LANSA
Integrator
Web Services

A
Step by Step
Soap Agent

LANSA Integrator Web Services – A Step by Step SOAP Agent

Introduction

This document is intended as a step by step guide to build a SOAP Agent using LANSA Integrator.

This is not a training course, and the theory behind each step will not be explored. Each step must be followed in order for the SOAP Agent to work.

This document is intended for LANSA analysts and developers. It is assumed that you are able to create a LANSA process and/or function and understand RDML.

Scenario

This SOAP Agent is intended to act as a client program to an existing HR system that has a web service interface. The client allows a department and section to be entered and then passed as a SOAP request to the HR web service. A list of employees for that department and section will be returned.

Requirements

These requirements may also apply to other server platforms but are written for the iSeries.

iSeries

An iSeries server running OS/400 release V5R1M0 or later and Java Version 1.4.

LANSA for the iSeries V10.0 updated with EPC756 or later.

LANSA for the iSeries V11.0 updated with EPC752 or later.

LANSA Integrator with the latest EPCs applied. See the LANSA web site for details.

Applicable licences for LANSA development and for LANSA Integrator installed on the iSeries.

The HTTP server for the iSeries, or Apache, configured to use JSMDirect and JSMAdmin.

LANSA Integrator configured to enable connection via Integrator Studio.

Windows

A PC with JDK and JRE 1.4 or later.

Integrator Studio with the latest EPCs applied. See the LANSA web site for details.

For help in configuring or confirming these requirements, consult either the LANSA Integrator Guide or the www.lansa.com website. If any issues arise in meeting these requirements please contact your software distributor's help desk.

Attachment

A WSDL file is supplied with these instructions. This file can be used if you have not yet set up, or do not have a SOAP Server available to test the SOAP Agent set up using these instructions.

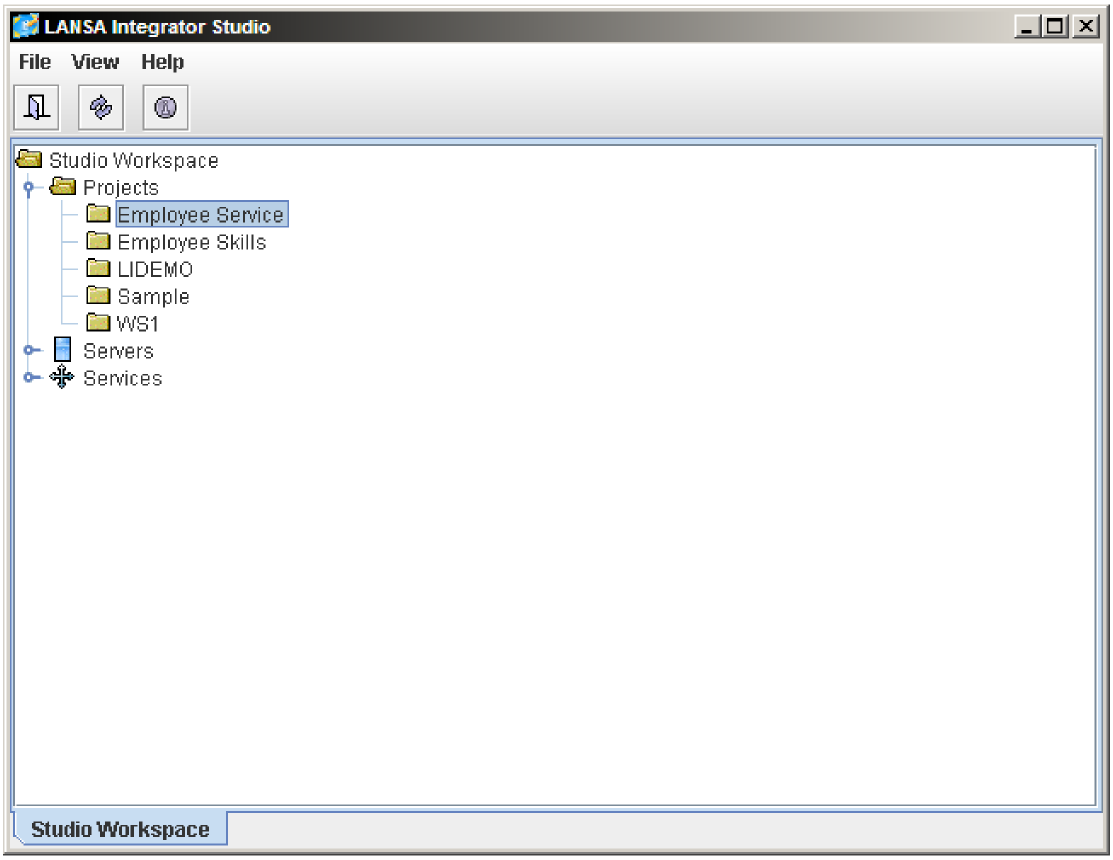
1. When the LANSA Integrator Studio has started you will see three items:
    - Projects
    - Servers and
    - Services.

A Studio **Project** is a collection of files related to a particular task or application using LANSA Integrator.

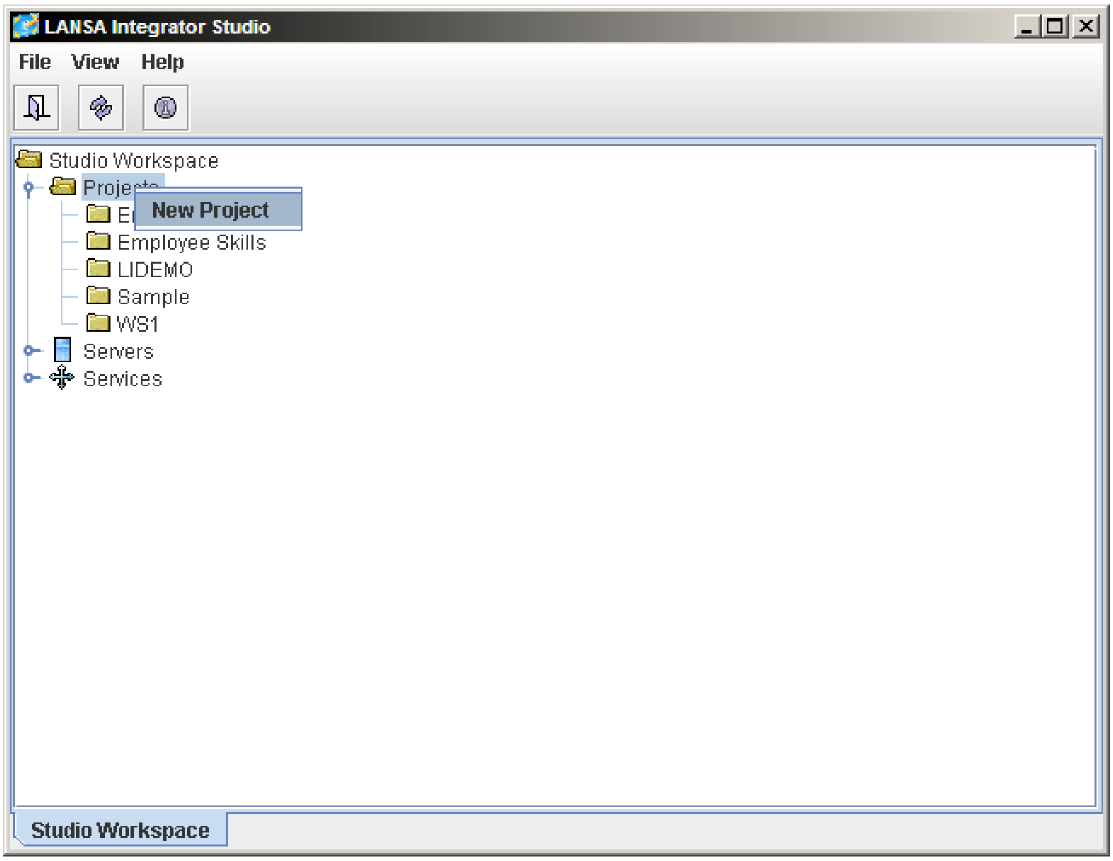
Servers allow you to view the contents of the instance directory of a JSM server.

Services are the defined connection to the database files that support JSMDirect and JSMProxy on a JSM server.

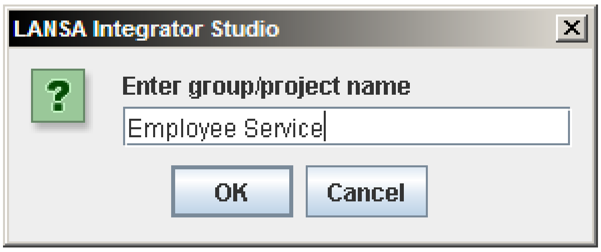
Open the 'Projects' folder then open the 'Employee Service' project and go to step 2.



If the project does not exist then right mouse click on 'Projects' and select 'New Project' from the pop-up menu.

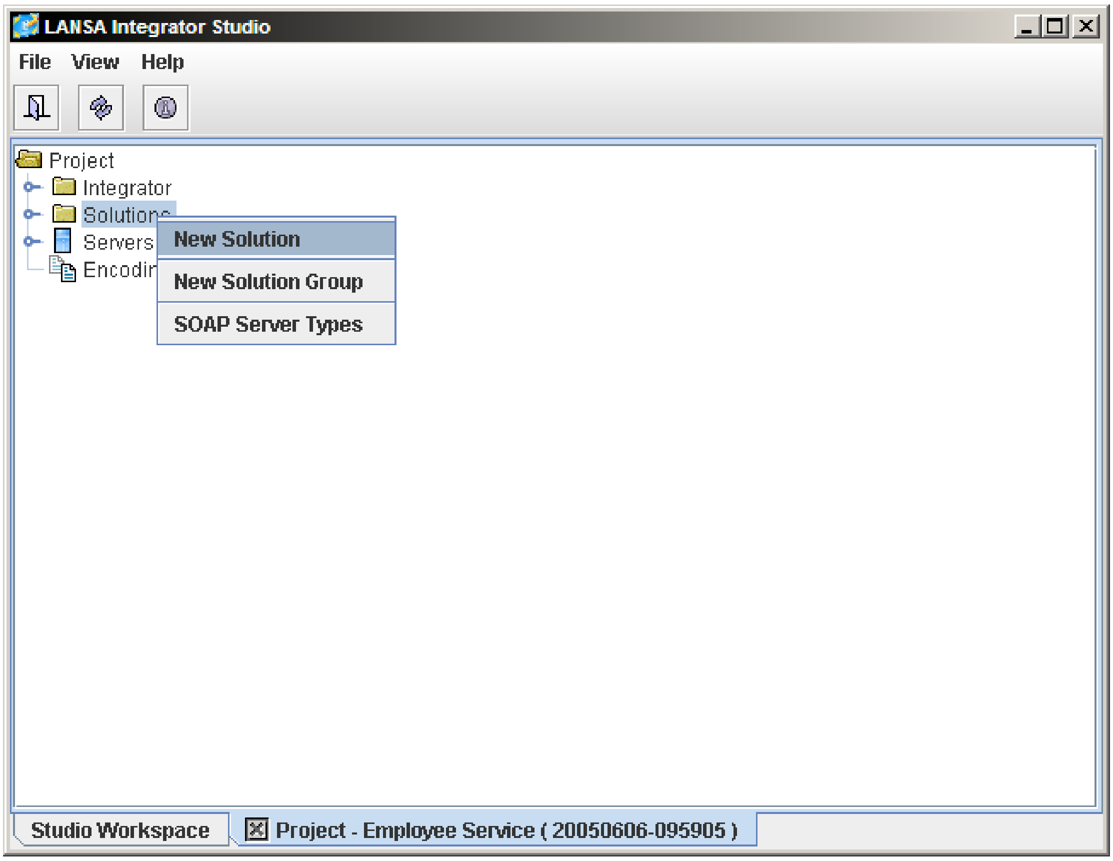


In the dialog box enter the name of the new project as 'Employee Service' and click 'OK' to continue.



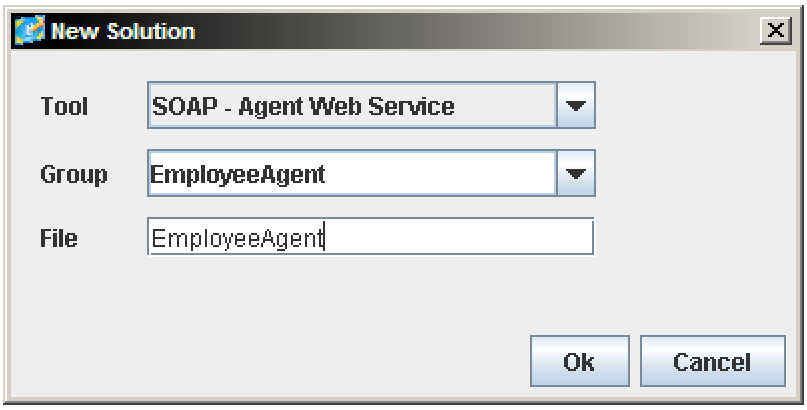
Enter group/project name: Employee Service

2. Once the ‘Employee Service’ project workspace has opened, right mouse click on ‘Solutions’ and select ‘New Solution’ from the pop-up menu.



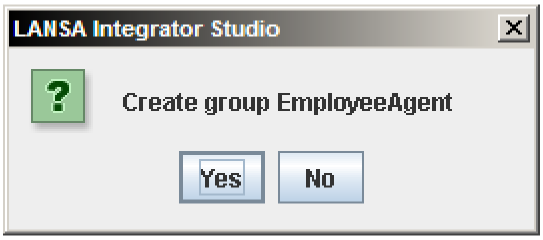
In the new solution dialog box select ‘SOAP – Agent Web Service’ from the drop down list, enter the group and filename as ‘EmployeeAgent’. It should be noted that the File name entered will be used by the SOAP Wizard as a naming prefix for creating objects, so enter a descriptive name for the agent.

Next click ‘Ok’.



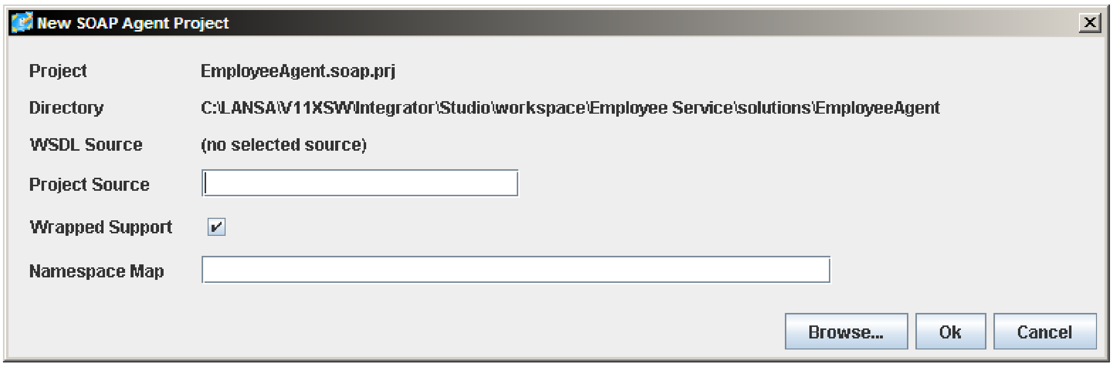
Tool: SOAP - Agent Web Service
Group: EmployeeAgent
File: EmployeeAgent

Since the Solutions group doesn’t exist, the dialog below will appear to confirm the creation of the group. Click ‘Yes’.



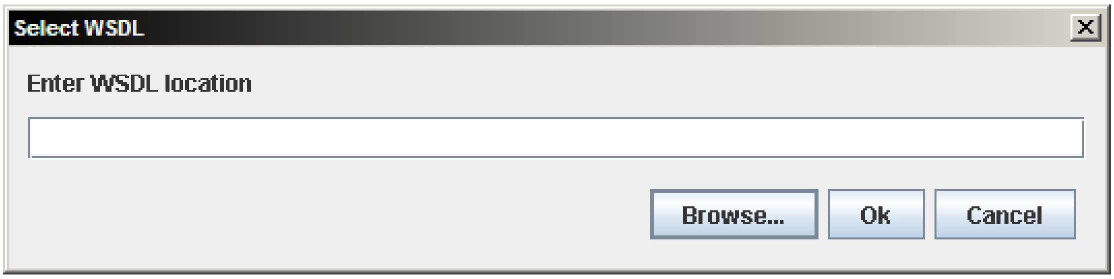
Create group EmployeeAgent

3. A SOAP Agent is always based on the WSDL of the web service it will be interacting with. The dialog below is used to select the WSDL for this SOAP agent. To do this first click ‘Browse’ to locate it.



