

Using the SOA Gateway with LE

The SOA Gateway interfaces with LE ...

The following is a sample SOA Gateway 'WebService' definition for a LE program exposed as a 'Web Service'

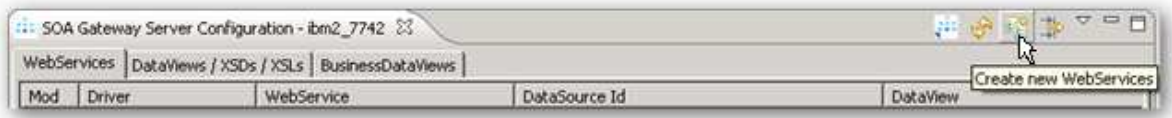
Before a SOA Gateway WebService can be used, the mapping between the physical layout (the LE program's parameters) and the SOA Gateway (XML) representation must be in place. This mapping is called the SOA Gateway 'DataView'. In addition to the manual approach described in detail in the Data Views section, there are a number of semi-automated methods aiding in the generation of SOA Gateway DataViews for LE resources

1. dynamically create a DataView for a LE program
-

Creating a SOA Gateway DataView for a LE program

A new DataView can be generated for a LE program using the 'WebService Discovery wizard', which will not only generate the DataView (XRD), but also a XSD, plus the SOA Gateway Resource definition itself.

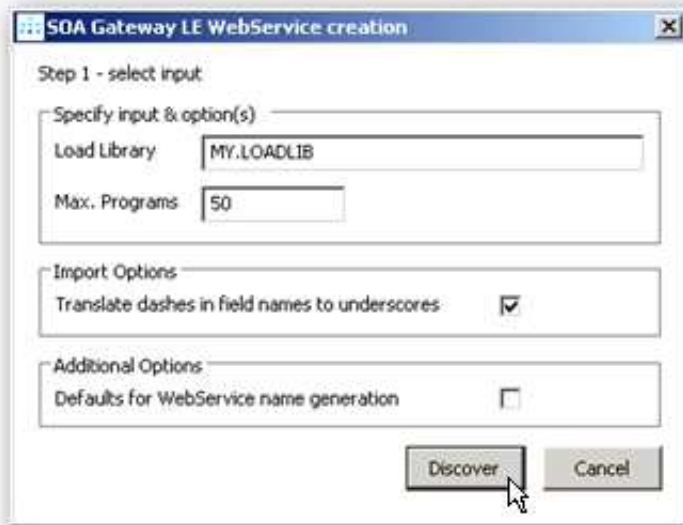
1. Download the COBOL-source for the LE sample program to your local disk drive from here and compile it.
2. Start the Discovery Wizard from the Servers View as described here' or from the Configuration view as follows:



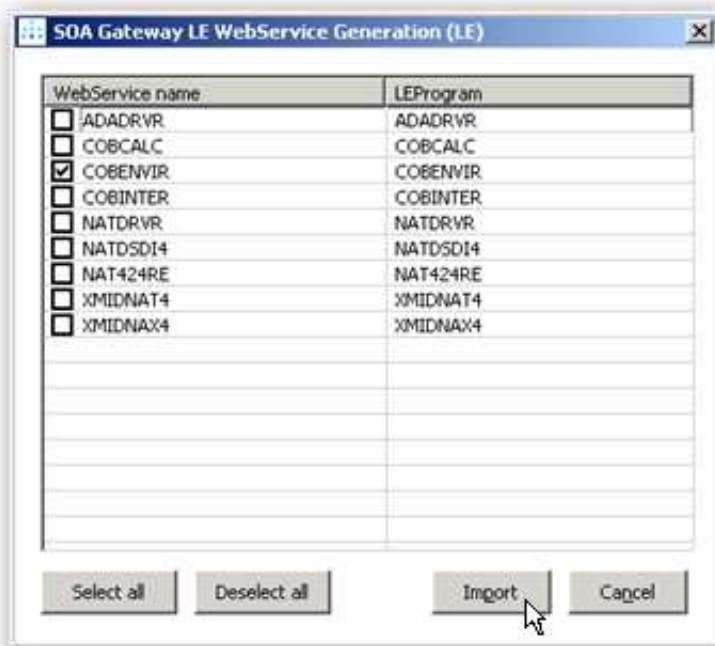
3. Select the *LE_Driver* and click **Next >**.



