ExtremeWireless™

ExtremeLocation iOS SDK Reference Guide

9036426-00 Rev AA

Published August 2019

Legal Notice

Extreme Networks, Inc. reserves the right to make changes in specifications and other information contained in this document and its website without prior notice. The reader should in all cases consult representatives of Extreme Networks to determine whether any such changes have been made.

The hardware, firmware, software or any specifications described or referred to in this document are subject to change without notice.

Trademarks

Extreme Networks and the Extreme Networks logo are trademarks or registered trademarks of Extreme Networks, Inc. in the United States and/or other countries.

All other names (including any product names) mentioned in this document are the property of their respective owners and may be trademarks or registered trademarks of their respective companies/owners.

For additional information on Extreme Networks trademarks, please see:
www.extremenetworks.com/company/legal/trademarks

Open Source Declarations

Some software files have been licensed under certain open source or third-party licenses. End-user license agreements and open source declarations can be found at:
www.extremenetworks.com/support/policies/software-licensing

Table of Contents

Preface

This section discusses the conventions used in this guide, ways to provide feedback, additional help, and other Extreme Networks® publications.

Conventions

This section discusses the conventions used in this guide.

Text Conventions

The following tables list text conventions that are used throughout this guide.

Table 1: Notice Icons

Icon	Notice Type	Alerts you to...
	General Notice	Helpful tips and notices for using the product.
	Note	Important features or instructions.
	Caution	Risk of personal injury, system damage, or loss of data.
	Warning	Risk of severe personal injury.
New!	New Content	Displayed next to new content. This is searchable text within the PDF.

Table 2: Text Conventions

Convention	Description
`Screen displays`	This typeface indicates command syntax, or represents information as it appears on the screen.
The words **enter** and **type**	When you see the word "enter" in this guide, you must type something, and then press the Return or Enter key. Do not press the Return or Enter key when an instruction simply says "type."
[Key] names	Key names are written with brackets, such as **[Return]** or **[Esc]**. If you must press two or more keys simultaneously, the key names are linked with a plus sign (+). Example: Press **[Ctrl]**+**[Alt]**+**[Del]**
Words in italicized type	Italics emphasize a point or denote new terms at the place where they are defined in the text. Italics are also used when referring to publication titles.

Terminology

When features, functionality, or operation is specific to a switch family, such as ExtremeSwitching, the family name is used. Explanations about features and operations that are the same across all product families simply refer to the product as the switch.

Providing Feedback to Us

Quality is our first concern at Extreme Networks, and we have made every effort to ensure the accuracy and completeness of this document. We are always striving to improve our documentation and help you work better, so we want to hear from you! We welcome all feedback but especially want to know about:

- Content errors or confusing or conflicting information.
- Ideas for improvements to our documentation so you can find the information you need faster.
- Broken links or usability issues.

If you would like to provide feedback to the Extreme Networks Information Development team, you can do so in two ways:

- Use our short online feedback form at https://www.extremenetworks.com/documentation-feedback/.
- Email us at documentation@extremenetworks.com.

Please provide the publication title, part number, and as much detail as possible, including the topic heading and page number if applicable, as well as your suggestions for improvement.

Getting Help

If you require assistance, contact Extreme Networks using one of the following methods:

Extreme Portal	Search the GTAC (Global Technical Assistance Center) knowledge base, manage support cases and service contracts, download software, and obtain product licensing, training, and certifications.
The Hub	A forum for Extreme Networks customers to connect with one another, answer questions, and share ideas and feedback. This community is monitored by Extreme Networks employees, but is not intended to replace specific guidance from GTAC.
Call GTAC	For immediate support: 1-800-998-2408 (toll-free in U.S. and Canada) or +1 408-579-2826. For the support phone number in your country, visit: www.extremenetworks.com/support/contact

Before contacting Extreme Networks for technical support, have the following information ready:

- Your Extreme Networks service contract number and/or serial numbers for all involved Extreme Networks products
- A description of the failure
- A description of any action(s) already taken to resolve the problem
- A description of your network environment (such as layout, cable type, other relevant environmental information)
- Network load at the time of trouble (if known)

- The device history (for example, if you have returned the device before, or if this is a recurring problem)
- Any related RMA (Return Material Authorization) numbers

Subscribing to Service Notifications

You can subscribe to email notifications for product and software release announcements, Vulnerability Notices, and Service Notifications.