Project EmployeeAgent.soap.prj
Directory C:\LANSA\V11XSW\Integrator\Studio\workspace\Employee Service\solutions\EmployeeAgent
WSDL Source (no selected source)
Project Source
Wrapped Support ☑
Namespace Map

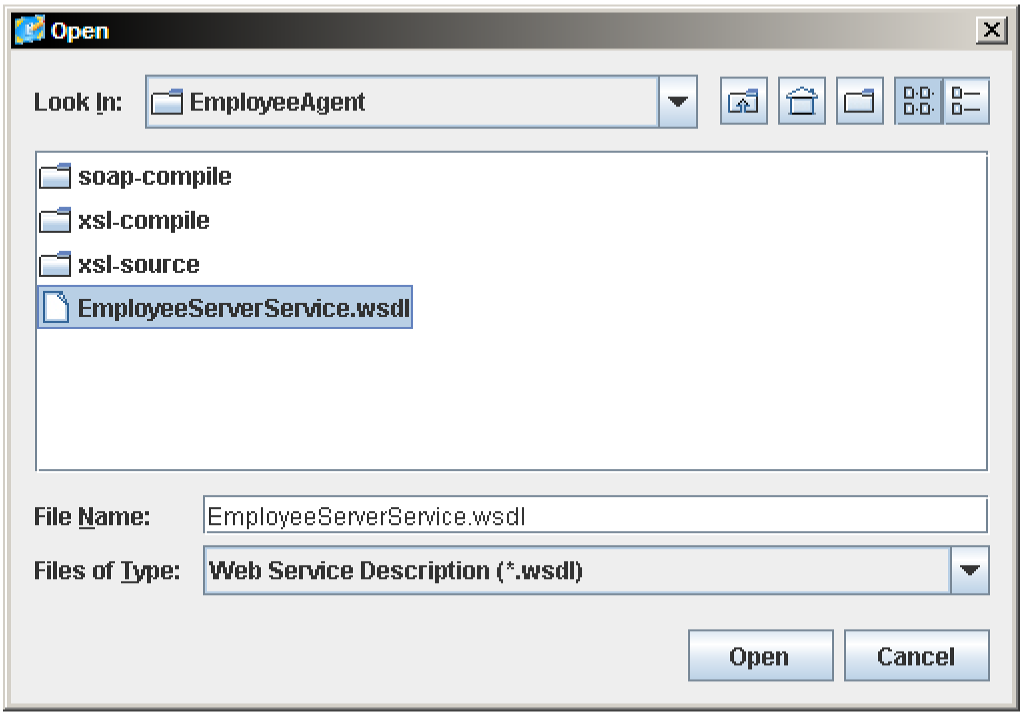
This dialog is where the location of the WSDL is entered. If the WSDL is stored on a remote server, the URL can be typed in. The WSDL for this example is stored locally, click ‘Browse’ to find the WSDL.



Enter WSDL location

Press 'OK' to continue.

Open the folder in which the 'EmployeeServerService.wsdl' file is stored, select the file by clicking on it and then click 'Open'

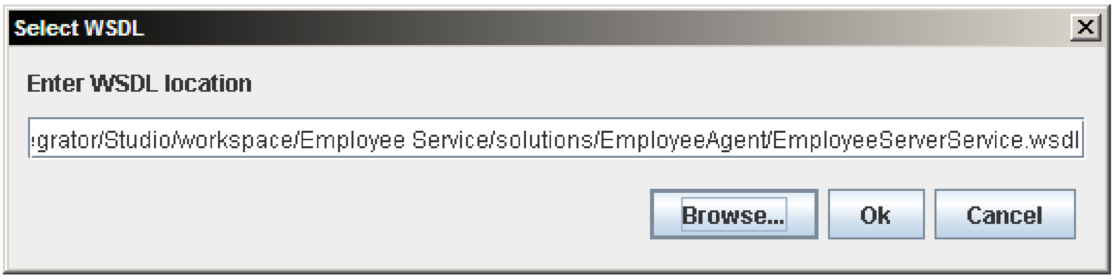


Look In: EmployeeAgent

File Name: EmployeeServerService.wsdl

Files of Type: Web Service Description (*.wsdl)

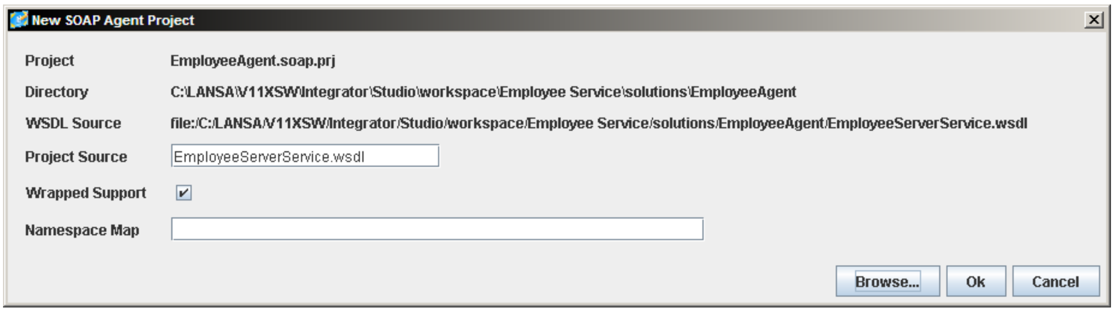
The selected WSDL file is redisplayed with the full path. Click 'Ok' to confirm the selection.



Enter WSDL location

grator/Studio/workspace/Employee Service/solutions/EmployeeAgent/EmployeeServerService.wsdl

The Name dialog is then redisplayed, to continue click 'Ok'.



Project EmployeeAgent.soap.prj

Directory C:\LANSA\V11XSW\Integrator\Studio\workspace\Employee Service\solutions\EmployeeAgent

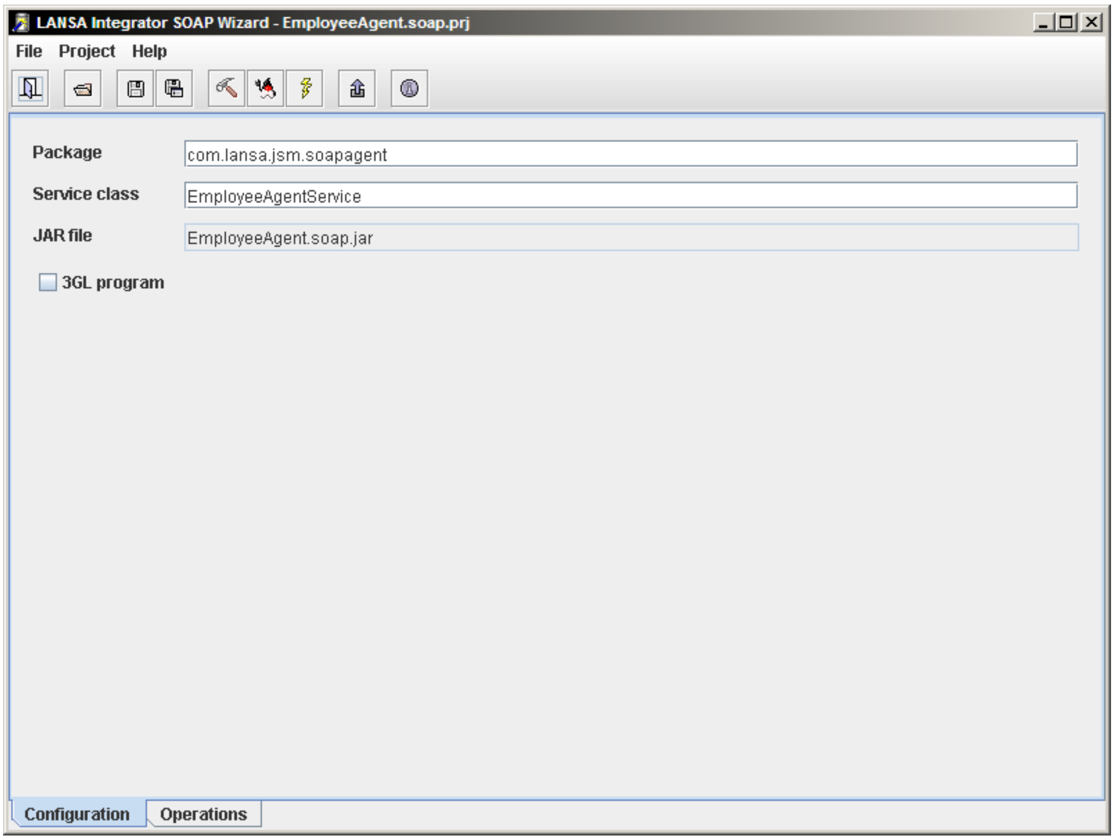
WSDL Source file:/C:/LANSA/V11XSW/Integrator/Studio/workspace/Employee Service/solutions/EmployeeAgent/EmployeeServerService.wsdl

Project Source EmployeeServerService.wsdl

Wrapped Support ☑

Namespace Map

4. The SOAP Wizard is then opened for the new SOAP agent. The configuration tab is displayed with default values.



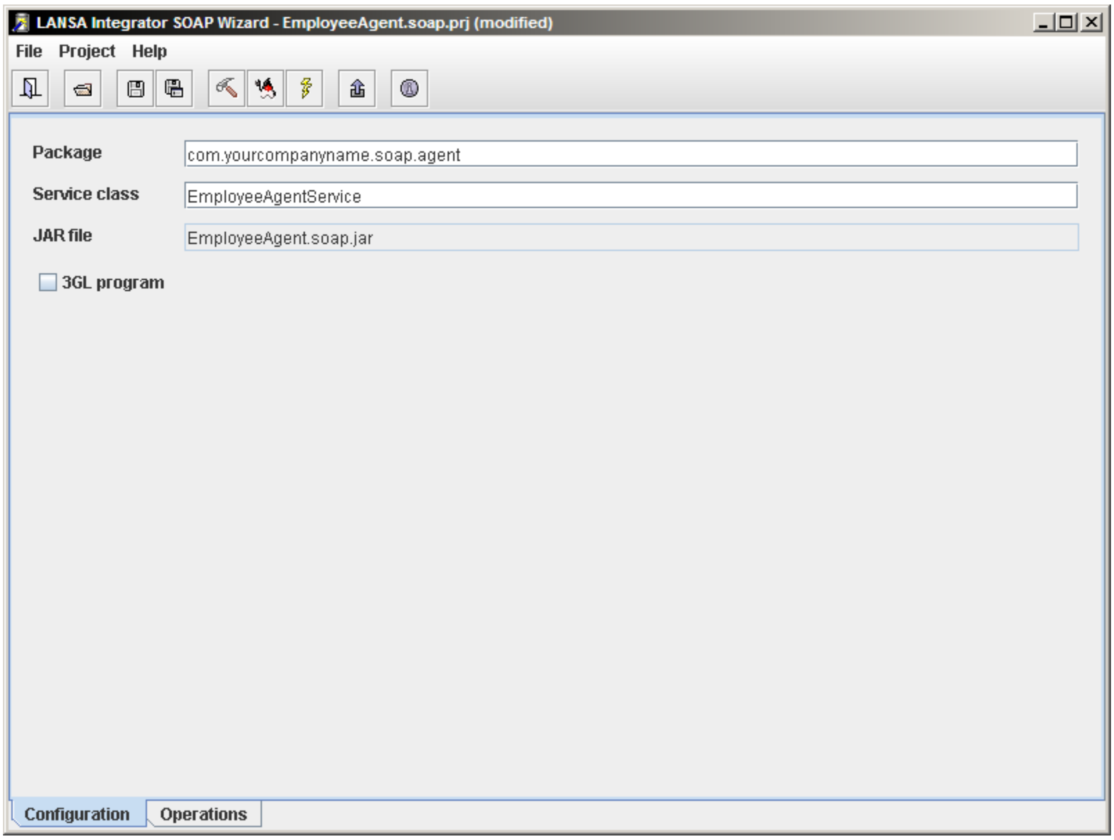
Package: com.lansa.jsm.soapagent

Service class: EmployeeAgentService

JAR file: EmployeeAgent.soap.jar

☐ 3GL program

The default values of the package need to be changed. Change the values below with information relating to your situation. Then select the ‘Operations’ tab at the bottom of the SOAP wizard.


LANSA Integrator SOAP Wizard - EmployeeAgent.soap.prj (modified)
File
Project
Help
Configuration
Operations