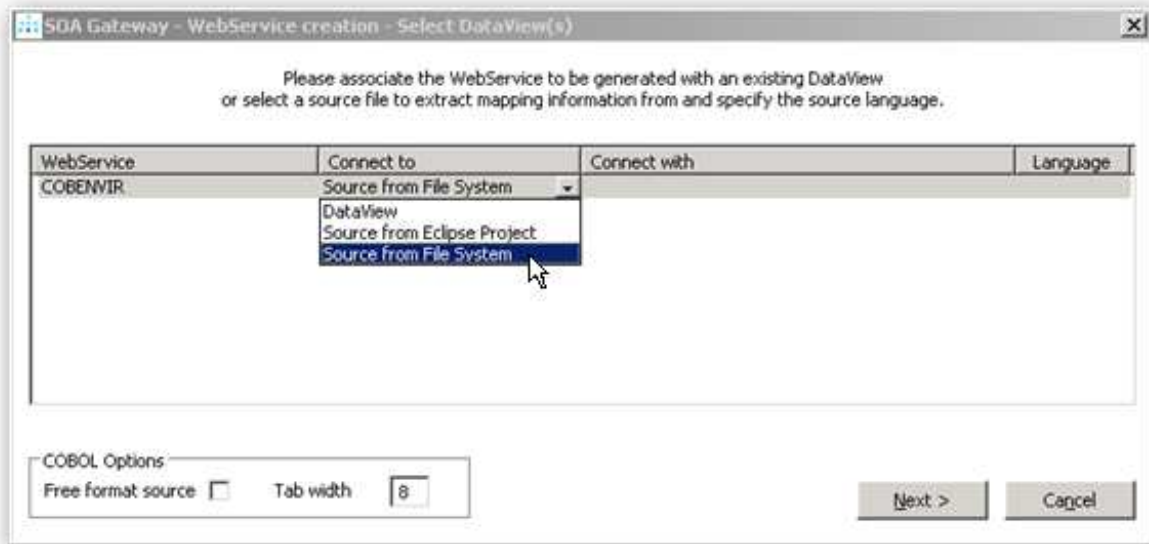
4. Enter the name of the load library the sample has been compiled into. Click **Discover**



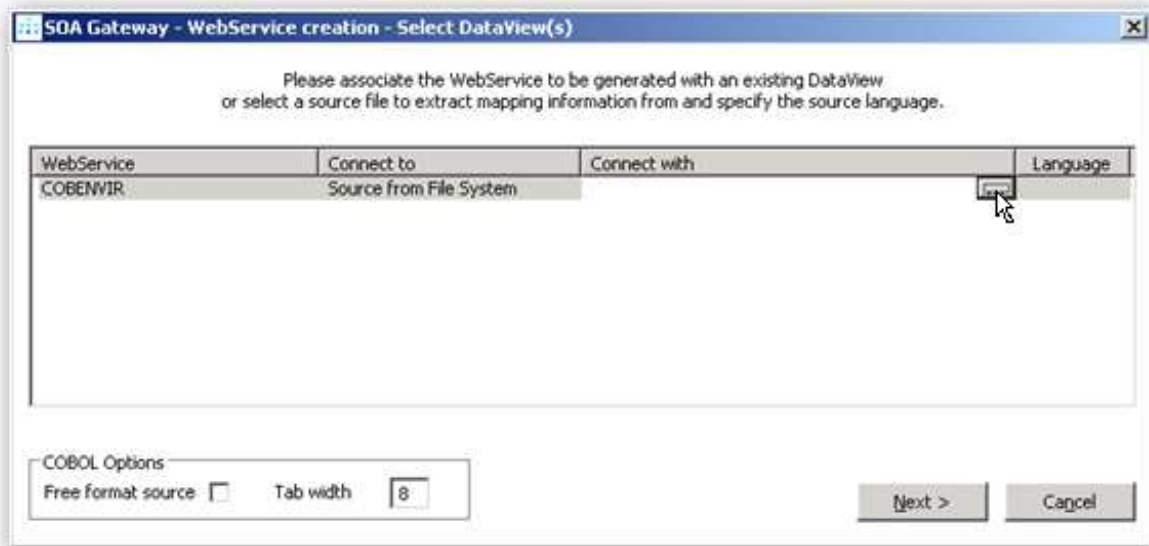
5. Select the *COBENVIR* sample, click **Import**



