IBM MobileFirst Platform Foundation

IBM MobileFirst™ Platform Foundation V7.1.0

C# client-side API for native Windows 8 Universal and Windows Phone 8 Universal apps

14 August 2015

Copyright Notice

Trademarks

About IBM

See the About IBM page (www.ibm.com/ibm/us/en/).

Contents

Tables

About this document

This document is intended for developers who want to access IBM MobileFirst™ Platform Foundation services from Windows 8 Universal or Windows Phone 8 Universal applications that are written in C#. The document guides you through the available classes and methods.

1 API overview

The IBM MobileFirst Platform Foundation C# client-side API for native Windows 8 Universal and Windows Phone 8 Universal applications exposes two main capabilities:

- Calling back-end services to retrieve data and perform back-end transactions.
- Writing custom log lines for reporting and auditing purposes.

The IBM MobileFirst Platform Foundation C# client-side API for native Windows 8 Universal and Windows Phone 8 Universal applications is available as part of the IBM MobileFirst Platform Studio.

Type	Name	Description	Implemented By
Properties file	`wlclient.properties`	Properties file that contains the necessary data to connect to IBM MobileFirst Platform Server.	IBM
Class	`WLClient`	Singleton class that exposes methods to communicate with the MobileFirst Server, in particular `invokeProcedure` for calling a back-end service.	IBM
Class	`WLProcedure InvocationData`	Class that contains all data necessary to call a procedure.	IBM
Class	`WLRequestOptions`	Class that you use to add request parameters, headers, and invocation context.	IBM
Interface	`WLResponseListener`	Interface that defines methods that a listener for the `WLClient invokeProcedure` method implements to receive notifications about the success or failure of the method call.	Application developer
Class	`WLResponse`	Class that contains the result of a procedure invocation.	IBM

Type	Name	Description	Implemented By
Class	`WLFailResponse`	Class that extends `WLResponse`. This class contains error codes, messages, and the status in `WLResponse`. This class also contains the original response `DataObject` from the server.	IBM
Class	`WLProcedureInvocationResult`	Class that extends `WLResponse`. This class contains the result of calling a back-end service, which includes statuses and data items that the adapter function retrieves from the server.	IBM
Class	`WLProcedureInvocationFailResponse`	Class that extends `WLFailResponse` and that you can use to retrieve the invocation error messages.	IBM
Class	`WLErrorCode`	Class that contains an error code and its message that arrive from the MobileFirst Server.	IBM
Class	`BaseChallengeHandler`	Abstract base class for all challenge handlers.	IBM
Class	`ChallengeHandler`	Abstract class that must be extended to create custom challenge handlers.	IBM
Class	`WLChallengeHandler`	Abstract base class for MobileFirst challenge handlers. You must extend it to implement your own version of a MobileFirst challenge handler.	IBM
Class	`WLPush`	Class that contains all the methods that are required to work with Push notifications.	IBM
Interface	`WLOnReadyToSubscribeListener`	Interface that defines the method that is notified when a device is ready to subscribe.	Application developer

Type	Name	Description	Implemented By
Interface	`WLEventSourceListener`	Interface that defines the method that receives the notification message arrives from the subscribed event source.	Application developer
Interface	`WLNotificationListener`	Interface that defines the method that receives the notification.	Application developer
Class	`WLPushOptions`	Class that contains the subscription parameters that can be specified while you subscribe to push notifications.	IBM
Class	`WLAuthorizationManager`	Class that manages the entire OAuth flow, from client registration to token generation.	IBM
Class	`WLAuthorizationPersistencePolicy`	Enum that represents the authorization header persistence policy for `WLAuthorizationManager`. This policy controls whether or the authorization header is persisted across multiple runs of the application or not.	IBM
Class	`WLResourceRequest`	Class that encapsulates a resource request and provides several `send` methods, with different inputs for the body of a request.	IBM

Table 1-1: IBM MobileFirst Platform Foundation C# client-side API for Windows 8 Universal and Windows Phone 8 Universal applications – packages, classes, interfaces, and files

2 API reference

2.1 Example Code

The following code samples show how to use the IBM MobileFirst Platform Foundation C# client-side API for native Windows 8 Universal and Windows Phone 8 Universal applications.

2.1.1 Example: connecting to the MobileFirst Server and calling a procedure

Initializing the MobileFirst client

```
WLClient client = WLClient.getInstance();
client.connect(new MyConnectResponseListener());
```

Implementation of a Response Listener for connect

```
public class MyConnectResponseListener : WLResponseListener{
  public void onFailure(WLFailResponse response) {
    Debug.WriteLine("Response fail: " + response.getErrorMsg());
  }
  public void onSuccess(WLResponse response) {
    WLProcedureInvocationData invocationData = new
WLProcedureInvocationData("myAdapterName", "myProcedureName");
    invocationData.setParameters(new Object[]{"stringParam"});
    String myContextObject = new String("This is my context object");
    WLRequestOptions options = new WLRequestOptions();
    options.setInvocationContext(myContextObject);
    WLClient.getInstance().invokeProcedure(invocationData, new
MyInvokeListener(), options);
  }
}
```

Implementation of a Response Listener for Procedure Invocation

```
public class MyInvokeListener : WLResponseListener {
  public void onFailure(WLFailResponse response) {
    Debug.WriteLine("Response failed: " + response.getErrorMsg());
  }
  public void onSuccess(WLResponse response) {
```



```
    WLProcedureInvocationResult invocationResponse =
((WLProcedureInvocationResult) response);
    JObject items;
    try {
      items = invocationResponse.getResponseJSON();
      // do something with the items
    } catch (JSONException e) {
      }
  }
}
```

2.2 Class WLClient

This singleton class exposes methods that you use to communicate with the MobileFirst Server.

