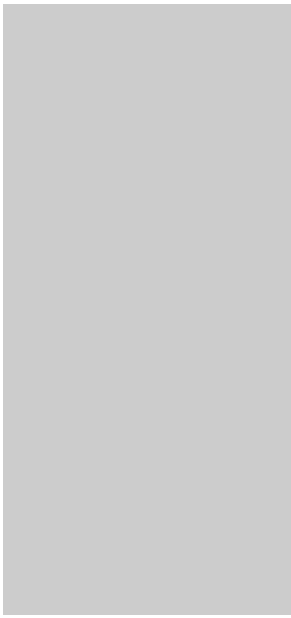




**Application Developers'**  
**Intro School**  
Specifications Binder  
Segment 7





## Table of Contents

Activity Instructions .....	4
Overview .....	4
Activity Steps .....	4
Key Learning Points .....	5
Learning Materials .....	5
Step 1 – Prepare for this Activity .....	6
Step 2 – Prepare the Environment .....	6
Step 3 – Update and Code JSP Pages .....	7
Step 4 – Execute Scripts, Verify, Debug and Fix .....	7
Step 5 – Save Project Files .....	8
Step 6 – Prepare the Environment .....	8
Step 7 – Update and Code Java Module .....	8
Step 8 – Save Project Files .....	8
Step 9 – Execute Scripts, Verify, Debug and Fix .....	8
Step 10 – Integrate Project Components .....	9
Step 11 – Create Integration Test Script .....	9
Step 12 – Execute Scripts, Verify, Debug and Fix .....	10
Step 13 – Perform Regression Testing .....	10
Step 14 – Reflect on Key Learning Points .....	11
Technical Specifications .....	12
Purpose .....	12
Project Description .....	12
Functionalities .....	12
Architectural Overview .....	13
Database Tables .....	13
Calling Sequence .....	14
Page Description .....	14
Page Layout .....	15
Systems Integration .....	17
Testing .....	18

## Activity Instructions

### Overview

This segment is divided into three parts. The first part involves modifying one of the web pages you coded in a previous segment as well as adding a new web page to accommodate a new requirement for international roaming service activation. You will code and test this front-end separately.

The second part involves coding the modules that will run at the back-end. You will modify existing code to accommodate the new data requirements of the additional feature. You will code and test this back-end separately.

The last part of the segment involves integrating these modules and testing them together. You will also be asked to create a test script that you will use in integration testing. If in the course of integration, you changed your code that you unit tested, you will then perform regression testing on these updated codes.

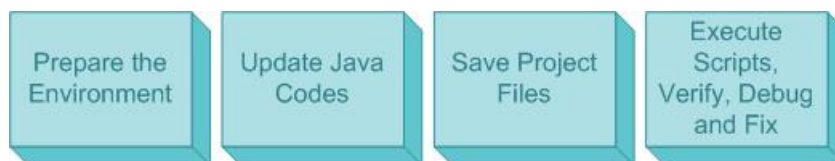
This three-part activity is designed for you to experience systems integration in a small scale, as well as multiple levels of testing as specified in the company's SDLC.

### Activity Steps

#### Part 1: Front-end



#### Part 2: Back-end



#### Part 3: Testing (Integration, Regression Testing & Test Plan)



To complete the activity you need to follow the steps above. These steps are explained later in greater detail in this section of the binder.

## Key Learning Points

At the end of the module, you need to learn and understand:

- Integration and regression testing concepts
- How to develop a test script
- Version control for application development

## Learning Materials

The following information may be found within this binder, the Reference Guide or the Performance Support Tool. Because you have used the materials in the previous segments you should be familiar with what the resources contain and how they will be used. This section describes the purpose and location of each document.

