

# BlackBerry MDS Studio

Version 4.1

## Getting Started Guide

BlackBerry MDS Studio Version 4.1 Getting Started Guide

Last modified: 22 March 2006

Part number: SWD\_MDS(EN)-006.000

At the time of publication, this documentation is based on BlackBerry MDS Studio Version 4.1.

Send us your comments on **product documentation**: <https://www.blackberry.com/DocsFeedback>.

© 2005 Research In Motion Limited. All Rights Reserved. The BlackBerry and RIM families of related marks, images and symbols are the exclusive properties of Research In Motion Limited. RIM, Research In Motion, 'Always On, Always Connected', the "envelope in motion" symbol and BlackBerry are registered with the U.S. Patent and Trademark Office and may be pending or registered in other countries.

Intel and Pentium are registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. Microsoft and Windows are registered trademarks of Microsoft Corporation in the United States and/or other countries. Sun, Java, and JavaScript are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. or other countries. All other brands, product names, company names, trademarks, and service marks are the properties of their respective owners.

The BlackBerry device and/or associated software are protected by copyright, international treaties and various patents, including one or more of the following U.S. patents: 6,278,442; 6,271,605; 6,219,694; 6,075,470; 6,073,318; D445,428; D433,460; D416,256. Other patents are registered or pending in various countries around the world. Visit [www.rim.com/patents.shtml](http://www.rim.com/patents.shtml) for a current listing of applicable patents.

This document is provided "as is" and Research In Motion Limited and its affiliated companies ("RIM") assumes no responsibility for any typographical, technical or other inaccuracies in this document. RIM reserves the right to periodically change information that is contained in this document; however, RIM makes no commitment to provide any such changes, updates, enhancements or other additions to this document to you in a timely manner or at all. RIM MAKES NO REPRESENTATIONS, WARRANTIES, CONDITIONS OR COVENANTS, EITHER EXPRESS OR IMPLIED (INCLUDING WITHOUT LIMITATION, ANY EXPRESS OR IMPLIED WARRANTIES OR CONDITIONS OF FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, MERCHANTABILITY, DURABILITY, TITLE, OR RELATED TO THE PERFORMANCE OR NON-PERFORMANCE OF ANY SOFTWARE REFERENCED HEREIN OR PERFORMANCE OF ANY SERVICES REFERENCED HEREIN). IN CONNECTION WITH YOUR USE OF THIS DOCUMENTATION, NEITHER RIM NOR ITS AFFILIATED COMPANIES AND THEIR RESPECTIVE DIRECTORS, OFFICERS, EMPLOYEES OR CONSULTANTS SHALL BE LIABLE TO YOU FOR ANY DAMAGES WHATSOEVER BE THEY DIRECT, ECONOMIC, COMMERCIAL, SPECIAL, CONSEQUENTIAL, INCIDENTAL, EXEMPLARY OR INDIRECT DAMAGES, EVEN IF RIM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, INCLUDING WITHOUT LIMITATION, LOSS OF BUSINESS REVENUE OR EARNINGS, LOST DATA, DAMAGES CAUSED BY DELAYS, LOST PROFITS, OR A FAILURE TO REALIZE EXPECTED SAVINGS.

This document might contain references to third party sources of information, hardware or software and/or third party web sites ("Third-Party Information"). RIM does not control, and is not responsible for, any Third-Party Information, including, without limitation the content, accuracy, copyright compliance, compatibility, performance, trustworthiness, legality, decency, links, or any other aspect of Third-Party Information. The inclusion of Third-Party Information in this document does not imply endorsement by RIM of the third party in any way. Any dealings with third parties, including, without limitation, compliance with applicable licenses and terms and conditions, are solely between you and the third party. RIM shall not be responsible or liable for any part of such dealings.

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.