1 Go to www.extremenetworks.com/support/service-notification-form.
2 Complete the form with your information (all fields are required).
3 Select the products for which you would like to receive notifications.

Note
You can modify your product selections or unsubscribe at any time.

4 Click **Submit**.

Documentation and Training

To find Extreme Networks product guides, visit our documentation pages at:

Current Product Documentation	www.extremenetworks.com/documentation/
Archived Documentation (for earlier versions and legacy products)	www.extremenetworks.com/support/documentation-archives/
Release Notes	www.extremenetworks.com/support/release-notes
Hardware/Software Compatibility Matrices	https://www.extremenetworks.com/support/compatibility-matrices/
White papers, data sheets, case studies, and other product resources	https://www.extremenetworks.com/resources/

Training

Extreme Networks offers product training courses, both online and in person, as well as specialized certifications. For more information, visit www.extremenetworks.com/education/.

1 ExtremeLocation iOS SDK

Sample Apps
Configuring Your XCode Project
Using the Core Framework

The ExtremeLocation iOS SDK includes everything you need to write apps that integrate with the ExtremeLocation platform.

Included in the SDK are:

- The core framework, `FTMCore.framework`
- Two sample apps
- Documentation

Apps that integrate with the core framework can be written in Swift or Objective-C. An app's deployment target must be iOS 9.0 or later.

Sample Apps

The easiest way to get started is by looking at the sample apps.

- SampleAppSwift - Demonstrates a basic integration with the core framework.
- SampleAppObjC - Equivalent to SampleAppSwift, but written in Objective-C.

To add the core framework as an embedded binary and run the sample app:

1 Open a sample app project in Xcode.
2 In the tree on the left, select the root node (e.g., SampleAppSwift).
3 In the content panel on the right, select the target.
4 Select General.
5 Scroll to the Embedded Binaries section and click **+**.
6 Select **Add Other...**.
7 Open `FTMCore.framework`, located in the root folder of the SDK.
8 Check **Copy items if needed**, and select **Create groups**.

The sample apps only work properly when run on a device because Bluetooth functionality is not available in the simulator. Modify the project as needed to add a team or to configure signing otherwise.

To run the sample apps, you also need to have a valid app key and secret. These values are passed into the `FTMSession` class. Please visit https://manage.extremelocation.com to request an app key and secret.

Configuring Your XCode Project

To integrate the core framework into your own app:

1 Embed the FTMCore.framework in your app.

2 Configure `Info.plist` and background modes.

Note

Your app's `Info.plist` must include these four usage descriptions:

- NSBluetoothPeripheralUsageDescription - The core framework must be able to interact with Bluetooth peripherals in order to function properly.
- NSLocationAlwaysUsageDescription - The **always** mode of location services is required, as opposed to **when in use**. Otherwise your app would not be notified when coming into range of beacons.
- NSLocationAlwaysAndWhenInUseUsageDescription - iOS 11 requires this key.
- NSLocationWhenInUseUsageDescription - iOS 11 requires this key.

Choose descriptions that are appropriate for your app.

If you intend to monitor beacons broadcasting in **payment** mode (rather than **private** mode), the **bluetooth-central** background mode must be enabled. This can be done in one of two ways:

- In Xcode, select the target, then select **Capabilities** > **Turn on Background Modes**, and check the box for **Uses Bluetooth LE accessories**.
- In your `Info.plist` file, add `UIBackgroundModes` with an entry for **bluetooth-central**.

