IBM Worklight

# IBM Worklight V6.1.0

## Objective-C client-side API for native iOS apps

17 March 2014

## About IBM®

See http://www.ibm.com/ibm/us/en/.

# Contents

## Tables

## About this document

This document is intended for iPhone and iPad developers who want to access IBM® Worklight® services from native iOS applications written in Objective-C. The document guides you through the classes and methods available.

# 1 API overview

The IBM Worklight Objective-C client-side API for native iOS apps exposes four main capabilities:

- Calling back-end services to retrieve data and perform back-end transactions.
- Writing custom log lines for reporting and auditing purposes.
- Writing custom Challenge Handlers to create user authentication.
- Subscribing and unsubscribing to push notifications.

| Type | Name | Description | Implemented By |
|---|---|---|---|
| File | `worklight.plist` | Property list file that contains the necessary information to initialize `WLClient`. | Application developer |
| Class | `WLClient` | Singleton class that exposes methods to communicate with the Worklight Server, in particular `invokeProcedure` for calling a back-end service and push notification subscription services. | IBM |
| Class | `WLProcedure InvocationData` | Class that contains all the necessary data to call a procedure. | IBM |
| Protocol | `WLDelegate` | Protocol that defines methods that a delegate 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 call. | IBM |
| Class | `WLFailResponse` | Class that is derived from `WLResponse` and contains the status in `WLResponse`, error codes, and messages. This class also contains the original response DataObject from the server. | IBM |
| Class | `WLProcedure InvocationResult` | Class that contains the result of calling a back-end service, including statuses and data items that the adapter function retrieves from the server. | IBM |
| Class | `WLCookieExtractor` | Class constructor that you use to share cookies between the web code and native code in the application. | IBM |

| Type | Name | Description | Implemented By |
|---|---|---|---|
| Class | `ChallengeHandler` | Base class that must be overridden to define custom challenge handling for user custom authentication. | Application developer |
| Enum | `WLErrorCode` | Enumeration of error codes that arrive from the Worklight Server. | IBM |
| Class | `OCLogger` | The `OCLogger` class provides some enhanced capabilities, such as capturing log calls, log level control at both the global and individual package scope, and package filtering. The `OCLogger` class also provides a method call to send captured logs to the Worklight Server. | IBM |
| TypeDef | `OCLogType` | TypeDef that shows the various levels of logging that are supported in the `OCLogger` class. | IBM |
| Class | `WLPush` | Class that exposes methods to subscribe and unsubscribe to push notifications. | IBM |
| Class | `WLPushOptions` | Class that contains the subscription parameters. | IBM |
| Interface | `OnReadyToSubscribe Listener` | Interface that defines the method that is notified when a device is ready to subscribe. | Application Developer |

*Table 1-1: IBM Worklight Objective C API for iOS classes, protocols, and files*

# 2 API reference

## 2.1 Example Code

The following code samples show how to use the IBM Worklight Objective-C client-side API. All API classes, methods, and enums are described after these examples.

### 2.1.1 Example 1: calling a back-end service that does not require authentication

MyClass.m

```
-(void) someMethod{
    …
    WLDelegate *connectDelegate = [MyConnectDelegate new];
    [[WLClient sharedInstance]
wlConnectWithDelegate:connectDelegate];
}
```

MyConnectDelegate.m <WLDelegate>

```
/**
 * called if connectDelegate succeeded
 */
-(void) onSuccess(WLResponse *)response {
  // initialize a procedureInvocationData object
  WLProcedureInvocationData *invocationData =
    [[WLProcedureInvocationData alloc]
      initWithAdapter:@"demoAdapter" procedure:@"getDemoAccount"];
    [invocationData setParameters:
      [NSArray arrayWithObjects:@"123-456-789", @"california", nil]];

  // invoke the procedure
  WLDelegate *invokeProcedureDelegate =
    [MyInvokePRocedureDelegate new];
  [[WLClient sharedInstance] invokeProcedure:invocationData
    withDelegate:invokeProcedureDelegate];
}
```

MyInvokeProcedureDelegate <WLDelegate>

```
/**
 * called if invokeProcedure succeeded
 */
-(void)onSuccess:(WLResponse *)response{
    // status
    NSLog(@"Response status is %@", [response status]);

    // print the response data
    NSLog(@"Invocation response success status: %d. Invocation result
data is %@",
        [[response invocationResult] isSuccessful],
        [[response invocationResult] getResponse]);
}

/**
 * called if invokeProcedure failed
 */
-(void)onFailure:(WLFailResponse *)failResponse{
    // status
    NSLog(@"Response status is %@". Error code %@ (%@)., [response
status],
    failResponse errorCode],
    [failResponse errorMsg]);
}
```

### 2.1.2 Example 2: calling a back-end service that requires form-based authentication

```
-(void) someMethod{
...
 [[WLClient sharedInstance]
wlConnectWithDelegate:connectDelegate];

FormChallengeHandler *challengeHandler = [[FormChallengeHandler
alloc] initWithController:self];
      challengeHandler.username = @”username”;
      challengeHandler.password = @”password”;
      [[WLClient sharedInstance]
registerChallengeHandler:[challengeHandler
initWithRealm:@”securityRealm”]];

}

/**
 * called if connectDelegate succeeded
 */
-(void) onSuccess(WLResponse *)response {
       // initialize a procedureInvocationData object
      WLProcedureInvocationData *invocationData =
             [[WLProcedureInvocationData alloc]
initWithAdapter:@"demoAdapter"
                        procedure:@"getDemoAccount"];
       [invocationData setParameters: [NSArray
arrayWithObjects:@"123-456-789",                @"california", nil]];
      // invoke the procedure
       WLDelegate *invokeProcedureDelegate =
      [MyInvokePRocedureDelegate new];
      [[WLClient sharedInstance] invokeProcedure:invocationData
            withDelegate:invokeProcedureDelegate];
}

@implementation FormChallengeHandler

/**
 * handling the challenge handler at the client side
 */
- (void)handleChallenge: (WLResponse *)response {

      NSDictionary *params = [NSDictionary
dictionaryWithObjectsAndKeys:@”username”, @"j_username", @”password”,
@"j_password", nil];
    NSDictionary *headers = [NSDictionary
dictionaryWithObjectsAndKeys:@"aaa", @"custom1", nil];



}
```

```
/**
 * called in onSuccess of challenge handler
 */
-(void)onSuccess:(WLResponse *)response {
    if([self isCustomResponse:response]){
            [self submitFailure:response];
    }
    else {
            [self submitSuccess:response];
    }
       /**
      * Successfully logged in
      */
}

/**
 * called in onFailure of challenge handler
 */
-(void)onFailure:(WLFailResponse *)response{
    [self submitFailure:response];
       /**
      * Failed to log in
      */
}
@end
```

## 2.2 Class WLClient

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

### 2.2.1 Method wlConnectWithDelegate:

#### Syntax

```
-(void)wlConnectWithDelegate:(WLDelegate *)delegate;
```

#### Description

This method uses the connection properties and the application ID from the `worklight.plist` file to initialize communication with the

Worklight Server. The server checks the validity of the application version.

Note: **This method must be called before any other `WLClient` method that calls the server, such as `logActivity` and `invokeProcedure`.**

If the server returns a successful response, the `onSuccess` method is called. If an error occurs, the `onFailure` method is called.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *delegate* | `WLDelegate` | A class that conforms to the `WLDelegate` protocol. |

*Table 2-1: Method wlConnectWithDelegate: parameters*

### 2.2.2 Method wlConnectWithDelegate:cookieExtractor:

#### Syntax

```
-(void) wlConnectWithDelegate:(id <WLDelegate>)delegate
cookieExtractor:(WLCookieExtractor *) cookieExtractor;
```

#### Description

This method uses the connection properties and the application ID from the `worklight.plist` file to initialize communication with the Worklight Server. The server checks the validity of the application version.

Note: **This method must be called before any other `WLClient` method that calls the server such as `logActivity` and `invokeProcedure`.**

If the server returns a successful response, the `onSuccess` method is called. If an error occurs, the `onFailure` method is called.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *delegate* | `WLDelegate` | A class that conforms to the `WLDelegate` protocol. |
| *cookieExtractor* | `WLcookieExtractor` | Optional. It can be `nil`. This parameter is used to share the cookies between the native code and the web code in the application. |

*Table 2-2: Method wlConnectWithDelegate:cookieExtractor: parameters*

#### Example

```
-(void) someMethod{
…
WLDelegate *connectDelegate = [MyConnectDelegate new];
[[WLClient sharedInstance] wlConnectWithDelegate:connectDelegate
cookieExtractor:[WLCookieExtractor new]];

}
```

### 2.2.3 Method invokeProcedure:withDelegate:

#### Syntax

```
-(void)invokeProcedure:(WLProcedureInvocationData
*)procedureInvocationData
withDelegate:(id <WLDelegate>)delegate;
```

#### Description

This method calls an adapter procedure. This method is asynchronous. The response is returned to the callback functions of the provided delegate.

If the call succeeds, `onSuccess` is called. If it fails, `onFailure` is called.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *procedureInvocationData* | `WLProcedureInvocationData` | The invocation data for the procedure call. |
| *delegate* | `WLDelegate` | The delegate object that is used for the `onSuccess` and `onFailure` callback methods. |

*Table 2-3: Method invokeProcedure:withDelegate: parameters*

### 2.2.4 Method invokeProcedure:withDelegate:options:

#### Syntax

```
-(void)invokeProcedure:(WLProcedureInvocationData
*)procedureInvocationData     withDelegate:(id <WLDelegate>)delegate
options:(NSDictionary *) options;
```

#### Description

This method is similar to `invokeProcedure:options`, with an additional `options` parameter to provide more data for this procedure call.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *options* | NSDictionary | A map with the following keys and values:<br>`timeout` – `NSNumber`:<br>The time, in milliseconds, for this `invokeProcedure` to wait before the request fails with `WLErrorCodeRequestTimeout.` The default timeout is 10 seconds. To disable the timeout, set this parameter to 0.<br>`invocationContext`:<br>An object that is returned with `WLResponse` to the delegate methods. You can use this object to distinguish different `invokeProcedure` calls. |

*Table 2-4: Method invokeProcedure:withDelegate:options: parameters*

#### Example

```
NSNumber *invocationContextCounter = [NSNumber numberWithInt:1];
NSNumber *timeout = [NSNumber numberWithInt:3000];
NSDictionary *options = [NSDictionary dictionaryWithObjectsAndKeys:
invocationContextCounter, @"invocationContext", timeout, @"timeout",
nil];
```

### 2.2.5 Method subscribeWithToken:adapter:eventSource: eventSourceID:notificationTypes:delegate:

#### Syntax

```
-(void) subscribeWithToken:(NSData *)deviceToken adapter:(NSString
*)adapter eventSource: (NSString *)eventSource eventSourceID: (int)id
notificationType:(UIRemoteNotificationType) types delegate:(id
<WLDelegate>)delegate
```

#### Description

Calling this method is the same as calling the method subscribeWithToken:adapter:eventSource:eventSourceID:notificationTypes:delegate:options: with the option `nil`.

### 2.2.6 Method subscribeWithToken:adapter:eventSource:eventSourceID:notificationTypes:delegate:options:

#### Syntax

```
-(void) subscribeWithToken:(NSData *)deviceToken adapter:(NSString
*)adapter eventSource: (NSString *)eventSource eventSourceID: (int)id
notificationType:(UIRemoteNotificationType) types delegate:(id
<WLDelegate>)delegate
options: (NSDictionary *)options
```

#### Description

This method subscribes the application to receive push notifications from the specified event source and adapter.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *deviceToken* | NSData | The token received from the method `application:didRegisterForRemoteNotificationsWithDeviceToken`. Save the device token in case `unsubscribedWithToken:adapter:eventSource:delegate:` is called. |
| *adapter* | NSString | The name of the adapter. |
| *eventSource* | NSString | The name of the event source. |
| *eventSourceID* | int | An ID that you assign to the event source that is returned by the Worklight Server with each notification from this event source. You can use the ID in your notification callback function to identify the notification event source.<br><br>The ID is passed on the notification payload. To save space in the notification payload, pass a short integer, otherwise it is used to pass the adapter and event source names. |

| Name | Type | Description |
|---|---|---|
| *notificationType* | `UIRemoteNotificationType` | Constants that indicate the types of notifications that the application accepts. For more information, see the Apple documentation. |
| *delegate* | `id <WLDelegate>` | A standard IBM Worklight delegate with `onSuccess` and `onFailure` methods to indicate success or failure of the subscription to the Worklight Server. |
| *options* | `NSDictionary` | Optional. This parameter contains data that is passed to the Worklight Server, which is used by the adapter. |

*Table 2-5: Method subscribeWithToken:adapter:eventSource: eventSourceID:notificationTypes:delegate:options: parameters*

### 2.2.7 Method unsubscribeAdapter:eventSource:delegate:

#### Syntax

```
-(-(void) unsubscribeAdapter:(NSString *)adapter
eventSource:(NSString *)eventSource delegate: (id
<WLDelegate>)delegate
```

#### Description

This method unsubscribes to notifications from the specified event source in the specified adapter.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *adapter* | `NSString` | The name of the adapter. |
| *eventSource* | `NSString` | The name of the event source. |
| *delegate* | `id <WLDelegate>` | A standard IBM Worklight delegate with the `onSuccess` and `onFailure` methods to indicate success or failure of the unsubscription to the Worklight Server. |

*Table 2-6: Method unsubscribeAdapter:eventSource:delegate: parameters*

### 2.2.8 Method isSubscribedToAdapter:eventSource:

#### Syntax

```
-(BOOL) isSubscribedToAdapter:(NSString *)adapter
eventSource:(NSString *)eventSource;
```

#### Description

This method returns **true** if the current logged-in user on the current device is already subscribed to the adapter and event source. The method checks the information received from the server in the success response for the login request. If the information that is sent from the server is not received, or if there is no subscription, this method returns **false**.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *adapter* | NSString | The name of the adapter. |
| *eventSource* | NSString | The name of the event source. |

*Table 2-7: Method isSubscribedToAdapter:eventSource: parameters*

### 2.2.9 Method updateDeviceToken:delegate:

#### Syntax

```
-(void) updateDeviceToken:(NSData *)deviceToken  delegate:(id
<WLDelegate>)delegate;
```

#### Description

This method compares the device token to the one registered in the Worklight Server with the current logged-in user and current device. If the device token is different, the method sends the updated token to the server.

The registered device token from the server is received in the success response for the login request. It is available without the need for an additional server call to retrieve. If a registered device token from the server is not available in the application, this method sends an update to the server with the device token.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *deviceToken* | `NSData` | The token received from the method `application:didRegisterForRemoteNotificationsWithDeviceToken`. Save the device token in case `unsubscribedWithToken:adapter:eventSource:delegate` is called. |
| *delegate* | `id <WLDelegate>` | A standard IBM Worklight delegate with `onSuccess` and `onFailure` methods to indicate successful or failure of the device token update with the Worklight Server. |

*Table 2-8: Method updateDeviceToken:delegate: parameters*

### 2.2.10 Method getEventSourceIDFromUserInfo:

#### Syntax

```
-(int)getEventSourceIDFromUserInfo: (NSDictionary *)userInfo
```

#### Description

This method returns the `eventSourceID` that the Worklight Server sends in the push notification.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *userInfo* | `NSDictionary*` | The `NSDictionary` received in the `application:didReceiveRemoteNotification` method. |

*Table 2-9: Method getEventSourceIDFromUserInfo: parameters*

### 2.2.11 Method logActivity

#### Syntax

```
-(void)logActivity:(NSString *)activityType;
```

#### Description

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

The activity is stored in the application statistics tables (the `GADGET_STAT_N` tables).

#### Parameters

| Name | Type | Description |
|---|---|---|
| *activityType* | NSString | A string that identifies the activity. |

*Table 2-10: Method logActivity parameters*

### 2.2.12 Method setHeartbeatInterval

#### Syntax

```
-(void) setHeartBeatInterval :(NSInteger)val;
```

#### Description

This method sets the interval, in seconds, at which the client (device) sends a heartbeat signal to the server. You use the heartbeat signal to prevent a session with the server from timing out because of inactivity. Typically, the heartbeat interval has a value that is less than the server session timeout. The server session timeout is defined in the `worklight.properties` file. By default, the value of the heartbeat interval is set to 420 seconds (7 minutes).

To disable the heartbeat signal, set a value that is less than, or equal to zero.

Note: **The client sends a heartbeat signal to the server only when the application is in the foreground. When the application is sent to the background, the client stops sending heartbeat signals. The client resumes sending heartbeat signals when the application is brought to the foreground again.**

#### Parameters

| Name | Type | Description |
|---|---|---|
| *NSInteger* | interval | The interval, in seconds, at which the heartbeat signal is sent to the server. |

*Table 2-11: Method setHeartbeatInterval parameters*

### 2.2.13 Method registerChallengeHandler

#### Syntax

```
-(void) registerChallengeHandler:(BaseChallengerHandler *)
challengeHandler;
```

### Description

You can use this method to register a custom Challenge Handler, which is a class that inherits from `ChallengeHandler`. See example 1: *Adding a custom Challenge Handler*.

You can also use this method to override the default Remote Disable / Notify Challenge Handler, by registering a class that inherits from `WLChallengeHandler`. See example 2: *Customizing the Remote Disable / Notify*.