Research In Motion Limited  
295 Phillip Street  
Waterloo, ON N2L 3W8  
Canada

Research In Motion UK Limited  
Centrum House, 36 Station Road  
Egham, Surrey TW20 9LF  
United Kingdom

Published in Canada



# Contents

|   |          |
|---|----------|
| <b>Getting started with BlackBerry MDS Studio .....</b>   | <b>1</b> |
| BlackBerry MDS Studio and web service applications .....  | 1        |
| Application development lifecycle.....  | 1        |
| Creating applications with application components.....  | 1        |
| Testing an application .....  | 2        |
| Publishing an application.....  | 3        |
| An example of screen and data component use .....   | 3        |
| Part-order data model.....  | 3        |
| Using screen and data components to design a user interface for the part-order web service..... | 4        |
| Customers screen.....   | 4        |
| Customer Details screen .....   | 5        |
| Customer Address screen.....  | 5        |
| Customer Preferences screen.....  | 6        |
| Customer Order screen .....   | 6        |
| Part Info screen .....  | 7        |
| Add Part screen.....  | 7        |
| <br>  |          |
| <b>Getting started tutorials .....</b>  | <b>9</b> |
| Creating a length conversion application using the quick start approach .....                   | 9        |
| Create a BlackBerry MDS Studio project .....  | 9        |
| Customize the user interface on the main screen .....   | 10       |
| Test the project .....  | 10       |
| Publish the project.....  | 11       |
| Creating a length conversion application using the bottom-up approach.....                      | 11       |
| Create a BlackBerry MDS Studio project.....   | 12       |
| Design the main screen and user interface.....  | 12       |
| Create communication between the user interface and the length conversion web service .....     | 14       |
| Optimize wireless bandwidth using message binding.....  | 17       |
| Test the project .....  | 19       |
| Publish the project.....  | 19       |



# Getting started with BlackBerry MDS Studio

BlackBerry MDS Studio and web service applications  
Application development lifecycle

## BlackBerry MDS Studio and web service applications

The BlackBerry Mobile Data System (MDS) Studio is a visual drag-and-drop developer platform used to design BlackBerry applications. The applications created in BlackBerry MDS Studio center around making web services available to BlackBerry users. The scope of this guide focuses on the development of applications within BlackBerry MDS Studio. For a more comprehensive overview of BlackBerry MDS Studio and its integration into the BlackBerry wireless architecture, see the BlackBerry MDS Studio Features and Technical Overview.

## Application development lifecycle

The BlackBerry MDS Studio provides an environment to create, test, and publish BlackBerry applications. Using this guide, you will be exposed to these three stages in the application development cycle through step-by-step tutorials.

## Creating applications with application components

Each BlackBerry MDS application consists of a set of screen, data, and message components.

1. Use screen components to allow the user to interface with the web service.
2. Use data components to manage information received from the user interface or web service.
3. Use message components to relay information across the wireless network.

### Screens

Screens arrange user interface elements such as labels, buttons, drop-down menus, and text boxes. Create screen components to allow users to navigate and utilize web services.

### Data

Data is the intermediary layer between screens and message. Data components are classified as either keyed or keyless. Keyed data components exist in a data collection and are each identified by a unique primary field of the same type such as an ID number. Keyless data components cannot be managed in a data collection and must exist within a keyed data component, a message, a screen parameter, or a variable.

Data is stored in local variables, global variables, or screen parameters.

- Use local variables to temporarily store information on the current screen.

- Use global variables to store information that is available throughout the application.
- Use screen parameters to pass information to other screens or scripts.

## Messages

Create messages to transport data to and from the web service. Outbound messages send data to the web service while inbound messages send data to the application. To interact with the web service, match the data contained in a message to a web service operation. Matching message data to web service operations is known as binding.

## Mapping

Mapping directs information between application components. For example, to display information stored in a data component, map the data component to a screen component such as a text box. Mapping between a data and screen component enables the application to collect or display data. Mapping between a data and message component directs information between the web service and the application.

## Application workflow

Workflows define the actions performed by buttons and menu selections. Use buttons and menu selections transition between screens, run scripts, or send data to the web service.

## Selecting an application design approach

The BlackBerry MDS Studio provides three application design approaches.

1. The Quick Start method creates the screens, messages, and data components and the relationships between the components. Use this approach to test a data source or examine a fully functional application.
2. The Bottom-up method creates the messages and data components and the relationships between the two. Use this approach for full control of the user interface design while still automating message and data components.
3. The Top-down method has no pre-generated components. Use this approach for a highly customized application.

