## Contents

1. **SUGGESTED READING** ........................................................................................................... 3

2. **WHAT IS INFORMIX-4GL** .................................................................................................. 4
   2.1 **WHAT ARE THE INFORMIX-4GL DEVELOPMENT ENVIRONMENTS?** ................. 4

3. **INSTALLING THE 4GL SOFTWARE** ...................................................................................... 5

4. **STARTING INFORMIX** ....................................................................................................... 5

5. **WORKING WITH INFORMIX-4GL** .................................................................................... 7
   5.1 **SETTING UP YOUR ENVIRONMENT?** ........................................................................... 7
   5.2 **THE MENU-DRIVEN ENVIRONMENT** ......................................................................... 8
       5.2.1 Invoking the Menu-Driven Environment ................................................................... 9
   5.3 **THE COMMAND-LINE ENVIRONMENT** ...................................................................... 10
       5.3.1 Working with INFORMIX-4GL Forms .................................................................... 11
       5.3.2 Working with INFORMIX-4GL Source Code Modules ......................................... 13

6. **I4GL FUNCTIONS AS WEB SERVICES** ............................................................................... 20
   6.1 **WHAT IS W4GL?** ........................................................................................................ 20
   6.2 **WHAT IS W4GLC?** .................................................................................................... 20

7. **CREATING WEB SERVICE FROM I4GL FUNCTIONS** ..................................................... 21
   7.1 **PREREQUISITES** ......................................................................................................... 22
   7.2 **CREATING THE SOA DATABASE** ............................................................................... 22
   7.3 **ACCESSING THE MENU-DRIVEN W4GL TOOL** ...................................................... 23
   7.4 **DEFINING THE HOST AND APPLICATION SERVER** .................................................. 23
       7.4.1 Defining the Host .................................................................................................... 23
       7.4.2 Defining the Application Server ............................................................................. 24
   7.5 **DEFINING THE 4GL SOA DATABASE** ......................................................................... 24
   7.6 **CREATING A WEB SERVICE FROM A 4GL FUNCTION** ............................................ 25
       7.6.1 Creating a Web Service .......................................................................................... 25
       7.6.2 Deploying and Packaging the Web Service ............................................................ 27

8. **CONSUMING I4GL WEB SERVICES** .................................................................................. 30
   8.1 **SUBSCRIBING TO A WEB SERVICE** .......................................................................... 30
   8.2 **GENERATING THE CONSUMPTION CODE** ............................................................... 32
   8.3 **USING THE I4GL SOA DEMONSTRATION PROGRAM** .............................................. 33
1. Introduction

This lab describes how to use IBM INFORMIX-4GL to create 4GL applications and web services. This lab is especially useful for application developers who are asked to port their 4GL applications to different environments or to make their applications visible on the web.

2. Suggested Reading

IBM Informix 11.5 Information Center
This is the official online searchable Informix product documentation.
http://publib.boulder.ibm.com/infocenter/idshelp/v115/index.jsp

Deploy 4GL functions as Web Services
This developerworks article describes how to use existing 4GL functions as Web services.
www.ibm.com/developerworks/data/library/techarticle/dm-0902mottupalli

IBM Informix 4GL v7.50.xC2: Publication Library
This document provides links to documentation in PDF in all available languages for IBM® Informix® 4GL v7.50.xC2.
www.ibm.com/support/docview.wss?uid=swg27013652
3. What is Informix 4GL

Informix 4GL, also known as I4GL, is a fourth-generation application development language to create custom database management applications that access Informix databases. It was developed in the mid-1980s to provide power and flexibility without the need for third-generation languages like C and COBOL.

I4GL provides all the standard constructs of third-generation languages: variable assignments, looping, testing, and flow of control, as well as a rich collection of data types and programming statements for managing the interactions of a user with an Informix database engine.

I4GL runs only on UNIX systems and is designed to be a character-based environment.

3.1 What are the INFORMIX-4GL development environments?

Informix 4GL consists of 2 different development versions - **compiled** or **interpreted**. Each software version requires a separate license.