### Parameters

| Name | Type | Description |
|---|---|---|
| *`challengeHandler`* | `BaseChallengeHandler` | The Challenge Handler to register. |

*Table 2-12: Method registerChallengeHandler parameters*

### Example 1: Adding a custom Challenge Handler

To add a custom Challenge Handler, you must create it, then register it on the start point of the application.

```
FormChallengeHandler *formChallengeHandler = [[FormChallengeHandler
alloc] initWithRealm:@"myCustomRealm"];
[[WLClient sharedInstance]
registerChallengeHandler:formChallengeHandler];

//
//  FormChallengeHandler.m

#import "FormChallengeHandler.h"

@implementation FormChallengeHandler



-(void) handleChallenge: (WLResponse *)response {
NSLog(@"FormChallengeHandler :: handleChallenge");
// Here you can show login form for example

// Here is code snippet to handle post submit of the login form:
NSString *username = @"username";
NSString *password = @"password";

NSDictionary *headers = [NSDictionary
dictionaryWithObjectsAndKeys:@"aaa",@"customHeader1",@"bbb",@"customHeader2",
nil];
NSDictionary *params = [NSDictionary dictionaryWithObjectsAndKeys:username,
@"j_username", password, @"j_password", nil];

// User can use the the api submitLoginForm or his custom function.
[self submitLoginForm:@"j_security_check" requestParameters:params
requestHeaders:headers requestTimeoutInMilliSeconds:30000
requestMethod:@"POST"];

}

//Failure delegate for submitLoginForm
-(void) onFailure:(WLFailResponse *)response {
[self submitFailure:response];
NSLog(@"FormChallengeHandler :: onFailureWithResponse");
}

//Success delegate for submitLoginForm
-(void)onSuccess:(WLResponse *)response{
[self submitSuccess:response];
NSLog(@"FormChallengeHandler :: onSuccessWithResponse");
}

-(BOOL) isCustomResponse:(WLResponse *) response {
NSRange authRange = [response.responseText rangeOfString:@"my unique
identifier in the response"];
if (authRange.length > 0) {
       NSLog(@"FormChallengeHandler :: isCustomResponse");
       return YES;
}
return NO;
}
@end
```

### Example 2: Customizing the Remote Disable / Notify

To customize the Remote Disable / Notify, you must create an instance of type `WLChallengeHandler`, and then register it in the start point of the application with the specific realm name `wl_remoteDisableRealm`.

```
// Register on application start point
[[WLClient sharedInstance]
registerChallengeHandler:[[CustomRemoteChallengeHandler alloc]
initWithRealm:@"wl_remoteDisableRealm"]];

//
//  CustomRemoteChallengeHandler.m

#import "CustomRemoteChallengeHandler.h"

@implementation CustomRemoteChallengeHandler
-(void) handleFailure: (NSDictionary *)failure {
// here you get the remote disable data
//message
NSString * msg = [failure valueForKey:@"message"];
//downloadLink to market
NSString * downloadLink = [failure valueForKey:@"downloadLink"];

// show your block message and exit application
...
}



//Notifying an application
-(void) handleChallenge: (NSDictionary *)challenge{
// here you get the notification data
NSString * msg = [challenge valueForKey:@"message"];
NSString * msgId = [failure valueForKey:@"messageId"];

//Needs to call setMessageId
[self setMessageId:msgId]

// show your message
...
//In the end call to submitAnswer
[self submitAnswer]
}

@end
```

### 2.2.14 Method addGlobalHeader

#### Syntax

```
-(void) addGlobalHeader: (NSString *) headerName
headerValue:(NSString *)value;
```

#### Description

You use this method to add a global header, which is sent on each request.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *headerName* | NSString | The header name/key. |
| *value* | NSString | The header value. |

*Table 2-13: Method addGlobalHeader parameters*

### 2.2.15 Method removeGlobalHeader

#### Syntax

```
-(void) removeGlobalHeader: (NSString *) headerName;
```

#### Description

You use this method to remove a global header, which is no longer sent with each request.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *headerName* | NSString | The header name to be removed. |

*Table 2-14: Method removeGlobalHeader parameters*

### 2.2.16 Method setEventTransmissionPolicy

#### Syntax

```
-(void) setEventTransmissionPolicy: (WLEventTransmissionPolicy *)
policy
```

#### Description

You use this method to configure the transmission of events from the client to the server, according to the transmission policy that is provided.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *policy* | `WLEventTransmissionPolicy` | The policy instance that will be used. |

*Table 2-15: Method setEventTransmissionPolicy parameters*

### 2.2.17 Method transmitEvent

#### Syntax

```
-(void) transmitEvent: (NSMutableDictionary *) eventJson
```

#### Description

You use this method to transmit an event to the server. This method is equivalent to calling the `transmitEvent:immediately` method, with a value of `false` for the *immediately* parameter.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *eventJson* | `WLMutableDictionary` | The event to transmit. |

*Table 2-16: Method transmitEvent parameters*

### 2.2.18 Method transmitEvent:immediately

#### Syntax

```
-(void) transmitEvent:(NSMutableDictionary *) eventJson
immediately:(BOOL)immediately
```

#### Description

You use this method to transmit a provided event object to the server.

An event object is added to the transmission buffer. The event object is either transmitted immediately, if the immediate parameter is set to `true`, otherwise it is transmitted according to the transmission policy. One of the properties for the event object might be the device context, which comprises geolocation and WiFi data. If no device context is transmitted as part of the event, the current device context, as returned by `WLDevice getContext`, is added automatically to the event during the transmission process.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *`eventJson`* | `WLMutableDictionary` | The event object that is being transmitted. The event object is either a literate object, or a reference to an object. |
| *`immediately`* | `Boolean` | A boolean flag that indicates whether the transmission should be immediate (`true`), or should be based on the transmission policy's interval (`false`). If *`immediately`* is `true`, previously buffered events are transmitted, as well as the current event. The default value is `false`. |

*Table 2-17: Method transmitEvent:immediately parameters*

### 2.2.19 Method purgeEventTransmissionBuffer

#### Syntax

```
-(void) purgeEventTransmissionBuffer
```

#### Description

You use this method to purge the internal event transmission buffer. All events that are awaiting transmission are permanently lost.

#### Parameters

None.

## 2.3 Class WLProcedureInvocationData

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

- The name of the adapter and procedure to call.
- The parameters that the procedure requires.

### 2.3.1 Method initWithAdapter:procedure:

#### Syntax

```
-(id)initWithAdapter:(NSString *)adapter procedure:(NSString
*)procedure
```

### Description

This method initializes with the adapter name and the procedure name.

### Parameters

| Name | Type | Description |
|---|---|---|
| *adapter* | NSString | The name of the adapter. |
| *procedure* | NSString | The name of the adapter procedure. |

*Table 2-18: Method initWithAdapter:procedure: parameters*

## 2.3.2 Method setParameters

### Syntax

```
-(void)setParameters:(NSArray *) parameters;
```

### Description

This method sets the procedure parameters.

### Parameters

| Name | Type | Description |
|---|---|---|
| *parameters* | NSArray | The Array contains Objects that can be parsed with JSON, such as `NSString` and `NSNumber`. For Boolean values, use `[NSNumber numberWithBool:]` |

*Table 2-19: Method setParameters parameters*

### Example

```
NSArray *params = [NSarray arrayWithObjects:@"string",
  [NSNumber numberWithInt:7],
  [NSNumber numberWithFloat:65.878],
  [NSNumber numberWithBool: YES]];
```

### 2.3.3 Method setCompressResponse:

#### Syntax

```
-(void)setCompression :(BOOL)compressResponse;
```

#### Description

This method specifies whether or not the responses from the server must be compressed. The default value is `false`.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *compressRespo nse* | BOOL | Specifies whether or not the responses from the server must be compressed. |

*Table 2-20: Method setCompressResponse: parameters*

### 2.3.4 Method initWithAdapterName

#### Syntax

```
      -(id)initWithAdapterName:(NSString *)adapter
procedureName:(NSString *)procedure
compressResponse:(BOOL)compressResponse
```

#### Description

This method initializes with the adapter name, procedure name, and `compressResponse`.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *adapter* | NSString | The name of the adapter. |
| *procedure* | NSString | The name of the adapter procedure. |
| *compressRespo nse* | BOOL | Specifies whether or not the response from the server must be compressed. |

*Table 2-21: Method initWithAdapterName parameters*

## 2.4 Protocol WLDelegate

#### Description

This protocol defines methods that a delegate for the `WLClient invokeProcedure/wlConnectWithDelegate` method implements to receive notifications about the success or failure of the method call.

### 2.4.1 Method onSuccess:

#### Syntax

```
-(void)onSuccess:(WLResponse *)response;
```

#### Description

This method is called after a successful call to the `WLCLient invokeProcedure` method. `WLResponse` contains the results from the server, along with any context object and status.

### 2.4.2 Method onFailure:

#### Syntax

```
-(void)onFailure:(WLFailResponse *)response;
```

#### Description

This method is called if any failure occurred during the execution of the `WLCLient invokeProcedure`. The `WLFailResponse` instance contains the error code and error message. Optionally, it can also contain the results from the server and any context object and status.

## 2.5 Class WLResponse

This class contains the result of a procedure call. IBM Worklight passes this class as an argument to the delegate methods of `WLClient invokeProcedure`.

### 2.5.1 Property status

#### Parameters

| Name | Type | Description |
|---|---|---|
| *`status`* | `NSString` | This property retrieves the HTTP status from the response. |

*Table 2-22: Property status parameters*

### 2.5.2 Property invocationResult

#### Parameters

| Name | Type | Description |
|---|---|---|
| *invocationResult* | `WLProcedureInvocationResult` | This property is the response data from the server. |

*Table 2-23: Property invocationResult parameters*

### 2.5.3 Property invocationContext

#### Parameters

| Name | Type | Description |
|---|---|---|
| *invocationContext* | `NSObject` | This property is the context object that is passed when the `invokeProcedure` method is called. |

*Table 2-24: Property invocationContext parameters*

### 2.5.4 Property responseText

#### Parameters

| Name | Type | Description |
|---|---|---|
| *responseText* | `NSString` | This property is the original response text from the server. |

*Table 2-25: Property responseText parameters*

### 2.5.5 Method getResponseJson

#### Syntax

```
-(NSDictionary *)getResponseJson;
```

#### Description

This method returns the value `NSDictionary` in case the response is a JSON response, otherwise it returns the value `nil`. `NSDictionary` represents the root of the JSON object.

## 2.6 Class WLFailResponse

This class is derived from `WLResponse` and contains the status in `WLResponse`, error codes, and messages. It also contains the original response DataObject from the server.

### 2.6.1 Property errorMsg

#### Parameters

| Name | Type | Description |
|---|---|---|
| *errorMsg* | `NSString` | This property is the error message for the developer, which is not necessarily suitable for the user. |

*Table 2-26: Property errorMsg parameters*

### 2.6.2 Property errorCode

#### Parameters

| Name | Type | Description |
|---|---|---|
| *errorCode* | `WLErrorCode` | This property is the error code. The `WLErrorCode` section contains a description of possible error codes. |

*Table 2-27: Property errorCode parameters*

### 2.6.3 Property invocationResult

#### Parameters

| Name | Type | Description |
|---|---|---|
| *invocationResult* | `NSObject` | This property represents the call results from the server. It can contain a different object for each callback of `WLClient`, as described in the tables of the `WLResponse` class. |

*Table 2-28: Property invocationResult parameters*

### 2.6.4 Property invocationContext

#### Parameters

| Name | Type | Description |
|---|---|---|
| *invocationContext* | `NSObject` | This property is the context object that is passed when the `invokeProcedure` method is called. |

*Table 2-29: Property invocationContext parameters*

### 2.6.5 Property responseText

#### Parameters

| Name | Type | Description |
|---|---|---|
| *responseText* | `NSString` | This property is the message, which is the original response text from the server. |

*Table 2-30: Property responseText parameters*

### 2.6.6 Method getErrorMessageFromCode

#### Syntax

```
+(NSString *) getErrorMessageFromCode: (WLErrorCode) code;
```

#### Description

This method returns a string message from a `WLErrorCode`.

### 2.6.7 Method getErrorMessageFromJSON

#### Syntax

```
+(NSString *) getErrorMessageFromJSON: (NSDictionary *) jsonResponse;
```

#### Description

This method returns an error message from the JSON response.

### 2.6.8 Method getWLErrorCodeFromJSON

#### Syntax

```
+(WLErrorCode) getWLErrorCodeFromJSON: (NSDictionary *) jsonResponse;
```

#### Description

This method returns the `WLErrorCode` from the JSON response.

## 2.7 Class WLProcedureInvocationResult

This class contains the result of calling a back-end service, including statuses and data items that the adapter function retrieves from the server.

### 2.7.1 Method isSuccessful

#### Syntax

```
-(BOOL)isSuccessful;
```

#### Description

This method returns `YES` if the call was successful, and `NO` otherwise.

### 2.7.2 Method procedureInvocationErrors

#### Syntax

```
-(NSArray *) procedureInvocationErrors;
```

#### Description

This method returns an `NSArray` of applicative error messages that the server collects during the procedure call.

### 2.7.3 Property response

#### Parameters

| Name | Type | Description |
|---|---|---|
| *response* | `NSDictionary` | This property is an `NSDictionary`, which represents the JSON response that the Worklight Server returns. |

*Table 2-31: Property response parameters*

## 2.8 Class WLCookieExtractor

This class provides access to external cookies that `WLClient` can use when it issues requests to the Worklight Server. You use this class to share session cookies between a web view and a natively implemented page.

This class has no API methods. An object of this class must be passed as a parameter to `wlConnectWithDelegate:cookieExtractor`.

### 2.8.1 Class constructor WLCookieExtractor

#### Syntax

```
public WLCookieExtractor(Application app)
```

#### Parameters

| Name | Type | Description |
|---|---|---|
| *app* | `Application` | An Android application instance. |

*Table 2-32: Class constructor WLCookieExtractor parameters*

## 2.9 Class ChallengeHandler

You use this base class to create custom Challenge Handlers. You must extend this class to implement your own Challenge Handler logics. You use this class to create custom user authentication.

### 2.9.1 Method isCustomResponse

#### Syntax

```
-(BOOL)_isCustomResponse:(WLResponse *)response;
```

#### Description

This method must be overridden by extending the `ChallengeHandler` class. In most cases, you call this method to test whether there is a custom challenge to be handled in the response. Default Challenge Handlers might handle some responses. If this method returns `YES`, Worklight calls the `handleChallenge` method.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *response* | `WLResponse` | The `WLResponse` to be tested |

*Table 2-33: Method isCustomResponse parameters*

### 2.9.2 Method handleChallenge

#### Syntax

```
-(void) handleChallenge: (WLResponse *)response;
```

#### Description

Worklight must call this method whenever `isCustomResponse` returns a `YES` value. You must implement this method. This method must handle the challenge, for example to show the login screen.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *response* | `WLResponse` | The `WLResponse` to be handled. |

Table 2-34: Method handleChallenge parameters

### 2.9.3 Method submitSuccess

#### Syntax

```
-(void) submitSuccess:(WLResponse *) response;
```

#### Description

You must call this method when the challenge is answered successfully, for example after the user successfully submits the login form. Then, this method sends the original request.

Calling this method informs Worklight that the challenge was successfully handled. This method can also be used to inform Worklight that the response that was received is not a custom response that your challenge handler can handle. In this case, control is returned to Worklight to handle the response.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *response* | `WLResponse` | The received `WLResponse`. |

Table 2-35: Method submitSuccess parameters

### 2.9.4 Method submitFailure

#### Syntax

```
-(void) submitFailure:(WLResponse *) response;
```

#### Description

You must call this method whenever the challenge is answered with an error. This method is inherited from `BaseChallengeHandler`.

Calling this method tells Worklight 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 Worklight 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.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *response* | `WLResponse` | The received `WLResponse`. |

*Table 2-36: Method submitFailure parameters*

### 2.9.5 Method submitLoginForm

#### Syntax

```
-(void) submitLoginForm:
      (NSString *)requestUrl
      requestParameters:(NSDictionary *) parameters
      requestHeaders:(NSDictionary *) headers
      requestTimeoutInMilliSeconds:(int) timeout
      requestMethod:(NSString *) method;
```

#### Description

You use this method to send collected credentials to a specific URL. You can also specify request parameters, headers, and timeout.

The success/failure delegate for this method is the instance itself, which is why you must override the `onSuccess` / `onFailure` methods.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *requestUrl* | `NSString` | Absolute URL if the user sends an absolute url that starts with `http://` or `https://`<br><br>Otherwise, URL relative to the Worklight Server |
| *parameters* | NSDictionary | The request parameters |
| *headers* | NSDictionary | The request headers |
| *timeout* | `int` | To supply custom timeout, use a number over 0.<br><br>If the number is under 0, Worklight uses the default timeout, which is 10,000 milliseconds. |
| *method* | `NSString` | The HTTP method that you must use.<br><br>Acceptable values are `GET, POST.`<br><br>The default value is `POST`. |

*Table 2-37: Method submitLoginForm parameters*

### 2.9.6 Method submitAdapterAuthentication

#### Syntax

```
-(void) submitAdapterAuthentication: (WLProcedureInvocationData *)
wlInvocationData: options:(NSDictionary *)requestOptions;
```

#### Description

You use this method to invoke a procedure from the Challenge Handler.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *wlInvocationData* | `WLProcedureInvocationData` | The invocation data, for example the name of the procedure, or the name of the method. |
| *requestOptions* | `NSDictionary` | A map with the following keys and values:<br>`timeout` – `NSNumber`:<br>The time, in milliseconds, for this `invokeProcedure` to wait before the request fails with `WLErrorCodeRequestTimeout`. The default timeout is 10,000 milliseconds. To disable the timeout, set this parameter to 0.<br>`invocationContext:`<br>An object that is returned with `WLResponse` to the delegate methods. You can use this object to distinguish different `invokeProcedure` calls. |

*Table 2-38: Method submitAdapterAuthentication parameters*

### 2.9.7 Method onSuccess

#### Syntax

```
-(void) onSuccess:(WLResponse *)response;
```

#### Description

This method is the success delegate for `submitLoginForm` or `submitAdapterAuthentication`.

This method is called when a successful HTTP response is received (200 OK). This method does not indicate whether the challenge was successful or not. A 200 HTTP response can flow back indicating problems with authentication on the server or requesting additional information.

Some examples of a 200 HTTP response are as follows:

- First init request returns a normal 200 HTTP response that requests a pkms login form
- Authentication failed on the server.
- A 200 HTTP response, indicating that the account is locked on the server due to too many failed login attempts.

Note: Worklight does not attempt to determine what the 200 response means.

This is a good place to check whether the response is a custom response and handle it accordingly. If the response is not a custom response, you can call `submitSuccess` to indicate that all is good from your challenge handler's perspective, and that Worklight can handle the response instead.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *response* | `WLResponse` | The received response. |

*Table 2-39: Method onSuccess parameters*

### 2.9.8 Method onFailure

#### Syntax

```
-(BOOL) isCustomResponse:( (WLFailResponse *)response;
```

#### Description

This method is the failure delegate for `submitLoginForm` or `submitAdapterAuthentication`.

This method is called when a response does not have a 200 HTTP status code. This method does not indicate whether the challenge was successful or not. In some cases `onFailure` is an indication of a normal challenge handling sequence.

An example of when the `onFailure` method is called is when a 401 Unauthorized response is received.

A successful handshake can entail several 401 response iterations and therefore several `onFailure` calls. This behavior is all part of the normal handshake between two parties that are trying to establish identity. Worklight handles the handshakes for core challenges iteratively until all of the credentials are established and the necessary challenges are processed.

This is a good place to check whether the response is a custom response and handle it accordingly. If the response is not a custom response, you can call `submitSuccess` to indicate that all is good from your challenge handler's perspective, and that Worklight can handle the response instead.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *response* | `WLResponse` | The received fail response. |

*Table 2-40: Method onFailure parameters*

### Examples of ChallengeHandler class methods

```
-(BOOL) isCustomResponse:(WLResponse *) response {
  // Return true if you want your challenge handlers to handle the
response via handleChallenge call.
}

-(void) handleChallenge: (WLResponse *)response {
  // call submitFailure to relinquish control back to Worklight for
handling and indicate challenge was unsuccessful.
  // call submitLoginForm to request user login or handle challenge
accordingly.
}

-(void)handleLoginResponse:(WLResponse *) response {
  if (![self isCustomResponse:response]) {
    //Call submitSuccess to relinquish control back to Worklight for
normal handling when the response is not a custom response.
    [self submitSuccess:response];
  }
}

//Failure delegate for submitLoginForm
-(void)onFailure:(WLFailResponse *)response {
  NSLog(@"Challenge handler onFailure:");
  [self handleLoginResponse:response];
}

//Success delegate for submitLoginForm
-(void)onSuccess:(WLResponse *)response {
  NSLog(@"Challenge handler onSuccess:");
  [self handleLoginResponse:response];
}
```

## 2.10 Enum WLErrorCode

#### Definition

```
typedef NSUInteger WLErrorCode;
enum {
  WLErrorCodeUnexpectedError,
  WLErrorCodeUnresponsiveHost,
  WLErrorCodeRequestTimeout,
  WLErrorCodeProcedureError,
  WLErrorCodeApplicationVersionDenied};
```

#### Description

The following list shows the various error codes that you can find and their description:

- UNEXPECTED_ERROR: unexpected error
- REQUEST_TIMEOUT: request timed out
- UNRESPONSIVE_HOST: service currently unavailable
- PROCEDURE_ERROR: procedure invocation error
- APP_VERSION_ACCESS_DENIAL: application version denied

## 2.11 Class OCLogger

### 2.11.1 Method getInstanceWithPackage

#### Syntax

```
+(OCLogger*) getInstanceWithPackage: (NSString*)
package;
```

#### Description

This method gets or creates an instance of this logger. If an instance exists for the `package` parameter, that instance is returned.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *package* | `NSString` | The package or tag that must be printed with log messages. The value is passed through to `NSLog` and recorded when log capture is enabled. |

*Table 2-41: Method getInstanceWithPackage parameters*

### 2.11.2 Method setLevel

#### Syntax

```
+(void) setLevel: (OCLogType) level;
```

#### Description

This method sets the level from which log messages must be saved and printed. For example, passing `OCLogger_INFO` logs `INFO`, `WARN`, and `ERROR`.

