

BlackBerry Plug-in for Microsoft Visual Studio

Version 1.1

Developer Guide

BlackBerry Plug-in for Microsoft Visual Studio Version 1.1 Developer Guide

Last modified: 14 October 2008

Part number: 15350540

At the time of publication, this documentation is based on BlackBerry Plug-in for Microsoft Visual Studio Version 1.1

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

©2008 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, BlackBerry, "Always On, Always Connected" and the "envelope in motion" symbol are registered with the U.S. Patent and Trademark Office and may be pending or registered in other countries.

iDEN is a trademark of Motorola, Inc. Intellisense, Microsoft, Vista, Visual Basic, Visual C#, Visual Studio, and Windows are trademarks of Microsoft Corporation. Java, JavaDoc, JavaScript, and NetBeans are trademarks of Sun Microsystems, Inc. Plazmic is a trademark of Plazmic Inc. All other trademarks are the property of their respective owners.

The BlackBerry smartphone and other devices 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 the U.S. and in various countries around the world. Visit www.rim.com/patents for a list of RIM (as hereinafter defined) patents.

This documentation including all documentation incorporated by reference herein such as documentation provided or made available at www.blackberry.com/go/docs is provided or made accessible "AS IS" and "AS AVAILABLE" and without condition, endorsement, guarantee, representation, or warranty of any kind by Research In Motion Limited and its affiliated companies ("RIM") and RIM assumes no responsibility for any typographical, technical, or other inaccuracies, errors, or omissions in this documentation. In order to protect RIM proprietary and confidential information and/or trade secrets, this documentation may describe some aspects of RIM technology in generalized terms. RIM reserves the right to periodically change information that is contained in this documentation; however, RIM makes no commitment to provide any such changes, updates, enhancements, or other additions to this documentation to you in a timely manner or at all.

This documentation might contain references to third-party sources of information, hardware or software, products or services including components and content such as content protected by copyright and/or third-party web sites (collectively the "Third Party Products and Services"). RIM does not control, and is not responsible for, any Third Party Products and Services including, without limitation the content, accuracy, copyright compliance, compatibility, performance, trustworthiness, legality, decency, links, or any other aspect of Third Party Products and Services. The inclusion of a reference to Third Party Products and Services in this documentation does not imply endorsement by RIM of the Third Party Products and Services or the third party in any way.

EXCEPT TO THE EXTENT SPECIFICALLY PROHIBITED BY APPLICABLE LAW IN YOUR JURISDICTION, ALL CONDITIONS, ENDORSEMENTS, GUARANTEES, REPRESENTATIONS, OR WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY CONDITIONS, ENDORSEMENTS, GUARANTEES, REPRESENTATIONS OR WARRANTIES OF DURABILITY, FITNESS FOR A PARTICULAR PURPOSE OR USE, MERCHANTABILITY, MERCHANTABILITY QUALITY, NON-INFRINGEMENT, SATISFACTORY QUALITY, OR TITLE, OR ARISING FROM A STATUTE OR CUSTOM OR A COURSE OF DEALING OR USAGE OF TRADE, OR RELATED TO THE DOCUMENTATION OR ITS USE, OR PERFORMANCE OR NON-PERFORMANCE OF ANY SOFTWARE, HARDWARE, SERVICE, OR ANY THIRD PARTY PRODUCTS AND SERVICES REFERENCED HEREIN, ARE HEREBY EXCLUDED. YOU MAY ALSO HAVE OTHER RIGHTS THAT VARY BY STATE OR PROVINCE. SOME JURISDICTIONS MAY NOT ALLOW THE EXCLUSION OR LIMITATION OF IMPLIED WARRANTIES AND CONDITIONS. TO THE EXTENT PERMITTED BY LAW, ANY IMPLIED WARRANTIES OR CONDITIONS RELATING TO THE DOCUMENTATION TO THE EXTENT THEY CANNOT BE EXCLUDED AS SET OUT ABOVE, BUT CAN BE LIMITED, ARE HEREBY LIMITED TO NINETY (90) DAYS FROM THE DATE YOU FIRST ACQUIRED THE DOCUMENTATION OR THE ITEM THAT IS THE SUBJECT OF THE CLAIM.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW IN YOUR JURISDICTION, IN NO EVENT SHALL RIM BE LIABLE FOR ANY TYPE OF DAMAGES RELATED TO THIS DOCUMENTATION OR ITS USE, OR PERFORMANCE OR NON-PERFORMANCE OF ANY SOFTWARE,

HARDWARE, SERVICE, OR ANY THIRD PARTY 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. 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.

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.

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

Research In Motion UK Limited
200 Bath Road
Slough, Berkshire SL1 3XE
United Kingdom

Published in Canada

Contents

1	Designing BlackBerry MDS Runtime Applications	5
	Creating BlackBerry user interfaces	5
	Coding in JavaScript using Intellisense.....	5
	Managing data and offline capability.....	6
	Handling messages when a BlackBerry device is outside of a wireless coverage area	6
	Message queue sizes.....	6
	Support for push events.....	7
	BlackBerry MDS Runtime application development cycle	7
2	Using web services.....	9
	Manage web references.....	9
	Set the web service URL.....	9
	Add a web reference	9
	Update a web reference definition	10
	Show web reference details	10
	Change discovered class properties	10
	Unsupported or partially supported WSDL types	11
	Simple type constraints	12
	Required attributes	13
	Union type	13
	Mixed content	14
	Occurrence constraints	14
	Choice group	15
	Choice group occurrences.....	15
	Extended type	15
	Restricted type.....	16
	Substitution group.....	17
	Abstract elements/types.....	17
	Unique element	18
	Key and Keyref.....	19
	Recursive definitions.....	20
	Nested input arrays (multi-dimensional).....	20
	Tool specific limitations and workarounds.....	21

Axis - AnyType type.....	21
SQLXML 3.0 - Mixed content	21
Sun Studio/NetBeans - AnyType Type.....	22
Microsoft Visual Studio .NET - AnyType type	22
Microsoft Visual Studio .NET - Timespan (empty complexType)	23
Microsoft Visual Studio .NET - Dynamic content	24
Create an instance of the web service.....	25
Invoke a web service method	25
Minimize data over the wireless network.....	25
Handle asynchronous messages	26
Invoke the web reference method	26
Handle the returned data.....	26
Retrieve data returned by the completed message	26
Handle SOAP faults.....	27
Limitation.....	27
3 Using push integration.....	29
Exposing functions as push functions	29
DeviceMethod code attribute.....	29
DeviceMethodOptions code attribute.....	30
RibbonAlert code attribute.....	31
DialogAlert code attribute	31
Adding a BlackBerry reference	32
Add a BlackBerry reference from a BlackBerry MDS Runtime Application project in this solution	32
Add a Blackberry reference from a BlackBerry MDS Runtime Application project outside this solution	32
Invoking a BlackBerry device method using Microsoft Visual C#.....	33
Public properties of the BlackBerry reference class	33
Specifying the recipients of the push message	33
4 BlackBerry MDS and JavaScript.....	35
Enhanced JavaScript comments.....	35
summary tag.....	35
param tag.....	35
returns tag	36
Arrays.....	36

Initialize and populate an array	37
Determine the length of an array	37
Add an element to an array.....	37
Remove the last element from an array	37
Remove an element from an array.....	38
Find an element in an array	38
Numbers.....	38
Convert the base of a number	38
Set the precision of a decimal number.....	38
Convert a String to number.....	38
Concatenate a String and a number.....	38
Variables.....	39
Example: Declaring variables	39
Example: Declaring global variables	39
GPS support.....	39
blackberry.location.....	39
blackberry .network.....	41
5 Developing BlackBerry MDS Runtime Applications	43
Managing BlackBerry MDS classes and data.....	43
Collections.....	43
Enumerations	45
Global Variables.....	46
Persistent and non-persistent data storage.....	48
Create an instance of a Class	48
Disable class auto-instantiation	48
Display collections and arrays of classes.....	49
Objects	49
Managing the user interface.....	50
Forms	50
Arranging form controls	55
Manage forms using script.....	55
Data bindings.....	59
Stylesheets	70
Adding hot keys	72
Managing communication.....	73
Change the message protocol.....	73

Managing runtime errors.....	74
Change the standard error script.....	74
Integrating existing BlackBerry applications	74
Adding menu items inside BlackBerry built-in applications.....	74
Add menu items to built-in applications.....	75
6 Testing BlackBerry MDS Runtime Applications.....	77
Configure break points.....	77
Start the debugger.....	77
Debug the application in the BlackBerry Browser.....	77
Output debug statements	78
7 Publishing BlackBerry MDS Runtime Applications.....	79
Publish a project	79
Set application registry preferences.....	80
Set application repository preferences.....	81
Change the version number of the application.....	82
Set the base URI for the application.....	82
Publish a project using the standalone publishing utility	83

Designing BlackBerry MDS Runtime Applications

The BlackBerry® Plug-in for Microsoft® Visual Studio® is designed to provide a methodology similar to other applications created with Microsoft Visual Studio.

Familiarity with Microsoft Visual Studio is an asset when designing BlackBerry® MDS Runtime Applications. The BlackBerry Plug-in makes use of the look, feel and general design approaches used by Microsoft Visual Studio .NET.

Each screen in the BlackBerry Plug-in is designed using a drag-and-drop approach to populate a form with form controls, such as buttons and text boxes. Transitions between forms can be triggered using the properties window in the UI or script.

Event handling is done by creating JavaScript® functions that are assigned to the control's events in the Properties window.

Creating BlackBerry user interfaces

The BlackBerry® Plug-in for Microsoft® Visual Studio® is designed to assemble applications from a set of form controls, classes and collections, and messages. The plug-in enables you to create instances of these components, customize their properties, and use them in the application using script and graphical user interfaces.

Classes and collections include built-in Personal Information Management and JavaScript® classes and collections, discovered classes and collections provided by web services, and user-defined classes and collections. Collections are merely containers for classes which are stored using a primary key and can be retrieved using built-in methods.

Forms are used to arrange form controls of a BlackBerry® MDS Runtime Application. Each control's events can have a script assigned to it. Button and menu item controls can also be used to trigger transitions to another form. Form controls also include labels, buttons, text boxes, masked text boxes, picture boxes, radio groups, check boxes, checked list boxes, list boxes, layout panels, date-time pickers, repeaters, and menu items.

Messages are used to communicate application data across the wireless network between the client application on the BlackBerry® device and the data source.

Coding in JavaScript using Intellisense

The BlackBerry® Plug-in for Microsoft® Visual Studio® uses JavaScript® for its scripting language. The plug-in also supports Microsoft® Intellisense®, a form of automatic code completion common to the Microsoft Visual Studio development environment.

When you type, Intellisense provides a list of available methods or attributes for the class or a list of parameters for the method. Not only does this minimize the amount of keystrokes required, it also limits the need to consult external documentation.

Managing data and offline capability

BlackBerry® MDS Runtime Applications are designed to store and process data locally on the BlackBerry® device, without the need for a continuously available network connection. The communication model used to send and receive application messages is fully asynchronous. This means that if an outbound message is generated at a time when the BlackBerry device is outside of a wireless coverage area, the message is placed into an outbound queue until the wireless connection becomes available. When the message is placed into this queue, the application continues functioning, regardless of whether the message has been sent. Messages are transmitted from the queue in the background when the network connection becomes available.

Similarly, inbound messages can be queued by the BlackBerry® MDS Services and transmitted to the BlackBerry device only when the wireless connection becomes available. BlackBerry MDS Runtime Applications are designed to process inbound messages as asynchronous events. The asynchronous communication model is designed for applications that are inherently tolerant to sudden or unpredictable loss of connection. This feature, combined with the ability to store and process data locally, enables BlackBerry MDS Runtime Applications to be designed in such a way that application functionality and enterprise data are available, even when wireless connections are not available.

In BlackBerry MDS Runtime Applications, the communication model consists of the following items:

- delivery mode for application messages
- handling messages when a BlackBerry device is outside of a wireless coverage area
- message queue sizes
- categories of notification messages
- size limitations for messages

Handling messages when a BlackBerry device is outside of a wireless coverage area

When a BlackBerry® device is outside of a wireless coverage area, outbound standard and reliable application messages remain queued on the BlackBerry device until the BlackBerry device returns to a wireless coverage area, at which time the messages are transmitted. Best-effort messages are dropped when a BlackBerry device is outside of a wireless coverage area.

Message queue sizes

BlackBerry® MDS Runtime Applications have restricted inbound and outbound message queue sizes. These restrictions are imposed by policies sent to the BlackBerry MDS Runtime software on the BlackBerry® device. The system administrator of the BlackBerry® MDS Services can push new policies to the BlackBerry MDS Runtime, which might change the limitations of an application's inbound and outbound queue sizes.

Support for push events

The push-based architecture that is used for BlackBerry® email is available for custom BlackBerry® MDS Runtime Applications. Using this push-based architecture, server-side applications can push new data, alerts and notifications to users proactively.

BlackBerry MDS Runtime application development cycle

BlackBerry® MDS Runtime Applications have the following development lifecycle:

Development steps	Description
Create application	Use the BlackBerry® Plug-in for Microsoft® Visual Studio® to develop applications. Use the development environment to create classes and collections and forms and assemble them into a BlackBerry MDS Runtime application.
Test and Debug the application	Use the debugger and BlackBerry® Smartphone Simulator to test and debug the BlackBerry MDS Runtime Application. The BlackBerry Smartphone Simulator allows you to simulate the application without loading the application on a physical BlackBerry® device.
Publish application	Use the publishing wizard within the BlackBerry Plug-in for Microsoft Visual Studio to deposit the application into the BlackBerry MDS Application repository and publish the application to the BlackBerry MDS Services application registry. The BlackBerry MDS Application repository is installed with the BlackBerry MDS Services component of the BlackBerry® Enterprise Server. The BlackBerry Enterprise Server administrator must provide publishing credentials.
Implement application	The BlackBerry Enterprise Server administrator can configure applications to be pushed to users or make the applications available for download. Once enabled, use the BlackBerry device with the BlackBerry MDS Runtime to search for and download published BlackBerry MDS Runtime applications from the BlackBerry Enterprise Server.
Run application	Use a BlackBerry device to run BlackBerry MDS Runtime applications and to begin to interact with enterprise applications.

