MPACT LOCATION & ANALYTICS

Client Software Development Kit

VERSION 2.0.0

MN002604A01

TABLE OF CONTENTS

Chapter 3 iOS API

Chapter 4 Windows CE API

Appendix A Customer Support

ABOUT THIS GUIDE

This chapter is organized into the following sections:

Using the Documentation

The following sections provide information about the document and notational conventions used in the guides, and provides a list of related documentation:

Document Conventions

The following conventions are used in this manual to draw your attention to important information:

NOTE: Indicates tips or special requirements.

CAUTION: Indicates conditions that can cause equipment damage or data loss.

WARNING! Indicates a condition or procedure that could result in personal injury or equipment damage.

Revision History

This guide has the following release and revision milestone history:

Release	*Date*	*Change*
1.0 Revision A	August, 2014	Release of initial 1.0 baseline of MPact.
1.0.1 Revision B	December, 2014	Updated to 1.0.1 feature baseline.
1.0.2 Revision C	March, 2015	Updated to 1.0.2 feature baseline.
2.0.0 Revision A	February, 2016	Updated to 2.0.0 feature baseline and incorporated new Zebra templates.

Notational Conventions

The following notational conventions are used in this document:

- Italics are used to highlight specific items in the general text, and to identify chapters and sections in this and related documents
- Bullets (•) indicate:
  - lists of alternatives
  - lists of required steps that are not necessarily sequential
  - action items
- Sequential lists (those describing step-by-step procedures) appear as numbered lists

Related Documentation

MPact Location and Analytics documentation includes the following:

- *MPact Location & Analytics Deployment Guide*
- *MPact Location & Analytics Server Reference Guide*
- *MPact Location & Analytics Android Toolbox User Guide*
- *MPact Location & Analytics iOS Toolbox User Guide*
- *MPact Location & Analytics Client Software Development Kit*
- *MPact Location & Analytics Server API Reference Guide*
- *MPact Location & Analytics Hardware Installation Guide*

Zebra Technologies Corporation ("Zebra") End-User Software License Agreement

BY INSTALLING AND/OR USING THIS PRODUCT, YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT AND AGREE TO BE BOUND ITS TERMS. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, ZEBRA IS NOT WILLING TO LICENSE THE PRODUCT TO YOU, AND YOU MUST NOT INSTALL OR USE THIS PRODUCT.

Definitions

Grant of License

Zebra Technologies Corporation ("Zebra") grants you ("Licensee" or "you") a personal, nonexclusive, nontransferable, revocable, nonassignable, limited license to use the software and documentation ("Product(s)") subject to the terms and conditions of this Agreement. You shall use the Products only for your internal business purposes, exclusively to support Zebra devices. Any use of the Products outside of the conditions set forth herein is strictly prohibited and will be deemed a breach of this Agreement resulting in immediate termination of your License. In the event of a breach of this Agreement, Zebra will be entitled to all available remedies at law or in equity (including immediate termination of the license without notice, immediate injunctive relief and repossession of all Products unless Licensee is a Federal agency of the United States Government).

You shall not distribute, sublicense, rent, loan, lease, export, re-export, resell, ship or divert or cause to be exported, re-exported, resold, shipped or diverted, directly or indirectly, the Products under this Agreement. You shall not, and shall not permit others to: (i) modify, translate, decompile, bootleg, reverse engineer, disassemble, or extract the inner workings of the Products, (ii) copy the look-and-feel or functionality of the Products; (iii) remove any proprietary notices, marks, labels, or logos from the Products; (iv) rent or transfer all or some of the Products to any other party without Zebra's prior written consent; or (v) utilize any computer software or hardware which is designed to defeat any copy protection device, should the Products be equipped with such a protection device.

Title to all copies of Products will not pass to Licensee at any time and remains vested exclusively in Zebra. All intellectual property developed, originated, or prepared by Zebra in connection with the Products remain vested exclusively in Zebra, and this Agreement does not grant to Licensee any intellectual property rights.

Portions of the Products are protected by United States patent and copyright laws, international treaty provisions, and other applicable laws. Therefore, you must treat the Products like any other copyrighted material (e.g., a book or musical recording) except that you may make one copy of the Product solely for back-up purposes. Unauthorized duplication of the Products constitutes copyright infringement, and in the United States is punishable in federal court by fine and imprisonment.

Limited Warranty

Zebra warrants for a period of ninety (90) days from your receipt of the Products to you that the Software, under normal use, will perform substantially in accordance with Zebra's published specifications for that release level of the Software. The written materials are provided "AS IS" and without warranty of any kind. Zebra's entire liability and your sole and exclusive remedy for any breach of the foregoing limited warranty will be, at Zebra's option, the provision of a downloadable patch or replacement code, or a refund of the unused portion of your bargained for contractual benefit up to the amount paid for the Products.

