RDz Web Services Tutorial 02
Testing Web Services

Content

1. Tutorial Overview

The purpose of this exercise is to look at ways of testing Web Services. Some potential ways of testing a Web Service are

- **write a CICS program that invokes the Web Service,**
- **generate a Java program using Rational Application Developer (RAD) or WebSphere Developer for zSeries (WDz),**
- **use the debugger that comes with WDz,**
- **use the supplied Web Services Explorer that comes with RAD and WDz,**
- **use the TCP/IP Monitor that comes with RAD and WDz,**

or a combination of these.

In this tutorial we will use the Web Services Explorer and the TCP/IP Monitor. Although it is outside the scope of this tutorial, with RAD or WDz it is easy to generate a Java program to test the Web Service.

One of the major items in WSDL is the location (end-point) of the Web Service. Since WSDL is often generated by tools, the end-point location is not always correct, so you will want to verify that the end-point information is accurate. In this tutorial we will check the end-point information and will update this information to indicate the location of the Web Service that we will test.

In a later tutorial we will explore writing CICS Web Service requester programs and the use of the EXEC CICS INVOKE WEB SERVICE command.

For this tutorial, we will be invoking the supplied CICS Web Services example. We will just invoke the Web Service and see that a valid response is returned.

41.1 Scenario

In our scenario, you are a programmer at a mutual fund company. You have been given a WSDL file from a new Web Service that has been implemented on CICS TS V3.1 and you have been asked to test the service.

You will analyze the WSDL, verify the end-point information, and test the Web Service.

51.2 Tutorial Requirements

Please note that there are often several ways to perform functions in WDz. This tutorial will present one of the ways. If you are familiar with WDz, you will notice that some of the statements are general, and not necessarily true for every situation. The main intent of this tutorial is to reinforce the discussions on SOAP and WSDL.

The following are other assumptions made in this tutorial.

Assumption 1: A VMware image was supplied that contains WDz V7.0 and the files that are needed for this tutorial.
Assumption 2: The VMware image starts a Windows XP OS using "unilp" as login and password.
Assumption 3: You have some familiarity with RAD or WDz.
Assumption 4: A preconfigured CICS with the Web Services example is available and you have connectivity to the z/OS system that is running the CICS.

1.3 Section Overview

1. Tutorial Overview (this section)

2. Prerequisites

Before Starting the lab some prerequisite steps have to be performed to ensure your environment is configured properly.

3. Locate the Web Service Description (WSDL) for the Web Service
There are several ways to get the WSDL for a Web Service. In this section of the lab we will discuss the various options for obtaining WSDL and when you might use each option.

4. Analyze the WSDL for the Web Service
WSDL wasn't designed to be 'human-readable', but some understanding of WSDL will aid your understanding of the Web Service, and should anything go wrong, will greatly help in debugging.

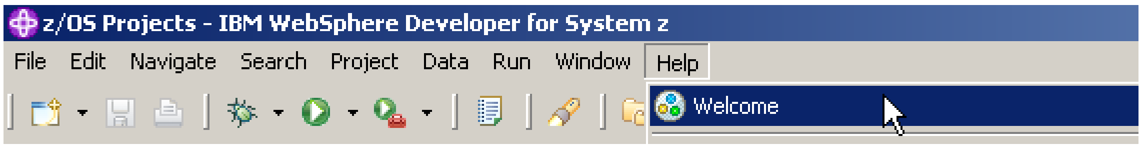
5. Change the WSDL End-point
The endpoint of a Web Service is the actual location of the Web Service. Although the WSDL that you have been supplied properly defines the Web Service interface, the endpoint specified in the WSDL is wrong. In this part of the tutorial, you will change the endpoint of the Web Service in the WSDL that is supplied.

6. Test the Web Service
In this part of the tutorial you use the Web Services Explorer, a part of Rational Application Developer and WebSphere Developer for zSeries, to invoke the Web Services and to see if it returns results as expected.

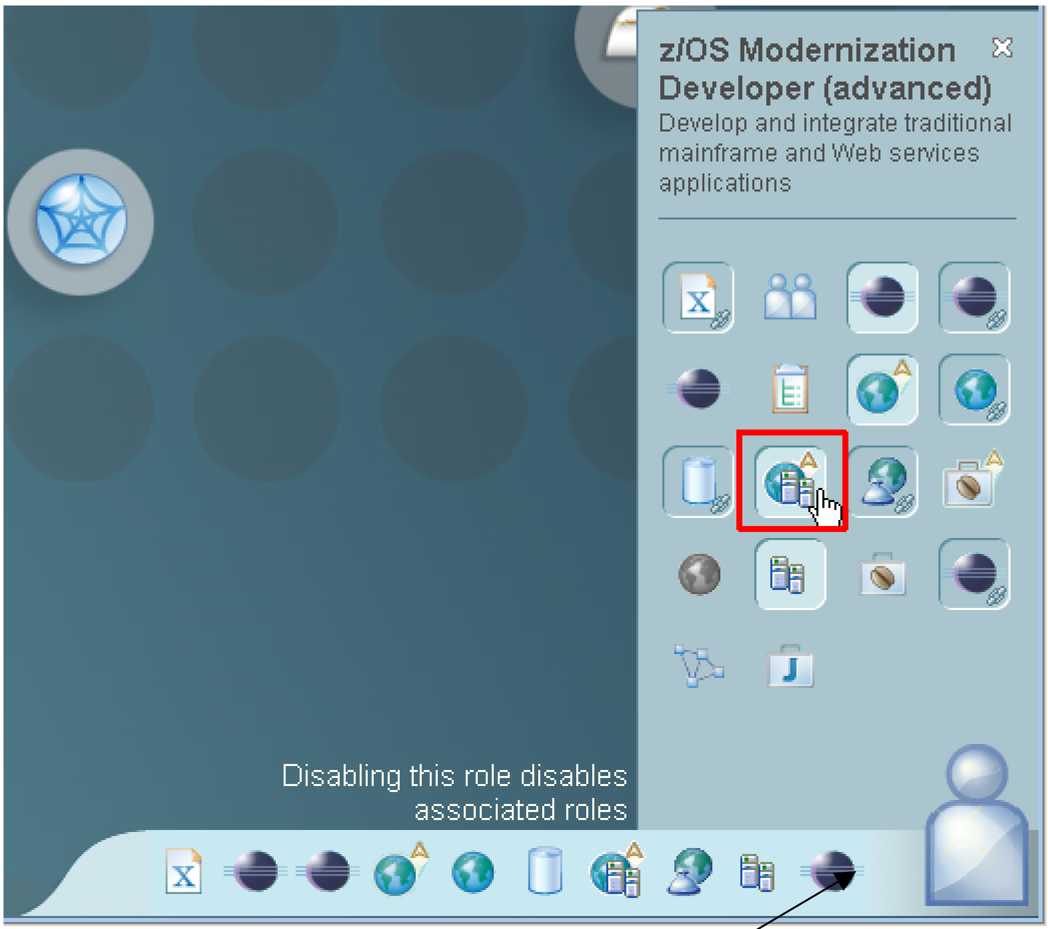
7. Look at Web Service flows using the TCP/IP Monitor Server
When developing or testing Web Services, it is sometimes beneficial to take a look at the Web Service flows (request/response) as they appear 'on-the-wire'. In this part of the tutorial you will run a program that intercepts the flows between the service requester and the service provider.

2. Prerequisites

The WebSphere Developer for zSeries Workbench is initialized based on roles. This means that depending on your enabled roles certain features are enabled and others are disabled. You can specify your role on the Welcome screen which is displayed at the first initialization of your workspace.



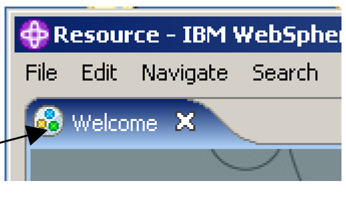
1.If the Welcome screen is not displayed in your workspace retrieve it by clicking Help ? Welcome.


z/OS Modernization Developer (advanced)
Develop and integrate traditional mainframe and Web services applications
Disabling this role disables associated roles

2.Your currently enabled roles are displayed in the lower right corner (note, that your currently enabled roles can differ from the screenshot below).

Click on the Symbol vaguely looking like a human body in the right lower corner and select the z/OS Modernization Developer (advanced) Icon. Verify your screen looks like shown above.

This role will automatically enable further associated roles like the Web Service Developer.



3.Close the Welcome window.

3. Locate the Web Service Description

WSDL (Web Services Definition Language) is an XML vocabulary that is a description of a service interface. The WSDL document contains a list of the available operations that can be invoked, a description of the information that is sent to and/or received from the service (including ‘fault’ information when problems occur), the protocol and transport used to access the service (e.g. SOAP over HTTP), and the location of the service.

There are several ways to get WSDL: e-mail; FTP or some other file transfer; from a web site like www.xmethods.com; using WSIL (Web Services Inspection Language), or using a UDDI server (Universal Description Discovery and Integration).

E-mail: WSDL is usually stored in a single file. This WSDL could easily be an attachment you might receive from your business partner to access their service.

FTP or other file transfer: Again, since the WSDL is simply a file, normal file transfer programs are a common way of exchanging WSDL.

Web sites: There are web sites available on the internet that offer Web Services that you can use or test with. Some of these are free, but some may charge a fee. An example of a web site that offers Web Service access is http://www.xmethods.com.

WSIL: Web Services Inspection Language is a language that allows you to query a Web Service provider site about the Web Services they offer and their interfaces. Support of this facility is not required by a Web Services provider. Although this facility may be useful in testing, it is doubtful that a production Web Service location will want to support ad hoc queries about their Web Services. Because of this, it is unlikely that you will see this facility available on a production Web Service web site.

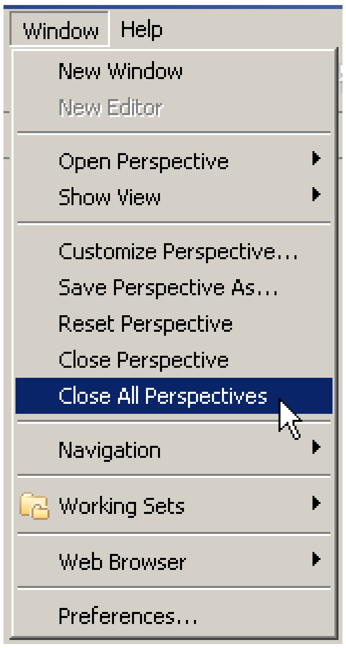
UDDI: A Universal Description, Discovery and Integration server is also known as a Business Registry. This server is similar in function to yellow pages and allows a business to register information about their available Web Services. A potential consumer of a Web Service can access a UDDI server using an architected API and request information about Web Service providers. UDDI servers allow a taxonomy or grouping, so potential consumers can ask for information on a specific Web Service provider, or can ask for a list of providers that offer a particular type of service. You could, for example, request a list of all banks that offer mortgage services. There are public UDDI servers, or you could run your own UDDI server. A UDDI server is supplied with WAS ND (WebSphere Application Server Network Deployment).

For this tutorial we have supplied the WSDL in the local PC file system.

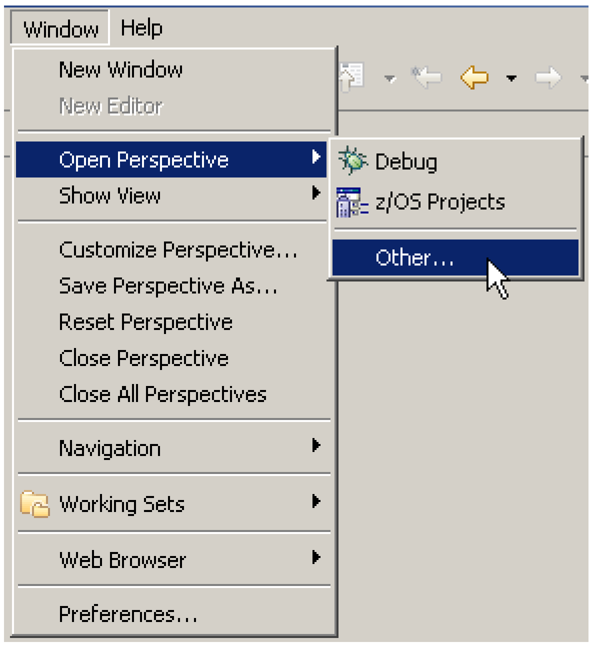
4. Create a Project

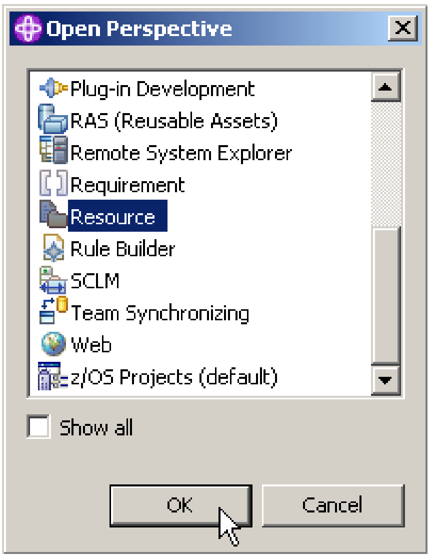
Computer artifacts are physical files that execute or are used by your software. Artifacts may include specifications, designs, code, use cases, class diagrams, etc. An artifact is one of many kinds of tangible byproduct produced during the development of software.

The various artifacts that we create in WDz are organized into Projects. We will create a simple project named WSDLTest. We will be working from the Resource perspective, but the functionality we will be using is available in any type of project in any type of perspective. The menus and wizards we will be using look at file extensions to determine which functionality should be offered for that file.