Using web services

If you have a web reference in your solution, you can invoke the methods exposed by the web service by creating an instance of the web service and then invoking the method.

Manage web references

You can add a web service as a web reference to the solution. When you add a web reference, the methods and classes provided by the web service are available to the application.

Set the web service URL

To set the implementation URL of the web service, perform the following actions:

1. In the Solution Explorer, expand the **Web References** folder.
2. Navigate to the web service.
3. Click the web service.
4. In the Properties window, in the **Implementation URL** field, type the location of the web service file.
5. To set the password for authentication by the web service, in the Password field, type the password.
6. To set the username, in the Username field, type the username.

If you do not enter a username and password and authentication is required by the web service, a dialog box will prompt the user to enter a username and password at runtime when the web method is invoked.

Add a web reference

1. In the Solution Explorer, right-click on the BlackBerry® MDS Runtime project name.
2. Click **Add Web Reference**.
3. To find and add a web reference, perform one of the following actions:

Action	Procedure
Type the URL for the web service.	<ol style="list-style-type: none"> 1. In the URL field, type the URL. 2. Click Go.
Add a web service that is located in the solution.	> Click Web services in this solution .
Add a web service that is located on the local machine.	> Click Web services on the local machine .
Add a web service that is located on your network.	> Click Browse UDDI Servers on the local network .

4. Click on the web service you want to add. In the **Web reference name** field, specify a name. Click **Add reference**.

If this is the first web reference you add to the project, a **Web References** folder is created in the Solution Explorer.

Update a web reference definition


If a change is made to the web service that you are using for your web reference, you must perform the following to see the changes in your project:

1. In the Solution Explorer, expand the project.
2. Expand the **Web References** folder.
3. Right-click the web reference you wish to update.
4. Click **Update web reference**.

When you update a web reference, you may have to manually change your script to handle the changes to the web reference definition.

Show web reference details

In the Solution Explorer, expand the project. Expand the **Web References** folder. Expand the web reference. Perform any of the following actions:

Action	Procedure
Show the messages returned by the web service.	<ol style="list-style-type: none"> 1. Expand the web service. 2. To see the message's Completed message, expand the message.
Show the classes that are exposed by the web reference.	<ol style="list-style-type: none"> 1. Click the class. 2. In the Properties window, in the Fields field click the  button. 3. To see information about a specific member variable, in the Class Field Editor dialog, click the member variable.

Change discovered class properties

Discovered classes are classes returned by a web service.

To change the properties of a discovered class, perform the following action:

1. In the Solution Explorer, expand the **Web References** folder.
2. Navigate to the class you want to edit.
3. Click on the class.
4. In the Properties window, perform the following actions:

Action	Procedure
Change the class type.	<p>> In the Class Type field, specify a class type.</p> <p>If you specify Collection or PersistentCollection, you must specify a primary key in the Primary Key field.</p>
Specify the member fields that this class will return.	<p>You can choose which fields are and are not returned by the web service.</p> <p>See for detailed instructions.</p>

Action	Procedure
Specify the primary key.	> In the Primary Key field, specify the primary for this Collection.

Unsupported or partially supported WSDL types

The following table defines the binding between XML-Schema built-in types and BlackBerry® MDS Runtime data types.

XML-Schema built-in	BlackBerry MDS type	BlackBerry MDS array	Restrictions
anyType			
anySimpleType	string		
string	string		
normalizedString	string		A string that does not contain line feeds, carriage returns, or tabs.
token	string		A string that does not contain line feeds, carriage returns, tabs, leading or trailing spaces, or multiple spaces.
byte	integer		A signed 8-bit integer.
unsignedByte	integer		
base64Binary	binary		
hexBinary	binary		
integer	long		An integer value.
positiveInteger	long		An integer containing only positive values.
negativeInteger	long		An integer containing only negative values
nonNegativeInteger	long		An integer containing only non-negative values
nonPositiveInteger	long		An integer containing only non-positive values
int	integer		A signed 32-bit integer
unsignedInt	long		An unsigned 32-bit integer .
long	long		A signed 64-bit integer.
unsignedLong	long		An unsigned 64-bit integer.
short	integer		A signed 16-bit integer.
unsignedShort	integer		An unsigned 16-bit integer.
decimal	decimal		A decimal value.
float	decimal		Single precision 32-bit floating point value.
double	decimal		Double precision 64-bit floating point value.
boolean	boolean		
time	date		Time of day.
dateTime	date		
duration	long		Duration of time in years, months, days, hours, minutes, seconds.
date	date		A calendar date.
gMonth	date		A Gregorian month.
gYear	date		A Gregorian calendar year.

XML-Schema built-in	BlackBerry MDS type	BlackBerry MDS array	Restrictions
gYearMonth	date		A specific month in a specific Gregorian year.
gDay	date		A specific Gregorian day of the month.
gMonthDay	date		A specific day in a specific Gregorian month.
Name	string		A string that contains a valid XML name.
QName	string		A namespace qualified name.
NCName	string		A non-colonized name.
anyURI	string		A URI formatted string.
language	string		A string that contains a valid language ID.
ID	string		An XML ID.
IDREF	string		An XML ID reference.
IDREFS	string	true	A white space separated list of IDs.
ENTITY	string		An XML entity.
ENTITIES	string	true	A white space separated list of entities.
NOTATION	string		An XML notation.
NMTOKEN	string		An XML name token.
NMTOKENS	string	true	A white space separated list of XML name tokens.

Limitations

The BlackBerry MDS® Runtime does not have a base object type so anyType is not supported.

Workaround

Avoid the use of anyType.

Simple type constraints

An XML-Schema can impose further constraints on the acceptable values for XML elements and attributes. The following table defines BlackBerry® MDS Runtime support for these constraints.

Constraint	Description	Supported
enumeration	Defines a list of acceptable values.	Yes
fractionDigits	Specifies the maximum number of decimal places allowed.	No
length	Specifies the exact number of characters or list items allowed.	No
maxExclusive	Specifies the upper bounds for numeric values (the value must be less than this value).	No
maxInclusive	Specifies the upper bounds for numeric values (the value must be less than or equal to this value).	No
maxLength	Specifies the maximum number of characters or list items allowed.	No
minExclusive	Specifies the lower bounds for numeric values (the value must be greater than this value).	No
minInclusive	Specifies the lower bounds for numeric values (the value must be greater than or equal to this value).	No
minLength	Specifies the minimum number of characters or list items allowed.	No
pattern	Defines the exact sequence of characters that are acceptable.	No
totalDigits	Specifies the exact number of digits allowed.	No
whiteSpace	Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled.	No

Limitation

Sending data from a BlackBerry® MDS Runtime Application that does not conform to the constraints imposed by the XML-Schema data model might cause problems at runtime.

Workaround

You can restrict the values in your application messages by using screen control validation such as the EditText Entry Type and Format properties and script logic.

Required attributes

XML-Schema permits an attribute to be defined as a required data value.

```
<xs:attribute name="lang" type="xs:string" use="required"/>
```

Limitation

The BlackBerry® Mobile Data System data model does not support required fields.

Workaround

As a workaround you can use the mandatory property of certain screen controls to enforce the field's presence.

Union type

An XML-Schema union permits values from more than one type in the same data element.

Example: Union type

```
<xs:element name="zips" type="zipUnion"/>
<xs:simpleType name="zipUnion">
  <xs:union memberTypes="USState listOfMyIntType"/>
</xs:simpleType>
```

Example: Instance

```
<zips>CA</zips>
<zips>95630 95977 95945</zips>
<zips>AK</zips>
```

As illustrated by the example each instance is either of one type or the other.

Limitation

Union types are not supported.

Workaround

None.

Mixed content

XML-Schema allows a data type to have mixed content. Mixed content is character data intermingled with XML elements.

Example: Instance

```
<letter>
Dear Mr.<name>John Smith</name>.
Your order <orderid>1032</orderid>
will be shipped on <shipdate>2001-07-13</shipdate>.
</letter>
```

Example: Mixed content

```
<xs:element name="letter">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="orderid" type="xs:positiveInteger"/>
      <xs:element name="shipdate" type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Limitation

Mixed content is not supported.

Workaround

None.

Occurrence constraints

XML-Schema uses occurrence constraints to define how often an element can occur in a data type.

Example: Occurrence constraints

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="fullName" type="xs:string"/>
      <xs:element name="childName" type="xs:string"
        maxOccurs="10" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Limitation

The BlackBerry® MDS Runtime supports mapping occurrences of more than 1 to an array type. The exact number of occurrences is not enforced.

Workaround

If the minimum occurrence value is non-zero then the developer will have to enforce this. If the maximum occurrence value is not unbounded then the developer will have to enforce this.

Choice group

XML-Schema can constrain a type so that only one of its child elements may appear in an instance.

```
<xs:element name="person">
  <xs:complexType>
    <xs:choice>
      <xs:element name="employee" type="employee"/>
      <xs:element name="member" type="member"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Limitation

The BlackBerry® Mobile Data System data model supports a choice group by binding each child element to a data field. At runtime the BlackBerry® MDS Runtime Application should supply only one of the fields and set the others to null.

Workaround

The developer will have to provide logic to ensure that only one of the choice elements is populated with data.

Choice group occurrences

An XML-Schema choice may have occurrence constraints.

```
<xs:element name="person">
  <xs:complexType>
    <xs:choice maxOccurs="unbounded">
      <xs:element name="employee" type="employee"/>
      <xs:element name="member" type="member"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Limitation

The BlackBerry® Mobile Data System data model supports a choice group with occurrences by mapping each child to a data field array. At runtime the order of the choice elements is not preserved.

Workaround

If maintaining order is important, use a sequence group.

Extended type

The extension element is used in XML-Schema to extend an existing simple type or complex type.

```
<xs:complexType name="personInfo">
```

```
<xs:sequence>
  <xs:element name="firstName" type="xs:string"/>
  <xs:element name="lastName" type="xs:string"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="fullPersonInfo">
  <xs:complexContent>
    <xs:extension base="personInfo">
      <xs:sequence>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="country" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The BlackBerry® MDS Runtime maps each complex type to a data component and indicates inheritance by using the prototype mechanism.

Limitation

BlackBerry® MDS Services will shear subclass type at runtime for response messages. Because the BlackBerry® MDS Runtime does not support polymorphism it is not possible to send a subtype in place of a declared base type in a request message.

Workaround

None.

Restricted type

XML-Schema makes it possible to derive a new type by restricting the definition of an existing complex type. A complex type derived by restriction is very similar to its base type, except that its declarations are more limited than the corresponding declarations in the base type. The values of the new type are a subset of the values represented by the base type.

```
<xs:complexType name="customer">
  <xs:sequence>
    <xs:element name="firstName" type="xs:string"/>
    <xs:element name="lastName" type="xs:string"/>
    <xs:element name="country" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="NorwegianCustomer">
  <xs:complexContent>
    <xs:restriction base="customer">
      <xs:sequence>
        <xs:element name="firstName" type="xs:string"/>
        <xs:element name="lastName" type="xs:string"/>
        <xs:element name="country" type="xs:string" fixed="Norway"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

```
</xs:complexContent>
</xs:complexType>
```

Limitation

Restrictions of a base type are ignored by the BlackBerry® MDS Runtime.

Workaround

The developer can use screen controls or script to enforce the restrictions.

Substitution group

XML-Schema provides a mechanism called substitution groups that allow elements to be substituted for other elements. A substitution group must be the same type or derived from the same type it is a substitution for.

Example: Substitution

```
<xs:element name="comment" type="xs:string"/>
<xs:element name="customerComment" type="xs:string" substitutionGroup="comment"/>
```

Example: Instance

```
<item>
  <comment>Use gold wrap if possible</comment>
</item>
```

Example: Alternate Instance

```
<item>
  <customerComment>Want this for the holidays!</customerComment>
</item>
```

Limitation

Substitution groups are not supported.

Workaround

None.

Abstract elements/types

XML-Schema provides a mechanism to force substitution for a particular element or type. When an element or type is declared to be abstract, it cannot be used in an instance document. Declaring an element as abstract requires the use of a substitution group. Declaring a type as abstract requires the use of a type derived from it (and identified with an `xsi:type` attribute) in the instance document.

Example: Abstract type

```
<complexType name="Vehicle" abstract="true"/>

<complexType name="Car">
  <complexContent>
    <extension base="target:Vehicle"/>
  </complexContent>
</complexType>
```

```
</complexContent>
</complexType>

<complexType name="Plane">
  <complexContent>
    <extension base="target:Vehicle"/>
  </complexContent>
</complexType>

<element name="transport" type="target:Vehicle"/>
```

In the example above the vehicle type cannot be used in an instance document. One of the derived types must be used instead.

Example: Instance

```
<transport xsi:type="Car"/>
```

Limitation

Abstract Base classes will produce a warning in Microsoft® Visual Studio® .NET. Derived classes will be created correctly but there will be no class created for the base class.

Workaround

None.

Unique element

The unique element is used to specify that an element or an attribute value must be unique within a certain scope. The scope is an XPath expression that selects a set of elements within the document. The attributes or elements that should be unique within this scope are identified by another XPath expression relative to the selected elements.

Example: Instance

```
<purchaseReport>
  <regions>
    <zip code="95819"/>
    <zip code="63143"/>
  </regions>
</purchaseReport>
```

Unique constraint

```
<element name="purchaseReport">
  <unique name="dummy1">
    <selector xpath="regions/zip"/>
    <field xpath="@code"/>
  </unique>
</element>
```

The example above specifies that the code attribute values of the set of zip elements within the regions element must be unique.

Limitation

Unique constraints are ignored.

Workaround

Enforce uniqueness via primary key on data collection or through script.

Key and Keyref

XML-Schema also has a mechanism to ensure that a value in an instance document is referred to by another value. The mechanism identifies one value as a key and another as a key reference.

Example: Instance

```
<regions>
  <zip code="95819">
    <part number="872-AA" quantity="1"/>
    <part number="926-AA" quantity="1"/>
    <part number="833-AA" quantity="1"/>
    <part number="455-BX" quantity="1"/>
  </zip>

  <zip code="63143">
    <part number="455-BX" quantity="4"/>
  </zip>
</regions>

<parts>
  <part number="872-AA">Lawnmower</part>
  <part number="926-AA">Baby Monitor</part>
  <part number="833-AA">Lapis Necklace</part>
  <part number="455-BX">Sturdy Shelves</part>
</parts>
```

Key/key reference constraint:

```
<element name="regions" type="RegionsType">
  <keyref name="dummy2" refer="pNumKey">
    <selector xpath="zip/part"/>
    <field xpath="@number"/>
  </keyref>
</element>

<key name="pNumKey"> <selector xpath="parts/part"/>
  <field xpath="@number"/>
</key>
```

In the example above the part numbers in the zip elements refer to the part numbers in the parts element.

Limitation

The BlackBerry® Mobile Data System data model does not support foreign keys so key and keyref elements are ignored.

Workaround

Enforce the constraint through script.

Recursive definitions

XML-Schema permits an element or type definition to have recursive elements.