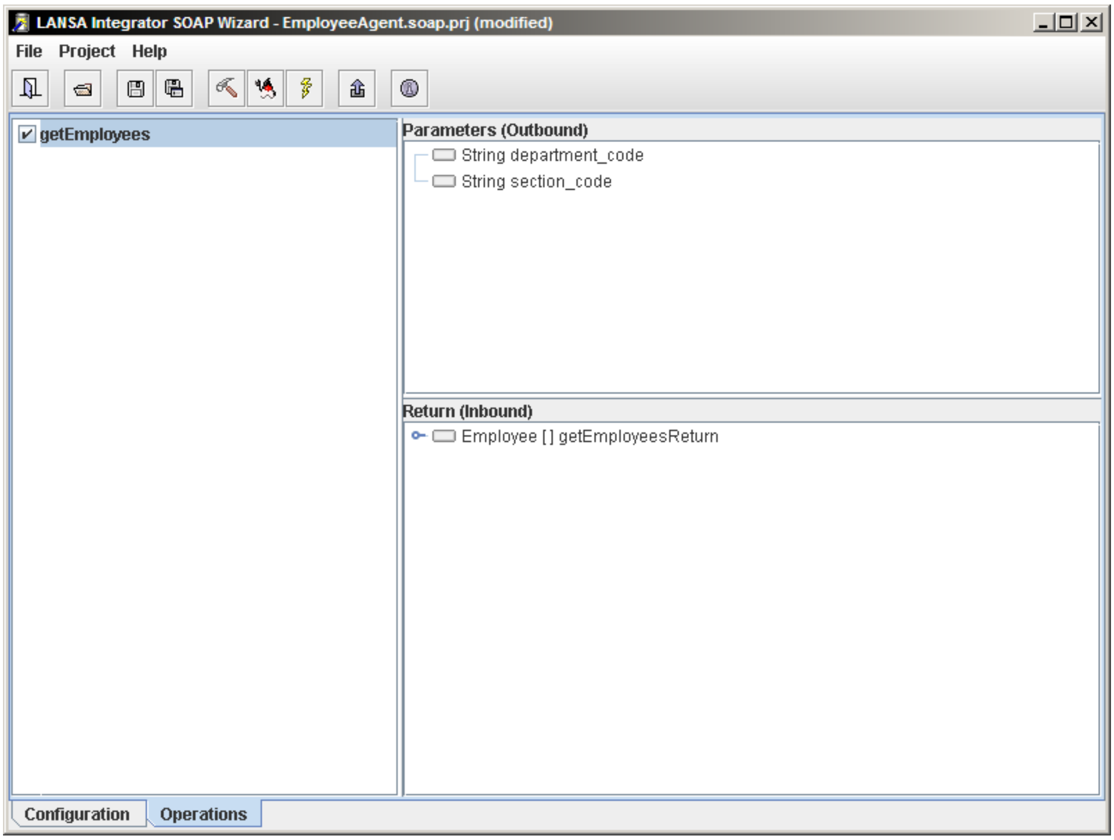
Package: com.yourcompanyname.soap.agent

Service class: EmployeeAgentService

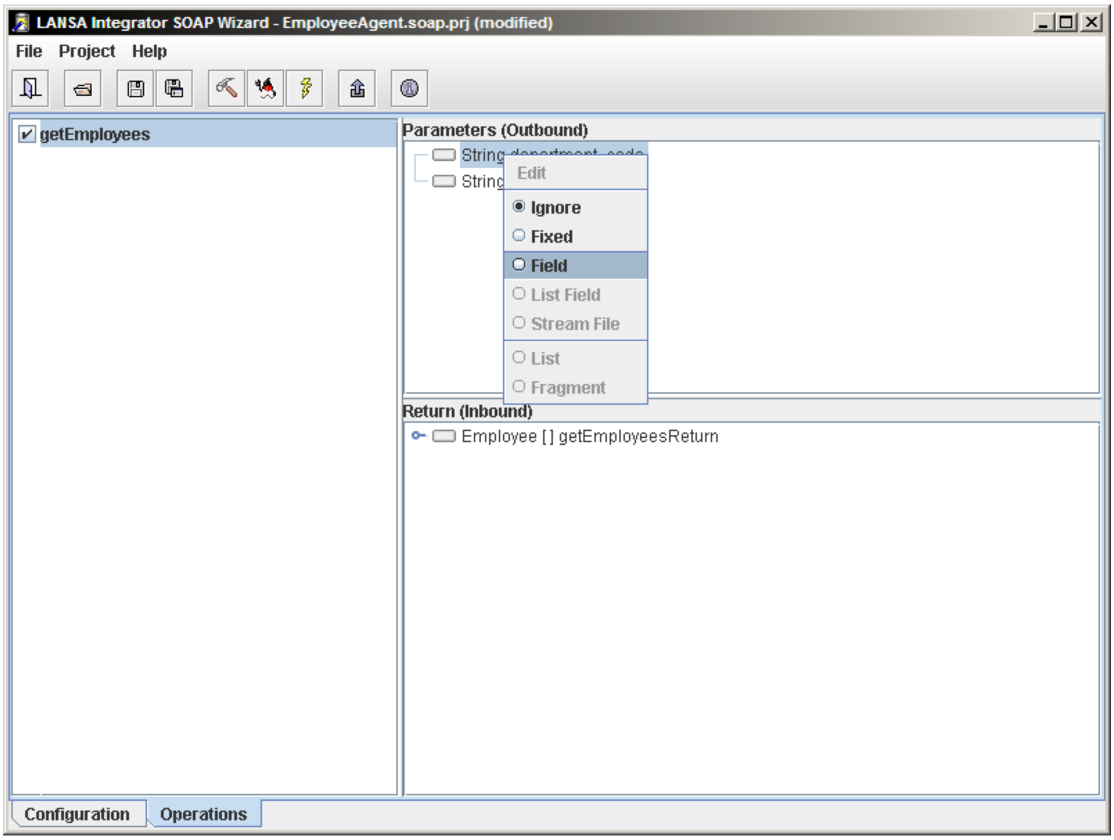
JAR file: EmployeeAgent.soap.jar

☐ 3GL program

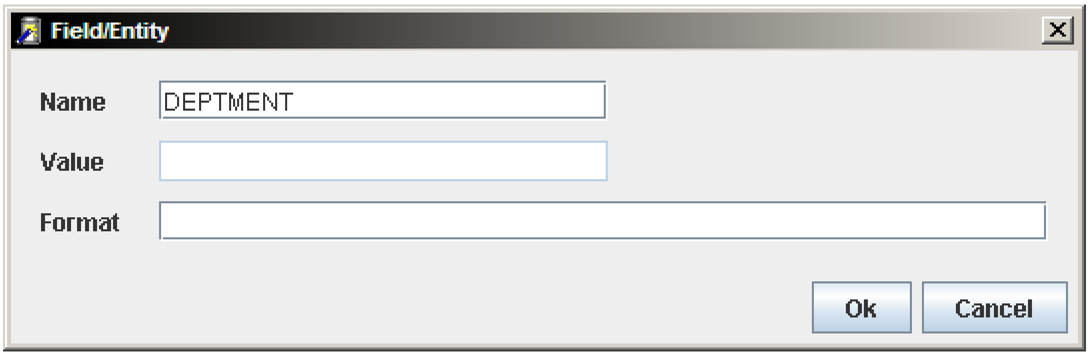
5. The SOAP Agent starts by displaying the operations in the left hand pane the agent will support. A SOAP operation refers to published tasks the web service is able to perform, similar to a method in a Visual LANSA form or a subroutine in a LANSA function. Click the check box next to ‘getEmployees’ to select the operation.



6. Soap operations have inbound and outbound parameters. These parameters can be optionally marked for use and then have the origin or destination of their value determined, for example a LANSA field. The initial icons for the outbound and inbound parameters indicate they are currently not marked for use. To mark the outbound parameters as having their values come from a field, right mouse click on 'String department_code' and select 'Field' from the pop-up menu.

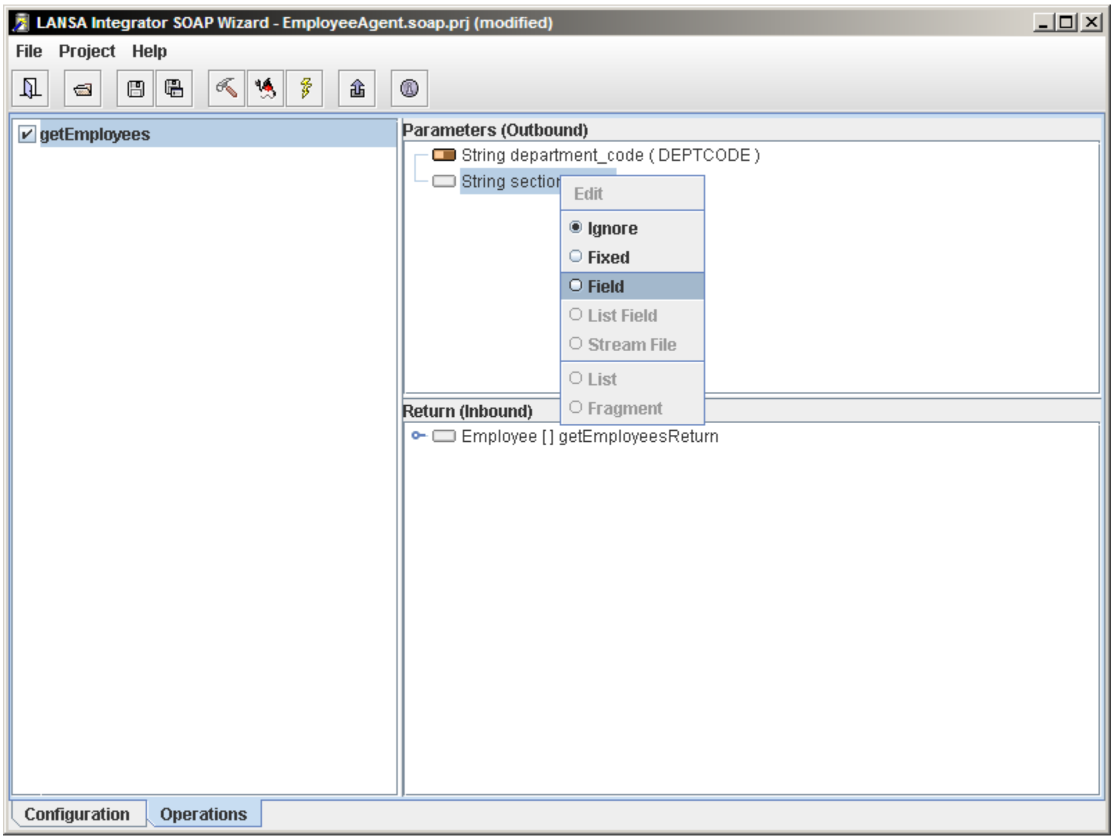


The field 'DEPTMENT' is entered in the Field dialog box as the origin of the parameter's value. Click 'OK' to continue.

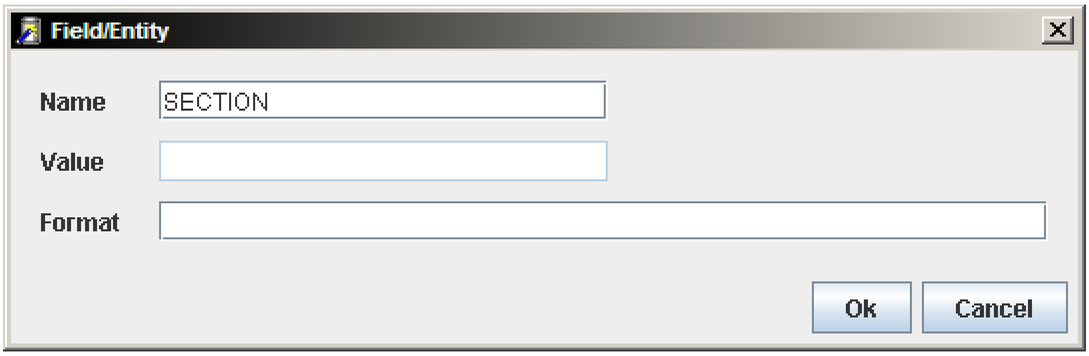


Name: DEPTMENT
Value:
Format:

Note that the icon for the parameter has now changed. This icon indicates the parameter is marked for use and is mapped to a field. The name of the field is in brackets at the end of the parameter. Right mouse click on the second outbound parameter ‘String section_code’ and select ‘Field’ from the pop-up menu.

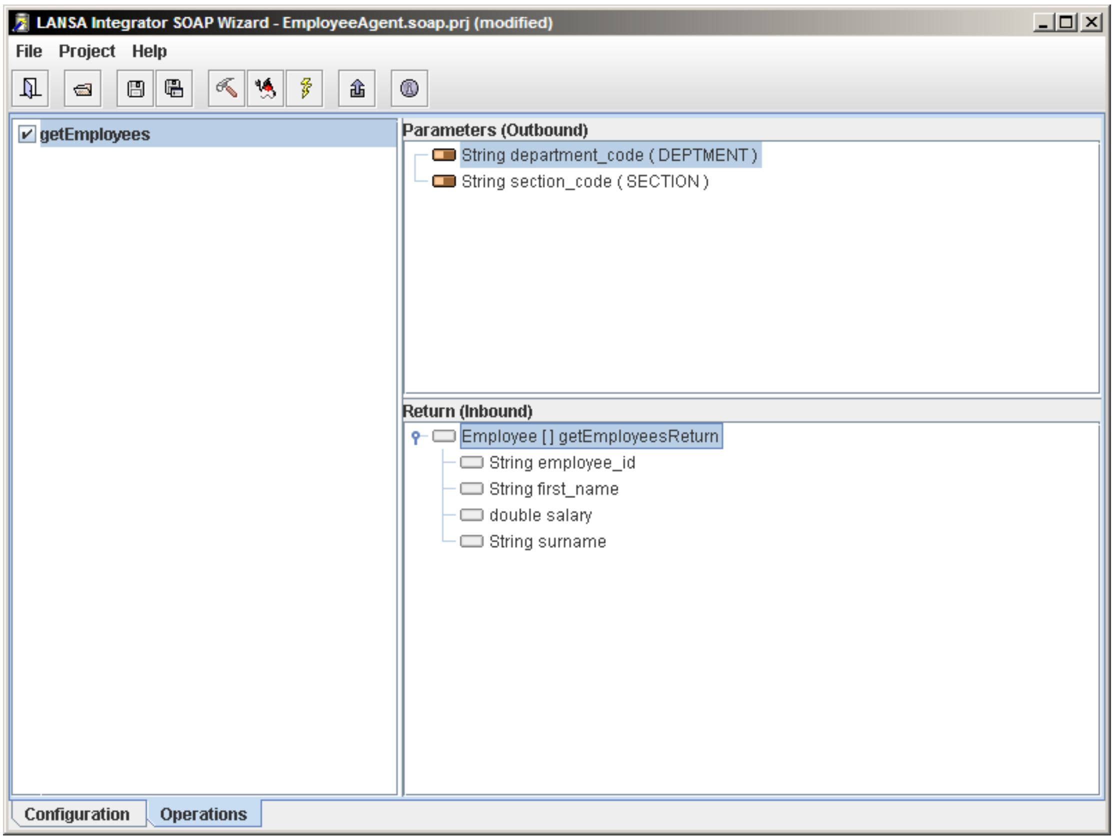


Enter ‘SECTION’ as the field that will hold the value of the parameter. Click ‘Ok’.



Name: SECTION
Value:
Format:

7. The return or inbound parameter will hold the values returned by the web service. The parameter can be expanded to see that it is made up of several elements.

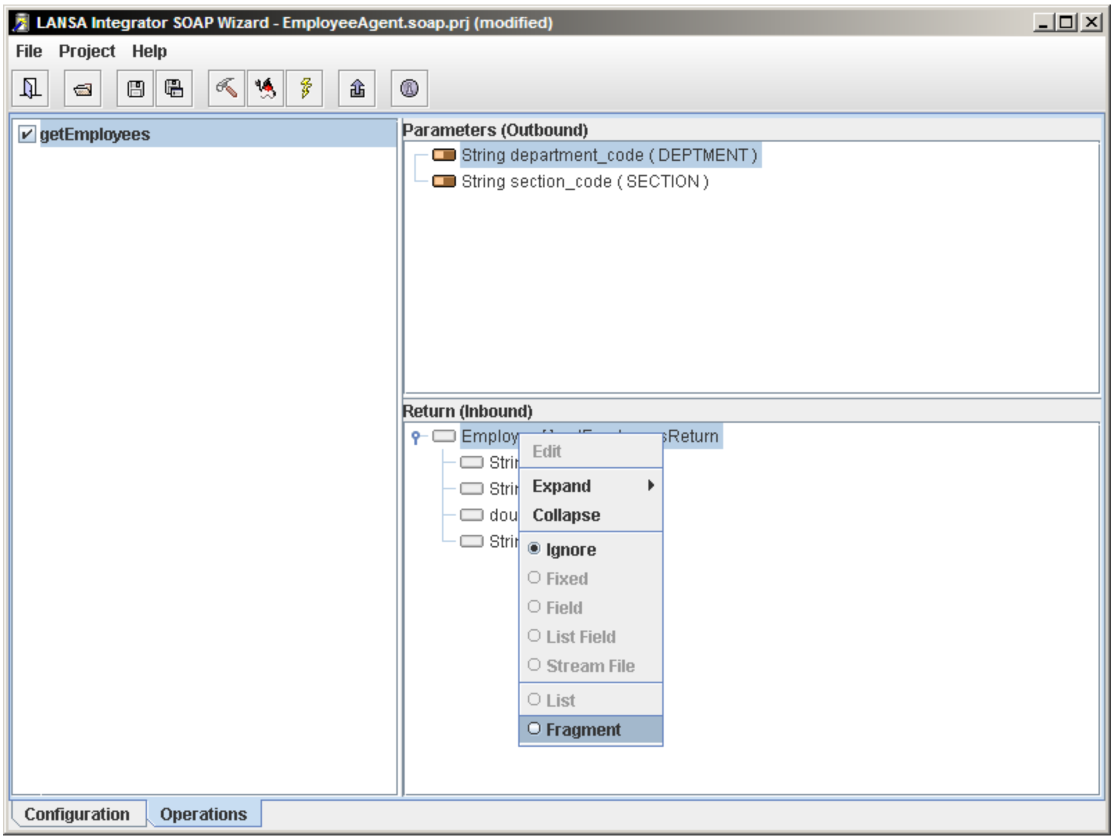


String department_code (DEPTMENT)
String section_code (SECTION)

Employee [] getEmployeesReturn
String employee_id
String first_name
double salary
String surname

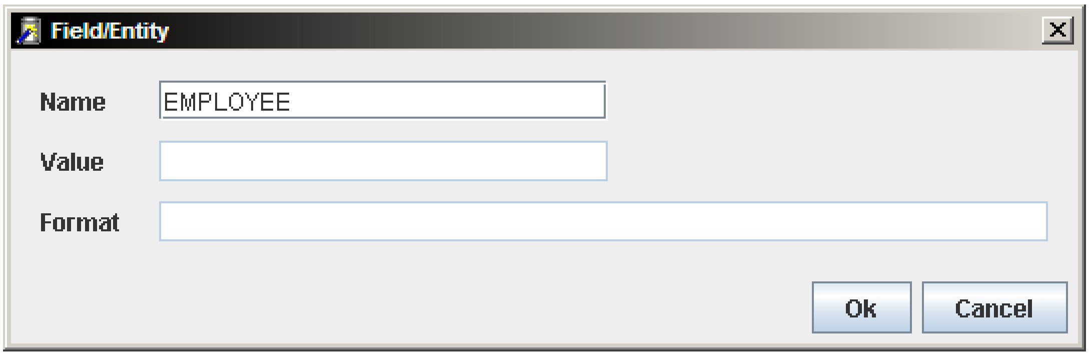
To map these values into fields that the LANSA function will use, right mouse click on ‘Employee[] getEmployeeReturn’. Note that only valid mapping options will be enabled by the SOAP wizard.