3 Add a run script to your project, and then copy the following script:

```
#
# This script removes unused architectures from Footmarks frameworks, which are
# fat binaries that make it possible to develop apps in both the simulator and
# on device. Apple does not allow app store submissions to contain simulator
# architectures.
#
# In Xcode, select the target, then Build Phases, and make sure this script
# comes after the "Embed Frameworks" step.
#
# Based on:
# http://ikennd.ac/blog/2015/02/stripping-unwanted-architectures-from-dynamic-
libraries-in-xcode/
#

APP_PATH="${TARGET_BUILD_DIR}/${WRAPPER_NAME}"
find "$APP_PATH"-name'FTM*.framework'-type d | while read-r FRAMEWORK
do
    FRAMEWORK_EXECUTABLE_NAME=$(defaults read"$FRAMEWORK/Info.plist"
CFBundleExecutable)
    FRAMEWORK_EXECUTABLE_PATH="$FRAMEWORK/$FRAMEWORK_EXECUTABLE_NAME"
    echo"Executable is $FRAMEWORK_EXECUTABLE_PATH"
    EXTRACTED_ARCHS=()

    for ARCH in$ARCHS
    do
        echo"Extracting $ARCH from $FRAMEWORK_EXECUTABLE_NAME"
        lipo -extract"$ARCH""$FRAMEWORK_EXECUTABLE_PATH"-o"$FRAMEWORK_EXECUTABLE_PATH-
$ARCH"
        EXTRACTED_ARCHS+=("$FRAMEWORK_EXECUTABLE_PATH-$ARCH")
    done
```

```
    echo"Merging extracted architectures: ${ARCHS}"
    lipo -o"$FRAMEWORK_EXECUTABLE_PATH-merged"-create"${EXTRACTED_ARCHS[@]}"
    rm"${EXTRACTED_ARCHS[@]}"

    echo"Replacing original executable with thinned version"
    rm"$FRAMEWORK_EXECUTABLE_PATH"
    mv"$FRAMEWORK_EXECUTABLE_PATH-merged""$FRAMEWORK_EXECUTABLE_PATH"
done
```

4 Get your app key and secret from https://manage.extremelocation.com to use the core framework.

Using the Core Framework

Signing In

When integrating with the core framework, the first task that must be done is establishing a session with the ExtremeLocation Cloud.

```
FTMSession.signIn(appKey:"…key…",appSecret:"…secret…",username:nil){
    (error:Error?)in
}
```

The call to `signIn` can happen anywhere in the application flow, but it should be called before any other core framework API. For background processing to work properly, such as when iOS launches the app due to coming into range of a beacon, the `signIn` call needs to happen early on, such as in `application(_:didFinishLaunchingWithOptions:)`. To obtain an app key and secret, please seehttps://manage.extremelocation.com.

Receiving Experiences

Implement `FTMExperienceManagerDelegate` to receive experiences from the ExtremeLocation Cloud. For details, see

```
FTMExperienceManagerDelegate.experienceManagerDidReceiveExperiences()
FTMExperienceManager.delegate=selffuncexperienceManagerDidReceiveExperiences(_experiences:
Set<FTMExperience>)
    {// Process the experiences.}
```

The Beacon Manager

Some apps might only be interested in processing experiences. Other apps might also want access to the beacons in the surrounding environment. The `FTMBeaconManager` class maintains a list of in-range beacons and communicates state changes with its delegate.

Any time the beacons in the surrounding environment change, `FTMBeaconManagerDelegate.beaconManagerDidRangeBeacons()` is called. To access the latest beacons, call `FTMBeaconManager.beacons`.

Finding the Nearest Beacon

The `FTMNearestBeaconRequest` class allows a request to be made for the beacon that is nearest to the current device. The beacons in the surrounding environment are analyzed, and the ExtremeLocation Cloud determines which beacon is the nearest.

2 Handling Beacon and Experience Responses

FMBeaconManager Delegate Methods
FMExperienceManager Delegate Methods

This section allows you to integrate with the ExtremeLocation platform via the mobile SDK.

FMBeaconManager Delegate Methods

Below is a list of beacon delegate methods defined in `FMBeaconManagerDelegate`:

- didRangeBeacons

  Returns all ExtremeLocation beacons in range of the user.

```
- (void)beaconManager:(FMBeaconManager *)manager
      didRangeBeacons:(NSArray *)beacons             inRegion:(FMBeaconRegion *)region;
```

- didEnterRegion

  Invoked when a user walks within range of a new ExtremeLocation beacon.

