ISM Configuration Step by Step Guide SOAP Monitor

Version 1.0

Date	Version	Author	Change
11 Oct 2012	1.0	Michael Wager	Draft Complete
2 Oct 2012	0.9	Michael Wager	Added *Resolve merged namespace conflicts* section
26 Sept 2012	0.2	Michael Wager	Clean up Examples
25 Sept 2012	0.1	Michael Wager	Initial Version

Overview

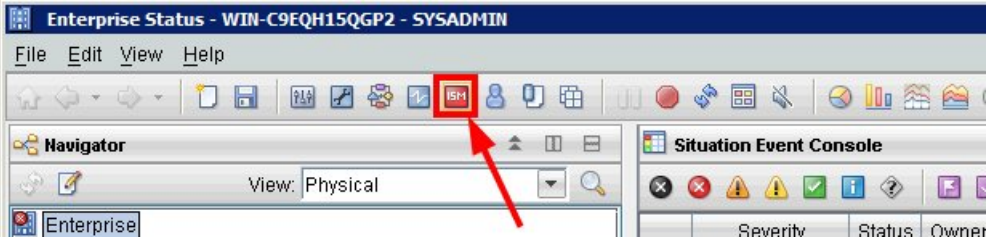
The following guide shows how to configure ISM to monitor the SOAP `getPrice` operation on web server `mywebserver.com` on port `8080`. It shows how to achieve this using the ISM Configuration GUI and also using the ISM Configuration CLI.

ISM does not support testing SOAP requests using links to WSDL files. Neither does ISM support WSDL import statements. This guide shows how to convert multiple WSDL / XSD file(s) into a single file.

ISM supports simple, complex and array parameters for its SOAPInput and SOAPOuput parameters. It also supports attributes. This guide contains detailed examples on all of these parameter types.

1 ISM Configuration GUI

The ISM Configuration GUI is displayed by clicking the ISM Button in the Tivoli Enterprise Portal (TEP):



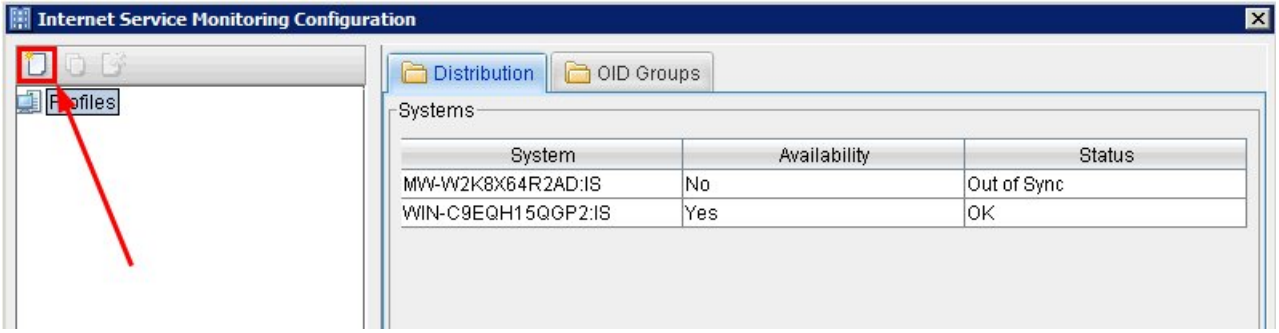
To create a monitor element you need to follow all of the following steps:

- Create a Profile
- Add monitor type to profile
- Create the SOAP profile element
- Deploy the profile to an ISM Agent

1.1 Create a Profile

Create a profile to store the HTTP profile monitor element.

1. Click the **Create New Profiles** button:



2. Type in a name.
3. Click **OK**.

This adds the profile to the navigator tree.

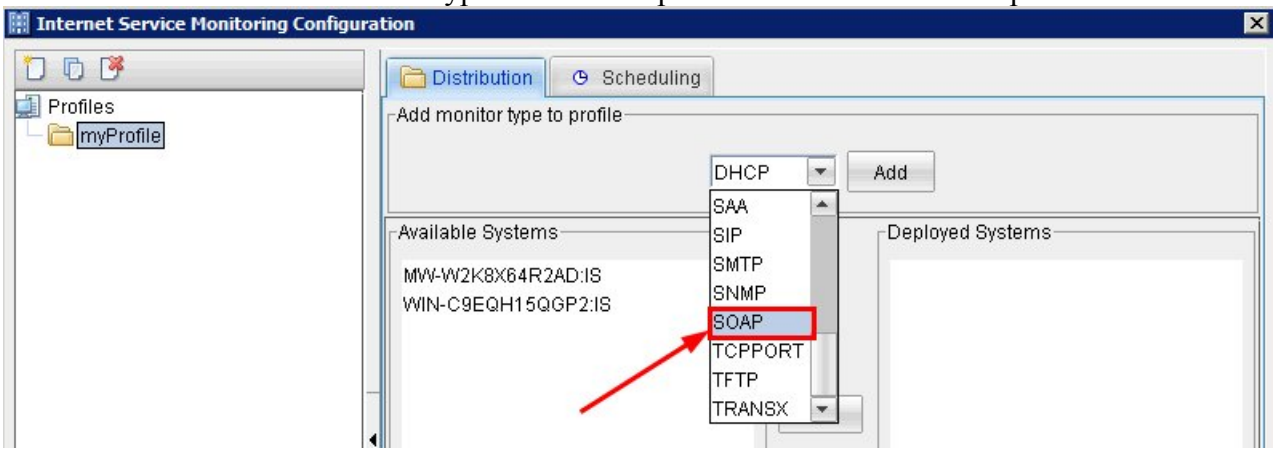
The following example creates a Profile called `myProfile`:

1.2 Add monitor type to Profile

Add a SOAP monitor to the profile to store the SOAP profile monitor element.