Disclaimer

THIS LIMITED WARRANTY IS THE ONLY WARRANTY PROVIDED BY ZEBRA, AND ZEBRA MAKES, AND YOU RECEIVE, NO OTHER WARRANTIES OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY, OR IN ANY COMMUNICATION WITH YOU. ZEBRA SPECIFICALLY DISCLAIMS ANY WARRANTY INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILTY, NONINFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. ZEBRA DOES NOT WARRANT THAT THE PRODUCTS WILL MEET YOUR REQUIREMENTS, OR THAT THE OPERATION OF THE PRODUCTS WILL BE UNINTERRUPTED OR ERROR FREE, OR THAT DEFECTS IN THE PRODUCTS WILL BE CORRECTED. ZEBRA MAKES NO WARRANTY WITH RESPECT TO THE

CORRECTNESS, ACCURACY, OR RELIABILITY OF THE PRODUCTS. Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to you.

Limitation of Liability

THE TOTAL LIABILITY OF ZEBRA UNDER THIS AGREEMENT FOR DAMAGES SHALL NOT EXCEED THE FAIR MARKET VALUE OF THE PRODUCTS LICENSED UNDER THIS AGREEMENT. IN NO EVENT WILL ZEBRA BE LIABLE IN ANY WAY FOR INCIDENTAL, CONSEQUENTIAL, INDIRECT, SPECIAL OR PUNITIVE DAMAGES OF ANY NATURE, INCLUDING WITHOUT LIMITATION, LOST BUSINESS PROFITS, OR LIABILITY OR INJURY TO THIRD PERSONS, WHETHER FORESEEABLE OR NOT, REGARDLESS OF WHETHER ZEBRA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Some jurisdictions do not permit limitations of liability for incidental or consequential damages, so the above exclusions may not apply to you. This Limitation of Liability provision survives the termination of this Agreement and applies notwithstanding any contrary provision in this Agreement. Licensee must bring any action under this Agreement within one (1) year after the cause of action arises.

Maintenance

Unless provided for in a separate agreement, Zebra shall not be responsible for maintenance or field service of the Products.

High Risk Activities

The Products are not fault-tolerant and are not designed, manufactured or intended for use or resale as on-line control software in hazardous environments requiring fail-safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines, or weapons systems, in which the failure of the Products could lead directly to death, personal injury, or severe physical or environmental damage ("High Risk Activities"). Zebra and its suppliers specifically disclaim any express or implied warranty of fitness for High Risk Activities, and if you elect to use the Products in any High Risk Activities, you agree to indemnify, defend, and hold Zebra harmless from and against any and all costs, damages, and losses related to that use.

U.S. Government

If you are acquiring the Products on behalf of any unit or agency of the U.S. Government, the following shall apply. Use, duplication, or disclosure of the Products is subject to the restrictions set forth in subparagraphs (c) (1) and (2) of the Commercial Computer Software - Restricted Rights clause at FAR 52.227-19 (JUNE 1987), if applicable, unless being provided to the Department of Defense. If being provided to the Department of Defense, use, duplication, or disclosure of the Products is subject to the restricted rights set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 (OCT 1988), if applicable. Products may or may not include a Restricted Rights notice, or other notice referring specifically to the terms and conditions of this Agreement. The terms and conditions of this Agreement shall each continue to apply, but only to the extent that such terms and conditions are not inconsistent with the rights provided to you under the aforementioned provisions of the FAR and DFARS, as applicable to the particular procuring agency and procurement transaction.

Assignment

Except as otherwise provided in this section, neither party may assign this Agreement, or any of its rights or obligations under this Agreement, without the prior written approval of the other party, which will not be unreasonably withheld. Any attempted assignment, delegation, or transfer without the necessary approval will be void. Notwithstanding the foregoing, for any Zebra acquisition, merger, consolidation, reorganization, or similar transaction, or any spin-off, divestiture, or other separation of a Zebra business, Zebra may, without the prior written consent of the other party: (i) assign its rights and obligations under this Agreement, in whole or in part, or (ii) split and assign its rights and obligations under this Agreement so as to retain the benefits of this Agreement for both Zebra and the assignee entity(ies) (and their respective Affiliates) following the split.

Governing Law

This Agreement shall be governed by the laws of the United States of America to the extent that they apply and otherwise by the laws of the State of New York without regard to its conflict of laws provisions or by the internal substantive laws of the country to which the Products is shipped if end-user customer is a sovereign governmental entity. The terms of the U.N. Convention on Contracts for the International Sale of Goods do not apply. In the event that the Uniform Computer information Transaction Act, any version of this Act, or a substantially similar law (collectively "UCITA") becomes applicable to a Party's performance under this Agreement, UCITA does not govern any aspect of this End User License Agreement or any license granted under this End-User License Agreement, or any of the parties' rights or obligations under this End User License Agreement. The governing law will be that in effect prior to the applicability of UCITA.