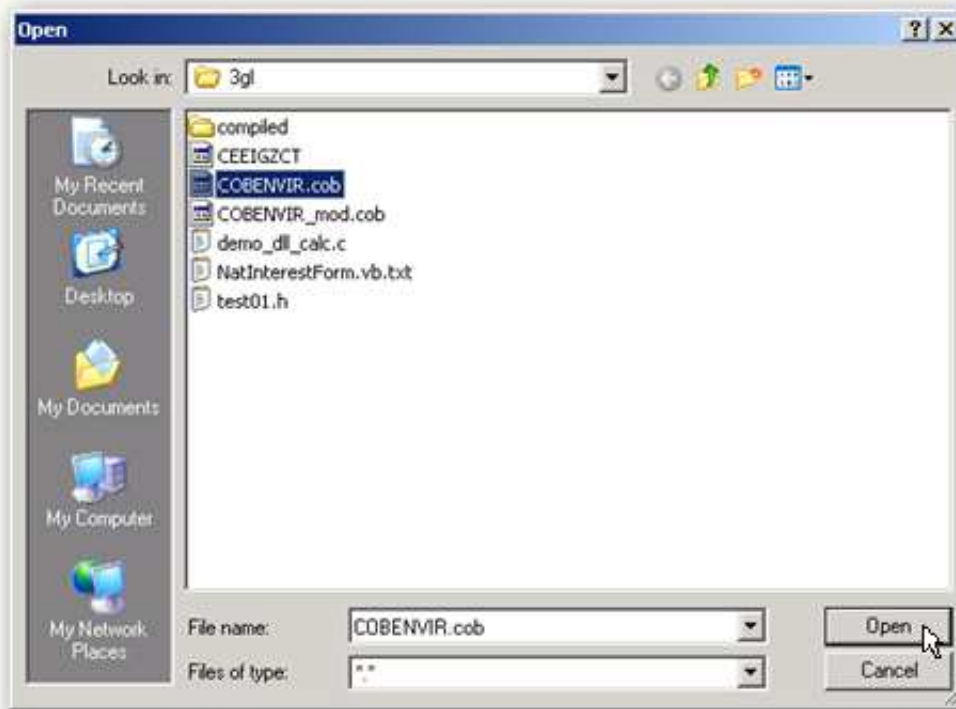
6. You will now associate the load module with the source, which will be used to generate the SOA Gateway DataView. From the *Connect to* column's dropdown box select *Source from File System*.