The tutorials in this guide use the Quick Start and Bottom-up approach to design a BlackBerry MDS length conversion application. Using these tutorials you learn to apply the previous concepts to application development in the BlackBerry MDS Studio.

## Testing an application

The BlackBerry MDS Studio testing environment provides you with the ability to evaluate the functionality of your application via a computer-simulated BlackBerry device. Load your application into the device and use the mouse and keyboard to simulate the BlackBerry buttons.

## Publishing an application

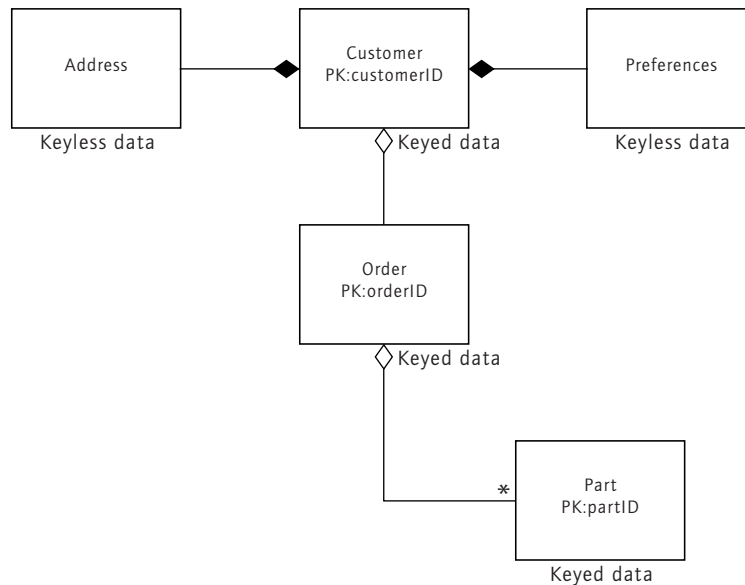
To make a BlackBerry MDS Studio application available to BlackBerry users, it must be published. Publishing an application places a description of the application in a registry and deposits the files in a repository. The registry is a list comprising the applications available to BlackBerry users and the repository is the collection of application files listed in the registry. To install applications on the BlackBerry device, users browse the registry and select applications. These applications are sent wirelessly to the device from the repository and installed.

## An example of screen and data component use

The following example uses a part-order application to show how screens dynamically interact with data components to display information to the user. This application enables the user to access customer information and add parts to a pre-existing order.

### Part-order data model

A data model shows the relationships between data components in an application. The part-order example uses the following data model:



Data components

This model shows the data components related to one instance of the Customer data component. Customer is a keyed data component because it is identified by a primary key, specifically a customer ID number. Keyed data components exist in a set called a data collection. Each component in a data collection has a unique primary key value. Having Customer as a data collection compartmentalizes the data components for customer address, preferences, orders, and parts. This way, each customer's information is contained into a single module and can be referred to by an ID number. For similar reasons, the Order and Part data components are data collections contained within the Customer data collection: a customer can have multiple orders containing multiple parts per order. Orders can only be referred to by accessing the Customer data component first and Parts can only be referred to by accessing the Order data component.

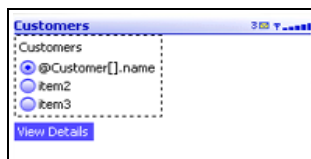
## Using screen and data components to design a user interface for the part-order web service

The part-order interface has a screen to

1. Select a customer
2. View individual customer details
3. View customer preferences
4. View customer address
5. View orders
6. View parts
7. Add parts

### Customers screen

On the Customers screen, the user views and selects from a list of customers. Using the View Details button, the user can view the details of the selected customer.



Customers screen

The names of the Customers are stored in the Name field of the Customer data collection. To display the Customer names in the selection list, map the Name field to the selection list. This allows the selection list to extract and display the Name data when the user enters the Customers screen.

> To map the Name field to the selection list, set the initial value of the list to `Customer[] . Name`.

To view the details of the selected customer, the data component of the customer must be stored in a local variable.