Compliance with Laws

Licensee will comply with all applicable laws and regulations, including export laws and regulations of the United States. Licensee will not, without the prior authorization of Zebra and the appropriate governmental authority of the United States, in any form export or re-export, sell or resell, ship or reship, or divert, through direct or indirect means, any item or technical data or direct or indirect products sold or otherwise furnished to any person within any territory for which the United States Government or any of its agencies at the time of the action, requires an export license or other governmental approval. Violation of this provision will be a material breach of this Agreement, permitting immediate termination by Zebra.

Third Party Software

The Products may contain one or more items of Third-Party Software. The terms of this Agreement govern your use of any Third-Party Software UNLESS A SEPARATE THIRD-PARTY SOFTWARE LICENSE IS INCLUDED, IN WHICH CASE YOUR USE OF THE THIRD-PARTY SOFTWARE WILL THEN BE GOVERNED BY THE SEPARATE THIRD-PARTY LICENSE.

Open Source Software

The Products may contain one or more items of Open Source Software. Open Source Software is software covered by a publicly available license governed solely under Copyright law, whereas the complete terms and obligations of such license attach to a licensee solely through the act of copying, using and/or distribution of the licensed software, such obligations often include one or more of attribution obligations, distribution obligations, copyleft obligations, and intellectual property encumbrances. The use of any Open Source Software is subject to the terms and conditions of this Agreement as well as the terms and conditions of the corresponding license of each Open Source Software package. If there is a conflict between the terms and conditions of this Agreement and the terms and conditions of the Open Source Software license, the applicable Open Source Software license takes precedence. Copies of the licenses for the included Open Source Software, if any, as well as their attributions, acknowledgements, and software information details, are provided in the electronic copy of this Agreement, which is available in the Legal Notices or README file associated with the Product. Zebra is required to reproduce the software licenses, acknowledgments and copyright notices as provided by the authors and owners, thus, all such information is provided in its native language form, without modification or translation. Depending on the license terms of the specific Open Source Software, source code may not be provided. Please reference and review the entire Open Source Software information to identify which Open Source Software packages have source code provided or available. For instructions on how to obtain a copy of any source code made publicly available by Zebra related to Open Source Software distributed by Zebra, you may send your request (including the Zebra Product name and version, along with the Open Source Software specifics) in writing to: Zebra Technologies Corporation, Open Source Software Director, Legal Department, 3 Overlook Point, Lincolnshire, IL 60069 USA.

Obtaining Software Licenses

To obtain software licenses for *MPact Location & Analytics Server, Toolbox or Client Software Development Kit*, provide the following information:

- Identification
- Email address
- Payment

CHAPTER 1MPACT ARCHITECTURE OVERVIEW

MPact is a new real-time micro-locationing system. The MPact locationing solution leverages *Bluetooth Low Energy* (BLE) beacon to allow a Bluetooth 4.0 device to be precisely located. When the phone/tablet supporting BLE gets close to a beacon, it recognizes a wireless beacon from the beacon using its Bluetooth radio. Because the radio is low energy, and its signal does not propagate far, the client is precisely located by the beacons it recognizes.

MPact Client API allows an application to scan MPact beacons and send the information about the closest beacon to the MPact Server. The MPact Client API supports a java interface and is integrated into a partner applications.

Third party developers can optionally utilize an MPact client *software development kit* (SDK) to integrate MPact technology into their own unique mobile applications. The SDK consists of the following:

- *MPact client library*
- *API documentation*
- *Sample application*

The following platforms can utilize the SDK:

- *iOS versions 7.1.2 and higher*
- *Android versions 4.3 and higher*
- *Windows CE versions 7 and higher*

For more information on these APIs available to the iOS, Android and Windows CE platforms, see *Android API on page 2-1*, *iOS API on page 3-1* and *Windows CE API on page 4-1*.

For information on notification restrictions based on platform support, see *Platform Specific Notification Support on page 1-2*.

1.1 MPact Client Library

The MPact client library is specifically designed to integrate into third-party mobile applications to interoperate with the MPact system. The client library provides limited functionality, specifically supporting the interoperation between Bluetooth enabled beacons and the back-end MPact Server. The core pieces of the third-party integrated solution are:

- *Partner application*: Executes on a smartphone to embed the MPact client library
- *MPact client application*: Listens for beacons and forwards data from the nearest deployed beacon to the MPact Server
- *MPact Server*: Data forwarded by the client library application is converted into location information for site management by the MPact administrator

- *Partner cloud service*: Provides a back-end for the partner mobile application. The partner cloud service leverages location data generated by MPact through the MSI Locationing API.

The client library does not include a configurable *user-interface* (UI) or support for retail deployment objectives. Instead, it provides a single interface into the underlying MPact BLE functionality. The client library is leveraged within both MPact partner applications and the MPact Toolbox.