Select ‘Fragment’ from the pop-up menu. Marking a parameter or element as a Fragment means that this section can be handle separately from the other parameters or elements. This enables complex structures to be handled by the SOAP agent.



String department_code (DEPTMENT)
String section_code (SECTION)

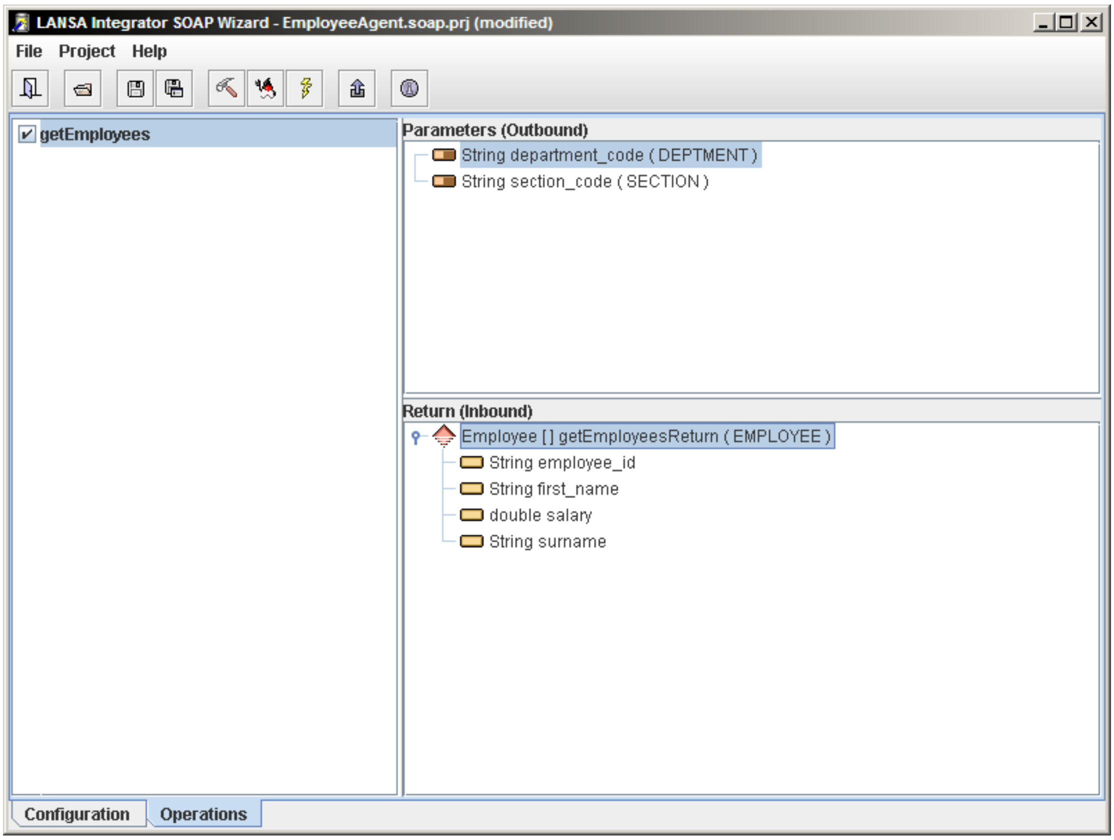
Enter the name of the Fragment as ‘EMPLOYEE’ and click ‘Ok’.



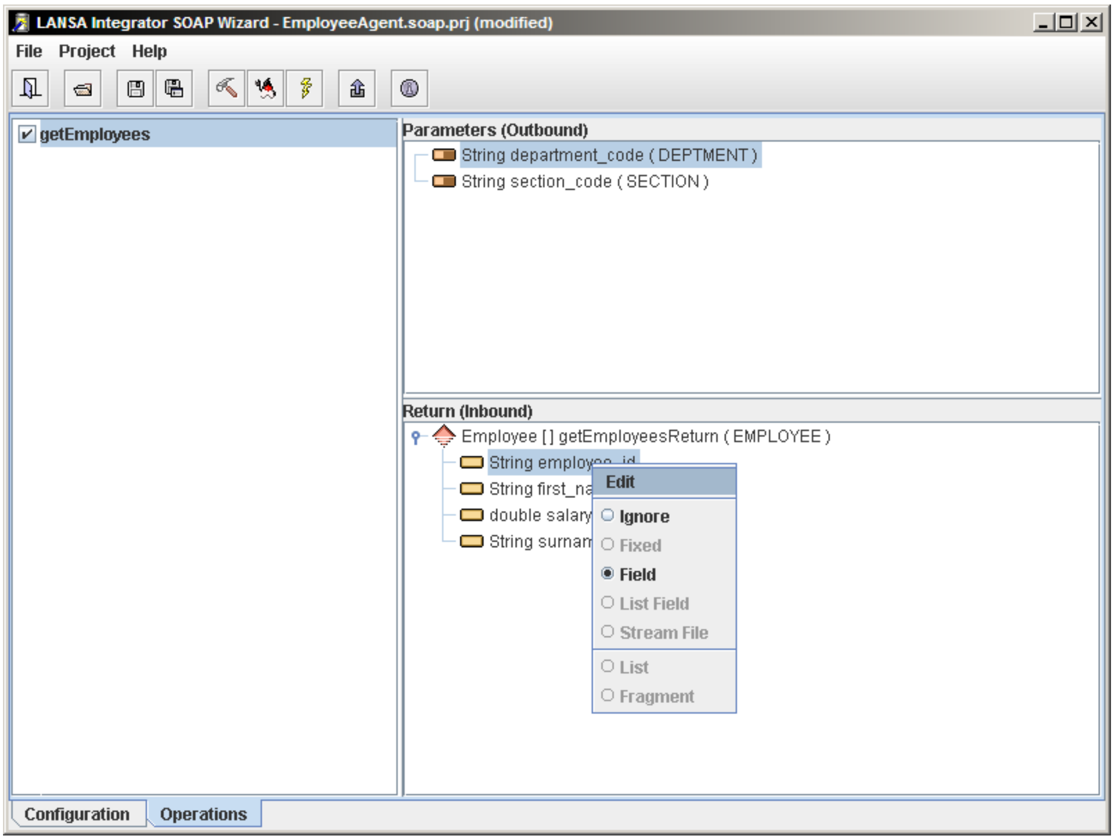
Name EMPLOYEE

Note the icons for the return parameter have changed. The icon for 'Employee [] getEmployeeReturn' indicates this has been marked as a Fragment. The name of the Fragment is at the end of the parameter in brackets.

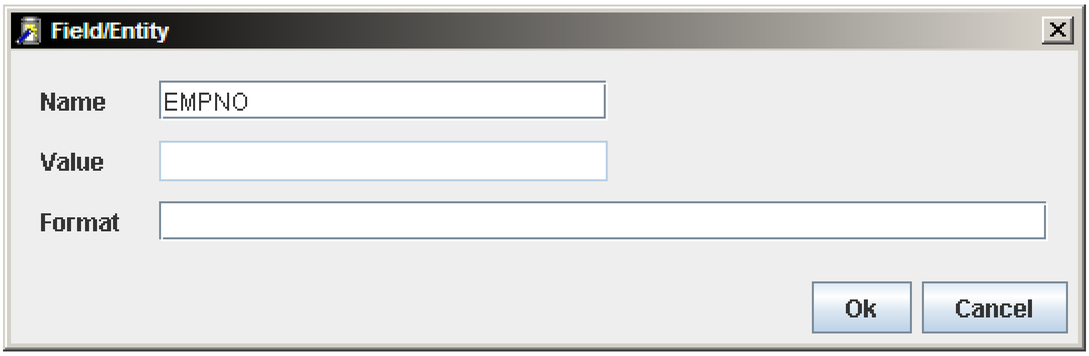
The icon of the elements indicates these have been marked as fields. This has been done by the SOAP Wizard because this Fragment is logically made of fields.


LANSA Integrator SOAP Wizard - EmployeeAgent.soap.prj (modified)
File Project Help
☑ getEmployees
Parameters (Outbound)
String department_code (DEPTMENT)
String section_code (SECTION)
Return (Inbound)
Employee [] getEmployeesReturn (EMPLOYEE)
String employee_id
String first_name
double salary
String surname
Configuration
Operations

To specify which field the value of the first element gets passed to, right mouse click on ‘String employee_id’ and select ‘Edit’ from the pop-up menu.



Enter the name of the field as “EMPNO” and click ‘Ok’ to continue.

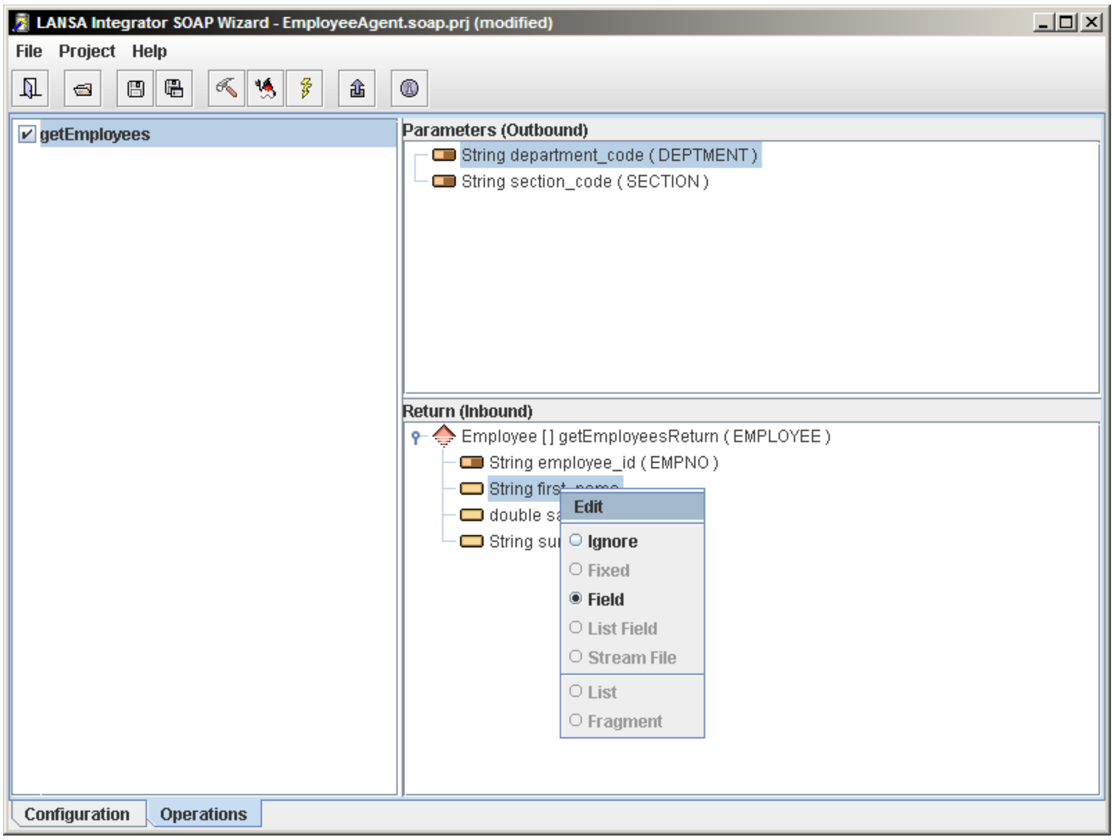


Name: EMPNO

Value:

Format:

Right mouse click on ‘String first_name’ and select ‘Edit’ from the pop-up menu.



Parameters (Outbound)

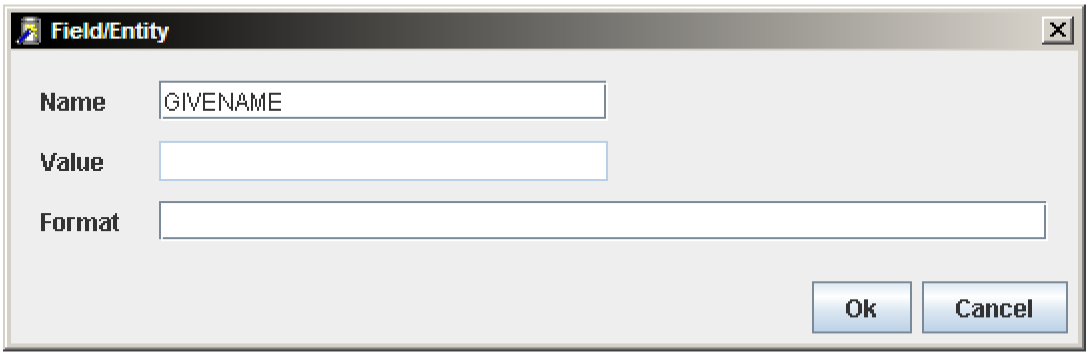
- String department_code (DEPTMENT)
- String section_code (SECTION)

Return (Inbound)

- Employee [] getEmployeesReturn (EMPLOYEE)
  - String employee_id (EMPNO)
  - String first_name
  - double s
  - String sur

● Field

Enter ‘GIVENAME’ as the name of the field for this element and click ‘Ok’.

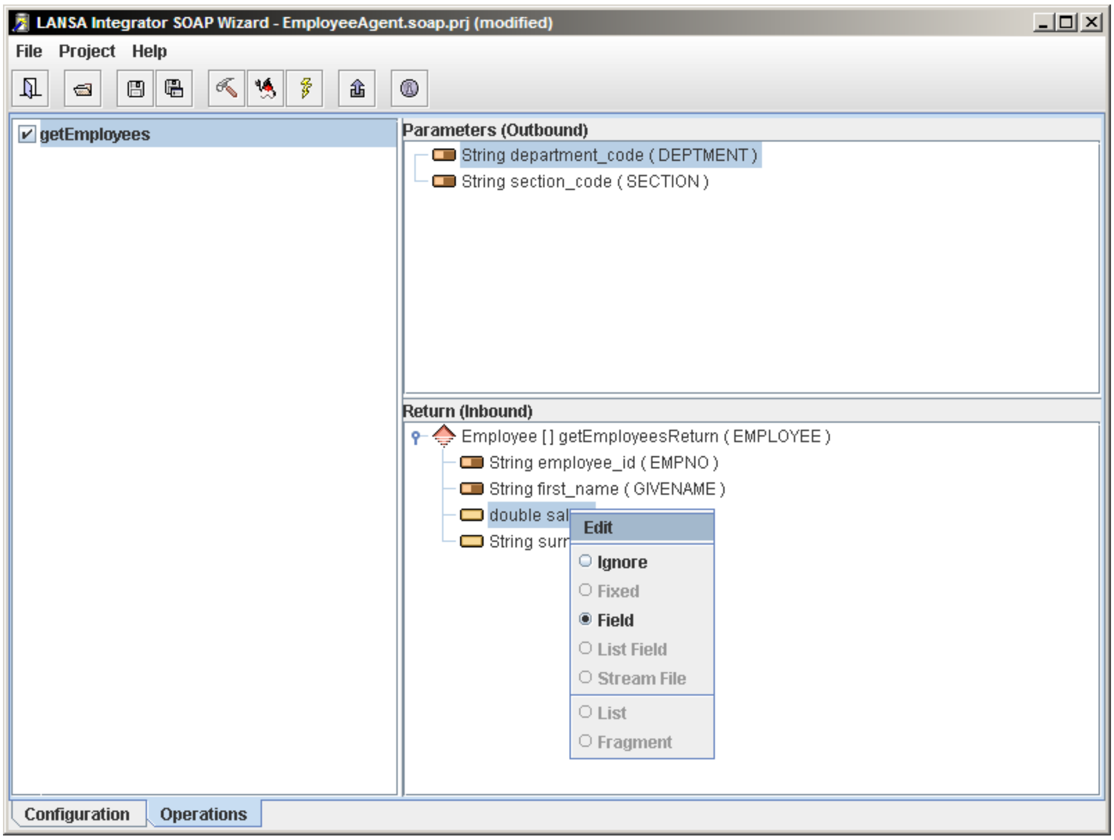


Name: GIVENAME

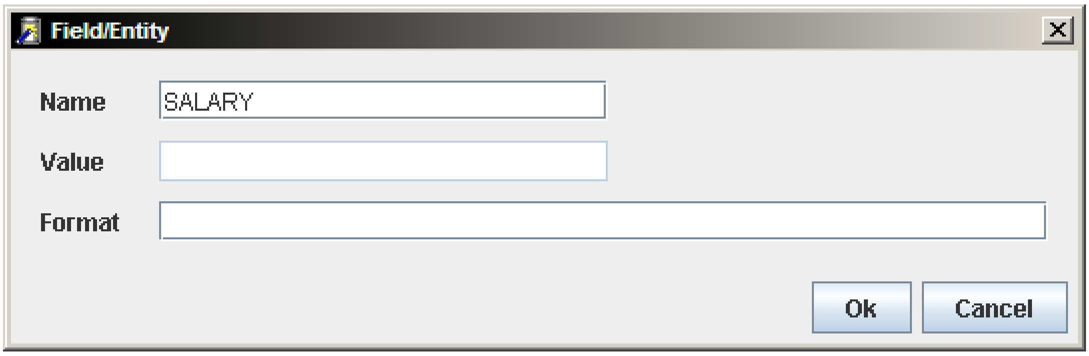
Value:

Format:

Right mouse click on ‘double salary’ and select ‘Edit’ from the pop-up menu.