#### Parameters

| Name | Type | Description |
|---|---|---|
| **level** | `OCLogType` | The valid values of this input parameter are `OCLogger_DEBUG`, `OCLogger_ERROR`, `OCLogger_INFO`, `OCLogger_LOG`, and `OCLogger_WARN`. |

*Table 2-42: Method setLevel parameters*

### 2.11.3 Method getLevel

#### Syntax

```
+(OCLogType) getLevel;
```

#### Description

This method gets the current `OCLogger_LEVEL` and returns `OCLogger_LEVEL`.

### 2.11.4 Method send

#### Syntax

```
+(void) send;
```

#### Description

This method sends the log file when the log buffer exists and is not empty.

### 2.11.5 Method setCapture

#### Syntax

```
+(void) setCapture: (BOOL) flag;
```

#### Description

Global setting: turn persisting of the log data passed to the log methods of this class, on or off.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *flag* | BOOL | Flag to indicate whether the log data must be saved persistently. |

*Table 2-43: Method setCapture parameters*

### 2.11.6 Method getCapture

#### Syntax

```
+(BOOL) getCapture;
```

#### Description

This method gets the current value of the capture flag, indicating that the `OCLogger` is recording log calls persistently. This method returns the current value of the capture flag.

### 2.11.7 Method setMaxFileSize

#### Syntax

```
+(void) setMaxFileSize: (int) bytes;
```

#### Description

This method sets the maximum size of the local log file. When the maximum file size is reached, no more data is appended. This file is sent to a server.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *bytes* | int | The maximum size of the file in bytes, the minimum is 10,000 bytes. |

*Table 2-44: Method setMaxFileSize parameters*

### 2.11.8 Method getMaxFileSize

#### Syntax

```
+(int) getMaxFileSize;
```

#### Description

This method gets the current setting for the maximum file size threshold.

### 2.11.9 Method info

#### Syntax

```
-(void) info: (NSString*) message;
-(void) metadata:(NSDictionary*) metadata info:
(NSString*) text, ...;
```

#### Description

This method logs at `INFO` level.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *message* | NSString | The message to log. |
| *metadata* | NSDictionary | The metadata to append to the log output. |

*Table 2-45: Method info parameters*

### 2.11.10 Method debug

#### Syntax

```
-(void) debug: (NSString*) message;
-(void) metadata:(NSDictionary*) metadata debug:
(NSString*) text, ...;
```

#### Description

This method logs at `DEBUG` level.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *message* | NSString | The message to log. |
| *metadata* | NSDictionary | The metadata to append to the log output. |

*Table 2-46: Method debug parameters*

### 2.11.11 Method error

#### Syntax

```
-(void) error: (NSString*) message;
-(void) metadata:(NSDictionary*) metadata error:
(NSString*) text, ...;
```

#### Description

This method logs at `ERROR` level.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *message* | String | The message to log. |
| *metadata* | NSDictionary | The metadata to append to the log output. |

*Table 2-47: Method error parameters*

### 2.11.12 Method log

#### Syntax

```
-(void) log: (NSString*) message;
-(void) metadata:(NSDictionary*) metadata log:
(NSString*) text, ...;
```

#### Description

This method logs at `LOG` level.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *message* | String | The message to log. |
| *metadata* | NSDictionary | The metadata to append to the log output. |

*Table 2-48: Method log parameters*

### 2.11.13 Method warn

#### Syntax

```
-(void) warn: (NSString*) message;
-(void) metadata:(NSDictionary*) metadata warn:
(NSString*) text, ...;
```

#### Description

This method logs at `WARN` level.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *message* | String | The message to log. |
| *metadata* | NSDictionary | The metadata to append to the log output. |

*Table 2-49: Method warn  parameters*

## 2.12 TypeDef OCLogType

#### Definition

```
public static final OCLogger_DEBUG
public static final OCLogger_ERROR
public static final OCLogger_INFO
public static final OCLogger_LOG
public static final OCLogger_WARN
```

#### Description

The following list shows the various levels of logging that are supported in the `OCLogger` class:

- DEBUG
- ERROR
- INFO
- LOG
- WARN

# 3 Push Notifications

The Worklight Server sends notifications to the Apple Push Notification service (APNs). The APNs then sends the notifications to the relevant phones.

To enable push notifications in your application, follow these steps.

- Add the `<pushSender>` element to the application descriptor of the Native API application.

```
<nativeIOSApp>
            ..
        <pushSender password=""/>
         ..
</nativeIOSApp>
```

- Copy the `WorklightAPI` folder from the Native API application, and paste it into your native iOS application in Xcode.
- Copy the `worklight.plist` file into the Xcode project.
- Copy the `apns-sandbox-certificate.p12` or `apns-production.p12` keys into the application root folder of your IBM Worklight application.
- Deploy your IBM Worklight application.

## 3.1 Class WLPush

This class exposes all the methods that are required for push notifications.

### 3.1.1 Method setOnReadyToSubscribeListener

#### Syntax

```
-(void) setOnReadyToSubscribeListener(OnReadyToSubscribeListener
listener)
```

#### Description

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

#### Parameters

| Name | Type | Description |
|---|---|---|
| *OnReadyToSubscribeListener* | WLOnReadyToSubscribeListener | Mandatory. When the device is ready to subscribe to push notifications, the `onReadyToSubscribe` method is called. |

Table 3-1: Method setOnReadyToSubscribeListener parameters

### 3.1.2 Method registerEventSourceCallback

#### Syntax

```
-(void) registerEventSourceCallback:(NSString *)alias :(NSString
*)adapter :(NSString *)eventsource :(id
<EventSourceListener>)eventSourceListener
```

#### Description

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

#### Parameters

| Name | Type | Description |
|---|---|---|
| *alias* | NSString | Mandatory string. A short ID that you use to identify the event source when the push notification arrives.<br><br>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 usually limited in length. |
| *adapter* | NSString | Mandatory string. The name of the adapter that contains the event source. |
| *eventSource* | NSString | Mandatory string. The name of the event source. |
| *eventSourceListener* | EventSourceListener | Mandatory listener class. When a notification arrives, the `EventSourceListener.onReceive` method is called. |

Table 3-2: Method registerEventSourceCallback parameters

### 3.1.3 Method tokenFromClient

#### Syntax

```
-(void)  setTokenFromClient : (NSString *) token
```

#### Description

This method sends the token from the client application to the Worklight Server.

### 3.1.4 Method subscribe

#### Syntax

```
-(void) subscribe :(NSString *)alias :(WLPushOptions *)options : (id
<WLDelegate>)responseListener
```

#### Description

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

#### Parameters

| Name | Type | Description |
|---|---|---|
| *`alias`* | `NSString` | Mandatory string. The event source alias, as defined in `registerEventSourceCallback`. |
| *`options`* | `WLPushOptions` | Optional. This instance contains the custom subscription parameters that the event source in the adapter supports. |
| *`responseListener`* | `WLDelegate` | Optional. The listener object, whose callback methods, `onSuccess` and `onFailure`, are called. |

*Table 3-3: Method subscribe parameters*

### 3.1.5 Method isSubscribed

#### Syntax

```
-(BOOL) isSubscribed :(NSString *)alias
```

#### Description

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

#### Parameters

| Name | Type | Description |
|---|---|---|
| *alias* | NSString | Mandatory string. The event source alias. |

*Table 3-4: Method isSubscribed parameters*

### 3.1.6 Method isPushSupported

#### Syntax

```
-(BOOL)isPushSupported
```

#### Description

This method checks whether push notification is supported.

### 3.1.7 Method unsubscribe

#### Syntax

```
-(void) unsubscribe :(NSString *)alias :(id
<WLDelegate>)responseListener
```

#### Description

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

#### Parameters

| Name | Type | Description |
|---|---|---|
| *alias* | NSString | Mandatory string. The event source alias, as defined in `registerEventSourceCallback`. |
| *responseListener* | WLDelegate | Optional. The listener object whose callback methods, `onSuccess` and `onFailure`, are called. |

*Table 3-5: Method unsubscribe parameters*

## 3.2 Class WLPushOptions

This class contains the subscription parameters.

### 3.2.1 Method addSubscriptionParameter

#### Syntax

```
-(void) addSubscriptionParameter :(NSString *)name :(NSString *)value
```

#### Description

You use this method to add a subscription parameter.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *name* | NSString | Mandatory. The name of the subscription parameter. |
| *value* | NSString | Mandatory. The value of the subscription parameter. |

*Table 3-6: Method addSubscriptionParameter parameters*

### 3.2.2 Method addSubscriptionParameters

#### Syntax

```
-(void) addSubscriptionParameters :(NSDictionary *)parameters
```

#### Description

You use this method to add subscription parameters.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *parameters* | NSDictionary | Mandatory. The `NSDictionary` that contains the list of subscription parameters. |

*Table 3-7: Method addSubscriptionParameters parameters*

### 3.2.3 Method getSubscriptionParameter

#### Syntax

```
-(NSString *) getSubscriptionParameter :(NSString *)name
```

#### Description

This method returns the value of the given subscription parameter.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *name* | `NSString` | Mandatory. The name of the subscription parameter. |

*Table 3-8: Method getSubscriptionParameter parameters*

### 3.2.4 Method getSubscriptionParameters

#### Syntax

```
-(NSDictionary *) getSubscriptionParameters
```

#### Description

This method returns the map that contains the subscription parameters.

## 3.3 Interface OnReadyToSubscribeListener

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

### 3.3.1 Method onReadyToSubscribe

#### Syntax

```
-(void) setOnReadyToSubscribeListener:(id
<WLOnReadyToSubscribeListener>)listener
```

#### Description

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

## 3.4 Receiving notifications

### 3.4.1 Method didReceiveRemoteNotification

#### Syntax

```
- (void)application:(UIApplication*)application
didReceiveRemoteNotification:(NSDictionary*)userInfo
```

#### Description

To receive notifications, you must implement the `didReceiveRemoteNotification` method in the `AppDelegate` of the client application. `userInfo` contains the notification message.

## 3.5 Getting the token from APNs

### 3.5.1 Method didRegisterForRemoteNotificationsWithDeviceToken

#### Syntax

```
- (void)application:(UIApplication*)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData*)deviceToken
```

#### Description

The client application must get the token from APNs and pass it to the Worklight Server. To get the token, you must implement the `didRegisterForRemoteNotificationsWithDeviceToken` method in the `AppDelegate` of the client application. The client application must then pass the device token to the Worklight Server.

# 4 Location services

## 4.1 Class WLAcquisitionFailureCallbacksConfiguration

This class defines the configuration of the callbacks that are called when there is an acquisition failure.

The setters of this class return a reference to this object so as to enable chaining calls.

### 4.1.1 Method getGeoFailureCallback

#### Syntax

```
- (id<WLGeoFailureCallback>)getGeoFailureCallback
```

#### Description

This method returns the geolocation failure callback. The default value is `null`.

#### Parameters

None.

### 4.1.2 Method getWifiFailureCallback

#### Syntax

```
- (id<WLWifiFailureCallback>)getWifiFailureCallback
```

#### Description

This method returns the WiFi failure callback. The default value is `null`.

#### Parameters

None.

### 4.1.3 Method setGeoFailureCallback

#### Syntax

```
- (WLAcquisitionFailureCallbacksConfiguration
*)setGeoFailureCallback:(id<WLGeoFailureCallback>)geoFailureCallbacks
```

#### Description

Sets the geolocation failure callback.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *geoFailureCallbacks* | `id<WLGeoFailureCallback>` | A reference to an object that conforms to the `WLGeoFailureCallback` protocol. |

*Table 4-1: Method setGeoFailureCallback: parameters*

### 4.1.4 Method setWifiFailureCallback

#### Syntax

```
- (WLAcquisitionFailureCallbacksConfiguration
*)setWifiFailureCallback:(id<WLWifiFailureCallback>)wifiFailureCallbacks
```

#### Description

Sets the WiFi failure callback.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *wifiFailureCallbacks* | `id<WLWifiFailureCallback>` | A reference to an object that conforms to the `WLWifiFailureCallback` protocol. |

*Table 4-2: Method setWifiFailureCallback: parameters*

## 4.2 Class WLAcquisitionPolicy

This class controls how geo and WiFi locations are acquired.

The setters of this class return a reference to this object so as to enable chaining calls.

### 4.2.1 Method getGeoPolicy

#### Syntax

```
- (WLGeoAcquisitionPolicy *)getGeoPolicy
```

#### Description

This method returns the geolocation acquisition policy.

#### Parameters

None.

### 4.2.2 Method getWifiPolicy

#### Syntax

```
- (WLWifiAcquisitionPolicy *)getWifiPolicy
```

#### Description

This method returns the WiFi acquisition policy.

#### Parameters

None.

### 4.2.3 Method setGeoPolicy

#### Syntax

```
- (WLAcquisitionPolicy *)setGeoPolicy:(WLGeoAcquisitionPolicy
*)geoPolicy
```

#### Description

This method sets the geolocation acquisition policy. When `null`, it can be used to stop geolocation acquisition.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *geoPolicy* | `WLGeoAcquisitionPolicy` | The geolocation acquisition policy to set. |

*Table 4-3: Method setGeoPolicy: parameters*

### 4.2.4 Method setWifiPolicy

#### Syntax

```
- (WLAcquisitionPolicy *)setWifiPolicy:(WLWifiAcquisitionPolicy
*)wifiPolicy
```

#### Description

This method sets the WiFi acquisition policy. When `null`, it can be used to stop WiFi acquisition.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *wifiPolicy* | WLWifiAcquisitionPolicy | The WiFi acquisition policy to set. |

*Table 4-4: Method setGeoPolicy: parameters*

## 4.3 Protocol WLArea

This protocol provides the parent interface for geometric shapes.

## 4.4 Class WLCallbackFactory

This class enables you to use blocks whenever a callback object is needed in the Worklight location services API.

### 4.4.1 Method createGeoCallback

#### Syntax

```
+ (id<WLGeoCallback>)createGeoCallback:(void (^)(WLGeoPosition
*))callbackBlock
```

#### Description

This method creates a geo callback that wraps the given block.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *callbackBlock* | void(^)(WLGeoPosition *) | The block that will be delegated to by the returned WLGeoCallback instance. |

*Table 4-5: Method createGeoCallback: parameters*

### 4.4.2 Method createGeoFailureCallback

#### Syntax

```
+ (id<WLGeoFailureCallback>)createGeoFailureCallback:(void
(^)(WLGeoError *))callbackBlock
```

#### Description

This method creates a geo failure callback that wraps the given block.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *callbackBlock* | `void(^)(WLGeoError *)` | The block that will be delegated to by the returned WLGeoFailureCallback instance. |

*Table 4-6: Method createGeoFailureCallback: parameters*

### 4.4.3 Method createTriggerCallback

#### Syntax

```
+ (id<WLTriggerCallback>)createTriggerCallback:(void
(^)(id<WLDeviceContext> deviceContext))callbackBlock
```

#### Description

This method creates a trigger callback that wraps the given block.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *callbackBlock* | `void(^)(id<WLDevice Context>)` | The block that will be delegated to by the returned WLTriggerCallback instance. |

*Table 4-7: Method createTriggerCallback: parameters*

### 4.4.4 Method createWifiConnectedCallback

#### Syntax

```
+ (id<WLWifiConnectedCallback>)createWifiConnectedCallback:(void
(^)(WLWifiAccessPoint * connectedAccessPoint, NSNumber *
signalStrength))callbackBlock
```

#### Description

This method creates a WiFi connected callback that wraps the given block.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *callbackBlock* | `void(^)(WLWifiAccessPoint *,NSNumber *)` | The block that will be delegated to by the returned WLWifiConnectedCallback instance. |

*Table 4-8: Method createWifiConnectedCallback: parameters*

### 4.4.5 Method createWifiFailureCallback

#### Syntax

```
+ (id<WLWifiFailureCallback>)createWifiFailureCallback:(void
(^)(WLWifiError * error))callbackBlock
```

#### Description

This method creates a WiFi failure callback that wraps the given block.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *callbackBlock* | `void(^)(WLWifiError *)` | The block that will be delegated to by the returned WLGeoCallback instance. |

*Table 4-9: Method createWifiFailureCallback: parameters*

## 4.5 Class WLCircle

This class identifies a circle, defined by its center point and a radius. This class is immutable.

### 4.5.1 Method getCenter

#### Syntax

```
- (WLCoordinate *)getCenter
```

#### Description

This method returns the center of the circle.

#### Parameters

None.

### 4.5.2 Method getRadius

#### Syntax

```
- (double)getRadius
```

#### Description

This method returns the radius of the circle, in meters.

#### Parameters

None.

### 4.5.3 Method initWithCenter

#### Syntax

```
- (id)initWithCenter:(WLCoordinate *)center radius:(double)radius
```

#### Description

This method creates a new circle.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *center* | WLCoordinate | The center of the circle. |
| *radius* | double | The radius of the circle, in meters. |

*Table 4-10: Method initWithCenter: parameters*

## 4.6 Class WLCoordinate

This class defines a coordinate on the globe.

### 4.6.1 Method getAccuracy

#### Syntax

```
- (double)getAccuracy
```

#### Description

This method returns the accuracy of the coordinate, in meters.

#### Parameters

None.

### 4.6.2 Method getAltitude

#### Syntax

```
- (NSNumber *)getAltitude
```

#### Description

This method returns the altitude of the coordinate, in meters, if available. If unavailable, `null` is returned.

#### Parameters

None.

### 4.6.3 Method getAltitudeAccuracy

#### Syntax

```
- (NSNumber *)getAltitudeAccuracy
```

#### Description

This method returns the altitude accuracy of the coordinate, in meters, if available. If unavailable, `null` is returned.

#### Parameters

None.

### 4.6.4 Method getHeading

#### Syntax

```
- (NSNumber *)getHeading
```

#### Description

This method returns the heading of the coordinate, in degrees (0–360), if available. If unavailable, `null` is returned.

.

#### Parameters

None.

### 4.6.5 Method getLatitude

#### Syntax

```
- (double)getLatitude
```

#### Description

This method returns the latitude of the coordinate.

#### Parameters

None.

### 4.6.6 Method getLongitude

#### Syntax

```
- (double)getLongitude
```

#### Description

This method returns the longitude of the coordinate.

#### Parameters

None.

### 4.6.7 Method getSpeed

#### Syntax

```
- (NSNumber *)getSpeed
```

#### Description

This method returns the speed of the coordinate, in meters per second, if available. If unavailable, `null` is returned..

#### Parameters

None.