1.2 Platform Specific Notification Support

Third-party partner applications are optionally notified when certain MPact system conditions occur. These notifications assist retailers in making specific advertisements (welcome messages etc.) available to mobile devices as they roam to various areas where these advertised products reside.

NOTE: Category values are assigned to beacons within the MPact Server application for notifications to function. Category values are the products grouped for tracking under the parent product category.

The following notification types are supported:

- Closest Beacon - Called at periodic intervals with the identity of the closest beacon. The method terminates when there is no closest beacon identified. The notification includes the name of the strongest beacon, its RSSI and estimated percentage of remaining battery life.

NOTE: The battery life is not available when the beacon type is set to iBeacon mode.

- Region - Initiated whenever a client device crosses a border transition of an area where iBeacon or MPact mode beacons are deployed. There is a notification when a beacon that matches the UUID is detected for the first time (for instance, when entering a store). Another notification is generated when no additional beacons are detected matching the UUID (for example, when leaving a store). Even applications in the background become active for a few seconds when a transition occurs. For more information, see *setBeaconUUID* for Android, or *iBeaconUUID* for iOS.
- Automatic Import of Regions - Regions are automatically imported. This allows developers to know which regions should be monitored and what notifications are added based on Major and Minor values.

NOTE: There are 19 region notification based on Major and Minor and one for UUID, for a total of 20. These 19 region notification are picked randomly by the client.

The following lists the notification support available to the iOS and Android platforms by their support versions.

Scenario Notification	***Battery Save iOS 7.1.2 and Higher***	***SecureCast iOS 7.1.2 and Higher***	***iBeacon iOS 7.1.2 and Higher***	***MPact iOS 7.1.2 and Higher***	***Battery Save Android 4.3 & 4.4***	***iBeacon Android 4.3 & 4.4***	***MPact Android 4.3 & 4.4***	***SecureCast Android 4.3 & 4.4***
Background Closest Beacon	No	Yes	No	No	Yes	Yes	Yes	Yes
Background Region	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Foreground Closest Beacon	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Foreground Region	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Sleep Mode Closest Beacon	No	No	No	No	Yes	Yes	Yes	Yes
Sleep Mode Region	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes

NOTE: If using iOS, an upgrade to version 7.1.2 or higher is recommended for optimal client notification support. The iOS SDK is supported on Xcode 6.1.

CHAPTER 2ANDROID API

2.1 Package com.zebra.mpact.mpactclient

2.1.1 Interface Summary

MPactClientConsumer	An interface for an Android `Activity` or `Service` that wants to interact with the MPact Server. The interface is used in conjunction with MPactClient, and provides a callback when the MPact Service is ready to use. For more information, see *MPactClientConsumer*.
MPactClientNotifier	Optional callback methods for the MPact Client. Use this method to receive extra information from the MPactClient. For more information, see *MPactClientNotifier*.
MPactLogNotifier	An interface for receiving logs from the MPactLogger class.

2.1.2 Class Summary

MPactCatergoryValus	This class holds the properties that describe the state of an MPact Category Value. For more information, see *MPactClient*.
MPactClient	A class used to interact with the MPact Server from an `Activity` or `Service`. This class is used in conjunction with MPactClientConsumer interface. For more information, see *MPactClient*.
MPactLogger	This class is for enabling the generation of debug log messages by the MPact SDK.
MPactSDKVersion	Contains versioning information for the Android version client SDK. For more information, see *MPactSDKVersion*.
MPactServerInfo	Holds network information relevant to the MPact Server.
MPactTag	Holds the properties that describe the state of an MPact beacon.

2.1.3 Enum Summary

MPactBeaconType	Format of the beacon. For more information, see *MPactBeaconType*.
MPactLogLevel	Debug logging Levels for the MPact SDK. The MPactLogZone combined with the MPactLogLevel determine the messages generated.
MPactLogZone	Available debug logging zones for the MPact SDK.
MPactProximity	The maximum distances considered for determining the closest beacon . The MPact SDK ignores beacons determined beyond the specified proximity range.

2.2 Adding MPact Client API to Your Project

The MPact Client API is distributed as a java jar module. Add the file MPactClient.jar to the libs directory in your project directory to get access to the objects comprising the MPact Client API.

NOTE: The MPact Client API has a dependency on the JSON library.

2.2.1 Project Manifest

The MPact Client library makes use of the Internet and BlueTooth LE. Third part applications require the following permissions be added.

- android.permission.INTERNET
- android.permission.BLUETOOTH
- android.permission.BLUETOOTH_ADMIN

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/
```

The MPact Client library starts a local service to process BLE beacons. Third party applications need to add MPactService to their project manifest.