```
<xs:element name="BrowseNode">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="BrowseNodeId" type="xs:string" minOccurs="0"/>
      <xs:element name="Name" type="xs:string" minOccurs="0"/>
      <xs:element name="Children" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="tns:BrowseNode" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

The BrowseNode element has a Children element that contains a sequence of BrowseNodes.

Limitation

The BlackBerry® MDS Runtime does not support recursive definitions.

Workaround

None.

Nested input arrays (multi-dimensional)

Some WSDL operations will input an array of a type that also holds an array.

```
<operation name="addCustomers">
  <input message="tns:addCustomersRequest"/>
  <output message="tns:addCustomersResponse"/>
</operation>

<message name="addCustomersRequest">
  <part name="parameters" element="tns:addCustomers"/>
</message>

<xs:element name="addCustomers">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="customers" type="tns:Customer" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:complexType name="Customer">
  <xs:sequence>
```

```

    <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
    <xs:element name="addresses" type="tns:Address" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

```

In the example above the addCustomers element has a sequence of Customer types. Each Customer type has a sequence of Address types.

Tool specific limitations and workarounds

Axis - AnyType type

If the signature of a method returns an instance of type Object or a collection of type Object, Axis generates an element of XML-Schema anyType type.

```

<xs:element name="helloResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="helloReturn" type="xs:anyType"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

Limitation

The XML-Schema built-in type anyType is not supported.

Workaround

Use a more specific array type like String[].

```
public String[] hello() throws java.rmi.RemoteException;
```

SQLXML 3.0 - Mixed content

The SQLXML tool will generate elements with mixed content.

```

<xs:complexType name="SqlXml" mixed="true">
  <xs:sequence>
    <xs:any/>
  </xs:sequence>
</xs:complexType>

```

Limitation

Mixed content is not supported.

Workaround

Replace result set with objects of a specific type.

Sun Studio/NetBeans - AnyType Type

If the signature of a method returns an instance of type `Object` or a collection of type `Object`, Sun® Studio generates an element of XML-Schema `anyType` type.

```
<complexType name="collection">
  <complexContent>
    <restriction base="soap-enc:Array">
      <attribute ref="soap-enc:arrayType" wsdl:arrayType="anyType[]" />
    </restriction>
  </complexContent>
</complexType>
```

Limitation

The XML-Schema built-in type `anyType` is not supported.

Workaround

Use a specific type.

```
public String[] hello() throws java.rmi.RemoteException;
```

To generate this construct the `Strict` flag must be unchecked in the tool.

Microsoft Visual Studio .NET - AnyType type

If the signature for a web method returns an instance of type `object`, .NET generates an XML-Schema element with no type information.

```
<xs:element minOccurs="0" maxOccurs="1" name="helloResult"/>
```

If the signature for a web method returns an array of type `object` or an `ArrayList`, .NET generates an XML-Schema element with no type information.

```
<xs:complexType name="ArrayOfAnyType">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="anyType" nillable="true"/>
  </xs:sequence>
</xs:complexType>
```

Limitation

If not specified the type of an element is XML-Schema `anyType`. The XML-Schema built-in type `anyType` is not supported.

Workaround

Use a distinct type or serialize as a specific type.

```
[WebMethod]
public string hello()
{
}
```

or

```
[WebMethod]
[return: System.Xml.Serialization.XmlElementAttribute(typeof(string))]
public object hello()
```

```
{
}
```

Microsoft Visual Studio .NET - TimeSpan (empty complexType)

If the signature of a web method returns an object with no fields, .NET generates an empty complex type.

```
<xs:complexType name="Comp">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1" name="ts" type="tns:TimeSpan"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="TimeSpan"/>
```

Construct results from:

```
[WebMethod]
public Comp hello()
{
}

public class Comp
{
  public TimeSpan ts;
}
```

Limitation

The BlackBerry® MDS Runtime does not support binding to an empty complexType.

Workaround

Wrap TimeSpan with a string, and serialize as type duration.

```
using System.Runtime.Remoting.Metadata.W3cXsd2001;
public class Comp
{
  private TimeSpan ts;

  [System.Xml.Serialization.XmlElementAttribute(DataType="duration")]
  public string dur
  {
    get
    {
      return SoapDuration.ToString(ts);
    }
    set
    {
      ts = SoapDuration.Parse(value);
    }
  }
}
```

Microsoft Visual Studio .NET - Dynamic content

If the signature of an operation returns a DataSet, .NET will generate an element with a reference to the XML-Schema schema to define the content and then an XML-Schema any element to provide the dynamic content itself.

```
<xs:element minOccurs="0" maxOccurs="1" name="getContactsResult">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="xs:schema" />
      <xs:any />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Construct results from:

```
[WebMethod]
public DataSet getContacts(string connection)
{
    SqlConnection conn = new SqlConnection(connection);
    SqlDataAdapter adapter = new SqlDataAdapter();

    string query = "SELECT * FROM contacts";
    adapter.SelectCommand = new SqlCommand(query, conn);
    DataSet dataset = new DataSet();
    adapter.Fill(dataset);
    return dataset;
}
```

Limitation

Dynamic content is not supported. All data definitions must be predefined.

Workaround

Replace DataSet with your own class and populate manually.

```
[WebMethod]
[return: System.Xml.Serialization.XmlArrayItemAttribute(typeof(Contact))]
public ArrayList getContacts(string connection)
{
    string query = "SELECT * FROM contacts";
    SqlConnection conn = new SqlConnection(connection);
    SqlCommand cmd = new SqlCommand(query, conn);
    conn.Open();
    SqlDataReader reader;
    reader = cmd.ExecuteReader();

    ArrayList response = new ArrayList();
    while (reader.Read())
    {
        Contact contact = new Contact();
        contact.id = reader.GetInt32(0);
        contact.name = reader.GetString(1);
        contact.email = reader.GetString(2);
        response.Add(contact);
    }
    reader.Close();
}
```

```

        conn.Close();
    }
    return response;
}

public class Contact
{
    public int id;
    public string name;
    public string email;
}

```

Create an instance of the web service

Use the syntax: `new WebReferenceName().ServiceName`.

```
var myWebService = new localhost().weatherService;
```


Invoke a web service method

Use the syntax `webServiceInstance.webServiceMethod()`.

```
var myWebService = new localhost().weatherService;
myWebService.getForecastAsync("New York");
```

Minimize data over the wireless network

You can specify the class fields you want to transfer between the web service and the BlackBerry® MDS Runtime application to reduce the amount of network traffic.

1. In the Solution Explorer, navigate to the web reference that contains the class for which you want to specify the fields to transfer.
2. Expand the web service.
3. Click the class.
4. In the Properties window, in the **Fields** field, click the  button.
5. In the **Field Chooser** dialog box, perform either of the following:

Action	Procedure
Remove a field from the list of fields that are sent from the web service to the application.	<ol style="list-style-type: none"> 1. In the Current Fields list, click the field that you do not want to transfer. 2. Click Remove.
Add field to the list of fields that are sent from the web service to the application.	<ol style="list-style-type: none"> 1. In the Available Fields list, click the field that you want to transfer. 2. Click Add.

Fields that are inherited from a parent class are unavailable. To specify that these fields are not transferred, edit the **Fields** property of the parent class.

6. Click **OK**.

Handle asynchronous messages

All messages sent to and returned from a web service are asynchronous. This means that the application does not block while waiting for the completed message to be returned.

To send and handle the receipt of the data from the web service, you must invoke the web reference, handle the returned data, and bind the data returned by the completed message.

Invoke the web reference method

1. In your script, type the following code to create an instance of the web service:

```
function updateCustomer()
{
    var myWebService = new localhost().shoppingService;
}
```


2. Type the following code to invoke the web service method:

```
function updateCustomer()
{
    var myWebService = new localhost().shoppingService;

    var myCustomer = Customer.create();
    myCustomer.name = "Widget Inc."
    myWebService.updateCustomerAsync(myCustomer.name);
}
```

Handle the returned data

You must assign an event handler to the Completed message.

1. In the Solution Explorer, expand the project.
2. Expand the **Web References** folder.
3. Expand the web reference.
4. Expand the service that contains the method you are invoking.
5. Expand the method you are invoking.
6. Click the method's associated completed method.
7. In the Properties pane, click the  button.
8. In the **Received** field, specify a script by selecting a pre-existing script or double-clicking the Received field to generate a function in the form designer file. This script will run upon receipt of the completed message.

Retrieve data returned by the completed message

The data that the web service returns is an object.

- > To retrieve the data that the web service returns to local data, in the script you specified to handle the Completed message, type the following:

```
function updateCustomerreceived()
{
```

```
var myVar = updateCustomerCompleted.name;  
Dialog.display(myVar);  
}
```

Handle SOAP faults

A WSDL operation definition may specify one or more SOAP faults that may be output during the invocation of the operation. SOAP faults for each operation are mapped to corresponding BlackBerry® MDS Runtime messages and are processed similarly to result messages linked to the request. SOAP fault syntax is not used for ASP .NET web services. ASP .NET returns SOAP faults as part of the response to the request that was called.

SOAP fault messages appear nested within the request message in the Solution Explorer. You can click the SOAP fault message node to view and modify properties in the Properties pane.

Limitation

BlackBerry® MDS Services handles SOAP faults defined in the SOAP 1.1 specification.

Using push integration

Server-side applications can securely push content applications via the BlackBerry® Mobile Data System server. The BlackBerry MDS provides the ability to push content to the BlackBerry® Browser or to a custom BlackBerry® MDS Runtime Application that has been designed to listen for incoming pushes.

To ensure that the push capability is available, contact your system administrator.

If your application is not running in the foreground, the Device methods will not be executed until the application comes to the foreground.

The following sections describe how the push mechanism works, and how to expose functions that can be used by the push mechanism.

Exposing functions as push functions

To expose and configure push functions, add code attributes to regular functions in your BlackBerry® Runtime MDS Application.

You must add code attributes directly above the function you want to expose. You must use square parentheses around the code attributes.

You can use the following four code attributes:

Attribute	Description
DeviceMethod	Exposes the function as a push function
DeviceMethodOptions	Allows you to configure the message as LogSecureMessage and whether the push mechanism should keep or delete the last pushed message
RibbonAlert	An alert will appear on the ribbon on the device when a push message arrives
DialogAlert	A dialog box will appear on the device when a push message arrives. The dialog box can display a custom message.

DeviceMethod code attribute

Use the DeviceMethod attribute to actually expose the function as a push function.

The DeviceMethod attribute must appear before a valid JavaScript® function. If the attribute does not appear before a function, it will generate an error at build time.

The DeviceMethod attribute can be set for any JavaScript function except for onError. If the attribute is set for onError, it will generate an error at build time.

Syntax

To use the DeviceMethod attribute with a function that takes no parameters, use the following syntax:

```
[DeviceMethod]
function myFunction() {
```

```
...  
}
```

If the function takes any parameters, you must specify the parameters in the `DeviceMethod` code attribute as follows:

```
[DeviceMethod(param1 = ParameterType, param2 = ParameterType, ...)]  
function myFunction(param1, param2, ...) {  
...  
}
```

Example: Using the `DeviceMethod` attribute with parameters

```
[DeviceMethod(id = Integer, customer = Customer)]  
function myFunction(id, customer) {  
...  
}
```

Example: Using the `DeviceMethod` attribute with an array parameter

```
[DeviceMethod(id = Integer, customers = Customer[])]  
function myFunction(id, customers) {  
...  
}
```

DeviceMethodOptions code attribute

Use the `DeviceMethodOptions` attribute to set the following two options for the push method:

- `LogSecureMessage`: Use this option to specify that the transmission of the pushed message is `LogSecureMessage`.
- `KeepLastMessage`: Use this option to specify that the push mechanism stores the last pushed message on the device.

The `DeviceMethodOptions` attribute is set using the **true** and **false** keywords.

If either of these options are not specified, they are set to false by default.

The `DeviceMethodOptions` attribute can only be specified for a function that also has the `DeviceMethod` code attribute.

