PRODUCTS AND SERVICES REFERENCED HEREIN INCLUDING WITHOUT LIMITATION ANY OF THE FOLLOWING DAMAGES: DIRECT, CONSEQUENTIAL, EXEMPLARY, INCIDENTAL, INDIRECT, SPECIAL, PUNITIVE, OR AGGRAVATED DAMAGES, DAMAGES FOR LOSS OF PROFITS OR REVENUES, FAILURE TO REALIZE ANY EXPECTED SAVINGS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, LOSS OF BUSINESS OPPORTUNITY, OR CORRUPTION OR LOSS OF DATA, FAILURES TO TRANSMIT OR RECEIVE ANY DATA, PROBLEMS ASSOCIATED WITH ANY APPLICATIONS USED IN CONJUNCTION WITH RIM PRODUCTS OR SERVICES, DOWNTIME COSTS, LOSS OF THE USE OF RIM PRODUCTS OR SERVICES OR ANY PORTION THEREOF OR OF ANY AIRTIME SERVICES, COST OF SUBSTITUTE GOODS, COSTS OF COVER, FACILITIES OR SERVICES, COST OF CAPITAL, OR OTHER SIMILAR PECUNIARY LOSSES, WHETHER OR NOT SUCH DAMAGES WERE FORESEEN OR UNFORESEEN, AND EVEN IF RIM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW IN YOUR JURISDICTION, RIM SHALL HAVE NO OTHER OBLIGATION, DUTY, OR LIABILITY WHATSOEVER IN CONTRACT, TORT, OR OTHERWISE TO YOU INCLUDING ANY LIABILITY FOR NEGLIGENCE OR STRICT LIABILITY.

THE LIMITATIONS, EXCLUSIONS, AND DISCLAIMERS HEREIN SHALL APPLY: (A) IRRESPECTIVE OF THE NATURE OF THE CAUSE OF ACTION, DEMAND, OR ACTION BY YOU INCLUDING BUT NOT LIMITED TO BREACH OF CONTRACT, NEGLIGENCE, TORT, STRICT LIABILITY OR ANY OTHER LEGAL THEORY AND SHALL SURVIVE A FUNDAMENTAL BREACH OR BREACHES OR THE FAILURE OF THE ESSENTIAL PURPOSE OF THIS AGREEMENT OR OF ANY REMEDY CONTAINED HEREIN; AND (B) TO RIM AND ITS AFFILIATED COMPANIES, THEIR SUCCESSORS, ASSIGNS, AGENTS, SUPPLIERS (INCLUDING AIRTIME SERVICE PROVIDERS), AUTHORIZED RIM DISTRIBUTORS (ALSO INCLUDING AIRTIME SERVICE PROVIDERS) AND THEIR RESPECTIVE DIRECTORS, EMPLOYEES AND INDEPENDENT CONTRACTORS.

IN ADDITION TO THE LIMITATIONS AND EXCLUSIONS SET OUT ABOVE, IN NO EVENT SHALL ANY DIRECTOR, EMPLOYEE, AGENT, DISTRIBUTOR, SUPPLIER, INDEPENDENT CONTRACTOR OF RIM OR ANY AFFILIATES OF RIM HAVE ANY LIABILITY ARISING FROM OR RELATED TO THE DOCUMENTATION.

Prior to subscribing for, installing or using any Third Party Products and Services it is your responsibility to ensure that your airtime service provider has agreed to support all of their features. Some airtime service providers may not offer Internet browsing functionality with a subscription to BlackBerry® Internet Service. Check with your service provider for availability, roaming arrangements, service plans and features. Installation or use of Third Party Products and Services with RIM’s products and services may require one or more patent, trademark, copyright or other licenses in order to avoid infringement or violation of third party rights. You are solely responsible for determining whether to use, Third Party Products and Services and if any third party licenses are required to do so. If required you are responsible for acquiring them. You should not install or use Third Party Products and Services until all necessary licenses have been acquired. Any Third Party Products and Services that are provided with RIM’s products and services are provided as a convenience to you and are provided “AS IS” with no express or implied conditions, endorsements, guarantees, representations or warranties of any kind by RIM and RIM assumes no liability whatsoever, in relation thereto. Your use of Third Party Products and Services shall be governed by and subject to you agreeing to the terms of separate licenses and other agreements applicable thereto with third parties, except to the extent expressly covered by a license or other agreement with RIM.