```
<application
              <service android:enabled="true"
                  android:exported="false"
                  android:isolatedProcess="false"
                  android:label="MPact"
                  android:name="com.zebra.mpact.mpactclient.MPactService">
              </service>
          </application>
```

2.3 MPact Sample Code

This section provides sample Android projects and example code.

2.3.1 Session Cookies

The MPact Client library uses session cookies, but does not manage its own cookies. A third party application needs to implement a Cookie Manager for the MPact Server to function properly.

2.3.2 MPact Sample Projects

The following Android Studio projects are included in the MPact Client SDK:

- *MPactClientSample* - This app show how to use most of the MPact Client API. Scanning can be enabled/disabled and several different configurations parameters, such as UUID, can be modified. There is also the option of establishing a connection to the MPact Server.
- *MPactNofify* - This app generates notifications about the region state to the device in the background. Notifications continue even if the app is killed by the user. The app auto-launches in the background on boot.

2.3.3 MPactActivity Example Code

In the example below, an Activity implements the MPactClientConsumer interface, binds to the MPactService through the MPactClient, then starts looking for MPact beacons when it gets the callback saying the service is ready.

```
public class MPactActivity extends Activity implements MPactClientConsumer {
      protected static final String TAG = "MPactActivity";
      private static CookieManager cookieManager;
      private MPactClient mpactClient =
      MPactClient.getInstanceForApplication(this);
      @Override
      protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.activity_ranging);
          mpactClient.bind(this);
          // Setup cookies
          cookieManager = new CookieManager();
          CookieHandler.setDefault(cookieManager);
      }
       @Override
      protected void onDestroy() {
          super.onDestroy();
          mpactClient.unBind(this);
      }
       @Override
      public void onMPactClientServiceConnect() {
```

```
            mpactClient.setNotifier(new MPactClientNotifier() {
                @Override
                public void didDetermineClosestTag(MPactTag mpactTag) {
                    Log.i(TAG, "Tag " + mpactTag.getTagID() + " is near.");
                }
                @Override
                public void didDetermineState(int state) {
                    if(state == MPactClientNotifier.INSIDE) {
                        Log.i(TAG, "Entered an area with tags.");
                    } else {
                        Log.i(TAG, "There are no tags nearby.");
                    }
                }
            });
            try {
                mpactClient.Start();
            } catch (RemoteException e) {    }
        }
    }
```

2.3.4 AndroidManifest.xml Sample Code

In the example below, AndroidManifest.xml: auto launches the app when the device is powered up, prevents two copies of the app from being created and enables the MonitorService to re-launch the app if killed.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.zebra.mpact.mpactnotify" >

    <!-- This permission is needed to allow the app to turn on the screen
      when it creates the notification-->
    <uses-permission android:name="android.permission.WAKE_LOCK" />

    <application
        android:name=".NotifyApplication"
        android:allowBackup="true"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

       <!-- The launchMode option prevents two copies of the app from being
       created -->
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:launchMode="singleInstance" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

               <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <service
            android:name="com.zebra.mpact.mpactclient.MPactService"
            android:enabled="true"
            android:exported="false"
            android:isolatedProcess="false"
            android:label="MPact" >
        </service>
```

```
        <!-- This section enables auto-launching the app when the device is
         first powered up -->
        <receiver android:name=".LaunchReceiver" >
            <intent-filter>
              <action android:name="android.intent.action.BOOT_COMPLETED" />
            </intent-filter>
        </receiver>

        <!-- This section enables a service that will relaunch the app if it
        is killed -->
        <service
            android:name=".MonitorService"
            android:enabled="true"
            android:exported="false" >
        </service>

    </application>

</manifest>
```

2.3.5 LaunchReceiver Sample Code

In the example below, LaunchReciever.java starts the app using a received broadcast message.

```
package com.zebra.mpact.mpactnotify;

import android.app.Application;
import android.content.Intent;

public class NotifyApplication extends Application {
    public void onCreate() {
        super.onCreate();

        // Start up the background beacon monitoring service
        Intent myIntent = new Intent(this, MonitorService.class);
        startService(myIntent);
    }
}
```

2.3.6 NotifyApplication Sample Code

In the example below, NotifyApplication.java starts the background beacon monitoring service.

```
package com.zebra.mpact.mpactnotify;

import android.content.BroadcastReceiver;

/**
 * The only purpose of this class is to startup the application through receiving a
 * broadcast message from the system.
 */
public class LaunchReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(android.content.Context context, android.content.Intent
intent) {
        /* Uncomment this out to make the main activity visible in the task switcher
        Intent myIntent = new Intent(context, MainActivity.class);
        myIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(myIntent);
        */
    }
}
```