### Input

What	Why	Where
Activity Instructions	Guides you through the process of completing the activity	Activity Instructions section in this binder
Technical Specifications	Used as a blueprint for coding java logic	Technical Specifications section in this binder

### Reference Material

What	Why	Where
Linux Overview and Linux Glossary	Provides information on Linux commands and “how to” connect to a Linux server	Reference Guide
Java Overview and Java Glossary	Provides information on Java syntax	Reference Guide
Java Packaged Classes	Provides information on packaged classes relevant to the case	c:\appdevschool\references\
SQL Overview and SQL Syntax Reference	Provides information on SQL syntax	Reference Guide
Levels of Testing	Provides information on Unit Testing procedures and documentation	SDLC in the Performance Support Tool
Frequently Asked Questions	Provides answers to questions found throughout the Activity Instructions	Frequently Asked Questions in the Performance Support Tool

## Outputs

What	Why	Where
Updated subscriber_info.jsp	Additional controls to enable international roaming activation and display the results of successful activation.	Server Web Host
activate_roaming.jsp	Data entry form used to set effectivity date of service activation	Server Web Host
Updated SubscriberBean, Subscriber and SubscriberManager		Linux Server

## Step 1 – Prepare for this Activity

1. Review the information contained in each tab of the Activity Binder.

Briefly review the Activity Overview. Browse the sections of the technical specifications. Get an overview of the information they contain. You will use these sections extensively when you are coding and debugging so it is helpful to gain a general understanding of these materials.

2. Mark your Reference Guide and Performance Support Tool.

Throughout this binder you will encounter questions inside text boxes. These are relevant to the tasks at hand. You will find the answers to these questions from the either the Reference Guide or Performance Support Tool provided to you.

3. Confirm your understanding of the application logic.

To confirm your understanding of the application logic, try to answer the questions below:

- a. What table/s does the roaming activation affect? Should affect?
- b. What java classes and web pages will be affected by this change in requirement? By how much?

## Step 2 – Prepare the Environment

1. Launch Eclipse. You will use the same web project you used in Segment 3.

*Note: Segment 7 just modifies Segment 3. If you have an incomplete Segment 3 you will not be able to finish Segment 7. In segment 3 you used the web project from Segment 2 and created a new java project to code the underlying Java objects.*

### Step 3 – Update and Code JSP Pages

You will now code the JSP file. To perform this step, refer to the Calling Sequence, Page Description and Page Layout section in the Technical Specifications tab.

1. Update subscriber\_info.jsp.
  2. Create activate\_roaming.jsp.
- 

### Step 4 – Execute Scripts, Verify, Debug and Fix

To perform this step, refer to the Unit Testing tab of this segment binder.

1. Execute steps for each cycle of your Unit Test Script and verify your actual results against the expected results.
2. If you find discrepancies, analyze your code to determine the cause of the error. Correct the error and rerun your test scripts to retest your code.
3. Create a Unit Test Report Form using the template provided at the `c:\appdevschool\reports` folder.
4. Save the file as `Segment 7a - Unit Test Report.doc` on the same folder.
5. Click on the backup icon at `c:\appdevschool\apps` so that the instructor can check your work.

## Step 5 – Save Project Files

After you have completed the coding and debugging of your JSP files save the changes

---

## Step 6 – Prepare the Environment

1. You will now modify the underlying components of the website. This is similar to what you did in Segment 3. Open a java project to modify, save and compile the components for Segment 7.
- 

## Step 7 – Update and Code Java Module

You are now ready to code your Java modules. To perform this step, refer to the Project Description, Functionalities, and Calling Sequence sections in the Technical Specifications tab for the application logic.

Similar to the previous segment, you will use the vi text editor provided by Linux to write your java code. Refer to the Linux Glossary in the Reference Guide for vi commands you will use.

---

## Step 8 – Save Project Files

1. Save your project files.
- 

## Step 9 – Execute Scripts, Verify, Debug and Fix

1. Run the website.
2. If there are errors in your code, you will receive an error message in the following general format in your console:  

```
<filename>:<line number>: error message
```
3. Edit your code in the editor, save your changes and compile.
4. Should you perform a successful/"clean" compile, i.e. syntax and runtime errors were not found in your code, you will not receive a message.
5. Save your changes.
6. Execute steps for each cycle of your Unit Test Script and verify your actual results against the expected results.
7. If you find discrepancies, analyze your code to determine the cause of the error. Correct the error and rerun your test scripts to retest your code.
8. Commit your changes to the CVS.