**Compiled** 4GL programs are created using the **INFORMIX-4GL Compiler Development Software**. Compiled programs are UNIX **executables** with a .4ge suffix – a preprocessor transforms the 4GL module(s) to C source code which is then compiled and linked as object code.
Interpreted 4GL programs are created using the INFORMIX-4GL RDS Development Software. Interpreted programs are p-code files with a .4gi suffix that are portable across machines and UNIX environments – an Informix compiler transforms each 4GL module to pseudo-code and then concatenates the p-code files into a single .4gi file. A runner is used to execute the single .4gi file.

Compiled programs generally execute faster than interpreted programs – however, interpreted versions are much quicker to compile, providing for a rapid development environment.

4. Installing the 4GL Software

Informix 4GL software should be installed before installation of Informix. Refer to the installation instructions provided with the software.

Generally, only one of the two software versions - compiled or interpreted - should be installed. Choose the software version that best meets your business requirements.

For demonstration purposes, both software versions have been installed on the VMWare image

5. Starting Informix

You will need the Informix database to be up and running in order to work with Informix-4GL.

Follow the steps below to start the Informix instance:

- Login into the VMWare image as the informix user.

   Note: Refer to the document entitled “VMWare Basics” for information about how to start and login to the VMWare Image.
Right click on the desktop area and choose Open Terminal.

Set the environment for the database server by typing the following commands in the command terminal.

```
    cd scripts
    . set4GL
```

Create the default 4gl working directory you will use for this lab.

```
    mkdir /home/informix/scripts/4gl
```

Enter the following command in the command terminal to see if the Informix Server is up and running.

```
    onstat -
```

- If the Informix server is NOT running then you will see the following message:

  ```
  Shared memory not initialized for INFORMIXSERVER
  'demo_on'
  ```
If the Informix Server is running then you will see a message similar to the following:

IBM Informix Dynamic Server Version 11.50.UC1WIDE --
On-Line - Up 00:00:15 - 38204 Kbytes

Enter the following command to start the server ONLY if it is NOT already running:

```
oninit
```

---

## 6. Working with INFORMIX-4GL

### 6.1 Setting up your Environment?

- Set the following Informix 11.5 environment variables:
  - **INFORMIXDIR**: Full path to the directory where I-4GL is installed. On the VM Image, it is a different from the Informix Server Installation directory
  - **PATH**: Make sure \$INFORMIXDIR/bin is included in the \$PATH
  - **INFORMIXSQLHOSTS**: Name of the SQLHOSTS file with full path
  - **ONCONFIG**: Name of the Informix configuration file

- Set the following INFORMIX-4GL environment variables:
  - **LD_LIBRARY_PATH**: Full path to the directories used to search for client or shared Informix general libraries
  - **DBPATH**: Identifies database servers that contain databases, specifies a list of directories (in addition to the current directory) in which DB–Access looks for command scripts (.sql files) and also defines the path to search for program resource files including FORM files and Message files
Set the following INFORMIX-4GL + SOA environment variables:

- **LOCALE**: en_US.8859-1
- **JAVA_HOME**: Full path to the installation directory of JRE 1.4 or higher
- **AXIS2C_HOME**: Full path to the AXIS2 installation directory
  
  Apache's AXIS2 C Web service API is preinstalled along with 4GL 7.50.xC2 and used to create and deploy 4GL functions as Web services

- **AXIS_JAR_DIR**: Full path to the AXIS2 JAVA classes
- **PATH**: Full path to JAVA and AXIS2 executables
- **CLASSPATH**: Full path to user-defined classes and packages used in Java programs. This also includes AXIS2 java classes

The above variables have already been setup for you in the file `set4GL`. Use the following command to view that environment file:

```
more /home/informix/scripts/set4GL
```

### 6.2 The Menu-Driven Environment

The menu-driven application development system visually guides you through the steps required to create an executable 4GL program. This includes the creation and compilation of ASCII forms (.per files) and source code modules (.4gl files), generation of object files, and final assembly of an executable program.