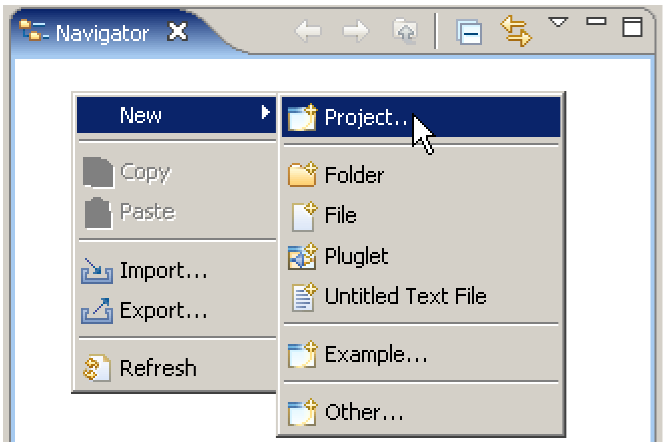


4.From the menu bar, select Window → Close All Perspectives.

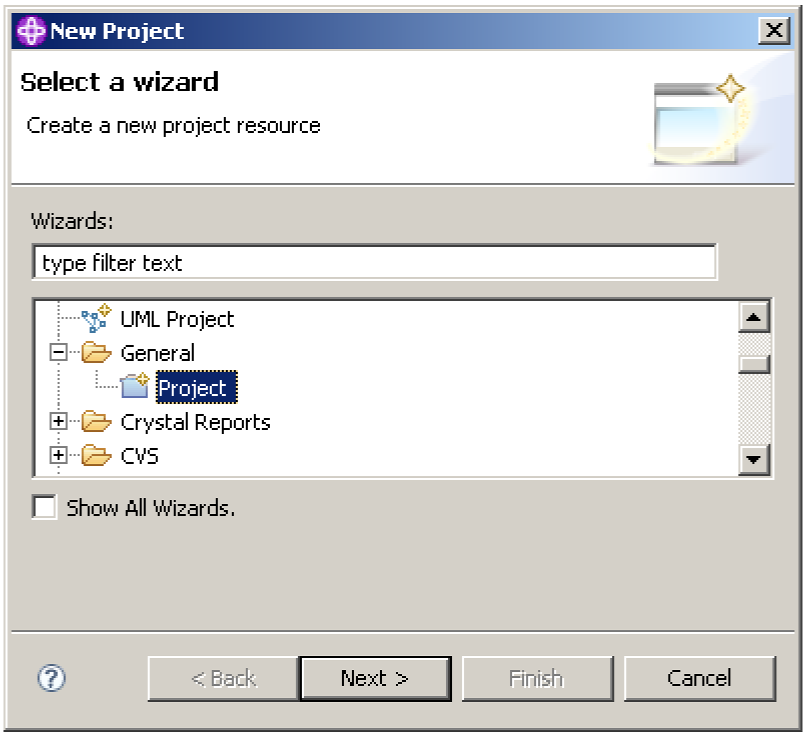




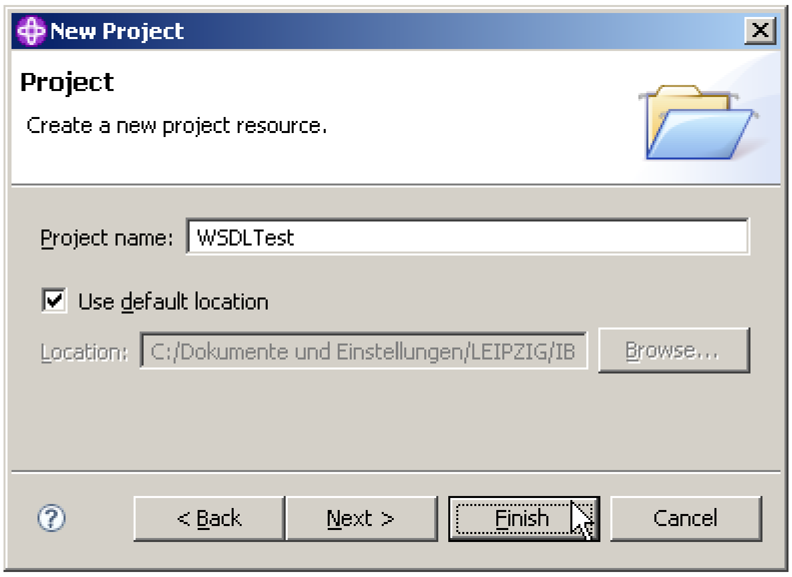
5.From the menu bar, select Window ? Open Perspective ? Other and select Resource to open the resource perspective. Click OK.



6.From the Navigator view, right-click in an open area, and from the context menu select New → Project….



7.Expand General and select Project.
Click the Next button.



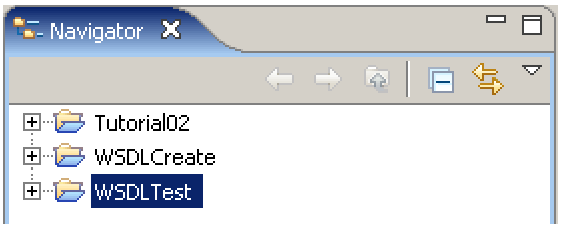
Project name: WSDLTest

☑ Use default location

Location: C:/Dokumente und Einstellungen/LEIPZIG/IB

Browse...

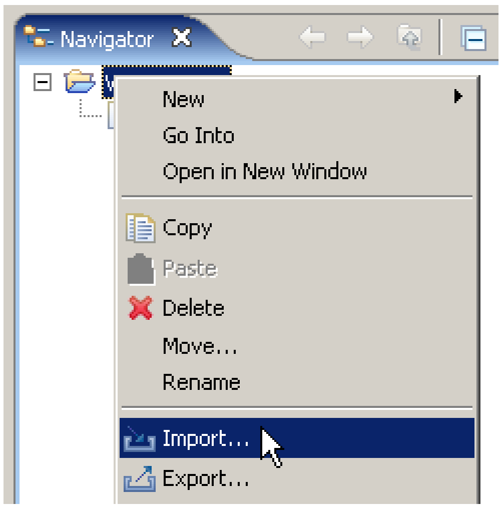
8.In the New Project dialog, specify the Project name as WSDLTest, ensure Use default location is checked, then click the Finish button.



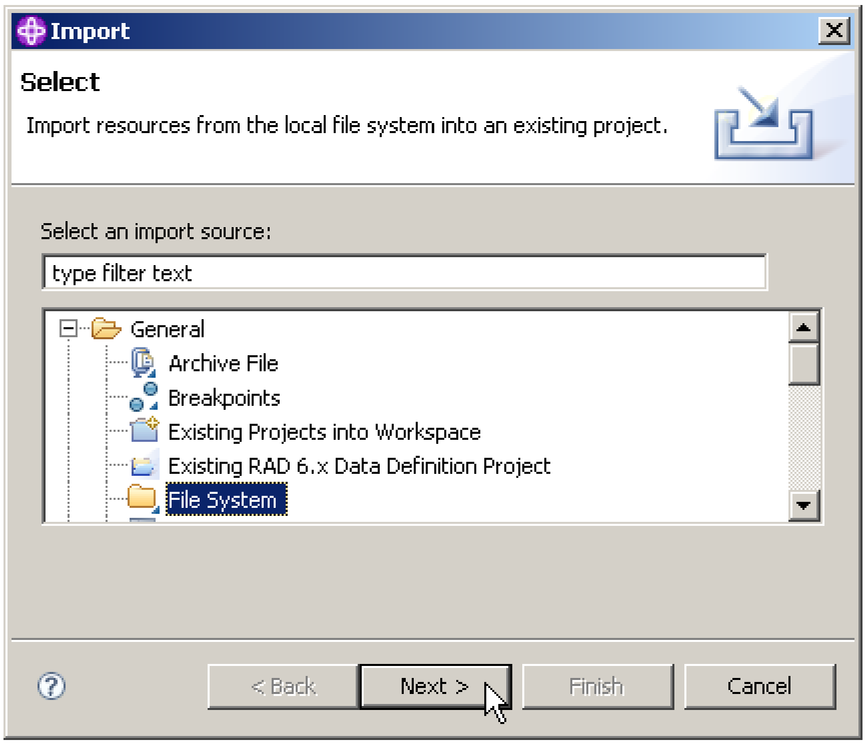
The Navigator view now contains a new entry WSDLTest.
And if you expand it you will see that it is empty besides the project description file .project.

5. Analyze the WSDL for the Web Service

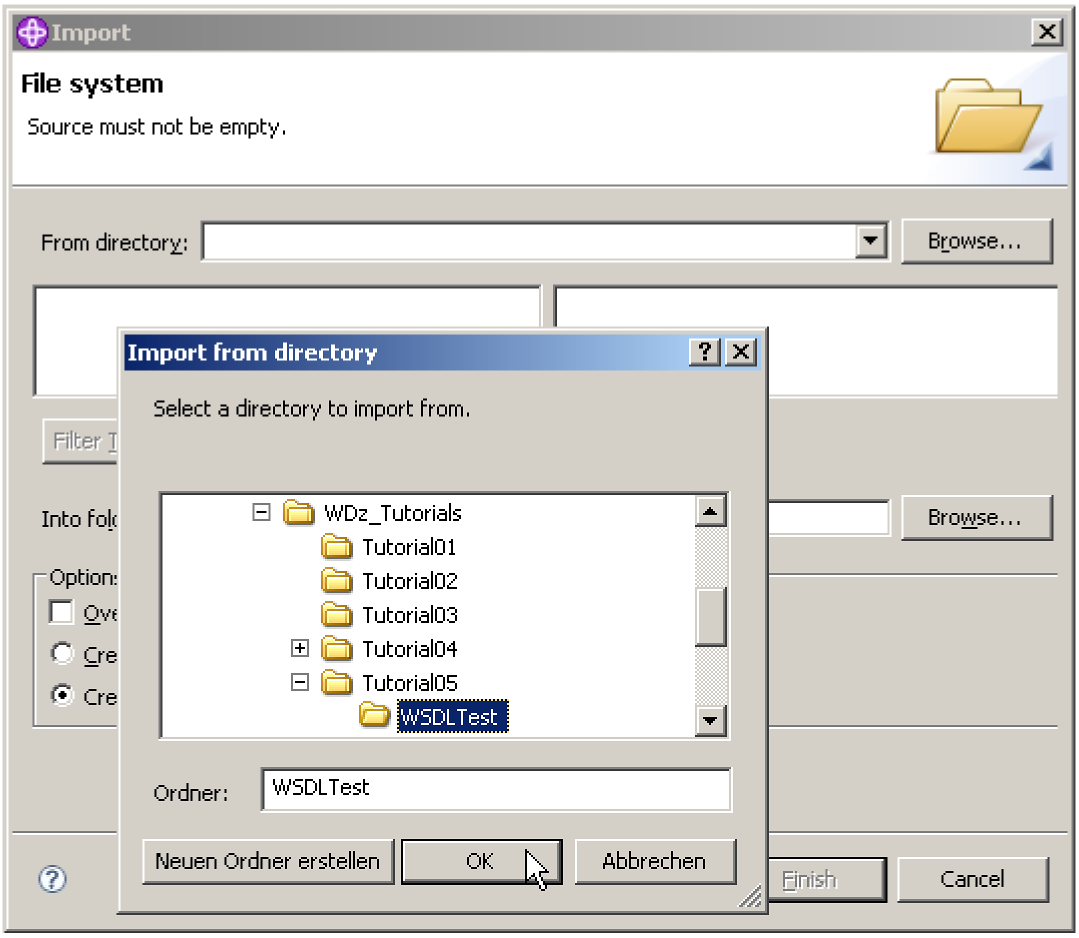
In this part of the tutorial, we will import the WSDL into our project and look at its contents.



9.From the Navigator view, right-click the WSDLTest project, and from the context menu, select Import.


Import
Select
Import resources from the local file system into an existing project.
Select an import source:
type filter text
General
Archive File
Breakpoints
Existing Projects into Workspace
Existing RAD 6.x Data Definition Project
File System
< Back
Next >
Finish
Cancel

10.In the Select dialog of the Import wizard, expand General and select File System. Click the Next button.



Import

File system

Source must not be empty.

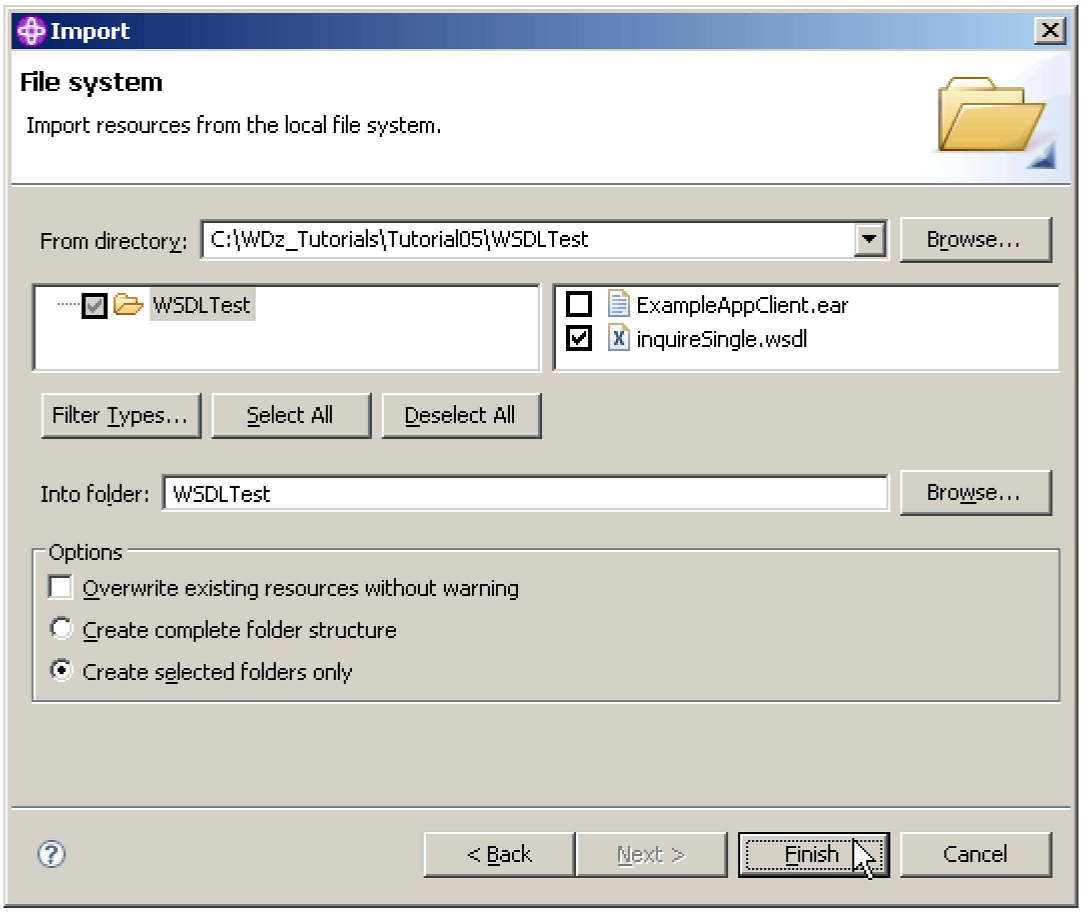
From directory:

Import from directory

Select a directory to import from.