Syntax

To set the options for a function, use the following syntax:

```
[DeviceMethod]  
[DeviceMethodOptions(LogSecureMessage = true/false, KeepLast = true/false)]  
function myFunction() {  
...  
}
```

Example: Set `LogSecureMessage` to true

```
[DeviceMethod(message = String)]  
[DeviceMethodOptions(LogSecureMessage = true)]  
function handleMessage(message) {  
...  
}
```

Example: Set LogSecureMessage to true and KeepLast to false

```
[DeviceMethod(message = String)]
[DeviceMethodOptions(LogSecureMessage = true, KeepLast = false)]
function handleMessage(message) {
  ...
}
```

RibbonAlert code attribute

Use the RibbonAlert code attribute to display an alert on the ribbon on the device when the push message arrives. The RibbonAlert code attribute can only be specified for a function that also has the DeviceMethod code attribute. The RibbonAlert code attribute does not take any parameters. It has an empty constructor.

Syntax

To set the RibbonAlert attribute for a function, use the following syntax:

```
[DeviceMethod]
[RibbonAlert]
function myFunction() {
  ...
}
```

Example: Set RibbonAlert to true

```
[DeviceMethod(message = String)]
[RibbonAlert]
function handleMessage(message) {
  ...
}
```

DialogAlert code attribute

Use the DialogAlert code attribute to display a dialog box on the device when the push message arrives. This attribute can only be specified for a function that also has the DeviceMethod code attribute.

The DialogAlert takes an optional String parameter in its constructor. The String is displayed in the dialog box when the message arrives. If no String is specified, a dialog box will still appear on the device, but it will not display a detailed message.

Syntax

To set the DialogAlert attribute for a function without a custom message in the dialog box, use the following syntax:

```
[DeviceMethod]
[DialogAlert]
function myFunction() {
  ...
}
```

To pass in a custom message to display in the dialog box, use the following syntax:

```
[DeviceMethod]
[DialogAlert("Custom dialog text")]
```

```
function myFunction() {  
    ...  
}
```

Example 1: Display a dialog box with the message, "A message has arrived."

```
[DeviceMethod]  
[DialogAlert("A message has arrived.")]  
function myFunction() {  
    ...  
}
```

Adding a BlackBerry reference

Similar to a web reference, you can add a BlackBerry® reference to a Microsoft® Visual Basic® .NET, or Microsoft Visual C#® project. After you add the BlackBerry reference, Visual Studio® adds a BlackBerry References folder and generates code stubs for the functions that have the DeviceMethod code attribute.

Add a BlackBerry reference from a BlackBerry MDS Runtime Application project in this solution

If the BlackBerry® reference you are adding is from a BlackBerry® MDS Runtime project that is already in the solution, perform the following:

1. In the Solution Explorer, right-click on the Microsoft® Visual Basic® .NET, or Microsoft Visual C#® project.
2. Click **Add Blackberry reference**.
3. Click **BlackBerry application in this solution**.
4. Click **Next**.
5. In the BlackBerry Project List, select the project to add.
6. Click **Next**.
7. To change the name of the BlackBerry reference, type a new name in the text field.
8. Click **Finish**.

If this is the first Blackberry reference you add to the project, a **Blackberry References** folder is added to the project. In the **Blackberry References** folder, is the file with the name you specified in step 7 which contains the code stubs of the methods exposed by this reference and classes that are not primitives that may be used as parameters for these methods.

Add a Blackberry reference from a BlackBerry MDS Runtime Application project outside this solution

If the Blackberry® reference you are adding is in a BlackBerry® MDS Runtime project that is not open in this solution, perform the following:

1. In the Solution Explorer, right-click on the Microsoft® Visual Basic® .NET or Microsoft Visual C#® project.

2. Click **Add Blackberry reference**.
3. Click **BlackBerry application from a specific location**.
4. Click **Next**.
5. Click **Browse**.
6. Navigate to the location of the **.mdsproj** file which contains the Blackberry reference.
7. Select the **.mdsproj** file.
8. Click **Open**.
9. Click **Next**.
10. If you want to change the name of the BlackBerry reference type a new name in the text field.
11. Click **Finish**.

If this is the first Blackberry reference you add to the project, a **Blackberry References** folder is added to the project. In the **Blackberry References** folder, is the file with the name you specified which contains the code stubs of the methods exposed by this reference and classes that are not primitives that may be used as parameters for these methods.

Invoking a BlackBerry device method using Microsoft Visual C#

Public properties of the BlackBerry reference class

The following properties are created for any BlackBerry® reference class that is generated:

Property	Description
bool AllowAutoRedirect	Specifies if the HTTP request used to submit the push will allow redirects. By default, this is set to true.
string Url	The URL of the push listener.
string AppUri	The URI of the BlackBerry® MDS Runtime Application. The default value is discovered from the BlackBerry reference.
string Version	The version of the BlackBerry MDS Runtime Application. The default value is from the BlackBerry reference.
string Locale	The locale of the BlackBerry MDS Runtime Application. The default value is discovered from the BlackBerry reference.
StringCollection Recipients	A collection of recipients to push to. By default, this list is empty.
System.Net.ICredentials Credentials	ICredentials object that allows authentications to be passed to the push listener URL.

Specifying the recipients of the push message

You must use the Device classes public properties to specify the BlackBerry® devices that received the push message.

There are two ways to specify the recipients of the push message: Bulk and Target.

For target sending, you must specify the application URI and a list of BlackBerry device PIN(s). If any of the BlackBerry devices whose PIN is specified by the recipients property is unavailable at the time of the push, the message will not be sent to any of the BlackBerry devices.

To send a bulk push, you must leave the Recipients list empty and specify the application URI, version and locale. The push message will be sent to all BlackBerry devices with the discovered application URI, version and locale. If the recipients list is non-empty, the version and locale properties are ignored even if they have a value assigned.

Invoke a BlackBerry device method using Microsoft Visual C#

The following example shows the execution of the method, DoPushCustomer, which is marked with the DeviceMethod code attribute. The name of the BlackBerry® reference is devicehost.

The following code example uses Microsoft® Visual C#®.

```
devicehost.Device dev = new WindowsApplication1.devicehost.Device();
dev.Url = "http://myMDSserver:7090/mds/PushListener";
devicehost.Customer cust = new devicehost.Customer();
cust.firstName = "John";
cust.lastName = "Doe";
dev.DoPushCustomer(cust);
```

Set credentials using Microsoft Visual Basic .NET

```
'Set the appropriate properties for the BBClient
Dim credentials As New MDSCredentials
With credentials
    .UserName = "admin"
    .Password = "admin"
End With

With BBClient
    .Credentials = credentials
    .Url = "http://localhost:17090/mds/PushListener"
    .Locale = "en"
End With
```

BlackBerry MDS and JavaScript

Enhanced JavaScript comments

Enhanced JavaScript® comments enable you to provide and access context-sensitive tips within the code editor. The tips are enabled by entering comments using standard XML tags. Microsoft® Intellisense® uses the type information specified in the comments to provide dynamic tips while you edit scripts. For example, tips are displayed when you hover over a method, and an window enables you to complete code as you type.

Enhanced JavaScript comments begin with "///" and must be placed within a function body, conventionally at the beginning of the statements block.

Tag	Description
<summary>	The summary tag describes the method.
<param>	The param tag describes the parameters of a method or constructor. These tags should be in the same order as the parameters for the method or constructor, and should use the same names.
<returns>	The returns tag describes the return value of a method. This tag should be used only if the function returns a value.

```
function Testfunction(p1,p2,p3)
{
    ///

```

summary tag

The summary tag describes the method. It has no attributes.

Syntax

```
<summary>description</summary>
```

param tag

The param tag describes the parameters of a method or constructor. The tags should be in the same order as the parameters for the method or constructor, and should use the same names.

Attributes

Type	Description
Name	This attribute specifies the name of the parameter, and is mandatory. The name must be exactly the same as the corresponding parameter for the method.
type	This attribute specifies the name of the type of the parameter. Type information is optional, but you can be more explicit by also setting the integer attribute.
integer	This attribute specifies whether or not a number type is an integer. The integer attribute is optional.
isArray	This attribute specifies whether or not the type is an array. The isArray attribute is optional.
optional	This attribute specifies whether or not the parameter can be optional. This attribute is optional.

Syntax

```
<param name="parameterName" optional="true|false" type="ParameterType"
integer="true|false" isArray="true|false">Description</param>
```

returns tag

The returns tag describes the return value of a method. You should use this tag only if the function described returns a value.

Attributes

Type	Description
type	This attribute specifies the value type.
integer	This attribute specifies whether or not a number type is an integer.
isArray	This attribute specifies whether or not the type is an array.
maybeNull	This attribute specifies whether or not the value can be null.

Syntax

```
<returns type="ValueType" integer="true|false" isArray="true|false"
maybeNull="true|false">Description</returns>
```

Arrays

You can convert JavaScript® arrays to BlackBerry® MDS Application arrays. BlackBerry MDS Application arrays are boundary-bound, while JavaScript arrays are unbound. BlackBerry MDS Application arrays support a limited number of methods.

You can create a BlackBerry MDS Application array by referencing fields on collection items. You can index a BlackBerry MDS Application array using [] notation, and you can manipulate it using the MDSArray functions.

Arrays have the length property, which is the number of elements in the array.

Initialize and populate an array

You can use the following code fragment to initialize and populate a JavaScript® array named **jsArray**. The script initializes **jsArray** and assigns it to the global variable **gvMyMDSdataArray** of type array. The script converts the JavaScript array to a BlackBerry® MDS Application array. The constructor generates a JavaScript array object that the script converts to a BlackBerry MDS Application array when the script assigns it to the global variable **gvMyMDSdataArray**.

```
//Create an array using "new", [literal, literal] etc.
var jsArray = new Array();
var firstOne = MyMDSClass.create();
var secondOne = MyMDSClass.create();

//Populate firstOne, secondOne and push them into the array
jsArray.push(firstOne, secondOne);

//Convert the JavaScript array to a BlackBerry MDS array: gvMyMDSdataArray
//Declare the array as a BlackBerry MDS global variable with a data type of array
gvMyMDSdataArray = jsArray;

//Add the third and fourth elements directly to the BlackBerry MDS array
gvMyMDSdataArray.push(MyMDSClass.create(), MyMDSClass.create());

//Populate the third and fourth elements
gvMyMDSdataArray[2].firstField = "hi"; //expect a string
gvMyMDSdataArray[2].secondField = 23; //expect an integer
gvMyMDSdataArray[3].firstField = "MDS";
gvMyMDSdataArray[3].secondField = 45; //converted to an integer
```

Determine the length of an array

Use the syntax *arrayName.length*.

```
if (gvMyMDSdataArray.length == 0)
{
    Dialog.display("No data found.", DIALOG.OK);
}
else
{
    Dialog.display(gvMDSdataArray.length + " item(s) found." , DIALOG.OK);
}
```

Add an element to an array

Use the syntax: *arrayName.push()*.

```
myArray.push(newStuff1);
myArray.push(newStuff2, newStuff3);
```

Remove the last element from an array

Use the syntax: *arrayName.pop()*.

```
var lastElement = myArray.pop(); //the last element in myArray is now stored in lastElement
```

Remove an element from an array

Use the syntax *arrayName.remove()*.

You specify an element by its value. If the array is of non-primitive Class type, you specify an element by its index.

```
gvStringArray.remove("My String");
gvDataArray.remove(gvCurrentData);
gvDataARray.remove(gvCurrentIndex);
```

Find an element in an array

The *contains()* method is only available to BlackBerry® MDS Application arrays; it is not available to JavaScript® arrays.

To determine if a specified value or Class instance is contained in an array, use the syntax: *arrayName.contains()*.

```
var isInArray = gvArray.contains(element);
```

Numbers

A number in JavaScript® has a 64-bit floating-point format. The BlackBerry® Plug-in for Microsoft® Visual Studio® supports both Integer literal values and floating-point literal values.

Convert the base of a number

Use the syntax *Number.toString(base)*.

```
var myInteger = 3;

//convert to String representing myInteger in its binary format
var binaryStr = myInteger.toString(2); //binaryStr is "11"

//convert a number literal to hexadecimal format
var hexStr = (123).toString(0x10); //hexStr is "7B"
```

Set the precision of a decimal number

Use the *toFixed()* method.

```
var myDecimal = 0.56;
myDecimal = (myDecimal + 0.1).toFixed(2); //result is 0.57 instead of 0.5700000000000001
```

Convert a String to number

Use the *parseInt()* method.

```
var myString = "555, can you find the number?";
var myNumber = parseInt(myString); //myNumber is 555
```

Concatenate a String and a number

Data conversion in JavaScript® is flexible.

```
var aNumber = "55" + 45; //aNumber is "5545". The Integer is converted to a String value
and then concatenated.
var aNumber += Form1.textBox1.value; //in this example, .value returns a String.
var aNumber += parseInt(Form1.textBox1.value); //in this example, the right side returns an
Integer.
```

Variables

JavaScript® supports both explicit and implicit variable declarations. Implicit declarations occur when you assign a variable without using the keyword `var`.

When you create the variable implicitly, the variable is created as a JavaScript global variable.

Example: Declaring variables

You must use explicit declarations for local variables.

```
var varA; //explicit declaration without initialization
var varB = 100; //explicit declaration with initialization
varC = 200; //implicit declaration with initialization. varC is created as a global
variable.
varD; //invalid: No assignment occurs.
```

Example: Declaring global variables

```
varWithGlobalScope = 100; //a global variable is placed outside of the top level functions

scope = "a global"; //another global variable declaration
function funcA()
{
    var localV; //a local variable
}

function funcB()
{
    var scope = "a local"; //the global variable "scope" is hidden by the local variable
"scope"
}
```

GPS support

blackberry.location

The location object is only available on GPS-enabled BlackBerry® devices.

Call `blackberry.location.refreshLocation()` before retrieving latitude or longitude properties to retrieve the most current results. This method returns true on success.

Call `blackberry.location.onLocationUpdate()` to register a callback method that is called when the location is updated. Pass the callback method as the parameter. The method returns true on success.

Call `blackberry.location.removeLocationUpdate()` to unregister a callback method added using the `onLocationUpdate` method. This method uses the function to be removed as its parameter. The method returns `true` on success.