The installed INFORMIX-4GL development software version governs how to access to the menu-driven application:

- **i4gl** (compiled software version)
- **r4gl** (interpreted software version)
The 4GL menu-driven development application is the same irrespective what command is used. However, the commands invoked upon selection of certain menu options are different based the development version installed.

6.2.1 Invoking the Menu-Driven Environment

The menu-driven Environment uses ring menus at every level.

- Using the same terminal window opened in the previous section, enter the following command to access the menu-driven environment:

  `i4gl`

- You are provided with a window similar to the following:

  ![Image of menu-driven environment]

- The options in the main menu functionally group the various tasks that are performed on each component of a completed application program.
You are provided with a submenu to create and edit source code modules (Module), create and edit screen forms (Form), and assemble modules into a complete program (Program), and if you installed INFORMIX-SQL on your system, you can access its main menu (Query-language).

- Explore the various sub-menus
  - You can access the various options of each ring menu by either typing the capital letter that identifies the option or by moving the cursor with the space bar or the arrow keys and pressing the Enter or Return key when the desired option is highlighted.
  - All menus and submenus in the Menu-driven Environment offer an Exit option to navigate through the menus or to terminate the development session.

- Connect to the i4gldemo database you created in the previous sections
  - Choose Query-Language and use the arrow keys to select the i4gldemo database

  Does the new Menu displayed after connection to the i4gldemo database look familiar?

- Exit i4gl.

### 6.3 The Command-Line Environment

The command-line environment allows you to create the ASCII files that define the forms (.per files) and source code modules (.4gl files) that make up a program. You can manually create these files using the editor of your choice.
6.3.1 Working with INFORMIX-4GL Forms

You can compile Informix-4GL Forms using the command `form4gl` command. The structure of this command is identical whether you use the compiled version or the interpreted (RDS) version of INFORMIX-4GL.

- The `form4gl` command is invoked as follows:
  
  ```
  form4gl { [-l <screen lines of display (default = 24)>]
  [-c <display columns >]
  [-v]
  [-q] } <file name>.per
  ```

  The options definitions are:
  
  - `-l` indicates the lines available on the terminal screen for displaying the form
  - `-c` indicates the columns available on the terminal screen for displaying the form
  - `-v` makes the compiler verify that the allocated display space for fields matches the definitions of the fields
  - `-q` makes the compiler operate in silent mode; no messages are sent to the terminal screen unless there are errors in compilation

- Generate a Default Form using `form4gl`
  
  ```
  mkdir /home/informix/scripts/4gl/forms
  cd /home/informix/scripts/4gl/forms
  form4gl -d
  ```
For the next few prompts, type the following information:

- Enter the desired name for the form:
  
  \texttt{customer.per}

- Enter the name of the database used by the form:
  
  \texttt{i4gldemo}

- Enter a table name:
  
  \texttt{customer}

- Enter a table name:
  
  <Press the ENTER key>

- View the \texttt{customer.per} Form file you created

  more customer.per

A 4GL form can have up to 5 sections of which 3 are required. They appear in the following order:

- DATABASE (Required) - identifies the database (if any) on which you want to base the form

- SCREEN (Required) - Shows the exact layout of the form as you want it to appear on the screen.

- TABLES (Optional) - Identifies the table for any screen fields defined with the name of the column in a table of the specified database.

- ATTRIBUTES (Required) - Describes each field on the form and assigns names to each field (which the INFORMIX-4GL program uses). The link between the field in the SCREEN section and that in the ATTRIBUTES section is the field-tag.

- INSTRUCTIONS (Optional) - You can specify non-default delimiters and can define screen records and screen arrays in this section.

Can you identify the sections above in the file \texttt{customer.per}?
The form4gl –d command created a form (.per suffix) and compiled it too (.frm). Normally, you would create your form and *manually* compile it using form4gl. Try compiling yourself:

```
rm customer.frm
form4gl customer.per
```

Was there a new `customer.frm` file created in your working directory?