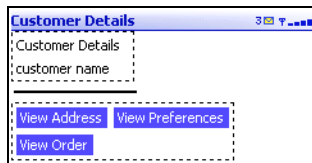
> To store the customer data component in a local variable, map the selection list to the local variable `selectedCustomer { type: Customer }`.

When the user clicks the View Details button, the application passes the selected Customer instance to the Customer Details screen as a screen parameter.

| Data type             | Data expression                                  | Description   |
|-----------------------|--|---|
| Data collection used  | <code>Customer [ ] . Name</code>                 | name of customer                                    |
| Screen local variable | <code>selectedCustomer { type: Customer }</code> | selected value is passed to Customer Details screen |

## Customer Details screen

The Customer Details screen receives the data component Customer as a screen parameter. To view the details of the selected customer, assign screen controls to the fields of the `selectedCustomer` parameter. For example, set the initial value of an editBox to `selectedCustomer . Name` to display the customer's name on the screen.



Customer Details screen

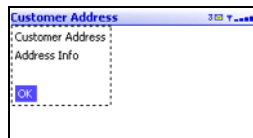
| Data type        | Data expression               | Description       |
|------------------|-------------------------------|-------------------|
| Screen parameter | <code>selectedCustomer</code> | selected customer |

From the Customer Details screen, the user can navigate to the Customer Address, Preferences, or Order screen using the buttons. The Customer Details screen then passes the corresponding data component as a screen parameter to the appropriate screen.

| Screen actions    | Description   |
|-------------------|---|
| View address.     | Navigate to the Customer Address screen passing <code>selectedCustomer . address</code> as a parameter.     |
| View preferences. | Navigate to the Customer Address screen passing <code>selectedCustomer . preferences</code> as a parameter. |
| View order.       | Navigate to Customer Address screen passing <code>selectedCustomer . order</code> as a parameter.           |

## Customer Address screen

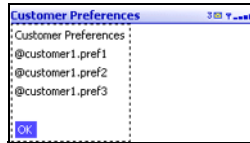
The Customer Address screen displays the address for the selected customer: `selectedCustomer . address`. The parameter of this screen is the Address data component, nested in the Customer data instance selected from the Customers screen. The user can view the address and return to the Customer Details screen.



Customer Address screen

## Customer Preferences screen

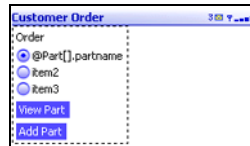
The Customer Preferences screen displays the preferences for the selected customer: `selectedCustomer.userprefs`. The parameter of this screen is the Preferences data component, nested in the Customer data instance selected in the Customers screen. The user can view the preferences and return to the Customer Details screen.



Customer Preferences screen

## Customer Order screen

The Customer Order screen displays a customer's order information and a list of parts. On this screen, the user selects a part from the list. After selection, the details of the part can be viewed or the part can be added to the order.



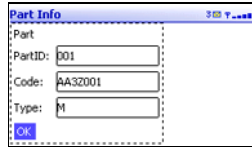
Customer Order screen

Both the Part Info and Add Part screen require the data component of the selected part. To make this data component available to those screens, map the selection list to the `selectedPart` local variable. This mapping causes the screen control to store the information to the local variable. When the user navigates to another screen using the View Part and Add Part buttons, the `selectedPart` local variable is passed as a screen parameter to the respective screen.

| Data type             | Data expression                          | Description  |
|-----------------------|--|--|
| Data collection used  | <code>order.part[] . id</code>           | The array of part data components is nested inside the Order instance. Each part has a unique ID number. The ID number field is the primary key of the part data collection. |
| Screen parameter      | <code>selectedCustomer.order</code>      | The Order data component originates from the Customer Details screen and the application passes it as a screen parameter to this screen.                                     |
| Local screen variable | <code>selectedPart { type: Part }</code> | Map the selection list to the local variable <code>selectedPart { type: Part }</code> .  |

## Part Info screen

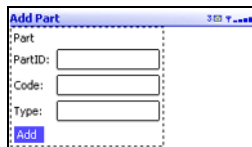
The Part Info screen displays the information for the part data passed from the Customer Order screen: `selectedPart { type: Part }`. The parameter of this screen is the Part data component, selected from the list of parts embedded in the Order data instance. The user can view the part information and return to the Customer Order screen.