Note
This is one of two methods that get invoked even when a user does not have your app running and comes within range of an ExtremeLocation beacon. Thus, it is extremely powerful. If you push a notification and the user clicks it, then your app starts running. If they do not click on the notification, you get approximately five seconds of processing time before your app goes back to not running again.

```
-(void)beaconManager:(FMBeaconManager *)manager
      didEnterRegion:(FMBeaconRegion *)region;
```

- didExitRegion

  Invoked when a user walks out of an ExtremeLocation beacon's broadcast range. Usually there is about a 30 second delay to determine that the user is completely out of range and not hovering on the beacon's boundary.

Note
This is the second method that gets triggered even if your app is not running.

```
-(void)beaconManager:(FMBeaconManager *)manager
       didExitRegion:(FMBeaconRegion *)region;
```

- bluetoothDidSwitchState

  When a user's Bluetooth is disabled, this method will get invoked. This is a good place to present an alert to request that the user enables Bluetooth.

```
- (void) bluetoothDidSwitchState:(CBCentralManagerState)state;
```

- locationServicesFailedWithError

  When a user's location services are disabled entirely or disabled for your app, this method will get invoked.

```
- (void) locationServicesFailedWithError: (NSError *)error;
```

FMExperienceManager Delegate Methods

This section describes the method that gets invoked when the ExtremeLocation Cloud service returns an experience in response to some user action (e.g. entering a beacon's proximity).

didCompleteExperiences

Returns one or more Experience objects. Typically, one Experience will be returned. Video, Image & Alert Experiences will be returned as instances of their respective classes. In turn, you will not have to parse them. Custom experiences defined in our Cloud Service, will be returned as a FMCustomExp object, which will contain the raw JSON data. The various Experience objects and their respective fields can be found in the "Footmarks_SDK.h" header file.

```
- (void) didCompleteExperiences: (NSArray*) experiences;
```

3 Experience Properties

All experiences inherit from the base class, Experience. Currently, there are four experience types: Video, Image, Alert and Custom. Experiences are created in the ExtremeLocation UI. The properties for the Experience base class and each subclass are described below:

FMExperience (Objective-C)

```
@property FMExperienceType type: Enum that indicates the given Experience type
typedef enum : int
{
    FMExperienceTypeCustom = 0,
    FMExperienceTypeVideo = 1,
    FMExperienceTypeImage = 2,
    FMExperienceTypeAlert = 3
} FMExperienceType;

@property FMExperienceAction action: Enum that instructs how to act on the given
experience.
typedef enum : int
{
    // Do not show the Experience content to the user. Depending on
    // your implementation, it may make sense to store the Experience
    // and display it sometime in the future
    FMExperienceActionDoNothing = 0,
    // Display the data contained in the Experience right away without
    // giving the user a choice. For example, if a VideoExp arrived,
    // it would auto play the next time the user opened the app.
    FMExperienceActionAutoShow = 1,
    // Ask the user if they would like to see the Experience content
    // prior to presenting it.
    FMExperienceActionAskPermission = 3
} FMExperienceAction;

typedef enum : int
{
    // The trigger is a beacon type. The user has detected a beacon.
    FMTriggerTypeBeacon = 0,
    // The trigger is a geo fence. The user entered
    // or exited a designated geo location.
    FMTriggerTypeGeozone = 1,
    // The trigger is unknown.
.
    FMTriggerTypeUnknown = 2
} FMTriggerType;

@property NSString* customDetails: Additional details added to the experience that did
not fall into any of the standard properties.

@property NSString* expTitle: The Experiences title. Typically Experiences will be
displayed with a Title and Description

@property NSString* expDescription: The Experiences description. This field contains the
message that you are trying to get across to your customers.

@property NSString* notificationTitle: This field should contain the text you would like
to display in a Notification to the user.
```

```
@property NSString* notificationDescription: The notificationDescription is an additional
field that can be used to better describe what you are notifying the user about.
@property BOOL showNotif: Indicates whether or not to display a notification to the user
for the given Experience.

@property NSDictionary* content: Contains various values that differ across each
Experience type. The expected values for each Experience type are described below.

@property NSString *triggerIdentifier: The identifier for the trigger source.

@property FMTriggerType triggerType: The type of trigger that fired the experience.
```