Certain features outlined in this documentation require a minimum version of BlackBerry® Enterprise Server software, BlackBerry® Desktop Software, and/or BlackBerry® Device Software and may require additional development or Third Party Products and Services for access to corporate applications.

This product includes software developed by the Apache Software Foundation (http://www.apache.org/) and/or licensed pursuant to Apache License, Version 2.0 (http://www.apache.org/licenses/). For more information, see the NOTICE.txt file included with the software. Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

The terms of use of any RIM product or service are set out in a separate license or other agreement with RIM applicable thereto. NOTHING IN THIS DOCUMENTATION IS INTENDED TO SUPERSEDE ANY EXPRESS WRITTEN AGREEMENTS OR WARRANTIES PROVIDED BY RIM FOR PORTIONS OF ANY RIM PRODUCT OR SERVICE OTHER THAN THIS DOCUMENTATION.
Chapter 6: Creating widgets

This chapter describes how to create a widget that uses PIM information from a BlackBerry® smartphone and how to create a widget that uses GPS location information.
Chapter 6: Creating widgets

6.1 Create a widget that uses PIM information

In this exercise, you will create a BlackBerry widget that displays all contacts that have a work address, and that displays hyperlinks that show contact information when you click them.

1. Open the config.xml file for editing. It matches the following example:

```xml
<?xml version="1.0" encoding="utf-8" ?>
<widget xmlns="http://www.w3.org/ns/widgets"
 xmlns:rim="http://www.blackberry.com/ns/widgets"
 version="1.0.0">
  <name> </name>
  <description> </description>
  <author href="http://www.rim.com/"
    rim:copyright="no copyright">
    John Doe
  </author>
  <license href="http://www.license.com">This is a sample license</license>
  <content src="index.html" />
</widget>
```

2. Make the following changes to the config.xml file:
   
a. Change the value of the name parameter to Lab: PIM widget. This name appears when you view the application icon in the Downloads folder on the BlackBerry smartphone.
   
b. Change the description parameter to BlackBerry Widget PIM Lab. This description displays when the user views the Application listing in the BlackBerry Options folder on the Home screen.
   
c. Modify the author parameter by adding your name and email address. Use the following format:
   ```xml
   email = "johndoe@emailaddress.com">
    John Doe
   </author>
   
   d. Add feature parameters to provide the BlackBerry widget with access to methods and properties found in the blackberry.system and blackberry.ui.menu and blackberry.pim namespaces. Add these feature parameters after the <content src> parameter, using the following format:
   ```xml
   <feature id="blackberry.system" />
   <feature id="blackberry.ui.menu" />
   ```
Chapter 6: Creating widgets

<feature id="blackberry.ui.menu.MenuItem" />
<feature id="blackberry.pim.Address" />
<feature id="blackberry.pim.Contact" />

3. Open the index.html file for editing. It matches the following example:

Note: In the index.html file, there is an existing reference to an external JavaScript® file named pim.js.