2.2.1 Method getInstance

Syntax

```
public static WLClient getInstance()
```

Description

This method gets the singleton instance of `WLClient`.

2.2.2 Method connect

Syntax

```
public void connect(WLResponseListener
responseListener)
```

Description

This method sends an initialization request to the MobileFirst Server, establishes a connection with the server, and validates the application version.

Important: You must call this method before any other `WLClient` methods that communicate with the MobileFirst Server.

Parameters

Type	Name	Description
`WLResponseListener`	`responseListener`	When the server returns a successful response, the `WLResponseListener` `onSuccess` method is called. If an error occurs, the `onFailure` method is called.

Table 2-1: Method connect parameter

2.2.3 Method connect

Syntax

```
public void connect(WLResponseListener
responseListener, WLRequestOptions requestOptions)
```

Description

This method sends an initialization request to the MobileFirst Server, establishes a connection with the server, and validates the application version.

Important: You must call this method before any other `WLClient` methods that communicate with the MobileFirst Server.

Parameters

Type	Name	Description
WLResponseListener	`responseListener`	When the server returns a successful response, the `WLResponseListener onSuccess` method is called. If an error occurs, the `onFailure` method is called.
WLRequestOptions	`requestOptions`	`WLRequestOptions` instance

Table 2-2: Method connect parameters

2.2.4 Method invokeProcedure

Syntax

```
public void invokeProcedure (
WLProcedureInvocationData invocationData,
WLResponseListener responseListener,
WLRequestOptions requestOptions)
```

```
public void invokeProcedure (
WLProcedureInvocationData invocationData,
WLResponseListener listener)
```

Description

This method sends an asynchronous call to an adapter procedure. The response is returned to the callback functions of the provided responseListener.

If the invocation succeeds, the `onSuccess` method is called. If the invocation fails, the `onFailure` method is called.

Parameters

Type	Name	Description
WLProcedure InvocationData	`invocationData`	The invocation data for the procedure call.

Type	Name	Description
WLResponseListener	responseListener	The listener object whose callback methods onSuccess and onFailure are called.
WLRequestOptions	requestOptions	Optional. Invocation options.

Table 2-3: Method invokeProcedure parameters

2.2.5 Method logActivity

Syntax

```
public void logActivity(String activityType)
```

Description

This method reports a user activity for auditing or reporting purposes.

The MobileFirst Server maintains a separate database table to store application statistics.

Important: Ensure that reports.exportRawData is set to `true` in the `worklight.properties` file. Otherwise, the activity is not stored in the database. You must also ensure that the following properties are entered in the `worklight.properties` file:

- `wl.reports.db.type`
- `wl.reports.db.url`
- `wl.reports.db.username`
- `wl.reports.db.password`

Parameters

Type	Name	Description
String	activityType	A string that identifies the activity type.

Table 2-4: Method logActivity parameters

2.2.6 Method setHeartbeatInterval

Syntax

```
public void setHeartbeatInterval(int value)
```

Description

This method sets the interval, in seconds, at which the MobileFirst Server sends the heartbeat signal. You use the heartbeat signal to ensure that the session with the server is kept alive when the app does not issue any call to the server, such as `invokeProcedure`.

By default, the interval is set to 7 minutes.

Parameters

Type	Name	Description
int	value	An interval value in seconds, at which the heartbeat signal is sent to MobileFirst Server.

Table 2-5: Method setHeartbeatInterval parameters

2.3 Class WLProcedureInvocationData

This class contains all data necessary to call a procedure, including the following elements:

- The names of the adapter and procedure to call.
- The parameters that the procedure requires.
- The optional `boolean` parameter to enable or disable compression. You can enable the compression by setting the vaue of the parameter to `true`. By default, the compression is not enable. However, the data is compressed when it exceeds the threshold value that is defined in the `worklight.properties` file. Use `compress.response.threshold` to define the threshold value in the `worklight.properties` file.

Example

```
// Set true to enable data compression.
    WLProcedureInvocationData invocationData = new
WLProcedureInvocationData("myAdapterName", "myProcedureName", true);
```

2.3.1 Method setParameters

Syntax

```
public void setParameters(Object [] parameters)
```

Description

This method sets the request parameters.

Parameters

Type	Name	Description
Object []	parameters	An array of objects of primitive types (`String`, `Integer`, `Float`, `Boolean`, `Double`). The order of the objects in the array is the order in which they are sent to the adapter.

Table 2-6: Method setParameters parameters

Example


```
invocationData.setParameters(new Object[]{"stringParam", true, 1.0,
1});
```

2.4 Class WLRequestOptions

This class contains the request parameters, headers, and invocation context.

2.4.1 Method addParameter

Syntax

```
public void addParameter(String name,String value)
```

Description

This method adds a request parameter with the given name and value.

Parameters

Type	Name	Description
String	name	The name of the parameter.
String	value	The value of the parameter.

Table 2-7: Method addParameter parameters

2.4.2 Method addParameters

Syntax

```
public void addParameters(Dictionary<String,
String> parameters)
```

Description

This method adds a table of request parameters.

Parameters

Type	Name	Description
Dictionary <String,String>	parameters	Request parameters table.

Table 2-8: Method addParameters parameters

2.4.3 Method getParameter

Syntax

```
public String getParameter(String name)
```

Description

This method returns the value of the parameter that is set.

Parameters

Type	Name	Description
String	name	The name of the parameter.