### 6.3.2 Working with INFORMIX-4GL Source Code Modules

INFORMIX-4GL source code modules are comprised of one or more ASCII files whose names end with a .4gl suffix.

The 4GL source code modules remain the same irrespective of whether you are developing using the interpreted or the compiled development version – only the compilation and executions steps are different.

#### 6.3.2.1 Working with 4GL Source Code using the INTERPRETED version

Interpreted (RDS) development versions require the `fglpc` command in order to compile source code. This command creates p-code object files (.4go extension files).

* The `fglpc` command is invoked as follows:

```
fglpc { [-ansi]
    [-a]
    [-anyerr]
    [-p <pathname>] } <.4gl source code module> ...
```
The options definitions are:

- `ansi` forces the compiler check all SQL statements for ANSI compliance
- `a` forces the compiler check array bounds at runtime.
- `anyerr` forces the runner set the status variable after evaluating expressions (overriding the WHENEVER ERROR statement in the code)
- `p` stores objects in the directory specified in `<pathname>`.

If there is more than one source code module, they can be compiled together by including them in a list. Successful compilation of source code files creates corresponding `.4go` files in the same directory – else, the appropriate `.err` files are created in the current directory.

To create an executable from error-free compiled source code files, simply concatenate all the object files (.4go) into one p-code executable file with a `.4gi` suffix. You can use the UNIX command `cat` to create the executable:

```
cat <filename>.4go ... <executable filename>.4gi
```

To execute programs in the interpreted version of the software, use the command `fglgo`.

- The `fglgo` command is invoked as follows:

  ```
  fglgo [-a]
        [-anyerr] <program name>.4gi
  <program arguments>
  ```

  The options definitions are:
  
  - `a` forces the runner check array bounds at runtime
  - `anyerr` forces the runner set the status variable after evaluating expressions (overriding the WHENEVER ERROR statement in the code)
  
  `<program name>.4gi` is the name of the p-code program to execute.
-<program arguments> are command-line options and arguments that your program was designed to receive at runtime.

The interpreted (RDS) INFORMIX-4GL development environment comes pre-installed with a demonstration program which both compiles and concatenates the object files.

- View the script to create the demonstration program
  ```
  more /opt/IBM/4gl/demo/fgl/en_us/0333/r4gldemo
  ```

Can you identify the command to compile the source files?
Can you identify the command to concatenate the object files?

- Compile and create the demonstration 4GL program
  ```
  mkdir /home/informix/scripts/4gl/rds
  cd /home/informix/scripts/4gl/rds
  r4gldemo
  ```

  Enter Y to any prompts.

- Execute the demonstration program executable created on the previous step (interpreted versions require an .4gi suffix)
  ```
  fglgo demo4.4gi
  ```

- Explore the various menus and sub-menus of your choice. Some functions to explore include:
  - Query some tables (Customer, Orders menu options)
  - Add/Update a new Customer or Order
  - Generate a report under the Reports menu option
Files with `.4go` extensions can also be executed using the utility `fglgo`.

Let's quickly look at a 4gl source file and then compile it to create a p-code object file:

```
more custmenu.4gl
fglpc custmenu.4gl
```

Look to see if a p-code file called `custmenu.4go` was created?

P-code object files with a `.4go` extension can also be executed using the `fglgo` utility. Execute the following command and explore the menu options:

```
fglgo custmenu.4go
```

Exit the menu.

---

### 6.3.2.2 Working with 4GL Source Code using the COMPILED version

Compiled development versions require the `c4gl` command in order to compile and link the source code module(s). This command also creates an executable file.

The `c4gl` command is invoked as follows:

```
c4gl { [-ansi]
    [-a]
    [-e]
    [-anyerr]
    [-args]
    [-o <program nam>.4ge] } <.4gl source code module>
    <.ec files> ...
```
The options definitions are:

-`-ansi` forces the compiler to check all SQL statements for ANSI compliance

-`-a` forces the compiler to check array bounds at runtime

-`-e` forces the compiler to perform only preprocessing steps; no compilation is performed

-`-anyerr` makes the runner set the status variable after evaluating expressions (overriding the WHENEVER ERROR statement in the code)

-`-args` are other arguments that you want to use with the C compiler.