LANSA Integrator SOAP Wizard - EmployeeAgent.soap.prj (modified)
File
Project
Help
☑ getEmployees
Parameters (Outbound)
String department_code (DEPTMENT)
String section_code (SECTION)
Return (Inbound)
Employee [] getEmployeesReturn (EMPLOYEE)
String employee_id (EMPNO)
String first_name (GIVENAME)
double sal
String surr
Edit
○ Ignore
○ Fixed
● Field
○ List Field
○ Stream File
○ List
○ Fragment
Configuration
Operations

Enter ‘SALARY’ as the field for this element and click ‘Ok’.

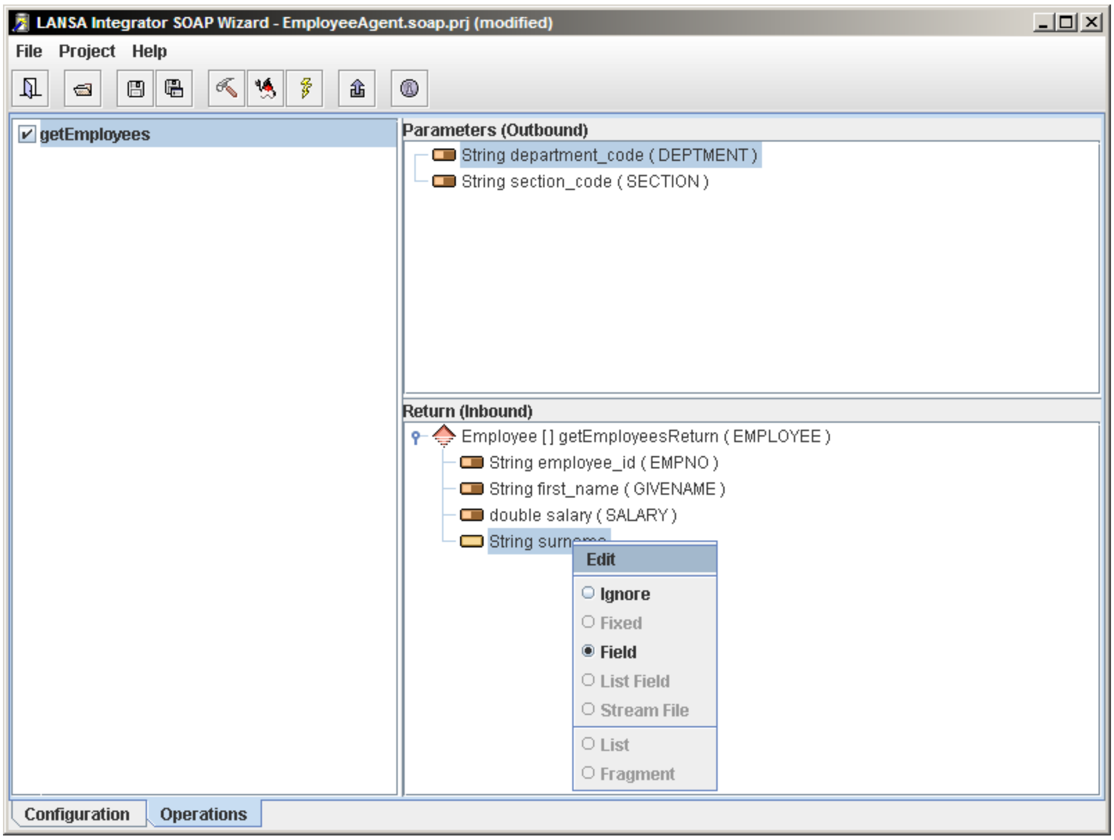


Name: SALARY

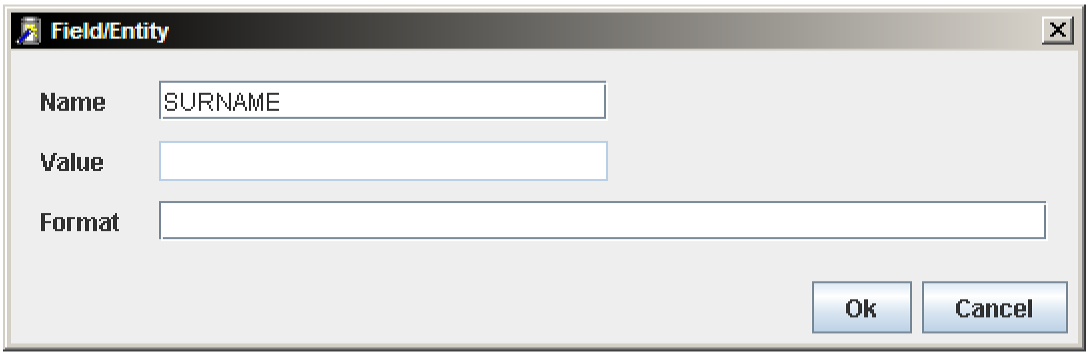
Value:

Format:

Right mouse click on ‘String surname’ and select ‘Edit’ from the pop-up menu.


LANSA Integrator SOAP Wizard - EmployeeAgent.soap.prj (modified)
File
Project
Help
☑ getEmployees
Parameters (Outbound)
String department_code (DEPTMENT)
String section_code (SECTION)
Return (Inbound)
Employee [] getEmployeesReturn (EMPLOYEE)
String employee_id (EMPNO)
String first_name (GIVENAME)
double salary (SALARY)
String surname
Edit
○ Ignore
○ Fixed
● Field
○ List Field
○ Stream File
○ List
○ Fragment
Configuration
Operations

Enter ‘SURNAME’ as the field for this element and click ‘Ok’.

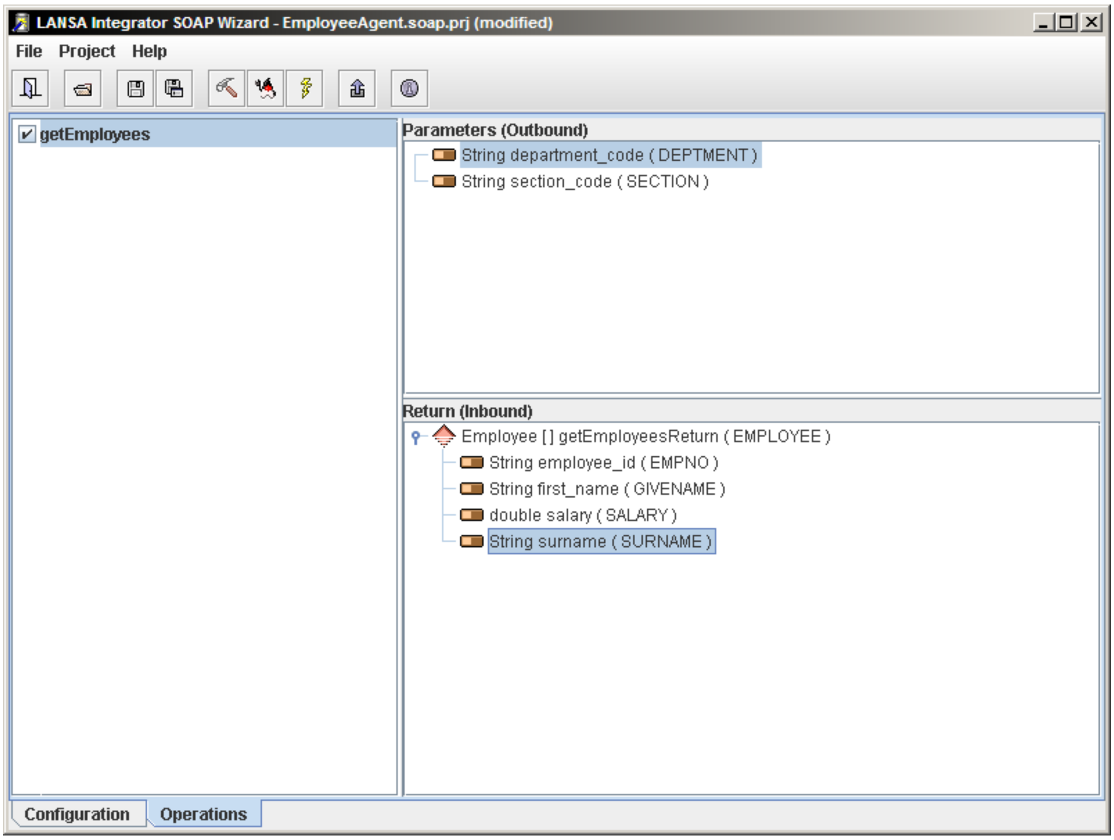


Name SURNAME

Value

Format

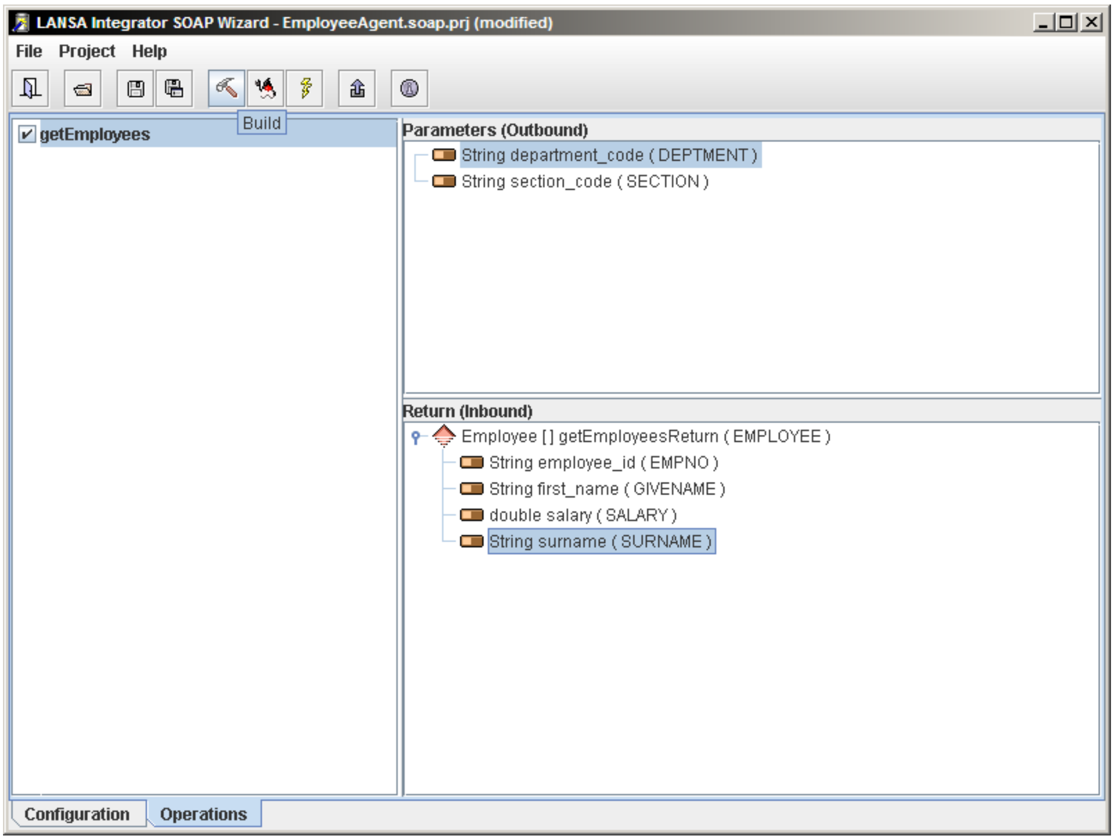
Note the icon for the elements has now changed to indicate the value will be mapped to a field and the name of the field is in brackets at the end of the element.



String department_code (DEPTMENT)
String section_code (SECTION)

Employee [] getEmployeesReturn (EMPLOYEE)
String employee_id (EMPNO)
String first_name (GIVENAME)
double salary (SALARY)
String surname (SURNAME)

To generate the jar file required to send and receive the SOAP data click build (the hammer button) on the toolbar.



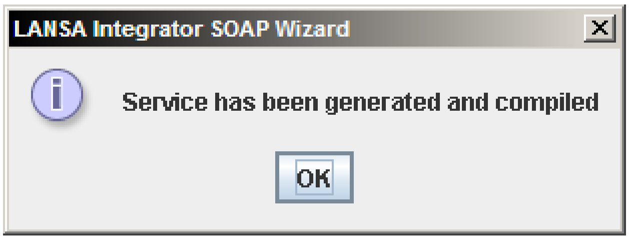
Parameters (Outbound)

- String department_code (DEPTMENT)
- String section_code (SECTION)

Return (Inbound)

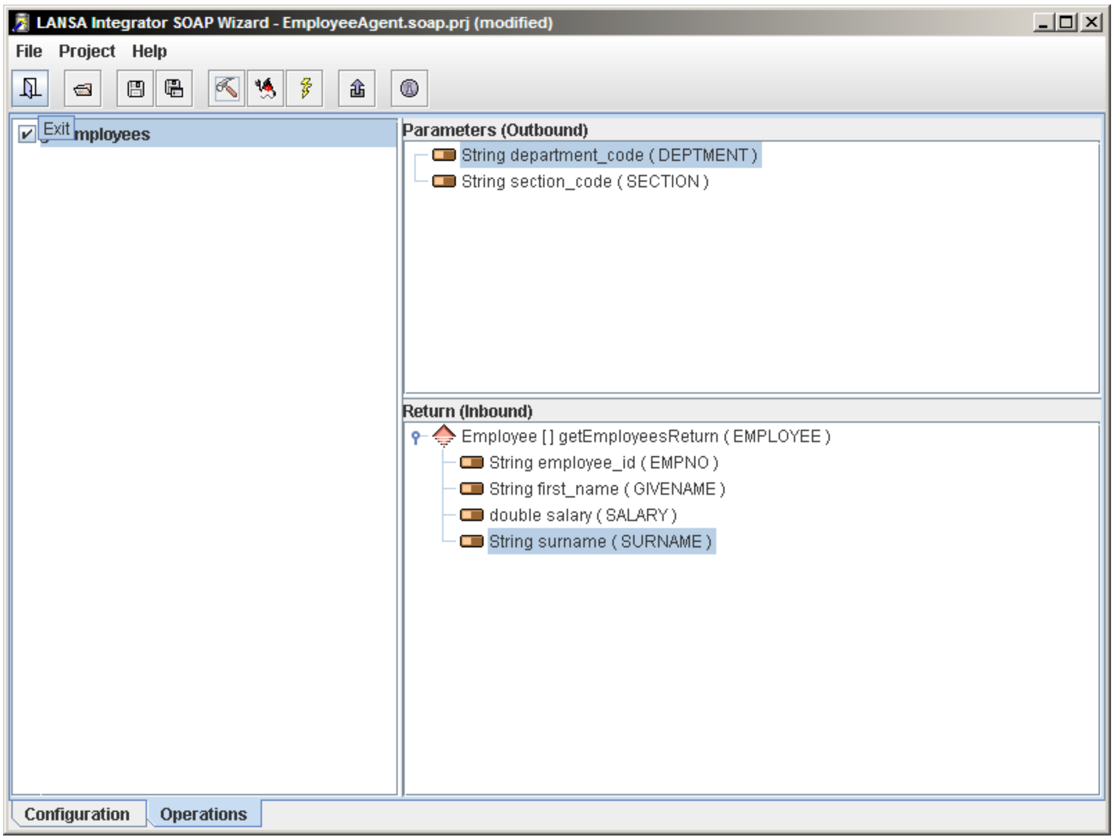
- Employee [] getEmployeesReturn (EMPLOYEE)
  - String employee_id (EMPNO)
  - String first_name (GIVENAME)
  - double salary (SALARY)
  - String surname (SURNAME)

The dialog below will appear once the generation has completed successfully. Click ‘OK’ to continue. If you have any errors, first check to see if you have made any mistakes. If not, then contact your software distributor's support desk for further help.