Call `blackberry.location.setAidMode()` to set the GPS aid mode to use by passing 0 for Cellsite, 1 for Assisted or 2 for Autonomous.

Properties

Name	Description
<code>GPSSupported</code>	This property returns <code>True</code> if GPS is supported.
<code>latitude</code>	Obtain the current latitude. If GPS is unavailable, this property is set to 0.
<code>longitude</code>	Obtain the current longitude. If GPS is unavailable, this property is set to 0.

Syntax

```
if ( blackberry.location.GPSSupported ) {
// register callback
blackberry.location.onLocationUpdate( locationCB );

setModeCellsite();
//setModeAssisted();
//setModeAutonomous();

} else {
Dialog.display("GPS not supported");
}

void setModeCellsite()
{
// set to cellsite mode
blackberry.location.setAidMode(0);

// refresh properties
blackberry.location.refreshLocation();

Dialog.display("Cellsite Mode" );
}

void setModeAssisted()
{
// set to assisted mode
blackberry.location.setAidMode(1);

// refresh properties
blackberry.location.refreshLocation();

Dialog.display("Assisted Mode" );
}

void setModeAutonomous()
{
// set to autonomous mode
blackberry.location.setAidMode(2);
```

```
// refresh properties
blackberry.location.refreshLocation();

Dialog.display("Autonomous Mode" );
}

// called when location object changes
void locationCB()
{
Dialog.display("Latitude " +
blackberry.location.latitude);
Dialog.display("Longitude " +
blackberry.location.longitude);
}
```

blackberry.network

This object contains one of the following network identifiers:

- GPRS
- CDMA
- iDEN™
- WLAN
- 3G

Syntax

```
Dialog.display("This BlackBerry device is communicating on the " + blackberry.network + " wireless network.");
```


Developing BlackBerry MDS Runtime Applications

Managing BlackBerry MDS classes and data

The BlackBerry® Plug-in for Microsoft® Visual Studio® supports the following four types of classes: user-defined, discovered, built-in, and JavaScript.

Class	Description
User-defined	Created by the programmer and added to the project.
Discovered	Returned by a web service.
Built-in	BlackBerry® Personal Information Management (PIM) classes provided by the tool.
JavaScript	A subset of classes provided by the JavaScript® language which the BlackBerry® Plug-in for Microsoft® Visual Studio® uses.

The BlackBerry Plug-in for Microsoft Visual Studio also supports collections which are classes that are encapsulated in a collection container. An instance of a collection item can be retrieved using retrieval methods for the collection.

Collections

The BlackBerry® Plug-in for Microsoft® Visual Studio® supports data collections in addition to arrays.

When you create a collection item, it becomes part of the collection for retrieval. You can use a collection to manage collection items the same way you would manage records in a database table. You can use a primary key to add items to a collection. For example, when an application creates an instance of an Order collection item, the BlackBerry® MDS Runtime adds the instance to the existing collection of Order collection items.

You must use a collection when you require storage that is analogous to a database. The BlackBerry Plug-in for Microsoft Visual Studio does not support database tables.

If you want the data to be saved in flash storage on the device, set the collection storage mode to persistent . If you set the storage mode to non-persistent, the collection will behave like a hash table.

You can perform the following database operations on a collection of keyed data items:

- retrieve a collection item using the primary key
- retrieve an array of collection items using the where clause (one field condition only)
- sort returned results from a collection using any field

Consider the following when you use collections:

- When a web message returns a collection item, the item is added to the collection.

- If a collection item with the same primary key is returned, the collection item is updated.

You can create collection items only when the items have a defined primary key.

Managing collections using script

To create an instance of a user-defined class or collection, use the `create()` function.

User-defined collection items support methods to retrieve, update and remove items from the collection.

Create a collection item

Use the syntax `collectionName.create(primaryKey)`.

If the item already exists, the existing item is retrieved and a new item is not created.

```
var newEmployee = Employee.create(ID);
newEmployee.fullName = "John Smith";
newEmployee.startDate = new Date(2005, 1, 2);
newEmployee.salary = 40000;
```

The Employee item is automatically added to the Employee collection.

Update a collection item

Use the syntax `collectionName.fieldName`.

```
var newEmployee = Employee.find(ID); //retrieve the item using the primary key "ID"
if (newEmployee != null)
{
    newEmployee.salary += 30000;
}
```

Retrieve a collection item using a primary key

Use the syntax `collectionName.find(primaryKey)`.

```
var employee = Employee.find(ID);
Dialog.display("You found employee '" + employee.name + "'");
```

Retrieve a collection item using a where clause

Use the syntax `collectionName.findWhere(whereClause)`.

```
var employee = Employee.findWhere("unit='" + department.name + "'");
department.personnelArray = employee; //add the employee(s) found to the department
DepartmentInfo.display(department);
```

The matching items are returned as a BlackBerry® MDS Application array.

Retrieve all items in a collection

Use the syntax `collectionName.all()`, which returns an array.

```
var employees = Employee.all();
var len = employees.length;
for (var i=0; i<len; i++)
{
    if (employees[i].name == "John Smith");
    {
        Dialog.display("Employee found.");
    }
}
```

Remove an item from a collection

To retrieve an item from a collection, use the syntax `collectionName.find(primaryKey)`.

To remove the item from the collection, use the syntax `collectionItem.remove()`.

```
var dropEmployee = Employee.find(ID);
if (dropEmployee != null)
{
dropEmployee.remove();
}
```

Remove all the items from a collection

Use the syntax `collectionName.removeAll()`.

```
Employee.removeAll(); //The Employee collection contains no items
```

Add a collection to the project

1. In the Solution Explorer, right-click the project name or the folder in which you want to add the collection.
2. Select **Add > New Item**.
3. In the Visual Studio installed templates pane, click **Class**.
4. In the **Name** field, type the name of the collection.
5. Click **Add**. The collection definition file is created and opened.
6. In the **Class Name** field, type the name of the collection.
7. In the **Class Type** field, specify the collection type (either `Collection`, or `PersistentCollection` and the field to use as the primary key for this class).

You may have to complete this step after defining the fields for this class.

See "Persistent and non-persistent data storage" on page 48 for more information.

8. If you are creating a subclass of an existing class, in the **Parent** field, specify the existing superclass.
9. In the fields table, define the class members.
10. To commit the changes, on the **File** menu, click **Save**.

Enumerations

In the BlackBerry® Plug-in for Microsoft® Visual Studio® you can define custom enumerations for your application or use built-in enumerations with built-in data to integrate your application with other BlackBerry® device applications.

Access an enumeration element

Use the syntax `enumType.enumName`.

```
gvFirstTask.priority = PRIORITY.HIGH;
```

Compare enumeration elements

Use the syntax `enumType.enumName`.

```
if (gvFirstTask.priority == PRIORITY.HIGH)
{
//perform an action
}
```

Global Variables

Global variables are available from anywhere in the BlackBerry® Plug-in for Microsoft® Visual Studio® application.

Global variables can contain primitive data types, enumerated values, classes, collection items, or arrays of these types.

The way a global variable performs depends on whether the data contained by the global variable is persistent or non-persistent. If the data is persistent, the application creates an instance of the global variable with either the last stored value, the default value you set, or the system default value for the global variable data type. With non-persistent global variables, the application creates an instance of the global variable with the default value you set or the system default value.

Because global variables can be accessed by multiple resources (a script, a web service) and have no locks, there is no guarantee on the value that the global variable contains. For instance, if a web service changes the global variable, a script that accesses it will be unaware of the change. Even if you add a check to the global variable before accessing it can yield an unexpected value since another script could change it right after the check is executed. Use collections and local variables for storing data.






Persistent global variables are useful for storing information needed to initialize the application, or to remember user preferences (similar to cookies). Global variables are not recommended for the storage of a persistent list of items or in place of a database.

If you need to store lists of items, use collections.

Manage global variables


1. In the Solution Explorer, expand the project.
2. Expand **Globals**.
3. Double-click **Globals.mds**.
4. Perform one of the following actions:


Action	Procedure
Rename a global variable.	> In the Name field, type a new variable name.
Change the data type.	> In the Type field, specify a new type. When you change the data type, the default value is cleared.
Change the data storage mode.	> To change the data storage mode to persistent, select the Persistent check box. > To change the data storage mode to non-persistent, clear the Persistent check box.

Create an array of values.	<ol style="list-style-type: none"> 1. Select the Array check box. 2. In the Default Value field, click the button. 3. In the Default Array Values dialog, click the  button. 4. In the Value field, specify the value. 5. To create additional values, repeat steps c through d. 6. Click OK.
Change a value in an array.	<ol style="list-style-type: none"> 1. In the Default Value field, click the  button. 2. In the Default Array Values dialog, select the value you want to edit. 3. In the Value field, specify a new value. 4. To change any additional values, repeat steps b through c. 5. Click OK.
Change the order of values in an array.	<ol style="list-style-type: none"> 1. In the Default Value field, click the  button. 2. In the Default Array Values dialog, select the value you want to move. 3. To move the value up in the array, click the up arrow. To move the value down in the array, click the down arrow. 4. To change the order of any additional values, repeat steps b through c. 5. Click OK.
Remove a value from an array	<ol style="list-style-type: none"> 1. In the Default Value field, click the  button. 2. In the Default Array Values dialog, select the value you want to remove. 3. Click the  button. 4. To remove any addition values, repeat steps b through c. 5. Click OK.
Remove a global variable	<p>Click the Name field of the variable you want to remove.</p> <p>Click Remove Variable.</p>

5. To commit the changes, on the **File** menu, click Save **Globals.mds**.

Add a global variable

1. In the Solution Explorer, expand the project.
2. Expand the **Globals** folder.
3. Double-click **Globals.mds**.
4. In the **Name** column, type a name for the global variable.
5. In the **Type** column, select the variable type from the drop-down list.
6. To store the global variable information each time the application runs, select the **Persistent** check box.
7. If the global variable is an array, select the **Array** check box.
8. To specify the default value, in the **Default Value** field, click the  button.
9. Type the default value in the **Enter Default Value** field. If the global variable type has a finite set of values, click the value from the drop-down list.

10. If the variable is an array, click the  button to append the value to the end of the array. To change the order of values in the array, select the value and click the up and down arrows.
11. Click **OK**.
12. To commit the changes, on the **File** menu, click **Save Globals.mds**.

Persistent and non-persistent data storage

You can store data as persistent or non-persistent.

- **Persistent:** Use this approach to maintain the state information of the data committed to flash memory on the device. The BlackBerry® MDS Runtime processes the state information when the application starts. Applications that store data in a persistent state can offer the user more functionality and an improved user experience compared to applications that do not such as a browser application.
- **Non-persistent:** Use this approach to store temporary information to process an event. Use non-persistent data storage to manage information that is relevant for the current application only. This is similar to using an in memory hash table.

Create an instance of a Class

To create an instance of a user-defined class or collection item, use the syntax `className.create()`.

```
var aPaper = Paper.create() //Paper is a user-defined class
aPaper.topic = 'Some Topic';
aPaper.publishDate = new Date(2005, 3, 15);
aPaper.description = "Some description.";
magazine.papers.push(aPaper);
```

If the class you are creating is part of a collection, the `create()` function creates a new instance and a pointer is returned to the item in the collection.

Disable class auto-instantiation

By default, the BlackBerry® MDS Runtime always creates class instances. Non-array fields are set to their default values, which are either system defaults or values provided through the Class Editor. This can cause problems when sending or receiving null instances of these classes to or from the back end web service.

When auto-instantiation is disabled, class instances such as globals, form local variables, or form parameters, remain null unless they are explicitly created or assigned an instance.

1. In the Solution Explorer, expand the project.
2. Expand the **Properties** folder.
3. Double-click **Settings.mds**.
4. Click the **Disable Class Auto-instantiation** check box.
5. To commit the changes, click **File > Save Settings.mds**.

Display collections and arrays of classes


You can use the Repeater control to display the items within a collection or array of classes.

When you bind an initial value to a Repeater control, the Repeater control can become a data source for the nested controls. The Repeater will then iterate through all the items and display the fields for each item in the Repeater.

The same information can also be presented in table format using a DataGridView control. Data binding rules for a Repeater and a DataGridView are exactly the same. Nested controls on a Repeater for displaying field information are replaced by columns on a DataGridView control. Each collection item or class array member is displayed in its own row in the table.


Optionally, you can also display primitive-type fields of a collection and class array using choice controls such as a RadioGroup, ComboBox, or ListBox.

To bind the collection, Collection1, to the Repeater, perform the following:

1. Drag a Repeater control from the BlackBerry Controls toolbox onto the form.
2. Click the Repeater.
3. In the Properties window, expand **DataBindings**.
4. In the **Rows** field, click the  button.
5. In the Data Bindings Editor dialog, expand **Collections**.
6. Click **Collection1**.
7. Click **OK**.

The Repeater control displays the collection's fields in its nested controls. Typically, these controls are used for read-only display.

To display a specific field, firstName, for each item in Collection1, perform the following:

1. Drag a TextBox control from the BlackBerry Controls toolbox onto the form inside the Repeater.
2. Click the TextBox.
3. In the Properties window, expand **DataBindings**.
4. In the **Text** field, click the  button.
5. In the Data Bindings Editor dialog, expand the form folder.
6. Expand **Repeater**.
7. Click **firstName**.
8. Click **OK**.

Objects

JavaScript® objects are created by invoking constructor functions. A constructor function is any function that follows the keyword **new** and initializes an object.

For user-defined classes or collections, use the create() function call and not the keyword **new** to initialize an object.