Ordner: WSDLTest

11.Click the Browse button on the top right and navigate to C:\WDz_Tutorials\WDz Tutorial05\WSDLTest and click OK.



Import

File system

Import resources from the local file system.

From directory: C:\WDz_Tutorials\Tutorial05\WSDLTest

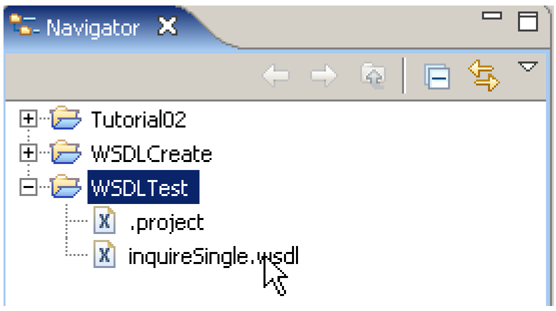
[✓] WSDLTest

[ ] ExampleAppClient.ear
[✓] inquireSingle.wsdl

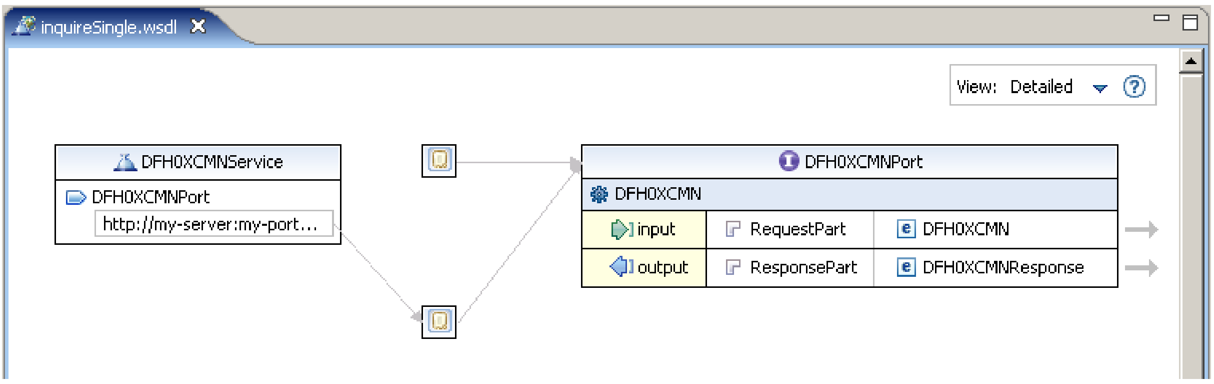
Into folder: WSDLTest

Options
[ ] Overwrite existing resources without warning
○ Create complete folder structure
● Create selected folders only

12.Back on the File system page of the Import wizard, check the box next to InquireSingle.wsdl, ensure that the Into folder is WSDLTest, ensure that Create selected folders only is checked and click the Finish button.

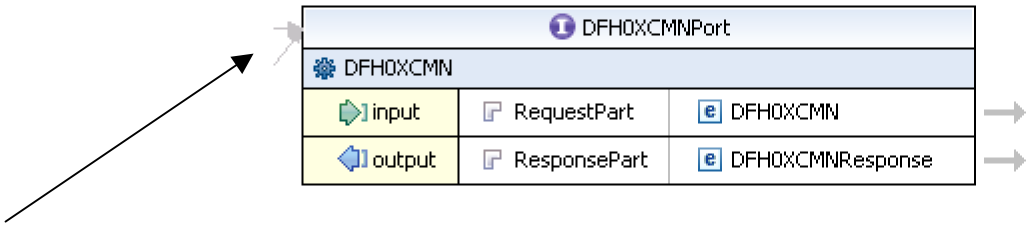


13.From the Navigator view, expand the WSDLTest project. Double-click on inquireSingle.wsdl. This will open the file in a WSDL editor, which is supplied with WDz.



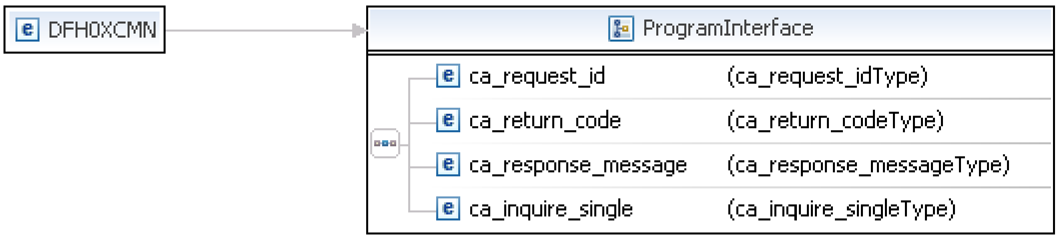
Note that inquireSingle.wsdl is part of the supplied CICS Web Service example which has already been installed on the System z mainframe. We will use the inquireSingle.wsdl file to test one of the functions in the CICS supplied Web Services example application.

We will now investigate the InquireSingle.wsdl. If you completed Tutorial04, you should recognize most of the WSDL-elements.

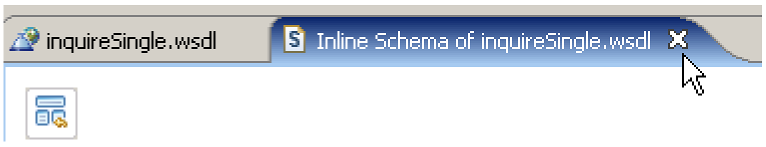


14.DFH0XCMNPort is the name of the Port Type. A Port Type is a collection of operations. In this case, we only have one operation called DFH0XCMN. The operation has an input, an output, and will not return a fault.

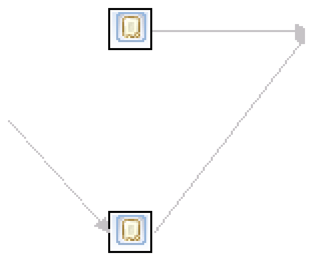
Note that the DFH0XCMNRequest message contains one part. The part is named RequestPart which refers to DFH0XCMN.


DFH0XCMN
ProgramInterface
ca_request_id (ca_request_idType)
ca_return_code (ca_return_codeType)
ca_response_message (ca_response_messageType)
ca_inquire_single (ca_inquire_singleType)

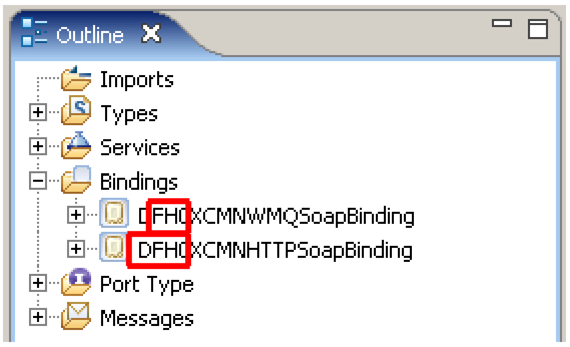
15.To see the Inline Schema of InquireSingle.wsdl, just click on the arrow on the very right of your input or output. You may have to click on DFH0XCMNPort to see the arrows.



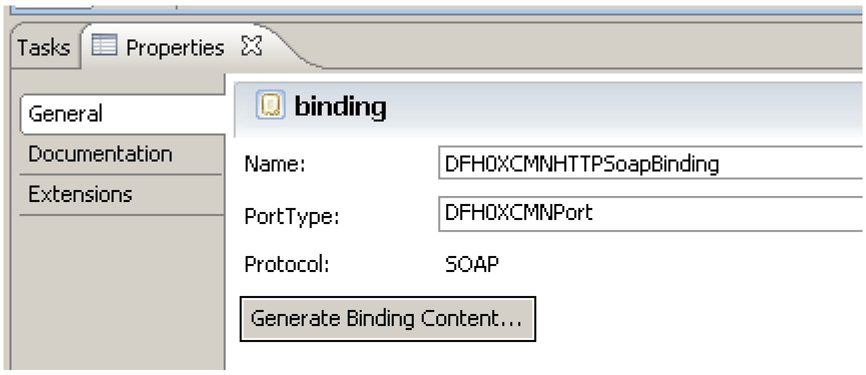
16.Close the Inline Schema and …



have a look at the bindings. Note that there are two bindings. Move the mouse over the small rectangle to see the name of the binding.



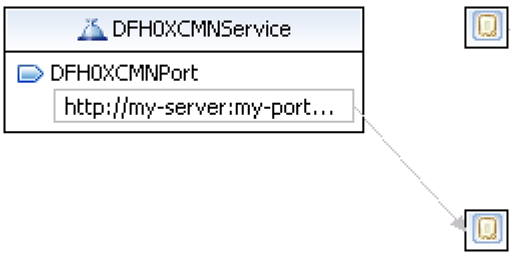
You can also see the names of the bindings in the Outline view.

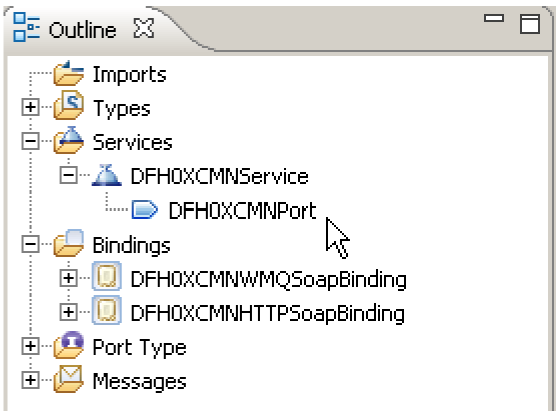


Name: DFH0XCMNHTTPSoapBinding
PortType: DFH0XCMNPort
Protocol: SOAP

You can also see the names of the bindings in the Properties view. If the Properties view is not available, go wo Window → Show View → Properties .

From the names of the bindings you can tell that one binding is for SOAP over HTTP and one is for SOAP over MQ. It shouldn't be too surprising that they are identical. Whether the SOAP is sent over HTTP or MQ, the SOAP will be exactly the same. WSDL allows you to have multiple bindings for a Port Type.



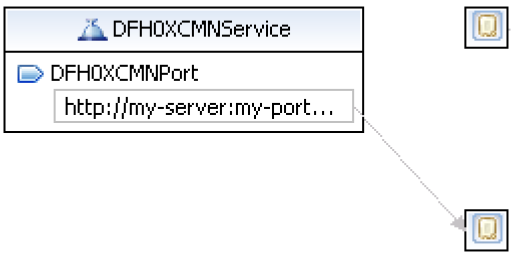


17.Finally, have a look at the Service DFH0XCMNService. A Service entry defines the location of the service.

Click on DFH0XCMNPort under DFH0XCMNService and notice (Again in the properties view) that this is the location of the Binding that defines SOAP over HTTP. Interestingly, there is no location specified for the Binding that defines SOAP over MQ (No link from Service to Binding)

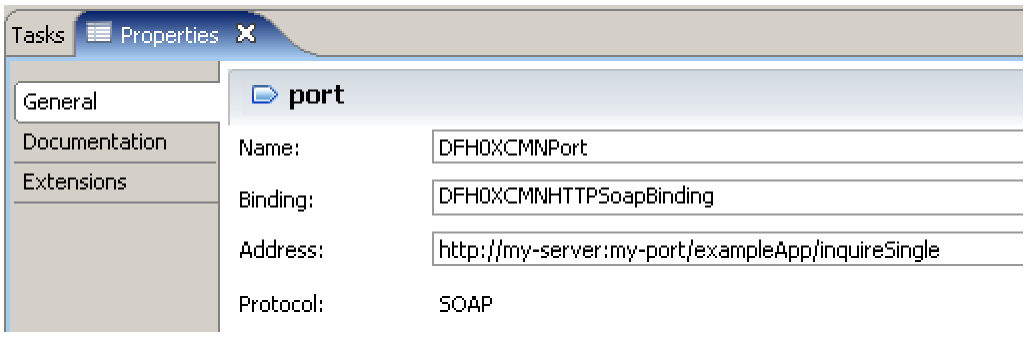
6. Change the WSDL End-point

In this part of the tutorial we will change the end-point of the Web Service. The end-point is the physical location of the service.

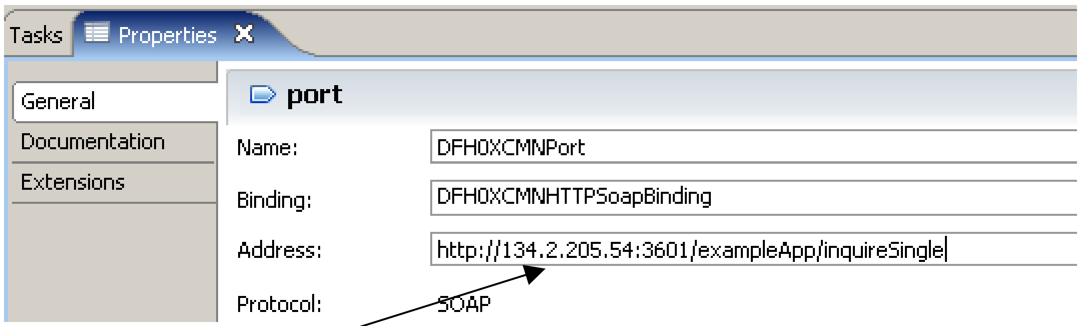


18.We still use the WSDL editor for the file inquireSingle.wsdl in the Navigator window. In the graphical view of the editor, in the Service element DFH0XCMNService, click on the address under DFH0XCMNPort to make the end-point editable.