Part Info screen

## Add Part screen

The Add Part screen enables the user to add a part to the existing customer order.



Add Part screen

| Data type             | Data expression   | Description   |
|-----------------------|---|---|
| Screen parameter      | <code>order { type: Order }</code>  | The parameter of this screen is the Order data component to which the part is added.  |
| Screen local variable | <code>newPart { type: Part }</code>   | Typing information in the text box screen controls creates an instance of the <code>newPart</code> local variable. The contents of the text boxes are assigned to the data fields of <code>newPart</code> . The first edit control, <code>PartID</code> (the primary key), is mapped to the local variable. |
| Script parameters     | <code>order { type: Order },<br/>newPart { type: Part }</code>  | The parameters of the script are the Order data component and part.   |
| Script                | <pre>order.parts.add(newPart); msgPartRequest.part = newPart; msgPartRequest.Send(); scr_Order.display();</pre> | The Add Part screen invokes a script to process the part request from a part-order web service. The script sends an application message to the web service that contains information about the requested part and then displays the Order Details screen.   |



# Getting started tutorials

Creating a length conversion application using the quick start approach  
Creating a length conversion application using the bottom-up approach

## Creating a length conversion application using the quick start approach

In this section, you create a length conversion application that works as a request-response application: the user requests a conversion from one length unit to another unit and the web service responds with the converted value. With the quick start approach, the BlackBerry MDS Studio creates and maps the screen, data, and message components. It also binds messages to the web service and creates screen local and global variables. In this section, you complete the following tasks:

- create a BlackBerry MDS Studio project
- customize the user interface on the main screen
- publish the application

### Create a BlackBerry MDS Studio project

Using the **Create New Project** wizard, you select the quick start design approach and length conversion web service

1. In the BlackBerry MDS Studio, on the **File** menu, click **New Project**.
2. Select the **Quick Start Approach Wizard** option.
3. Click **Next**.
4. Click **Browse**.
5. Click the **MDS4.1 Sample Web Services** tab.
6. In the web services list, click the **Convert** web service
7. Click **OK**.
8. Click **Next**.
9. In the **Binding Information** tree, clear the **convertTemperature**, **convertWeight**, and **convertData** check boxes.



**Note:** The BlackBerry MDS Studio analyzes the Web Service Description Language (WSDL) and shows the available operations. The wizard generates an application structure that matches the web service definition and the selected operations. By default, all operations are selected in the Binding Information list of operations. In this example, you need the `convertLength` operation only.

10. Click **Next**.
11. In the **Project Name** field, type **QuickConversion**.
12. Click **Finish**.



**Tip:** View the newly created BlackBerry MDS Studio project in the Navigator pane.

## Customize the user interface on the main screen

Using the Screen Editor palette and the Properties pane, you can design and customize a user interface.

1. In the Navigator pane, expand the **Screens** tree.
2. Double-click **scr\_Main**.

The main screen appears in the Screen Editor. The Screen Editor is a visual editor.

### Change the screen title

1. In the Screen Editor pane, click **Main Screen**.
2. In the Properties pane, in the **Title** property, click ....
3. In the edit field, change the title of the screen to **Convert Length**.
4. Click **OK**.

### Change the button text

1. In the Screen Editor, click **Go**.
2. In the Properties, in the **Caption** property, click ....
3. In the edit field, change the button text to **Convert**.
4. Click **OK**.

### Save the project

1. In the Navigator pane, click the **QuickConversion** project.
2. On the **File** menu, click **Save**.




## Test the project

Testing the application installs it to the BlackBerry device simulator.

1. In the BlackBerry MDS Studio, in the Navigator pane, right-click the **QuickConversion** folder.
2. Click **Test**.
3. In the BlackBerry device simulator, on the Home screen, click the BlackBerry MDS Application icon.

## Publish the project

Publishing the application places a description of the application on the registry list and deposits the application file into a repository. Users of BlackBerry devices that have the BlackBerry MDS Runtime can view and download a published application.