1. Select the profile in the navigator tree:

2. Select the **SOAP** monitor type from the drop-down menu to add to the profile:

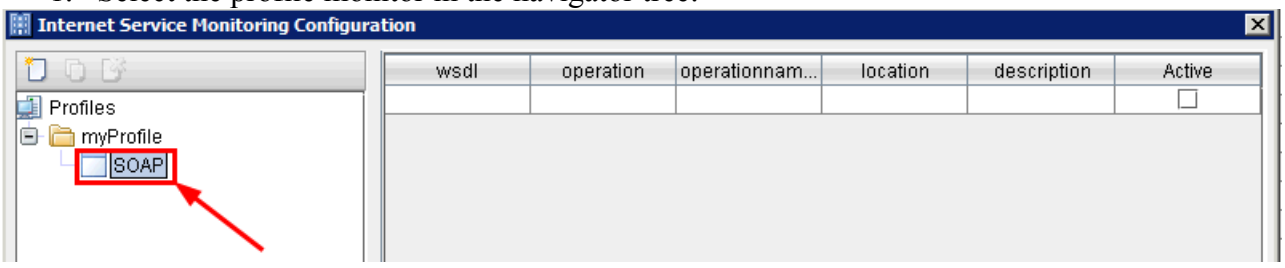


3. Click the **Add** button to add the SOAP monitor type to the navigator tree under the profile:

1.3 Create the SOAP profile element

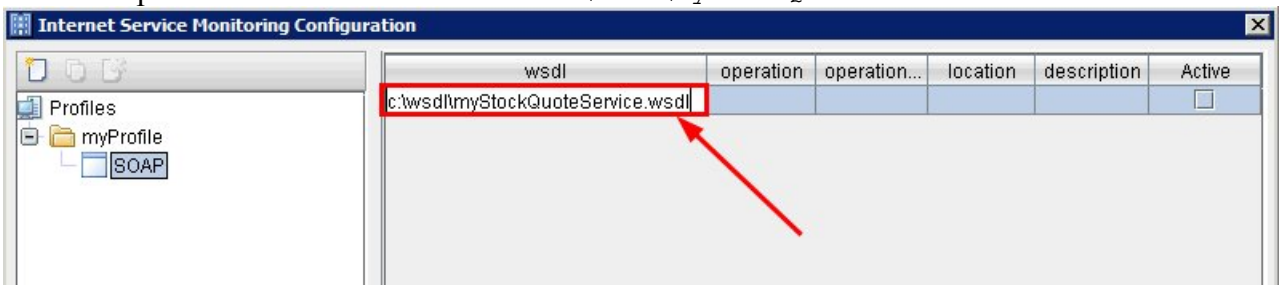
The following steps show how to create the SOAP profile element.

1. Select the profile monitor in the navigator tree:



2. Type in the WSDL file name and click **Apply**.

This example sets the WSDL file name to `c:\wsdl\myStockQuoteService.wsdl`:



Note: wsdl refers to the path of a **local copy** of the WSDL file. ISM does not support a link to a WSDL file. Additionally, ISM does not support WSDL file containing **import** statements. For more information refer to: Appendix A - WSDL Guide

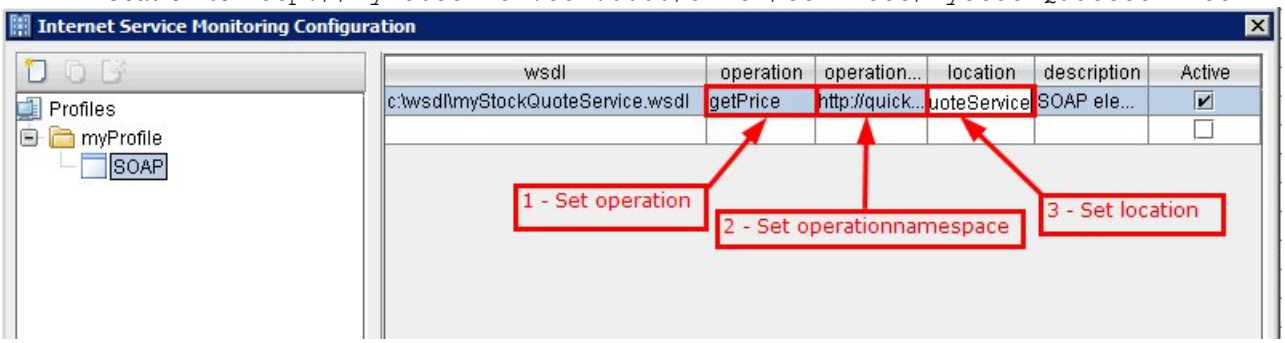
3. Type in the operation, operation namespace and location, and click **Apply**:

This example sets

operation to `getPrice`

operationnamespace to `http://quickstart.samples/xsd`

location to `http://mywebserver.com:8080/axis2/services/MyStockQuoteService`

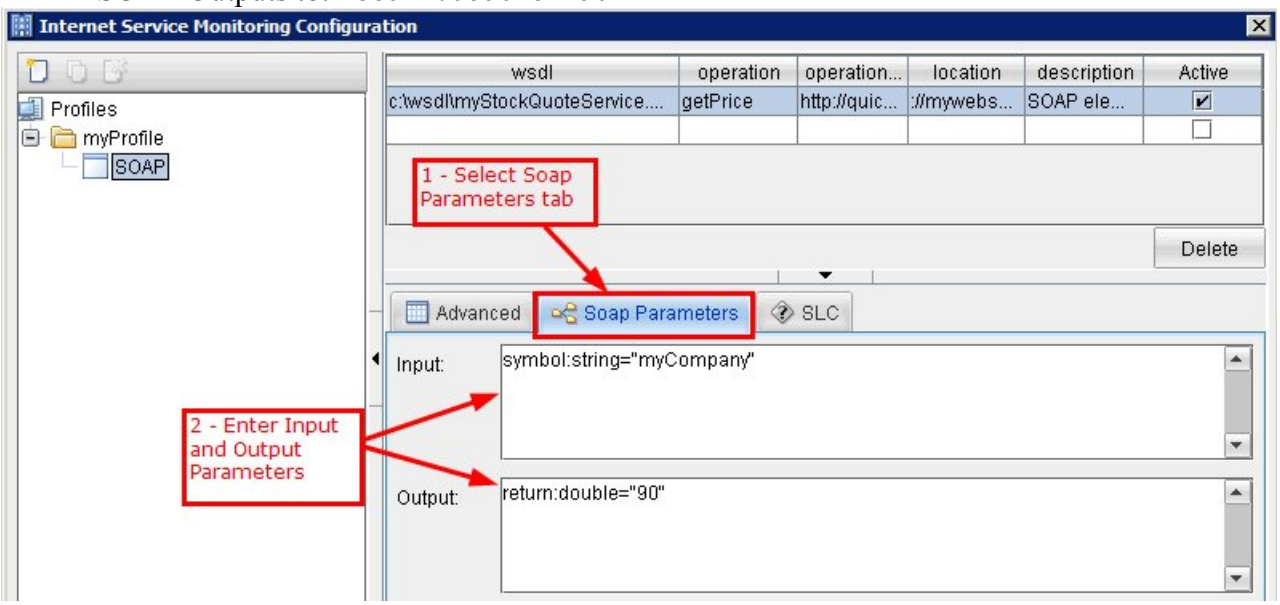


4. Select the Soap Parameters tab and enter the Input and Output parameters:

This example sets

SOAP Inputs to: `symbol:string="myCompany"`

SOAP Outputs to: `return:double="90"`

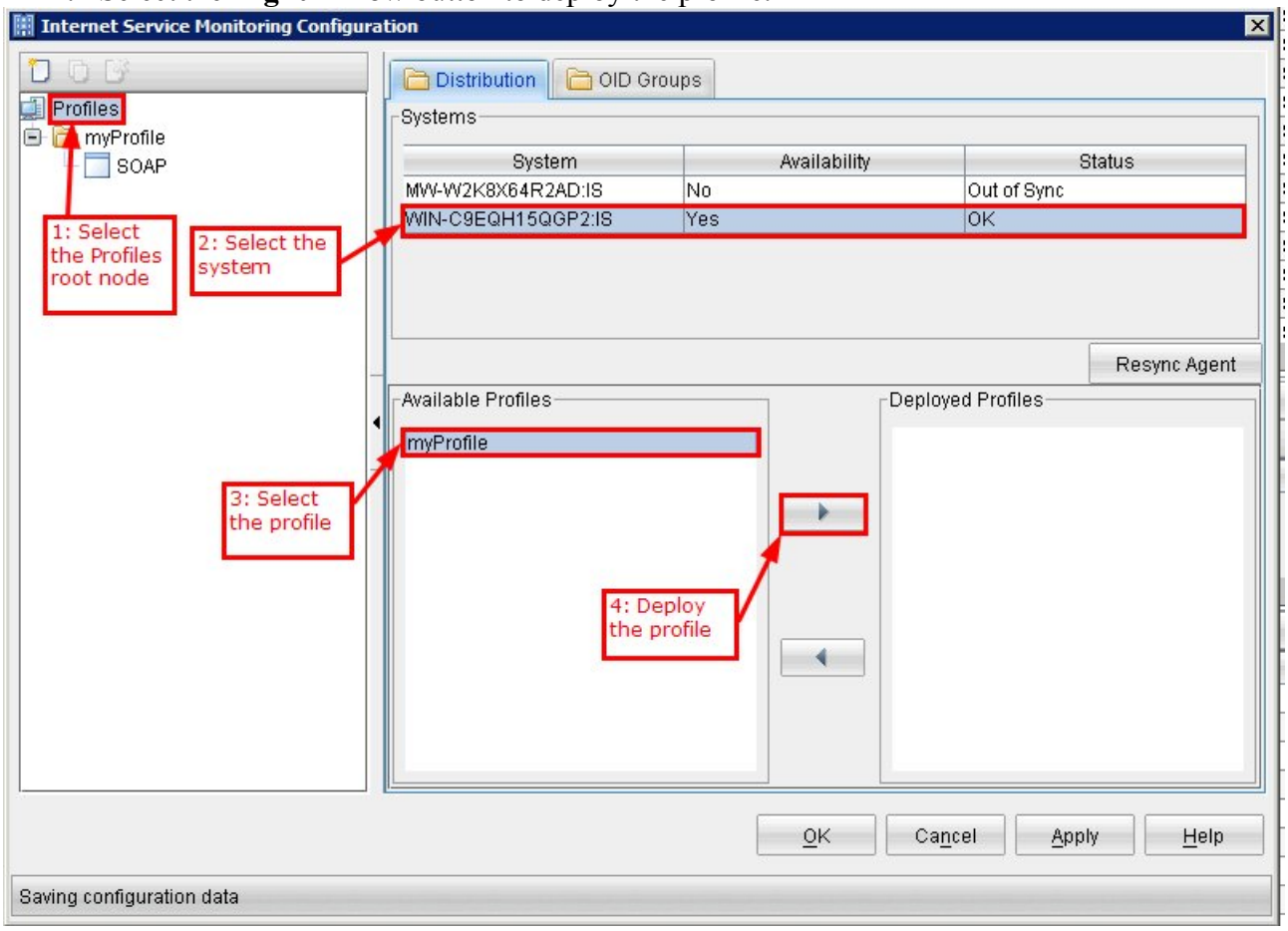


Note: ISM supports simple, complex and array parameters for its SOAP Input/Ouput Parameters. For more information refer to Appendix B - Input/Output Parameters Guide

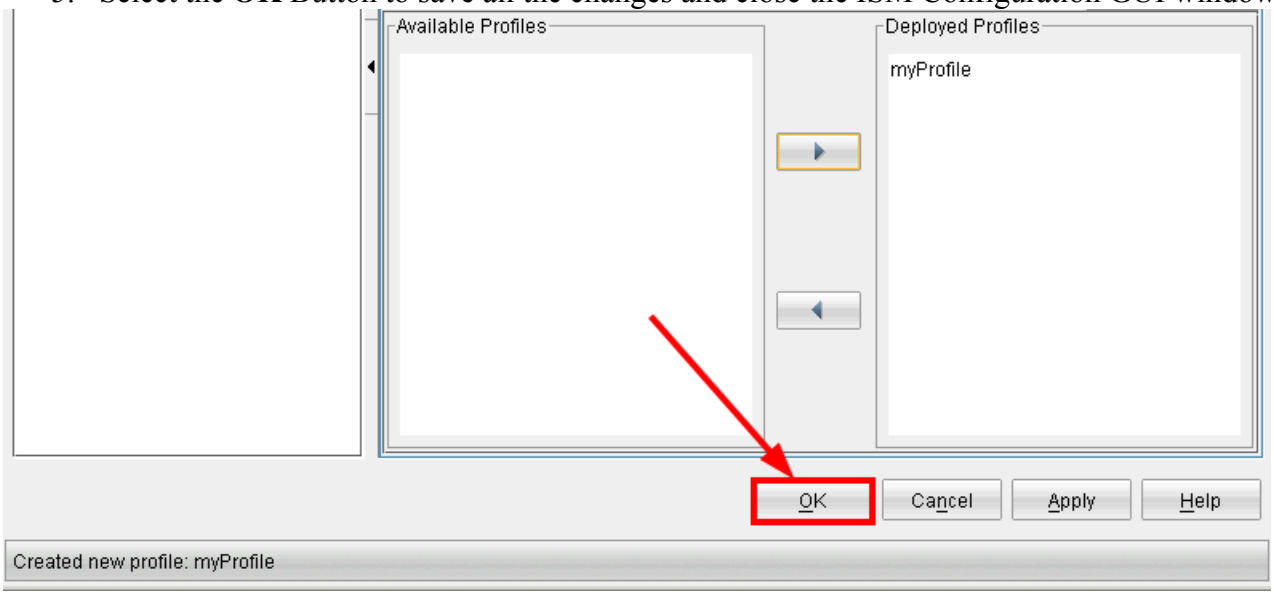
1.4 Deploy the profile to an ISM Agent

The SOAP profile to monitor the `getPrice` operation on `mywebserver.com` on port `8080` has been created. This now needs to be deployed to an ISM agent before the monitoring can begin.

1. Select the **Profiles** root node.
2. Select the system.
3. Select the profile to deploy.
4. Select the **Right Arrow** button to deploy the profile.