2.3.7 MonitorService Sample Code

In the example below, the code creates a service, running in the background, and restarts the service if killed.

```
package com.zebra.mpact.mpactnotify;


import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.Service;
import android.app.TaskStackBuilder;
import android.content.Context;
import android.content.Intent;
import android.os.IBinder;
import android.os.RemoteException;
import android.support.v4.app.NotificationCompat;
import android.util.Log;


import com.zebra.mpact.mpactclient.MPactBeaconType;
import com.zebra.mpact.mpactclient.MPactClient;
import com.zebra.mpact.mpactclient.MPactClientConsumer;
import com.zebra.mpact.mpactclient.MPactClientNotifier;
import com.zebra.mpact.mpactclient.MPactLogLevel;
```

```
import com.zebra.mpact.mpactclient.MPactLogZone;
import com.zebra.mpact.mpactclient.MPactLogger;
import com.zebra.mpact.mpactclient.MPactProximity;
import com.zebra.mpact.mpactclient.MPactTag;

public class MonitorService extends Service
        implements MPactClientConsumer
{
    static MPactClient mpactClient = null;
    static RegionNotifier regionNotifier = null;

    @Override
    public IBinder onBind(Intent intent) {
        // Binding is not supported
        return null;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        if(mpactClient == null) {
            // Get instance of the MPact Service
            mpactClient =
MPactClient.getInstanceForApplication(getApplicationContext());
        }

        // Bind to the MPact Service
        mpactClient.bind(this);

        // This return code will respawn the app if it gets killed
        return START_STICKY;
    }

    @Override
    public void onMPactClientServiceConnect() {
        // Setup parameters and start the MPact Client
        if(regionNotifier == null) {
            // Get instance of the MPact Service
```

```
            regionNotifier =
RegionNotifier.getInstanceForApplication(getApplicationContext());
        }

        mpactClient.setNotifier(regionNotifier);
        mpactClient.setiBeaconUUID("fe913213-b311-4a42-8c16-47faeac938db");
        try {
            mpactClient.Start();
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }
}
```

2.4 Hierarchy For Package com.zebra.mpact.mpactclient

Package Hierarchy: com.zebra.mpact.mpactclient

Class Hierarchy

- java.lang.Object
  - com.zebra.mpact.mpactclient.**MPactClient**
  - com.zebra.mpact.mpactclient.**MPactLogger**
  - com.zebra.mpact.mpactclient.**MPactSDKVersion**
  - com.zebra.mpact.mpactclient.**MPactServerInfo**
  - com.zebra.mpact.mpactclient.**MPactTag**

Interface Hierarchy

- com.zebra.mpact.mpactclient.**MPactClientConsumer**
- com.zebra.mpact.mpactclient.**MPactClientNotifier**
- com.zebra.mpact.mpactclient.**MPactLogNotifier**

Enum Hierarchy

- java.lang.Object
  - java.lang.Enum<E> (implements java.lang.Comparable<T>, java.io.Serializable)
  - com.zebra.mpact.mpactclient.**MPactBeaconType**
  - com.zebra.mpact.mpactclient.**MPactLogLevel**
  - com.zebra.mpact.mpactclient.**MPactLogZone**

2.5 MPactCatergoryValue

This class holds properties describing the state of an MPact Category Value. Categories and Values are defined on the MPact Server, and assigned to individual beacons.

2.5.1 MPactCatergoryValue

Constructor for com.zebra.mpact.mpactclient.**MPactCategoryValue**.

2.5.2 GetCatergory

Method in class com.zebra.mpact.mpactclient.**MPactCategoryValue**.

Returns:

The category string associated with this object.

2.5.3 SetCatergory

Method in class com.zebra.mpact.mpactclient.**MPactCategoryValue**.GetCatergory.

Parameters:

category - The category string associated with this object.

2.5.4 GetValue

Method in class com.zebra.mpact.mpactclient.**MPactCategoryValue**

Returns:

The value string associated with this object.

2.5.5 SetValue

Method in class com.zebra.mpact.mpactclient.**MPactCategoryValue**.GetCatergory.

Parameters: value - The value string associated with this object

2.6 MPactClient

A class used to interact with the MPact Server from an Activity or Service. This class is used in conjunction with the *MPactClientConsumer* interface.

2.6.1 MPactClient

Constructor for com.zebra.mpact.mpactclient.**MPactClient**

```
protected MPactClient(android.content.Context context)
```

2.6.2 getInstanceForApplication

Static method in class com.zebra.mpact.mpactclient.**MPactClient**

```
public static MPactClient getInstanceForApplication(android.content.Context
context)
```

Gets a single instance of this class. A context must be provided to access this method. However, it is used from a non-Activity or non-Service class. It is attached to another singleton or a subclass of the Android Application class.

2.6.3 bind

Method in class com.zebra.mpact.mpactclient.**MPactClient**.

`public void bind (`**MPactClientConsumer** consumer)

Binds an Android `Activity` or `Service` to the MPactService. The Activity or Service must implement the *MPactClientConsumer* interface so it gets a callback when the service is ready.

Parameters:

consumer - The `Activity` or `Service` receives the callback when the service is ready.