1. In the Navigator pane, click the **QuickConversion** folder.
2. On the toolbar, click **Publish Wizard** .
3. In the **Description** field, type **sample bottom-up application**.
4. Click **Next**.
5. In the **File Repository** drop-down list, click the location in which to store the application files.
  -  **Tip:** When you select a file repository, the File Repository icon turns green to indicate that the BlackBerry MDS Studio can deposit the application bundle (.jar file) to the selected repository location. If the icon is red or yellow, the BlackBerry MDS Studio cannot access the repository location. Select another location, or verify the user name and password in the BlackBerry MDS Studio > Options > Preferences.
6. In the **Application Registry** drop-down list, click a BlackBerry MDS Application registry to publish the description of this application to.
  -  **Tip:** When you select an application registry, the Application Registry icon turns green to indicate the registry is available. If the icon is red or yellow, the BlackBerry MDS Studio cannot access the registry. Select another registry.
7. Click **Next**.
8. Select the **Deployment Keywords** check box.
9. In the **Keyword** field, type **conversion**.
10. Click **Add**.
11. Click **Finish**.

The BlackBerry MDS Studio saves the application files to the selected BlackBerry MDS Application repository and publishes the application description to the selected BlackBerry MDS Application registry. You can now download the BlackBerry MDS Unit Conversion application to a BlackBerry device.

## Creating a length conversion application using the bottom-up approach

In this section, you create a length conversion application using the bottom-up approach. For a description of the application, see "Creating a length conversion application using the quick start approach" on page 3. Applying the bottom-up approach instead of quick start allows you the flexibility of designing and creating the user interface without having to create the underlying data and message structure. At the end of this section, comparing the quick start and bottom-up length conversion application will enable you to distinguish between the two approaches and select the most appropriate one for your next application.

In this section, you complete the following tasks:

- create a BlackBerry MDS Studio project

- design the main screen and user interface
- create communication between the user interface and the length conversion web service
- optimize wireless bandwidth using message binding
- test the project
- publish the project

## Create a BlackBerry MDS Studio project

1. In the BlackBerry MDS Studio, on the **File** menu, click **New Project**.
2. Select the **Bottom Up Approach Wizard** option.
3. Click **Next**.
4. Click **Browse**.
5. Click the **MDS4.1 Sample Web Services** tab.
6. In the web services list, click the **Convert** web service.
7. Click **OK**.
8. In the **Binding Information** tree, clear the **convertTemperature**, **convertWeight**, and **convertData** check boxes.
9. Click **Next**.
10. In the **Project name** field, type **LengthConverter**.
11. Click **Finish**.

The BlackBerry MDS Studio generates the application based on the web service definition. The bottom-up wizard creates an application, including the data and message components. The BlackBerry MDS Studio binds messages to the data source. Once generated, the datasource visualizer and data view open.

## Design the main screen and user interface

The BlackBerry MDS Studio generates the main screen, which is the default entry point into your application. Use the Screen Editor to create controls such as labels, lists, and text boxes.



**Tip:** To design the user interface in full screen mode, from the toolbar, click the **Change Screen Editor Skin** drop down list. Select **Fullscreen**.

The screenshot shows a user interface for a length converter. It consists of two main sections enclosed in dashed boxes. The top section contains three labels: 'From:', 'To:', and 'Value:'. Each label is followed by an input field. The 'From:' and 'To:' fields are dropdown menus with '@LengthUnit' selected. The 'Value:' field is a text input. Below this section is a horizontal line. The bottom section contains a 'Result:' label followed by a text input containing '@result.toValue'. Below the entire interface is a blue 'Convert' button.

Final LengthConverter Interface

## Remove the default region

1. In the Navigator pane, expand **Screens**.
2. Double-click **scr\_Main**.
3. In the Outline pane, click **region\_Main**.
4. Press DELETE.

## Create a new region to group the screen controls

1. In the Screen Editor pane, on the Screen Editor palette, click **Region**.
2. Place the region on the screen canvas.
3. In the Properties pane, in the **Layout** drop down list, click **Grid**.

## Add labels to the region

1. In the Screen Editor, on the Screen Editor palette, click **Label**.
2. Place the **Label** in the region.
3. In the Properties pane, in the **Caption** property, click ...
4. In the edit field, type **From:**.
5. Click **OK**.
6. Add two more labels to the right of the previous label with the following captions:
  - To:
  - Value:

## Add drop-down lists

1. On the Screen Editor palette, click **Choice**.
2. Place the choice control to the right of the **From** label.
3. Expand **Enumerations**.