5. Select the **OK** Button to save all the changes and close the ISM Configuration GUI window:



2 ISM Configuration CLI

The ISM Configuration CLI can be found at the following locations at the Tivoli Enterprise Portal:

	Windows	Unix
Tivoli Enterprise Portal desktop	`C:\IBM\ITM\CNP\`	`/opt/IBM/ITM/arch/cj/lib/`
Tivoli Enterprise Portal browser	`C:\IBM\ITM\CNB\classes\`	`/opt/IBM/ITM/arch/cw/classes/`

To create a monitor element you need to follow all of the following steps:

- Create a Profile
- Create the SOAP monitor profile element
- Deploy the profile to an ISM Agent

2.1 Create a Profile

A profile needs to be created that will store the SOAP profile monitor element.

The following example shows how to create a profile called `myProfile`:

Windows	`ismconfig.cmd -config -new myProfile`
Unix	`./ismconfig.sh -config -new myProfile`

Administrator: Command Prompt

```
C:\IBM\ITM\CNP>ismconfig.cmd -config -listprofiles
Internet Service Monitoring Configuration
Copyright (c) IBM 2011, 2012
Version: ITCAM_ISM_7.3_0396

Profiles
---------------
C:\IBM\ITM\CNP>ismconfig.cmd -config -new myProfile
Internet Service Monitoring Configuration
Copyright (c) IBM 2011, 2012
Version: ITCAM_ISM_7.3_0396


C:\IBM\ITM\CNP>ismconfig.cmd -config -listprofiles
Internet Service Monitoring Configuration
Copyright (c) IBM 2011, 2012
Version: ITCAM_ISM_7.3_0396

Profiles
---------------
  myProfile
```

Note: The `listprofiles` command lists the profiles. The first `listprofiles` command shows that no profiles exist. The final `listprofiles` command shows that the `myProfile` profile has been created.

2.2 Create the SOAP monitor profile element

The following example shows how to create a SOAP profile element to monitor the `getPrice` operation on web server `mywebserver.com` on port `8080`:

Windows	`ismconfig.cmd -config "-add monitor=SOAP profile=myProfile wsdl=c:\\wsdl\\myStockQuoteService.wsdl operation=getPrice operationnamespace=http://quickstart.samples/xsd location=http://mywebserver.com:8080/axis2/services/MyStockQuoteService @SOAPInputs [symbol:string='myCompany'] @SOAPOutputs [return:double='42']"`
Unix	`./ismconfig.sh -config "-add monitor=SOAP profile=myProfile wsdl=c:\\wsdl\\myStockQuoteService.wsdl operation=getPrice operationnamespace=http://quickstart.samples/xsd location=http://mywebserver.com:8080/axis2/services/MyStockQuoteService @SOAPInputs [symbol:string='myCompany'] @SOAPOutputs [return:double='42']"`

Note: `wsdl` - refers to the path of a **local copy** of the WSDL file. ISM does not support a link to a WSDL file. Additionally, ISM does not support WSDL file containing **import** statements. For more information refer to: Appendix A - WSDL Guide

`@SOAPInputs @SOAPOutputs` - ISM supports simple, complex and array parameters for its SOAP Input/Ouput Parameters. For more information refer to Appendix B - Input/Output Parameters Guide

For more information regarding other parameters consult the Administrators Guide section:
Configuring the SOAP monitor service tests

Administrator: Command Prompt

```
C:\IBM\ITM\CNP>ismconfig.cmd -config "-add monitor=SOAP profile=myProfile
wsdl=c:\\wsdl\\myStockQuoteService.wsdl operation=getPrice
operationnamespace=http://quickstart.samples/xsd
location=http://mywebserver.com:8080/axis2/services/MyStockQuoteService @SOAPInputs [
symbol:string='myCompany' ] @SOAPOutputs [ return:double='42' ]"
Internet Service Monitoring Configuration
Copyright (c) IBM 2011, 2012
Version: ITCAM_ISM_7.3_0396

C:\IBM\ITM\CNP>ismconfig.cmd -config "-listelts profile=myProfile monitor=SOAP"
Internet Service Monitoring Configuration
Copyright (c) IBM 2011, 2012
Version: ITCAM_ISM_7.3_0396

(Profile: myProfile)
[
  Index 0
  Checksum guicli_1348040142958_13943_2673
  ##############################################################################
  Active
  Arguments: {
    timeout = '10'
    password = ''
    description = 'SOAP http://mywebserver.com:8080/axis2/services/MyStock element.'
    retestinterval = '10'
    location = 'http://mywebserver.com:8080/axis2/services/MyStockQuoteService'
    operationnamespace = 'http://quickstart.samples/xsd'
    wsdl = 'c:\wsdl\myStockQuoteService.wsdl'
    poll = '300'
    failureretests = '0'
    username = ''
    operation = 'getPrice'
  }
  SOAP Parameters:
    Inputs:
      symbol:string='myCompany'
    Outputs:
      return:double='42'
]
```

Note: The `listelts` command lists the profile elements. The final `listelts` command shows that the SOAP profile element has been created.

2.3 Deploy the profile to an ISM Agent

The SOAP profile to monitor the `getPrice` operation on `mywebserver.com` on port `8080` has been created. The profile needs to be deployed to an ISM agent before the monitoring can begin.

The following example shows how to deploy profile `myProfile` to agent `WIN-C9EQH15QGP2:IS`

Windows	`ismconfig.cmd -config -deploy "profile=myProfile agent=WIN-C9EQH15QGP2:IS"`
Unix	`./ismconfig.sh -config -deploy "profile=myProfile agent=WIN-C9EQH15QGP2:IS"`

Administrator: Command Prompt

```
C:\IBM\ITM\CNP>ismconfig.cmd -config -deploy "profile=myProfile agent=WIN-
C9EQH15QGP2:IS"
Internet Service Monitoring Configuration
Copyright (c) IBM 2011, 2012
Version: ITCAM_ISM_7.3_0396


C:\IBM\ITM\CNP>ismconfig.cmd -config -listdeployment "profile=myProfile"
Internet Service Monitoring Configuration
Copyright (c) IBM 2011, 2012
Version: ITCAM_ISM_7.3_0396

Profile: myProfile
---------------
  WIN-C9EQH15QGP2:IS
```

Note: The `listdeployment` command lists the deployment of a profile. The final `listdeployment` command shows that the SOAP profile element has been deployed.

Appendix A WSDL Guide

The WSDL parameter used to configure the ISM SOAP element refers to the path of a local copy of the WSDL file. In addition, ISM does not support the import statement. The following steps show how to convert a server WSDL link (that imports other WSDL/XSD links) into a single local WSDL file.

A1 Obtain Local Copy of WSDL file(s)

The ISM SOAP monitor does not support links to a WSDL file.