Table 2-9: Method getParameter parameters

2.4.4 Method getParameters

Syntax

```
public Dictionary<String,String> getParameters()
```

Description

This method returns the parameters table.

2.4.5 Method getResponseListener

Syntax

```
public WLResponseListener getResponseListener()
```

Description

This method returns the response listener for this request.

2.4.6 Method addHeader

Syntax

```
public void addHeader(String name, String value)
```

Description

This method adds a header with the given name and value.

Parameters

Type	Name	Description
String	name	The name of the header.
String	value	The value of the header.

Table 2-10: Method addHeader parameters

2.4.7 Method setHeaders

Syntax

```
public void setHeaders(WebHeaderCollection
extraHeaders)
```

Description

This method sets the request with the given headers.

Parameters

Type	Name	Description
WebHeaderCollection	extraHeaders	The headers to be set.

Table 2-11: Method setHeaders parameters

2.4.8 Method getHeaders

Syntax

```
public WebHeaderCollection getHeaders()
```

Description

This method returns the headers that are set for this request.

2.4.9 Methods getInvocationContext, setInvocationContext

Syntax

```
public Object getInvocationContext()
```

```
public void setInvocationContext(Object
invocationContext)
```

Parameters

Type	Name	Description
Object	`invocationContext`	An object that is returned with `WLResponse` to the listener methods `onSuccess` and `onFailure`. You can use this object to identify and distinguish different `invokeProcedure` calls. This object is returned as is to the listener methods.

Table 2-12: Methods getInvocationContext, setInvocationContext parameter

2.4.10 Method getAppUserId

Syntax

```
public String getAppUserId()
```

Description

This method returns the application userId that is used by the Push service.

2.4.11 Method setAppUserId

Syntax

```
public void setAppUserId(java.lang.String
appUserId)
```

Description

This method sets the application userId that is used by the Push server.

Parameters

Type	Name	Description
java.lang.String	appUserId	The application user identity.

Table 2-13: Method setAppUserId parameter

2.5 Interface WLResponseListener

This interface defines methods that the listener for the `WLClient.invokeProcedure` method implements to receive notifications about the success or failure of the method call.

2.5.1 Method onSuccess

Syntax

```
public void onSuccess (WLResponse response)
```

Description

This method is called after successful calls to the `WLClient connect` or `invokeProcedure` methods.

Parameters

Type	Name	Description
WLResponse	response	The response that the server returns, along with any invocation context object and status.

Table 2-14: Method onSuccess parameters

2.5.2 Method onFailure

Syntax

```
public void onFailure (WLFailResponse response)
```

Description

This method is called if any failure occurred during the execution of the `WLClient connect` or `invokeProcedure` methods.

Parameters

Type	Name	Description
WLFailResponse	response	A response that contains the error code and error message. Optionally, this response contains the results from the server, and any invocation context object and status.

Table 2-15: Method onFailure parameters

2.6 Class WLResponse

This class contains the result of a procedure invocation. IBM MobileFirst Platform Foundation passes this class as an argument to the listener methods of the `WLClient invokeProcedure` method.

2.6.1 Method getStatus

Syntax

```
public HttpStatusCode getStatus()
```

Description

This method retrieves the `HTTP` status from the response.

2.6.2 Method getInvocationContext

Syntax

```
public Object getInvocationContext()
```

Description

This method retrieves the invocation context object that is passed when the `invokeProcedure` method is called.

2.6.3 Method getResponseText

Syntax

```
public String getResponseText()
```

Description

This method retrieves the original response text from the server.

2.6.4 Method getResponseJSON

Syntax

```
public JObject getResponseJSON()
```

Description

This method retrieves the response text from the server in JSON format.

2.7 Class WLFailResponse

This class extends `WLResponse`. This class contains error codes, messages, the status in `WLResponse`, and the original response `DataObject` from the server.

2.7.1 Method getErrorCode

Syntax

```
public WLErrorCode getErrorCode ()
```

Description

The `WLErrorCode` section describes the possible errors.

2.7.2 Method getErrorMsg

Syntax

```
public String getErrorMsg()
```

Description

This method returns an error message that is for the developer and not necessarily suitable for the user.

2.8 Class WLProcedureInvocationResult

This class extends `WLResponse`. This class contains statuses and data that an adapter procedure retrieves.

2.8.1 Method getResult

Syntax

```
public JObject getResult()
```

Description

This method returns a `JObject` that represents the JSON response from the server.

2.8.2 Method isSuccessful

Syntax

```
public boolean isSuccessful()
```

Description

This method returns `true` if the procedure invocation was technically successful. Application errors are returned as part of the retrieved data, and not in this flag.

2.9 Class WLProcedureInvocationFailResponse

This class extends `WLFailResponse`. This class contains statuses and data that an adapter procedure retrieves.

2.9.1 Method getProcedureInvocationErrors

Syntax

```
public List<String> getProcedureInvocationErrors()
```

Description

This method returns a list of applicative error messages that are collected while the procedure is called.

2.9.2 Method getResult

Syntax

```
public JObject getResult()
```

Description

This method returns a `JObject` that represents the JSON response from the server.

2.10 Class WLErrorCode

This class contains the error code and its description that the server returns.

2.10.1 Method getDescription

Syntax

```
public String getDescription()
```

Description

This method returns the description of this error code instance.

2.10.2 Method valueOf

Syntax

```
public static WLErrorCode valueOf(String errorCode)
```

Description

This method returns the error code instance of the `errorCode` that is given.

Error Codes

`UNEXPECTED_ERROR` – Unexpected `errorCode` occurred. Please try again.