4. Click **LengthUnit**.
5. Click **Apply**.
6. Click **OK**.
7. In the Properties pane, in the **Appearance** drop-down list, click **dropdown**.
8. To refresh the choice control, click the screen.
9. To the right of the **To** label, add another choice control. Repeat steps 3 to 8.

### Add a text box to enter length value

1. Right of the **Value** label, create an **EditBox** control.
2. In the Properties pane, in the **Entry type** drop-down list, click **number**.

### Add a button

1. Under the region, create a button control.
2. In the Workflow Wizard, in the **Button Caption** edit box, enter **Convert**.
3. Click **Define later**.
4. Click **Finish**.

### Add a separator

- > Under the region, create a **Separator** control.

### Add a region

- > Under the separator, create a **region** control.

### Add a response label

- > In the region, create a label control with the Caption property set to **Result**.

### Add a text box to display the converted length

1. Right of the **Result** label, create an edit box control.
2. In the Properties pane, in the **Entry Type** drop-down list, click **number**.
3. In the Properties pane, in the **Read Only** drop-down list, click **true**.

## Create communication between the user interface and the length conversion web service

To transmit data to the web service, create a screen local variable and map data to it for storage. Data selected or entered from the drop-down lists and text box can be mapped to this local variable before being sent as a message to the web service. A request message bound to the web service and containing the data from the local variable enables the conversion web service.

1. Click the screen canvas.
2. In the Properties pane, in the **Local Variables** property, click ....
3. In the Local Variables dialog box, click +.



**Note:** The BlackBerry MDS Studio proposes the creation of the local variable called **lengthParameters** that can store the initial and final length unit and the initial value.

4. Click **OK**.

### Assign controls to the lengthParameter local variable

1. On the screen canvas, click the **choice1** control.
2. In the Properties pane, in the **Mapping** property, click ....
3. Expand **Local Variables**.
4. Expand **lengthParameters**.
5. Click **fromUnit**.
6. Click **OK**.
7. On the screen canvas, click the **choice2** control.
8. Repeat steps 2 through 4.
9. Click **toUnit**.
10. Click **OK**.
11. On the screen canvas, click the **editBox1** control.
12. Repeat steps 2 through 4.
13. Click **fromValue**.
14. Click **OK**.

### Set the Convert button to send data to the web service for conversion

Using the button, the user can activate the web service. When pressed, the BlackBerry MDS Application sends data on the main screen to the lengthParameters local variable. The convertLengthRequest message sends the local variable information to the web service. The conversion web service converts the length value.

1. Click the **Convert** button control.
2. In the **Wizards** menu, select **Workflow Wizard**.
3. Select the **Send a message** check box.
4. Click **Next**.
5. In the list of data types, expand **Local Variables**.
6. Click **lengthParameters**.
7. Click **Next**.
8. Click **convertLengthRequest**.

9. Click **Next**.
10. In the left pane, click **lengthParameters** to assign a screen parameter.
11. In the right pane, click **parameters** to assign a message parameter.
12. Click **Assign**.
13. Click **Finish**.



**Tip:** You can view the script the BlackBerry MDS Studio generates in **Navigator > Scripts**. Double-click **script\_Onscr\_Mainbutton1Click**. The script displays the syntax for sending a message to the web service when the user clicks the Convert button.

## Create a global variable to receive data from the web service

When the web service converts the length, it generates a response message containing the converted length data. The LengthConverter application stores this data in a global variable. To display the converted length to the user you map the Result edit box to the global variable.

1. In the Navigator pane, expand **Data**.
2. Right-click **Globals**.
3. Click **New Global**.
4. In the Create New Global dialog box, in the **Name** field, type **result**.
5. Click **Finish**.
6. In the Global Variable Editor, click the **Data Type** arrow.
7. Expand **data component**.
8. Click **Result**.
9. Click **OK**.