(Note, that you can change it as well in the properties view)



Name: DFH0XCMNPort
Binding: DFH0XCMNHTTPSoapBinding
Address: http://my-server:my-port/exampleApp/inquireSingle
Protocol: SOAP



Name: DFH0XCMNPort
Binding: DFH0XCMNHTTPSoapBinding
Address: http://134.2.205.54:3601/exampleApp/inquireSingle
Protocol: SOAP

19.The WDz package that was installed on hobbit.informatik.uni-tuebingen.de (134.2.205.54) contains a sample application that we will use. To access it, we need to change the predefined, hypothetical address of the WSDL file. Be very careful to use the correct values, and do not overlook the use of port no. 3601 .

Because it is more comfortable, we specify the end-point using the properties view. Just change the Address from http://my-server:my-port/exampleApp/inquireSingle to http://134.2.205.54/exampleApp/inquireSingle .

Why all this ? If you use a standard browser, and enter

http://134.2.205.54:3601/exampleApp/inquireSingle

you will get this response:

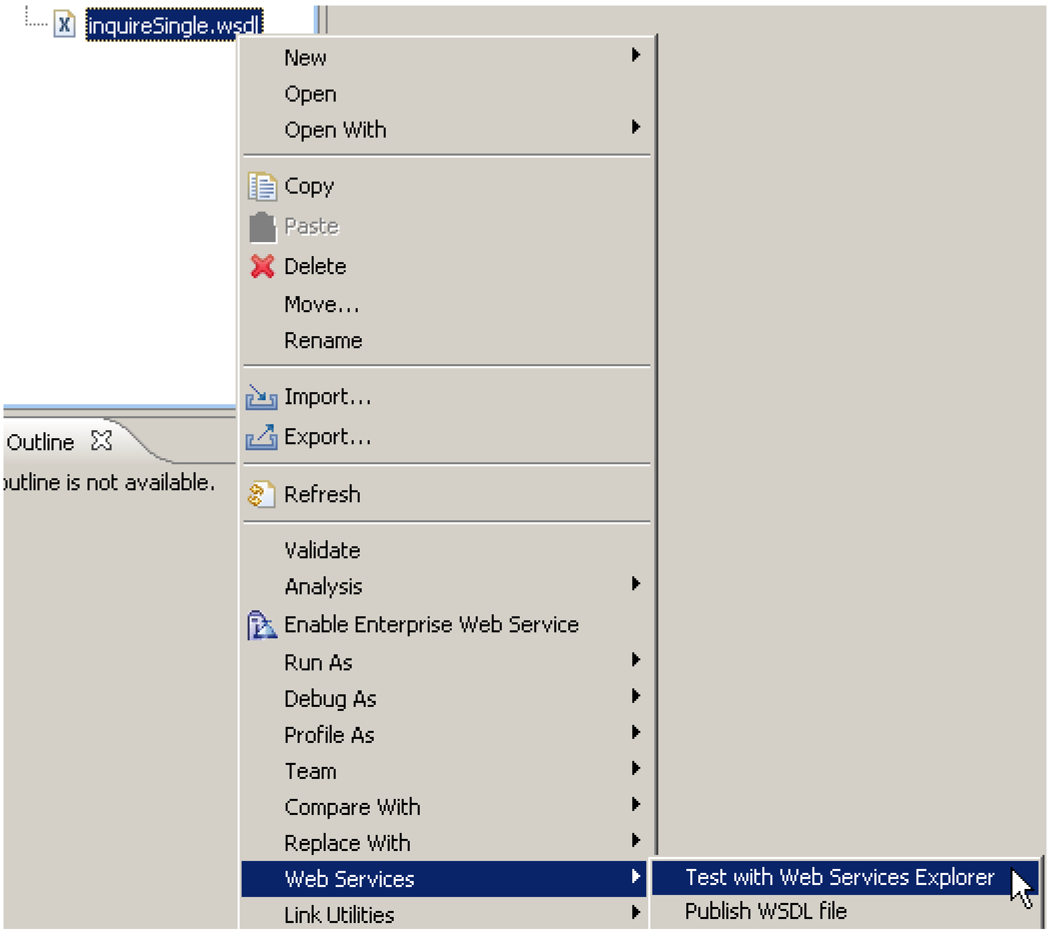
```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Server</faultcode>
      <faultstring>Internal Server Error</faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

20.Don't forget to save the WSDL file (CTRL + S)

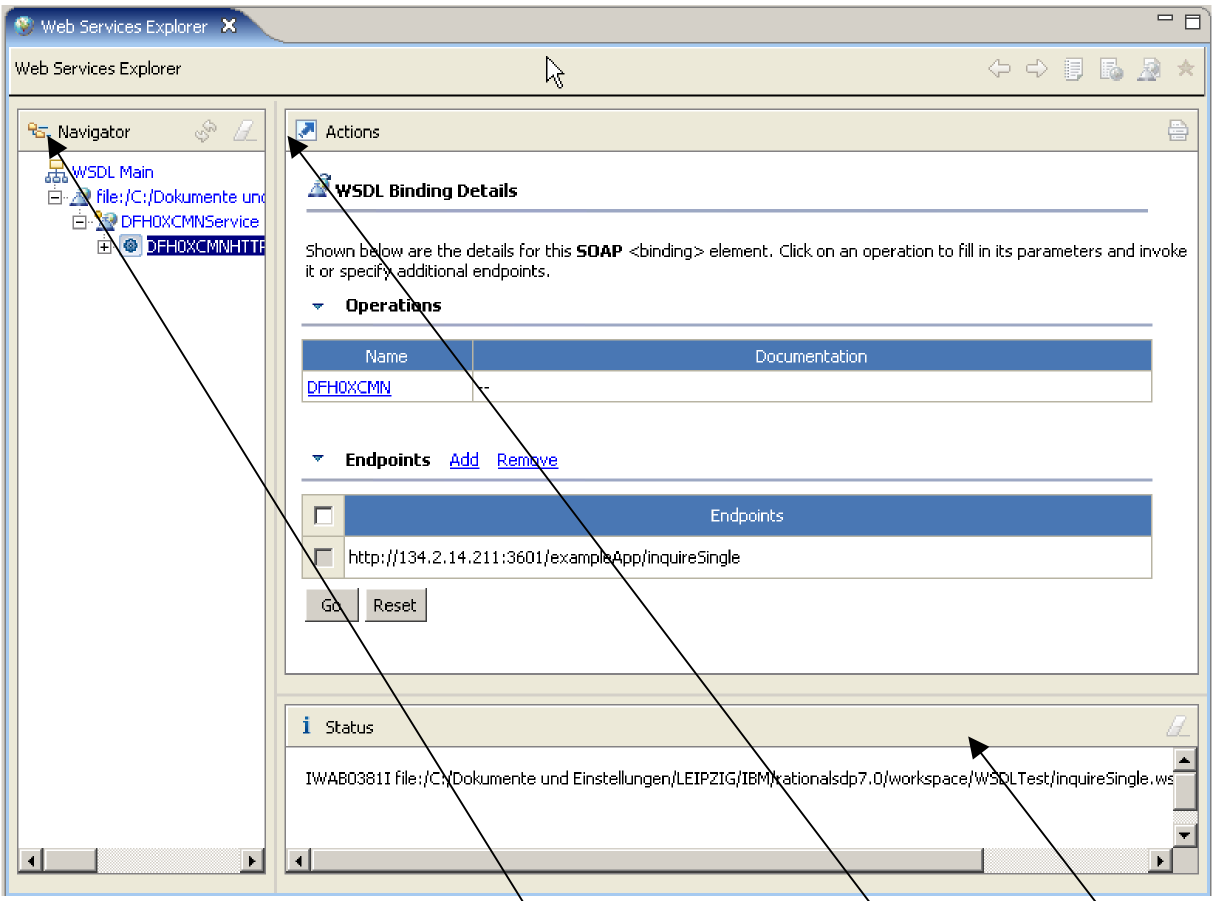
and close the WSDL editor.

7. Test the Web Service

The simplest test of a Web Service is just to invoke the Web Service and see if it returns the expected results. We will use the Web Services Explorer. Thus you don't need to write a Web Service requester (client) program.



21.From the Navigator view, right-click on inquireSingle.wsdl and from the context menu, select Web Services → Test with Web Services Explorer.



WSDL Binding Details

Shown below are the details for this **SOAP** <binding> element. Click on an operation to fill in its parameters and invoke it or specify additional endpoints.

Operations

Name	Documentation
DFHOXCMN	--

Endpoints Add Remove

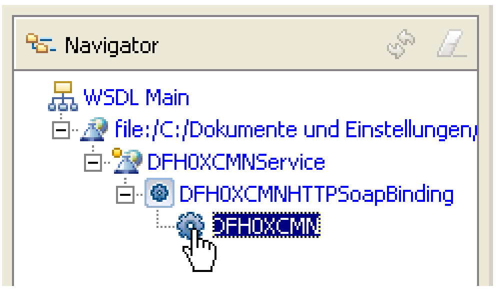
	Endpoints
☐	http://134.2.14.211:3601/exampleApp/inquireSingle

Go Reset

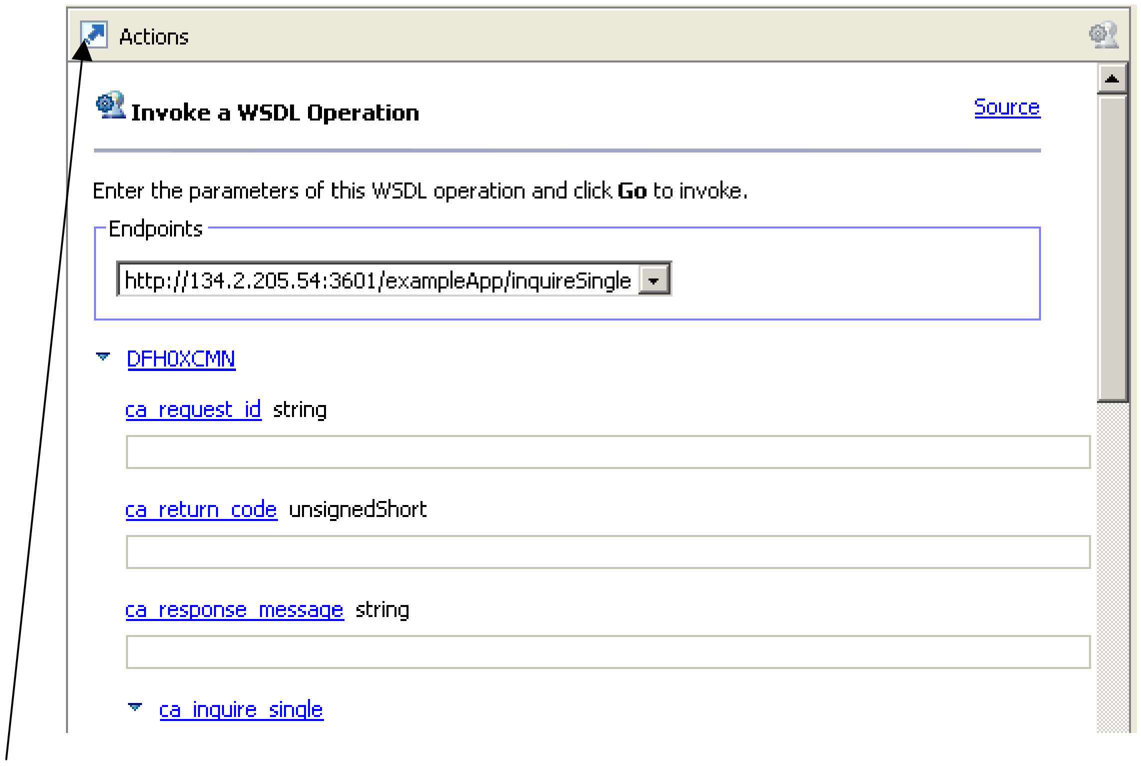
IWAB0381I file:/C:/Dokumente und Einstellungen/LEIPZIG/IBM/rationalsdp7.0/workspace/WSDLTest/inquireSingle.ws

22.You will notice that a Web Browser view opens.

The Web Browser view consists of 3 panes: Navigator pane (left), Actions pane (right) and Status pane (bottom).



23.In the Navigator pane of the Web Services Explorer, expand DFH0XCMHNTTPSoapBinding. Select the operation DFH0XCMN (the business operation) by double clicking on it..



Enter the parameters of this WSDL operation and click **Go** to invoke.

Endpoints: http://134.2.205.54:3601/exampleApp/inquireSingle

DFHOXCMN

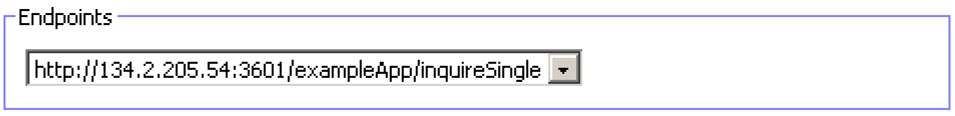
ca_request_id string

ca_return_code unsignedShort

ca_response_message string

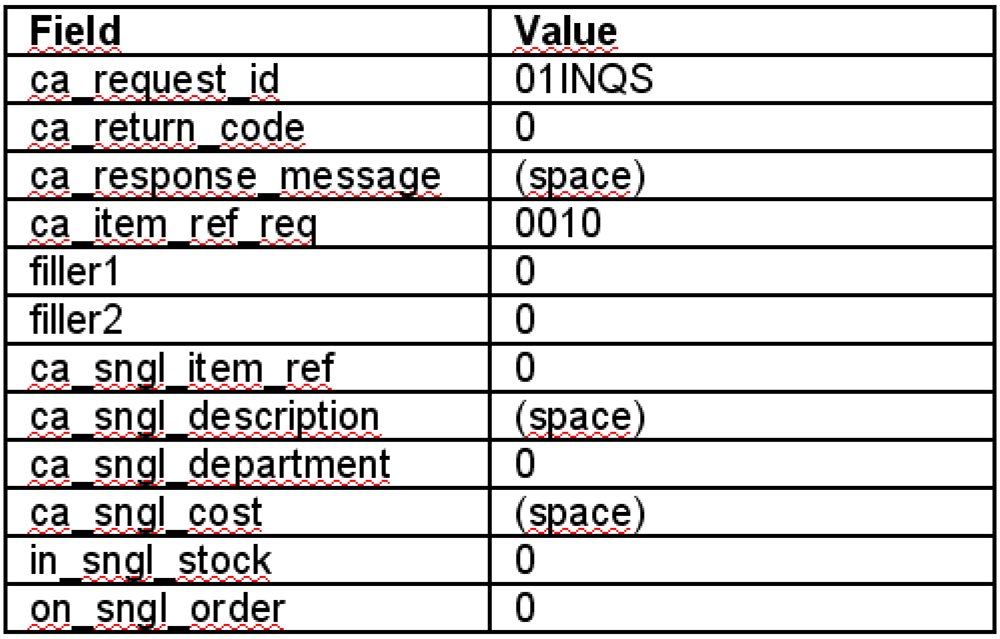
ca_inquire_single

24.Your Actions pane should look similar to the one above:


Endpoints: http://134.2.205.54:3601/exampleApp/inquireSingle

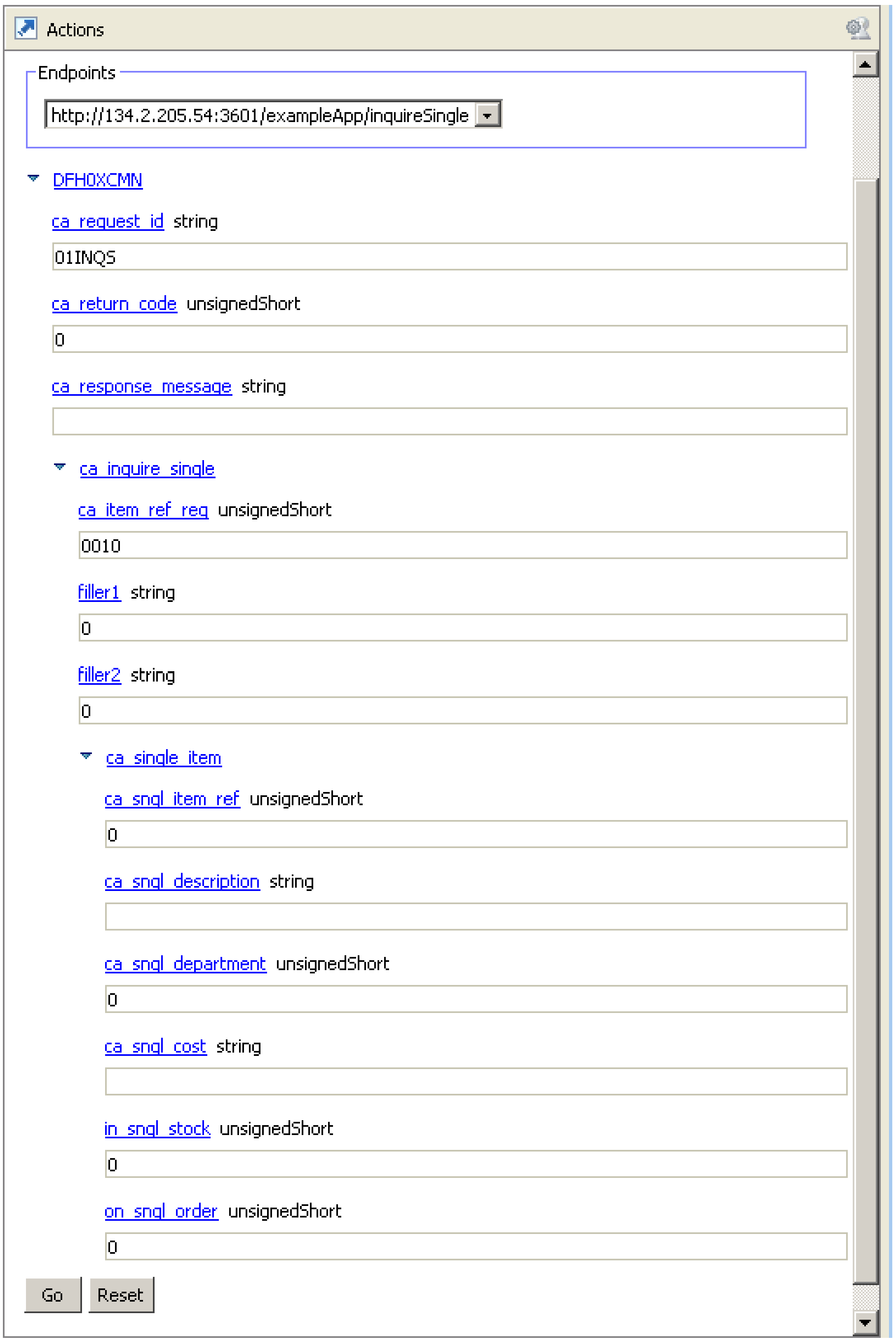
25.From the drop down menu in the Actions pane, enter the same endpoint we specified before (if not already chosen by default).

You see in the Actions pane a number of fields labeled ca_request_id, ca_return_code, etc. You can enter a value in each one of these fields.



Field	Value
ca_request_id	01INQS
ca_return_code	0
ca_response_message	(space)
ca_item_ref_req	0010
filler1	0
filler2	0
ca_sngl_item_ref	0
ca_sngl_description	(space)
ca_sngl_department	0
ca_sngl_cost	(space)
in_sngl_stock	0
on_sngl_order	0

26.Type in the following values (you may need to scroll down).
These values are for a request for information on item number 0010 in our inventory.


Actions

Endpoints

http://134.2.205.54:3601/exampleApp/inquireSingle

DFH0XCMN

ca_request_id string: 01INQS

ca_return_code unsignedShort: 0

ca_response_message string:

ca_inquire_single

ca_item_ref_req unsignedShort: 0010

filler1 string: 0

filler2 string: 0

ca_single_item

ca_sngl_item_ref unsignedShort: 0

ca_sngl_description string:

ca_sngl_department unsignedShort: 0

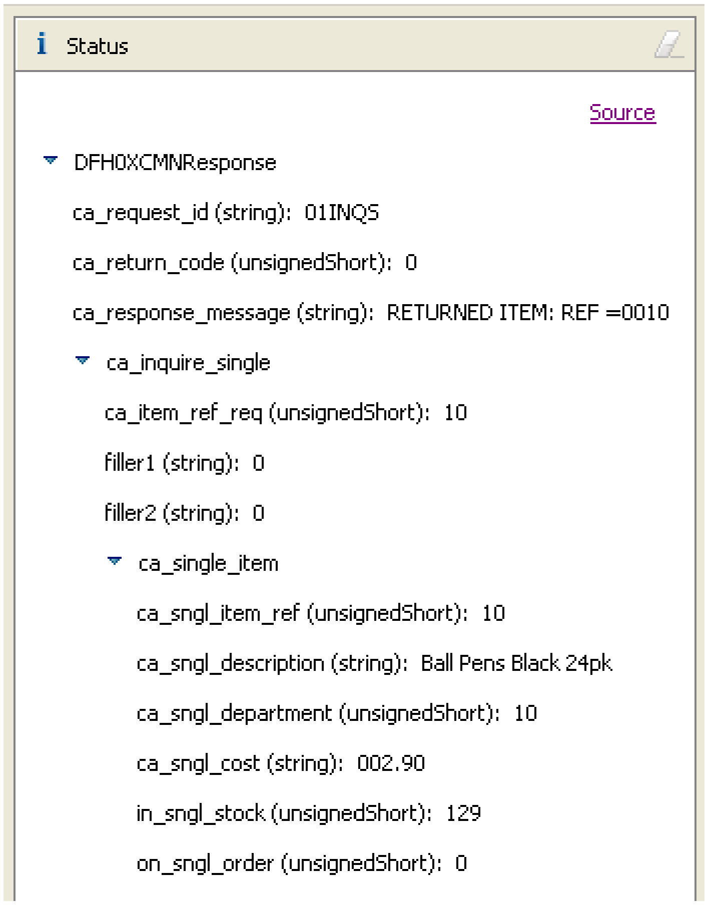
ca_sngl_cost string:

in_sngl_stock unsignedShort: 0

on_sngl_order unsignedShort: 0

The result should look like this.

Click the Go button at the bottom of the Actions pane

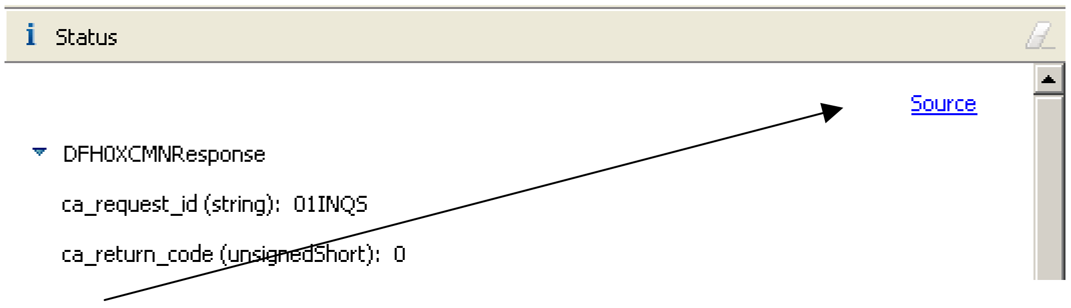


Status

Source