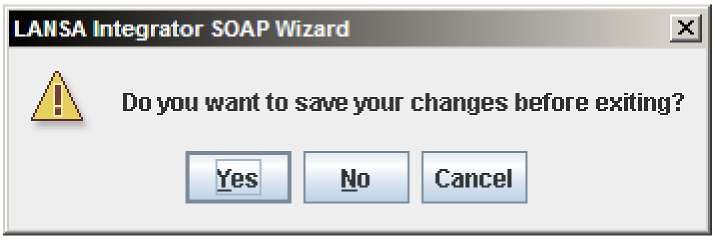


Service has been generated and compiled

Now exit the SOAP Wizard by clicking exit (the first button) on the toolbar.

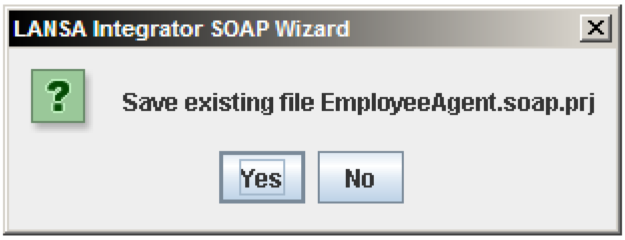

LANSA Integrator SOAP Wizard - EmployeeAgent.soap.prj (modified)
File
Project
Help
Exit
mployees
Parameters (Outbound)
String department_code (DEPTMENT)
String section_code (SECTION)
Return (Inbound)
Employee [] getEmployeesReturn (EMPLOYEE)
String employee_id (EMPNO)
String first_name (GIVENAME)
double salary (SALARY)
String surname (SURNAME)
Configuration
Operations

You will then be prompted to save your changes, click ‘YES’.



Do you want to save your changes before exiting?

The next dialog prompts you to recreate the project file, click ‘YES’



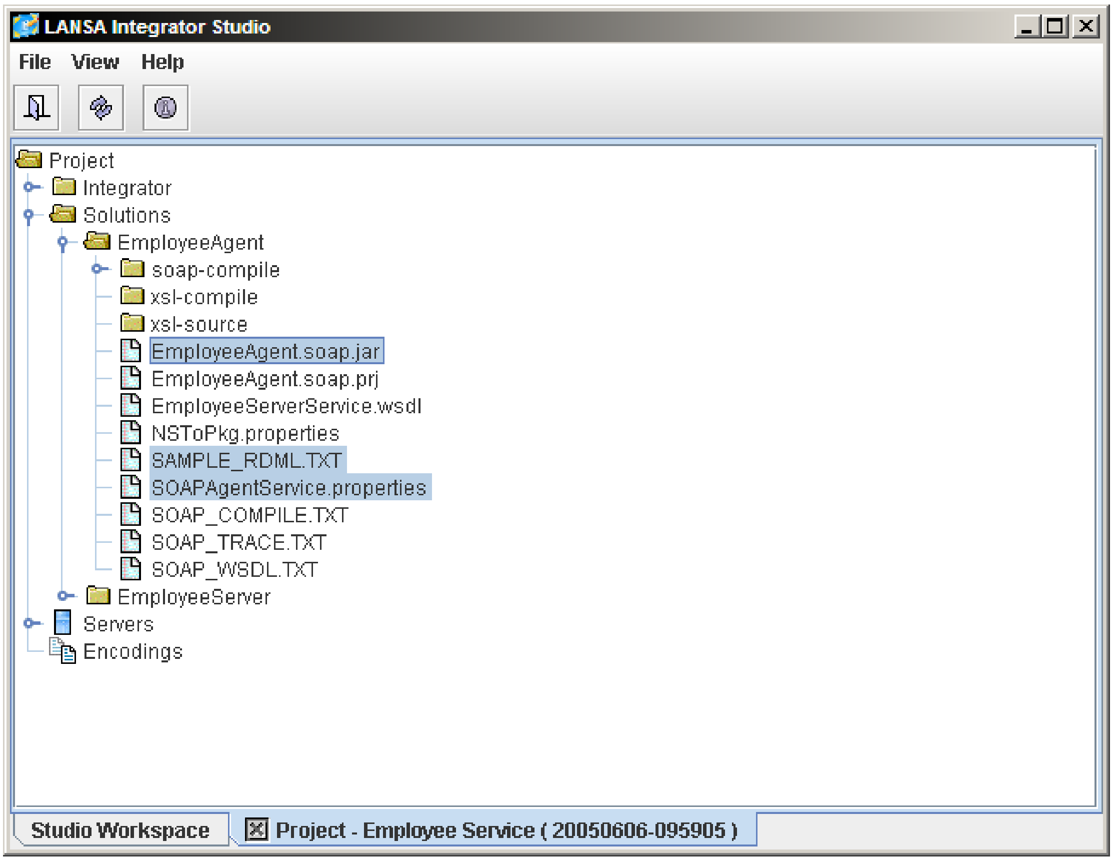
Save existing file EmployeeAgent.soap.prj

9. You are returned to the studio workspace for the EmployeeAgent project.

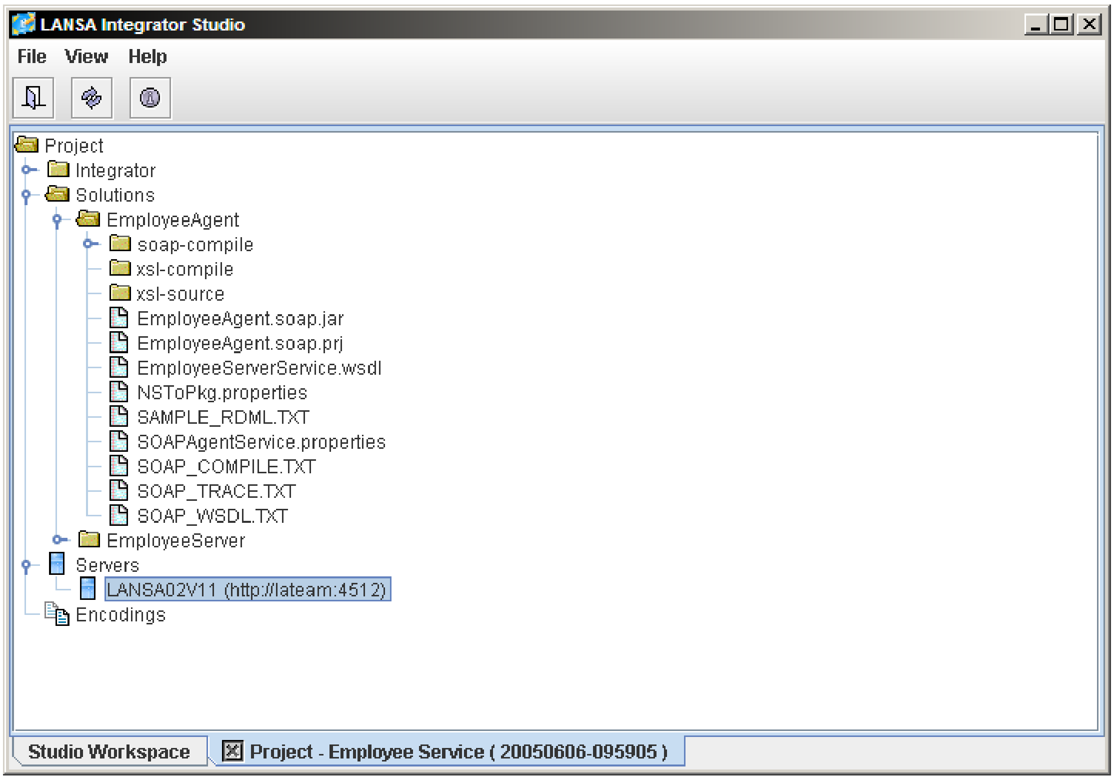
Open the 'EmployeeAgent' Solutions folder if it is not already open.

Note that the following files have been generated during the build.

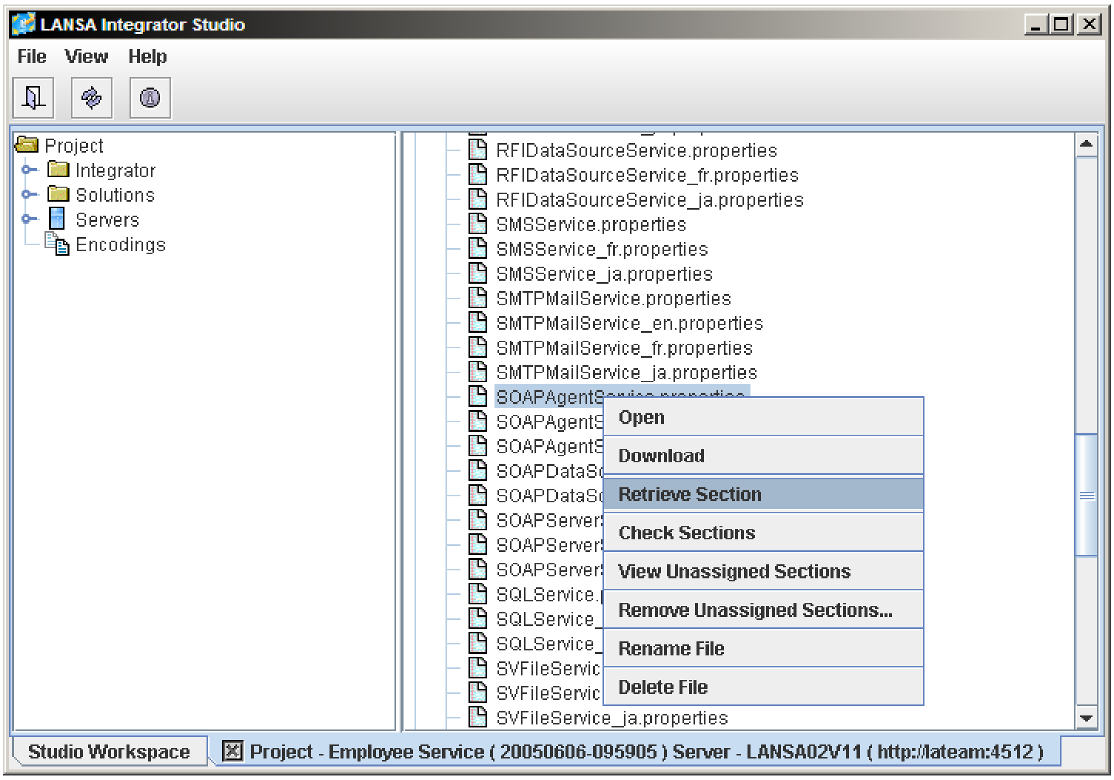
- The 'EmployeeAgent.SOAP.jar' is the jar file to support the sending and receiving of SOAP data to the client function.
- The 'SAMPLE_RDML.TXT' file contains an incomplete sample LANSA function. The function lacks the business logic required to process the SOAP request and response.
- The 'SOAPAgentService.properties' file has the configuration entries to register the SOAP Agent jar file to the SOAP Agent service.



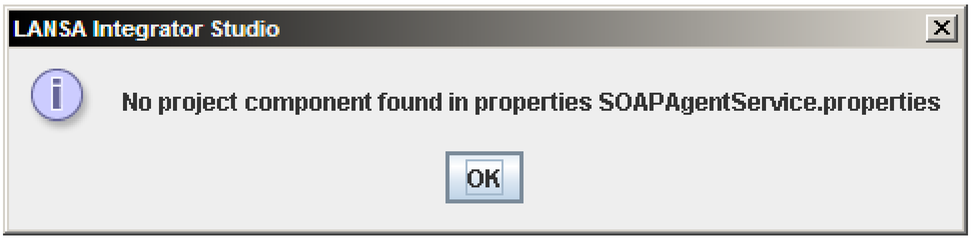
10. Double click or right mouse click and select ‘Open Server’ on the Server entry for your LANSA Integrator server that will be used for the SOAP agent.



The local studio files are now shown in the left pane and your LANSA Integrator server files in the right. To place the configuration entries in the SOAP Agent service properties, open the properties folder on your LANSA Integrator server in the right hand pane. Find the file 'SOAPAgentService.properties' file, right mouse click and select the 'Retrieve Section' option from the pop-up menu.

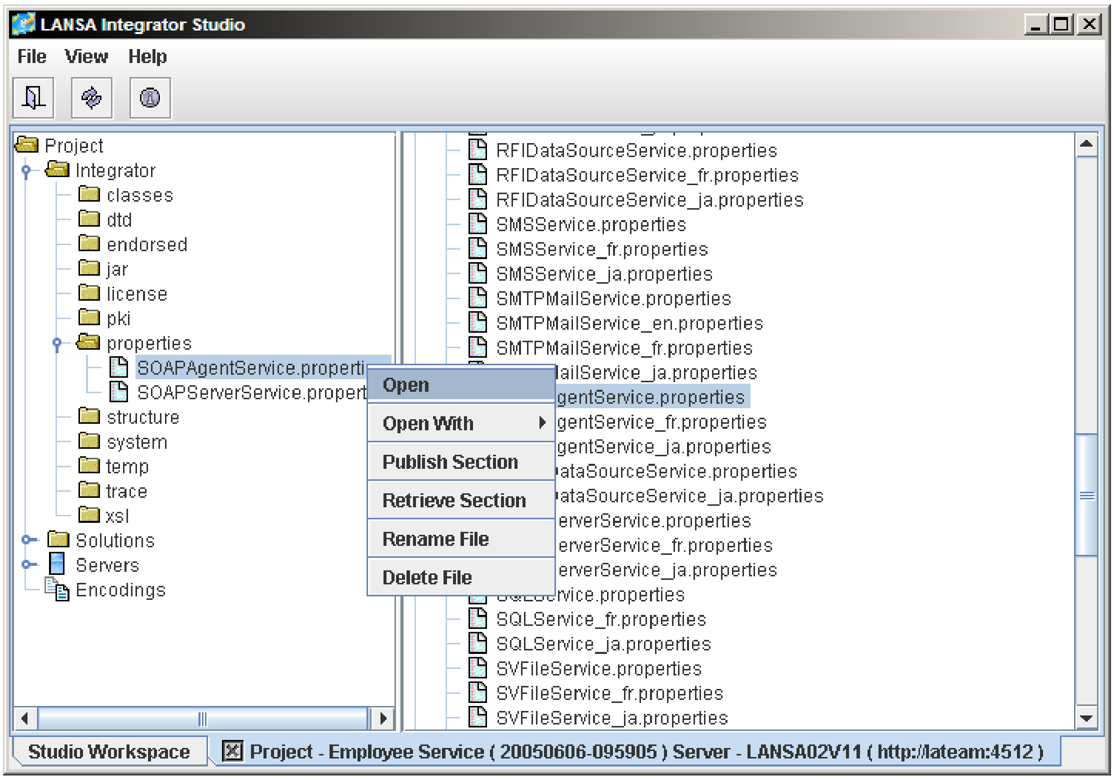

LANSA Integrator Studio
File View Help
Project
Integrator
Solutions
Servers
Encodings
RFIDataSourceService.properties
RFIDataSourceService_fr.properties
RFIDataSourceService_ja.properties
SMSService.properties
SMSService_fr.properties
SMSService_ja.properties
SMTPMailService.properties
SMTPMailService_en.properties
SMTPMailService_fr.properties
SMTPMailService_ja.properties
SOAPAgentService.properties
SVFileService_ja.properties
Open
Download
Retrieve Section
Check Sections
View Unassigned Sections
Remove Unassigned Sections...
Rename File
Delete File
Studio Workspace
Project - Employee Service (20050606-095905) Server - LANSA02V11 (http://lateam:4512)

A dialog will display due to the fact no configuration entries for this project in the properties file currently exist in the local studio workspace. Click 'OK' to continue.