Example: Built-in Object constructor

```
var scriptObject = new Object();
var currentDate = new Date();
var lastHoliday = new Date(2005, 11, 25); //11 is December because months are zero-based
```

Example: User-defined constructor

```
var myClass = myClass.create();
```

Example: Array constructor

```
var myArray = new Array();
array[0] = 1;
array[1] = 2;
array[2] = 3;
array[3] = "the end of the array"; //valid: an array can contain elements of different
types
```

```
//initialize the array in the constructor function
var myArray = new Array(1,2,3, "the end of the array");
```

```
//initialize with an array literal
var myArray = [1,2,3 "the end of the array"];
```

Managing the user interface

Forms

Forms are analogous to a singleton class. You do not need to create an instance of a form in your BlackBerry® MDS Runtime Application.

The form and the public fields of the controls in the form are always accessible by the application. For example, to access the value of `button1` in `Form1`, use the following code:

```
function myFunction(String caption)
{
    Form1.button1.value = caption;
}
```

Form display and refresh event model

The order of events that occurs when a form loads is the following:

1. Form initialize event

The first event that occurs when a form loads is the initialize event associated with the form.

This event occurs when the form is loaded using the `FormName.display()` function or the `OpenForm ClickAction` from another form.

This event does not occur when the form is refreshed.

2. Controls' initialize event

After the form has initialized, the initialized event for each control occurs. Any values set in the control's Initialized event will overwrite any values set in the form's Initialized event.

This event does not occur when a form is refreshed.

Implicit refresh

If you change the `.value` or `.visible` property of a form control through script, the changes are applied to the control when the code line is executed, but the events that are triggered by a full form refresh do not occur.

3. Initial values are resolved

Any DataBindings specified for the controls are set. Any values set by the previous Initialized events are overwritten.

This is also the first event that occurs if the form is displayed but not initialized. This happens when the form is displayed by not using the mechanisms which cause the form's Initialized event to occur.

4. Form Activated event

This event occurs whenever the form is refreshed. It does not occur when you return to the form using the Escape key, or using the `Screen.close(false)` function. Return to a form using `Screen.close(true)` to ensure that transaction changes are kept and that the form you return refreshes.

The form designer file

Each form has a form designer file associated with it. All of the functions associated with a form by default are written in the form designer file. The form designer file is created automatically when the form is created.

The members in the form designer file are global. That is, any form can access the functions created in any script file. Access to a form control must begin with the form name. There is no context for the **this** keyword, so when you want to access form control, you must specify the context explicitly.

```
function Form1_button1_click()
{
    var myCaption = Form1.button1.value;
    Dialog.display(myCaption);
}
```

You must create functions used by the form or a control within the form in that form's designer file as a best practice.

Transitioning to another form

You can specify the transition to a new form (to add a new form to the application's stack) through the GUI or through script.

Transition to another form through script


You can use the `Form.display()` function in the script. *Form* specifies the form to which to transition.

Example: Assign a button click event in Form1 using script to invoke a transition to Form2

```
function Form1_button1_Click()
{
    Form2.display();
}
```

```
}
```

Transition to another form through the GUI

1. Select the button or menu item.
2. In the Properties window, expand the **ClickAction** property.
3. In the **ActionType** field, select **OpenForm**.
4. In the **FormName** field, specify the form to which to transition.
5. If you want to pass any parameters to the form, in the **FormParameters** field, click the  button and choose the parameters to pass.

You cannot pass primitives as parameters between forms. To pass a primitive, create a class which stores the primitive value and then pass that class.

Refreshing a form


You can refresh a form in two ways in order to update the data in the form's controls.

Refresh a form through a script

To trigger the form to refresh you can use the `Screen.refresh()` function

```
function Form1_button1_Click()  
{  
    Screen.refresh();  
}
```


Refresh a form when a web message is returned

1. Click the form (not a control).
2. In the **Properties** window, in the **RefreshMessages** field, click the  button.
3. In the Refresh Message Editor dialog, select the message check box.
4. Click **OK**.

When the web service returns the message, the form will automatically refresh.

Pass data between forms using script

You cannot pass primitives as parameters. However, you can create a class to store the primitive value and pass it between forms.

1. In the Solution explorer, double-click the form to which you want to pass the parameters.
2. Click the form (not a control).
3. In the Properties window, select **Parameters**.
4. Click the  button in the **Parameters** field.
5. Click **Add**.
6. In the **Name** field, type the parameter name.

7. In the **Type** field, specify the parameter type.
By default, only discovered and user-defined classes are available in the Type field. To show built-in classes, select the **Show built-ins** check box.
8. Repeat steps 4 to 6 to add more parameters.
9. Click **OK**.
10. In the form from which you want to pass the parameters, use the `Form.display()` function.

Example: Pass parameters to Form2 when button1 is clicked

```
Form1_button1_click()
{
    var expItem = myObject.create();
    expItem.myField = "Hello world";
    Form2.display(expItem);
}
```

The parameters passed from a form in the function call are bound to the parameters defined in the order provided on the form receiving the parameters.

Although these parameters are defined, you can omit any of them by passing a null value in the parameter list. For example:


```
Form1_button1_click()
{
    var expItem1 = myObject.create();
    var expItem2 = myObject.create();
    expItem1.myField = "Hello";
    expItem2.myField = "World";
    Form2.display(null , expItem2);
}
```


The parameters passed in can be used as a data source by form controls.

Passing data between forms using the UI

You cannot pass primitives as parameters. However, you can create a class to store the primitive value and pass it between forms.

Pass data between forms using the UI

1. In the Solution explorer, double-click the form to which you want to pass the parameters.
2. Click the form (not a control).
3. In the Properties window, select **Parameters**.
4. Click the  button in the **Parameters** field.
5. Click **Add**.
6. In the **Name** field, type the parameter name.
7. In the **Type** field, specify the parameter type.
By default, only discovered and user-defined classes are available in the Type field. To show built-in classes, select the **Show built-ins** check box.

- Repeat steps 4 to 6 to add more parameters.
- Click **OK**.
- In the Solution Explorer, double-click the form passing the parameters.
- Click the button or menuitem that will trigger the form transition.
- In the Properties window, expand **ClickAction**.
- In the **FormName** field, specify the form to which you are passing the parameters.
- In the **FormParameters** field, click the  button.
- In the Open Parameters dialog, in the **Available Global Variables, Form Parameters, and Local Variables** pane, select the parameter to pass.
- Click **Bind Argument**.
- Repeat steps 15 and 16 for any additional parameters to pass.
- Click **OK**.

The parameters passed in the function call are bound to the parameters defined in the order provided in the form receiving the parameters.

The parameters passed in can be used as a data source by form controls.


Access form parameters in script

Form parameters can be accessed using script with the syntax *FormName.ParameterName*.

```
function Form1_Initialized()  
{  
    var myParam = Form1.param1;  
    Dialog.display(myParam);  
}
```

Define local variables in a form

You cannot define primitives as local variables. However, you can create a class to store the primitive value.

- Click the form (not a control).
- In the Properties window, click on the **LocalVariables** field.
- Click the  button.
- Click **Add**.
- In the **Name** field, type the name of the newly created variable.
- In the **Type** field, specify the variable type.

By default, only discovered and user-defined classes are available in the Type field. To show built-in classes, select the **Show built-ins** check box.

- Click **OK**.

Access form local variables through script

You can access local data in the script.

```
function Form1_button1_click()
{
    Dialog.display(Form1.myLocalVar.myField);
}
```

Arranging form controls

To arrange form controls you can use the `FlowLayoutPanel` and `TableLayoutPanel` controls.

The layout panel controls are available in the BlackBerry® controls toolbox.

Unlike the Microsoft® Windows® application arrangement which uses X, Y coordinates to set the position of form controls, the BlackBerry layout panels arrange the controls by snapping them into position as specified by the panel's settings.

Layout panels are also useful if you want to group form controls together for visibility. If you specify a Layout panel's visible field as false, then all the controls within the panel do not appear on the form.

FlowLayoutPanel control

You can use the `FlowLayoutPanel` control to specify the arrangement of controls in the following formats:

- **TopDown:** The controls are arranged vertically with one control per line.
- **LeftToRight:** The controls are arranged horizontally from left to right. The controls wrap to the next line if necessary.
- **Edge:** The controls are arranged vertically with two controls per line. The first control snaps to the left side of the panel. The second control snaps to the right side of the panel.

The layout of the `FlowLayoutPanel` can be specified by performing the following:

1. Click the `FlowLayoutPanel` control.
2. In the Properties window, in the **FlowDirection** field, specify the arrangement.

TableLayoutPanel control

You can use the `TableLayoutPanel` control to specify the arrangement of controls in a table format.

To specify the size of the `TableLayoutPanel`, perform the following:

1. Click the `TableLayoutPanel` control.
2. In the Properties window, in the **ColumnCount** field, specify the number of columns in the table.
3. In the **RowCount** field, specify the number of rows.

The Table columns and rows will resize automatically when controls are added.

If you remove a column or row from the `TableLayoutPanel` control, any controls within the removed cells are also removed.

Manage forms using script

You can use script to transition to another form, refresh a form, or close a form.

You can use script to initialize and update form controls, retrieve data from form controls, and show or hide form controls.

Transition to another form

Form transitions can be triggered by a button action, a menu action, or through script.

When you create a BlackBerry® MDS Runtime Application, by default Form1 is displayed. To transition to another form using script, use the syntax *formName.display()*. You can use the *display()* method to pass parameters to the form to display. When a script calls the *display()* method, the BlackBerry MDS Runtime places the new form on the stack.

```
Form2.display(Form1.localVar);
```

Close a form

To close the current form and display the previous form, use the syntax: *Screen.close(boolean)*. The *Screen.close()* method removes the current form from the stack.

You can also use the *Screen.close()* method to control the lifecycle of data modifications. To save changes to the current screen, set the close screen parameter to true; to discard changes to the screen, set the parameter to false. By default, if the parameter is not set, the changes made to the screen are discarded. The current transaction remains open, regardless of whether the application saves or discards the changes on the screen.

A user can close a screen by pressing the Escape button or clicking the Back menu item on the BlackBerry® device. These user actions remove the current screen from the stack and display the previous screen. The application discards data changes to the current screen and does not include the changes in the current transaction.

```
//Event handler for an OK button click on the form
function Form1_OKButton_Click()
{
    Screen.close(true);
}

//Event handler for a Cancel button click on a form
function Form1_CancelButton_Click()
{
    Screen.close(false);
}
```

Display a dialog box

To display a dialog box (similar to a JavaScript alert), use the following syntax:

```
function Form1_button1_click()
{
    Dialog.display("Message text");
}
```

Access form controls

You can refer to each form control by name as a field of its parent form. Use the syntax *FormName.ControlName.FieldName*.

The return type and value varies depending on the type of form control.

Example: Setting a TextBox control field by name

```
//In the following code snippet, employee name is a text box
loginName = loginForm.employeeName.value;
loginName = "Mr. " + loginName;
loginForm.employeeName.value = loginName;
```

Initializing a form control**Initialize editable form controls**

You can use script to initialize or update the content of editable form controls (for example, a TextBox, MaskedTextBox, Label, PictureBox, or Button). Form controls have a `.value` property that you can initialize in a script and display as the value of the control.

The `.value` property of an editable form control expects a String value. The script converts any non-String data types to a String value and assigns the String value to the `.value` property.

```
Form1.employeeName.value = gvEmployee.name; //employeeName is a TextBox control
Form1.employeeSalary.value = gvEmployee.salary; //employeeSalary is a TextBox control
```

Initialize choice form controls

You can use script to initialize or update the content of choice form controls (single-selection or multiple-selection lists). Choice controls have a `.value` property that can be set in script. The `.value` property is displayed as a list of values in the choice control. The `.value` property of a choice control expects an array of String values.

The array type depends on how you initialize the choice control (the initial values). If you initialize the choice control using a class, then the `.value` property returns an array of that class. If the choice control is initialized using a String array, then the property value returns an array of Strings.

```
Form1.ComboBox1.value = ["Mr.", "Mrs.", "Ms."];
```

Initialize a CheckBox control

You can use script to set that the CheckBox control is selected by assigning a boolean value to the `.value` property of the CheckBox. If `.value` is set to true, the CheckBox is selected; if it is false, it is not selected. By default, if the `.value` property is not set, the CheckBox is not selected.

```
Form1.CheckBox1.value = true;
```

Retrieve data from a form control

Through script, you can use a variety of fields to retrieve data from a form control. Use the syntax `FormName.ControlName.FieldName`.

Retrieve data from an editable form control

You can use the `.value` property to retrieve the String representation of content from a textBox, MaskedtextBox, Label, PictureBox, or Button control.

```
currentEmployee.name = Form1.employeeNameTextBox.value;

//.value (String) converted to an Integer
currentEmployee.salary = Form1.employeeSalaryTextbox.value;
```

Retrieve data from a single-choice control

You can use the `.value`, `.selected`, and `.selectedValue` properties to retrieve data from a single-choice control.

The `.value` property returns an array of String representations of the choice items.

The `.selected` property returns a zero-based Integer representation of the index of the selected value.

The `.selectedValue` property returns the selected Object.

```
//get the name of the employees from a ComboBox of employee names
//employeeNames in this case points to an array of Strings
var employeeNames = Form1.employeeNamesComboBox.value;

//get the index of the employee selected
var selectedEmployeeIndex = Form1.employeeNameComboBox.selected;

//get the Object value of the selected employee
var selectedEmployee = Form1.employeeComboBox.selectedValue;
//this is the same as the following line:
var selectedEmployeeAlternate = Form1.employeeNamesComboBox.value[selectedEmployeeIndex];
```

Retrieve data from a multiple-choice control

You can use the `.value`, `.selected`, and `.selectedValue` properties to retrieve data from a multiple-choice control.