```html
<html>
<head>

<meta name="HandheldFriendly" content="True" />
<script type="text/JavaScript" src="pim.js"></script>

</head>
<body>

BlackBerry Widget PIM Lab

</body>
</html>

a. Add the following JavaScript method as inline code. The method adds menu items to the BlackBerry widget application when it is called. Add the JavaScript method in the `<head>` parameter, using the following format:

```javascript
function addMenus () {

    //create the menu items

    var mi_sep = new blackberry.ui.menu.MenuItem(true, 1);

    var mi_display = new blackberry.ui.menu.MenuItem(false, 2, "Display Contacts", displayContactsWithAddress);

    var mi_create = new blackberry.ui.menu.MenuItem(false, 3, "Create Test Contacts", createTestContacts);

    // Add items to our menu

    blackberry.ui.menu.clearMenuItems();
}
```
Chapter 6: Creating widgets

```javascript
blackberry.ui.menu.addMenuItem(mi_sep);
blackberry.ui.menu.addMenuItem(mi_display);
blackberry.ui.menu.addMenuItem(mi_create);
}
</script>

b. Add an onload JavaScript event to the body of the HTML. When the BlackBerry widget loads the page, it will call the `addMenus()` method to create the application menu items.

```html
<html>
<head>
<meta name="HandheldFriendly" content="True" />
<script type="text/JavaScript" src="lab.js"></script>
<script type="text/JavaScript">
function addMenus () {
    //create the menu items
    var mi_sep = new blackberry.ui.menu.MenuItem(true, 1);
    var mi_create = new blackberry.ui.menu.MenuItem(false, 2, "Create Test Contacts", createTestContacts);
    // Add items to our menu
    blackberry.ui.menu.clearMenuItems();
    blackberry.ui.menu.addMenuItem(mi_sep);
    blackberry.ui.menu.addMenuItem(mi_create);
}
</script>
</head>
<body onload="addMenus();">
```
c. Save and close the `index.html` file.

4. Open the `lab.js` file for editing. It matches the following example:

```javascript
function createAddress(address1, address2, city, country, stateProvince, zipPostal) {
    // Create address object
    var newAddress = new blackberry.pim.Address();
    newAddress.address1 = address1;
    newAddress.address2 = address2;
    newAddress.city = city;
    newAddress.country = country;
    newAddress.stateProvince = stateProvince;
    newAddress.zipPostal = zipPostal;
    return newAddress;
}

function createContact(first, last, address) {
    // Create the contact
    var contact = new blackberry.pim.Contact();
    contact.firstName = first;
    contact.lastName = last;
    contact.workAddress = address;
    contact.save();
}
```
function createTestContacts() {
  // load the contacts list with 3 test contacts:
  createContact("Jane", "Doe", null)
}

  a. Modify the createTestContact function to create an additional contact with a mailing address.

function createAddress(address1, address2, city, country, stateProvince, zipPostal) {
  // Create address object
  var newAddress = new blackberry.pim.Address();
  newAddress.address1 = address1;
  newAddress.address2 = address2;
  newAddress.city = city;
  newAddress.country = country;
  newAddress.stateProvince = stateProvince;
  newAddress.zipPostal = zipPostal;
  return newAddress;
}

function createContact(first, last, address) {
  // Create the contact
  var contact = new blackberry.pim.Contact();
  contact.firstName = first;
  contact.lastName = last;
  contact.workAddress = address;
contact.save();

}

function createTestContacts() {

  // load the contacts list with 3 test contacts:
  createContact("John", "Doe", createAddress("295 Phillip Street", ",", "Waterloo", "Canada", "Ontario", "N2L 3W8");

  createContact("Jane", "Doe", null)
}

5. Build and open your BlackBerry widget.
   a. Create a file named lab.zip, containing the config.xml and lab.js and index.html files.

   b. Run the BlackBerry® Widget Packager utility to build the BlackBerry widget application. This process generates a bin/lab.cod file in the Lab folder.
c. Open the BlackBerry Smartphone Simulator and load the **COD** file for the BlackBerry widget you just created.
d. Open the **Applications** menu item from the **Options** folder on the BlackBerry Smartphone Simulator Home screen, and confirm that the details for the Lab: PIM Widget application reflects the changes you made to the **config.xml** file.

![Applications menu](image1)