`REQUEST_TIMEOUT` – Request timed out.

`UNRESPONSIVE_HOST` – The service is currently unavailable.

`PROCEDURE_ERROR` – Procedure invocation `errorCode`.

`PROCEDURE_PROTECTED_ERROR` – Procedure is protected.

`APP_VERSION_ACCESS_DENIAL` – Application version denied.

`APP_VERSION_ACCESS_NOTIFY` – Notify application version changed.

2.11 Class BaseChallengeHandler

This class is an abstract base class for all challenge handlers.

2.11.1 Constructor

Syntax

```
public BaseChallengeHandler(String realm)
```

Description

This method creates a `BaseChallengeHandler` object for a particular `realm`.

2.11.2 Method handleChallenge

Syntax

```
public abstract void handleChallenge(T challenge)
```

Description

This method must be implemented by the subclass to handle the challenge logic. For example, show a login form in a challenge from a `FormBasedAuthenticator`.

2.11.3 Method submitFailure

Syntax

```
protected void submitFailure(WLResponse wlReponse)
```

Description

You must call this method when the challenge is answered with an error. The method is inherited from `BaseChallengeHandler`. Calling this method tells IBM MobileFirst Platform Foundation that the challenge was unsuccessful and that you no longer want to take any actions to attempt to resolve the problem. This method returns control to IBM MobileFirst Platform Foundation for further handling. For example, call this method only when you know that several authentication attempts were unsuccessful and you do not want the user to continue attempting to authenticate into the realm.

2.12 Class ChallengeHandler

This class is an abstract class that you must extend to create custom challenge handlers.

2.12.1 Constructor

Syntax

```
public ChallengeHandler(String realmName)
```

Description

This method creates a `ChallengeHandler` object for a particular `realm`.

2.12.2 Method isCustomResponse

Syntax

```
public abstract bool isCustomResponse(WLResponse
response)
```

Description

You must implement this method to return whether a response from the server is a challenge for this `ChallengeHandler`. The implementation must parse the response to determine whether or not the response is a challenge for this `ChallengeHandler`. For example, a `ChallengeHandler` for a realm with a form-based authenticator must parse the response to search for the `j_security_test` parameter and return `true` if found.

2.12.3 Method submitSuccess

Syntax

```
protected void submitSuccess(WLResponse response)
```

Description

You must call this method from the subclass within the `onSuccess` of your `ChallengeHandler`.

2.12.4 Method submitLoginForm

Syntax

```
protected void submitLoginForm(String requestURL,
Dictionary<String, String> requestParameters,
Dictionary<String, String> requestHeaders, int
requestTimeoutInMs, String requestMethod)
```

Description

This helper method submits a login form by making an `HTTP` request to the specified requestURL.

Parameters

Type	Name	Description
String	`requestURL`	The full or relative URL to which the request must be made.
Dictionary <String, String>	`requestParameters`	A `Dictionary` object with name-value pairs of request parameters.
Dictionary <String, String>	`requestHeaders`	A `Dictionary` object consisting of the additional headers that must be sent along with the `HTTP` request
int	`requestTimeoutInMs`	The time in milliseconds the request must wait before timing out.
String	`requestMethod`	The method to use. Specify `get` or `post`.

Table 2-16: Method submitLoginForm parameters

2.12.5 Method submitAdapterAuthentication

Syntax

```
protected void submitAdapterAuthentication(String
WLProcedureInvocationData invocationData,
WLRequestOptions requestOptions)
```

Description

This helper method submits a response to a challenge made by an `AdapterAuthenticator` by using an `invokeProcedure` call to the adapter procedure.

Parameters

Type	Name	Description
WLProcedureInvocationData	invocationData	The WLProcedureInvocationData object that contains the name of the adapter, the procedure, and an optional parameter to enable or disable compression of the adapter response.
WLRequestOptions	requestOptions	A WLRequestOptions object with request options.

Table 2-17: Method submitAdapterAuthentication parameters

2.13 Class WLChallengeHandler

This class is an abstract base class for MobileFirst challenge handlers. You must extend it to implement your own version of a MobileFirst challenge handler, for example, the RemoteDisableChallengeHandler.

2.13.1 Constructor

Syntax

```
public WLChallengeHandler(String realm)
```

Description

This method creates a WLChallengeHandler object for a particular realm.

2.13.2 Method submitChallengeAnswer

Syntax

```
public void submitChallengeAnswer(Object answer)
```

Description

Sends the answer back to the server.

2.14 Class WLPush

This class contains all the methods that are required to work with Push notifications. You cannot instantiate this class directly. To get a reference to this class, use the getPush() method of WLClient.

To enable Push notifications, add the pushSender element to the application descriptor of your native API application.

```
<nativeWindows8App>
…
<pushSender clientSecret="wns secret key" packageSID="wns unique identifier"/>
```



```
...
</nativeWindows8App>
```

2.14.1 Method registerEventSourceCallback

Syntax

```
public void registerEventSourceCallback(String
alias, String adapter, String eventSource,
WLEventSourceListener eventSourceListener)
```

Description

This method registers a `WLEventSourceListener` that is called whenever a notification arrives from the specified event source.

Parameters

Type	Name	Description
String	`alias`	Mandatory string. A short ID that you use to identify the event source when the push notification arrives. You can provide a short alias, rather than the full names of the adapter and event source. This action frees space in the notification text, which is limited in length.
String	`adapter`	Mandatory string. The name of the adapter that contains the event source
String	`eventSource`	Mandatory string. The name of the event source.
WLEventSource Listener	`eventSource Listener`	Mandatory listener class. When a notification arrives, the `WLEventSourceListener.onReceive()` method is called.