The `.value` property returns an array of String representations of the choice items. The `.selected` property returns an array of zero-based Integers of the indices of the selected values. The `.selectedValue` property returns an array of Objects of the selected items.

```
//get the names of the employees from a CheckedListBox of employee names
//employeeNames in this case points to an array of Strings
var employeeNames = Form1.employeeNamesCheckedList.value;

//get the index of the employee selected
//selectedEmployeeIndex in this case, points to an array of Integers
var selectedEmployeeIndex = Form1.employeeNameCheckedList.selected;

//get the Object values of the selected employee
var selectedEmployees = Form1.employeeComboBox.selectedValue;
```

Retrieve data from a check box

You can use the `.value` property to retrieve whether the CheckBox control is selected or not.

```
if (Form1.CheckBox1.value = true)
{
    Dialog.display("The CheckBox is selected.");
}
```

Set display conditions of form controls

Through script, you can show or hide form controls by assigning a boolean value to the `.visible` property. Use the syntax `FormName.ControlName.visible = boolean`.

You can change the visibility of any form control except the Repeater.

If a form control is contained within a layout panel, the control inherits its visibility property from the layout panel.

```
if (Form1.CheckBox1.value == true) {
    Form1.Label1.visible = true;
    Form1.menuItem1.visible = true;
```

```

}

//hide a region if there are no items in the collection
if (Collection1.all().length == 0)
{
    Form1.collectionLayoutPanel.visible = false;
}

```

Data bindings


You can populate the values of form controls with values from a data source such as a collection or a local variable. For controls that use a collection or an array of classes as its data source, you can specify a filter (similar to an SQL query's WHERE statement) to populate the control with a subset of the collection.

Populating a control using data bindings

You can populate the value of a control through the GUI or through script.

Data bindings bound using the GUI take precedence over hard coded values, even on refresh.

Populate the value of a control through the GUI

1. Click the control.
2. In the Properties window, expand the **DataBindings**.
3. In the sub-property that appears ("Text" for controls which have a value of type String), click the  button to choose the data source.
4. In the Data Bindings Editor, navigate to the variable or collection to fill the control. If you want to manually enter the initial value, select the **Manually enter InitialValue syntax** check box in the Advanced section.

Only data sources with legal values for the control are visible. However, some data sources appear that cannot be selected since they have sub-values which are legal for the control.
5. If you want to set up a filter for the data source, specify the filter settings in the Filter section in the Data Bindings Editor dialog.
6. If you want to sort the values from the data source, specify the order settings in the Sort section in the Data Bindings Editor dialog.
7. Click **OK**.

Populate the value of a control through script

1. Open the form designer for the form with the control.
2. In the initialization function, assign an initial value.

Example: Assigning an initial value to a button control

```

function Form1_button1_initialize()
{
    Form1.button1.value = "Click Me.";
}

```

To commit the changes, on the **File** menu, click **Save**.

Button control valid data bindings

You can bind the following types of values to the initial value of the Button form control:

- Static text

Global variable	Local variable	Form parameter
Primitive	Field	Field
<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String 	<ul style="list-style-type: none"> • binary • boolean • date • decimal • Integer • long • String 	<ul style="list-style-type: none"> • binary • boolean • date • decimal • Integer • long • String
Enumeration	<ul style="list-style-type: none"> • enumeration • class field • collection item field 	<ul style="list-style-type: none"> • enumeration • class field • collection item field
Field		
<ul style="list-style-type: none"> • binary • boolean • date • decimal • Integer • long • String • enumeration • class field • collection item field 		

CheckBox control valid data bindings

You can bind the following types of values to the initial value of the CheckBox form control:

Global variable	Local variable	Form parameter	Repeater (control is nested inside)
Primitive	Field	Field	Field
<ul style="list-style-type: none"> • boolean 	<ul style="list-style-type: none"> • boolean • class field of type boolean 	<ul style="list-style-type: none"> • boolean • class field of type boolean 	<ul style="list-style-type: none"> • boolean • class field of type boolean

CheckedListBox control valid data bindings

You can bind the following types of values to the initial value of the CheckedListBox form control:

- Static text

Enumeration	Global variable	Local variable	Form parameter	Collection
<ul style="list-style-type: none"> • Any built-in enumeration • Any user-defined enumeration • Any discovered enumeration 	Primitive Array <ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String Enumeration array <ul style="list-style-type: none"> • Built-in enumeration • User-defined enumeration • Discovered enumeration Field Array <ul style="list-style-type: none"> • boolean • date • decimal • Integer • String • long • enumeration • data component 	Field Array <ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String • enumeration • class field • collection item field 	Field Array <ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String • enumeration • class field • collection item field 	Field <ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String • enumeration • class field • collection item field

ComboBox control valid data bindings

You can bind the following types of values to the initial value of the ComboBox form control:

- Static text

Enumeration	Global variable	Local variable	Form parameter	Collection
<ul style="list-style-type: none"> • Any built-in enumeration • Any user-defined enumeration • Any discovered enumeration 	Primitive Array <ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String Enumeration array <ul style="list-style-type: none"> • Built-in enumeration • User-defined enumeration • Discovered enumeration Field Array <ul style="list-style-type: none"> • boolean • date • decimal • Integer • String • long • enumeration • data component 	Field Array <ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String • enumeration • class field • collection item field 	Field Array <ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String • enumeration • class field • collection item field 	Field <ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String • enumeration • class field • collection item field

DataGridView control valid data bindings

You can bind the following types of values to the initial value of the DataGridView form control:

Global variable	Local variable	Form parameter	Collection
Array <ul style="list-style-type: none"> • class • collection item 	Field Array <ul style="list-style-type: none"> • class • collection item 	Field Array <ul style="list-style-type: none"> • class • collection item 	<ul style="list-style-type: none"> • any built-in PIM collection • any user-defined collection • any discovered collection
Field Array <ul style="list-style-type: none"> • class • collection item 			

DateTimePicker control valid data bindings

Although the following data bindings are valid, it is recommended you use data types that can be converted to a valid Date-Time.

You can bind the following types of values to the initial value of the DateTimePicker form control:

- Static text

Global variable	Local variable	Form parameter	Repeater (control is nested inside)
Primitive	Field	Field	Field
<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String 	<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String 	<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String 	<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String
Enumeration	<ul style="list-style-type: none"> • enumeration • class field • collection item field 	<ul style="list-style-type: none"> • enumeration • class field • collection item field 	<ul style="list-style-type: none"> • enumeration • class field • collection item field
Field			
<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String • enumeration • class field • collection item field 			

Form valid data bindings

You can bind the following types of values to the initial value of the Form.

A Form's initial value refers to the title of the form.

- Static Text

Global variable	Local variable	Form parameter
Primitive	Field	Field
<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String 	<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String 	<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String
Enumeration	<ul style="list-style-type: none"> • enumeration • class field • collection item field 	<ul style="list-style-type: none"> • enumeration • class field • collection item field
Field		
<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String • enumeration • class field • collection item field 		

Label control valid data bindings

You can bind the following types of values to the initial value of the Label form control:

- Static text

Global variable	Local variable	Form parameter	Repeater (control is nested inside)
Primitive	Field	Field	Field
<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String 	<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String 	<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String 	<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String
Enumeration	<ul style="list-style-type: none"> • enumeration • class field • collection item field 	<ul style="list-style-type: none"> • enumeration • class field • collection item field 	<ul style="list-style-type: none"> • enumeration • class field • collection item field
Field			
<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String • enumeration • class field • collection item field 			

ListBox control valid data bindings

You can bind the following types of values to the initial value of the ListBox form control:

- Static text

Enumeration	Global variable	Local variable	Form parameter	Collection
<ul style="list-style-type: none"> • Any built-in enumeration • Any user-defined enumeration • Any discovered enumeration 	Primitive Array <ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String Enumeration array <ul style="list-style-type: none"> • Built-in enumeration • User-defined enumeration • Discovered enumeration Field Array <ul style="list-style-type: none"> • boolean • date • decimal • Integer • String • long • enumeration • data component 	Field Array <ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String • enumeration • class field • collection item field 	Field Array <ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String • enumeration • class field • collection item field 	Field <ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String • enumeration • class field • collection item field

MaskedTextBox control valid data bindings

You can bind the following types of values to the initial value of the MaskedTextBox form control:

- Static text

Global variable	Local variable	Form parameter	Repeater (control is nested inside)
Primitive	Field	Field	Field
<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String 	<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String 	<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String 	<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String
Enumeration	<ul style="list-style-type: none"> • enumeration • class field • collection item field 	<ul style="list-style-type: none"> • enumeration • class field • collection item field 	<ul style="list-style-type: none"> • enumeration • class field • collection item field
Field			
<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String • enumeration • class field • collection item field 			

MenuItem control valid data bindings

You can bind the following types of values to the initial value of the MenuItem form control:

- Static Text

Global variable	Local variable	Form parameter
Primitive	Field	Field
<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String 	<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String 	<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String
Enumeration	<ul style="list-style-type: none"> • enumeration • class field • collection item field 	<ul style="list-style-type: none"> • enumeration • class field • collection item field
Field		
<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String • enumeration • class field • collection item field 		

PictureBox control valid data bindings

You can bind the following types of values to the initial value of the PictureBox form control.

A PictureBox's initial value refers to the URL of the image.

- Static Text (a static URL to an image)

Global variable	Local variable	Form parameter
Primitive	Field	Field
<ul style="list-style-type: none"> • binary • String 	<ul style="list-style-type: none"> • binary • String 	<ul style="list-style-type: none"> • binary • String
Field		
<ul style="list-style-type: none"> • binary • String 		

RadioGroup valid data bindings

You can bind the following types of values to the initial value of the RadioGroup form control:

- Static text

Enumeration	Global variable	Local variable	Form parameter	Collection
<ul style="list-style-type: none"> • Any built-in enumeration • Any user-defined enumeration • Any discovered enumeration 	Primitive Array <ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String Enumeration array <ul style="list-style-type: none"> • Built-in enumeration • User-defined enumeration • Discovered enumeration Field Array <ul style="list-style-type: none"> • boolean • date • decimal • Integer • String • long • enumeration • data component 	Field Array <ul style="list-style-type: none"> • boolean • date • decimal • Integer • String • enumeration • class field • collection item field 	Field Array <ul style="list-style-type: none"> • boolean • date • decimal • Integer • String • enumeration • class field • collection item field 	Field <ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String • enumeration • class field • collection item field

Repeater control valid data bindings

You can bind the following types of values to the initial value of the Repeater form control:

Global variable	Local variable	Form parameter	Collection
Array <ul style="list-style-type: none"> • class • collection item 	Field Array <ul style="list-style-type: none"> • class • collection item 	Field Array <ul style="list-style-type: none"> • class • collection item 	<ul style="list-style-type: none"> • any built-in PIM collection • any user-defined collection • any discovered collection
Field Array <ul style="list-style-type: none"> • class • collection item 			

TextBox control valid data bindings

You can bind the following types of values to the initial value of the TextBox form control:

- Static text

Global variable	Local variable	Form parameter	Repeater (control is nested inside)
Primitive	Field	Field	Field
<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String 	<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String 	<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String 	<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String
Enumeration	<ul style="list-style-type: none"> • enumeration • class field • collection item field 	<ul style="list-style-type: none"> • enumeration • class field • collection item field 	<ul style="list-style-type: none"> • enumeration • class field • collection item field
Field			
<ul style="list-style-type: none"> • boolean • date • decimal • Integer • long • String • enumeration • class field • collection item field 			

Stylesheets

You can use a stylesheet to organize sets of style properties for a form or form controls. You can apply the styles you define in a stylesheet to any form control.

Styles

A style is a named set of formatting properties. You can apply styles to change the appearance of forms and form controls. A style specifies one or more of the following properties:

- font face
- font size
- font style
- foreground color
- background color

- background image

Designing styles for form controls

In the BlackBerry® Plug-in for Microsoft® Visual Studio®, when you assign a style to a form control, the form control reflects the properties of the style. A nested form control inherits the style of the parent control. You can apply a style to a nested control override specific properties.

Stylesheets used by the BlackBerry Plug-in for Microsoft Visual Studio are not CSS stylesheets used in web development.

Add a stylesheet

1. In the Solution Explorer, right-click the project name or the folder in which you want to add the stylesheet.
2. Select **Add > New Item**.
3. In the Visual Studio installed templates pane, click **Stylesheet**.
4. In the **Name** field, type the name of the stylesheet.
5. Click **Add**.

The stylesheet opens.

Add a style to a stylesheet

1. In the Solution Explorer, double-click the stylesheet.
2. In the stylesheet, click **New Style**.
If this is the first style you add to the stylesheet, a blank style is created when you create the stylesheet.
3. In the **Style Name** field, type the name of the style.
4. In the **left Font Name** drop-down list, specify the font face.
A preview of the font style appears in the box located below the style fields.
5. In the **right Font Name** drop-down list, specify the font size.
6. To specify bold, click **Bold**.
Bold face is not supported for all fonts.
7. To specify italic, click **Italic**.
8. To specify underline, click **Underline**.
9. To specify the font color, click the **Font Color** arrow and specify the color in the palette dialog box.
10. To specify the background color, click the **Background Color** arrow and specify the color in the palette dialog box.
11. To use an image as the background, specify one in the **Background Image** drop-down list.
12. To commit the changes, on the **File** menu, click **Save stylesheet_name.mds**.

Manage styles

1. In the Solution Explorer, double-click the stylesheet.

- In the stylesheet, click the style you want to edit in the **Available Styles** list.
- Perform any of the following actions:

Action	Procedure
Rename the style.	> In the Name field, type a new style name.
Change the font face.	> In the left Font Name drop-down list, specify a new font face.
Change the font size.	> In the right Font Name drop-down list, specify a new font size.
Turn bold on or off.	> Click Bold . Bold face is not supported for all fonts.
Turn italics on or off.	> Click Italic .
Turn underline on or off	> Click Underline .
Change the font color.	Click the Font color arrow. Specify the color in the palette dialog box.
Change the background color.	Click the Background color arrow. Specify the color in the palette dialog box.
Change the background image.	> In the Background Image drop-down list, specify a new image.
Remove the style.	> Click Remove .

- To commit the changes, on the **File** menu, click **Save stylesheet_name.mds**.

Adding hot keys

Hot keys enable a user to press a key to perform specific actions within a BlackBerry® MDS Runtime Application.