### 4.6.8 Method initWithLatitude:longitude

#### Syntax

```
- (id)initWithLatitude:(double)latitude longitude:(double)longitude
```

#### Description

This method initializes the coordinate with the given values.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *latitude* | double | The coordinate's latitude value. |
| *longitude* | double | The coordinate's longitude value. |

*Table 4-11: Method initWithLatitude:longitude: parameters*

### 4.6.9 Method initWithLatitude:longitude:accuracy

#### Syntax

```
- (id)initWithLatitude:(double)latitude longitude:(double)longitude
accuracy:(double)accuracy
```

#### Description

This method initializes the coordinate with the given values.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *latitude* | double | The coordinate's latitude value. |
| *longitude* | double | The coordinate's longitude value. |
| *accuracy* | double | The coordinate's accuracy. |

*Table 4-12: Method initWithLatitude:longitude:accuracy parameters*

## 4.7 Protocol WLDevice

This protocol provides access to the device's context, which provides access to the acquired location information. This protocol also can be used to acquire location information.

### 4.7.1 Method acquireGeoPositionWithDelegate:failureDelegate:policy

#### Syntax

```
- (void)acquireGeoPositionWithDelegate:(id<WLGeoCallback>)onSuccess
failureDelegate:(id<WLGeoFailureCallback>)onFailure
policy:(WLGeoAcquisitionPolicy *)geoPolicy
```

#### Description

This method acquires a geographical position.

The device attempts to acquire a geographical position. This attempt could be based on geolocation data acquired by the device, or it could involve the use of WiFi. If the attempt is successful, the following actions take place:

- The device context might be updated. This action is dependent on the freshness of the data in the context, and the new position data being at least as accurate as the existing position data.
- The **onSuccess** function is invoked.
- If the device context was updated, triggers might be activated.

**Note:** Because `acquireGeoPosition` might activate triggers, you should not call `acquireGeoPosition` from a trigger callback. Potentially, this call could cause an endless loop of trigger evaluations leading to callbacks leading to `acquireGeoPosition` calls.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *onSuccess* | `id<WLGeoCallback>` | A reference to an object that conforms to the `WLGeoCallback` protocol. Its execute method is invoked when a position is acquired successfully. The position is passed as a parameter to this method. |
| *onFailure* | `id<WLGeoFailureCallback>` | A reference to an object that conforms to the `WLGeoFailureCallback` protocol. Its execute method is invoked when the position is not acquired successfully. The error is passed as a parameter to this method. |
| *geoPolicy* | `WLGeoAcquisitionPolicy` | The policy that is used to configure the acquisition. |

*Table 4-13: Method acquireGeoPositionWithDelegate:failureDelegate:policy: parameters*

### 4.7.2 Method getConnectedAccessPointFilteredByPolicy:withDelegate:failureDelegate

#### Syntax

```
- (void)getConnectedAccessPointFilteredByPolicy:(WLWifiAcquisitionPolicy
*)wifiPolicy withDelegate:(id<WLWifiAcquisitionCallback>)onSuccess
failureDelegate:(id<WLWifiFailureCallback>)onFailure
```

#### Description

This method acquires the currently connected WiFi access point. The connected access point is returned to the device, as specified by the `accessPointFilters` setting in the provided `wifiPolicy` parameter.

If the access attempt is successful, and ongoing WiFi acquisition is enabled (using `WLDevice startAcquisition`), the following actions take place:

- The device context is updated.
- The **onSuccess** function is invoked.
- Triggers are activated.

**Note:** Because this method might activate triggers, you must be careful when calling it from a trigger callback. Potentially, this could cause an endless loop of trigger evaluations leading to callbacks, in turn leading to `getConnectedAccessPointFilteredByPolicy:withDelegate` calls.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *wifiPolicy* | `WLWifiAcquisitionPolicy` | The policy that is used to configure the acquisition. |
| *onSuccess* | `id<WLWifiAcquisitionCallback>` | A reference to an object that conforms to the `WLWifiAcquisitionCallback` protocol. Its execute method is invoked when the visible access points are acquired successfully. The access points are passed as a parameter to this method. |
| *onFailure* | `id<WLWifiFailureCallback>` | A reference to an object that conforms to the `WLWifiFailureCallback` protocol. Its execute method is invoked when the acquisition is unsuccessful. The error is passed as a parameter to this method. |

*Table 4-14: Method acquireGetConnectedAccessPointFilteredByPolicy:withDelegate:failureDelegate:parameters*

### 4.7.3 Method getConnectedAccessPointWithDelegate:failureDelegate:

#### Syntax

```
-
(void)getConnectedAccessPointWithDelegate:(id<WLWifiConnectedCallback>)onSuccess
failureDelegate:(id<WLWifiFailureCallback>)onFailure
```

#### Description

This method acquires the currently connected WiFi access point information.

The device attempts to acquire the currently connected WiFi access point information. If the attempt is successful, the access point information is passed to the `onSuccess` callback function.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *onSuccess* | `id<WLWifiConnectedCallback>` | A reference to an object that conforms to the `WLWifiConnectedCallback` protocol. Its execute method is invoked when the access point connection information is acquired successfully. The access point connection information is passed as a parameter to this method. |
| *onFailure* | `id<WLWifiFailureCallback>` | A reference to an object that conforms to the `WLWifiFailureCallback` protocol. Its execute method is invoked when the acquisition is unsuccessful. The error is passed as a parameter to this method. |

*Table 4-15: Method getConnectedAccessPointWithDelegate:failureDelegate:parameters*

### 4.7.4 Method getDeviceContext

#### Syntax

```
- (id<WLDeviceContext>)getDeviceContext
```

#### Description

This method returns the current device context, which contains information about the acquired locations.

#### Parameters

None.

### 4.7.5 Method getLocationServicesConfig

#### Syntax

```
- (WLLocationServicesConfiguration *)getLocationServicesConfig
```

#### Description

This method returns the current location services configuration.

#### Parameters

None.

### 4.7.6 Method startAcquisition

#### Syntax

```
- (void)startAcquisition:(WLLocationServicesConfiguration
*)newConfiguration
```

#### Description

This method starts ongoing acquisition for sensors that are provided in the newConfiguration policy.

Ongoing acquisition is started for the Geo and WiFi sensors that are provided in the policy. When new sensor information is acquired, the device context is updated, and the specified triggers are evaluated for activation..

#### Parameters

| Name | Type | Description |
|---|---|---|
| *newConfiguration* | `WLLocationServicesConfiguration` | The configuration to use to specify acquisition policy, trigger configuration, and failure callbacks. Changes made to the configuration after this method has been called do not modify the runtime behavior unless the object is passed again in a new call to this method. |

*Table 4-16: Method startAcquisition: parameters*

### 4.7.7 Method stopAcquisition

#### Syntax

```
- (void)stopAcquisition
```

#### Description

This method stops the ongoing acquisition. The stop action is delegated to all relevant sensors, and all trigger states are cleared.

#### Parameters

None.

## 4.8 Protocol WLDeviceContext

This protocol provides information on position acquisitions.

### 4.8.1 Method addToEvent

#### Syntax

```
- (void)addToEvent:(NSMutableDictionary *)event
```

#### Description

This method adds the JSON representation, as returned by the `getJSON` method, to an event.

A typical use of this method would be to send a dynamic event as a result of a trigger being activated; that is, the event is constructed when the trigger is activated, and not when the trigger was created.

The `WLDeviceContext` instance received by the `execute` method of `WLTriggerCallback` adds its data to a dynamic event object, which is then passed to `WLClient transmitEvent:immediately`.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *event* | `NSMutableDictionary` | The JSON object to update. |

*Table 4-17: Method addToEvent: parameters*

### 4.8.2 Method getGeoPosition

#### Syntax

```
- (WLGeoPosition *)getGeoPosition
```

#### Description

This method returns the last geographical position acquired. If no position has been acquired, `null` is returned.

#### Parameters

None.

### 4.8.3 Method getJSON

#### Syntax

```
- (NSMutableDictionary *)getJSON
```

#### Description

This method returns the data formatted as a JSON object. If there was no ongoing acquisition for any sensor when this object was created, `null` is returned.

#### Parameters

None.

### 4.8.4 Method getLastModified

#### Syntax

```
- (NSNumber *)getLastModified
```

#### Description

This method returns a timestamp that matches the maximum timestamp of the geographical position or the WiFi location. If neither has a timestamp, `null` is returned.

#### Parameters

None.

### 4.8.5 Method getTimezoneOffset

#### Syntax

```
- (NSNumber *)getTimezoneOffset
```

#### Description

This method returns the timezone offset, in minutes, that should be added to the user’s local time to provide the UTC time.

#### Parameters

None.

### 4.8.6 Method getWifiLocation

#### Syntax

```
- (WLWifiLocation *)getWifiLocation
```

#### Description

This method returns the last WiFi location acquired. If no location has been acquired, `null` is returned.

#### Parameters

None.

## 4.9 Class WLEventTransmissionPolicy

The event transmission policy is used to control how events are transmitted to the server.

The setters of this class return a reference to this object so as to enable chaining calls.

### 4.9.1 Method getDefaultPolicy

#### Syntax

```
+ (WLEventTransmissionPolicy *)getDefaultPolicy
```

#### Description

This method returns a new policy, with all fields set to default values.

#### Parameters

None.

### 4.9.2 Method getInterval

#### Syntax

```
- (long long)getInterval
```

#### Description

This method returns the transmission interval, in milliseconds.

#### Parameters

None.

### 4.9.3 Method isEventStorageEnabled

#### Syntax

```
- (BOOL)isEventStorageEnabled
```

#### Description

This method returns a Boolean value indicating whether events can be stored persistently. If events can be stored persistently, the value `true` is returned; otherwise, `false` is returned. The default value is `false`.

#### Parameters

None.

### 4.9.4 Method setEventStorageEnabled

#### Syntax

```
- (WLEventTransmissionPolicy
*)setEventStorageEnabled:(BOOL)eventStorageEnabled
```

#### Description

This method receives a Boolean value that determines where events are stored. If the value is `true`, events may be stored persistently. If the value is `false`, events that are waiting for transmission are stored in memory. The default value is `false`.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *eventStorageEnabled* | Boolean | The value to set. |

*Table 4-18: Method setEventStorageEnabled: parameters*

### 4.9.5 Method setInterval

#### Syntax

```
- (WLEventTransmissionPolicy *)setInterval:(long long)interval
```

#### Description

This method sets the transmission interval, in milliseconds. The default value is 60000 (one minute). Before events are transmitted, they are accumulated in memory, storage, or both.

#### Parameters

| Name | Type | Description |
|---|---|---|
| ***interval*** | long | The interval to set. |

*Table 4-19: Method setInterval: parameters*

## 4.10 Class WLGeoAcquisitionPolicy

This class controls how geographical positions are acquired.

The setters of this class return a reference to this object so as to enable chaining calls.

### 4.10.1 Method getLiveTrackingProfile

#### Syntax

```
+ (WLGeoAcquisitionPolicy *)getLiveTrackingProfile
```

#### Description

This method is used to track devices, and get the best position information available.

A policy is returned with the following preset values:

- `enableHighAccuracy = true`
- `maximumAge = 100` (100 milliseconds)

#### Parameters

None.

### 4.10.2 Method getPowerSavingProfile

#### Syntax

```
+ (WLGeoAcquisitionPolicy *)getPowerSavingProfile
```

#### Description

This method is used to track devices, and get the best position information available.

A policy is returned with the following preset values:

- `enableHighAccuracy = false`
- `minChangeDistance = 1000` (1 kilometer)
- `maximumAge = 300000` (5 minutes)

#### Parameters

None.

### 4.10.3 Method getRoughTrackingProfile

#### Syntax

```
+ (WLGeoAcquisitionPolicy *)getRoughTrackingProfile
```

#### Description

This method is used to track devices, but at a rough granularity.

A policy is returned with the following preset values:

- `enableHighAccuracy = true`
- `desired accuracy = 200` (200 meters)
- `minChangeDistance = 50` (50 meters)
- `maximumAge = 60000` (60 seconds)

#### Parameters

None.

### 4.10.4 Method getDesiredAccuracy

#### Syntax

```
- (int)getDesiredAccuracy
```

#### Description

This method returns the desired accuracy in meters. This value is taken into account only when `isEnableHighAccuracy` returns `true`.

#### Parameters

None.

### 4.10.5 Method getMaximumAge

#### Syntax

```
- (double)getMaximumAge
```

#### Description

This method returns the maximum age value. A cached position can be returned from the acquisition if the age of that position is less than the returned value. The default and minimum value is 100 milliseconds.

#### Parameters

None.

### 4.10.6 Method getMinChangeDistance

#### Syntax

```
- (int)getMinChangeDistance
```

#### Description

This method returns the minimum distance in meters that the position must change by, since the last update, in order to receive a new updated position. The default value is 0.

#### Parameters

None.

### 4.10.7 Method getTimeout

#### Syntax

```
- (long long)getTimeout
```

#### Description

This method returns the duration, in milliseconds, that the policy waits for acquisitions before a `WLGeoError` value is sent. A value of -1 is used to indicate an infinite timeout. -1 is the default value.

#### Parameters

None.

### 4.10.8 Method init

#### Syntax

```
- (id)init
```

#### Description

This method initializes the geographical position.

#### Parameters

None.

### 4.10.9 Method isEnableHighAccuracy

#### Syntax

```
- (BOOL)isEnableHighAccuracy
```

#### Description

If it is possible to obtain high-accuracy measurements, for example by using GPS, this method returns the Boolean value `true`. Otherwise it returns the value `false`.

#### Parameters

None.

### 4.10.10 Method setDesiredAccuracy

#### Syntax

```
- (WLGeoAcquisitionPolicy *)setDesiredAccuracy:(int)desiredAccuracy
```

#### Description

This method sets the desired accuracy in meters. The desired accuracy is only taken into account when `isEnableHighAccuracy` returns `true`.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *desiredAccuracy* | int | The desired accuracy setting. |

*Table 4-20: Method setDesiredAccuracy: parameters*

### 4.10.11 Method setEnableHighAccuracy

#### Syntax

```
- (WLGeoAcquisitionPolicy
*)setEnableHighAccuracy(BOOL)enableHighAccuracy
```

#### Description

This method controls whether it is possible to obtain high-accuracy measurements, for example by using GPS. When the Boolean value `true` is returned, the value of `getDesiredAccuracy` is taken into account.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *enableHighAccuracy* | Boolean | The `setEnableHighAccuracy` setting. |

*Table 4-21: Method setEnableHighAccuracy: parameters*

### 4.10.12 Method setMaximumAge

#### Syntax

```
- (WLGeoAcquisitionPolicy *)setMaximumAge:(long long)maximumAge
```

#### Description

This method sets the maximum age of positions returned, in milliseconds. A cached position can be returned from the acquisition if the age of that position is less than the specified value. The default and minimum value is 100 milliseconds.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *maximumAge* | `long` | The maximum age value. |

Table 4-22: Method setMaximumAge: parameters

### 4.10.13 Method setMinChangeDistance

#### Syntax

```
- (WLGeoAcquisitionPolicy *)setMinChangeDistance(int)minChangeDistance
```

#### Description

This method sets the minimum distance in meters that the position must change by, since the last update, in order to receive a new updated position. Higher values can improve battery life. The default value is 0.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *minChangeDistance* | `int` | The minimum distance in meters that the position must change by, since the last update, in order to receive a new updated position. |

Table 4-23: Method setMinChangeDistance: parameters

### 4.10.14 Method setTimeout

#### Syntax

```
- (WLGeoAcquisitionPolicy *)setTimeout:(long long)timeout
```

#### Description

This method sets the duration, in milliseconds, that the policy waits for acquisitions. A value of -1 is used to indicate an infinite timeout.

If no position is acquired since the last position was acquired, or since the `WLLocationServicesConfiguration` class was called, a failure function is called..

#### Parameters

| Name | Type | Description |
|---|---|---|
| *timeout* | `long` | The timeout interval for position acquisitions, in milliseconds. |

*Table 4-24: Method setTimeout: parameters*

## 4.11 Protocol WLGeoCallback

This protocol is used to define callbacks for when a geographical position is acquired.

### 4.11.1 Method execute:

#### Syntax

```
- (void)execute:(WLGeoPosition *)pos
```

#### Description

This method is executed when a geographical position is acquired.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *pos* | `WLGeoPosition` | The acquired geographical position. |

*Table 4-25: Method execute: parameters*

## 4.12 Class WLGeoDwellInsideTrigger

A trigger definition that is activated when the device remains inside an area for a specified period of time. In order to reactivate the trigger, the device must first leave the area, and then return to the area.

The setters of this class return a reference to this object so as to enable chaining calls.

### 4.12.1 Method getArea

#### Syntax

```
- (WLArea)getArea
```

#### Description

This method returns the area for the trigger, as specified by the `setArea` method.

#### Parameters

None.

### 4.12.2 Method getBufferZoneWidth

#### Syntax

```
- (double)getBufferZoneWidth
```

#### Description

This method returns the trigger's buffer zone width. The value indicates, in meters, how much the area is changed. The value can be positive or negative. If the value is positive, the area becomes bigger. If the value is negative, the area becomes smaller. All geofence triggers operate on this new area.

#### Parameters

None.

### 4.12.3 Method getCallback

#### Syntax

```
- (WLTriggerCallback)getCallback
```

#### Description

This method returns the callback object, whose execute method is called when the trigger is activated.

#### Parameters

None.

### 4.12.4 Method getConfidenceLevel

#### Syntax

```
- (WLConfidenceLevel)getConfidenceLevel
```

#### Description

This method returns the confidence level. This indicates how a position's accuracy is taken into account.

#### Parameters

None.

### 4.12.5 Method getDwellingTime

#### Syntax

```
- (long long)getDwellingTime
```

#### Description

This method returns the minimum time the device needs to be inside the area before the trigger is activated.

#### Parameters

None.

### 4.12.6 Method getEvent

#### Syntax

```
- (JSONObject)getEvent
```

#### Description

This method returns the event to transmit. If `null`, then it is not transmitted. The current device context is automatically added to the event when it is transmitted.

#### Parameters

None.

### 4.12.7 Method init

#### Syntax

```
- (id)init
```

#### Description

This method initializes the trigger definition.

#### Parameters

None.

### 4.12.8 Method isTransmitImmediately

#### Syntax

```
- (BOOL)isTransmitImmediately
```

#### Description

This method returns `true` if the event should be transmitted immediately.

#### Parameters

None.

### 4.12.9 Method setArea

#### Syntax

```
- (WLGeoDwellInsideTrigger *)setArea:(id<WLArea>)area
```

#### Description

This method sets the area for which the trigger will activate.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *area* | `id<WLArea>` | The area the user wants to set. This value is passed as a parameter to the execute method of this object, which should be an instance of `WLCircle` or `WLPolygon`. |

*Table 4-26: Method setArea: parameters*

### 4.12.10 Method setBufferZoneWidth

#### Syntax

```
- (WLGeoDwellInsideTrigger
*)setBufferZoneWidth:(double)bufferZoneWidth
```

#### Description

This method sets the buffer zone width, in meters. The buffer zone width value determines how much the area is changed. If the value is positive, the area becomes bigger. If the value is negative, the area becomes smaller. All geofence triggers operate on the new area.