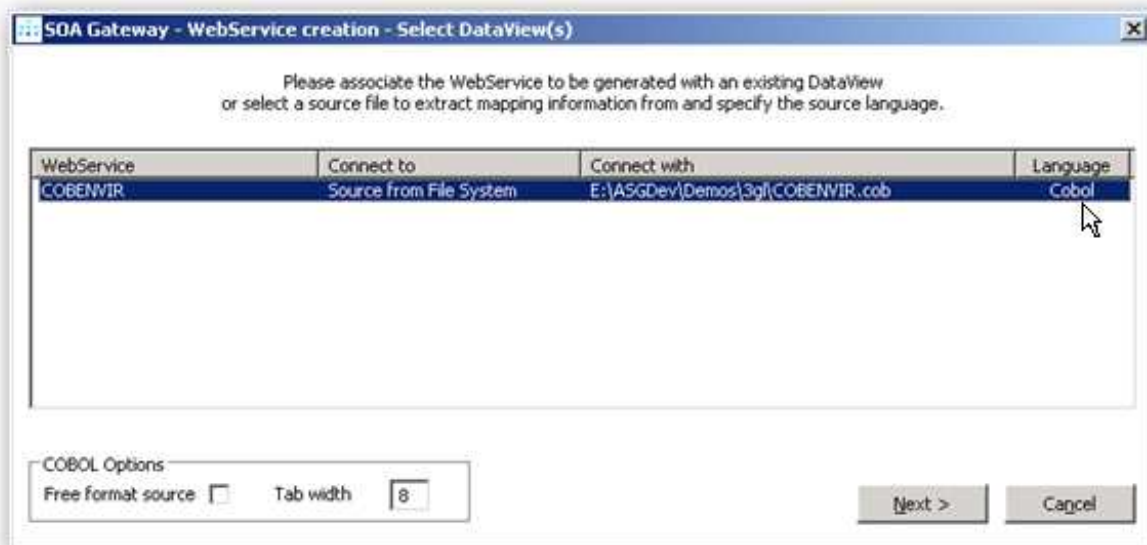
Advance to the *Connect with* column, click it, then click the push-button appearing at the right hand corner



7. In the File Dialog popping up, navigate to the location where the COBOL-source for the demo program COBENVIR has been downloaded to, select it



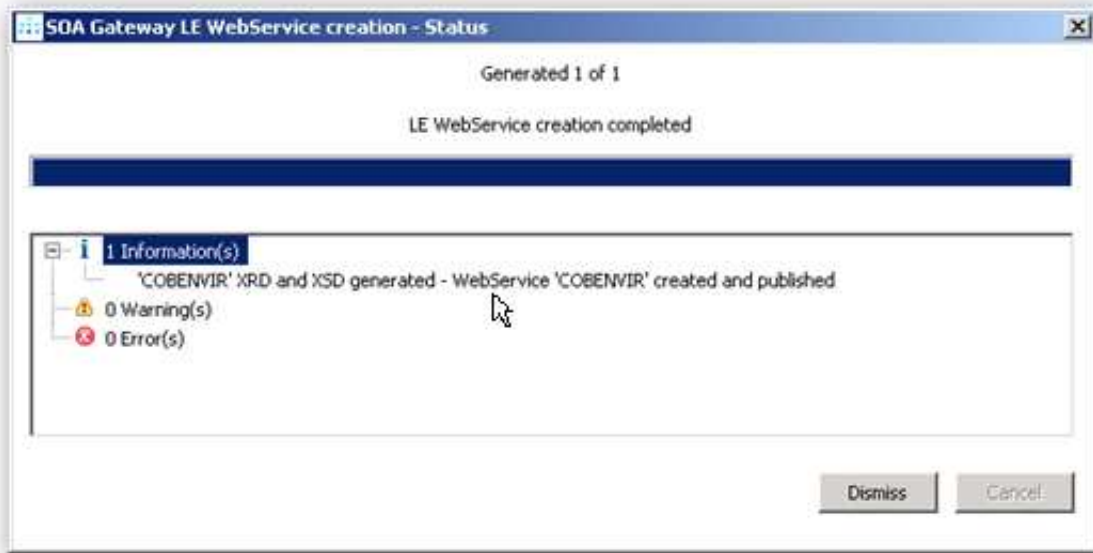
For files with an extension of *.cob* the *Language* will automatically be set to *Cobol*. Click the **Next** button.