Table 2-18: Method registerEventSourceCallback parameters

2.14.2 Method subscribe

Syntax

```
public void subscribe(String alias, WLPushOptions
pushOptions, WLResponseListener respListener)
```

Description

This method subscribes the user to the event source with the specified alias.

Parameters

Type	Name	Description
String	`alias`	Mandatory string. The event source alias, as defined in `registerEventSourceCallBack`.

Type	Name	Description
WLPushOptions	pushOptions	This instance contains the custom subscription parameters that the event source in the adapter supports.
WLResponseListener	respListener	The listener object whose callback methods are called by the MobileFirst runtime when a subscribe call succeeds or fails.

Table 2-19: Method subscribe parameters

2.14.3 Method unsubscribe

Syntax

```
public void unsubscribe(String alias,
WLResponseListener respListener)
```

Description

This method unsubscribes the user from the event source with the specified alias.

Parameters

Type	Name	Description
String	alias	Mandatory string. The event source alias, as defined in `registerEventSourceCallBack`.
WLResponseListener	respListener	The listener object whose callback methods are called by the MobileFirst runtime when a subscribe call succeeds or fails.

Table 2-20: Method unsubscribe parameters

2.14.4 Method isSubscribed

Syntax

```
public void isSubscribed(String alias)
```

Description

This method returns whether the currently logged-in user is subscribed to the specified event source alias.

Parameters

Type	Name	Description
String	alias	Mandatory string. The event source alias.

Table 2-21: Method isSubscribed parameters

2.14.5 Method subscribeTag

Syntax


```
public void subscribeTag(String tagName,
WLPushOptions pushOptions, WLResponseListener
respListener)
```

Description

This method subscribes the device to the tag.

Parameters

Type	Name	Description
String	tagName	Mandatory string. The name of the tag.
WLPushOptions	pushOptions	This instance contains the custom subscription parameters that the event source in the adapter supports.
WLResponseListener	respListener	The listener object whose callback methods are called by the MobileFirst runtime when a subscribe call succeeds or fails.

Table 2-22: Method subscribeTag parameters

2.14.6 Method unsubscribeTag

Syntax

```
public void unsubscribeTag(String tagName,
WLResponseListener respListener)
```

Description

This method unsubscribes the device from the tag.

Parameters

Type	Name	Description
String	tagName	Mandatory string. The name of the tag.
WLResponseListener	respListener	The listener object whose callback methods are called by the MobileFirst runtime when a subscribe call succeeds or fails.

Table 2-23: Method unsubscribeTag parameters

2.14.7 Method isTagSubscribed

Syntax

```
public void isTagSubscribed(String alias)
```

Description

This method returns whether the device is subscribed to the specified tag.

Parameters

Type	Name	Description
`String`	`tagName`	Mandatory string. The name of the tag.

Table 2-24: Method isTagSubscribed parameters

2.14.8 Property onReadyToSubscribeListener

Type

`WLOnReadyToSubscribeLister`

Access

`Read/Write`

Description

This property sets the `WLOnReadyToSubscribeListener` callback to be notified when the device is ready to subscribe to push notifications.

2.14.9 Property notificationListener

Type

`WLNotificationListener`

Access

`Read/Write`

Description

This property sets the `WLNotificationListener` callback to be notified when the push notification arrives.

2.15 Interface WLOnReadyToSubscribeListener

This interface defines the method that is notified when a device is ready to subscribe.

2.15.1 Method onReadyToSubscribe

Syntax

```
void onReadyToSubscribe()
```

Description

This method is called when the device is ready to subscribe to push notifications.

2.16 Interface WLEventSourceListener

This interface defines the method that receives the notification message.

2.16.1 Method onReceive

Syntax

```
void onReceive(String properties, String payload)
```


Description

This method is called when the notification arrives from the subscribed event source.

Parameters

Type	Name	Description
String	properties	A JSON block that contains the notifications properties of the platform.
String	payload	A JSON block that contains other data that is sent from the MobileFirst Server.

Table 2-25: Method onReceive parameters

2.17 Interface WLNotificationListener

This interface defines the method that receives the notification message.

2.17.1 Method onMessage

Syntax

```
void onMessage(String properties, String payload)
```

Description

This method is called when a push notification arrives.

Parameters

Type	Name	Description
String	properties	A JSON block that contains the notifications properties of the platform.
String	payload	A JSON block that contains other data that is sent from the MobileFirst Server. It also contains the tag name for tag and broadcast notification. The tag name appears in the “tag” element. The default tag name for broadcast notification is `Push.ALL`.

Table 2-26: Method onMessage parameters

2.18 Class WLPushOptions

This class contains the subscription parameters that can be specified while you subscribe to push notifications.

2.18.1 Constructor

Syntax

```
public WLPushOptions()
```

Description

This constructor creates a `WLPushOptions` object.

2.18.2 Method AddSubscriptionParameter

Syntax

```
public void AddSubscriptionParameter(String name,
String value)
```

Description

This method adds a subscription parameter.

Parameters

Type	Name	Description
String	name	Mandatory. The name of the subscription parameter.
String	value	Mandatory. The value of the subscription parameter.

Table 2-27: Method AddSubscriptionParameter parameters

2.18.3 Method GetSubscriptionParameter

Syntax

```
public void GetSubscriptionParameter(String name)
```

Description

This method returns the map that contains the subscription parameters.

Parameters

Type	Name	Description
String	name	Mandatory. The name of the subscription parameter.