A1.1 Download the main WDSL file

In the following example the file `stockquoteservice.wsdl` should be downloaded and used instead of using the link `http://mywebserver.com/stockquote/stockquoteservice.wsdl`

http://mywebserver.com/stockquote/stockquoteservice.wsdl

```
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://mywebserver.com/stockquote/service"
          xmlns:tns="http://mywebserver.com/stockquote/service"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns:defs="http://mywebserver.com/stockquote/definitions"
          xmlns="http://schemas.xmlsoap.org/wsdl/">

   <import namespace="http://mywebserver.com/stockquote/definitions"
           location="http://mywebserver.com/stockquote/stockquote.wsdl"/>

    <binding name="StockQuoteSoapBinding" type="defs:StockQuotePortType">
        <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetLastTradePrice">
           <soap:operation soapAction="http://mywebserver.com/GetLastTradePrice"/>
           <input>
               <soap:body use="literal"/>
           </input>
           <output>
               <soap:body use="literal"/>
           </output>
        </operation>
    </binding>

    <service name="StockQuoteService">
        <documentation>My first service</documentation>
        <port name="StockQuotePort" binding="tns:StockQuoteSoapBinding">
           <soap:address location="http://mywebserver.com/stockquote"/>
        </port>
    </service>
</definitions>
```

A1.2 Download any additional WDSL / XSD file(s)

If the main WSDL file contains any import statements, the locations that they point to need to be all downloaded. In the above example, the file `stockquoteservice.wsdl` contains a link to `stockquote.wsdl` that should also be downloaded.

http://mywebserver.com/stockquote/stockquote.wsdl

```
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://mywebserver.com/stockquote/definitions"
          xmlns:tns="http://mywebserver.com/stockquote/definitions"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns:xsd1="http://mywebserver.com/stockquote/schemas"
          xmlns="http://schemas.xmlsoap.org/wsdl/">

   <import namespace="http://mywebserver.com/stockquote/schemas"
           location="http://mywebserver.com/stockquote/stockquote.xsd"/>

    <message name="GetLastTradePriceInput">
        <part name="body" element="xsd1:TradePriceRequest"/>
    </message>

    <message name="GetLastTradePriceOutput">
        <part name="body" element="xsd1:TradePrice"/>
    </message>

    <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
           <input message="tns:GetLastTradePriceInput"/>
           <output message="tns:GetLastTradePriceOutput"/>
        </operation>
    </portType>
</definitions>
```

The `stockquote.wsdl` file imports another xsd file `stockquote.xsd`. This xsd file needs to be downloaded as well.

http://mywebserver.com/stockquote/stockquote.xsd

```
<?xml version="1.0"?>
<schema targetNamespace="http://mywebserver.com/stockquote/schemas"
       xmlns="http://www.w3.org/2001/XMLSchema">

    <element name="TradePriceRequest">
        <complexType>
            <all>
                <element name="tickerSymbol" type="string"/>
            </all>
        </complexType>
    </element>
    <element name="TradePrice">
        <complexType>
            <all>
                <element name="price" type="float"/>
            </all>
        </complexType>
    </element>
</schema>
```

A2 Convert Multiple WSDL files into a Single file

The ISM SOAP monitor does not support the WSDL import statements. Multiple files need to be merged into a single WSDL file.

A2.1 Create a new WSDL file merging all the individual WSDL files

To import WSDL files, replace the import statement with everything that is inside the <definitions>…</definitions> tags of the imported WSDL file.

In the following example, the file `stockquoteservice.wsdl` import statement is merged with the definitions in `stockquote.wsdl` into the merged file `mergedstockquoteservice.1.wsdl`:

stockquoteservice.wsdl

```
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://mywebserver.com/stockquote/service"
          xmlns:tns="http://mywebserver.com/stockquote/service"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns:defs="http://mywebserver.com/stockquote/definitions"
          xmlns="http://schemas.xmlsoap.org/wsdl/">

<import namespace="http://mywebserver.com/stockquote/definitions"
           location="http://mywebserver.com/stockquote/stockquote.wsdl"/>
…
```

Note: The **bold** sections show the import statement to be replaced in `stockquoteservice.wsdl`

stockquote.wsdl

```
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://mywebserver.com/stockquote/definitions"
          xmlns:tns="http://mywebserver.com/stockquote/definitions"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns:xsd1="http://mywebserver.com/stockquote/schemas"
          xmlns="http://schemas.xmlsoap.org/wsdl/">

   <import namespace="http://mywebserver.com/stockquote/schemas"
           location="http://mywebserver.com/stockquote/stockquote.xsd"/>

    <message name="GetLastTradePriceInput">
        <part name="body" element="xsd1:TradePriceRequest"/>
    </message>

    <message name="GetLastTradePriceOutput">
        <part name="body" element="xsd1:TradePrice"/>
    </message>

    <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
           <input message="tns:GetLastTradePriceInput"/>
           <output message="tns:GetLastTradePriceOutput"/>
        </operation>
    </portType>

</definitions>
```

Note: The **bold** sections show the definitions to copy from `stockquote.wsdl`

mergedstockquoteservice.1.wsdl

```
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://mywebserver.com/stockquote/service"
          xmlns:tns="http://mywebserver.com/stockquote/service"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns:defs="http://mywebserver.com/stockquote/definitions"
          xmlns="http://schemas.xmlsoap.org/wsdl/">

   <import namespace="http://mywebserver.com/stockquote/schemas"
           location="http://mywebserver.com/stockquote/stockquote.xsd"/>

    <message name="GetLastTradePriceInput">
        <part name="body" element="xsd1:TradePriceRequest"/>
    </message>

    <message name="GetLastTradePriceOutput">
        <part name="body" element="xsd1:TradePrice"/>
    </message>

    <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
           <input message="tns:GetLastTradePriceInput"/>
           <output message="tns:GetLastTradePriceOutput"/>
        </operation>
    </portType>

    <binding name="StockQuoteSoapBinding" type="defs:StockQuotePortType">
        <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetLastTradePrice">
           <soap:operation soapAction="http://mywebserver.com/GetLastTradePrice"/>
           <input>
               <soap:body use="literal"/>
           </input>
           <output>
               <soap:body use="literal"/>
           </output>
        </operation>
    </binding>

    <service name="StockQuoteService">
        <documentation>My first service</documentation>
        <port name="StockQuotePort" binding="tns:StockQuoteSoapBinding">
           <soap:address location="http://mywebserver.com/stockquote"/>
        </port>
    </service>
</definitions>
```

Note: The **bold** sections have been modified from the original file `stockquoteservice.wsdl`

A2.2 Resolve merged namespace conflicts

The merged WSDL file may have possible namespace conflicts that need to be resolved. To resolve all the namespace conflicts follow all of the following steps:

- A2.2.1 Resolve definitions namespaces
- A2.2.2 Resolve any namespaces in the original wsdl referring to the imported definitions
- A2.2.3 Resolve any namespaces in the imported wsdl file

A2.2.1 Resolve definitions namespaces

The merged definitions may have different namespaces. Compare the original files to compare and resolve any differences.