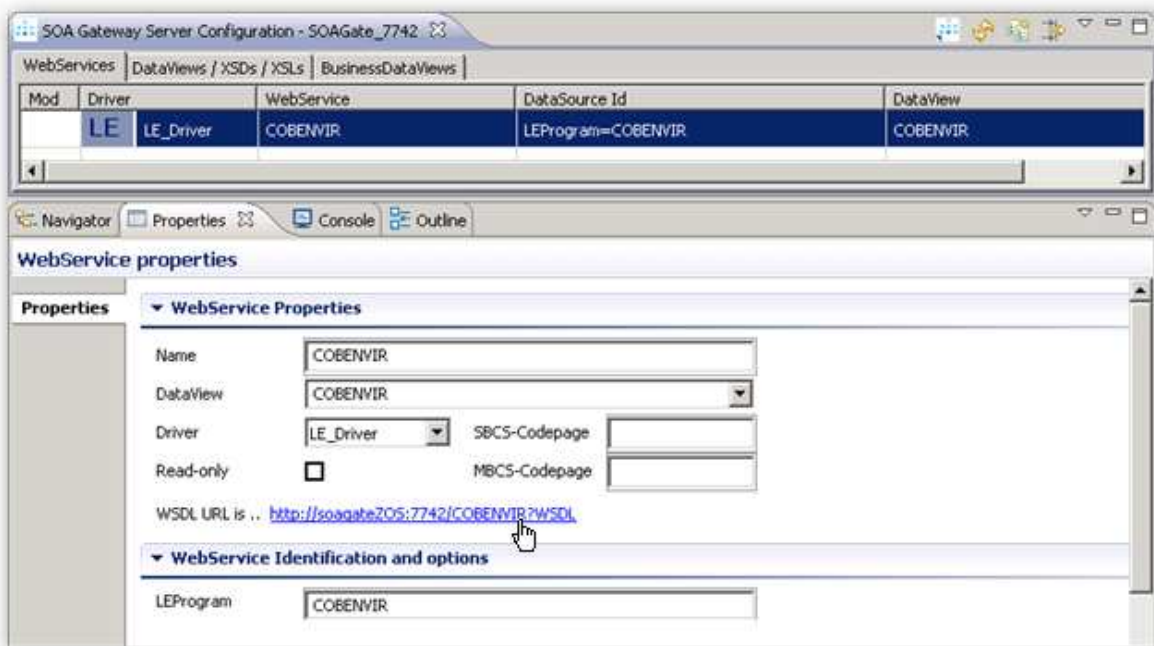
Note:

For COBOL sources not conforming to the standard COBOL source layout check the "Free format source" option to still have them parsed and the DataView generated correctly.

Progress and completion of the generation process will be shown with this dialog



8. The WebService has been created and is usable



9. Click the URL next to "WSDL URL is .." to display the WSDL exposed by the generated WebService:

```

- <schema targetNamespace="http://www.nisaris.com/namespaces/COBENVIR">
- <xs:element name="invokeInputElement">
- <xs:complexType>
- <xs:sequence>
- <xs:element name="COBENVIRRoot">
- <xs:complexType>
- <xs:sequence>
- <xs:element name="COBENVIRGroup">
- <xs:complexType>
- <xs:sequence>
- <xs:element name="COBOL_ENV" type="xs:string"/>
- <xs:element name="OS_ENV" type="xs:string"/>
- <xs:element name="COBOL_DATE" type="xs:string"/>
- </xs:sequence>
- </xs:complexType>
- </xs:element>
- </xs:sequence>
- </xs:complexType>
- </xs:element>
- </xs:sequence>
- </xs:complexType>
- </xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

```

- Overtyping the keyword `WSDL` with `INVOKE&COBOL_ENV=&OS_ENV=&COBOL_DATE=` and execute the request

```

- <COBENVIRRoot>
- <COBENVIRGroup>
- <COBOL_ENV/>
- <OS_ENV/>
- <COBOL_DATE>2009/06/21 10:52:02.271</COBOL_DATE>
- </COBENVIRGroup>
</COBENVIRRoot>

```

Note:

Although the three fields defined in the DataView are in fact "output only" they have to be specified in the request because the Discovery process generates them as "input / output" because it simply cannot know if input is required. This can be changed with the DataView editor manually, but this is beyond the scope of this description.