```plaintext
BlackBerry 5.0.0 Core Application 5.0.0
Research In Motion
lab (Lab: PIM Widget) 1.0.0
John Doe
```

e. Open the **Lab: PIM Widget** application from the **Downloads** folder.

![Downloads folder](image2)
f. Open the menu to see the menu item that was added when the `addMenus()` method was called during the `onload` event. Select this menu option.

g. Open the Contacts application and confirm the following two entries exist. These contacts were created by the Lab: PIM Widget application.
6. Open the lab.js file for edit.

   a. Add the following JavaScript method, named `displayContactsWithAddress`. This method searches the contacts database for any items that have either a home or work address.

   ```javascript
   function displayContactsWithAddress() {
     var feA = new blackberry.find.FilterExpression("homeAddress.address1", "!=","");
     var feB = new blackberry.find.FilterExpression("workAddress.address1", "!=","");
     var filter = new blackberry.find.FilterExpression(feA, "OR", feB);
     var contacts = blackberry.pim.Contact.find(filter); // should be an array of contacts.
     var field = parent.document.getElementById("contacts");
     for (i = 0; i < contacts.length; i++) {
       var uid = contacts[i].uid;
   ```
var name = contacts[i].firstName + " " + contacts[i].lastName;

field.innerHTML = field.innerHTML + "<a href="#" onclick='displayContact(" + uid + ");'">" + name + "</a> <br/>
" + uid + ");'">" + name + "<br/>
";

b. Add the following JavaScript method, named displayContact. This method retrieves a contact from the Contacts application and displays the name and address within the page element selectedContact.

function displayContact(uid) {
    var filter = new blackberry.find.FilterExpression("uid", 
        
    var contacts = blackberry.pim.Contact.find(filter); //shouldbe an array of contacts.

    contact = contacts[0];

    var txt = "";

    txt = txt + "Name:" + contact.firstName;
    txt = txt + " " + contact.lastName + "<br/>";
    txt = txt + "Address: " + contact.homeAddress.city;
    txt = txt + " " + contact.homeAddress.stateProvince;

    var field = parent.document.getElementById("selectedContact");
    field.innerHTML = txt;
}

c. Save and close the lab.js file.
7. Open the config.xml file for editing. Because the newly added JavaScript references new Web API namespaces, you must provide the BlackBerry widget with permission to access methods or properties in these namespaces.

   a. Add the following feature elements to the config.xml file:
   
   <feature id="blackberry.find.FilterExpression" />
   <feature id="blackberry.pim.Contact.find" />

8. Open the index.html file for editing and add the following HTML code. This code provides a hyperlink that, when clicked, invokes the displayContactsWithAddress method. This method updates the contents of the selectedContact div element.

   ...

   <body onload="initialLoad();">

     <h3><a href="#" onclick="displayContactsWithAddress()">Contacts with saved addresses</a></h3>

     <div id="selectedContact" />

   </body>

   </html>

9. Rebuild and open the BlackBerry widget. Click the Contacts with saved Addresses link and confirm that the results you see are reflective of the changes you made to the createTestContacts method in the lab.js file. The contacts displayed should be only those from the Contacts application that have an address.
10. Click the contact name and confirm that the address information for the contact appears. This information is retrieved from the contacts database.
6.2 Create a widget that uses GPS location information

In this exercise, you will create a BlackBerry widget that provides a list of addresses; when a user selects an address, the widget will display their location in the BlackBerry® maps application.

1. Open the config.xml file for editing. It matches the following example:

```xml
<?xml version="1.0" encoding="utf-8" ?>
<widget xmlns="http://www.w3.org/ns/widgets"
    xmlns:rim="http://www.blackberry.com/ns/widgets"
    version="1.0.0">
    <name> </name>
    <description> </description>
    <author href="http://www.rim.com/"
        rim:copyright="no copyright">
        John Doe
    </author>
    <license href="http://www.license.com">This is a sample license</license>
    <content src="index.html" />
</widget>
```

2. Make the following changes to the config.xml file:

   a. Change the value of the name parameter to Lab: GPS widget. This name appears when you view the application icon in the Downloads folder on the BlackBerry smartphone.

   b. Modify the description parameter. This description appears when you view the Application listing in the BlackBerry Options.

   c. Modify the author parameter by adding your name and email address. Use the following format:

   ```xml
   email = "johndoe@emailaddress.com">
   John Doe
   ```

   d. Save and close the config.xml file.

3. Open the index.html file for editing. It matches the following example:

   ```html
   In this exercise, you will create a BlackBerry widget that provides a list of addresses; when a user selects an address, the widget will display their location in the BlackBerry® maps application.
   ```

   **Note:** In the index.html file, there is an existing reference to an external JavaScript file named pim.js.
a. Add the following option buttons to the page. When the user clicks the option buttons, a JavaScript method named `showLocation` is invoked. It displays the selected location in the BlackBerry Maps application.

```html
<html>
<head>
  <meta name="HandheldFriendly" content="True" />
  <script type="text/JavaScript" src="lab.js"></script>
  <script type="text/JavaScript"
src="http://code.google.com/apis/gears/gears_init.js"></script>
</head>
<body>
  BlackBerry Widget GPS Lab
  