Table 2-28: Method GetSubscriptionParameter parameters

2.18.4 Property subscriptionParameters

Type

```
Dictionary <String, String>
```

Access

```
Read/Write
```

Description

This property gets/sets the subscription parameters.

2.19 Class WLAuthorizationManager

This class manages the entire OAuth flow, from client registration to token generation.

2.19.1 Method getInstance

Syntax


```
public static WLAuthorizationManager getInstance()
```

Description

This method gets the singleton instance of `WLAuthorizationManager`.

2.19.2 Method setAuthorizationPersistencePolicy

Syntax

```
public void setAuthorizationPersistencePolicy
(WLAuthorizationPersistencePolicy policy)
```

Description

This method sets the authorization header persistence policy.

Parameters

Type	Name	Description
WLAuthorizationPersistencePolicy	`policy`	Mandatory. The authorization header persistence policy.

Table 2-29: Method setAuthorizationPersistencePolicy parameter

2.19.3 Method getAuthorizationPersistencePolicy

Syntax

```
public WLAuthorizationPersistencePolicy
getAuthorizationPersistencePolicy()
```

Description

This method gets the current authorization header persistence policy.

2.19.4 Method getAuthorizationScope

Syntax

```
public System.String
getAuthorizationScope(System.Net.HttpWebResponse
response)
```

Description

Returns the scope that is required by the resource that produced the given response. The scope is returned as String.

This method expects to be given only response objects for which the method `isAuthorizationRequired(HttpWebResponse)` returns `true`.

Parameters

Type	Name	Description
HttpWebResponse	`response`	The `HTTP` response object.

Table 2-30: Method getAuthorizationScope parameter

2.19.5 Method getAuthorizationScope

Syntax

```
public System.String
getAuthorizationScope(System.String authenticationH
eader)
```

Description

Returns the scope that is required by the resource that produced the given response. The scope is returned as String.

This method expects to be given only headers from response objects for which the method `isAuthorizationRequired(HttpWebResponse)` returns `true`.

Parameters

Type	Name	Description
`String`	`authenticationHeader`	The value of the authentication header.

Table 2-31: Method getAuthorizationScope parameter

2.19.6 Method getAppIdentity

Syntax

```
public JObject getAppIdentity()
```

Description

This method retrieves the application identity and returns the JObject that contains the application identity. It returns null if the application identity is not currently available.

2.19.7 Method getDeviceIdentity

Syntax

```
public JObject getDeviceIdentity()
```

Description

This method retrieves the device identity and returns the JObject that contains the device identity. It returns null if the device identity is not currently available.

2.19.8 Method getUserIdentity

Syntax

```
public JObject getUserIdentity()
```

Description

This method retrieves the user identity and returns the JObject that contains the user identity. It returns null if the user identity is not currently available.

2.19.9 Method isAuthorizationRequired

Syntax


```
public bool
isAuthorizationRequired(System.HttpWebResponse
response)
```

Description

This method checks whether the response is a MobileFirst OAuth error. If the reponse is indeed a MobileFirst OAuth error, the method returns `true`. Otherwise, it returns `false`.

Parameters

Type	Name	Description
HttpWebResponse	response	The HTTP response object.

Table 2-32: Method isAuthorizationRequired parameter

2.19.10 Method isAuthorizationRequired

Syntax

```
public bool isAuthorizationRequired(int status,
System.String authenticationHeader)
```

Description

This method checks whether a response with the given status and given string as the authentication header is an MFP OAuth Error. If it is indeed a MobileFirst OAuth error, the method returns `true`. Otherwise, it returns `false`.

Parameters

Type	Name	Description
int	status	The HTTP status.
String	authenticationHeader	The value of the authentication header.

Table 2-33: Method isAuthorizationRequired parameters

2.19.11 Method addCachedAuthorizationHeader

Syntax

```
public void
addCachedAuthorizationHeader(System.Net.HttpWebRequ
est request)
```

Description

This method adds the cached authorization header to the given HTTP request object.

Parameters

Type	Name	Description
HttpWebRequest	request	The HTTP request object to which to add header.

Table 2-34: Method addCacheAuthorizationHeader parameter

2.19.12 Method obtainAuthorizationHeader

Syntax

```
public void obtainAuthorizationHeader(System.String
scope, WLResponseListener listener)
```

Description

This method makes an explicit call to obtain the authorization header.

Parameters

Type	Name	Description
String	scope	The scope that is required.
WLResponseListener	listener	The WLResponseListener whose onSuccess or onFailure methods are called when the request finishes.

Table 2-35: Method obtainAuthorizationHeader parameters

2.19.13 Method getCachedAuthorizationHeader

Syntax

```
public System.String getCachedAuthorizationHeader()
```

Description

This method retrieves the cached authorization header and returns as String.

If there is no cached header, this method returns an empty string.

2.20 Class WLAuthorizationPersistencePolicy

This enum represents the authorization header persistence policy for WLAuthorizationManager. This policy controls whether or the authorization header is persisted across multiple runs of the application or not.

It can be one of the followings:

- ALWAYS - If this policy is set, the authorization header will always be persisted.
- NEVER - If this policy is set, the authorization header will never be persisted. This means that it will be stored only in memory and be lost when the application closes.

2.21 Class WLResourceRequest

This class encapsulates a resource request. The resource might be an adapter on the MobileFirst Server, or an external resource. The class provides several send methods, with different inputs for the body of a request. In addition, the send methods support two types of response listener.