-`-o <program name>.4ge` is the name you want to give the executable. (By default, it is `a.out`.)

- `<.4gl source code module>` is a list of source code module files to compile and link into the program executable

- `<.ec files>` is a list of ESQL/C files to compile and link into the program executable

- `<.c files>` is a list of C language source code files to compile and link into the program executable

- `<.o files>` is a list of object code files to link into the program executable.

- `<library>` is the name of the library to use in resolving function names that are not part of INFORMIX-4GL or ESQL/C

If there is more than one source code module, they can be compiled together by including them in the `c4gl` command list. Successful compilation of all source code files creates a `.4ge` executable (binary) in the same directory – else, the respective `.err` files are created in the current directory.
To execute programs in the compiled version of the software, simply run the executable created as part of the compilation process.

- To execute a compiled 4GL program on the platform it was compiled on:
  
  `<program name>.4ge  <program arguments>`

  The options definitions are:
  
  `<program name>.4ge` is the name of the executable file
  
  `<program arguments>` are any command-line options and arguments that your program was designed to receive at runtime

The compiled INFORMIX-4GL development environment comes pre-installed with a demonstration program which both compiles and links the source files to create an executable file.

- View the script to create the demonstration program

  `more /opt/IBM/4gl/demo/fgl/en_us/0333/i4gldemo`

  Can you identify the command to compile the source files?

- Compile and create the demonstration 4GL program

  `mkdir /home/informix/scripts/4gl/compile`
  
  `cd /home/informix/scripts/4gl/compile`
  
  `i4gldemo`

  Enter `Y` to any prompts.
 Execute the demonstration program executable created on the previous step (interpreted versions require an .4ge suffix)

 ./demo4.4ge

 Explore the various menus and sub-menus of your choice. Some functions to explore include:

 - Query tables (Customer, Orders menu options)
 - Add/Update a new Customer or Order
 - Generate a report under the Reports menu option

 Is there any difference in the behavior of this compiled program versus the interpreted program in the previous section?

 Let’s use the c4gl utility to compile a Report 4GL module.

 Quickly look at a 4gl source file and then compile it to create a compiled executable:

 more report1.4gl
 c4gl report1.4gl -o report1

 Look to see if an executable called report1 was created?

 Execute the following command and analyze the output:

 ./report1 | more

 We are piping the output of the report to the Linux utility more to allow you to see the results of the report.

 Exit the report/more utility.
7. **I4GL Functions as Web Services**

Beginning with INFORMIX-4GL version 11.50.xC2, I4GL functions can be deployed or consumed as Web services through the Apache's AXIS2 C Web service API.

AXIS2 is the successor to the Apache Axis SOAP project. It is becoming increasingly popular as the platform for the next generation of Web services and Service-Oriented Architecture (SOA).

7.1 **What is W4GL?**

W4GL is a tool that is written using Informix 4GL and designed to create and deploy 4GL functions as Web services.

W4GL provides an interface to collect the required user inputs to create a web service, generate a web service configuration file, and deploy or package the web service.

7.2 **What is W4GLC?**

W4GLC is a *non-interactive* Perl script called by W4GL used to:

- generate a web service configuration file
- compile all files associated with a web service to create a shared object library file
- deploy a web service on the local server
- package a web service for deployment to another server.

This script is located in the `$INFORMIXDIR/bin` directory.
The `w4glc` command is invoked as follows:

```bash
w4glc {
    [-help] [-version] |
    [-silent] [-force] [-generate] [-compile]
    [-deploy] [-package]
}
<config-file>
```

The options definitions are:

- `-silent` - Do not display the progress
- `-force` - Avoid user prompts
- `-generate` - Generate the code
- `-compile` - Compile the generated code
- `-deploy` - Deploy the web service
- `-package` - Create a package (tar-format) file for web-service

---

## 8. Creating Web Service from I4GL Functions

The W4GL interactive tool allows you to enter information required to create a web service based on a 4GL function. It also provides the interface to generate configuration files and deploy them as a Web service.