In the following example, the file `mergedstockquoteservice.1.wsdl` is modified by comparing its definitions with those from the imported `stockquote.wsdl`. The changes are saved in a new file `mergedstockquoteservice.2.wsdl`:

mergedstockquoteservice.1.wsdl - snippet (identical namespaces to stockquoteservice.wsdl)

```
targetNamespace="http://mywebserver.com/stockquote/service"
          xmlns:tns="http://mywebserver.com/stockquote/service"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns:defs="http://mywebserver.com/stockquote/definitions"
          xmlns="http://schemas.xmlsoap.org/wsdl/">
```

Note: The **bold** sections show the differences:

- `targetNamespace` and `tns` are both `http://mywebserver.com/stockquote/`**`service`**
- `xmlns:defs` does not exist in `stockquote.wsdl`

stockquote.wsdl - snippet

```
targetNamespace="http://mywebserver.com/stockquote/definitions"
          xmlns:tns="http://mywebserver.com/stockquote/definitions"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns:xsd1="http://mywebserver.com/stockquote/schemas"
          xmlns="http://schemas.xmlsoap.org/wsdl/">
```

Note: The **bold** sections show the differences:

- `targetNamespace` and `tns` are both `http://mywebserver.com/stockquote/`**`definitions`**
- `xmlns:xsd1` does not exist in `mergedstockquoteservice.1.wsdl`

mergedstockquoteservice.2.wsdl

```
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://mywebserver.com/stockquote/service"
          xmlns:tns="http://mywebserver.com/stockquote/service"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns:xsd1="http://mywebserver.com/stockquote/schemas"
          xmlns:defs="http://mywebserver.com/stockquote/definitions"
          xmlns="http://schemas.xmlsoap.org/wsdl/">

   <import namespace="http://mywebserver.com/stockquote/schemas"
           location="http://mywebserver.com/stockquote/stockquote.xsd"/>

    <message name="GetLastTradePriceInput">
        <part name="body" element="xsd1:TradePriceRequest"/>
    </message>

    <message name="GetLastTradePriceOutput">
        <part name="body" element="xsd1:TradePrice"/>
    </message>

    <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
           <input message="tns:GetLastTradePriceInput"/>
           <output message="tns:GetLastTradePriceOutput"/>
        </operation>
    </portType>
...
```

Note: The **bold** sections show the changes made:

- `targetNamespace` and `tns` are unchanged, the imported definitions shown in *italics* now belong to a different namespace (`http://mywebserver.com/stockquote/service`)
- `xmlns:xsd1` is added since it did not exist and is used by the imported definitions

A2.2.2 Resolve any namespaces in the original wsdl referring to the imported definitions

The original WSDL definitions may refer to the imported namespace. These need to be modified to refer to the target name space.

In the following example, the file `mergedstockquoteservice.2.wsdl` is modified by modifying any namespace that refers to the imported namespace. The changes are saved in a new file `mergedstockquoteservice.3.wsdl`:

stockquoteservice.wsdl

```
...
targetNamespace="http://mywebserver.com/stockquote/service"
          xmlns:tns="http://mywebserver.com/stockquote/service"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns:defs="http://mywebserver.com/stockquote/definitions"
          xmlns="http://schemas.xmlsoap.org/wsdl/">

   <import namespace="http://mywebserver.com/stockquote/definitions"
           location="http://mywebserver.com/stockquote/stockquote.wsdl"/>

    <binding name="StockQuoteSoapBinding" type="defs:StockQuotePortType">
        <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetLastTradePrice">
           <soap:operation soapAction="http://mywebserver.com/GetLastTradePrice"/>
           <input>
               <soap:body use="literal"/>
           </input>
           <output>
               <soap:body use="literal"/>
           </output>
        </operation>
    </binding>
...
```

Note: The **bold** sections show the relevant namespace considerations:

- `import namespace` refers to `http://mywebserver.com/stockquote/definitions`
- `xmlns:defs` define the imported namespace as `defs`
- `binding type` uses the `defs` namespace

mergedstockquoteservice.3.wsdl

```
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://mywebserver.com/stockquote/service"
          xmlns:tns="http://mywebserver.com/stockquote/service"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns:xsd1="http://mywebserver.com/stockquote/schemas"
          xmlns="http://schemas.xmlsoap.org/wsdl/">
...
    <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
        <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetLastTradePrice">
           <soap:operation soapAction="http://mywebserver.com/GetLastTradePrice"/>
           <input>
               <soap:body use="literal"/>
           </input>
           <output>
               <soap:body use="literal"/>
           </output>
        </operation>
    </binding>
...
```

Note: The **bold** sections show the changes made:

- `binding type` is changed from the `defs` namespace to `tns`
- `xmlns:defs` is removed because it is no longer used

A2.2.3 Resolve any namespaces in the imported wsdl file

Review the namespaces of the definitions in the imported WSDL file and check that they are correct.

In the following example, the file `mergedstockquoteservice.3.wsdl` is reviewed. However as no changes are required it remains unchanged:

mergedstockquoteservice.3.wsdl

```
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://mywebserver.com/stockquote/service"
          xmlns:tns="http://mywebserver.com/stockquote/service"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns:xsd1="http://mywebserver.com/stockquote/schemas"
          xmlns="http://schemas.xmlsoap.org/wsdl/">

   <import namespace="http://mywebserver.com/stockquote/schemas"
           location="http://mywebserver.com/stockquote/stockquote.xsd"/>

    <message name="GetLastTradePriceInput">
        <part name="body" element="xsd1:TradePriceRequest"/>
    </message>

    <message name="GetLastTradePriceOutput">
        <part name="body" element="xsd1:TradePrice"/>
    </message>

    <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
           <input message="tns:GetLastTradePriceInput"/>
           <output message="tns:GetLastTradePriceOutput"/>
        </operation>
    </portType>

    <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
        <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetLastTradePrice">
           <soap:operation soapAction="http://mywebserver.com/GetLastTradePrice"/>
           <input>
               <soap:body use="literal"/>
           </input>
           <output>
               <soap:body use="literal"/>
           </output>
        </operation>
    </binding>

    <service name="StockQuoteService">
        <documentation>My first service</documentation>
        <port name="StockQuotePort" binding="tns:StockQuoteSoapBinding">
           <soap:address location="http://mywebserver.com/stockquote"/>
        </port>
    </service>
</definitions>
```

Note: The *italic* sections refer to the imported definitions.