- WLResponseListener - The onSuccess method of this listener is called and provided with an instance of the WLResponse class. The content of the response is to be

read into the `WLResponse` instance by the platform, and will be accessible through methods of `WLResponse`. In a failure, the `onFailure` method of the listener is called and provided with an instance of the `WLFailResponse` class that contains all the information about the failure.

- `WLHttpResponseListener` - The `onSuccess` method of this listener is called and provided with the original `HTTP` response object that was received from the server. The platform does not attempt to read or parse the response in any way. In a failure the `onFailure` method is called and provided with either the response from the server if one was received, or the exception that was thrown during the execution of this request.

Regardless of what type of listener was used, a successful response is any response with a status in the 2xx range. These responses are delivered to the `onSuccess` method. A response with a 4xx or 5xx status is considered a failure, and is delivered to the `onFailure` method.

2.21.1 Constructor

Syntax

```
public WLResouceRequest(String url, String method,
double timeout)
```

Description

This constructor constructs a new resource request with the specified URL, by using the specified HTTP method. Additionally this constructor sets a custom timeout.

Parameters

Type	Name	Description
String	`url`	The resource URL. Could be relative or absolute.
String	`method`	The `HTTP` method to use.
double	`Timeout`	Optional. The timeout in seconds for this request.

Table 2-36: Constructor WLResourceRequest parameters

Throws

`InvalidOperationException` – If the method name is not one of the valid HTTP method names.

2.21.2 Constructor

Syntax

```
public WLResourceRequest(Uri url, String method,
double timeout)
```

Description

This constructor constructs a new resource request with the specified URL, by using the specified HTTP method. Additionally this constructor sets a custom timeout.

Parameters

Type	Name	Description
Uri	url	The resource URL. Could be relative or absolute.
String	method	The HTTP method to use.
double	Timeout	Optional. The timeout in seconds for this request.

Table 2-37: Constructor WLResourceRequest parameters

Throws

InvalidOperationException – If the method name is not one of the valid HTTP method names.

2.21.3 Method getUrl

Syntax

```
public Uri getUrl()
```

Description

This method returns the Uri for this resource request. The Uri that is returned by this method is always the absolute Uri.

2.21.4 Method getMethod

Syntax

```
public String getMethod()
```

Description

This method returns the HTTP method for this resource request.

2.21.5 Method getQueryParameters

Syntax

```
public Dictionary<String, String>
getQueryParameters()
```

Description

This method returns the query parameters set for this resource request.

2.21.6 Method setQueryParameters

Syntax

```
public void setQueryParameters(Dictionary<String,
String> parameters)
```

Description

This method sets the query parameters for this resource request.

Parameters

Type	Name	Description
`Dictionary<String, String>`	`parameters`	A Dictionary containing query parameters.

Table 2-38: Method setQueryParameters parameter

2.21.7 Method setQueryParameter

Syntax

```
public void setQueryParameter(String name, String value)
```

Description

This method sets the value of the given query parameter name to the given value. If no such parameter exists, it will be added.

Parameters

Type	Name	Description
`String`	`name`	The name of the parameter to set.
`String`	`value`	The value of the parameter to set.

Table 2-39: Method setQueryParameter parameters

2.21.8 Method getHeaderNames

Syntax

```
public String[] getHeaderNames()
```

Description

This method returns the names of all the headers that were set for this resource request.

2.21.9 Method getAllHeaders

Syntax

```
public HttpHeaders getAllHeaders()
```

Description

This method returns all the headers that were set for this resource request.

2.21.10 Method getHeaders

Syntax

```
public String getHeaders(String headerName)
```

Description

This method returns all the headers for this resource request that have the given name.

Parameters

Type	Name	Description
String	headerName	The name of the headers to be returned.

Table 2-40: Method getHeaders parameter

2.21.11 Method getFirstHeader

Syntax

```
public String getFirstHeader(String headerName)
```

Description

This method returns the first header for this resource request with the given name.

Parameters

Type	Name	Description
String	headerName	The name of the first header to be returned.

Table 2-41: Method getFirstHeader parameter

2.21.12 Method removeHeader

Syntax

```
public void removeHeader(String headerName)
```

Description

This method removes the header for this resource request with the given name.

Parameters

Type	Name	Description
String	headerName	The name of the header to be removed.

Table 2-42: Method removeHeader parameter

2.21.13 Method setHeaders

Syntax

```
public void setHeaders(Dictionary<String, String>
headers)
```

Description

This method sets the headers for this resource request. If any of the headers had already been set, the new value overwrites the previous one.

Parameters

Type	Name	Description
Dictionary< String, String>	headers	The header as Dictionary object.

Table 2-43: Method setHeaders parameter

2.21.14 Method setHeader

Syntax

```
public void setHeader(String headerName, String
headerValue)
```

Description

This method sets the headers for this resource request. If the header exists, then the new value overwrites the previous one.

Parameters

Type	Name	Description
String	headerName	The name of header to set for this resource request.
String	headerValue	The value corresponding to headerName parameter.

Table 2-44: Method setHeader parameters

2.21.15 Method getTimeout

Syntax

```
public double getTimeout()
```

Description

This method returns the timeout in seconds for this resource request.

2.21.16 Method setTimeout

Syntax

```
public void setTimeout(double timeout)
```

Description

This method sets the timeout in second for this resource request.

Parameters

Type	Name	Description
double	timeout	The timeout for this resource request.

Table 2-45: Method setTimeout parameter

2.21.17 Method send

Syntax


```
public void send(WLResponseListener listener)
```

Description

This method sends this resource request asynchronously, without a request body.

Parameters

Type	Name	Description
WLResponseListener	listener	The listener whose onSuccess or onFailure methods is called when this request finishes.