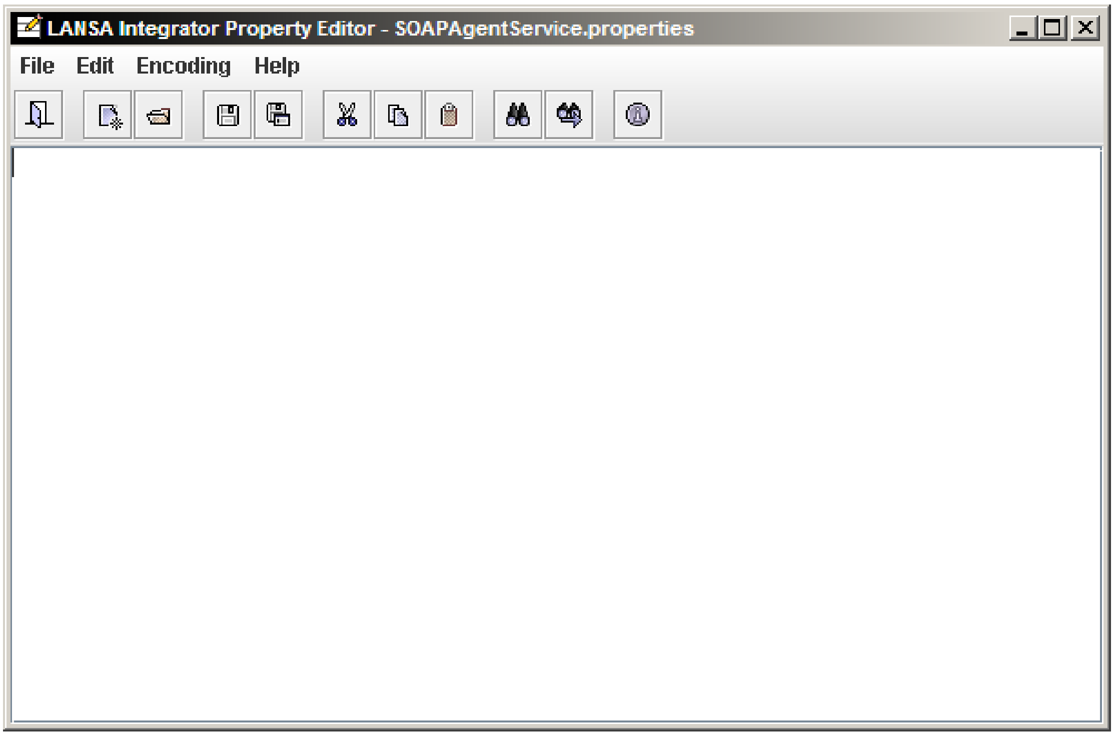
No project component found in properties SOAPAgentService.properties

Go to left hand pane to open the local copy of the ‘SOAPAgentService.properties’ file in the ‘Properties’ folder under the ‘Integrator’ folder. Right mouse click on the file and select ‘Open’ from the pop-up menu.

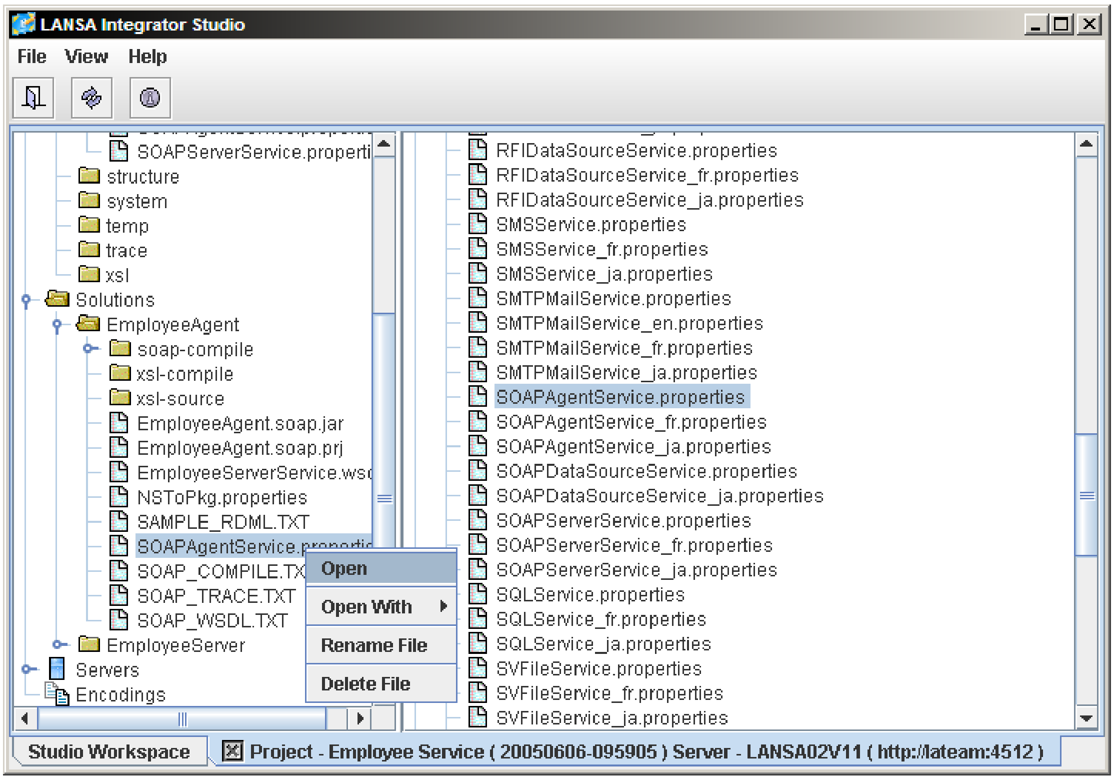


Project - Employee Service (20050606-095905) Server - LANSA02V11 (http://lateam:4512)

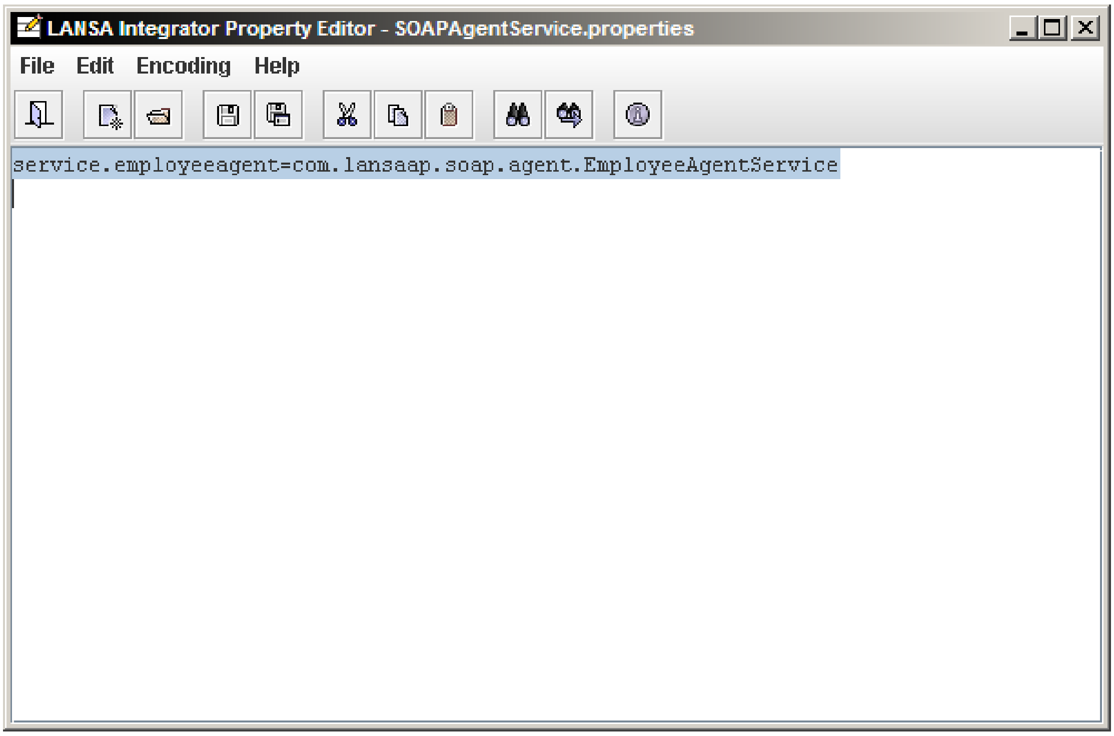
The properties editor will open showing the project currently has no configuration entries for this LANSA Integrator service.


LANSA Integrator Property Editor - SOAPAgentService.properties
File
Edit
Encoding
Help

Go back to the project workspace in studio and in the left hand pane go to the ‘EmployeeAgent’ solution group and find the SOAPAgentService.properties file. Select the file, right mouse click and select ‘Open’ from the pop-up menu.

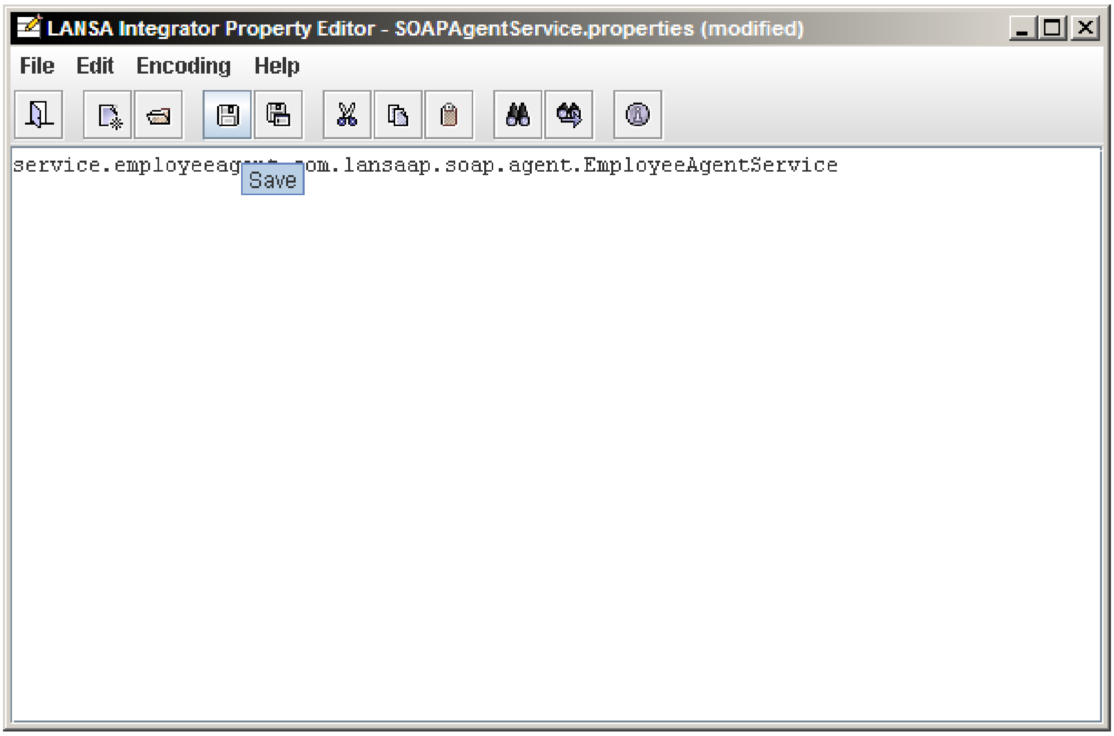

LANSA Integrator Studio
File View Help
SOAPServerService.properti
structure
system
temp
trace
xsl
Solutions
EmployeeAgent
soap-compile
xsl-compile
xsl-source
EmployeeAgent.soap.jar
EmployeeAgent.soap.prj
EmployeeServerService.wsc
NSToPkg.properties
SAMPLE_RDML.TXT
SOAPAgentService.properti
SOAP_COMPILE.TX
SOAP_TRACE.TXT
SOAP_WSDL.TXT
EmployeeServer
Servers
Encodings
Open
Open With
Rename File
Delete File
RFIDataSourceService.properties
RFIDataSourceService_fr.properties
RFIDataSourceService_ja.properties
SMSService.properties
SMSService_fr.properties
SMSService_ja.properties
SMTPMailService.properties
SMTPMailService_en.properties
SMTPMailService_fr.properties
SMTPMailService_ja.properties
SOAPAgentService.properties
SOAPAgentService_fr.properties
SOAPAgentService_ja.properties
SOAPDataSourceService.properties
SOAPDataSourceService_ja.properties
SOAPServerService.properties
SOAPServerService_fr.properties
SOAPServerService_ja.properties
SQLService.properties
SQLService_fr.properties
SQLService_ja.properties
SVFileService.properties
SVFileService_fr.properties
SVFileService_ja.properties
Studio Workspace
Project - Employee Service (20050606-095905) Server - LANSA02V11 (http://lateam:4512)

The configuration entry generated by the build now needs to be selected and copied either via Ctrl + C or the copy button on the toolbar.


LANSA Integrator Property Editor - SOAPAgentService.properties
File Edit Encoding Help

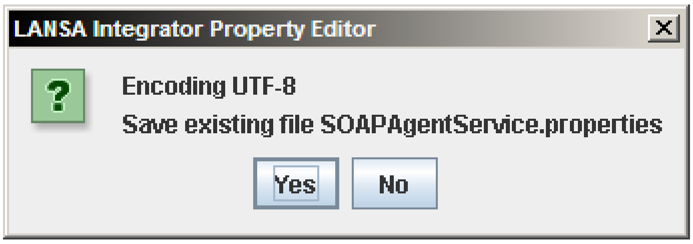
```
service.employeeagent=com.lansaap.soap.agent.EmployeeAgentService
```

Open the empty editor for the SOAPAgentSevice and paste in the configuration entry. Then click save (disk only button) on the toolbar.


LANSA Integrator Property Editor - SOAPAgentService.properties (modified)
File
Edit
Encoding
Help
Save

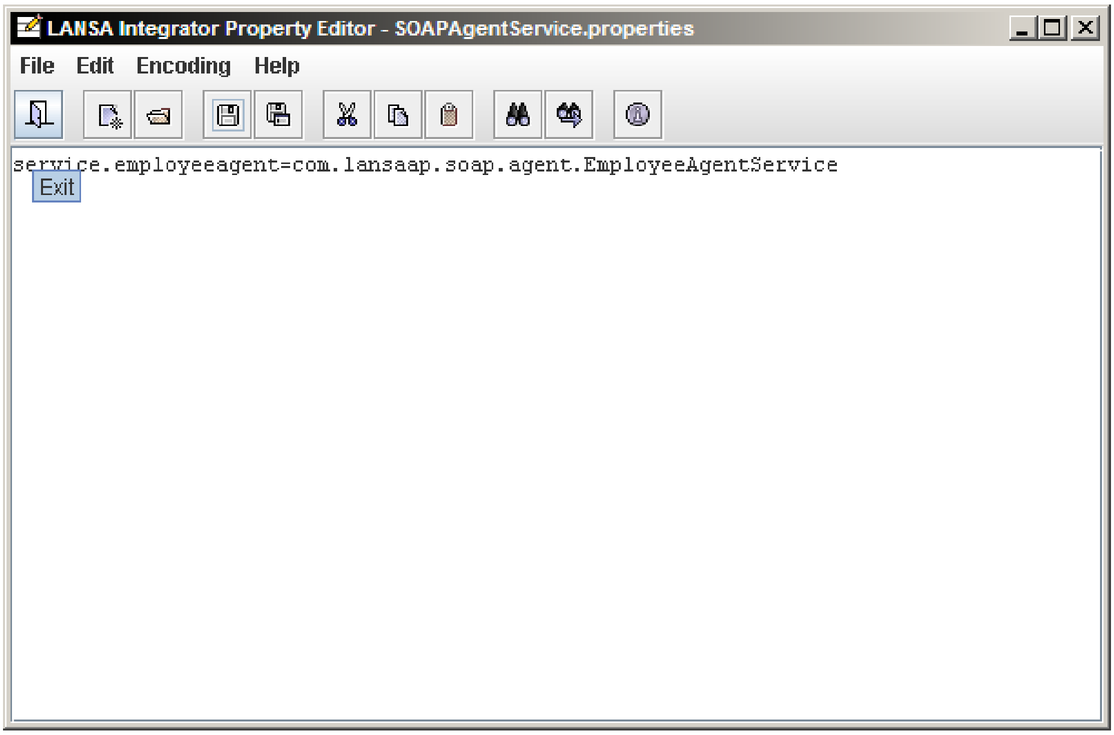
```
service.employeeagent=com.lansaap.soap.agent.EmployeeAgentService
```

Confirm the encoding of the file as UTF-8 by clicking ‘Yes’.