Setting hot keys

To add hot keys, perform the following actions:

- Click a control.
- In the Properties pane, click the **Events** button.
- In the **HotKey Press** field, click the key you want to configure.
- Click the drop-down list in the corresponding column to set the action for that key.

You can assign specific events to the following controls:

Control	Available hot keys
Button	All keys except for Enter.
Checkbox	All keys except for the Space bar.
CheckedListBox	All keys except for the Space bar.
ComboBox	All keys except for Enter and Backspace.
DataGridView	All keys.
DateTimePicker	The Enter key.
Form	A to Z and Backspace.

ListBox	The Enter key, the Space bar, and Backspace.
MaskedTextBox	The Enter key.
RadioGroup	All keys except for Enter and Backspace.
Repeater	All keys.
TextBox	The Enter key.
TextBox (Multi-line)	The Enter key and Shift.

Add graphical content

Graphic resources can be used by certain form controls and can be applied to styles.

The BlackBerry® Plug-in for Microsoft® Visual Studio® also supports Plazmic® content created with the Plazmic Content Developer's Kit. Visit <http://www.plazmic.com> for more information on Plazmic content.

You can add an image that you want to use in the project.

1. In the Solution Explorer, right-click the *Project Name* or the folder in which you want to add the image.
2. Click **Add > Existing Item**.
3. In the Add Existing Item dialog, in the **Files of type** drop-down list, specify the image file type.
4. Navigate to the image.
5. Click **Add**.

Managing communication

You can use two message delivery protocols, standard and best effort, depending on the level of reliability required by the application.

- **Standard:** Use this protocol if you need to guarantee that the message is delivered to the device. If the BlackBerry® device cannot receive the message (it is out of coverage), the message is delivered when the device can receive the message.
- **Best effort:** Use this protocol if you do not need to guarantee that the message is delivered. If the BlackBerry device cannot receive the message (it is out of coverage), the message is lost.

Change the message protocol

In the Solution Explorer, expand the project.

1. Expand the **Properties** folder.
2. Double click **Settings.mds**.
3. In the Settings file, in the **Message Delivery** drop-down list, specify the protocol.
4. To commit the changes, on the **File** menu, click **Save Settings.mds**.

Managing runtime errors

If an error occurs during the execution of an application, the BlackBerry® MDS Runtime invokes the `onError()` script. When you create a new BlackBerry MDS Runtime Application, the BlackBerry® Plug-in for Microsoft® Visual Studio® generates a default version of the `onError` script in the **Program.mds** file. You can customize the script.

The `onError()` script contains the supported error cases for each of the following categories:

- BlackBerry MDS Runtime errors
- BlackBerry® MDS Services errors
- SOAP connector errors

Change the standard error script

```
function onError() {
    switch (Error.category)
    {
        case Error.CATEGORY_MDS_RE_ERROR:
        {
            case Error.OUT_QUEUE_FULL:
                gvAllowSendMessage = false;
                break;
        }
    }
}
```

Integrating existing BlackBerry applications

Adding menu items inside BlackBerry built-in applications

BlackBerry® extensions enable you to directly access your BlackBerry® MDS Runtime Applications by extending menus in BlackBerry Applications. You can extend menus in the Address Book, Calendar, Email, and Tasks applications. These extended menu items automatically pass parameters to your application, depending on their source. For example, you can send an Event object from the Calendar application to a BlackBerry MDS Runtime Application, using extended menu items in the Calendar.

Applications and BlackBerry MDS object types are mapped as follows:

Application	BlackBerry MDS object type
Address Book	Contact
Calendar	Event
Email message or Message list	Message
Tasks	Task

Add menu items to built-in applications

1. To add BlackBerry® extensions, perform the following actions:
2. In the Solution Explorer, expand the project.
3. Expand the **Properties** folder.
4. Double-click **Settings.mds**.
5. Click the **BlackBerry Extensions** tab.

Perform the following actions:

Action	Procedure
Select the BlackBerry Application to extend.	> In the Menu Entry Location drop-down list, select the BlackBerry Application to which you want to add the menu item.
Set the label for the menu in the BlackBerry® MDS Runtime application.	> In the Display Label field, type a name for the menu item.
Set the BlackBerry MDS Runtime form to access.	> In the Application Entry point drop-down list, select the form to access in the BlackBerry MDS Runtime Application.

Save the **Settings.mds** file.

Integrate other applications

Invoke other BlackBerry® Applications from your BlackBerry® MDS Runtime Application using the System.exec function. Use this function to:

- Start a BlackBerry MDS Runtime application
- Start a BlackBerry application
- Start a Java® application

For more information on integrating other BlackBerry Applications, see the reference section of the online help.

Testing BlackBerry MDS Runtime Applications

To test a BlackBerry® MDS Runtime Application, start the debugger.

Configure break points

To configure break points in the script editor, perform one of the following actions:

Action	Procedure
Add a break point.	Right-click a line of code. Click Breakpoint > Insert Breakpoint .
Delete a break point.	Right-click a line of code that has a set break point. Click Breakpoint > Delete Breakpoint .
Disable a break point.	Right-click a line of code that has a set break point. Click Breakpoint > Disable Breakpoint .
Enable a break point.	Right-click a line of code that has a disabled break point. Click Breakpoint > Enable Breakpoint .

Start the debugger

> On the **Debug** menu, click **Start debugging**.

The BlackBerry® Plug-in for Microsoft® Visual Studio® performs the following actions in the order listed:

1. Launches the MDS development server
2. Launches the Blackberry Smartphone Simulator.
3. Publishes the BlackBerry® Runtime Application to the MDS development server.
4. Connects to the Blackberry Smartphone Simulator.
5. Removes older version of the BlackBerry Runtime Application if applicable.
6. Installs new version of the Blackberry Runtime Application.
7. Launches the Blackberry Runtime Application.

Debug the application in the BlackBerry Browser

1. On the **File** menu, click **Browse With**.
2. Select **BlackBerry Browser** and click **Browse**.

The BlackBerry® Plug-in for Microsoft® Visual Studio® performs the following actions in the order listed:

1. Launches the MDS development server
2. Launches the Blackberry Smartphone Simulator.
3. Connects to the Blackberry Smartphone Simulator.
4. Launches the browser application in the Blackberry® Browser and begins a debug session.

Output debug statements

To output debug statements using JavaScript®, use the syntax: `Debug.write("Message");`

Publishing BlackBerry MDS Runtime Applications


When you publish a BlackBerry® MDS Runtime Application, the BlackBerry® Plug-in for Microsoft® Visual Studio® generates a bundle. A bundle is a single .jar file containing the BlackBerry MDS Runtime Application files.

You require a registry and repository to publish a BlackBerry MDS Runtime Application. By default, the BlackBerry Plug-in for Microsoft Visual Studio sets preferences for a local registry and repository. To publish a BlackBerry MDS Runtime Application to a network server, you must configure the BlackBerry Plug-in for Microsoft Visual Studio to use the registry and repository specified by your system administrator.

Publish a project

Before publishing the application, make sure you are pointing the application to the correct implementation of the web service and not a test service running on your local machine.

See "Set the web service URL" on page 9 for more information.

1. Click the  button to start the BlackBerry® MDS Publishing wizard.
2. Click **Next**.
3. In the **Registry Alias** drop-down list, specify the BlackBerry MDS registry to which to publish. See "Set application registry preferences" on page 80 for more information.
4. In the BlackBerry MDS registry **Username** field, type a valid username for the registry.
5. In the BlackBerry MDS registry **Password** field, type the password for the username.
6. In the **Repository Alias** drop-down list, specify the BlackBerry MDS Registry to which to publish. See "Set application registry preferences" on page 80 for more information.
7. In the BlackBerry MDS Repository **Username** field, type a valid username for the registry.
8. In the BlackBerry MDS Repository **Password** field, type the password for the username.
9. Click **Next** and the wizard will attempt to connect to the registry and repository. If it is unsuccessful, check that your registry and repository settings are correct.
10. To add or manage searchable keywords for your application to make it easier to find in the BlackBerry MDS registry, perform the following actions:

Action	Procedure
Turn on or turn off searchable keywords.	<ul style="list-style-type: none"> > To turn on searchable keywords, select the Specify searchable keywords for your application check box. > To turn off searchable keywords, clear the Specify searchable keywords for your application check box.

Add a keyword.	<ol style="list-style-type: none"> 1. Click New. 2. In the Keyword dialog box, type a keyword. 3. Click OK.
Edit a keyword.	<ol style="list-style-type: none"> 1. Select the keyword. 2. Click Edit. 3. In the Keyword dialog box, type a new keyword. 4. Click OK.
Remove a keyword.	<ol style="list-style-type: none"> 1. Select the keyword. 2. Click Delete.

11. Click **Next**.
12. In the **Mandatory Display Description** field, type a description of the BlackBerry® MDS Runtime Application.
This field can be automatically populated by entering a description in the **Description** field in the **Settings.mds** file.
13. Click **Finish**.

Set application registry preferences

A registry is a service on the Internet or corporate intranet you can use to publish and search for applications. You register a BlackBerry® MDS Runtime Application in a BlackBerry MDS application registry. The registry is either a database or a Universal Description, Discovery, and Integration (UDDI) implementation, interlaced through a dedicated web service. The registry stores application-specific records of properties that you define when you publish the application.

To set the registry preferences, perform the following actions:

1. On the **Tools** menu, click **Options**.
2. Expand **BlackBerry**.
3. Click **Application Registries**.
4. Perform one of the following actions:

Action	Procedure
Add an application registry.	<ol style="list-style-type: none"> 1. Click Add. 2. In the New Registry dialog box, type the required information into the fields. 3. Click Test Configuration to make sure that you can connect to the Application Registry. If the test is unsuccessful, make sure you have entered the correct information. 4. Click OK.

Edit an application registry.	<ol style="list-style-type: none"> 1. In the Application Registries list, select the registry you want to edit . 2. Click Edit. 3. In the Edit Registry dialog box, edit the required fields. 4. Click Test Configuration to ensure that you can connect to the Application Registry. If the test is unsuccessful, make sure you have entered the correct information. 5. Click OK.
Delete an application registry.	<ol style="list-style-type: none"> 1. In the Application Registries list, select the registry you want to delete . 2. Click Delete.
Set a username and password.	<ol style="list-style-type: none"> 1. In the Application Registries list, select the registry for which you want to specify a user name and password for . 2. Click Edit. 3. In the Edit Registry dialog box, in the User Name field, type the user name. 4. In the Password field, type the password. 5. Click OK.
Change the default registry.	> In the Default Application Registry drop-down list, specify a new application registry.

5. Click **OK**.

Set application repository preferences

A repository is a location on the Internet or corporate intranet you can use to deposit a BlackBerry® MDS Runtime Application. You can use the BlackBerry® Provisioning System to download applications from a repository to your BlackBerry® device.

You store the BlackBerry MDS Runtime Application bundles that you want to distribute in a BlackBerry application repository. The BlackBerry® Plug-in for Microsoft® Visual Studio® supports Web-based Distribution Authoring and Versioning (WebDAV) implementation of repositories.

To set the repository preferences, perform the following actions:

1. On the **Tools** menu, click **Options**.
2. Expand **BlackBerry**.
3. Select **File Repositories**.
4. Perform one of the following actions:

Action	Procedure
Add a file repository.	<ol style="list-style-type: none"> 1. Click Add. 2. In the New Repository dialog box, type the required information into the fields. 3. Click Test Configuration to make sure that you can connect to the application repository. If the test is unsuccessful, make sure you have entered the correct information. 4. Click OK.

Edit a file repository.	<ol style="list-style-type: none">1. In the Application Repositories list, select the repository you want to edit2. Click Edit.3. In the Edit Repository dialog box, edit the required fields.4. Click Test Configuration to make sure that you can connect to the application repository. If the test is unsuccessful, make sure you have entered the correct information.5. Click OK.
Delete a file repository.	<ol style="list-style-type: none">1. Select the repository you want to delete in the Application Repositories list.2. Click Delete.
Set a username and password.	<ol style="list-style-type: none">1. In the Application Repositories list, select the repository for which you want to specify a user name and password.2. Click Edit.3. In the Edit Repository dialog box, in the User Name field, type the user name.4. In the Password field, type the password.5. Click OK.
Change the default repository.	> In the Default Application Repository drop-down list, specify a new application repository.

5. Click **OK**.

Change the version number of the application

The version number structure does not signify major and minor revisions. The first number changes when there are change made to the data definitions in the application. Because data definitions have changed, any data stored in persistent storage is deleted.

The second number changes when there are changes made to the messaging protocol. This has no effect on memory. The third number changes when there are any changes made to the functionality of the application. this number is automatically incremented for each build.

In the Solution Explorer, expand the project.

1. Expand the **Properties** folder.
2. Double-click **Settings.mds**.
3. In the **Version** fields, specify a version number.
4. To commit the changes, click **File > Save Settings.mds**.

Set the base URI for the application

If you change the URI on a published project, a new project is created with the new URI when you publish the project again. Changing the URI effectively breaks any association between the previous project and the new project.

You can use Uniform Resource Identifier (URI) to identify the application.

1. In the Solution Explorer, expand the project.

2. Expand the **Properties** folder.
3. Double-click **Settings.mds**.
4. In the **URI** field, type a new URI.
5. To commit the changes, click **File > Save Settings.mds**.

Publish a project using the standalone publishing utility

You can publish your applications using a freely distributable standalone publishing utility. The installer for this utility is in the **Redistributables** folder within the default install folder for BlackBerry® Plug-in for Microsoft® Visual Studio®. Run the installation on any computer using the Microsoft® .NET Framework 2.0.

1. Browse to **C:\Program Files\Research In Motion\BlackBerry VSx Plugin\Redistributables**.
2. Double-click **WinPublisher.exe** to install the utility.

After you install the publishing utility, use it to browse to the .jar file for your application, and then upload it to a BlackBerry® Enterprise Server.