Table 2-46: Method send parameter

2.21.18 Method send

Syntax

```
public void send(WLHttpResponseListener listener)
```

Description

This method sends this resource request asynchronously, without a request body.

Parameters

Type	Name	Description
WLHttpResponseListener	listener	The listener whose onSuccess or onFailure methods is called when this request finishes.

Table 2-47: Method send parameter

2.21.19 Method send

Syntax

```
public void send(String requestBody,
WLResponseListener listener)
```

Description

This method sends this resource request asynchronously, with a given string as the request body. If no content type header was set, this method will set it to `text/plain`.

Parameters

Type	Name	Description
String	requestBody	The request body text.
WLResponseListener	listener	The listener whose onSuccess or onFailure methods is called when this request finishes.

Table 2-48: Method send parameters

2.21.20 Method send

Syntax


```
public void send(Dictionary<String, String>
formParameters, WLResponseListener listener)
```

Description

This method sends this resource request asynchronously, with a given form of parameters as the request body. If no content type header was set, this method will set it to `application/x-www-form-urlencoded`.

Parameters

Type	Name	Description
`Dictionary<String, String>`	`formParameters`	The parameters to put in the request body.
`WLResponseListener`	`listener`	The listener whose `onSuccess` or `onFailure` methods is called when this request finishes.

Table 2-49: Method send parameters

2.21.21 Method send

Syntax

```
public void send(Dictionary<String, String>
formParameters, WLHttpResponseListener listener)
```

Description

This method sends this resource request asynchronously, with a given form of parameters as the request body. If no content type header was set, this method will set it to `application/x-www-form-urlencoded`.

Parameters

Type	Name	Description
`Dictionary<String, String>`	`formParameters`	The parameters to put in the request body.
`WLHttpResponseListener`	`listener`	The listener whose `onSuccess` or `onFailure` methods is called when this request finishes.

Table 2-50: Method send parameters

2.21.22 Method send

Syntax

```
public void send(JObject json, WLResponseListener
listener)
```

Description

This method sends this resource request asynchronously, with a given JSON object as the request body. If no content type header was set, this method will set it to `application/json`.

Parameters

Type	Name	Description
JObject	`json`	The JSON object to put in the request body.
WLResponseListener	`listener`	The listener whose `onSuccess` or `onFailure` methods is called when this request finishes.

Table 2-51: Method send parameters

2.21.23 Method send

Syntax

```
public void send(JObject json,
WLHttpResponseListener listener)
```

Description

This method sends this resource request asynchronously, with a given JSON object as the request body. If no content type header was set, this method will set it to `application/json`.

Parameters

Type	Name	Description
JObject	`json`	The JSON object to put in the request body.
WLHttpResponseListener	`listener`	The listener whose `onSuccess` or `onFailure` methods is called when this request finishes.

Table 2-52: Method send parameters

2.21.24 Method send

Syntax

```
public void send(byte[] data, WLResponseListener
listener)
```

Description

This method sends this resource request asynchronously, with the content of the given byte array as the request body.

Note: this method does not set any content type header. If such header is required, it must be set before you call this method.

Parameters

Type	Name	Description
byte[]	`data`	The byte array containing the request body.
WLResponseListener	`listener`	The listener whose `onSuccess` or `onFailure` methods is called when this request finishes.

Table 2-53: Method send parameters

2.21.25 Method send

Syntax


```
public void send(byte[] data,
WLHttpResponseListener listener)
```

Description

This method sends this resource request asynchronously, with the content of the given byte array as the request body.

Note: this method does not set any content type header. If such header is required, it must be set before you call this method.

Parameters

Type	Name	Description
byte[]	data	The byte array containing the request body.
WLHttpResponseListener	listener	The listener whose onSuccess or onFailure methods is called when this request finishes.

Table 2-54: Method send parameters

Appendix A - Notices

This information was developed for products and services that are offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive, MD-NC119

Armonk, NY 10504-1785

United States of America

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing

Legal and Intellectual Property Law

IBM Japan Ltd.

19-21, Nihonbashi-Hakozakicho, Chuo-ku

Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

Dept F6, Bldg 1

294 Route 100

Somers NY 10589-3216

USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These

examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. _enter the year or years_. All rights reserved.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is

collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect session information (generated by the application server). These cookies contain no personally identifiable information and are required for session management. Additionally, persistent cookies may be randomly generated to recognize and manage anonymous users. These cookies also contain no personally identifiable information and are required.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent. For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy (www.ibm.com/privacy) and IBM's Online Privacy Statement (www.ibm.com/privacy/details) the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" (www.ibm.com/software/info/product-privacy).

Appendix B - Support and comments

For the entire IBM MobileFirst™ documentation set, training material and online forums where you can post questions, see the IBM® MobileFirst Platform Getting Started website (www.ibm.com/mobile-docs).

Support

Software Subscription and Support (also referred to as Software Maintenance) is included with licenses purchased through Passport Advantage® and Passport Advantage Express. For additional information about the International Passport Advantage Agreement and the IBM International Passport Advantage Express Agreement, visit the Passport Advantage website (www.ibm.com/software/passportadvantage).

If you have a Software Subscription and Support in effect, IBM provides you assistance for your routine, short duration installation and usage (how-to) questions, and code-related questions. For additional details, consult your IBM Software Support Handbook (www.ibm.com/support/handbook).

Comments

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this document. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state.

Thank you for your support.

If you would like a response from IBM, please provide the following information:

- Name
- Address
- Company or Organization
- Phone No.
- Email address

