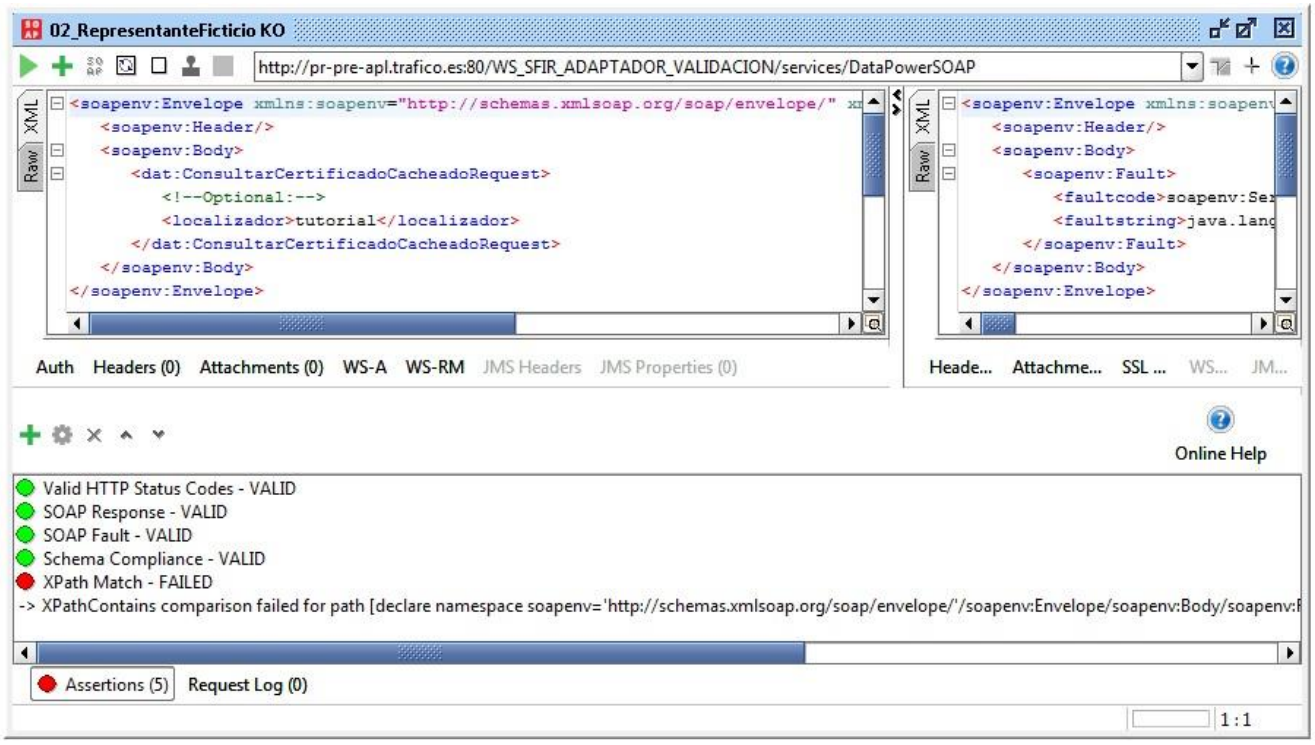

Ilustración 40: Fallo en aserción

Con esta batería de pruebas, ya se podría probar la operación DatapowerSOAP de SFIR1 fácilmente y con un solo click.

3 Exportando/Importando los proyectos

3.1 Exportando un proyecto

Es habitual tener que mover un proyecto de SoapUI con sus llamadas, certificados, pruebas o aserciones de un PC a otro. Para hacerlo fácilmente existe la opción de exportar. Para exportar un proyecto lo seleccionamos en el árbol de la izquierda mediante click derecho y seleccionamos la opción “Export Proyect”:

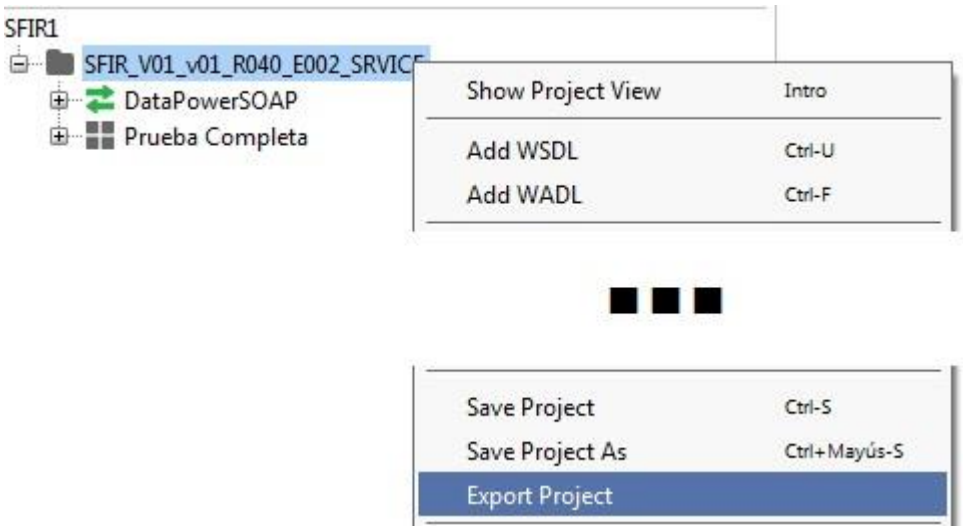

Ilustración 41: Menú exportar proyecto

SoapUI incluirá la configuración y los ficheros necesarios, como los certificados en un fichero zip.

3.2 Importando un proyecto

Para importar un proyecto, haremos click derecho en el workspace y seleccionaremos la opción “Import Packet Project”

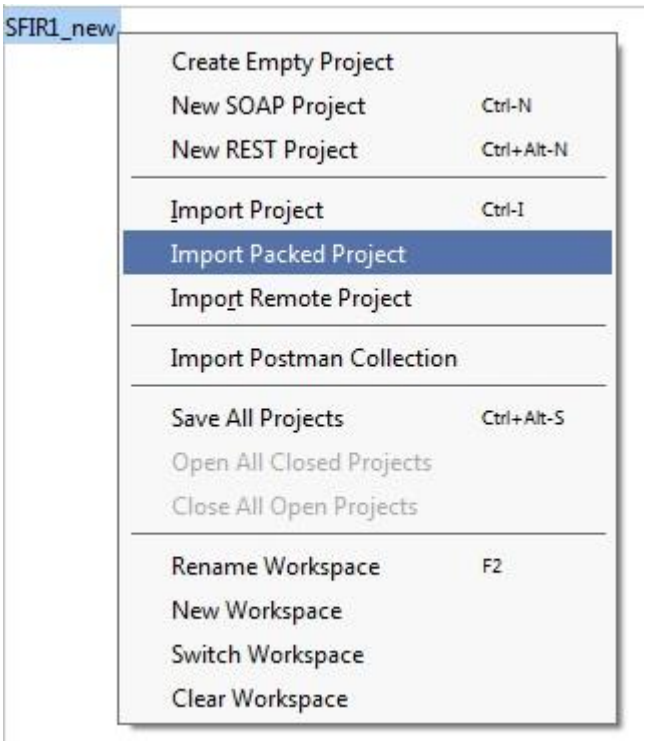

Ilustración 42: Menú importar proyecto

Al hacerlo aparecerá una ventana solicitando la ruta del fichero zip donde está el proyecto. Una vez seleccionado el fichero aparecerá una segunda ventana preguntando donde se desea guardar el proyecto. Una vez seleccionada la carpeta, SoapUI descomprime el zip y modifica las dependencias de forma que se pueda utilizar en la nueva ubicación:

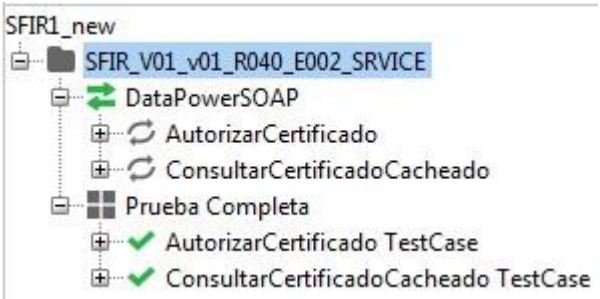

Ilustración 43: Proyecto importado

4 Anexos

4.1 Autenticación WS-Security

4.1.1 Autenticación por Certificado

Durante el tutorial se han realizado llamadas y pruebas con SFIR1. SFIR1 no exige seguridad de ningún tipo pero muchas aplicaciones de la DGT exigen autenticación por certificado. La autenticación por certificado, realmente es una firma. De la misma forma que al comprobar un usuario y una contraseña lo que se hace es comprobar que un dato secreto, la contraseña, corresponde con el usuario, un dato público. En el caso de la autenticación por certificado lo que se comprueba es que la firma del documento, algo para lo que se necesita la clave privada del certificado (parte secreta) corresponde con la clave pública que se incluye en la llamada.

Al ser una firma, cada vez que se modifica la petición es necesario volver a firmar la petición. Por este motivo es muy cómodo trabajar con la petición sin firmar y que sea el SoapUI el que firme las peticiones antes de enviarlas.

Para este ejemplo vamos a utilizar el servicio de @firma ValidateCertificate que exige autenticación por certificado. El servicio se expone en esta URL:

https://des-afirma.redsara.es/afirmaws/services/ValidateCertificate

y es el que utiliza SFIR para comprobar en la operación “AutorizarCertificado” si un certificado el válido o no.

Como se ha comentado con anterioridad es interesante probar las dependencias de una aplicación, en este caso, si la operación “ValidateCertificate” de @firma fallase por cualquier motivo, la operación “AutorizarCertificado” de SFIR1 también fallaría. Comprobando ambas se puede distinguir fácilmente si el fallo de la operación “AutorizarCertificado” es debido a un fallo de SFIR1 o a un fallo en @firma.

Para probar el servicio “ValidateCertificate” de @firma crearemos un proyecto nuevo dentro del mismo Workspace de SFIR1 de la misma forma que en el apartado 2.1 y utilizaremos la URL del servicio para cargar el wsdl directamente.

Para ello haremos click derecho en el proyecto “SFIR_V01_v01_R040_E002_SRVICE” y seleccionaremos “Add WSDL”. En la ventana emergente:

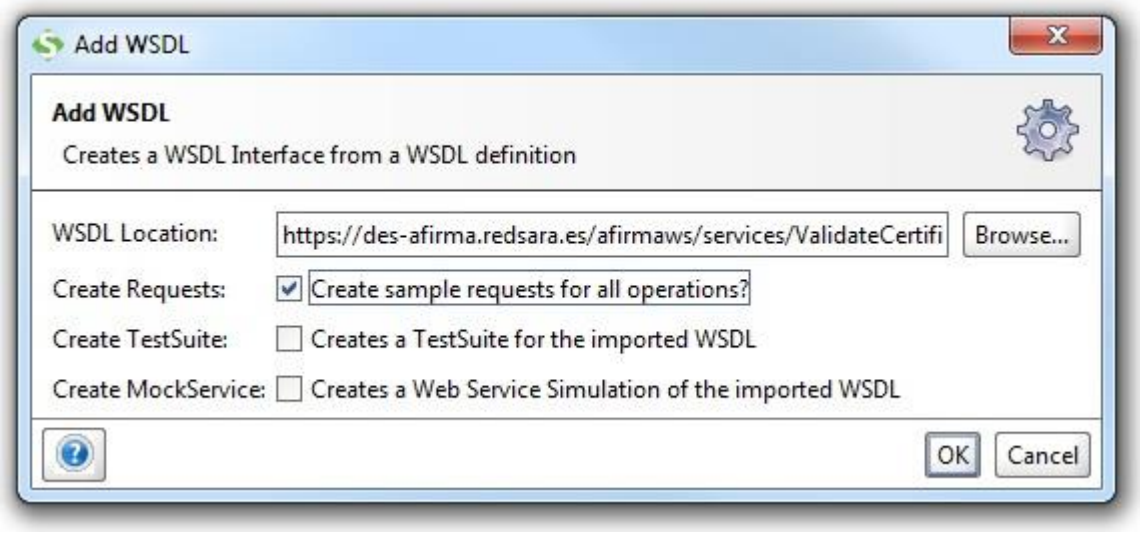

Ilustración 44: Añadiendo al proyecto el wsdl de Afirma

Como URL del wsdl usaremos la URL del servicio con la coletilla “?wsdl”:
https://des-afirma.redsara.es/afirmaws/services/ValidateCertificate?wsdl

También seleccionaremos la opción “Create sample request for all operation” para que nos cree una petición de ejemplo. Finalmente pulsaremos OK. Como resultado tendremos una nueva operación SOAP con una petición de ejemplo. Al igual que en el apartado 2.1.1 construiremos una llamada de prueba para el servicio y la renombraremos con el nombre del certificado a probar, así que se volverá a llamar “01_RepresentanteFicticio OK”. Para diferenciar las operaciones de SFIR de las operaciones de Afirma renombraremos la operación “ValidateCertificate” de @firma añadiéndole el prefijo “Afirma_” delante. El resultado final sería el siguiente:

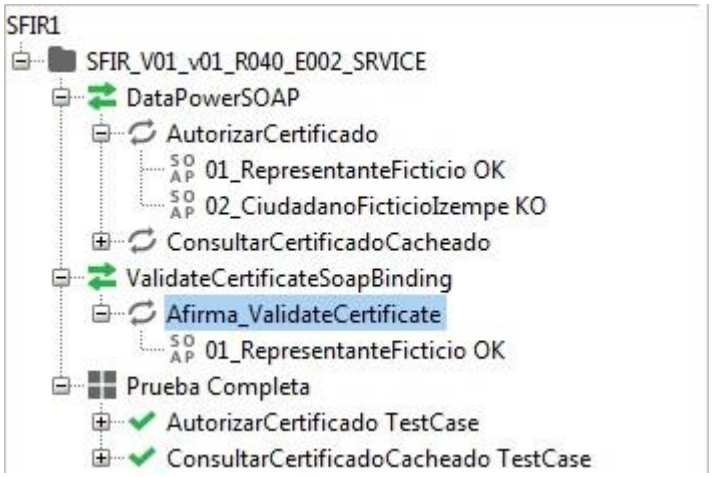

Ilustración 45: Árbol del proyecto Afirma

Como en todos los casos la petición generada por SoapUI no incluye datos para la llamada así que construimos una llamada que valide el certificado de Representante Ficticio:

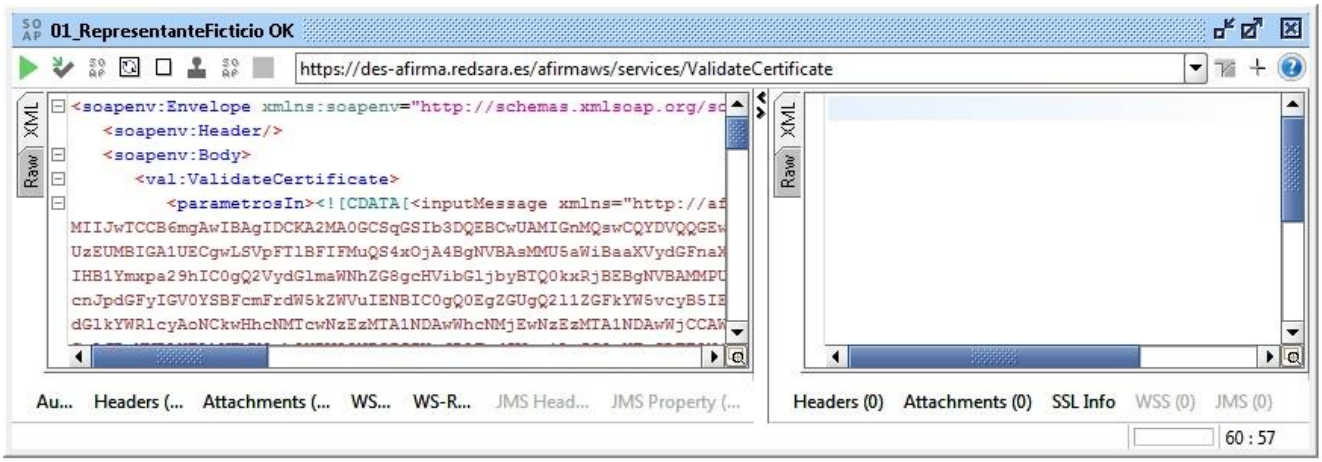

Ilustración 46: Llamada a @Firma

Nota: En este caso, como no somos propietarios de la aplicación, el formato de las llamadas debe extraerse de la documentación y el soporte que nos ofrezca el propietario, en este caso el MINHAP.

Si realizamos la llamada directamente la respuesta de @Firma es la siguiente:

```
<soapenv:Envelope>
   <soapenv:Header></soapenv:Header>
   <soapenv:Body >
      <soapenv:Fault>
         <faultcode>soapenv:Server.userException</faultcode>
```

```
         <faultstring>COD_EXCEPCION COD_117 DESCRIPCION Tag de seguridad
Security no encontrado en la petición SOAP.</faultstring>
         <detail>
            <ns1:hostname
xmlns:ns1="http://xml.apache.org/axis/">sacaeprema01.redsara.es</ns1:hostname
         </detail>
      </soapenv:Fault>
   </soapenv:Body>
</soapenv:Envelope>
```

Tabla 4. Respuesta de @Firma

En el error se puede ver que @Firma exige que el usuario se autentique para acceder al servicio.

Para firmar las peticiones con un certificado primero debemos cargar el certificado en SoapUI. Aunque no es necesario, es recomendable copiar el certificado en la carpeta del proyecto SoapUI antes de cargarlo con el objetivo de tener todos los ficheros juntos. Para cargar el certificado haremos click derecho en proyecto y seleccionaremos “Show Project View”:

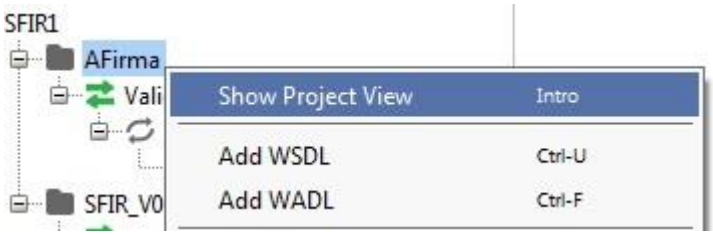

Ilustración 47: Mostrando las propiedades de un proyecto

Aparecerá una ventana con varias pestañas, seleccionaremos “WS-Security Configurations”:

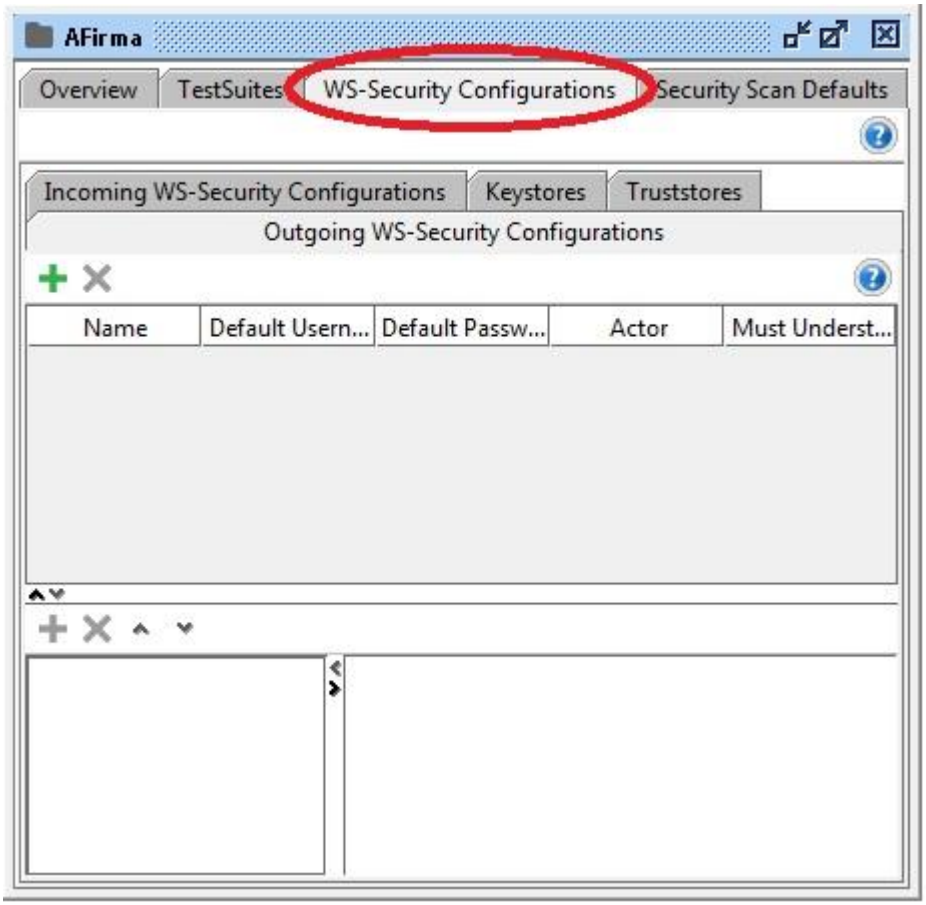

Ilustración 48: Menú "WS-Security Configurations"

En esta ventana realizaremos toda la configuración de seguridad de nuestra petición. Para empezar pulsaremos en la etiqueta “Keystores” y a continuación el símbolo más (+) para añadir nuestro certificado a los almacenes de confianza del SoapUI.

Ilustración 49: Añadiendo un certificado al keystore del proyecto SoapUI

Al pulsar el botón aparecerá una ventana donde se solicitará que indiquemos el fichero que contiene el certificado y a continuación otra solicitará la contraseña del fichero (en este caso “afirma”). Al terminar volveremos a la ventana original, donde ya aparecerá el nuevo certificado. Para asegurarnos de que el certificado se haya cargado correctamente comprobaremos que el campo “Status” marca “OK”:

Ilustración 50: Certificado cargado correctamente en el keystore

Una vez el certificado está cargado en el Keystore seleccionamos la pestaña "Outgoing WS-Security Configurations". En esta ventana se definen las políticas de seguridad del proyecto. Para añadir una nueva política de seguridad pulsamos símbolo más (+) e introducimos un nombre para la política. En este caso la llamaremos Autenticación_Afirma. Al terminar volveremos a la ventana original y veremos que se ha añadido una nueva política. Seleccionándola y pulsando el símbolo más (+) de la parte inferior podremos empezar a configurar la política.

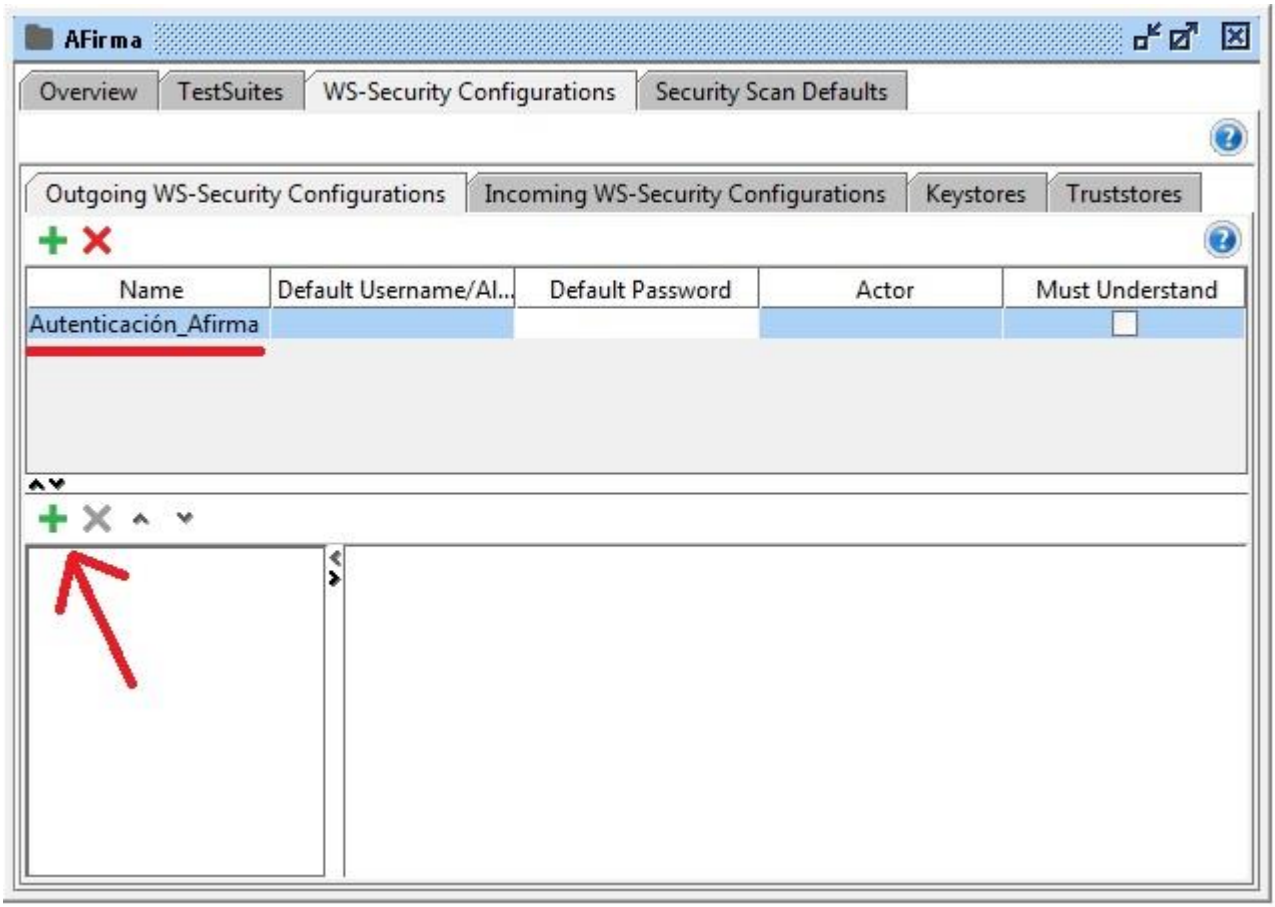

Ilustración 51: Definiendo la política de autenticación de Afirma

Como hemos comentado antes la autenticación por certificado técnicamente es una firma, por eso seleccionaremos la opción “Signature” en el menú desplegable:

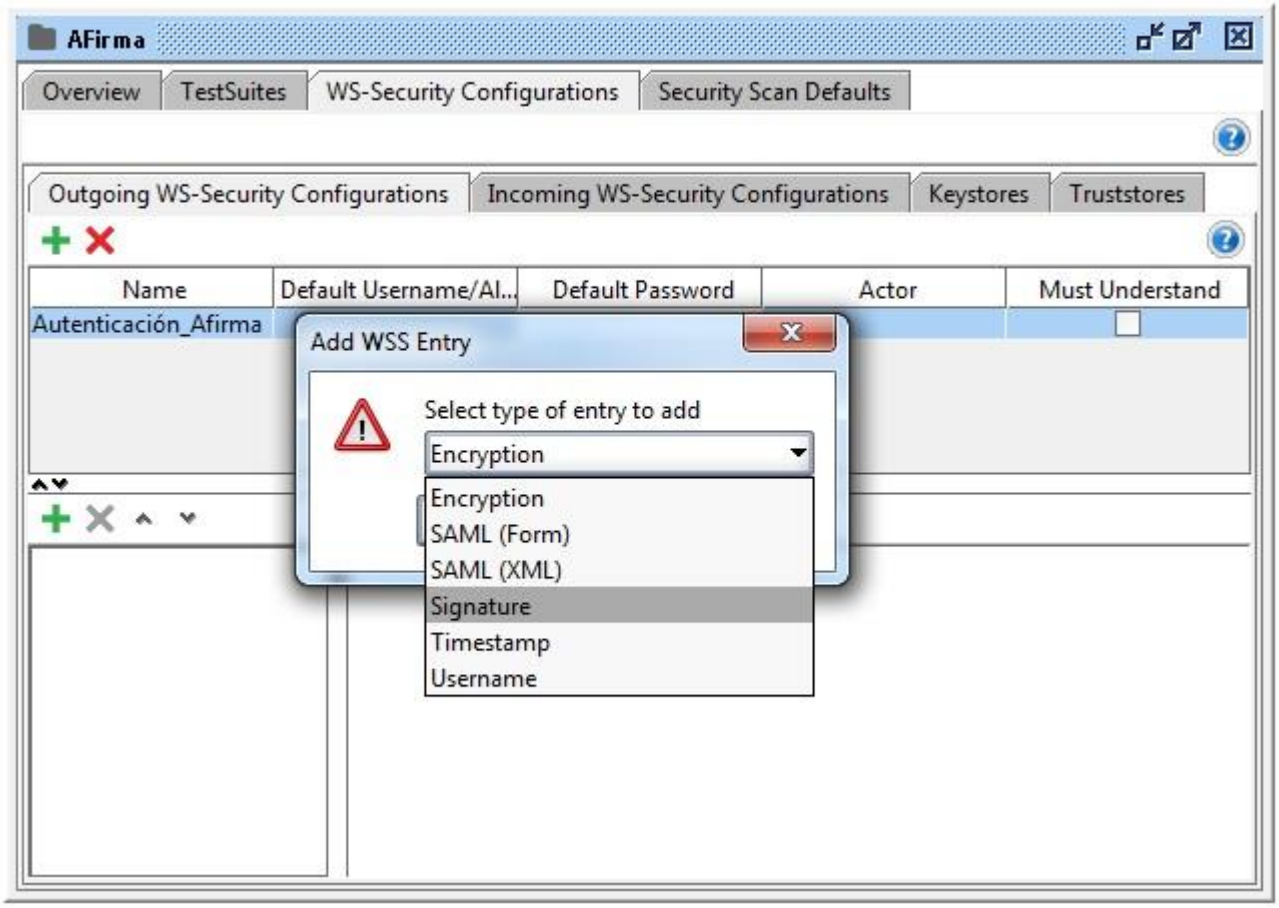

Ilustración 52: Menú desplegable de configuración WS-Security

Al pulsar aceptar se añadirá la entrada de firma a nuestra política, para la autenticación por certificado lo configuraremos de la siguiente manera:

- **Keystore:** Aquí seleccionaremos el certificado.
- **Alias:** Aquí seleccionaremos el alias del certificado, normalmente el único que aparezca.
- **Password:** Aquí introduciremos el password del p12.
- **KeyIdentifier:** Type: Seleccionaremos “Binary Security Token”
- **Use single certificate:** Marcaremos la opción “Use single certificate for signing”