2.6.4 unBind

Method in class com.zebra.mpact.mpactclient.**MPactClient**

```
public void unBind(MPactClientConsumer consumer)
```

Unbinds an Android Activity or Service to the MPactService. This is typically called in the onDestroy() method.

Parameters:

consumer - The Activity or Service that no longer needs to use the service.

2.6.5 isBound

Method in class com.zebra.mpact.mpactclient.**MPactClient.**

```
public boolean isBound(MPactClientConsumer consumer)
```

Informs whether the passed *MPactClientConsumer* is bound to the service

Parameters:

The Activity or Service that no longer needs to use the service.

Returns:

True if the consumer is bound to the service.

2.6.6 setNotifier

Method in class com.zebra.mpact.mpactclient.**MPactClient**

```
public void setNotifier(MPactClientNotifier notifier)
```

Specifies a class called each time the `MPactService` has a notification that should be sent. Only one *MPactClientNotifier* is active for a given application. If two different activities or services set different *MPactClientNotifier* instances, the last one set receives all the notifications.

Parameters:

`notifier - None`

See Also:

MPactClientNotifier

Start

2.6.7 Start

Method in class com.zebra.mpact.mpactclient.**MPactClient**

```
public void Start()
            throws android.os.RemoteException
```

Tells the MPactService to start looking for beacons and estimate the closest beacon at one second intervals.

Throws:

`android.os.RemoteException`

See Also:

MPactClientNotifier,

Stop

2.6.8 Stop

Method in class com.zebra.mpact.mpactclient.**MPactClient**

```
public void Stop()
           throws android.os.RemoteException
```

Tells the `MPactService` to stop looking for beacons and provides updates on the estimated closest beacon.

Throws:

`android.os.RemoteException`

See Also:

MPactClientNotifier

2.6.9 getVersion

Static method in class com.zebra.mpact.mpactclient.**MPactClient**

```
public static java.lang.String getVersion()
```

Returns:

Version of the MPact API

2.6.10 getServer

Method in class com.zebra.mpact.mpactclient.**MPactClient**

```
public MPactServerInfo getServer()
```

Returns:

Network information about the MPact Server

See Also:

MPactServerInfo

2.6.11 setServer

Method in class com.zebra.mpact.mpactclient.**MPactClient**

```
public void setServer(MPactServerInfo serverInfo)
```

Parameters:

```
serverInfo - Network information about the MPact Server
```

See Also:

MPactServerInfo

2.6.12 getiBeaconUUID

Method in class com.zebra.mpact.mpactclient.**MPactClient**

```
public java.lang.String getiBeaconUUID()
```

Returns:

The UUID of the iBeacon. The default UUID is returned when **setBeaconType(MPactBeaconType)** is set to **MPactBeaconType.BatterySave.**

2.6.13 setBeaconUUID

Method in class com.zebra.mpact.mpactclient.**MPactClient**

```
public void setiBeaconUUID(java.lang.String iBeaconUUID)
```

Parameters:

`iBeaconUUID` - The UUID of the iBeacon. Changes to this property are ignored if `setBeaconType(MPactBeaconType)` is set to `MPactBeaconType.BatterySave.`

2.6.14 getBeaconType

Method in class com.zebra.mpact.mpactclient.**MPactClient**

```
public MPactBeaconType getBeaconType()
```

Returns:

The location format used by MPact beacons

See Also:

MPactBeaconType

2.6.15 setBeaconType

Method in class com.zebra.mpact.mpactclient.**MPactClient**

```
public void SetBeaconType (MpactBeaconType beaconType)
```

Parameters:

`beaconType` - The location format used by the MPact beacons

See Also:

MPactBeaconType

2.6.16 getClosestTag

Method in class com.zebra.mpact.mpactclient.**MPactClient**

```
public MPactTag getClosestTag()
```

Returns:

Attributes of the closest beacon. Returns null if no beacon is determined to be closest.

See Also:

MPactTag

2.6.17 getClientID

Method in class com.zebra.mpact.mpactclient.**MPactClient**

```
public java.lang.String getClientID()
```

Returns:

The identity of this device. The clientId is sent to the MPact Server with the closest beacon updates.

2.6.18 getClientName

Method in class com.zebra.mpact.mpactclient.**MPactClient**

```
public java.lang.String getClientName()
```

Returns:

Human readable name for the device. The client name is sent to the MPact Server with the closest beacon updates.

2.6.19 setClientName

Method in class com.zebra.mpact.mpactclient.**MPactClient**

```
public void setClientName(java.lang.String clientName)
```

Parameters:

`clientName` - Human readable name for the device

2.6.20 getClientName

Method in class com.zebra.mpact.mpactclient.**MPactClient**

```
public java.lang.String getClientName()
```

Returns:

Human readable name for the device. The client name is sent to the MPact Server with the closest beacon updates.