9. Create a Unit Test Report Form using the template provided at the `c:\appdevschool\reports` folder.
10. Save the file as `Segment 7b - Unit Test Report.doc` on the same folder.
11. Click on the backup icon at `c:\appdevschool\apps` so that the instructor can check your work.

---

## Step 10 – Integrate Project Components

Your instructor will give you the name of the user whose Java codes you will be integrating with the JSP pages that you made.

1. Replace your working components with the ones from your assigned teammate.
2. Build your project and run it in the Tomcat server.
3. Depending on the extent that you and/or your partner followed the technical specifications, you may or may not encounter compile/runtime errors at this time.
4. If you do, debug the code and make note of the changes that you have made.

---

## Step 11 – Create Integration Test Script

To perform this Step you must refer to the SDLC Testing Levels section of the Performance Support Tool.

▪ [How do you write scripts from the specification document?](#)

1. Create an Integration Test Script using the template provided at the `c:\appdevschool\reports` folder.
2. Save the file as `Segment 7c - Integration Test Script.doc` on the same folder.
3. Click on the backup icon at `c:\appdevschool\apps` so that the instructor can check your work.

---

## Step 12 – Execute Scripts, Verify, Debug and Fix

Your instructor will give you a name of the user with whom you will exchange your scripts with.

▪ What are the qualities of a good test script?

1. Use your partner's script and test your integrated project. Verify your actual results against the expected results.
2. If you find discrepancies, analyze your code to determine the cause of the error. Correct the error and rerun your test scripts to retest your code.
3. Commit your changes to the CVS.
4. Create a Unit Test Report Form using the template provided at the `c:\appdevschool\reports` folder.
5. Save the file as `Segment 7c - Integration Test Report.doc` on the same folder.
6. Click on the backup icon at `c:\appdevschool\apps` so that the instructor can check your work.

---

## Step 13 – Perform Regression Testing

▪ What is Regression Testing?

To perform this step, refer to the Unit Testing tabs of this segment binder.

Because of the changes that you made to the code, you must perform regression testing to ensure that adjustments that you made did not result in errors for the individual components of the project (back-end and front-end).

1. Execute steps for each cycle of your Unit Test Script for both the front-end and back-end and verify your actual results against the expected results.
2. If you find discrepancies, analyze your code to determine the cause of the error. Correct the error and rerun your test scripts to retest your code.
3. Commit your changes.
4. Create a Unit Test Report Form using the template provided at the `c:\appdevschool\reports` folder.
5. Save the file as `Segment 7c - Unit Test Report.doc` on the same folder.
6. Submit your work to your instructor.

## Step 14 – Reflect on Key Learning Points

Inform your instructor that you have completed the coding and testing activities of this segment. He/she will review your work.

Review the key learning points for this segment. Answer the following questions and discuss them with your co-participants and faculty.

### General

1. What did you learn from this segment?
2. What was difficult about this assignment?
3. What will you do differently if you had to repeat this activity?

### Specific

1. What difficulties did you encounter in writing your test scripts?
2. What difficulties did you encounter in working with another person's code?
3. What strategies did you use to work with these difficulties?

---

## Technical Specifications

---

### Purpose

This document details the processing logic for this project. Sufficient information is provided to write the code. This technical specifications document is a hybrid of sections derived from the full-length Functional Specifications and DB Design Documents. In the future (depending on the scope of your project), you may encounter shorter or longer versions of these documents.

▪ [What are functional specifications? Why do we need it?](#)

---

### Project Description

The subscriber usage inquiry project has been expanded to include the international roaming activation request. It will be used in the following customer service interaction:

1. The customer calls up the customer service hotline and gives his/her account number to the CSR.
2. The CSR inputs the account number into the system. If entry is successful, will be shown the account information of the subscriber for verifying the caller's identity.
3. Once the caller's identity has been verified manually, the CSR can activate the international roaming service for the user given a valid date.
4. If successful, it will update the subscriber info table in the database.