```
DFH0XCMNResponse
  ca_request_id (string):  01INQS
  ca_return_code (unsignedShort):  0
  ca_response_message (string):  RETURNED ITEM: REF =0010
  ca_inquire_single
    ca_item_ref_req (unsignedShort):  10
    filler1 (string):  0
    filler2 (string):  0
    ca_single_item
      ca_sngl_item_ref (unsignedShort):  10
      ca_sngl_description (string):  Ball Pens Black 24pk
      ca_sngl_department (unsignedShort):  10
      ca_sngl_cost (string):  002.90
      in_sngl_stock (unsignedShort):  129
      on_sngl_order (unsignedShort):  0
```

27.The results from the Web Service request will be shown in the Status pane. You can double-click on the title Status to expand this pane. You should see a window similar to the above.



Status

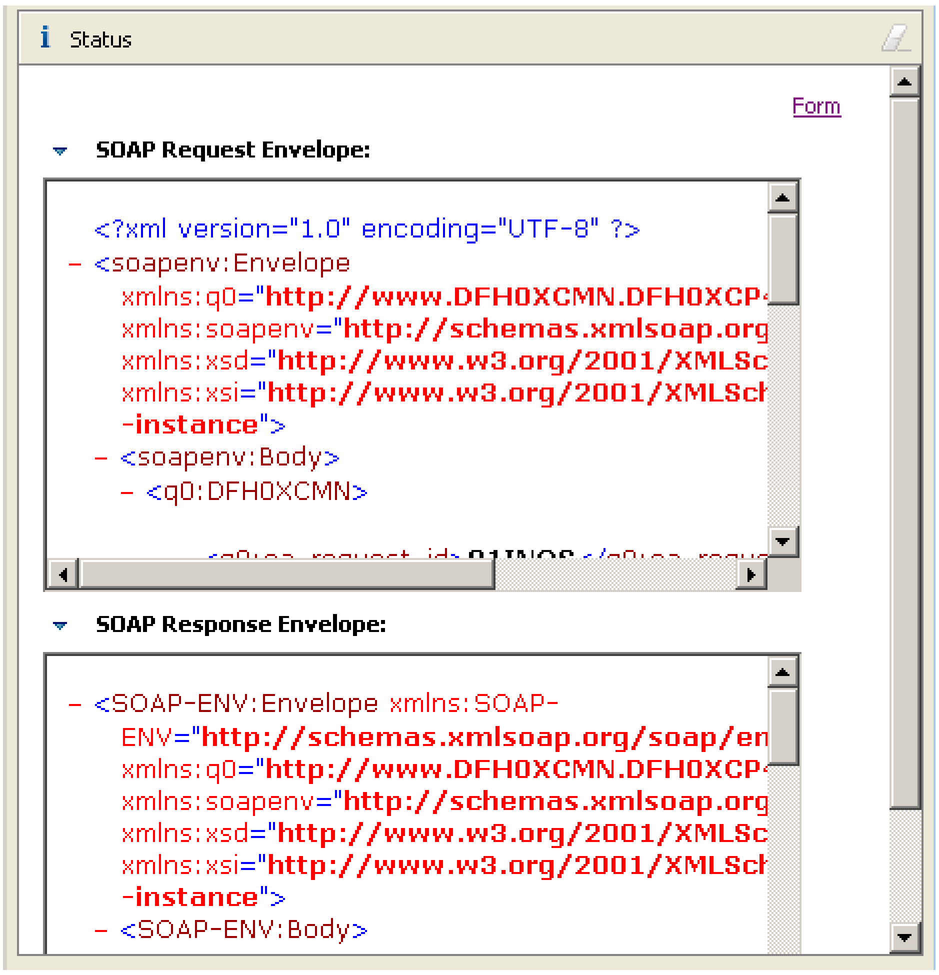
Source

DFH0XCMNResponse

ca_request_id (string): 01INQS

ca_return_code (unsignedShort): 0

28.Click on Source towards the top right in the Status pane



Status

Form

SOAP Request Envelope:

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <soapenv:Envelope
    xmlns:q0="http://www.DFH0XCMN.DFH0XCP
    xmlns:soapenv="http://schemas.xmlsoap.org
    xmlns:xsd="http://www.w3.org/2001/XMLSc
    xmlns:xsi="http://www.w3.org/2001/XMLSch
    -instance">
  - <soapenv:Body>
    - <q0:DFH0XCMN>
```

SOAP Response Envelope:

```
- <SOAP-ENV:Envelope xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/en
    xmlns:q0="http://www.DFH0XCMN.DFH0XCP
    xmlns:soapenv="http://schemas.xmlsoap.org
    xmlns:xsd="http://www.w3.org/2001/XMLSc
    xmlns:xsi="http://www.w3.org/2001/XMLSch
    -instance">
  - <SOAP-ENV:Body>
```

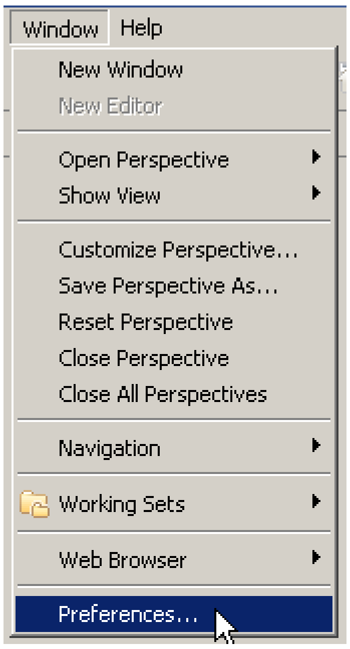
to get a detailed Source Code view of the SOAP request and response.