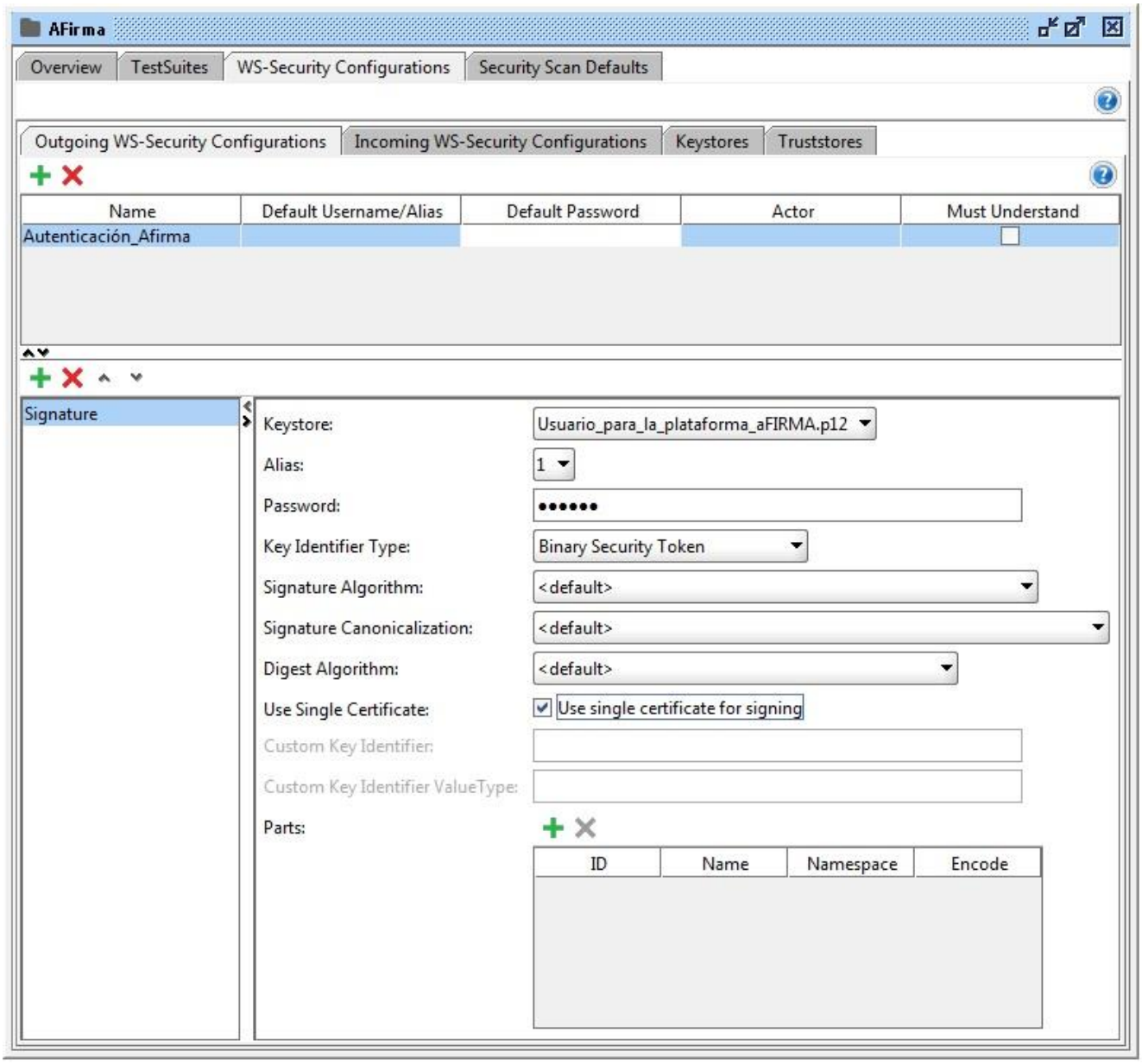

Ilustración 53: Configuración política de firmado

A continuación cerraremos la ventana y seleccionaremos la petición “01_RepresentanteFicticio OK” de la operación “ValidateCertificate” del proyecto Afirma.

A partir de aquí hay dos formas de firma la petición, una manual y otra automática. Para firmar la petición de forma manual haremos click derecho en el cuerpo de la petición, seleccionaremos “Outgoin WSS” y seleccionaremos “Apply Autenticacion_Afirma”:

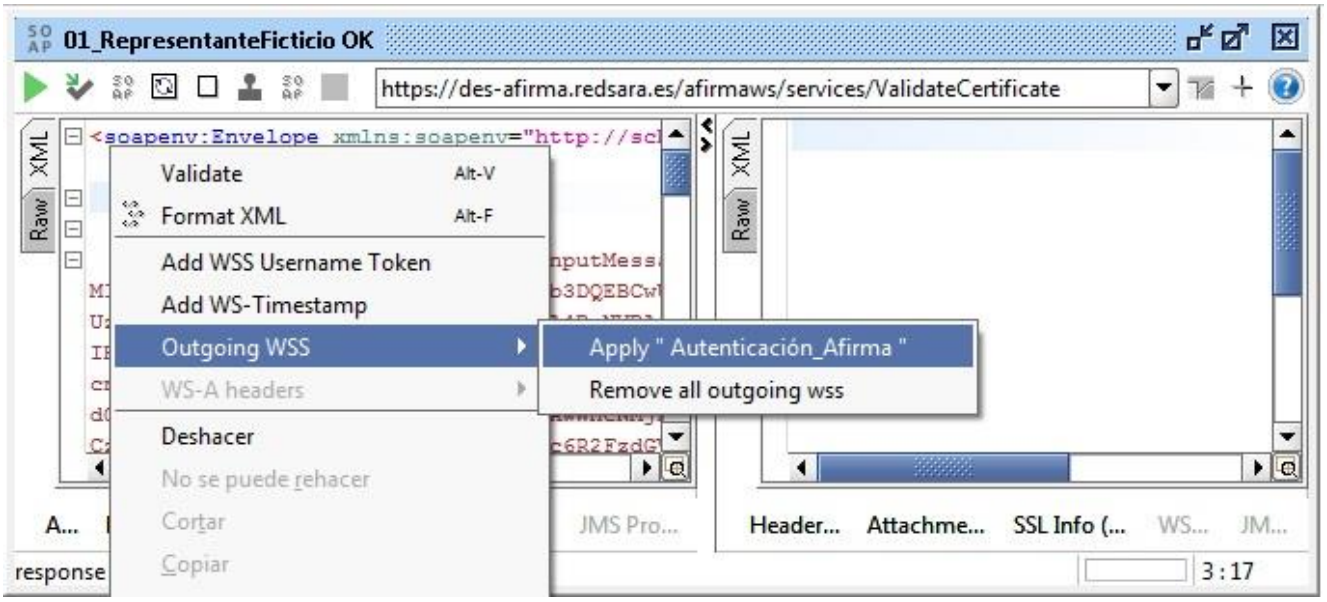

Ilustración 54: Firmado manual de peticiones

Se puede comprobar como SoapUI añade la etiqueta “security” la petición con la firma correspondiente. Este firmado es muy útil para comprobar que la política de firmado se ha configurado correctamente. Por ejemplo, en el caso de que no se haya cargado bien el certificado la etiqueta “security” aparecerá vacía. Cada vez que se modifique el mensaje es necesario volver a firma la petición ya que la firma se invalida.

Para evitar tener que firmar la petición cada vez que se modifica se puede configurar SoapUI para que lo haga automáticamente cada vez que se envía la petición. Para ello, eliminamos la firma si ya tenemos una. La firma se puede eliminar haciendo click derecho en la petición y seleccionar en “Outgoing WSS” la opción “Remove all outgoing wss”. Una vez hecho seleccionamos la pestaña “Auth” en la parte inferior izquierda de la ventana:

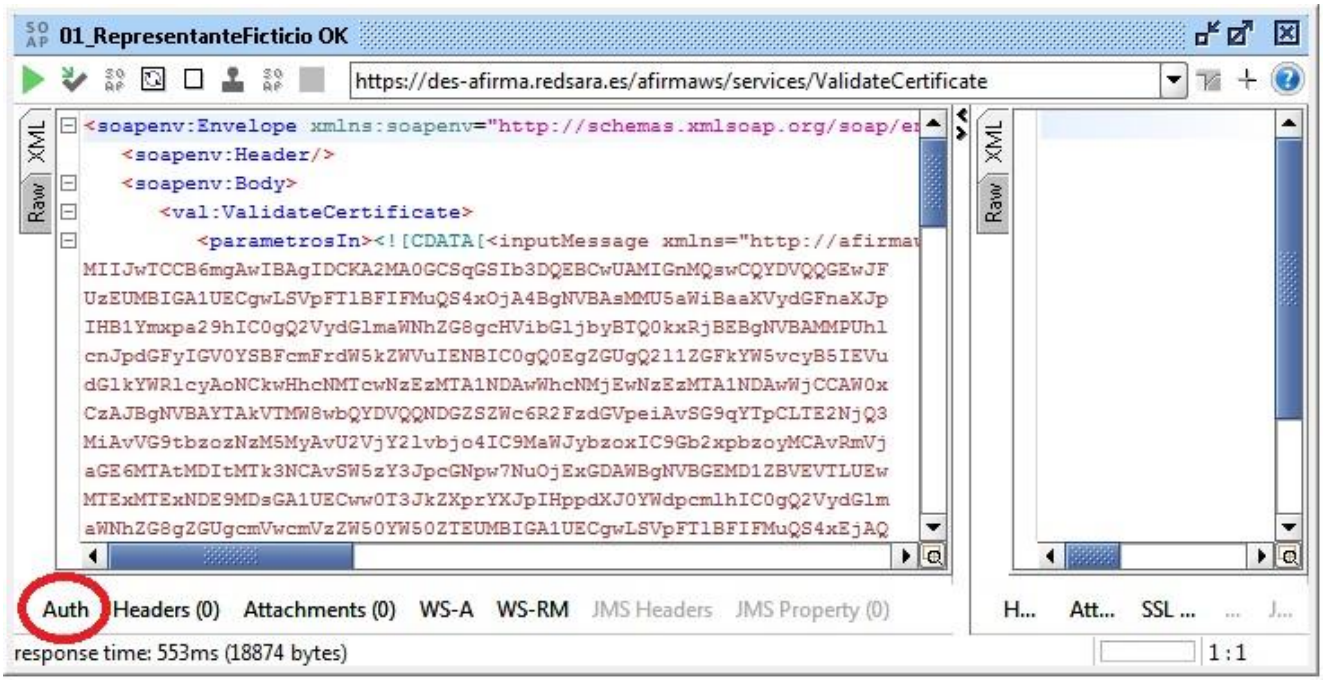

Ilustración 55: Pestaña "Auth" en la ventana de la petición

Esto abrirá una ventana justo debajo de la petición:

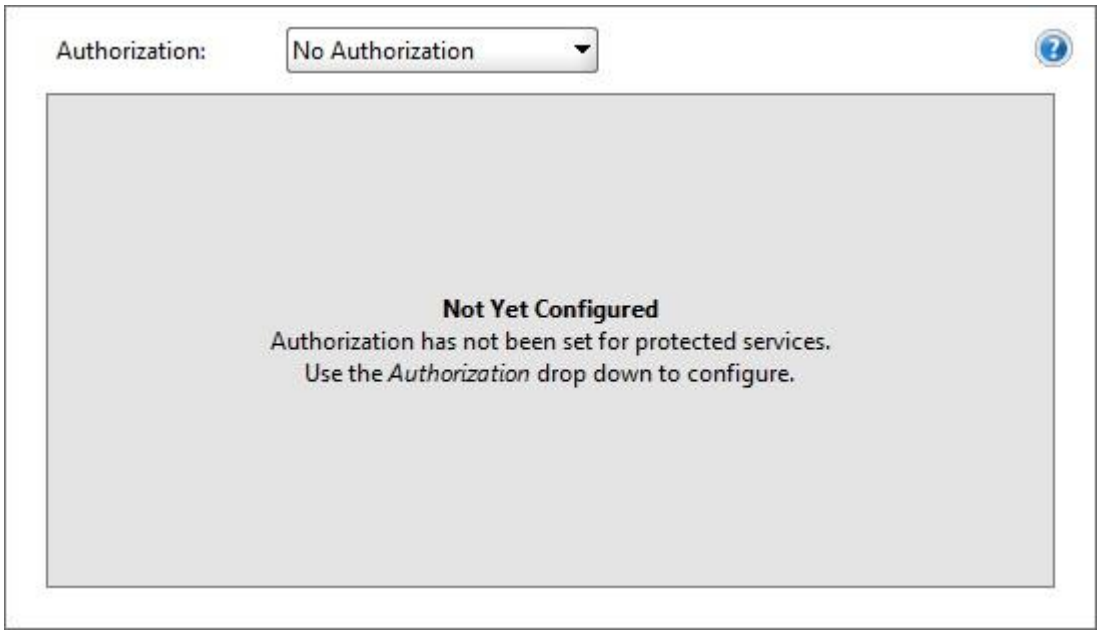

Ilustración 56: Ventana "auth"

Para configurar la autenticación, pulsamos el menú “Authorization” y seleccionamos “Basic”:

Ilustración 57: Seleccionando tipo de autenticación

Dejaremos todo como está salvo la opción “Outgoing WSS” En la que indicaremos el nombre de nuestra política de seguridad. “Autenticacion_Afirma”.

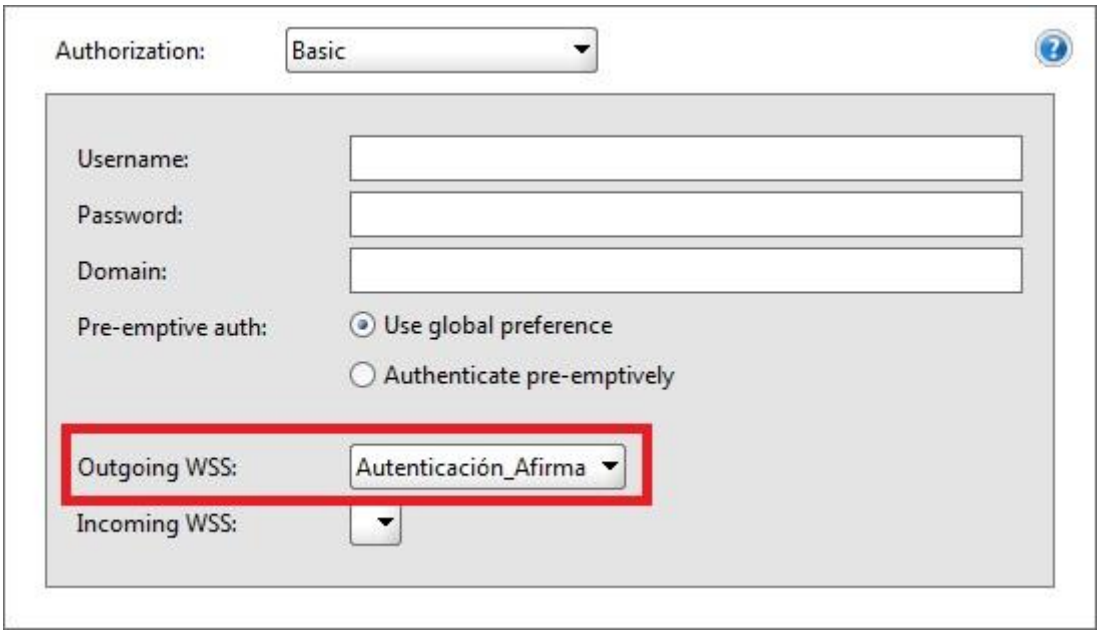

Ilustración 58: Seleccionando política de autenticación

Si pulsamos de nuevo la pestaña “Auth” la ventana de autenticación desaparecerá y podremos lanzar peticiones de la forma habitual. Aunque no aparezcan firmadas en el cuerpo de la petición SoapUI las firmará en cada envío de forma transparente.