*Note: For simplification purposes, recording of this activation as a billable transaction will not be included in this project.*

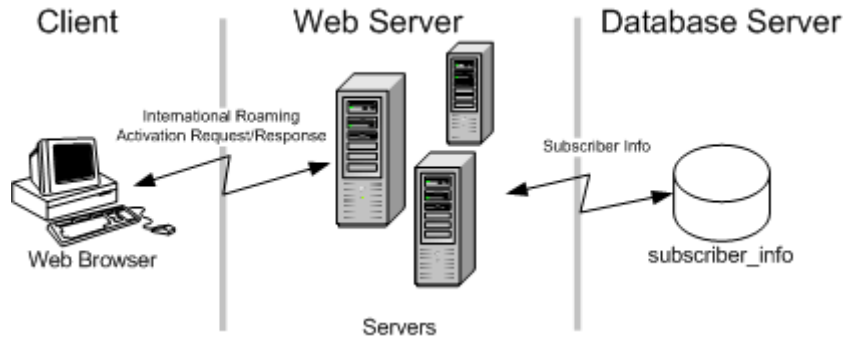
---

### Functionalities

- When the user enters the subscriber information page, the subscriber information should appear with a button to call the "Activate Roaming page."
- When the user clicks on this button, the "Activate Roaming" page should appear, where the user can enter the effectivity date of the roaming activation.
- The system must check whether the date entered by the user is:
  - not null or empty;
  - a valid date (i.e. days of the month must not exceed 31, months must not exceed 12, year should not be 0000), and;
  - more than or equal to the current date.
- The system should update the subscriber\_info table.

## Architectural Overview

The IR activation module that you are developing is a web application. The diagram below illustrates the system components involved in the proposed system architecture.



## Database Tables

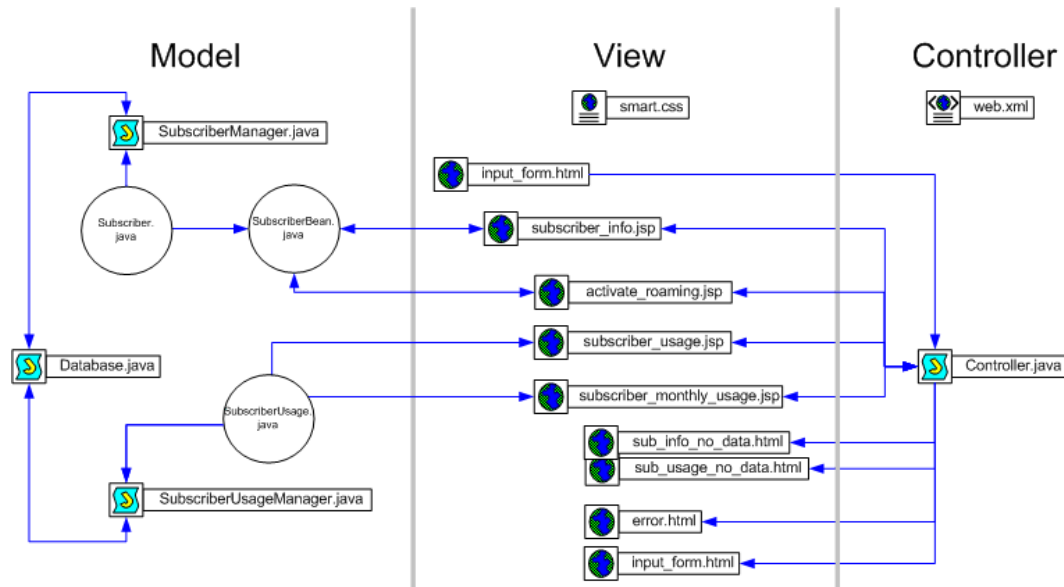
This project involves adding columns to the following table and updating the contents of the columns highlighted.

### SUBSCRIBER\_INFO

SI_IMSI	SI_NAME	SI_MSISDN	SI_IMSI	SI_ACC_NUM	SI_BIRTH_DAY	SI_MAIDEN_NAME	SI_IR_FLAG	SI_IR_DATE
Primary Key	varchar 30	char 20	char 20	char 20	varchar2 14	varchar 20	char 1	varchar2 14

## Calling Sequence

The diagram below illustrates how the new module interacts with other existing web pages.