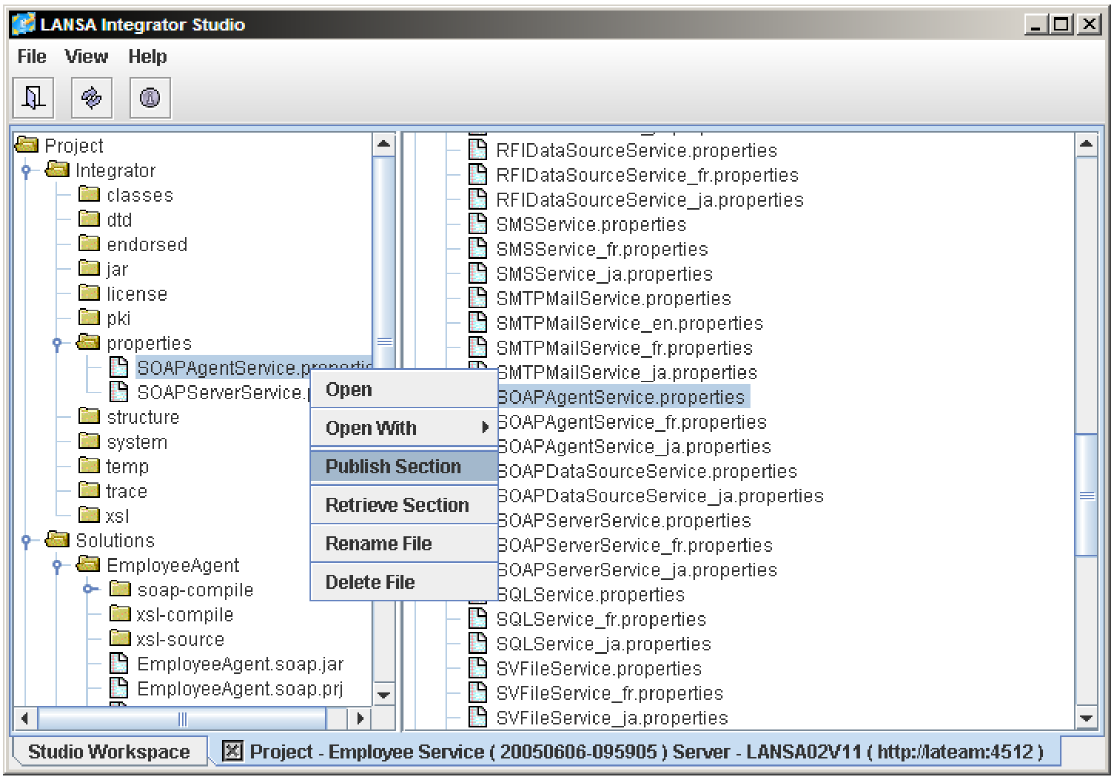
Encoding UTF-8
Save existing file SOAPAgentService.properties

Click exit (the first button) on the toolbar.

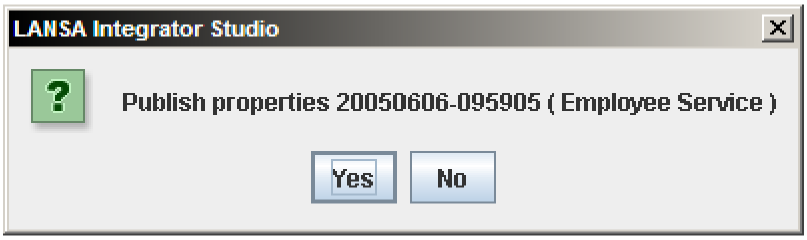

LANSA Integrator Property Editor - SOAPAgentService.properties
File Edit Encoding Help
Exit

```
service.employeeagent=com.lansaap.soap.agent.EmployeeAgentService
```

Right mouse click on the ‘SOAPAgentService.properties’ file under the properties folder which is under the Integrator folder in the left hand pane. Select ‘Publish Section’ from the pop-up menu.

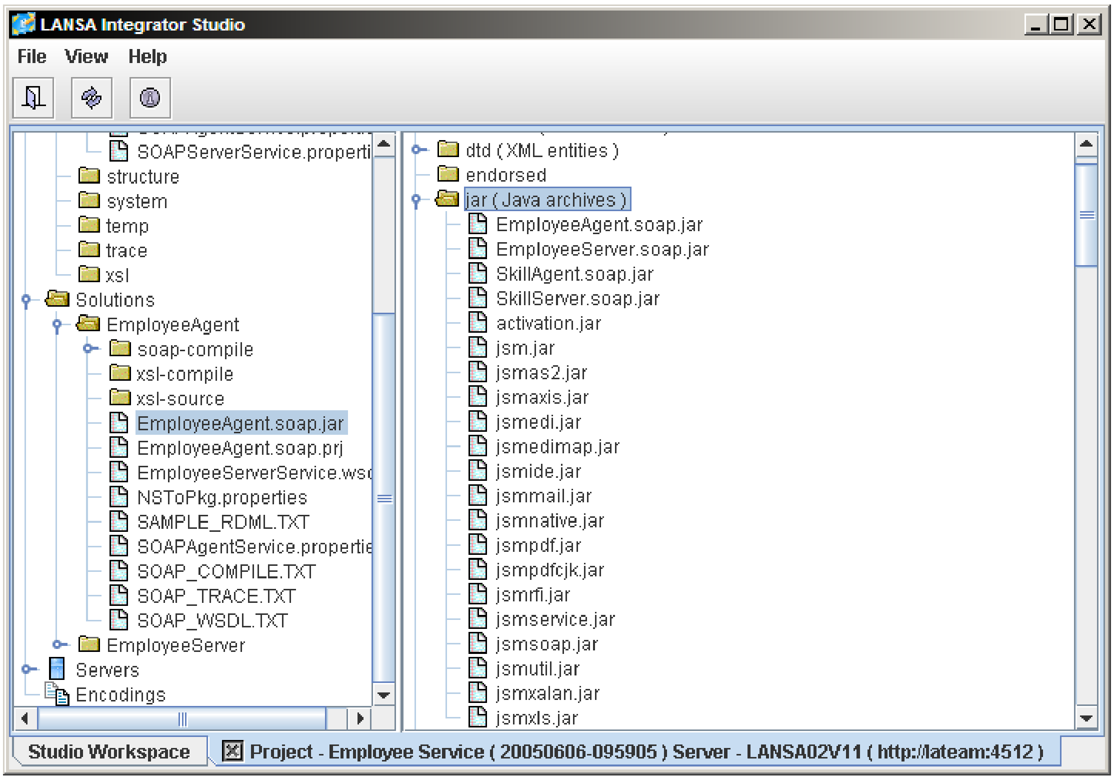

LANSA Integrator Studio
File View Help
Project
Integrator
classes
dtd
endorsed
jar
license
pki
properties
SOAPAgentService.properties
SOAPServerService.properties
structure
system
temp
trace
xsl
Solutions
EmployeeAgent
soap-compile
xsl-compile
xsl-source
EmployeeAgent.soap.jar
EmployeeAgent.soap.prj
Open
Open With
Publish Section
Retrieve Section
Rename File
Delete File
RFIDataSourceService.properties
RFIDataSourceService_fr.properties
RFIDataSourceService_ja.properties
SMSService.properties
SMSService_fr.properties
SMSService_ja.properties
SMTPMailService.properties
SMTPMailService_en.properties
SMTPMailService_fr.properties
SMTPMailService_ja.properties
SOAPAgentService.properties
SOAPAgentService_fr.properties
SOAPAgentService_ja.properties
SOAPDataSourceService.properties
SOAPDataSourceService_ja.properties
SOAPServerService.properties
SOAPServerService_fr.properties
SOAPServerService_ja.properties
SQLService.properties
SQLService_fr.properties
SQLService_ja.properties
SVFileService.properties
SVFileService_fr.properties
SVFileService_ja.properties
Studio Workspace
Project - Employee Service (20050606-095905) Server - LANSA02V11 (http://lateam:4512)

Click ‘Yes’ to finish the publishing of the configuration entries to your LANSA Integrator server.



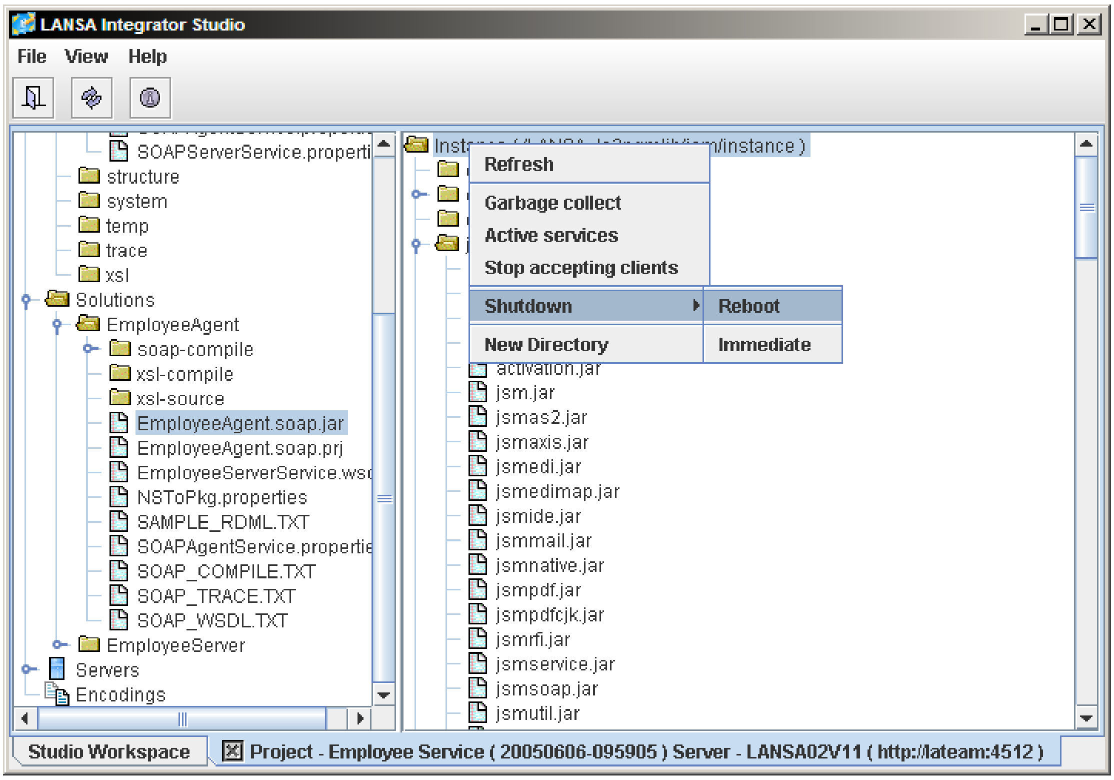
Publish properties 20050606-095905 (Employee Service)

11. In the left hand pane find the ‘EmployeeAgent.SOAP.jar’ file in the ‘EmployeeAgent’ solution group. Now click and drag the file to the ‘jar (Java Archive)’ folder in the right hand pane. This will copy the SOAP server jar file to your LANSA Integrator server so the LANSA function can process the SOAP requests.



Project - Employee Service (20050606-095905) Server - LANSA02V11 (http://lateam:4512)

12. In the right hand pane right mouse click on the 'Instance' folder for your LANSA Integrator server. Select 'Shutdown' and then 'Reboot' from the pop-up menu. This will stop and restart your LANSA Integrator server so the configuration entries and the jar file are picked up by your LANSA Integrator server.



13. Now a LANSA function need to be created that will process the SOAP requests and response. Below is a suggested RDML code for this function.

```
Function Options(*DIRECT)
*
Define Field(#JSMSTS) Type(*CHAR) Length(20)
Define Field(#JSMMSG) Type(*CHAR) Length(255)
Define Field(#JSMCMD) Type(*CHAR) Length(255)
*
Def_List Name(#EMPLANSA IntegratorST) Fields(#EMPNO #GIVENAME
#SURNAME #SALARY)
*
* Open service
*
Use Builtin(JSM_OPEN) To_Get(#JSMSTS #JSMMSG)
Execute Subroutine(CHECK) With_Parms(#JSMSTS #JSMMSG)
*
* Load service
Change Field(#JSMCMD) To('SERVICE_LOAD
SERVICE(SOAPAGENTSERVICE)  TRACE(*YES)')
Use Builtin(JSM_COMMAND) With_Args(#JSMCMD) To_Get(#JSMSTS
#JSMMSG)
Execute Subroutine(CHECK) With_Parms(#JSMSTS #JSMMSG)
*
Use Builtin(JSM_COMMAND) With_Args('OPEN
SERVICE(EMPLOYEEAGENT)') To_Get(#JSMSTS #JSMMSG)
Execute Subroutine(CHECK) With_Parms(#JSMSTS #JSMMSG)
*
Change Field(#JSMCMD) To('SET OPERATION(GETEMPLOYEES)')
Use Builtin(JSM_COMMAND) With_Args(#JSMCMD) To_Get(#JSMSTS
#JSMMSG)
Execute Subroutine(CHECK) With_Parms(#JSMSTS #JSMMSG)
* Get User Input
Request Fields(#DEPTMENT #SECTION)
* Set the 2 Parameters and bind their values
Change Field(#JSMCMD) To('SET PARAMETER(DEPARTMENT_CODE)
SERVICE_EXCHANGE(*FIELD)')
Use Builtin(JSM_COMMAND) With_Args(#JSMCMD) To_Get(#JSMSTS
#JSMMSG)
Execute Subroutine(CHECK) With_Parms(#JSMSTS #JSMMSG)
Change Field(#JSMCMD) To('SET PARAMETER(SECTION_CODE)
SERVICE_EXCHANGE(*FIELD)')
Use Builtin(JSM_COMMAND) With_Args(#JSMCMD) To_Get(#JSMSTS
#JSMMSG)
Execute Subroutine(CHECK) With_Parms(#JSMSTS #JSMMSG)
*
Use Builtin(JSM_COMMAND) With_Args('CALL') To_Get(#JSMSTS #JSMMSG)
Execute Subroutine(CHECK) With_Parms(#JSMSTS #JSMMSG)
* Process the Return array of employees
Begin_Loop
```

```
Use Builtin(JSM_COMMAND) With_Args('GET FRAGMENT(EMPLOYEE)
SERVICE_EXCHANGE(*FIELD)') To_Get(#JSMSTS #JSMMSG)
* SUBROUTINE(CHECK) WITH_PARMS(#JSMSTS #JSMMSG)
Leave If('#JSMSTS *NE OK')
Add_Entry To_List(#EMPLANSA IntegratorST)
*
End_Loop
Display Browselist(#EMPLANSA IntegratorST)
*
Use Builtin(JSM_COMMAND) With_Args('CLOSE') To_Get(#JSMSTS #JSMMSG)
Execute Subroutine(CHECK) With_Parms(#JSMSTS #JSMMSG)
*
* Unload service
*
Use Builtin(JSM_COMMAND) With_Args('SERVICE_UNLOAD')
To_Get(#JSMSTS #JSMMSG)
Execute Subroutine(CHECK) With_Parms(#JSMSTS #JSMMSG)
*
* Close service
*
Use Builtin(JSM_CLOSE) To_Get(#JSMSTS #JSMMSG)
Execute Subroutine(CHECK) With_Parms(#JSMSTS #JSMMSG)
*
* SUB ROUTINES
*
Subroutine Name(CHECK) Parms((#JSMSTS *RECEIVED) (#JSMMSG
*RECEIVED))
*
If Cond('#JSMSTS *NE OK')
Display Fields(#JSMSTS #JSMMSG)
*
Use Builtin(JSM_CLOSE) To_Get(#JSMSTS #JSMMSG)
*
Menu Msgtxt('Java service error has occurred')
*
Endif
*
Endroutine
```