  <input type="radio" name="location" onclick="showLocation('1')" />
  RIM Head Office
</body>
</html>
```
<input type="radio" name="location" onclick="showLocation('2')"/>

CN Tower <br/>

<input type="radio" name="location" onclick="showLocation('3')"/>

My Location <br/>

</body>

</html>

b. Save and close the index.html file.

4. Build and open your BlackBerry widget.

a. Create a file named lab.zip, containing the config.xml, index.html and lab.js files.

b. Run the BlackBerry Widget Packager utility to build the BlackBerry widget application. This process generates a bin/lab.cod file in the Lab folder.
c. Open the BlackBerry Smartphone Simulator and load the COD file for the BlackBerry widget that you created.

d. Open the Applications menu item from the Options menu, and confirm that the details for the Lab: GPS Widget application reflects the changes you made to the config.xml file.
e. Open the Lab: GPS Widget application from the **Downloads** folder.
5. Open the lab.js file for editing.

   a. Add the following JavaScript method named `mapLocation`. This method launches the BlackBerry Maps application and passes it GPS coordinates. The map of the given location is opened.

   ```javascript
   function mapLocation(lat, lon)
   {
     var args = new blackberry.invoke.MapsArguments(lat, lon);
     blackberry.invoke.invoke(blackberry.invoke.APP_MAPS, args);
   }
   ```

   b. Add the following two JavaScript methods: `createAddress` and `mapAddress`. The first method creates a `blackberry.contact.address` object and the second method launches the BlackBerry Maps application and passes it the address information. The map of the given location is opened.

   ```javascript
   function createAddress(address1, address2, city, country, stateProvince, zipPostal) {
     // Create address object
   }
   ```

   f. Confirm that the application loads correctly and displays a list of radio-button locations.
var newAddress = new blackberry.pim.Address();
newAddress.address1 = address1;
newAddress.address2 = address2;
newAddress.city = city;
newAddress.country = country;
newAddress.stateProvince = stateProvince;
newAddress.zipPostal = zipPostal;
return newAddress;
}

function mapAddress(address1, address2, city, country, stateProvince, zipPostal)
{
    var address = createAddress(address1, address2, city, country, stateProvince, zipPostal);
    var args = new blackberry.invoke.MapsArguments(address);
    blackberry.invoke.invoke(blackberry.invoke.APP_MAPS, args);
}

c. Add the following JavaScript method, named showLocation. This method executes when the user clicks one of the option buttons within the index.html page.

function showLocation(num)
{
    var inum = parseInt(num);
    switch(inum) {
        case 1:
            //RIM Head Office - Waterloo, Canada
            mapAddress('295 Phillip Street', '', 'Waterloo',}
Chapter 6: Creating widgets

'dCanada', 'ON', 'N2L3W8');

break;
case 2:
    //CN Tower - Toronto, Canada
    mapAddress('301 Front Street West', '', 'Toronto',
'dCanada', 'ON', 'M5V2T6');

break;
case 3:
    //Current location of BlackBerry Smartphone
    geoLocateMe();

break;
default:
    alert('Unknown address:' + inum);
}
}

d. Add the following JavaScript methods, named geoLocateMe, displayLocation and geoError. These methods uses the geolocation API from Gears to retrieve and display the current location of the BlackBerry smartphone.

function displayLocation(loc) {
    mapLocation(loc.latitude, loc.longitude);
}

function geoError(err) {
    alert(err.message);
}

function geoLocateMe() {
    var geo = google.gears.factory.create('beta.geolocation');
geo.getCurrentPosition(displayLocation, geoError);

}  
ed. Save and close the lab.js file.

6. Open the config.xml file for editing. Because the newly added JavaScript references methods within the BlackBerry widget API, you need to provide the BlackBerry widget with permission to access methods or properties in these namespaces.

   a. Add a feature element to provide the BlackBerry widget with access to methods and properties found in the blackberry.invoke and blackberry.invoke.MapsArguments and blackberry.pim.Address namespaces.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<widget xmlns="http://www.w3.org/ns/widgets"
   xmlns:rim="http://www.blackberry.com/ns/widgets"
   version="1.0.0">
  <name>Lab: PIM Widget</name>
  <description>BlackBerry Widget PIM Lab</description>
  <author href="http://www.rim.com/"
    rim:copyright="no copyright"
    email = "johndoe@emailaddress.com">
    John Doe
  </author>
  <license href="http://www.license.com">This is a sample license</license>
  <content src="index.html" />
  <feature id="blackberry.invoke" />
  <feature id="blackberry.invoke.MapsArguments" />
  <feature id="blackberry.pim.Address" />
</widget>
```
b. Add an access element to provide the BlackBerry widget with access to the Google® online code repository. The BlackBerry widget retrieves the `gears_init.js` file at runtime to make use of Gears APIs.

```
<?xml version="1.0" encoding="utf-8" ?>

<widget xmlns="http://www.w3.org/ns/widgets"

  xmlns:rim="http://www.blackberry.com/ns/widgets"

  version="1.0.0">

  <name>Lab: PIM Widget</name>

  <description>BlackBerry Widget PIM Lab</description>

  <author href="http://www.rim.com/

    rim:copyright="no copyright"

    email = "johndoe@emailaddress.com">

    John Doe

  </author>

  <license href="http://www.license.com">This is a sample license</license>

  <content src="index.html" />

  <feature id="blackberry.invoke" />

  <feature id="blackberry.invoke.MapsArguments" />

  <feature id="blackberry.pim.Address" />

  <access uri="http://code.google.com" subdomains="true"/>

</widget>
```

c. Save and close the config.xml file.
7. Rebuild and open the BlackBerry widget. Click the RIM Head Office option button.

The Maps application loads, displaying the RIM Head Office.
8. Click **Simulate** from the **BlackBerry Smartphone Simulator** menu, and then choose the **GPSLocation** submenu.

![BlackBerry 9530 Simulator](image)

9. Select an existing location from the drop-down list in the GPS location shortcut menu, or click the **Add** button to create your own location by entering custom GPS coordinates.

![Edit GPS Location](image)
d. Click the **Close** button of the GPS location shortcut menu to save your selection.

10. Return to the BlackBerry widget application, and then click the **My Location** option button. The Gears geolocation API retrieves the current GPS coordinates of the BlackBerry smartphone running the BlackBerry's widget application and launches BlackBerry Maps to display this location.