29.Double-click on the title Status to return to the previous Web Services Explorer view.

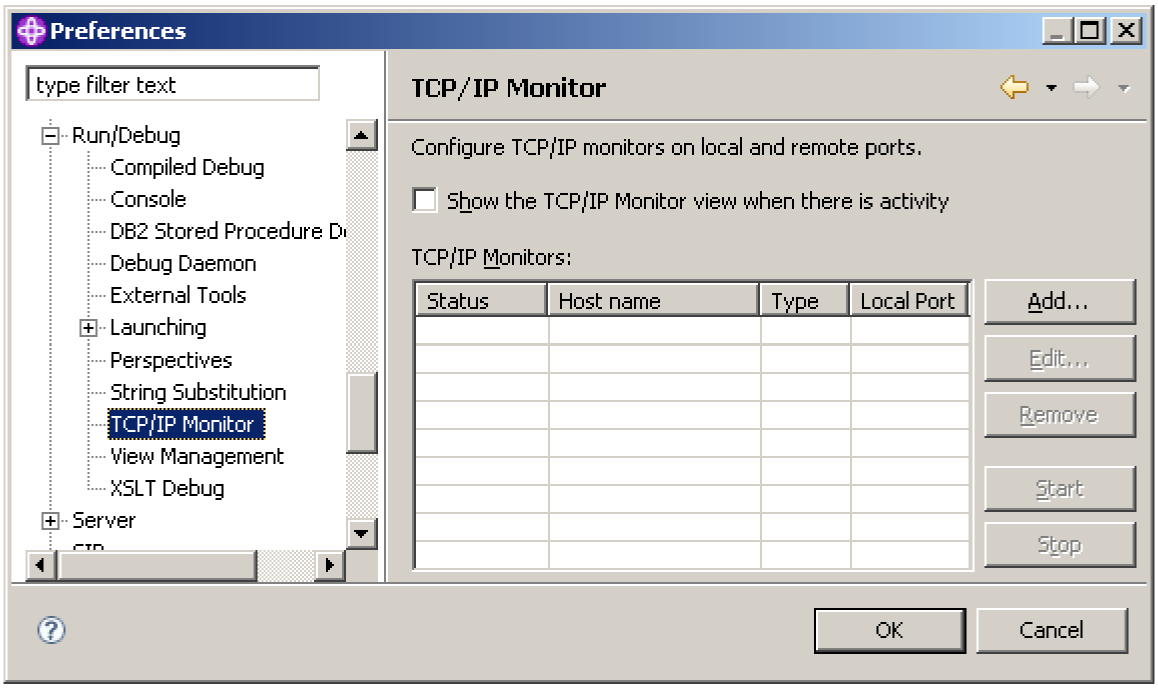
You successfully finished the Web Service test, but don't close the Web Services Explorer as you will need it again for the next chapter.

8. Look at Web Service Flows with the TCP/IP Monitor Server

An additional way to test Web Services is to look at the Web Services flows 'on the wire'. We can do this with the TCP/IP Monitor Server that is supplied with WDz. We configure the TCP/IP Monitor Server so that it receives the Web Service request from the Web Service requester, and forwards the request to the Web Service provider. The TCP/IP Monitor Server allows us to view requests/responses as they flow back and forth between the requester and the provider.



30.From the menu bar, select Window → Preferences.



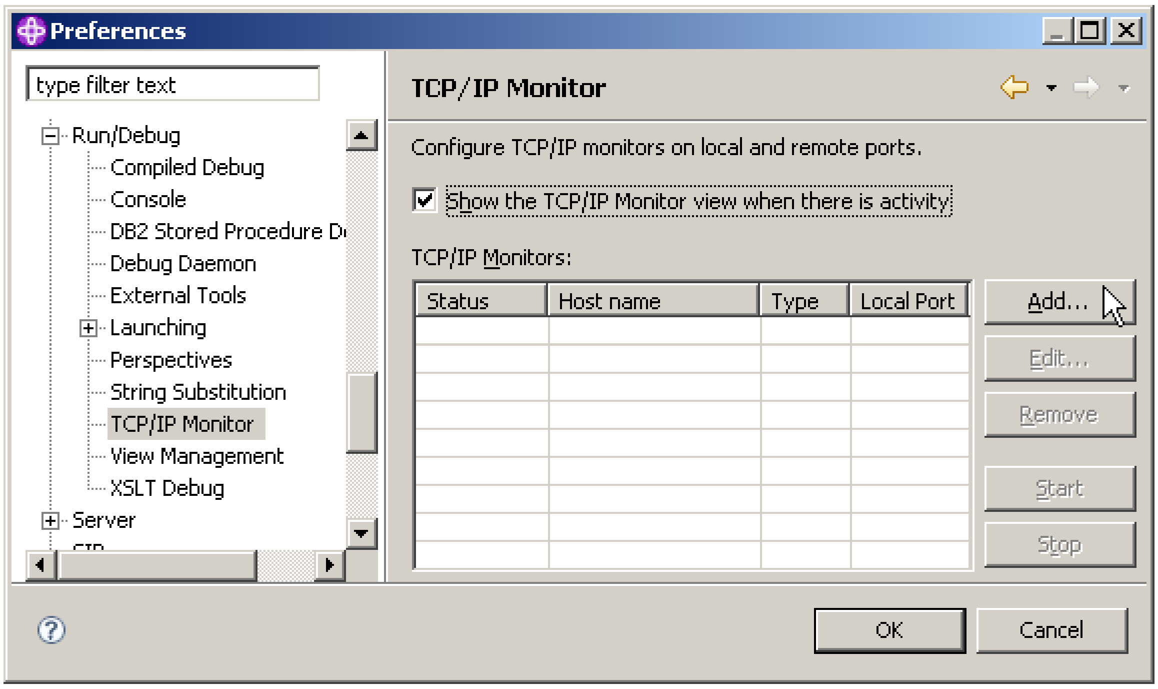
Configure TCP/IP monitors on local and remote ports.

☐ Show the TCP/IP Monitor view when there is activity

TCP/IP Monitors:

Status	Host name	Type	Local Port

31.In the Preferences dialog, expand Run/Debug and then select TCP/IP Monitor.


Preferences

TCP/IP Monitor

Configure TCP/IP monitors on local and remote ports.

☑ Show the TCP/IP Monitor view when there is activity

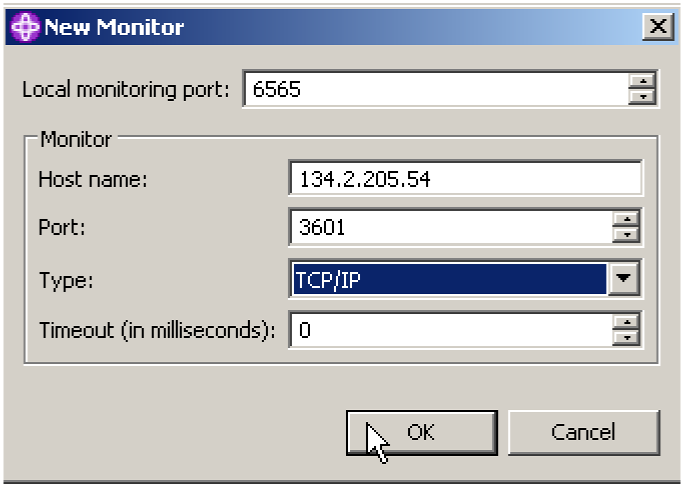
TCP/IP Monitors:

Status	Host name	Type	Local Port

**32.On the right, check Show the TCP/IP Monitor view when there is activity to display the TCP/IP Monitor view when there is activity through a TCP/IP monitoring server.
Click Add...**

Field	Value
Local Monitoring Port	Specify any unused port, i.e. 6565
Host name	139.18.4.35
Port	3601
Type	TCP/IP
Timeout	0

33.On the New Monitor panel, specify the above settings and


New Monitor

Local monitoring port: 6565

Monitor

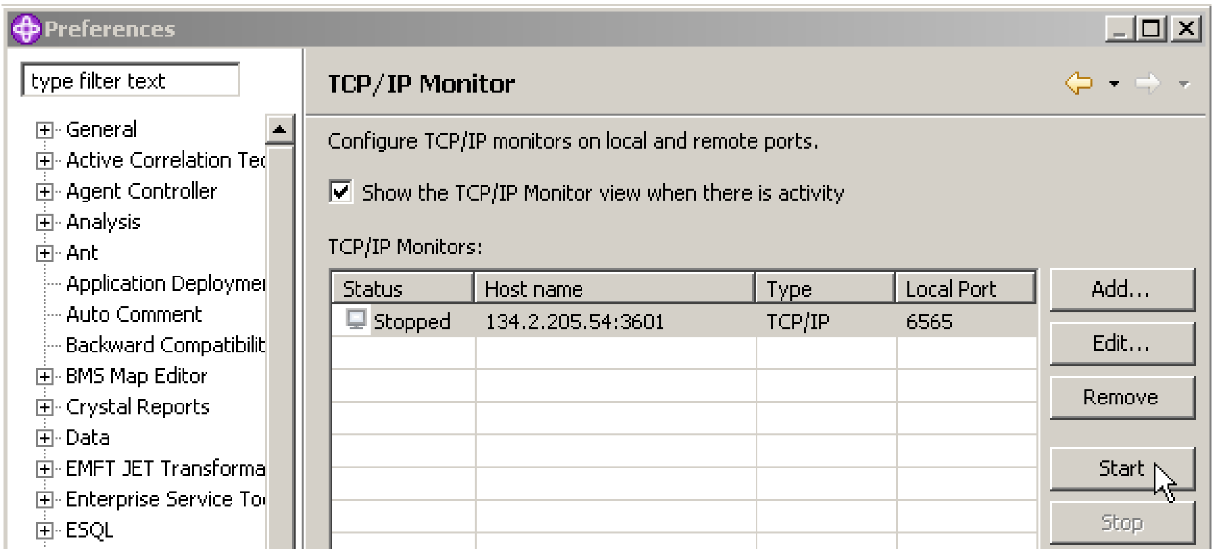
Host name: 134.2.205.54

Port: 3601

Type: TCP/IP

Timeout (in milliseconds): 0

click OK.


Preferences

type filter text

- General
- Active Correlation Tec
- Agent Controller
- Analysis
- Ant
- Application Deployme
- Auto Comment
- Backward Compatibilit
- BMS Map Editor
- Crystal Reports
- Data
- EMFT JET Transforma
- Enterprise Service To
- ESQL

TCP/IP Monitor

Configure TCP/IP monitors on local and remote ports.

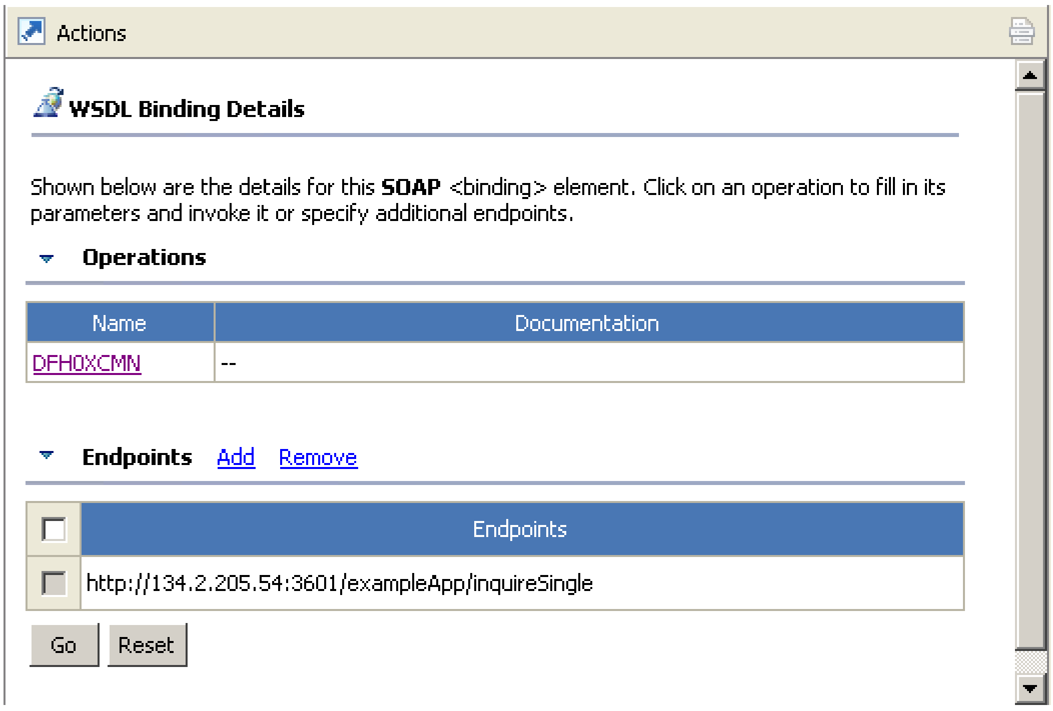
☑ Show the TCP/IP Monitor view when there is activity

TCP/IP Monitors:

Status	Host name	Type	Local Port
Stopped	134.2.205.54:3601	TCP/IP	6565

34.Click the Start button to start the TCP/IP Monitor Server. Notice that the Status changes to Started.