FMVideoExp (Objective-C)

FMVideoExp contains video content that should be made available to the user. This class also provides accessory methods to allow you to easily play the video if you would like.

```
@property FMDisplayType displayType: Enum that indicates what mode to display the video
in.
typedef enum : int
{
    FMDisplayTypeFullscreen = 0,
    FMDisplayTypeLarge = 1,
    FMDisplayTypeSmall = 2
} FMDisplayType;


@property FMContentProvider contentProvider: Enum that informs you of where the video is
hosted.
typedef enum : int
{
    FMContentProviderCustom = 0,
    FMContentProviderYoutube = 1,
    FMContentProviderVimeo = 2
} FMContentProvider;

@property (nonatomic, retain) NSString *vidURL: A URL that points to the video.
```

Below is an example of how you could display a video experience. You could use the properties however you like, but the intention and most common means of using the following properties is shown below. Depending on how you configure your experiences in the ExtremeLocation UI, you may not populate data for all of the properties. In this case, remember to do proper error checking and just retrieve what attributes makes sense.

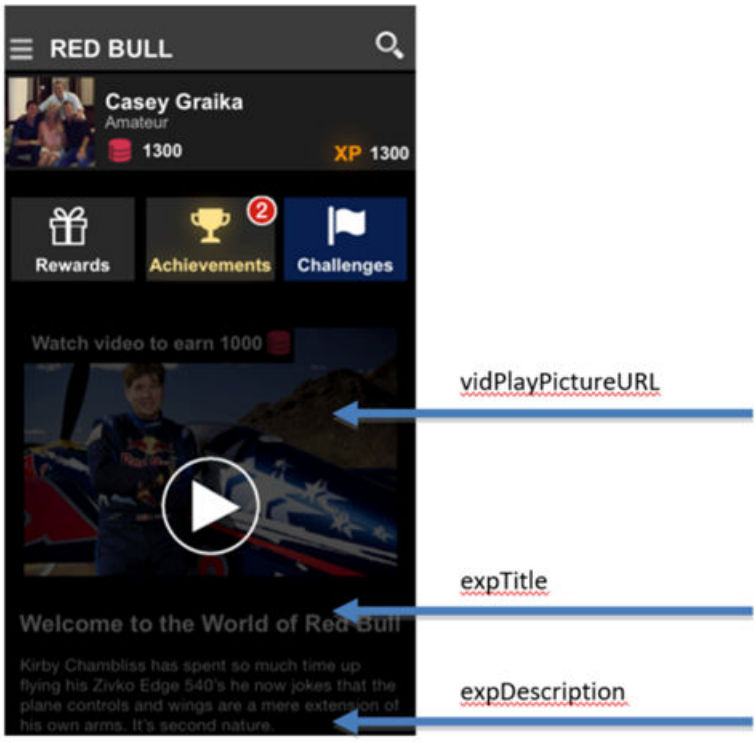

Figure 1: FMVideoExp Example

FMImageExp (Objective-C)

FMImageExp contains an image that should be presented to the user.

```
@property (nonatomic, retain) NSString *imgURL: A URL that points to the image.
```

Below are 2 examples of how an FMImageExp could be displayed. One using only the expTitle property and the other using both the expTitle and expDescription properties.