In our example, we will create a web service that will accept a zip code as input and return the corresponding city and state to the calling consumer. The 4GL function which will be deployed as a Web service is located in the following file:

```
/opt/IBM/4gl/demo/soa/en_us/0333/soademo.4gl
```
View the 4GL function `validate_visa_card()` and in the file:

```bash
more /opt/IBM/4gl/demo/soa/en_us/0333/soademo.4gl
```

Can you identify the input parameter the function accepts?
Can you identify the output parameters the function returns?

## 8.1 Prerequisites

The following prerequisites are needed to create 4gl-based web services:

- Informix 4GL 7.50.xC2 (Currently supported only on Linux®)
- JRE 1.4.2 or later
- AXIS2C 1.5 (Bundled with 4GL 7.50.xC2 and later.)
- PERL tool
- Informix 9 or later

## 8.2 Creating the SOA Database

The W4GL tool provides an interface to manage definitions of any Informix databases used by 4GL-SOA. However, before you can setup any definitions, the database needs to exist.

A database that includes a table which will be used by the web service has already been created for you.

- Verify using `dbaccess` that a database named `i4glsoa` exits.
- Verify also that this database contains one table named `statedetails` with 3 columns defined.
- EXIT dbaccess
8.3 Accessing the menu-driven W4GL Tool

- Create the working directories for the W4GL tool:
  ```
  mkdir /home/informix/scripts/4gl/temp
  mkdir /home/informix/scripts/4gl/soa
  cd /home/informix/scripts/4gl/soa
  ```

- Enter the following command to access the W4GL tool:
  ```
  w4gl
  ```

8.4 Defining the Host and Application Server

Before adding a Web service, you must add the host name of an application server and the application server on which the service needs to be deployed.

8.4.1 Defining the Host

- Choose the menu options **Host name** ➔ **Add** and type the following for each of the 2 fields mentioned below (use the UP and DOWN arrow keys to move between the fields):
  ```
  Host Name: suse1
  Temporary Directory: /home/informix/scripts/4gl/temp
  ```

- Press the ESC key to save the information

- Choose the **Exit** option leave the Host Name menu

**Note:** The **Machine ID** field is automatically populated with the next serial value.
8.4.2 Defining the Application Server

- Choose the menu options **App Server → Add** and type the following in the 2 fields mentioned below (use the UP and DOWN arrow keys to move between the fields):

  Server Name: my_app_server
  Host Name: suse1

  **Note:** The **Server ID** and **Port Number** fields, and all the fields under the **ENVIRONMENT VARIABLES** section, are automatically populated using information from your environment.

- Press the ESC key to save the information

- Choose the **Exit** option leave the App Server menu

8.5 Defining the 4GL SOA database

Any databases accessed by the 4GL function(s) need to be registered.

- Choose the menu options **Publish → Database → Add** and type the following:

  Database Name: i4glsoa

  **Note:** The **Database Server**, **Informix Version**, and **DB_LOCALE** fields are automatically populated using information from your environment.

- Press the ESC key to save the information

- Choose the **Exit** option leave the Database menu
8.6 Creating a Web Service from a 4GL Function

The W4GL provides several screens to add a web service. Web services can also be modified or removed using this tool.

8.6.1 Creating a Web Service

Creating a Web Service based on a 4GL function involves specifying details of the 4GL function, including the name of the function, the data types of its input and output parameters, and the name of the 4GL file that contains the function.