4.1.2 Autenticación por usuario y password (pendiente)

Aunque en este caso de uso no tenemos ningún servicio al que se acceda por usuario y password, este tipo de seguridad se puede configurar de la misma forma que la autenticación por certificado. Para ello seguirnos el mismo flujo que para crear la política con certificado:

1. Click derecho en el nombre del proyecto.
2. En el menú emergente seleccionar “Show Project View” (Ilustración 47).
3. Seleccionar la pestaña “WS-Security Configurations” (Ilustración 48).
4. Pulsar el símbolo más (+) e introducir un nombre para la política
5. Pulsar el símbolo más (+) de la parte inferior de la ventana (Ilustración 51).
6. Seleccionar “username” en el menú emergente.

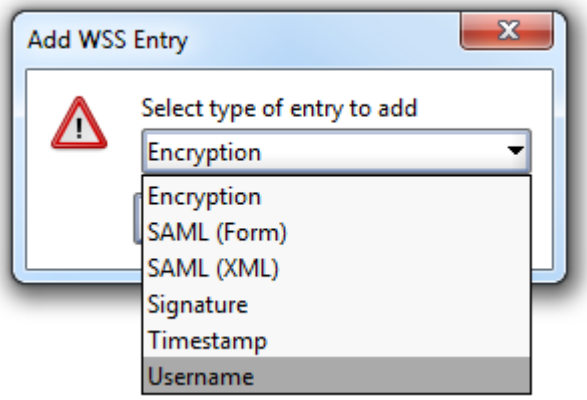

Ilustración 59: Política de usuario y contraseña

7. En los campos “Username” y “Password” introduciremos el usuario y la contraseña para acceder al servicio y en el menú “Password Type” indicaremos “PasswordText”. Los campos “Add nonce” y “Add created” (campo Timestamp) se pueden deshabilitar. Si se mantienen habilitados por requisitos del servicio será necesario aplicar la política como en el caso de la autenticación por certificado ya que se deberán recalcular para cada petición.

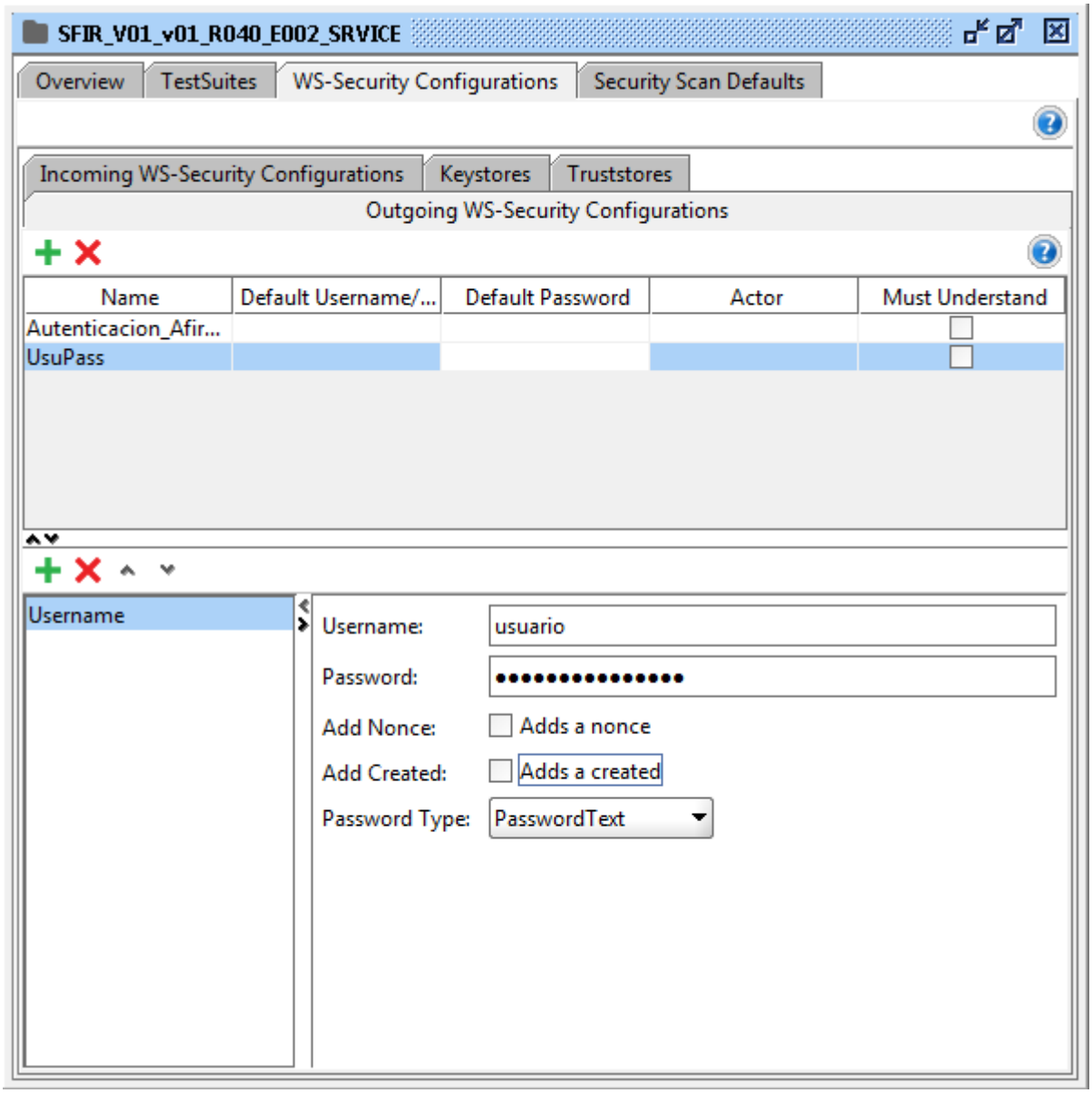

Ilustración 60: Configuración de la política de usuario y contraseña

8. Para aplicar la política deben seguirse los mismos pasos que se describen en las Ilustración 54 a Ilustración 58.