- `xsd1` namespace is used in the `message` definitions. This namespace has been added.
- `portType operation` uses the `tns` namespace. Although `tns` has changed from the imported definitions, it does not need to change since the messages that they are referring to are now also in the `tns` namespace.

The above process needs to be repeated for every import WSDL statement.

A2.3 Modify the new WSDL file merging all the individual XSD files

To import XSD files, replace the import statement with everything that is inside the <schema>…</schema> tags of the imported XSD file and nest them inside <types></types> tags.
In the following example, the file `mergedstockquoteservice.3.wsdl` is modified by merging with `stockquote.xsd` and saving the results in a new file `mergedstockquoteservice.4.wsdl`.

Note: The **bold** section represents the merged section

mergedstockquoteservice.4.wsdl

```
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://mywebserver.com/stockquote/service"
          xmlns:tns="http://mywebserver.com/stockquote/service"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns:xsd1="http://mywebserver.com/stockquote/schemas"
          xmlns="http://schemas.xmlsoap.org/wsdl/">
    <types>
       <schema targetNamespace="http://mywebserver.com/stockquote/schemas"
             xmlns="http://www.w3.org/2001/XMLSchema">
          <element name="TradePriceRequest">
             <complexType>
                 <all>
                    <element name="tickerSymbol" type="string"/>
                 </all>
             </complexType>
          </element>
          <element name="TradePrice">
             <complexType>
                 <all>
                    <element name="price" type="float"/>
                 </all>
             </complexType>
          </element>
       </schema>
    </types>
    <message name="GetLastTradePriceInput">
        <part name="body" element="xsd1:TradePriceRequest"/>
    </message>
    <message name="GetLastTradePriceOutput">
        <part name="body" element="xsd1:TradePrice"/>
    </message>

    <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
           <input message="tns:GetLastTradePriceInput"/>
           <output message="tns:GetLastTradePriceOutput"/>
        </operation>
    </portType>

    <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
        <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetLastTradePrice">
           <soap:operation soapAction="http://mywebserver.com/GetLastTradePrice"/>
           <input>
               <soap:body use="literal"/>
           </input>
           <output>
               <soap:body use="literal"/>
           </output>
        </operation>
    </binding>

    <service name="StockQuoteService">
        <documentation>My first service</documentation>
        <port name="StockQuotePort" binding="tns:StockQuoteSoapBinding">
           <soap:address location="http://mywebserver.com/stockquote"/>
        </port>
    </service>
</definitions>
```

Appendix B Input/Output Parameters Guide

ISM supports simple, complex and array parameters for its SOAPInput and SOAPOuput parameters.

B1 Syntax

The SOAPInputs and SOAPOutputs parameters both follow the same well-defined syntax. The following example shows how to view the syntax of these parameters using the inline help:

Administrator: Command Prompt

```
C:\IBM\ITM\CNP>ismconfig.cmd -config -help SOAPParams
Internet Service Monitoring Configuration
Copyright (c) IBM 2011, 2012
Version: ITCAM_ISM_7.3_0396

Topic name: SOAPParams

Description:
The @SOAPInputs/@SOAPOutputs groups in element/step creation/modification
commands are used to specify SOAP input (request) and output (response)
parameters, respectively. Their common format is as follows:

  (@SOAPInputs | @SOAPOutputs) '[' (<parameter> (',' <parameter>)*)? ']'
  <parameter>          ::= <identifier> ':' <type>
                           (<attributes>) ('=' <assignment>)?
  <identifier>         ::= [a-zA-Z_][a-zA-Z0-9_]*
  <type>               ::= (<simpletype> | <complextype>) <arraymodifier>?
  <attributes>         ::= '(' (<attribute> (',' <attribute>)*)? ')'
  <attribute>          ::= <attrname> (':' <attrtype>) '=' <attrvalue>
  <attrname>           ::= <identifier>
  <attrtype>           ::= <identifier>
  <attrvalue>          ::= <identifier>
  <arraymodifier>      ::= '[]'
  <simpletype>         ::= <identifier>
  <complextype>        ::= '{' <parameter> (',' <parameter>)* '}'
  <assignment>         ::= (<simpleassignment> | <complexassignment> |
                           <arrayassignment>)
  <simpleassignment>   ::= ([0-9]+ | ("|') <anychar>* ("|'))
  <complexassignment> ::= '{' <identifier> '=' <assignment>
                           (',' <identifier> '=' <assignment>)* '}'
                           (<attributes>)
  <arrayassignment>    ::= '[' <assignment>* ']'

Where <anychar> is any character, and allowed escape sequences are \\,
\t, \r, \n, ",  and '. As should be expected, <simpleassignment>,
<complexassignment> and <arrayassignment> can only be applied to simple,
complex and array types, respectively. If a value is specified in an output
type, this value will be taken as a regular expression specifying the
expected output.

Examples:
  To specify two simple request parameters (one integer, one string), a
  and b, with values 123 and "xyz", we supply the following:
    @SOAPInputs [a:int=123, b:string=\'xyz\']

    (Unfortunately the quotes must be escaped, so the shell does not
     eliminate them.)

  To specify a parameter whose type is an array of complex parameters, we
  supply the following:
    @SOAPOutputs [result: {x:integer, y:dateTime}[]
        = [{x=123}, {x=456, y=\'^1999\'}]]

    (In this example, the response will be checked for an array of two
     elements, one with an 'x' value of 123, and any 'y' value, and the
     other with an 'x' value of 456, and a 'y' value which begins with
     '1999'.)
```

B2 Simple Type

B2.1 Simple Type Example

The following parameters show how to specify a `myCompany` string as input and match the response '`90`':

```
@SOAPInputs [ symbol:string='myCompany' ]
@SOAPOutputs [ return:double='90' ]
```

The `@SOAPInputs` parameter is used by the ISM monitor to send the following SOAP request to the web server:

Request

```
<?xml version='1.0' encoding='utf-8' ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
<ns1:getPrice xmlns:ns1="http://quickstart.samples/xsd">
<ns1:symbol>myCompany</ns1:symbol>
</ns1:getPrice>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note: The **bold** sections show what is defined by the `@SOAPInputs` parameter.

The `@SOAPOutputs` parameter is used by the ISM monitor to match the following SOAP response from the web server:

Request

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Body>
<ns:getPriceResponse xmlns:ns="http://quickstart.samples/xsd">
<ns:return>90</ns:return>
</ns:getPriceResponse>
</soapenv:Body>
</soapenv:Envelope>
```

Note: The **bold** sections show what is matched by the `@SOAPOutputs` parameter.