The default value is 0, which leaves the area unchanged.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *bufferZoneWidth* | double | The buffer zone width the user wants to set. |

*Table 4-27: Method setBufferZoneWidth: parameters*

### 4.12.11 Method setCallback

#### Syntax

```
- (WLGeoDwellInsideTrigger
*)setCallback:(id<WLTriggerCallback>)callback
```

#### Description

This method sets the callback, whose execute method is called when the trigger is activated.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *callback* | id<WLTriggerCallback> | The callback the user wants to set. This parameter must conform to the `WLTriggerCallback` protocol. When the trigger is activated, its execute method will be called and the current device context is passed as a parameter. |

*Table 4-28: Method setCallback: parameters*

### 4.12.12 Method setConfidenceLevel

#### Syntax

```
- (WLGeoDwellInsideTrigger
*)setConfidenceLevel:(WLConfidenceLevel)confidenceLevel
```

#### Description

This method sets the confidence level. The value indicates how the accuracy of a position is taken into account.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *confidenceLevel* | `ConfidenceLevel` | The confidence level the user wants to set. |

*Table 4-29: Method setConfidenceLevel: parameters*

### 4.12.13 Method setDwellingTime

#### Syntax

```
- (WLGeoDwellInsideTrigger *)setDwellingTime:(long long)dwellingTime
```

#### Description

This method sets the time during which the device has dwelt within an area. The time is measured in milliseconds.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *dwellingTime* | `long` | The dwelling time in milliseconds. |

*Table 4-30: Method setDwellingTime: parameters*

### 4.12.14 Method setEvent

#### Syntax

```
- (WLGeoDwellInsideTrigger *)setEvent:(NSMutableDictionary *)event
```

#### Description

This method sets the event that is transmitted to the server.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *event* | `NSMutableDictionary` | The event the user wants to set. |

*Table 4-31: Method setEvent: parameters*

### 4.12.15 Method setTransmitImmediately

#### Syntax

```
- (WLGeoDwellInsideTrigger
*)setTransmitImmediately:(BOOL)transmitImmediately
```

#### Description

This method determines whether the event is transmitted immediately, or whether it is transmitted according to the transmission policy. If the value is `true`, the event is added to the transmission buffer, and the contents of the transmission buffer are flushed to the server. Otherwise the event is added only to the transmission buffer.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *`transmitImmediately`* | `Boolean` | A Boolean value that determines whether the event is transmitted immediately. |

*Table 4-32: Method setTransmitImmediately: parameters*

## 4.13 Class WLGeoDwellOutsideTrigger

A trigger definition that is activated when the device remains outside an area for a specified period of time. In order to reactivate the trigger, the device must first enter the area, and then leave the area again.

The setters of this class return a reference to this object so as to enable chaining calls.

### 4.13.1 Method getArea

#### Syntax

```
- (WLArea)getArea
```

#### Description

This method returns the area for the trigger, as specified by the `setArea` method.

#### Parameters

None.

### 4.13.2 Method getBufferZoneWidth

#### Syntax

```
- (double)getBufferZoneWidth
```

#### Description

This method returns the trigger's buffer zone width. The value indicates, in meters, how much the area is changed. The value can be positive or negative. If the value is positive, the area becomes bigger. If the value is negative, the area becomes smaller. All geofence triggers operate on this new area. .

#### Parameters

None.

### 4.13.3 Method getCallback

#### Syntax

```
- (WLTriggerCallback)getCallback
```

#### Description

This method returns the callback object, whose execute method is called when the trigger is activated.

#### Parameters

None.

### 4.13.4 Method getConfidenceLevel

#### Syntax

```
- (WLConfidenceLevel)getConfidenceLevel
```

#### Description

This method returns the confidence level. This indicates how a position's accuracy is taken into account.

#### Parameters

None.

### 4.13.5 Method getDwellingTime

#### Syntax

```
- (long long)getDwellingTime
```

#### Description

This method returns the minimum time the device needs to be outside the area before the trigger is activated.

#### Parameters

None.

### 4.13.6 Method getEvent

#### Syntax

```
- (JSONObject)getEvent
```

#### Description

This method returns the event to transmit. If `null`, then it is not transmitted. The current device context is automatically added to the event when it is transmitted.

#### Parameters

None.

### 4.13.7 Method init

#### Syntax

```
- (id)init
```

#### Description

This method initializes the trigger definition.

#### Parameters

None.

### 4.13.8 Method isTransmitImmediately

#### Syntax

```
- (BOOL)isTransmitImmediately
```

#### Description

This method returns `true` if the event should be transmitted immediately.

#### Parameters

None.

### 4.13.9 Method setArea

#### Syntax

```
- (WLGeoDwellOutsideTrigger *)setArea:(id<WLArea>) area
```

#### Description

This method sets the area for which the trigger will activate.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *`area`* | `id<WLArea>` | The area the user wants to set. This value is passed as a parameter to the execute method of this object, which should be an instance of `WLCircle` or `WLPolygon`. |

*Table 4-33: Method setArea: parameters*

### 4.13.10 Method setBufferZoneWidth

#### Syntax

```
- (WLGeoDwellOutsideTrigger
*)setBufferZoneWidth:(double)bufferZoneWidth
```

#### Description

This method sets the buffer zone width, in meters. The buffer zone width value determines how much the area is changed. If the value is positive, the area becomes bigger. If the value is negative, the area becomes smaller. All geofence triggers operate on the new area.

The default value is 0, which leaves the area unchanged.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *`bufferZoneWidth`* | `double` | The buffer zone width the user wants to |

| Name | Type | Description |
|---|---|---|
| | | set. |

Table 4-34: Method setBufferZoneWidth: parameters

### 4.13.11 Method setCallback

#### Syntax

```
- (WLGeoDwellOutsideTrigger
*)setCallback:(id<WLTriggerCallback>)callback
```

#### Description

This method sets the callback, whose execute method is called when the trigger is activated.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *callback* | `id<WLTriggerCallback>` | The callback the user wants to set. This parameter must conform to the `WLTriggerCallback` protocol. When the trigger is activated, its execute method will be called and the current device context is passed as a parameter. |

Table 4-35: Method setCallback: parameters

### 4.13.12 Method setConfidenceLevel

#### Syntax

```
- (WLGeoDwellOutsideTrigger *)setConfidenceLevel:(ConfidenceLevel)
confidenceLevel
```

#### Description

This method sets the confidence level. The value indicates how the accuracy of a position is taken into account.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *confidenceLevel* | `ConfidenceLevel` | The confidence level the user wants to set. |

Table 4-36: Method setConfidenceLevel: parameters

### 4.13.13 Method setDwellingTime

#### Syntax

```
- (WLGeoDwellOutsideTrigger *)setDwellingTime:(long long)dwellingTime
```

#### Description

This method sets the time during which the device has dwelt outside an area. The time is measured in milliseconds.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *dwellingTime* | long | The dwelling time in milliseconds. |

Table 4-37: Method setDwellingTime: parameters

### 4.13.14 Method setEvent

#### Syntax

```
- (WLGeoDwellOutsideTrigger *)setEvent:(NSMutableDictionary *)event
```

#### Description

This method sets the event that is transmitted to the server.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *event* | NSMutableDictionary | The event the user wants to set. |

Table 4-38: Method setEvent: parameters

### 4.13.15 Method setTransmitImmediately

#### Syntax

```
- (WLGeoDwellOutsideTrigger
*)setTransmitImmediately:(BOOL)transmitImmediately
```

#### Description

This method determines whether the event is transmitted immediately, or whether it is transmitted according to the transmission policy. If the value is `true`, the event is added to the transmission buffer, and the contents of the transmission buffer are flushed to the server. Otherwise the event is added only to the transmission buffer.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *`transmitImmediately`* | `Boolean` | A Boolean value that determines whether the event is transmitted immediately. |

*Table 4-39: Method setTransmitImmediately: parameters*

## 4.14 Class WLGeoEnterTrigger

A trigger definition that is activated when a device enters an area. To activate the trigger, the device must first have been outside the area, and then enter the area at the given confidence level.

The setters of this class return a reference to this object so as to enable chaining calls.

### 4.14.1 Method getArea

#### Syntax

```
- (WLArea)getArea
```

#### Description

This method returns the area for the trigger, as specified by the `setArea` method.

#### Parameters

None.

### 4.14.2 Method getBufferZoneWidth

#### Syntax

```
- (double)getBufferZoneWidth
```

#### Description

This method returns the trigger's buffer zone width. The value indicates, in meters, how much the area is changed. The value can be positive or negative. If the value is positive, the area becomes bigger. If the value is negative, the area becomes smaller. All geofence triggers operate on this new area.

#### Parameters

None.

### 4.14.3 Method getCallback

#### Syntax

```
- (WLTriggerCallback)getCallback
```

#### Description

This method returns the callback object, whose execute method is called when the trigger is activated.

#### Parameters

None.

### 4.14.4 Method getConfidenceLevel

#### Syntax

```
- (WLConfidenceLevel)getConfidenceLevel
```

#### Description

This method returns the confidence level. This indicates how a position's accuracy is taken into account.

#### Parameters

None.

### 4.14.5 Method getEvent

#### Syntax

```
- (JSONObject)getEvent
```

#### Description

This method returns the event to transmit. If `null`, then it is not transmitted. The current device context is automatically added to the event when it is transmitted.

#### Parameters

None.

### 4.14.6 Method init

#### Syntax

```
- (id)init
```

#### Description

This method initializes the trigger definition.

#### Parameters

None.

### 4.14.7 Method isTransmitImmediately

#### Syntax

```
- (BOOL)isTransmitImmediately
```

#### Description

This method returns `true` if the event should be transmitted immediately.

#### Parameters

None.

### 4.14.8 Method setArea

#### Syntax

```
- (WLGeoEnterTrigger *)setArea:(id<WLArea>)area
```

#### Description

This method sets the area for which the trigger will activate.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *area* | `id<WLArea>` | The area the user wants to set. |

*Table 4-40: Method setArea: parameters*

### 4.14.9 Method setBufferZoneWidth

#### Syntax

```
- (WLGeoEnterTrigger *)setBufferZoneWidth:(double)bufferZoneWidth
```

#### Description

This method sets the buffer zone width, in meters. The buffer zone width value determines how much the area is changed. If the value is positive, the area becomes bigger. If the value is negative, the area becomes smaller. All geofence triggers operate on the new area.

The default value is 0, which leaves the area unchanged.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *bufferZoneWidth* | `double` | The buffer zone width the user wants to set. |

*Table 4-41: Method setBufferZoneWidth: parameters*

### 4.14.10 Method setCallback

#### Syntax

```
- (WLGeoEnterTrigger *)setCallback:(id<WLTriggerCallback>)callback
```

#### Description

This method sets the callback, whose execute method is called when the trigger is activated.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *callback* | `WLTriggerCallback` | The callback the user wants to set. This parameter must conform to the `WLTriggerCallback` protocol. When the trigger is activated, its execute method will be called and the current device context is passed as a parameter. |

*Table 4-42: Method setCallback: parameters*

### 4.14.11 Method setConfidenceLevel

#### Syntax

```
- (WLGeoEnterTrigger
*)setConfidenceLevel:(ConfidenceLevel)confidenceLevel
```

#### Description

This method sets the confidence level. The value indicates how the accuracy of a position is taken into account.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *confidenceLevel* | `ConfidenceLevel` | The confidence level the user wants to set. |

*Table 4-43: Method setConfidenceLevel: parameters*

### 4.14.12 Method setEvent

#### Syntax

```
- (WLGeoEnterTrigger *)setEvent:(NSMutableDictionary *)event
```

#### Description

This method sets the event to be transmitted to the server.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *event* | `NSMutableDictionary` | The event the user wants to set. |

*Table 4-44: Method setEvent: parameters*

### 4.14.13 Method setTransmitImmediately

#### Syntax

```
- (WLGeoEnterTrigger
*)setTransmitImmediately:(BOOL)transmitImmediately
```

#### Description

This method determines whether the event is transmitted immediately, or whether it is transmitted according to the transmission policy. If the value is `true`, the event is added to the transmission buffer, and the contents of the transmission buffer are flushed to the server. Otherwise the event is added only to the transmission buffer.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *transmitImmediately* | Boolean | A Boolean value that determines whether the event is transmitted immediately. |

*Table 4-45: Method setTransmitImmediately: parameters*

## 4.15 Class WLGeoError

A WLGeoError object is created when an error is encountered during acquisition of a geographical position.

### 4.15.1 Method getErrorCode

#### Syntax

```
- (WLGeoErrorCodes)getErrorCode
```

#### Description

This method returns the error code.

#### Parameters

None.

### 4.15.2 Method getMessage

#### Syntax

```
- (NSString *)getMessage
```

#### Description

This method returns the message for the error.

#### Parameters

None.

### 4.15.3 Method initWithErrorCode:message

#### Syntax

```
- (id)initWithErrorCode:(WLGeoErrorCodes)errorcode message:(NSString
*)message
```

#### Description

This method returns the error code and the associated message.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *errorcode* | `WLGeoErrorCodes` | The error code. |
| *message* | `NSString` | The message for the error. |

*Table 4-46: Method initWithErrorCode:message: parameters*

## 4.16 Class WLGeoExitTrigger

A trigger definition that is activated when a device leaves an area. To activate the trigger, the device must first have been inside the area, and then leave the area at the given confidence level.

The setters of this class return a reference to this object so as to enable chaining calls.

### 4.16.1 Method getArea

#### Syntax

```
- (WLArea)getArea
```

#### Description

This method returns the area for the trigger, as specified by the `setArea` method.

#### Parameters

None.

### 4.16.2 Method getBufferZoneWidth

#### Syntax

```
- (double)getBufferZoneWidth
```

#### Description

This method returns the trigger's buffer zone width. The value indicates, in meters, how much the area is changed. The value can be positive or negative. If the value is positive, the area becomes bigger. If the value is negative, the area becomes smaller. All geofence triggers operate on this new area.

#### Parameters

None.

### 4.16.3 Method getCallback

#### Syntax

```
- (WLTriggerCallback)getCallback
```

#### Description

This method returns the callback object, whose execute method is called when the trigger is activated.

#### Parameters

None.

### 4.16.4 Method getConfidenceLevel

#### Syntax

```
- (WLConfidenceLevel)getConfidenceLevel
```

#### Description

This method returns the confidence level. This indicates how a position's accuracy is taken into account.

#### Parameters

None.

### 4.16.5 Method getEvent

#### Syntax

```
- (JSONObject)getEvent
```

#### Description

This method returns the event to transmit. If `null`, then it is not transmitted. The current device context is automatically added to the event when it is transmitted.

#### Parameters

None.

### 4.16.6 Method init

#### Syntax

```
- (id)init
```

#### Description

This method initializes the trigger definition.

#### Parameters

None.

### 4.16.7 Method isTransmitImmediately

#### Syntax

```
- (BOOL)isTransmitImmediately
```

#### Description

This method returns `true` if the event should be transmitted immediately.

#### Parameters

None.

### 4.16.8 Method setArea

#### Syntax

```
- (WLGeoExitTrigger *)setArea:(id<WLArea>)area
```

#### Description

This method sets the area for which the trigger will activate.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *area* | id<WLArea> | The area the user wants to set. |

*Table 4-47: Method setArea: parameters*

### 4.16.9 Method setBufferZoneWidth

#### Syntax

```
- (WLGeoExitTrigger *)setBufferZoneWidth:(double)bufferZoneWidth
```

#### Description

This method sets the buffer zone width, in meters. The buffer zone width value determines how much the area is changed. If the value is positive, the area becomes bigger. If the value is negative, the area becomes smaller. All geofence triggers operate on the new area.

The default value is 0, which leaves the area unchanged.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *bufferZoneWidth* | double | The buffer zone width the user wants to set. |

*Table 4-48: Method setBufferZoneWidth: parameters*

### 4.16.10 Method setCallback

#### Syntax

```
- (WLGeoExitTrigger *)setCallback:(id<WLTriggerCallback>)callback
```

#### Description

This method sets the callback, whose execute method is called when the trigger is activated.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *callback* | `WLTriggerCallback` | The callback the user wants to set. This parameter must conform to the `WLTriggerCallback` protocol. When the trigger is activated, its execute method will be called and the current device context is passed as a parameter |

*Table 4-49: Method setCallback: parameters*

### 4.16.11 Method setConfidenceLevel

#### Syntax

```
- (WLGeoExitTrigger
*)setConfidenceLevel:(ConfidenceLevel)confidenceLevel
```

#### Description

This method sets the confidence level. The value indicates how the accuracy of a position is taken into account.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *confidenceLevel* | `ConfidenceLevel` | The confidence level the user wants to set. |

*Table 4-50: Method setConfidenceLevel: parameters*

### 4.16.12 Method setEvent

#### Syntax

```
- (WLGeoExitTrigger *)setEvent:(NSMutableDictionary *)event
```

#### Description

This method sets the event to be transmitted to the server.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *event* | `NSMutableDictionary` | The event the user wants to set. |

*Table 4-51: Method setEvent: parameters*

### 4.16.13 Method setTransmitImmediately

#### Syntax

```
- (WLGeoExitTrigger
*)setTransmitImmediately:(BOOL)transmitImmediately
```

#### Description

This method determines whether the event is transmitted immediately, or whether it is transmitted according to the transmission policy. If the value is `true`, the event is added to the transmission buffer, and the contents of the transmission buffer are flushed to the server. Otherwise the event is added only to the transmission buffer.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *transmitImmediately* | `Boolean` | A Boolean value that determines whether the event is transmitted immediately. |

*Table 4-52: Method setTransmitImmediately: parameters*

## 4.17 Protocol WLGeoFailureCallback

This protocol is invoked when an error occurs during a geolocation acquisition.

### 4.17.1 execute

#### Syntax

```
- (void)execute:WLGeoError *errorObject
```

#### Description

This method is executed when an error occurs during acquisition.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *errorObject* | WLGeoError | The error that occurred. |

*Table 4-53: Method execute: parameters*

## 4.18 Class WLGeoPosition

This class provides the acquired geolocation coordinate.

### 4.18.1 Method getCoordinate

#### Syntax

```
- (WLCoordinate *)getCoordinate
```

#### Description

This method returns the acquired coordinate.

#### Parameters

None.

### 4.18.2 Method init

#### Syntax

```
- (id)init
```

#### Description

This method initializes the acquired coordinate.

#### Parameters

None.

### 4.18.3 Method initWithCoordinate:acquisitionTime:

#### Syntax

```
- (id)initWithCoordinate:(WLCoordinate *)coordinate
acquistionTime:(long long)acquisitionTime
```

#### Description

Creates a new `WLGeoPosition` instance, with an acquisition time.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *`coordinate`* | `WLCoordinate` | The acquired coordinate. |
| *`acquisitionTime`* | `long long` | The time of the acquisition. |

*Table 4-54: Method initWithCoordinate:acquisitionTime: parameters*

## 4.19 Class WLGeoPositionChangeTrigger

A trigger for tracking changes in the position of a device. You can specify a minimum distance that must be moved before the trigger will activate.

The setters of this class return a reference to this object so as to enable chaining calls.

### 4.19.1 Method getCallback

#### Syntax

```
- (WLTriggerCallback)getCallback
```

#### Description

This method returns the callback object, whose execute method is called when the trigger is activated.

#### Parameters

None.

### 4.19.2 Method getEvent

#### Syntax

```
- (JSONObject)getEvent
```

#### Description

This method returns the event to transmit. If `null`, then it is not transmitted. The current device context is automatically added to the event when it is transmitted.

#### Parameters

None.