35.Click the OK button to close the Preferences dialog.


Actions

WSDL Binding Details

Shown below are the details for this **SOAP** <binding> element. Click on an operation to fill in its parameters and invoke it or specify additional endpoints.

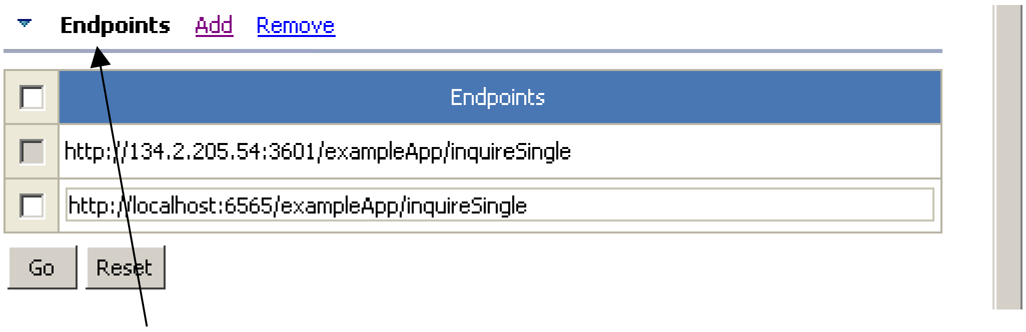
Operations

Name	Documentation
DFH0XCMN	--

Endpoints Add Remove

☐	Endpoints
☐	http://134.2.205.54:3601/exampleApp/inquireSingle

36.Back at the opened Web Services Explorer, from the Navigator pane, click on DFH0XCMNHTTPSoapBinding.

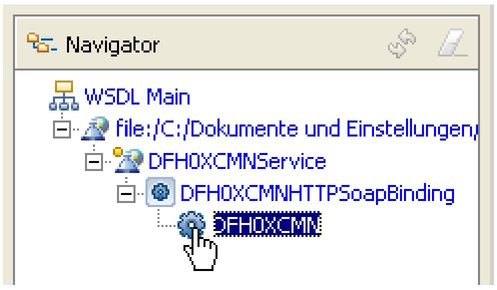


Endpoints
http://134.2.205.54:3601/exampleApp/inquireSingle
http://localhost:6565/exampleApp/inquireSingle

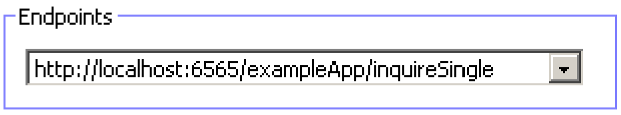
In the Actions pane, click on Add in the Endpoints section and a new Endpoint will appear. Change the Endpoint to http://localhost:6565/exampleApp/inquireSingle

Finally click Go.

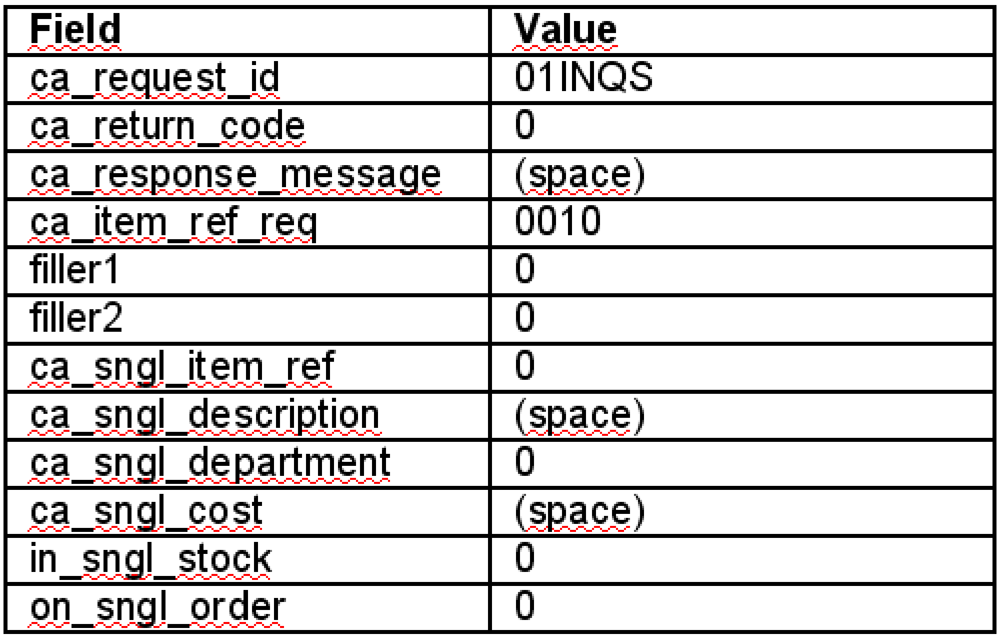
This sets the endpoint to the TCP/IP Monitor. After receiving the Web Services request, the TCP/IP Monitor will then send the request to the destination we specified when we configured the TCP/IP Monitor.



37.In the Navigator Pane, click DFH0XCMN.

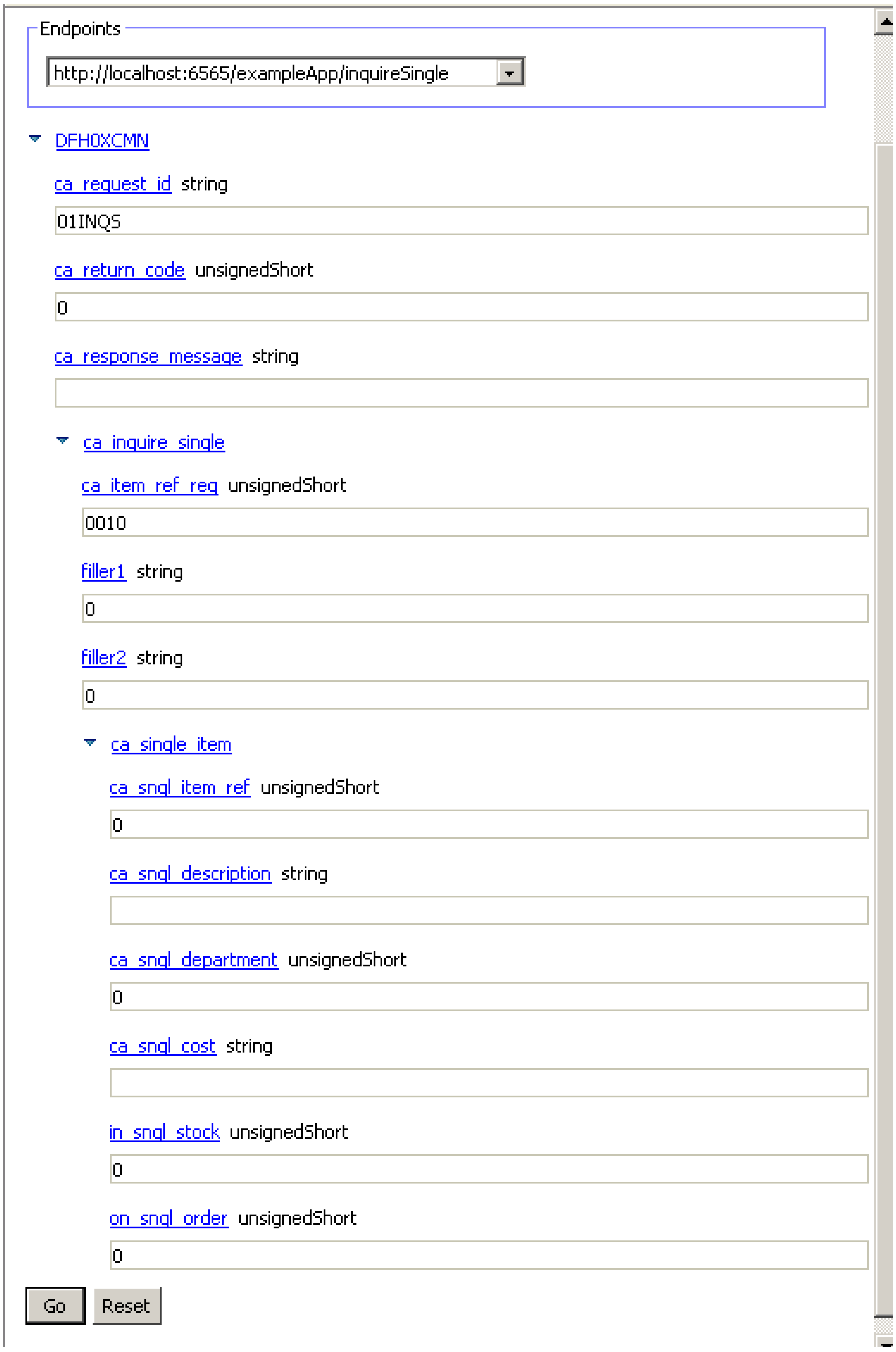


38.In the Actions pane, ensure that the Endpoint http://localhost:6565/exampleApp/inquireSingle is selected.



Field	Value
ca_request_id	01INQS
ca_return_code	0
ca_response_message	(space)
ca_item_ref_req	0010
filler1	0
filler2	0
ca_sngl_item_ref	0
ca_sngl_description	(space)
ca_sngl_department	0
ca_sngl_cost	(space)
in_sngl_stock	0
on_sngl_order	0

39.specify the above values.


Endpoints

http://localhost:6565/exampleApp/inquireSingle

DFH0XCMN

ca_request_id string

01INQS

ca_return_code unsignedShort

0

ca_response_message string

ca_inquire_single

ca_item_ref_req unsignedShort

0010

filler1 string

0

filler2 string

0

ca_single_item

ca_sngl_item_ref unsignedShort

0

ca_sngl_description string

ca_sngl_department unsignedShort

0

ca_sngl_cost string

in_sngl_stock unsignedShort

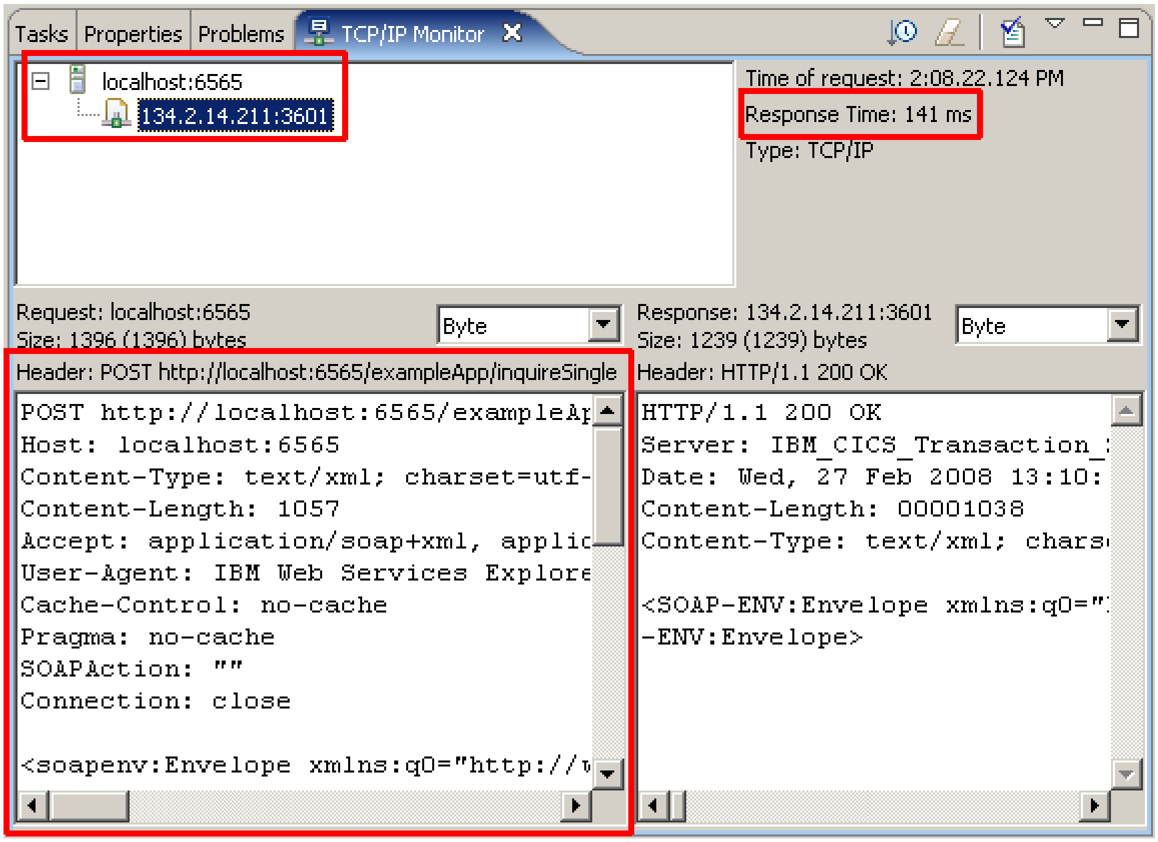
0

on_sngl_order unsignedShort

0

This should look like shown above.

Click the Go button.