## Map the web service response message to the result global variable

1. In the Navigator pane, expand **Messages**.
2. Double-click the **convertLengthResponse** message.
3. In the Message Editor, click the **Fields** tab.
4. Click the first **Data Mapping** cell.
5. Click ....
6. In the Choose Data Mapping dialog box, select the **Is Mapped** check box.
7. Click the **result** global variable.
8. Click **OK**.



**Note:** The convertLengthResponse message and the result global variable are related. When the web service sends a response, the LengthConverter application stores the value in the result global variable. The relationship canvas displays the visual representation between the message and data.

## Map the result global variable to the edit box to display the converted length

1. In the Navigator pane, expand **Screens**.
2. Double-click **scr\_Main**.
3. On the screen canvas, click **editbox2**.
4. In the Properties pane, in the **Initial Value** property click ....
5. In the Initial Value dialog box, expand **Globals**.
6. Expand **result**.
7. Click **toValue**.
8. Click **Apply**.
9. Click **OK**.

## Refresh the main screen when receiving the response message

The LengthConverter application receives the converted length in the web service response message. To display the converted length, the main screen must refresh when the application receives the response message.

1. Click the screen canvas.
2. In the Properties pane, in the **Refresh Messages** property, click ....
3. Click **Add**.
4. Click **OK**.

## Optimize wireless bandwidth using message binding

In this section, you learn how to optimize your bottom-up application using the message binding wizard. The response message generated by the bottom-up wizard contains fields for fromUnit, fromValue, toUnit and toValue. Instead of including each field within the response message, only the final converted length toValue is needed to convey the response to the user. By binding less data to the response message, the web service uses less wireless bandwidth.

## Remove a field from the response message

1. In the Navigator pane, expand **Messages**.
2. Double-click **convertLengthResponse**.
3. Click the **Fields** tab.
4. Click the first Data Mapping cell.
5. Click ....
6. Clear the **Is Mapped** check box.
7. Click **OK**.

## Change the data type that the response message uses

1. In the **Type** cell, click ....
2. Click **decimal**.
3. Click **Next**.
4. Click **OK**.
5. From the relationship canvas, click the **Result** icon.
6. Press DELETE.



**Note:** Delete only the Result data component, not the result global variable.

## Assign the response message to the converted length value in the web service

1. In the Navigator pane, expand **Messages**.
2. Right-click **convertLengthResponse**.
3. Click **Message Binding Wizard**.
4. Click **Next**.
5. Click the **convertLengthResponse** output message.
6. Click **Next**.
7. In the list of BlackBerry MDS Application message fields in the left pane, click the **convertLengthReturn: decimal** field.
8. In the list of WSDL message fields in the right pane, click the **toValue: double** field.
9. Click **Bind Field**.
10. Click **Finish**.



**Note:** The length converter web service no longer incorporates the other WSDL message fields in the response message.

## Change the global variable to match the response message data type

1. In the Navigator pane, expand **Data > Globals**.
2. Double-click **result**.
3. Click the drop-down arrow beside the **Data Type** field.
4. Click **decimal**.
5. Click **OK**.



**Note:** After changing the global variable data type, controls and messages previously mapped to it must be reassigned.

## Assign the global variable to the edit box to display the converted length

1. In the Navigator pane, double-click **scr\_Main**.

2. Click the **Result** edit box.
3. In the Properties pane, in the **Initial Value** property, click ...
4. Expand **Globals**.
5. Click **result: decimal**.
6. Click **Apply**.
7. Click **OK**.

### Assign the response message data to the result global variable

1. In the Navigator pane, double-click the **convertLengthResponse** message.
2. In the Message Editor, click the **Fields** tab.
3. In the **Data Mapping** cell, click ...
4. Select the **Is Mapped** check box.
5. Click the **result: decimal** global variable.
6. Click **OK**.

### Remove the Result data component

1. In the Navigator pane, expand **Definitions**.
2. Right-click **Result**.
3. Click **Delete**.
4. Click **OK**.

### Save the project

1. In the Navigator pane, click the **QuickConversion** project.
2. On the toolbar, click **Save**.

### Test the project

See "Test the project" on page 10 for more information

### Publish the project

See "Publish the project" on page 11 for more information.







© 2005 Research In Motion Limited

Published in Canada.