### 4.19.3 Method getMinChangeDistance

#### Syntax

```
- (double)getMinChangeDistance
```

#### Description

This method returns the sensitivity setting, in terms of a minimum change distance measured in meters, and set by the `setMinChangeDistance` method.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *`getMinChangeDistance`* | `double` | The sensitivity setting. |

*Table 4-55: Method getMinChangeDistance: parameters*

### 4.19.4 Method init

#### Syntax

```
- (id)init
```

#### Description

This method initializes the trigger definition.

#### Parameters

None.

### 4.19.5 Method isTransmitImmediately

#### Syntax

```
- (BOOL)isTransmitImmediately
```

#### Description

This method returns `true` if the event should be transmitted immediately.

#### Parameters

None.

### 4.19.6 Method setCallback

#### Syntax

```
- (WLGeoPositionChangeTrigger *)
setCallback:(id<WLTriggerCallback>)callback
```

#### Description

This method sets the callback, whose execute method is called when the trigger is activated.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *callback* | `WLTriggerCallback` | The callback the user wants to set. This parameter must conform to the `WLTriggerCallback` protocol. When the trigger is activated, its execute method will be called and the current device context is passed as a parameter. |

*Table 4-56: Method setCallback: parameters*

### 4.19.7 Method setEvent

#### Syntax

```
- (WLGeoPositionChangeTrigger *)setEvent:(NSMutableDictionary *)event
```

#### Description

This method sets the event to be transmitted to the server.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *event* | `NSMutableDictionary` | The event the user wants to set. |

*Table 4-57: Method setEvent: parameters*

### 4.19.8 Method setMinChangeDistance

#### Syntax

```
- (WLGeoPositionChangeTrigger
*)setMinChangeDistance:(double)minChangeDistance
```

#### Description

After the first acquisition, this trigger is activated only when the reported position has changed by at least the value of the `minChangeDistance` parameter. This behaviour is different from setting the parameter in the `WLGeoAcquisitionPolicy` class, because other triggers might still activate, due to changes in the position of the device, and no power is saved by using this method.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *`minChangeDistance`* | `double` | The minimum distance in meters by which the position must change in order to activate the trigger. The value must be greater than that of the `minChangeDistance` parameter set for the `WLGeoAcquisitionPolicy` class, otherwise it has no effect. |

*Table 4-58: Method setMinChangeDistance: parameters*

### 4.19.9 Method setTransmitImmediately

#### Syntax

```
- (WLGeoPositionChangeTrigger
*)setTransmitImmediately:(BOOL)transmitImmediately
```

#### Description

This method determines whether the event is transmitted immediately, or whether it is transmitted according to the transmission policy. If the value is `true`, the event is added to the transmission buffer, and the contents of the transmission buffer are flushed to the server. Otherwise the event is added only to the transmission buffer.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *`transmitImmediate`* | `Boolean` | A Boolean value that determines whether |

| Name | Type | Description |
|---|---|---|
| *ly* | | the event is transmitted immediately. |

*Table 4-59: Method setTransmitImmediately: parameters*

## 4.20 Class WLGeoTrigger

A class that defines a trigger that handles geographical positions. This is the parent interface for all the other geo triggers.

## 4.21 Class WLGeoUtils

This class provides access to utility functions for geolocation calculations.

### 4.21.1 Method getDistanceFromCoordinate:toArea

#### Syntax

```
+ (double)getDistanceFromCoordinate:(WLCoordinate *)coordinate
toArea(id<WLArea>)area
```

#### Description

This method calculates and returns the distance in meters from the coordinate to the area. The distance is positive for coordinates outside the area and negative for coordinates inside the area. This method is equivalent to calling `getDistanceFromCoordinate:toArea:bufferZoneWidth` with a value of 0 for the `bufferZoneWidth` parameter.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *coordinate* | `WLCoordinate` | The coordinate. |
| *area* | `id<WLArea>` | The area. |

*Table 4-60: Method getDistanceFromCoordinate:toArea: parameters*

### 4.21.2 Method getDistanceFromCoordinate:toArea:bufferZoneWidth

#### Syntax

```
+ (double)getDistanceFromCoordinate:(WLCoordinate *)coordinate
toArea(id<WLArea>)area bufferZoneWidth:(double)bufferZoneWidth
```

#### Description

This method calculates and returns the distance in meters of the coordinate from the area, taking into account the buffer zone. The distance is positive for coordinates outside the area and negative for coordinates inside the area.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *coordinate* | WLCoordinate | The coordinate. |
| *area* | id<WLArea> | The area. |
| *bufferZoneWidth* | double | The buffer zone width, measured in meters. The size of the area is increased, in all directions, by this value. Negative values decrease the size of the area. |

*Table 4-61: Method getDistanceFromCoordinate:toArea:bufferZoneWidth: parameters*

### 4.21.3 Method getDistanceFromCoordinate:toCircle

#### Syntax

```
+ (double)getDistanceFromCoordinate:(WLCoordinate *)coordinate
toCircle:(WLCircle *)circle
```

#### Description

This method calculates and returns the distance in meters of the coordinate from the circle. The distance is positive for coordinates outside the circle and negative for coordinates inside the circle.

This method is equivalent to calling `getDistanceFromCoordinate:toCircle:bufferZoneWidth` with a value of 0 for the `bufferZoneWidth` parameter.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *coordinate* | WLCoordinate | The coordinate. |
| *circle* | WLCircle | The ID of the required circle. |

*Table 4-62: Method getDistanceFromCoordinate:toCircle: parameters*

### 4.21.4 Method getDistanceFromCoordinate:toCircle:bufferZoneWidth

#### Syntax

```
+ (double)getDistanceFromCoordinate:(WLCoordinate *)coordinate
toCircle:(WLCircle *)circle bufferZoneWidth:(double)bufferZoneWidth
```

#### Description

This method calculates and returns the distance, in meters, of the coordinate from the circle, taking into account the buffer zone. The distance is positive for coordinates outside the circle and negative for coordinates inside the circle.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *coordinate* | `WLCoordinate` | The coordinate. |
| *circle* | `WLCircle` | The circle. |
| *bufferZoneWidth* | `double` | The buffer zone width, measured in meters. The radius of the circle is increased by this value. Negative values make the circle smaller. |

*Table 4-63: Method getDistanceFromCoordinate:toCircle:bufferZoneWidth parameters*

### 4.21.5 Method getDistanceFromCoordinate:toCoordinate

#### Syntax

```
+ (double)getDistanceFromCoordinate:(WLCoordinate *)coordinate1
toCoordinate:(WLCoordinate *)coordinate2
```

#### Description

This method calculates and returns the distance between two coordinates. The result is returned in meters, using a spherical model of the Earth.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *coordinate1* | WLCoordinate | The first coordinate. |
| *coordinate2* | WLCoordinate | The second coordinate. |

*Table 4-64: Method getDistanceFromCoordinate:toCoordinate: parameters*

### 4.21.6 Method getDistanceFromCoordinate:toPolygon

#### Syntax

```
+ (double)getDistanceFromCoordinate:(WLCoordinate *)coordinate
toPolygon:(WLPolygon *)polygon
```

#### Description

This method calculates and returns the distance, in meters, from the coordinate to the polygon. The distance is positive for coordinates outside the polygon and negative for coordinates inside the polygon.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *coordinate* | WLCoordinate | The coordinate. |
| *polygon* | WLPolygon | The polygon. |

*Table 4-65: Method getDistanceFromCoordinate:toPolygon: parameters*

### 4.21.7 Method getDistanceFromCoordinate:toPolygon:bufferZoneWidth:

#### Syntax

```
+ (double)getDistanceFromCoordinate:(WLCoordinate *)coordinate
toPolygon:(WLPolygon *)polygon
bufferZoneWidth:(double)bufferZoneWidth
```

#### Description

This method calculates and returns the distance, in meters, from the coordinate to the polygon, taking into account the buffer zone. The distance is positive for coordinates outside the polygon and negative for coordinates inside the polygon.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *coordinate* | WLCoordinate | The coordinate. |
| *polygon* | WLPolygon | The polygon. |
| *bufferZoneWidth* | double | The buffer zone width, measured in meters. The size of the polygon is increased, in all directions, by this value. Negative values decrease the size of the polygon. |

*Table 4-66: Method getDistanceFromCoordinate:toPolygon: parameters*

### 4.21.8 Method isCoordinate:insideArea:

#### Syntax

```
+ (BOOL)isCoordinate:(WLCoordinate *)coordinate
insideArea:(id<WLArea>)area
```

#### Description

This method returns a Boolean value based on whether a coordinate lies within an area. The value `true` is returned if the coordinate lies within the area, at the given level of confidence. The dimensions of the area used in this check incorporate any changes specified for the `bufferZoneWidth` parameter.

This method is equivalent to calling `isCoordinate:insideArea:bufferZoneWidth:confidenceLevel` with a value of 0 for the `bufferZoneWidth` parameter and LOW for the `confidenceLevel` parameter.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *coordinate* | WLCoordinate | The coordinate. |
| *area* | id<WLArea> | The area. |

*Table 4-67: Method isCoordinate:insideArea: parameters*

### 4.21.9 Method isCoordinate:insideArea:bufferZoneWidth:confidenceLevel

#### Syntax

```
+ (BOOL)isCoordinate:(WLCoordinate *)coordinate
insideArea:(id<WLArea>)area bufferZoneWidth:(double)bufferZoneWidth
confidenceLevel:(WLConfidenceLevel)confidenceLevel
```

#### Description

This method returns a Boolean value based on whether a coordinate lies within an area, taking into account the buffer zone and the confidence level. The value `true` is returned if the coordinate lies within the area, at the given level of confidence. The dimensions of the area used in this check incorporate any changes specified for the `bufferZoneWidth` parameter.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *coordinate* | WLCoordinate | The coordinate. |
| *area* | id<WLArea> | The area. |
| *bufferZoneWidth* | double | The buffer zone width, measured in meters. The size of the area is increased, in all directions, by this value. Negative values decrease the size of the area. |
| *confidenceLevel* | WLConfidenceLevel | The level of confidence indicates how accuracy is taken into account. |

*Table 4-68: Method isCoordinate:insideArea:bufferZoneWidth:confidenceLevel parameters*

### 4.21.10 Method isCoordinate:insideCircle

#### Syntax

```
+ (BOOL)isCoordinate:(WLCoordinate *)coordinate
insideCircle:(WLCircle *)circle
```

#### Description

This method returns a Boolean value based on whether a coordinate lies within a circle. The value `true` is returned if the coordinate lies within the circle.

This method is equivalent to calling `isCoordinate:insideCircle:bufferZoneWidth:confidenceLevel` with a value of 0 for the `bufferZoneWidth` parameter and LOW for the `confidenceLevel` parameter.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *coordinate* | WLCoordinate | The coordinate. |
| *circle* | WLCircle | The circle. |

*Table 4-69: Method isCoordinate:insideCircle: parameters*

### 4.21.11 Method isCoordinate:insideCircle:bufferZoneWidth:confidenceLevel

#### Syntax

```
+ (BOOL)isCoordinate:(WLCoordinate *)coordinate
insideCircle:(WLCircle *)circle
bufferZoneWidth:(double)bufferZoneWidth
confidenceLevel:(WLConfidenceLevel)confidenceLevel
```

#### Description

This method returns a Boolean value based on whether a coordinate lies within a circle, taking into account the buffer zone and the confidence level. The value `true` is returned if the coordinate lies within the circle, at the given level of confidence. The dimensions of the circle used in this check incorporate any changes specified for the `bufferZoneWidth` parameter.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *coordinate* | WLCoordinate | The coordinate. |
| *circle* | WLCircle | The circle. |
| *bufferZoneWidth* | double | The buffer zone width, measured in meters. The radius of the circle is increased by this value. Negative values make the circle smaller. |

| Name | Type | Description |
|---|---|---|
| *confidenceLevel* | WLConfidenceLevel | The level of confidence indicates how accuracy is taken into account. |

*Table 4-70: Method isCoordinate:insideCircle:bufferZoneWidth:confidenceLevel parameters*

### 4.21.12 Method isCoordinate:insidePolygon

#### Syntax

```
+ (BOOL)isCoordinate:(WLCoordinate *)coordinate
insidePolygon:(WLPolygon *)polygon
```

#### Description

This method returns a Boolean value based on whether a coordinate lies within a polygon. The value `true` is returned if the coordinate lies within the polygon.

This method is equivalent to calling `isCoordinate:insidePolygon:bufferZoneWidth:confidenceLevel` with a value of 0 for the `bufferZoneWidth` parameter and LOW for the `confidenceLevel` parameter.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *coordinate* | WLCoordinate | The coordinate. |
| *polygon* | WLPolygon | The circle. |

*Table 4-71: Method isCoordinate:insidePolygon: parameters*

### 4.21.13 Method isCoordinate:insidePolygon:bufferZoneWidth:confidenceLevel

#### Syntax

```
+ (BOOL)isCoordinate:(WLCoordinate *)coordinate
insidePolygon:(WLPolygon *)polygon
bufferZoneWidth:(double)bufferZoneWidth
confidenceLevel:(WLConfidenceLevel)confidenceLevel
```

#### Description

This method returns a Boolean value based on whether a coordinate lies within a polygon, taking into account the buffer zone and the confidence level. The value `true` is returned if the coordinate lies within the polygon, at the given level of confidence. The dimensions of the polygon used in this check incorporate any changes specified for the `bufferZoneWidth` parameter.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *coordinate* | WLCoordinate | The coordinate. |
| *polygon* | WLPolygon | The polygon. |
| *bufferZoneWidth* | double | The buffer zone width, measured in meters. The radius of the polygon is increased by this value. Negative values decrease the size of the polygon. |
| *confidenceLevel* | WLConfidenceLevel | The level of confidence indicates how accuracy is taken into account. |

*Table 4-72: Method isCoordinate:insidePolygon:bufferZoneWidth:confidenceLevel parameters*

### 4.21.14 Method isCoordinate:outsideArea

#### Syntax

```
+ (BOOL)isCoordinate:(WLCoordinate *)coordinate
outsideArea:(id<WLArea>)area
```

#### Description

This method returns a Boolean value based on whether a coordinate lies outside an area, at the given level of confidence. The value `true` is returned if the coordinate lies outside the area at the given level of confidence. The dimensions of the area used in this check incorporate any changes specified for the `bufferZoneWidth` parameter.

This method is equivalent to calling `isCoordinate:outsideArea:bufferZoneWidth:confidenceLevel` with a value of 0 for the `bufferZoneWidth` parameter and LOW for the `confidenceLevel` parameter.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *coordinate* | `WLCoordinate` | The coordinate. |
| *area* | `id<WLArea>` | The area. |

*Table 4-73: Method isCoordinate:outsideArea: parameters*

### 4.21.15 Method isCoordinate:outsideArea:bufferZoneWidth:confidenceLevel

#### Syntax

```
+ (BOOL)isCoordinate:(WLCoordinate *)coordinate
outsideArea:(id<WLArea>)area bufferZoneWidth:(double)bufferZoneWidth
confidenceLevel:(WLConfidenceLevel)confidenceLevel
```

#### Description

This method returns a Boolean value based on whether a coordinate lies outside an area, taking into account the buffer zone and the confidence level. The value `true` is returned if the coordinate lies within the area, at the given level of confidence. The dimensions of the area used in this check incorporate any changes specified for the `bufferZoneWidth` parameter.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *coordinate* | `WLCoordinate` | The coordinate. |
| *area* | `id<WLArea>` | The area. |
| *bufferZoneWidth* | `double` | The buffer zone width, measured in meters. The size of the area is increased, in all directions, by this value. Negative values decrease the size of the area. |
| *confidenceLevel* | `WLConfidenceLevel` | The level of confidence indicates how accuracy is taken into account. |

*Table 4-74: Method isCoordinate:outsideArea:bufferZoneWidth:confidenceLevel parameters*

### 4.21.16 Method isCoordinate:outsideCircle

#### Syntax

```
+ (BOOL)isCoordinate:(WLCoordinate *)coordinate
outsideCircle:(WLCircle *)circle
```

#### Description

This method returns a Boolean value based on whether a coordinate lies outside a circle. The value `true` is returned if the coordinate lies outside the circle.

This method is equivalent to calling `isCoordinate:outsideCircle:bufferZoneWidth:confidenceLevel` with a value of 0 for the `bufferZoneWidth` parameter and LOW for the `confidenceLevel` parameter.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *coordinate* | `WLCoordinate` | The coordinate. |
| *circle* | `WLCircle` | The circle. |

*Table 4-75: Method isCoordinate:outsideCircle: parameters*

### 4.21.17 Method isCoordinate:outsideCircle:bufferZoneWidth:confidenceLevel

#### Syntax

```
+ (BOOL)isCoordinate:(WLCoordinate *)coordinate
outsideCircle:(WLCircle *)circle
bufferZoneWidth:(double)bufferZoneWidth
confidenceLevel:(WLConfidenceLevel)confidenceLevel
```

#### Description

This method returns a Boolean value based on whether a coordinate lies outside a circle, taking into account the buffer zone and the confidence level. The value `true` is returned if the coordinate lies within the circle, at the given level of confidence. The dimensions of the circle used in this check incorporate any changes specified for the `bufferZoneWidth` parameter.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *coordinate* | WLCoordinate | The coordinate. |
| *circle* | WLCircle | The circle. |
| *bufferZoneWidth* | double | The buffer zone width, measured in meters. The radius of the circle is increased by this value. Negative values make the circle smaller. |
| *confidenceLevel* | WLConfidenceLevel | The level of confidence indicates how accuracy is taken into account. |

*Table 4-76: Method isCoordinate:outsideCircle:bufferZoneWidth:confidenceLevel parameters*

### 4.21.18 Method isCoordinate:outsidePolygon

#### Syntax

```
+ (BOOL)isCoordinate:(WLCoordinate *)coordinate
outsidePolygon:(WLPolygon *)polygon
```

#### Description

This method returns a Boolean value based on whether a coordinate lies outside a polygon. The value `true` is returned if the coordinate lies outside the polygon.

This method is equivalent to calling `isCoordinate:outsidePolygon:bufferZoneWidth:confidenceLevel` with a value of 0 for the `bufferZoneWidth` parameter and LOW for the `confidenceLevel` parameter.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *coordinate* | WLCoordinate | The coordinate. |
| *polygon* | WLPolygon | The polygon. |

*Table 4-77: Method isCoordinate:outsidePolygon: parameters*

### 4.21.19 Method isCoordinate:outsidePolygon:bufferZoneWidth:confidenceLevel

#### Syntax

```
+ (BOOL)isCoordinate:(WLCoordinate *)coordinate
outsidePolygon:(WLPolygon *)polygon
bufferZoneWidth:(double)bufferZoneWidth
confidenceLevel:(WLConfidenceLevel)confidenceLevel
```

#### Description

This method returns a Boolean value based on whether a coordinate lies outside a polygon, taking into account the buffer zone and the confidence level. The value `true` is returned if the coordinate lies within the polygon, at the given level of confidence. The dimensions of the polygon used in this check incorporate any changes specified for the `bufferZoneWidth` parameter.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *coordinate* | `WLCoordinate` | The coordinate. |
| *polygon* | `WLPolygon` | The polygon. |
| *bufferZoneWidth* | `double` | The buffer zone width, measured in meters. The size of the polygon is increased, in all directions, by this value. Negative values make the polygon smaller. |
| *confidenceLevel* | `WLConfidenceLevel` | The level of confidence indicates how accuracy is taken into account. |

*Table 4-78: Method isCoordinate:outsidePolygon:bufferZoneWidth:confidenceLevel parameters*

## 4.22 Class WLLocationServicesConfiguration

This class is the configuration for ongoing acquisition, including the acquisition policy, triggers, and failure callbacks for handling acquisition errors.