- Choose the menu options Web Service → Add → Detail and type the following:
  
  Service Name:  ws_visa_valid
  Function Name:  zipcode_details

  **Note:** The Service ID field is automatically populated with the next serial value. The Notes field can be left empty.

- Press the ESC key to save the information

- Choose the menu option Variable → Input and type the following on the first line corresponding to Input Parameter #1 (use the TAB key to move to the next field):
  
  Variable Name:  ZIP
  Variable Data Type:  CHAR(10)

- Press the ESC key to save the information
Choose the menu option **Output** and type the following on the *first* line (use the TAB key to move to the next field):

- **Variable Name:** CITY
- **Variable Data Type:** CHAR(100)

Type the following on the *second* line (use the TAB key to move to the next field/line):

- **Variable Name:** STATE
- **Variable Data Type:** CHAR(100)

Press the ESC key to save the information

Choose the **Exit** option leave the Variable menu

Choose the menu option **File** and type the following for the 2 fields mentioned below (use the UP and DOWN arrow keys to move between the fields):

- **Directory:** /opt/IBM/4gl/demo/soa/en_us/0333
- **File Name:** soademo.4gl

Press the ESC key to save the information

Choose the **Exit** option leave the Add menu

Choose the **Save-and-Exit** option and press the Enter key
Verify that the web service has been created by querying the `syspgm4gl` database:

- Choose the **Query** option from the Web Service Menu and press the ESC key

You will see the information you entered in the **WebService** and **Function Name** fields. Choosing options **Variable** and **File** will show your other entered information.

### 8.6.2 Deploying and Packaging the Web Service

Once the web service has been created or updated, it can be deployed and packaged when necessary.

#### 8.6.2.1 Generating the Web Service Configuration File

The **Generate** option creates the configuration file required to deploy a web service.

- Choose the menu options **Install → Generate** from the Web Service Menu and enter the following information in each field mentioned below:
  
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Name</td>
<td>Control-B, Query, ESC key, ESC key</td>
</tr>
<tr>
<td>Host Name</td>
<td>Control-B, Query, ESC key, ESC key</td>
</tr>
<tr>
<td>App-Server Name</td>
<td>Control-B, Query, ESC key, ESC key</td>
</tr>
<tr>
<td>Database Name</td>
<td>Control-B, Query, ESC key, ESC key</td>
</tr>
</tbody>
</table>

**Note:** The **Service ID, Machine ID, Server ID, Database, Temp Directory and Port Number** fields are automatically populated.

- Press the ESC key to save ALL the above information
Open another terminal window and analyze the contents of generated configuration file which includes the specifications of this web service:

more
/home/informix/scripts/4gl/soa/ws_visa_valid_suse1.4cf

Does the information in this configuration file look familiar?

- Close this terminal window opened to view the web service specification file above.

### 8.6.2.2 Deploying the Web Service

The *Deploy* option uses the W4GLC utility to generate the code for the web service using the configuration file generated above.

- Choose the menu option **Deploy** from the Install Menu

Note the default specification file that is populated for you:

ws_visa_valid_suse1.4cf

- Since this is the file that was created earlier and now needs to be used to deploy the associated web service, enter the following command in the first field below to use this *default* file name:

File Name: ESC key

- Where is the web service deployed?

  **Hint:** Look at the LOG file `/tmp/w4glerr.log` …

- What are the 3 files related to this web service?
8.6.2.3 Packaging the Web Service

The Package option creates a zip/tar file containing all 3 files from the deployment step above. Packing allows the web service information to easily be deployed to different hosts.

Because the code for the web service is “generated” using the local operating system, hosts where the web service is deployed must also have the same operating system and configuration.

- Choose the menu option Package from the Install Menu and enter the following command in the first field below to use the default file name:
  File Name: ESC key

- Where is the packaged web service files located?
  Hint: What is the value of TEMP directory defined on the HOST ...

- What are the contents of the packaged web service file?
  Hint: use the command tar –tvf <filename>

- Choose the Exit option four times to leave the W4GL tool.
9. Consuming I4GL Web Services

A Web service is only useful when published on a Web server for consumption by other applications.

A Web service consumer locates a Web service and invokes the operations it provides. Web Service Consumer applications can be any application that has network access to the Web Service Provider server. These servers can be accessed through internal (LAN) or external (WAN) networks and the internet.

9.1 Subscribing to a Web Service

The W4GL tool also provides an interface to consume or subscribe to a web service by specifying it’s web service definitions. This interface uses the web service name and the location of associated WSDL file you specify to generate the Axis2/C interface files. These files can then be copied to a location where an application server can consume them.