Time of request: 2:08.22.124 PM
Response Time: 141 ms
Type: TCP/IP

Request: localhost:6565
Size: 1396 (1396) bytes
Byte
Header: POST http://localhost:6565/exampleApp/inquireSingle

```
POST http://localhost:6565/exampleAp
Host: localhost:6565
Content-Type: text/xml; charset=utf-
Content-Length: 1057
Accept: application/soap+xml, applic
User-Agent: IBM Web Services Explore
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Connection: close

<soapenv:Envelope xmlns:q0="http://w
```

Response: 134.2.14.211:3601
Size: 1239 (1239) bytes
Byte
Header: HTTP/1.1 200 OK

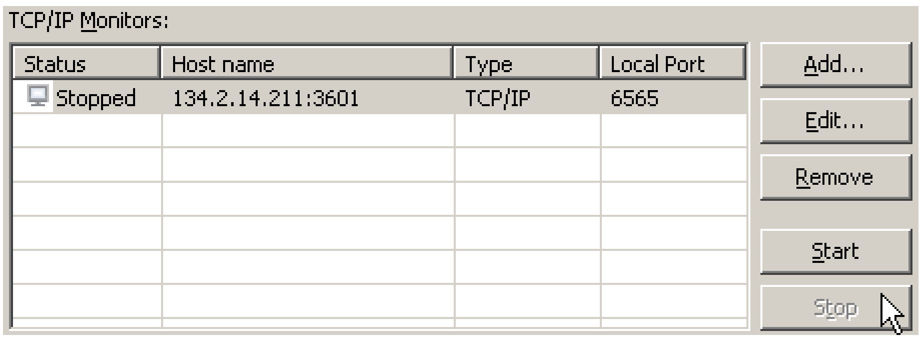
```
HTTP/1.1 200 OK
Server: IBM_CICS_Transaction_
Date: Wed, 27 Feb 2008 13:10:
Content-Length: 00001038
Content-Type: text/xml; chars

<SOAP-ENV:Envelope xmlns:q0="
-ENV:Envelope>
```

40.In addition to the information returned from the Web Service to the Web Services Explorer, you will see a TCP/IP Monitor view.
(Note that you may wish to resize the view to allow it to display more information. Double-click on the TCP/IP Monitor tab to maximize it.)

Note the response time of the request (141 ms on the screen shot above). This is the time from when the request left the TCP/IP Monitor server to the time it received the response from the request.

The TCP/IP Monitor Server shows you the information that was sent 'on the wire'. If you have multiple requests/responses, you can use the tree structure in the upper part of the TCP/IP Monitor view to select the request/response you want to analyze.

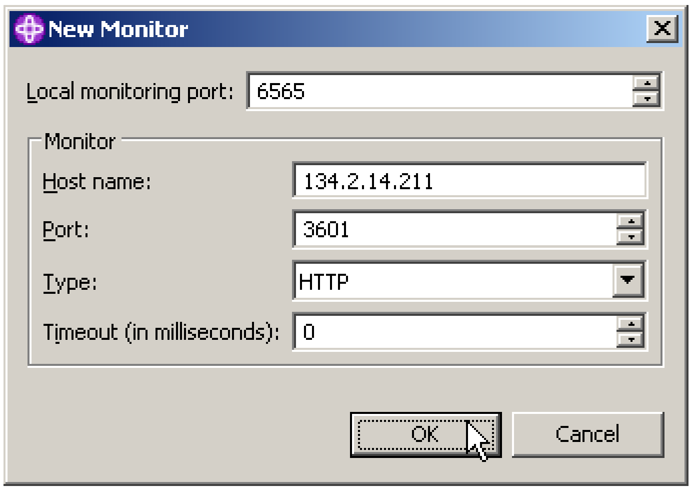
Also note that in addition to the SOAP request and response, you can see the additional HTTP information that flowed with the request and response.



TCP/IP Monitors:

Status	Host name	Type	Local Port
Stopped	134.2.14.211:3601	TCP/IP	6565

41.Now let us experiment a little bit with the TCP/IP Monitor.
Go back to the beginning of this chapter and stop the TCP/IP Monitor.

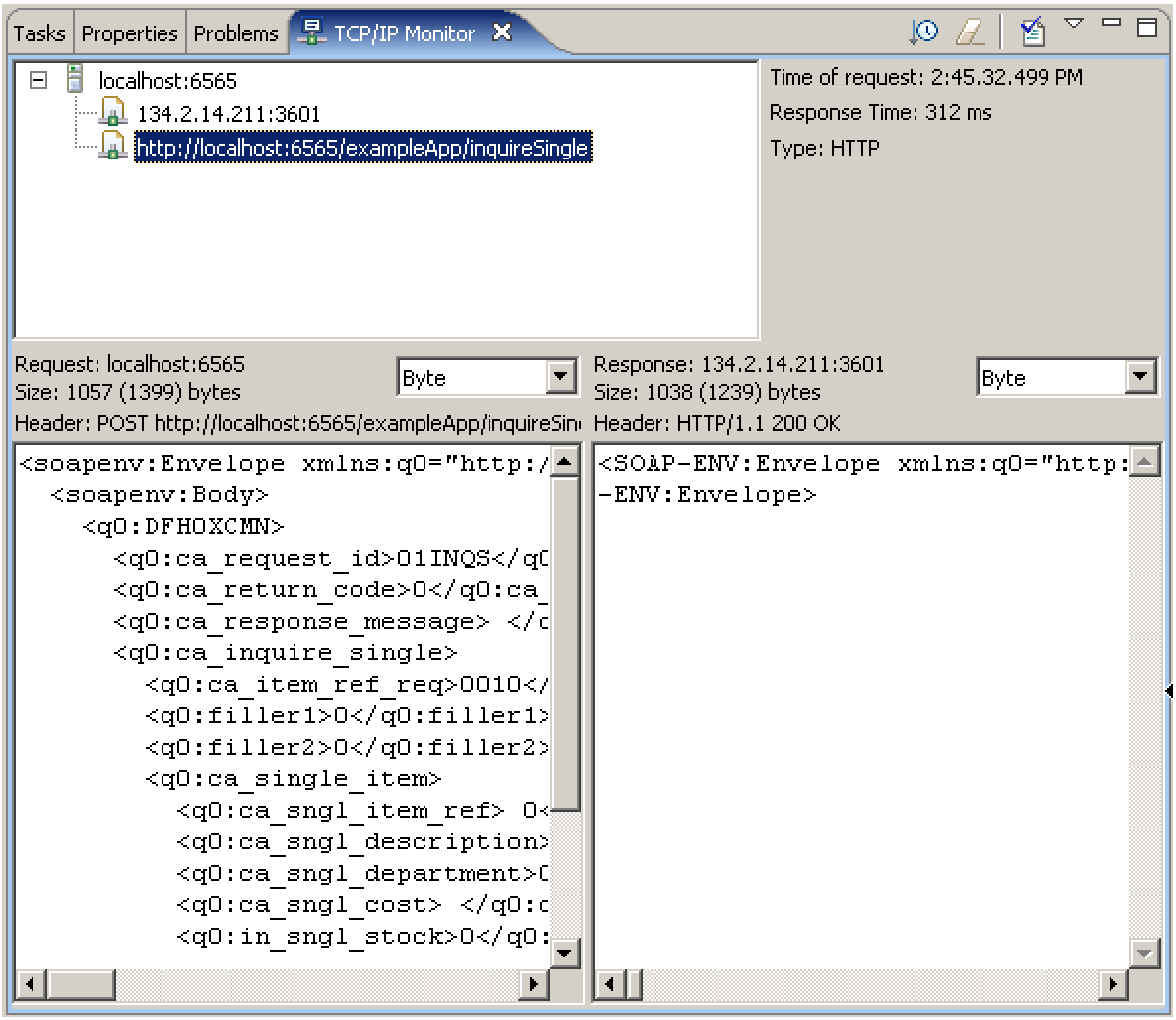

New Monitor

Local monitoring port: 6565

Monitor
Host name: 134.2.14.211
Port: 3601
Type: HTTP
Timeout (in milliseconds): 0

42.Click Add and enter the same values as before, but change the Type to HTTP (instead of TCP/IP).

43.Then proceed as you did above (repeat steps 34-39).



localhost:6565
134.2.14.211:3601
http://localhost:6565/exampleApp/inquireSingle

Time of request: 2:45.32.499 PM
Response Time: 312 ms
Type: HTTP

Request: localhost:6565
Size: 1057 (1399) bytes
Header: POST http://localhost:6565/exampleApp/inquireSin

Byte

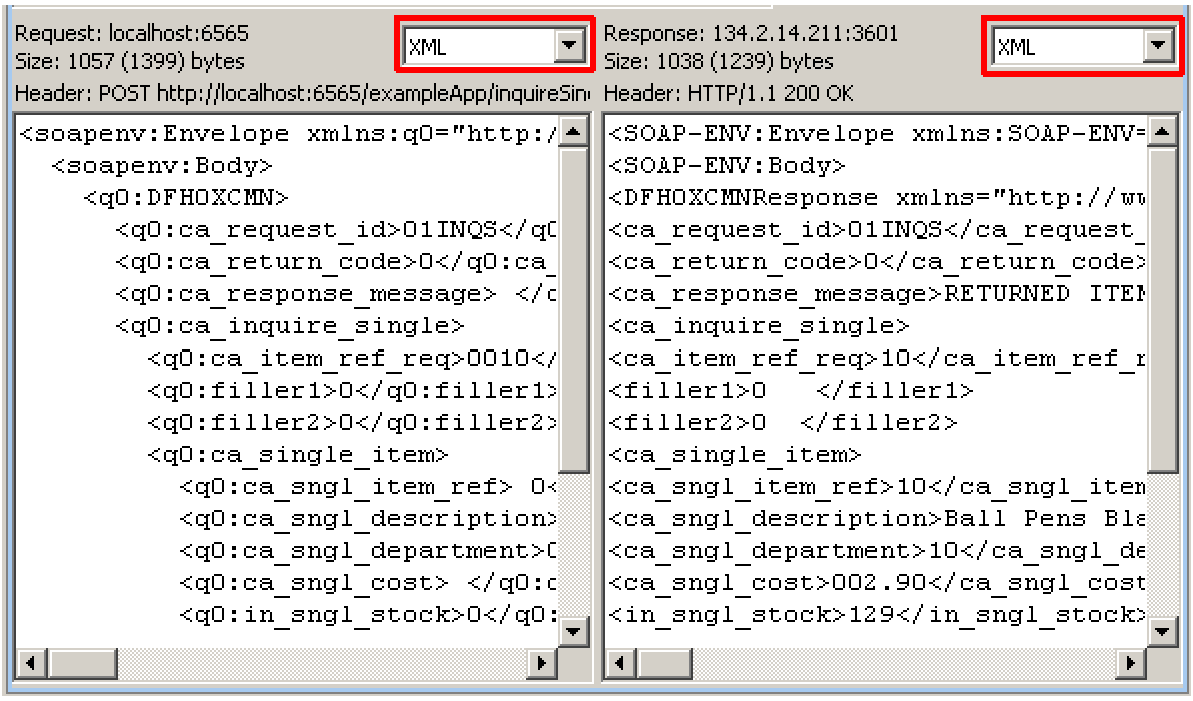
```
<soapenv:Envelope xmlns:q0="http:/
  <soapenv:Body>
    <q0:DFHOXCMN>
      <q0:ca_request_id>01INQS</q0
      <q0:ca_return_code>0</q0:ca_
      <q0:ca_response_message> </c
      <q0:ca_inquire_single>
        <q0:ca_item_ref_req>0010</
        <q0:filler1>0</q0:filler1>
        <q0:filler2>0</q0:filler2>
        <q0:ca_single_item>
          <q0:ca_sngl_item_ref> 0<
          <q0:ca_sngl_description>
          <q0:ca_sngl_department>0
          <q0:ca_sngl_cost> </q0:c
          <q0:in_sngl_stock>0</q0:
```

Response: 134.2.14.211:3601
Size: 1038 (1239) bytes
Header: HTTP/1.1 200 OK

Byte

```
<SOAP-ENV:Envelope xmlns:q0="http:
-ENV:Envelope>
```

44.In the TCP/IP Monitor view you can now only see the SOAP request and response messages.


Request: localhost:6565 XML
Size: 1057 (1399) bytes
Header: POST http://localhost:6565/exampleApp/inquireSin

```
<soapenv:Envelope xmlns:q0="http:/
  <soapenv:Body>
    <q0:DFHOXCMN>
      <q0:ca_request_id>01INQS</q0
      <q0:ca_return_code>0</q0:ca_
      <q0:ca_response_message> </c
      <q0:ca_inquire_single>
        <q0:ca_item_ref_req>0010</
        <q0:filler1>0</q0:filler1>
        <q0:filler2>0</q0:filler2>
        <q0:ca_single_item>
          <q0:ca_sngl_item_ref> 0<
          <q0:ca_sngl_description>
          <q0:ca_sngl_department>0
          <q0:ca_sngl_cost> </q0:c
          <q0:in_sngl_stock>0</q0:
```

Response: 134.2.14.211:3601 XML
Size: 1038 (1239) bytes
Header: HTTP/1.1 200 OK

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=
<SOAP-ENV:Body>
<DFHOXCMNResponse xmlns="http://ww
<ca_request_id>01INQS</ca_request_
<ca_return_code>0</ca_return_code>
<ca_response_message>RETURNED ITEM
<ca_inquire_single>
<ca_item_ref_req>10</ca_item_ref_r
<filler1>0   </filler1>
<filler2>0  </filler2>
<ca_single_item>
<ca_sngl_item_ref>10</ca_sngl_item
<ca_sngl_description>Ball Pens Bla
<ca_sngl_department>10</ca_sngl_de
<ca_sngl_cost>002.90</ca_sngl_cost
<in_sngl_stock>129</in_sngl_stock>
```

45.Now change the view to XML View to see the actual SOAP Message Request and Response in XML

46.Don't forget to stop the TCP/IP Monitor when you are finished

Congratulations, you successfully completed Web Services Tutorial 02!

Resources

Webtut02.zip contains resources as a zip Archiv. Download at

www.cedix.de/Vorles/Band3/Resources/webtut02.zip