The setters of this class return a reference to this object so as to enable chaining calls.

### 4.22.1 Method getFailureCallbacks

#### Syntax

```
- (NSMutableArray *)getFailureCallbacks
```

#### Description

This method returns the failure callbacks. During ongoing acquisition, the failure callbacks are called whenever errors occur.

#### Parameters

None.

### 4.22.2 Method getPolicy

#### Syntax

```
- (WLAcquisitionPolicy *)getPolicy
```

#### Description

This method returns the acquisition policy.

#### Parameters

None.

### 4.22.3 Method getTriggers

#### Syntax

```
- (WLTriggersConfiguration *)getTriggers
```

#### Description

This method returns the trigger configurations.

#### Parameters

None.

### 4.22.4 Method init

#### Syntax

```
- (id)init
```

#### Description

This method initializes the configuration.

#### Parameters

None.

### 4.22.5 Method setFailureCallbacks

#### Syntax

```
- (WLLocationServicesConfiguration
*)setFailureCallbacks:(NSMutableArray *)failureCallbacks
```

#### Description

During ongoing acquisition, the failure callbacks are called whenever errors occur.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *failureCallbacks* | NSMutableArray | The failure callbacks. |

*Table 4-79: Method setFailureCallbacks: parameters*

### 4.22.6 Method setPolicy

#### Syntax

```
- (WLLocationServicesConfiguration *)setPolicy:(WLAcquisitionPolicy
*)policy
```

#### Description

This method sets the acquisition policy.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *policy* | WLAcquisitionPolicy | The acquisition policy to set. |

*Table 4-80: Method setPolicy: parameters*

### 4.22.7 Method setTriggers

#### Syntax

```
- (WLLocationServicesConfiguration
*)setTriggers:(WLTriggersConfiguration *)triggers
```

#### Description

This method sets the triggers that are evaluated during ongoing acquisition.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *triggers* | `WLTriggersConfiguration` | The triggers to be evaluated. |

*Table 4-81: Method setTriggers: parameters*

## 4.23 Class WLPolygon

A polygon that is defined by a list of coordinates. This class is immutable.

### 4.23.1 Method get

#### Syntax

```
- (WLCoordinate *)get:(int)idx
```

#### Description

This method returns the index of the coordinate to retrieve.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *idx* | `int` | The index of the coordinate to retrieve. |

*Table 4-82: Method get: parameters*

### 4.23.2 Method getCoordinates

#### Syntax

```
- (NSMutableArray *)getCoordinates
```

#### Description

This method returns a copy of the coordinates that make up this polygon.

#### Parameters

None.

## 4.24 Protocol WLTriggerCallback

This protocol defines callbacks for when a trigger is activated.

### 4.24.1 Method execute

#### Syntax

```
- (void)execute:(id<WLDeviceContext>)deviceContext
```

#### Description

This method is executed when the trigger is activated.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *deviceContext* | `id<WLDeviceContext>` | The device context at the time the trigger was activated. |

*Table 4-83: Method execute: parameters*

## 4.25 Class WLTriggersConfiguration

A configuration object that contains the triggers. The policy should be set in an instance of the `WLLocationServicesConfiguration` class, which is then passed to the `startAcquisition` method of the `WLDevice` protocol.

The setters of this class return a reference to this object so as to enable chaining calls.

### 4.25.1 Method getGeoTriggers

#### Syntax

```
- (NSMutableDictionary *)getGeoTriggers
```

#### Description

This method returns the geolocation triggers.

#### Parameters

None.

### 4.25.2 Method getWifiTriggers

#### Syntax

```
- (NSMutableDictionary *)getWifiTriggers
```

#### Description

This method returns the WiFi triggers.

#### Parameters

None.

### 4.25.3 Method init

#### Syntax

```
- (id)init
```

#### Description

This method creates a new instance with default (empty) triggers.

#### Parameters

None.

### 4.25.4 Method setGeoTriggers

#### Syntax

```
- (WLTriggersConfiguration *)setGeoTriggers:(NSMutableDictionary
*)geoTriggers
```

#### Description

This method sets the geolocation triggers.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *geoTriggers* | NSMutableDictionary | The new triggers to set. Each trigger needs a unique key in the map. If the value is `null`, an empty map is set. |

*Table 4-84: Method setGeoTriggers: parameters*

### 4.25.5 Method setWifiTriggers

#### Syntax

```
- (WLTriggersConfiguration *)setWifiTriggers:(NSMutableDictionary
*)wifiTriggers
```

#### Description

This method sets the WiFi triggers.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *wifiTriggers* | NSMutableDictionary | The new triggers to set. Each trigger needs a unique key in the map. If the value is `null`, an empty map is set. |

*Table 4-85: Method setWifiTriggers: parameters*

## 4.26 Class WLWifiAccessPoint

This class returns a WiFi access point.

### 4.26.1 Method getMAC

#### Syntax

```
- (NSString *)getMAC
```

#### Description

This method returns the MAC address of the access point.

#### Parameters

None.

### 4.26.2 Method getSSID

#### Syntax

```
- (NSString *)getSSID
```

#### Description

This method returns the SSID of the access point.

#### Parameters

None.

### 4.26.3 Method initWithSSID:MAC

#### Syntax

```
- (id)initWithSSID:(NSString *)SSID MAC:(NSString *)MAC
```

#### Description

This method initializes the SSID and MAC address.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *SSID* | NSString | The SSID of the access point. |
| *MAC* | NSString | The MAC address of the access point. |

*Table 4-86: Method initWithSSID:MAC: parameters*

## 4.27 Class WLWifiAccessPointFilter

This class specifies which WiFi access points to detect.

The filter has an SSID name specification, and optionally a MAC specification. The MAC specification can be a wildcard value, represented by `WILDCARD` in the form of an asterisk (*). When the wildcard value is used, all MACs for the SSID are reported.

If no MAC is specified, and the given SSID is visible, then only its SSID is reported; only a single `WLWifiAccessPoint` instance results from an acquisition, regardless of the number of access points that have the SSID.

The SSID specification can also be a wildcard, in which case all visible networks are reported.

### 4.27.1 Method getMac

#### Syntax

```
- (NSString *)getMAC
```

#### Description

This method returns the MAC specification.

#### Parameters

None.

### 4.27.2 Method getSsid

#### Syntax

```
- (NSString *)getSSID
```

#### Description

This method returns the SSID specification.

#### Parameters

None.

### 4.27.3 Method init

#### Syntax

```
- (id)init:(NSString *)ssid
```

#### Description

This method creates a new filter that accepts any MAC address.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *ssid* | NSString | The SSID specification, which cannot be null. |

*Table 4-87: Method init: parameters*

#### 4.27.4 Method initWithSSID:MAC

##### Syntax

```
- (id)initWithSSID:(NSString *)ssid MAC:(NSString *)mac
```

##### Description

This method creates a new filter.

##### Parameters

| Name | Type | Description |
|---|---|---|
| *ssid* | `NSString` | The SSID specification, which cannot be null. |
| *mac* | `NSString` | The MAC specification. |

*Table 4-88: Method initWithSSID:MAC: parameters*

### 4.28 Protocol WLWifiAcquisitionCallback

This protocol receives the connected access point information as part of the `getConnectedAccessPointFilteredByPolicy` method of the `WLDevice` protocol.

#### 4.28.1 Method execute

##### Syntax

```
- (void)execute:(NSMutableArray *)accessPoints
```

##### Description

This method is executed when the list of visible WiFi access points is acquired.

##### Parameters

| Name | Type | Description |
|---|---|---|
| *accessPoints* | `NSMutableArray` | The visible access points acquired. |

*Table 4-89: Method execute: parameters*

## 4.29 Class WLWifiAcquisitionPolicy

This class controls how WiFi locations are acquired.

The setters of this class return a reference to this object so as to enable chaining calls.

### 4.29.1 Method getAccessPointFilters

#### Syntax

```
- (NSMutableArray *)getAccessPointFilters
```

#### Description

This method returns the access point filters. Only WiFi access points that match at least one of the filters are visible.

#### Parameters

None.

### 4.29.2 Method getInterval

#### Syntax

```
- (int)getInterval
```

#### Description

This method returns the polling interval, in milliseconds. WiFi polling is performed each interval. The default value is 10000 (10 seconds).

#### Parameters

None.

### 4.29.3 Method init

#### Syntax

```
- (id)init
```

#### Description

This method initializes the WiFi acquisition policy setting.

#### Parameters

None.

### 4.29.4 Method setAccessPointFilters

#### Syntax

```
- (WLWifiAcquisitionPolicy *)setAccessPointFilters:(NSMutableArray
*)accessPointFilters
```

#### Description

This method sets the access point filters. Only WiFi access points that match one of the access point filters are visible. If the connected access point does not match any of the filters, it is not visible during ongoing acquisition. If `null`, the list is considered to be empty.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *accessPointFilters* | NSMutableArray | The access point filters. |

*Table 4-90: Method setAccessPointFilters: parameters*

### 4.29.5 Method setInterval

#### Syntax

```
- (WLWifiAcquisitionPolicy *)setInterval:(int)interval
```

#### Description

This method sets a polling interval, specified in milliseconds. WiFi polling is performed each interval. The default value is 10000 (10 seconds).

#### Parameters

| Name | Type | Description |
|---|---|---|
| *interval* | int | The polling interval, specified in milliseconds. |

*Table 4-91: Method setInterval: parameters*

## 4.30 Protocol WLWifiConnectedCallback

This protocol invokes a callback for getting the connected WiFi access point.

### 4.30.1 Method execute

#### Syntax

```
- (void)execute:(WLWifiAccessPoint *)connectedAccessPoint
(int):signalStrength
```

#### Description

This method is executed when the connected WiFi access point is acquired.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *connectedAccessPoint* | `WLWifiAccessPoint` | The connected access point, including the SSID and MAC address. |
| *signalStrength* | `int` | The connected signal strength, as a percentage. |

*Table 4-92: Method execute: parameters*

## 4.31 Class WLWifiConnectTrigger

A trigger that activates when it connects for the first time to an access point that passes the policy’s filters. The trigger can reactivate only after disconnecting from or connecting to an access point that doesn’t pass the policy’s filters.

### 4.31.1 Method getCallback

#### Syntax

```
- (WLTriggerCallback)getCallback
```

#### Description

This method returns the callback object, whose execute method is called when the trigger is activated.

#### Parameters

None.

### 4.31.2 Method getConnectedAccessPoint

#### Syntax

```
- (WLWifiAccessPointFilter)getConnectedAccessPoint
```

#### Description

This method returns the callback object, as specified by the `setConnectedAccessPoint` method.

#### Parameters

None.

### 4.31.3 Method getEvent

#### Syntax

```
- (JSONObject)getEvent
```

#### Description

This method returns the event to transmit. If `null`, then it is not transmitted. The current device context is automatically added to the event when it is transmitted.

#### Parameters

None.

### 4.31.4 Method init

#### Syntax

```
- (id)init
```

#### Description

This method initializes the trigger definition.

#### Parameters

None.

### 4.31.5 Method isTransmitImmediately

#### Syntax

```
- (BOOL)isTransmitImmediately
```

#### Description

This method returns `true` if the event should be transmitted immediately.

#### Parameters

None.

### 4.31.6 Method setCallback

#### Syntax

```
-(WLWifiConnectTrigger *)setCallback:(id<WLTriggerCallback>)callback
```

#### Description

This method sets the callback, whose execute method is called when the trigger is activated.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *`callback`* | `WLTriggerCallback` | The callback the user wants to set. This parameter must conform to the `WLTriggerCallback` protocol. When the trigger is activated, its execute method will be called and the current device context is passed as a parameter. |

*Table 4-93: Method setCallback: parameters*

### 4.31.7 Method setConnectedAccessPoint

#### Syntax

```
- (WLWifiConnectTrigger
*)setConnectedAccessPoint:(WLWifiAccessPointFilter
*)connectedAccessPoint
```

#### Description

This method sets the filter that the connected WiFi access point must match in order for the trigger to activate.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *connectedAccessPoint* | WLWifiAccessPointFilter | The filter for the connected access point. |

*Table 4-94: Method setConnectedAccessPoint: parameters*

### 4.31.8 Method setEvent

#### Syntax

```
- (WLWifiConnectTrigger *)setEvent:(NSMutableDictionary *)event
```

#### Description

This method sets the event to be transmitted to the server.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *event* | NSMutableDictionary | The event the user wants to set. |

*Table 4-95: Method setEvent: parameters*

### 4.31.9 Method setTransmitImmediately

#### Syntax

```
- (WLWifiConnectTrigger
*)setTransmitImmediately:(BOOL)transmitImmediately
```

#### Description

This method determines whether the event is transmitted immediately, or whether it is transmitted according to the transmission policy. If the value is `true`, the event is added to the transmission buffer, and the contents of the transmission buffer are flushed to the server. Otherwise the event is added only to the transmission buffer.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *transmitImmediately* | Boolean | A Boolean value that determines whether the event is transmitted immediately. |

*Table 4-96: Method setTransmitImmediately: parameters*

## 4.32 Class WLWifiDisconnectTrigger

A trigger that activates when it disconnects for the first time from an access point that passes the policy's filters. The trigger can reactivate only after connecting to an access point that passes the policy's filters.

### 4.32.1 Method getCallback

#### Syntax

```
- (WLTriggerCallback)getCallback
```

#### Description

This method returns the callback object, whose execute method is called when the trigger is activated.

#### Parameters

None.

### 4.32.2 Method getConnectedAccessPoint

#### Syntax

```
- (WLWifiAccessPointFilter)getConnectedAccessPoint
```

#### Description

This method returns the callback object, as specified by the `setConnectedAccessPoint` method.

#### Parameters

None.

### 4.32.3 Method getEvent

#### Syntax

```
- (JSONObject)getEvent
```

#### Description

This method returns the event to transmit. If `null`, then it is not transmitted. The current device context is automatically added to the event when it is transmitted.

#### Parameters

None.

### 4.32.4 Method init

#### Syntax

```
- (id)init
```

#### Description

This method initializes the trigger definition.

#### Parameters

None.

### 4.32.5 Method isTransmitImmediately

#### Syntax

```
- (BOOL)isTransmitImmediately
```

#### Description

This method returns `true` if the event should be transmitted immediately.

#### Parameters

None.

### 4.32.6 Method setCallback

#### Syntax

```
-(WLWifiDisconnectTrigger
*)setCallback:(id<WLTriggerCallback>)callback
```

#### Description

This method sets the callback, whose execute method is called when the trigger is activated.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *callback* | `WLTriggerCallback` | The callback the user wants to set. This parameter must conform to the |

| Name | Type | Description |
|---|---|---|
| | | `WLTriggerCallback` protocol. When the trigger is activated, its execute method will be called and the current device context is passed as a parameter. |

Table 4-97: Method setCallback: parameters

### 4.32.7 Method setConnectedAccessPoint

#### Syntax

```
- (WLWifiDisconnectTrigger
*)setConnectedAccessPoint:(WLWifiAccessPointFilter
*)connectedAccessPoint
```

#### Description

This method sets the filter that the connected WiFi access point must match in order for the trigger to activate.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *connectedAccessPoint* | `WLWifiAccessPointFilter` | The filter for the connected access point. |

Table 4-98: Method setConnectedAccessPoint: parameters

### 4.32.8 Method setEvent

#### Syntax

```
- (WLWifiDisconnectTrigger *)setEvent:(NSMutableDictionary *)event
```

#### Description

This method sets the event for the access point.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *event* | `NSMutableDictionary` | The event the user wants to set. |

Table 4-99: Method setEvent: parameters

#### 4.32.9 Method setTransmitImmediately

##### Syntax

```
- (WLWifiDisconnectTrigger
*)setTransmitImmediately:(BOOL)transmitImmediately
```

##### Description

This method determines whether the event is transmitted immediately, or whether it is transmitted according to the transmission policy. If the value is `true`, the event is added to the transmission buffer, and the contents of the transmission buffer are flushed to the server. Otherwise the event is added only to the transmission buffer.

##### Parameters

| Name | Type | Description |
|---|---|---|
| **`transmitImmediately`** | `Boolean` | A Boolean value that determines whether the event is transmitted immediately. |

*Table 4-100: Method setTransmitImmediately: parameters*

## 4.33 Class WLWifiDwellInsideTrigger

A trigger definition that is activated when the device remains inside an area for a specified period of time. In order to reactivate the trigger, the device must first leave the area, and then return to the area. The area is defined by the visibility of a set of given access points.

The setters of this class return a reference to this object so as to enable chaining calls.

### 4.33.1 Method getAreaAccessPoints

#### Syntax

```
- (WLWifiAccessPointFilter)getAreaAccessPoints
```

#### Description

This method returns the area for the trigger, as specified by the `setAreaAccessPoints` method.

#### Parameters

None.

### 4.33.2 Method getCallback

#### Syntax

```
- (WLTriggerCallback)getCallback
```

#### Description

This method returns the callback object, whose execute method is called when the trigger is activated.

#### Parameters

None.

### 4.33.3 Method getConfidenceLevel

#### Syntax

```
- (WLConfidenceLevel)getConfidenceLevel
```

#### Description

This method returns the confidence level. This indicates how a position's accuracy is taken into account.

#### Parameters

None.

### 4.33.4 Method getDwellingTime

#### Syntax

```
- (long long)getDwellingTime
```

#### Description

This method returns the minimum time the device needs to be inside the area before the trigger is activated.

#### Parameters

None.

### 4.33.5 Method getEvent

#### Syntax

```
- (JSONObject)getEvent
```

#### Description

This method returns the event to transmit. If `null`, then it is not transmitted. The current device context is automatically added to the event when it is transmitted.

#### Parameters

None.

### 4.33.6 Method init

#### Syntax

```
- (id)init
```

#### Description

This method initializes the trigger definition.

#### Parameters

None.

### 4.33.7 Method isTransmitImmediately

#### Syntax

```
- (BOOL)isTransmitImmediately
```

#### Description

This method returns `true` if the event should be transmitted immediately.

#### Parameters

None.