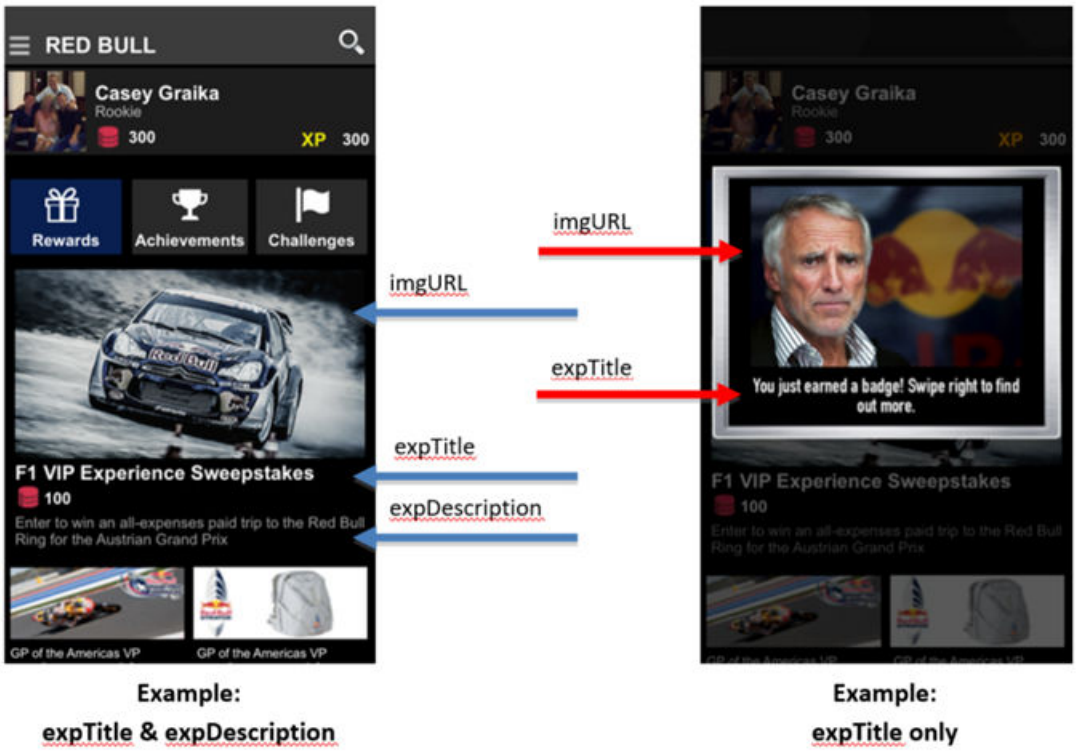

Figure 2: FMImageExp Example

FMAlertExp

An experience that is intended to simply present a notification to the user. This type of experience is meant to alert the user of updated news, trend, or any current related events and info. For this type of experience help breaks the different category between the other experiences.

Below is an example of how the FMAlertExp may be used. The text displayed in the UILocalNotification is pulled from the Experience's notificationTitle property and notificationDescription property. You may use one or both.