- Enter the following command to access the W4GL tool:
  
  ```
  mkdir /home/informix/scripts/4gl/sub_ws
  cd /home/informix/scripts/4gl/sub_ws
  w4gl
  ```

- Choose the menu options **Subscribe → Add → Detail** from the Web Service Menu to enter the following information in each field mentioned below:
Webservice Name: ws_visa_valid
Webservice Function: validate_visa_card
I4GL Function: zipcode_details
WSDL Directory: 
    /opt/IBM/4gl/AXIS2C/services/ws_visa_valid
WSDL File: zipcode_details.wsdl
Namespace: http://www.ibm.com/service
Target Directory: /home/informix/scripts/4gl/sub_ws
Target Filename: validate_zip.c

Press the ESC key to save the information

Choose the menu options **Variable → Input** from the Web Service Menu to enter the web service Input parameter as follows:

- Variable Name: ZIP
- Variable Data Type: CHAR(10)

Press the ESC key to save the information

Choose the menu options **Variable → Output** from the Web Service Menu to enter the web service Output parameters as follows:

- Variable Name: CITY
- Variable Data Type: CHAR(100)

Type the following on the **second** line (use the TAB key to move to the next field/line):

- Variable Name: STATE
- Variable Data Type: CHAR(100)
Press the ESC key to save the information

Choose the Exit twice (if necessary) option leave the Add menu

Choose the Save-and-Exit option and press the Enter key

Verify that the information to consume the web service has been created by querying the syspgm4gl database:

Choose the Query option from the Web Service SUBSCRIBE Menu and press the ESC key

You will see the information entered earlier for consuming the Web Service including it’s name, the WSDL file location, the target directory where the consumable file will be located and it’s C filename.

Choosing the Variable option will show the Input and Output parameters you entered.

9.2 Generating the Consumption Code

The web service consumption code generated is an Object file. This object file can be linked into an application allowing the application to call/consume the web service!

To create a consumable object file:

Choose the Compile option from the Web Service SUBSCRIBE Menu.

NOTE: The object file will automatically have been generated when the message Subscriber code has been compiled successfully is displayed.
Where is the object file located?

**Hint:** Check the value of web service “Target Directory” ...

What is the name of the consumable object file?

**Hint:** Check the value of “Target File Name” ...

Open another terminal window and analyze the contents of generated C file which includes AXIS2/C interface functions:

```
more /home/informix/scripts/4gl/sub_ws/validate_zip.c
```

Close the window after viewing ...

Choose the **Exit** option two times to leave W4GL tool.

### 9.3 Using the I4GL SOA demonstration program

The compiled INFORMIX-4GL development environment comes pre-installed with a demonstration SOA program which both publishes and subscribes to an INFORMIX-4GL Web Service.

View the script to create the demonstration program

```
more
/opt/IBM/4gl/demo/soa/en_us/0333/i4glsoademo.sh
```

Can you identify the 4GL program to recreate the SOA database and table? What is the name of the table?

Can you identify the 4GL program to load the SOA database table?
Can you identify the commands to create the web service configuration file? What is the name of the config file?

Can you identify the Perl command to deploy the web service using the above generated configuration file? What is the name of the script that deploys the web service?

Can you identify the command to start the AXIS2C application server? Where does the command executable reside?

Can you identify the command to create the executable for consuming the web service? Hint: the actual command resides in a makefile. What is the name executable?

➢ Enter the following command to create a working directory:
  mkdir /home/informix/scripts/4gl/soa_demo

➢ Enter the following command to move to the working directory:
  cd /home/informix/scripts/4gl/soa_demo

➢ Run the SOA demonstration program to publish and consume a web service
  i4glsoademo

Can you identify the responses to the supplied zip codes from the web service?
How many times was this web service consumed? **Hint:** how many “responses” were there?

This concludes the I4GL lab. If you have completed all the exercises and have extra time, feel free to continue exploring any of the utilities/functionality described here. Otherwise, you can close the open any open windows.