### 4.33.8 Method setAreaAccessPoints

#### Syntax

```
- (WLWifiDwellInsideTrigger *)setAreaAccessPoints:(NSMutableArray
*)areaFilters
```

#### Description

This method defines which access points are considered by the trigger. Wildcards are not permitted.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *areaFilters* | NSMutableArray | The access points the user wants to set for the WiFi location. |

*Table 4-101: Method setAreaAccessPoints: parameters*

### 4.33.9 Method setCallback

#### Syntax

```
-(WLWifiDwellInsideTrigger
*)setCallback:(id<WLTriggerCallback>)callback
```

#### Description

This method sets the callback, whose execute method is called when the trigger is activated.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *callback* | WLTriggerCallback | The callback the user wants to set. This parameter must conform to the `WLTriggerCallback` protocol. When the trigger is activated, its execute method will be called and the current device context is passed as a parameter. |

*Table 4-102: Method setCallback: parameters*

### 4.33.10 Method setConfidenceLevel

#### Syntax

```
- (WLWifiDwellInsideTrigger
*)setConfidenceLevel:(WLConfidenceLevel)confidenceLevel
```

#### Description

This method sets the confidence level. Only access points whose signal strength meets the confidence level are considered visible.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *confidenceLevel* | ConfidenceLevel | The minimum signal strength necessary for an access point. |

*Table 4-103: Method setConfidenceLevel: parameters*

### 4.33.11 Method setDwellingTime

#### Syntax

```
- (WLWifiDwellInsideTrigger *)setDwellingTime:(long long)dwellingTime
```

#### Description

This method sets the dwelling time. The method defines how long the device must be inside the area before the trigger is activated.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *dwellingTime* | long | The dwelling time in milliseconds. |

*Table 4-104: Method setDwellingTime: parameters*

### 4.33.12 Method setEvent

#### Syntax

```
- (WLWifiDwellInsideTrigger *)setEvent:(NSMutableDictionary *)event
```

#### Description

This method sets the event to be transmitted to the server.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *event* | NSMutableDictionary | The event the user wants to set. |

*Table 4-105: Method setEvent: parameters*

### 4.33.13 Method setTransmitImmediately

#### Syntax

```
- (WLWifiDwellInsideTrigger
*)setTransmitImmediately:(BOOL)transmitImmediately
```

#### Description

This method determines whether the event is transmitted immediately, or whether it is transmitted according to the transmission policy. If the value is `true`, the event is added to the transmission buffer, and the contents of the transmission buffer are flushed to the server. Otherwise the event is added only to the transmission buffer.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *transmitImmediately* | `Boolean` | A Boolean value that determines whether the event is transmitted immediately. |

*Table 4-106: Method setTransmitImmediately: parameters*

## 4.34 Class WLWifiDwellOutsideTrigger

A trigger definition that is activated when the device remains outside an area for a specified period of time. In order to reactivate the trigger, the device must first enter the area, and then leave the area. The area is defined by the visibility of a set of given access points.

The setters of this class return a reference to this object so as to enable chaining calls.

Note that confidence levels are not supported for `WLWifiDwellOutsideTrigger`. If you attempt to use a `setConfidenceLevel` method, an `UnsupportedOperationException` is thrown.

### 4.34.1 Method getAreaAccessPoints

#### Syntax

```
- (WLWifiAccessPointFilter)getAreaAccessPoints
```

#### Description

This method returns the area for the trigger, as specified by the `setAreaAccessPoints` method.

#### Parameters

None.

### 4.34.2 Method getCallback

#### Syntax

```
- (WLTriggerCallback)getCallback
```

#### Description

This method returns the callback object, whose execute method is called when the trigger is activated.

#### Parameters

None.

### 4.34.3 Method getDwellingTime

#### Syntax

```
- (long long)getDwellingTime
```

#### Description

This method returns the minimum time the device needs to be outside the area before the trigger is activated.

#### Parameters

None.

### 4.34.4 Method getEvent

#### Syntax

```
- (JSONObject)getEvent
```

#### Description

This method returns the event to transmit. If `null`, then it is not transmitted. The current device context is automatically added to the event when it is transmitted.

#### Parameters

None.

### 4.34.5 Method init

#### Syntax

```
- (id)init
```

#### Description

This method initializes the trigger definition.

#### Parameters

None.

### 4.34.6 Method isTransmitImmediately

#### Syntax

```
- (BOOL)isTransmitImmediately
```

#### Description

This method returns `true` if the event should be transmitted immediately.

#### Parameters

None.

### 4.34.7 Method setAreaAccessPoints

#### Syntax

```
-(WLWifiDwellOutsideTrigger *)setAreaAccessPoints:(NSMutableArray
*)areaFilters
```

#### Description

This method defines which access points are considered by the trigger. Wildcards are not permitted.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *areaFilters* | `NSMutableArray` | The access points the user wants to set for the WiFi location. |

*Table 4-107: Method setAreaAccessPoints: parameters*

### 4.34.8 Method setCallback

#### Syntax

```
-(WLWifiDwellOutsideTrigger
*)setCallback:(id<WLTriggerCallback>)callback
```

#### Description

This method sets the callback, whose execute method is called when the trigger is activated.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *callback* | `WLTriggerCallback` | The callback the user wants to set. This parameter must conform to the `WLTriggerCallback` protocol. When the trigger is activated, its execute method will be called and the current device context is passed as a parameter. |

*Table 4-108: Method setCallback: parameters*

### 4.34.9 Method setDwellingTime

#### Syntax

```
- (WLWifiDwellOutsideTrigger *)setDwellingTime:(long
long)dwellingTime
```

#### Description

This method sets the dwelling time. The method defines how long the device must be inside the area before the trigger is activated.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *dwellingTime* | `long` | The dwelling time in milliseconds. |

*Table 4-109: Method setDwellingTime: parameters*

### 4.34.10 Method setEvent

#### Syntax

```
- (WLWifiDwellOutsideTrigger *)setEvent:(NSMutableDictionary *)event
```

#### Description

This method sets the event to be transmitted to the server.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *event* | NSMutableDictionary | The event the user wants to set. |

Table 4-110: Method setEvent: parameters

### 4.34.11 Method setTransmitImmediately

#### Syntax

```
- (WLWifiDwellOutsideTrigger
*)setTransmitImmediately:(BOOL)transmitImmediately
```

#### Description

This method determines whether the event is transmitted immediately, or whether it is transmitted according to the transmission policy. If the value is `true`, the event is added to the transmission buffer, and the contents of the transmission buffer are flushed to the server. Otherwise the event is added only to the transmission buffer.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *transmitImmediately* | Boolean | A Boolean value that determines whether the event is transmitted immediately. |

Table 4-111: Method setTransmitImmediately: parameters

## 4.35 Class WLWifiEnterTrigger

A trigger definition that is activated when a device enters an area. To activate the trigger, the device must first have been outside the area, and then enter the area at the given confidence level.

The setters of this class return a reference to this object so as to enable chaining calls.

### 4.35.1 Method getAreaAccessPoints

#### Syntax

```
- (WLWifiAccessPointFilter)getAreaAccessPoints
```

#### Description

This method returns the area for the trigger, as specified by the `setAreaAccessPoints` method.

#### Parameters

None.

### 4.35.2 Method getCallback

#### Syntax

```
- (WLTriggerCallback)getCallback
```

#### Description

This method returns the callback object, whose execute method is called when the trigger is activated.

#### Parameters

None.

### 4.35.3 Method getConfidenceLevel

#### Syntax

```
- (WLConfidenceLevel)getConfidenceLevel
```

#### Description

This method returns the confidence level. This indicates how a position's accuracy is taken into account.

#### Parameters

None.

### 4.35.4 Method getEvent

#### Syntax

```
- (JSONObject)getEvent
```

#### Description

This method returns the event to transmit. If `null`, then it is not transmitted. The current device context is automatically added to the event when it is transmitted.

#### Parameters

None.

### 4.35.5 Method init

#### Syntax

```
- (id)init
```

#### Description

This method initializes the trigger definition.

#### Parameters

None.

### 4.35.6 Method isTransmitImmediately

#### Syntax

```
- (BOOL)isTransmitImmediately
```

#### Description

This method returns `true` if the event should be transmitted immediately.

#### Parameters

None.

### 4.35.7 Method setAreaAccessPoints

#### Syntax

```
- (WLWifiEnterTrigger *)setAreaAccessPoints:(NSMutableArray
*)areaFilters
```

#### Description

This method defines which access points are considered by the trigger. Wildcards are not permitted.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *areaFilters* | `NSMutableArray` | The area access points the user wants to set. |

*Table 4-112: Method setAreaAccessPoints: parameters*

### 4.35.8 Method setCallback

#### Syntax

```
-(WLWifiEnterTrigger *)setCallback:(id<WLTriggerCallback>)callback
```

#### Description

This method sets the callback, whose execute method is called when the trigger is activated.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *callback* | `WLTriggerCallback` | The callback the user wants to set. This parameter must conform to the `WLTriggerCallback` protocol. When the trigger is activated, its execute method will be called and the current device context is passed as a parameter. |

*Table 4-113: Method setCallback: parameters*

### 4.35.9 Method setConfidenceLevel

#### Syntax

```
- (WLWifiEnterTrigger
*)setConfidenceLevel:(ConfidenceLevel)confidenceLevel
```

#### Description

This method sets the confidence level. Only access points whose signal strength meets the confidence level are considered visible.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *confidenceLevel* | `ConfidenceLevel` | The minimum signal strength necessary for an access point. |

*Table 4-114: Method setConfidenceLevel: parameters*

### 4.35.10 Method setEvent

#### Syntax

```
- (WLWifiEnterTrigger *)setEvent:(NSMutableDictionary *)event
```

#### Description

This method sets the event to be transmitted to the server.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *event* | `NSMutableDictionary` | The event the user wants to set. |

*Table 4-115: Method setEvent: parameters*

### 4.35.11 Method setTransmitImmediately

#### Syntax

```
- (WLWifiEnterTrigger
*)setTransmitImmediately:(BOOL)transmitImmediately
```

#### Description

This method determines whether the event is transmitted immediately, or whether it is transmitted according to the transmission policy. If the value is `true`, the event is added to the transmission buffer, and the contents of the transmission buffer are flushed to the server. Otherwise the event is added only to the transmission buffer.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *transmitImmediately* | `Boolean` | A Boolean value that determines whether the event is transmitted immediately. |

*Table 4-116: Method setTransmitImmediately: parameters*

## 4.36 Class WLWifiError

A WLWifiError object is created when an error is encountered during acquisition of a WiFi access point.

### 4.36.1 Method getErrorCode

#### Syntax

```
- (WLWifiErrorCodes)getErrorCode
```

#### Description

This method returns the error code.

#### Parameters

None.

### 4.36.2 Method getMessage

#### Syntax

```
- (NSString *)getMessage
```

#### Description

This method returns the message for the error.

#### Parameters

None.

### 4.36.3 Method initWithErrorCode:message

#### Syntax

```
- (id)initWithErrorCode:(WLWifiErrorCodes)code message:(NSString
*)message
```

#### Description

This method returns the error code and the associated message.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *code* | WLWifiErrorCodes | The error code. |
| *message* | NSString | The message for the error. |

*Table 4-117: Method initWithErrorCode:message: parameters*

## 4.37 Class WLWifiExitTrigger

A trigger definition that is activated when a device leaves an area. To activate the trigger, the device must first have been inside the area, and then leave the area at the given confidence level.

The setters of this class return a reference to this object so as to enable chaining calls.

### 4.37.1 Method getAreaAccessPoints

#### Syntax

```
- (WLWifiAccessPointFilter)getAreaAccessPoints
```

#### Description

This method returns the area for the trigger, as specified by the `setAreaAccessPoints` method.

#### Parameters

None.

### 4.37.2 Method getCallback

#### Syntax

```
- (WLTriggerCallback)getCallback
```

#### Description

This method returns the callback object, whose execute method is called when the trigger is activated.

#### Parameters

None.

### 4.37.3 Method getConfidenceLevel

#### Syntax

```
- (WLConfidenceLevel)getConfidenceLevel
```

#### Description

This method returns the confidence level. This indicates how a position's accuracy is taken into account.

#### Parameters

None.

### 4.37.4 Method getEvent

#### Syntax

```
- (JSONObject)getEvent
```

#### Description

This method returns the event to transmit. If `null`, then it is not transmitted. The current device context is automatically added to the event when it is transmitted.

#### Parameters

None.

### 4.37.5 Method init

#### Syntax

```
- (id)init
```

#### Description

This method initializes the trigger definition.

#### Parameters

None.

### 4.37.6 Method isTransmitImmediately

#### Syntax

```
- (BOOL)isTransmitImmediately
```

#### Description

This method returns `true` if the event should be transmitted immediately.

#### Parameters

None.

### 4.37.7 Method setAreaAccessPoints

#### Syntax

```
-(WLWifiExitTrigger *)setAreaAccessPoints:(NSMutableArray
*)areaFilters
```

#### Description

This method defines which access points are considered by the trigger. Wildcards are not permitted.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *areaFilters* | `NSMutableArray` | The area access points the user wants to set. |

*Table 4-118: Method setAreaAccessPoints: parameters*

### 4.37.8 Method setCallback

#### Syntax

```
-(WLWifiExitTrigger *)setCallback:(id<WLTriggerCallback>)callback
```

#### Description

This method sets the callback, whose execute method is called when the trigger is activated.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *callback* | `WLTriggerCallback` | The callback the user wants to set. This parameter must conform to the `WLTriggerCallback` protocol. When the trigger is activated, its execute method will be called and the current device context is passed as a parameter. |

*Table 4-119: Method setCallback: parameters*

### 4.37.9 Method setConfidenceLevel

#### Syntax

```
- (WLWifiExitTrigger
*)setConfidenceLevel:(ConfidenceLevel)confidenceLevel
```

#### Description

This method sets the confidence level. Only access points whose signal strength meets the confidence level are considered visible.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *`confidenceLevel`* | `ConfidenceLevel` | The confidence level the user wants to set. |

*Table 4-120: Method setConfidenceLevel: parameters*

### 4.37.10 Method setEvent

#### Syntax

```
- (WLWifiExitTrigger *)setEvent:(NSMutableDictionary *)event
```

#### Description

This method sets the event to be transmitted to the server.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *`event`* | `NSMutableDictionary` | The event the user wants to set. |

*Table 4-121: Method setEvent: parameters*

### 4.37.11 Method setTransmitImmediately

#### Syntax

```
- (WLWifiExitTrigger
*)setTransmitImmediately:(BOOL)transmitImmediately
```

#### Description

This method determines whether the event is transmitted immediately, or whether it is transmitted according to the transmission policy. If the value is `true`, the event is added to the transmission buffer, and the contents of the transmission buffer are flushed to the server. Otherwise the event is added only to the transmission buffer.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *transmitImmediately* | Boolean | A Boolean value that determines whether the event is transmitted immediately. |

*Table 4-122: Method setTransmitImmediately: parameters*

## 4.38 Protocol WLWifiFailureCallback

This protocol is invoked when an error occurs in a WiFi acquisition.

### 4.38.1 execute

#### Syntax

```
- (void)execute:WLWifiError *errorObject
```

#### Description

This method is executed when an error occurs during acquisition.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *errorObject* | WLWifiError | The error that occurred. |

*Table 4-123: Method execute: parameters*

## 4.39 Class WLWifiLocation

This class contains a WiFi location, as determined by the visible access points and connected access point, filtered by a policy.

### 4.39.1 Method getConnectedAccessPoint

#### Syntax

```
- (WLWifiAccessPoint *)getConnectedAccessPoint
```

#### Description

This method returns information about the connected access point if it passes the policy filters. If the access point does not pass the policy filters, `null` is returned.

#### Parameters

None.

### 4.39.2 Method getConnectedSignalStrength

#### Syntax

```
- (NSNumber *)getConnectedSignalStrength
```

#### Description

This method returns the signal strength for the connected access point, as a percentage. `Null` is returned if `getConnectedAccessPoint` is `null`.

#### Parameters

None.

### 4.39.3 Method init

#### Syntax

```
- (id)init
```

#### Description

This method initializes the WiFi location.

#### Parameters

None.

### 4.39.4 Method initWithAccessPoints:connectedAccessPoint:connectedSignalStrength:acquisitionTime

#### Syntax

```
- (id)initWithAccessPoints:(NSMutableArray *)accessPoints
connectedAccessPoint:(WLWifiAccessPoint *)connectedAccessPoint
connectedSignalStrength:(NSNumber *)connectedSignalStrength
acquisitionTime:(long long)acquisitionTime
```

#### Description

This method creates a new WiFi location.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *accessPoints* | NSMutableArray | The acquired access points, which have passed the filter of the policy. |
| *connectedAccessPoint* | WLWifiAccessPoint | The connected access point. |
| *connectedSignalStrength* | NSNumber | The signal strength, as a percentage. |
| *acquisitionTime* | long | The time of the acquisition. |

*Table 4-124: Method initWithAccessPoints: parameters*

## 4.40 Class WLWifiTrigger

An abstract base class for WiFi triggers. This is the parent interface for all the other WiFi triggers.

## 4.41 Class WLWifiVisibleAccessPointsChangeTrigger

A trigger for tracking changes to the visible access points.

The setters of this class return a reference to this object so as to enable chaining calls.

### 4.41.1 Method getCallback

#### Syntax

```
- (WLTriggerCallback)getCallback
```

#### Description

This method returns the callback object, whose execute method is called when the trigger is activated.

#### Parameters

None.

### 4.41.2 Method getEvent

#### Syntax

```
- (JSONObject)getEvent
```

#### Description

This method returns the event to transmit. If `null`, then it is not transmitted. The current device context is automatically added to the event when it is transmitted.

#### Parameters

None.

### 4.41.3 Method init

#### Syntax

```
- (id)init
```

#### Description

This method initializes the trigger definition.

#### Parameters

None.

### 4.41.4 Method isTransmitImmediately

#### Syntax

```
- (BOOL)isTransmitImmediately
```

#### Description

This method returns `true` if the event should be transmitted immediately.

#### Parameters

None.

### 4.41.5 Method setCallback

#### Syntax

```
-(WLWifiVisibleAccessPointsChangeTrigger *)
setCallback:(id<WLTriggerCallback>)callback
```

#### Description

This method sets the callback, whose execute method is called when the trigger is activated.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *callback* | `WLTriggerCallback` | The callback the user wants to set. This parameter must conform to the `WLTriggerCallback` protocol. When the trigger is activated, its execute method will be called and the current device context is passed as a parameter. |

*Table 4-125: Method setCallback: parameters*

### 4.41.6 Method setEvent

#### Syntax

```
- (WLWifiVisibleAccessPointsChangeTrigger *)
setEvent:(NSMutableDictionary *)event
```

#### Description

This method sets the event to be transmitted to the server.

#### Parameters

| Name | Type | Description |
|---|---|---|
| *event* | `NSMutableDictionary` | The event the user wants to set. |

*Table 4-126: Method setEvent: parameters*

### 4.41.7 Method setTransmitImmediately

#### Syntax

```
- (WLWifiVisibleAccessPointsChangeTrigger *)
setTransmitImmediately:(BOOL)transmitImmediately
```

#### Description

This method determines whether the event is transmitted immediately, or whether it is transmitted according to the transmission policy. If the value is `true`, the event is added to the transmission buffer, and the contents of the transmission buffer are flushed to the server. Otherwise the event is added only to the transmission buffer.

Parameters

| Name | Type | Description |
|---|---|---|
| *transmitImmediately* | `Boolean` | A Boolean value that determines whether the event is transmitted immediately. |

*Table 4-127: Method setTransmitImmediately: parameters*

### 4.41.8 Method validate

Syntax

```
-(BOOL)validate:(WLWifiAcquisitionPolicy *)policy
```

Description

This method checks if the trigger can be evaluated to `true` under a policy. The value `true` is returned only if there is a WiFi location that could be matched by the policy.

Parameters

| Name | Type | Description |
|---|---|---|
| *policy* | `WLWifiAcquistionPolicy` | The policy to check. |

*Table 4-128: Method validate: parameters*

# Appendix A – Notices

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 at http://www.ibm.com/privacy and IBM's Online Privacy Statement at http://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” at http://www.ibm.com/software/info/product-privacy.

# Appendix B - Support and comments

For the entire IBM Worklight documentation set, training material and online forums where you can post questions, see the IBM website at:

http://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 at:

http://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 at:

http://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.

- Submit your comments in the IBM Worklight forums at:
- https://www.ibm.com/developerworks/mobile/worklight/connect.html

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

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