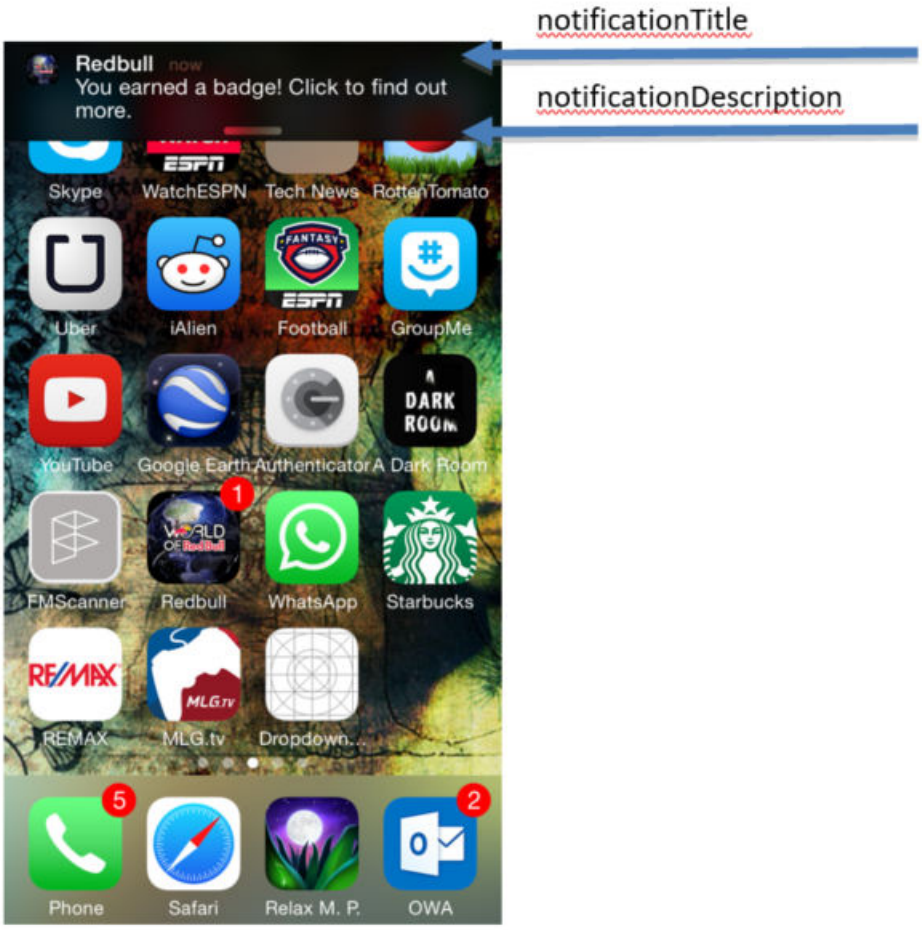

Figure 3: FMAlertExp Example

FMCustomExp

This is a catchall experience that can be modified to fit your use case in the ExtremeLocation UI. If used, the customized data will be returned in the base class's Attributes property.

4 Send Converted Experience

To obtain the analytics of the user interactions from the experience, use the methods below that are contained within the FMExperience class. After the user has taken action on the experience received from the FMExperienceManager delegate callbacks, call the send converted experience methods to send data analytics of the user actions.

FMExperience Methods

Below is a list of methods to invoke converted user experiences that are contained within the FMExperience class:

- sendConvertedExperienceWithType (Objective-C)

  Method that informs the ExtremeLocation Server that a user has performed some action on this FMExperience. Sending this information allows for improved analytics.

```
-(void)sendConvertedExperienceWithType:(FMConvertedAction *)action
    valueType:(FMConvertedValueType)vType andValue:(float)value;
```

- sendCustomConvertedExperienceWithType (Objective-C)

  Method that informs the ExtremeLocation Server that a user has performed some action on this FMExperience. Sending this information allows for improved analytics. This method also allows for custom actions and custom values to be specified, allowing for flexible analytics.

```
-(void)sendCustomConvertedExperienceWithType:(FMConvertedAction *)action
andCustomActionName:(NSString*)customAction valueType:(FMConvertedValueType)vType
andCustomValueName:(NSString*)customValName andValue:(float)value;
```

List of Converted Actions

Represents how the user interacted with the experience.

Action	Method Name	Method Description
No Action	FMConvertedActionNone	The user did not interact with the experience
Watched	FMConvertedActionWatched	The user watched the video within the experience
Clicked	FMConvertedActionClicked	The user clicked the experience displayed
Swiped	FMConvertedActionSwiped	The user swiped the experience displayed.
Listened	FMConvertedActionListened	The user listened to the audio within the experience

Action	Method Name	Method Description
Shared	FMConvertedActionShared	The user shared the experience
Opened	FMConvertedActionOpened	The user performed some action in order to open the experience
Automated	FMConvertedActionAutomated	The user was presented with the experience automatically
Retargeted	FMConvertedActionRetargeted	The user was re-targeted with the experience
Custom	FMConvertedActionCustom	A custom converted action was performed on the experience

List of Converted Values

Represents how the user measures the action of the experience.

Measure	Method Name	Method Description
Seconds	FMConvertedValueTypeSeconds	The value in seconds
Minutes	FMConvertedValueTypeMinutes	The value in minutes
Currency	FMConvertedValueTypeCurrency	The value in currency or exchange
Quantity	FMConvertedValueTypeQuantity	The value in quantify amount. This can be very large or small
Custom	FMConvertedValueTypeCustom	The value can pertain to any amount, length, or mass