B3 Complex Type

B3.1 Complex Type Example

The following parameter show how to specify a complex request `booking` containing `type` and `airline` complex types:

```
@SOAPInputs [ booking:{type:string,airline:string}={type='flight',airline='myAir'} ]
```

The `@SOAPInputs` parameter is used by the ISM monitor to send the following SOAP request to the web server:

Request

```
<?xml version='1.0' encoding='utf-8' ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
<ns1:bookFlight xmlns:ns1="http://quickstart.samples/xsd">
<ns1:booking xmlns:ns1="http://quickstart.samples/xsd">
<ns1:type>flight</ns1:type>
<ns1:airline>myAir</ns1:airline>
</ns1:booking>
</ns1:bookFlight>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note: The **bold** sections show what is defined by the `@SOAPInputs` parameter.

B3.2 Complex Type with Attributes

The following parameter example shows how to specify a complex request `booking` containing `requests` and `airline` complex types with attributes (shown in italics):

```
@SOAPInputs [ booking:{requests:string,airline:string}(type:string='flight')=
{requests(meals:string='western')='',
airline(from:string='per',dest:string='tokyo')='myAir'} ]
```

The `@SOAPInputs` parameter is used by the ISM monitor to send the following SOAP request to the web server:

Request

```
<?xml version='1.0' encoding='utf-8' ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
<ns1:bookFlight xmlns:ns1="http://quickstart.samples/xsd">
<ns1:booking xmlns:ns1="http://quickstart.samples/xsd" type="flight">
<ns1:requests meals="western"></ns1:requests>
<ns1:airline from="per" dest="tokyo">myAir</ns1:airline>
</ns1:booking>
</ns1:bookFlight>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note: The **bold** sections show what is defined by the `@SOAPInputs` parameter, and the ***bold italic*** sections show the attributes.

B4 Array Type

B4.1 Array of Simple Types

The following parameter shows how to specify an array of `input` strings:

```
@SOAPInputs [ input:string[]=['TESTSTRING1','TEST2'] ]
```

The `@SOAPInputs` parameter is used by the ISM monitor to send the following SOAP request to the web server:

Request

```
<?xml version='1.0' encoding='utf-8' ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
<ns1:echoString xmlns:ns1="http://quickstart.samples/xsd">
<ns1:input>TESTSTRING1</ns1:input>
<ns1:input>TEST2</ns1:input>
</ns1:echoString>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note: The **bold** sections show what is defined by the `@SOAPInputs` parameter.

B4.2 Array of Complex Types

The following parameter shows how to specify an array of `opposite` complex types:

```
@SOAPInputs [ opposite:{item1:string,item2:string}[]=
[{item1='big',item2='small'},{item1='slow',item2='fast'}] ]
```

The `@SOAPInputs` parameter is used by the ISM monitor to send the following SOAP request to the web server:

Request

```
<?xml version='1.0' encoding='utf-8' ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
<ns1:echoString xmlns:ns1="http://quickstart.samples/xsd">
<ns1:opposite xmlns:ns1="http://quickstart.samples/xsd">
<ns1:item1>big</ns1:item1>
<ns1:item2>small</ns1:item2>
</ns1:opposite>
<ns1:opposite xmlns:ns1="http://quickstart.samples/xsd">
<ns1:item1>slow</ns1:item1>
<ns1:item2>fast</ns1:item2>
</ns1:opposite>
</ns1:echoString>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note: The **bold** sections show what is defined by the `@SOAPInputs` parameter.

B4.3 Array of Complex Types with Attributes

The following parameter example shows how to specify an array of `itinerary` complex types with attributes (shown in italics):

```
@SOAPInputs [ itinerary:{from:string,dest:string}[]=
[{from(depart:string='4:00pm')='per',
dest(arrive:string='8:00pm')='syd'}(date:string='2012 03 12'),
{from(depart:string='10:00am')='syd',
dest(arrive:string='11:00am')='mel'}(date:string='2012 03 15'),
{from(depart:string='11:30am')='mel',
dest(arrive:string='4:00pm')='per'}(date:string='2012 03 19')] ]
```

The `@SOAPInputs` parameter is used by the ISM monitor to send the following SOAP request to the web server:

Request

```
<?xml version='1.0' encoding='utf-8' ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
<ns1:saveItinerary xmlns:ns1="http://quickstart.samples/xsd">
<ns1:itinerary xmlns:ns1="http://quickstart.samples/xsd" date="2012 03 12">
<ns1:from depart="4:00pm">per</ns1:from>
<ns1:dest arrive="8:00pm">syd</ns1:dest>
</ns1:itinerary>
<ns1:itinerary xmlns:ns1="http://quickstart.samples/xsd" date="2012 03 15">
<ns1:from depart="10:00am">syd</ns1:from>
<ns1:dest arrive="11:00am">mel</ns1:dest>
</ns1:itinerary>
<ns1:itinerary xmlns:ns1="http://quickstart.samples/xsd" date="2012 03 19">
<ns1:from depart="11:30am">mel</ns1:from>
<ns1:dest arrive="4:00pm">per</ns1:dest>
</ns1:itinerary>
</ns1:saveItinerary>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note: The **bold** sections show what is defined by the `@SOAPInputs` parameter, and the ***bold italic*** sections show the attributes.

B5 Complex Array Types

B5.1 Complex Type with an inner array

The following parameter example shows how to specify a `shoppingList` Complex Type with an inner array of items:

```
@SOAPInputs [ shoppingList:{item:string[]}={item=['Bread','Milk','Butter']} ]
```

The `@SOAPInputs` parameter is used by the ISM monitor to send the following SOAP request to the web server:

Request

```
<?xml version='1.0' encoding='utf-8' ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
<ns1:saveList xmlns:ns1="http://quickstart.samples/xsd">
<ns1:shoppingList xmlns:ns1="http://quickstart.samples/xsd">
<ns1:item>Bread</ns1:item>
<ns1:item>Milk</ns1:item>
<ns1:item>Butter</ns1:item>
</ns1:shoppingList>
</ns1:saveList>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note: The **bold** sections show what is defined by the `@SOAPInputs` parameter.

B5.2 Complex Type with an inner array and attributes

The following parameter shows how to specify a `shoppingList` Complex Type with an inner array of items with attributes (shown in italics):

```
@SOAPInputs [ shoppingList:{item:string[]}=
{item(shop:string='myShop')=['Bread','Milk','Butter']} ]
```

The `@SOAPInputs` parameter is used by the ISM monitor to send the following SOAP request to the web server:

Request

```
<?xml version='1.0' encoding='utf-8' ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
<ns1:saveList xmlns:ns1="http://quickstart.samples/xsd">
<ns1:shoppingList xmlns:ns1="http://quickstart.samples/xsd" shop="myShop">
<ns1:item>Bread</ns1:item>
<ns1:item>Milk</ns1:item>
<ns1:item>Butter</ns1:item>
</ns1:shoppingList>
</ns1:saveList>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note: The **bold** sections show what is defined by the `@SOAPInputs` parameter, and the ***bold italic*** sections show the attributes.