## Page Description

### subscriber\_info.jsp

This page should display the following relevant subscriber information: *subscriber name, MSISDN, IMSI, account number, birthday, mother's maiden name, IR activation flag, and IR effectivity date.*

From this page the user can click on:

- *Change Account* button to return to the input form page and enter another subscriber account.
- *Get Subscriber Usage* button to show usage history year-to-date
- *Get Subscriber Monthly Usage* button to show usage history for the current month in a vertical table.
- *Enable IR* to navigate to the activate roaming page. If the roaming feature is already activated for the user, then this should be disabled.

The page should follow a top-to-bottom-left-to right tab sequence. Images should not be part of the tabbing sequence.

The title bar should display "Subscriber Information"

## activate\_roaming.jsp

This page should display the following relevant subscriber information: *subscriber name, MSISDN, IMSI, account number, birthday, mother's maiden name, IR activation flag, and IR effectivity date* in a horizontal table.

There should be four controls in page that are used to activate international roaming:

- *IR\_Flag* drop-down list box so that the user can choose from two options to allow international roaming: Yes or No
- *IR\_eDate* input box where the user can enter the effectivity date of the IR activation.
- *Activate* button to verify that the contents of the *IR\_eDate* is a valid date, submit the contents of the *IR\_Flag* and *IR\_eDate* to the subscriberManager class that will update the database, and bring the user back to the subscriber info page.
- *Cancel* button to clear the contents of the *IR\_Flag* drop-down list box and the *IR\_eDate* text box and bring the user back to the subscriber Info page.

The page should follow a top-to-bottom-left-to right tab sequence. Images should not be part of the tabbing sequence.

The title bar should display “Activate International Roaming”

---

## Page Layout

### subscriber\_info.jsp (regular view)



Subscriber Name:	John Garcia
MSISDN:	12345678901234567890
IMSI:	12345678901234567890
Account Number:	12345678901234567890
Birthday:	07/01/1978
Mother's Maiden Name:	Macapagal
International Roaming (IR) Activated:	N
IR Effectivity Date:	

Input Form    Get Subscriber Usage    Get Subscriber Monthly Usage    Enable IR

© 2005 SMART Communications, Inc. All Rights Reserved.    Home | Sitemap | Contact Us

## activate\_roaming.jsp



*Simply Amazing!*

Subscriber Name:	John Garcia
MSISDN:	12345678901234567890
IMSI:	12345678901234567890
Account Number:	12345678901234567890
Birthday:	07/01/1978
Mother's Maiden Name:	Macapagal
IR flag:	N
IR Effectivity Date:	

Allow Int'l Roaming:

Yes ▾

Effectivity Date:

Please enter the date values in the following format:  
(where "X" is any character except 0-9)

- ddmmyy (231205)
- ddmmyyyy (23122005)
- ddXmmXyy (23-12-05 or 23a12b05 ...)
- ddXmmXyyyy (23.12.2005 or 23/12/2005 ...)

Activate

Cancel

## subscriber\_info.jsp (IR success)



*Simply Amazing!*

### International Roaming Successfully Activated!

Subscriber Name:	John Garcia
MSISDN:	12345678901234567890
IMSI:	12345678901234567890
Account Number:	12345678901234567890
Birthday:	07/01/1978
Mother's Maiden Name:	Macapagal
International Roaming (IR) Activated:	Y
IR Effectivity Date:	23.12.05

Input Form

Get Subscriber Usage

Get Subscriber Monthly Usage

Enable IR

## Systems Integration

The JSP pages uses the external files, functions and/or procedures listed below to accomplish the requirements defined.

<b>External File, Function or Procedure</b>	<b>Description or Purpose</b>	<b>Type</b>
smart.css	Defines the formatting styles of the elements of the company website. also creates classes that can be applied to elements	Cascading Style Sheet
controller.java	Marshals the web page sequence based on values passed to it	Servlet
SubscriberBean.java	Returns subscriber information by invoking its public methods	Servlet
Subscriber.java	Returns a text-delimited subscriber information	Servlet
Database.java	Returns subscriber information by invoking its public methods	Servlet

## Testing

Replace this page with the Unit Test Script and Unit Test Data documents.