2.7 MPactLogger

Generates MPact SDK debug log messages. The app writer has the option of generating log messages sent to the console and/or directly to a class that implements the MPactLogNotifier interface.

```
public void MPactLogger()
          extends java.lang.Object
```

2.7.1 getAppLoggingZone

Method in interface com.zebra.mpact.mpactclient.**MPactLogger**

```
public MPactLogLevel getAppLoggingZone(MPactLogZone zone)
```

Get the MPactLoggerNotifier interface logging level for the specified zone.

Parameters:

`zone` - The logging zone to change. See *MPactLogZone*.

Returns:

The current logging level for the requested zone.

See Also:

MPactLogZone, *MPactLogLevel*

2.7.2 setAppLoggingZone

Method in interface com.zebra.mpact.mpactclient.**MPactLogger**

```
public MPactLogLevel getAppLoggingZone(MPactLogZone zone)
```

Sets the zones and levels used for logging to the app. Messages are only received by the app if the MPactLoggerNotifier interface is implemented.

Parameters:

`zone` - The logging zone to change. See *MPactLogZone*.

`level` - The logging level requested for the zone.

Returns:

The current logging level for the requested zone.

See Also:

MPactLogZone, *MPactLogLevel*

2.7.3 getConsoleLoggingZone

Method in interface com.zebra.mpact.mpactclient.**MPactLogger**

```
public MPactLogLevel getConsoleLoggingZone(MPactLogZone zone)
```

Gets the system console logging level for the specified zone.

Parameters:

`zone` - The logging zone to change. See *MPactLogZone*.

Returns:

The current logging level for the requested zone.

See Also:

MPactLogZone, *MPactLogLevel*

2.7.4 setConsoleLoggingZone

Method in interface com.zebra.mpact.mpactclient.**MPactLogger**

```
public void setConsoleLoggingZone(MPactLogZone zone,
                                  MPactLogLevel level)
```

Sets the zones and levels used for logging to the system console.

Parameters:

`zone` - The logging zone to change. See *MPactLogZone*.

`level` - The logging level requested for the zone.

See Also:

MPactLogZone, *MPactLogLevel*

2.7.5 getInstance

Method in interface com.zebra.mpact.mpactclient.**MPactLogger**

```
public static MPactLogger getInstance()
```

Use this method to get an instance of the MPactLogger.

2.7.6 isLoggable

Method in interface com.zebra.mpact.mpactclient.**MPactLogger**

```
public boolean isLoggable(MPactLogZone zone,
                          MPactLogLevel level)
```

Helper method that determines if a message is logged.

Parameters:

`zone` - The logging zone to change. See *MPactLogZone*.

`level` - The logging level requested for the zone.

2.7.7 log

Method in interface com.zebra.mpact.mpactclient.**MPactLogger**

```
public void log(MPactLogZone zone,
       MPactLogLevel level,
       java.lang.String message)
```

Used by MPact Client SDK to log a message.

Parameters:

`zone` - The logging zone to change. See *MPactLogZone*.

`level` - The logging level requested for the zone.

`message` - The message logged.

2.7.8 setLogNotifier

Method in interface com.zebra.mpact.mpactclient.**MPactLogger**

```
public void log(MPactLogZone zone,
       MPactLogLevel level,
       java.lang.String message)
```

Used by MPact Client SDK to log a message.

Parameters:

`zone` - The logging zone to change. See *MPactLogZone*.

`level` - The logging level requested for the zone.

`message` - The message logged.

2.8 MPactClientConsumer

`public interface` **MPactClientConsumer**

An interface for an Android `Activity` or `Service` interacting with the MPact Server. The interface is used in conjunction with *MPactClient* and provides a callback when the MPact Service is ready to use.

Parameters:

`consumer` - The `Activity` or `Service` triggering the callback when the service is ready.

See Also:

MPactClient

2.8.1 onMPactClientServiceConnect

Method in interface com.zebra.mpact.mpactclient.**MPactClientConsumer**

```
void onMPactClientServiceConnect()
```

Called when the MPact service is running and ready to accept commands through the MPactClient

See Also:

MPactClient

2.8.2 getApplicationContext

Method in interface com.zebra.mpact.mpactclient.**MPactClientConsumer**

```
android.content.Context getApplicationContext()
```

Called by the *MPactClient* to get the context of your Service or Activity. This method is implemented by Service or Activity and should not be overridden.

Returns:

The application context of your service or activity

2.8.3 unbindService

Method in interface com.zebra.mpact.mpactclient.**MPactClientConsumer**

```
void unbindService(android.content.ServiceConnection connection)
```

Called by the *MPactClient* to unbind your MPactClientConsumer to the MPactService.

2.8.4 bindService

Method in interface com.zebra.mpact.mpactclient.**MPactClientConsumer**

```
boolean bindService(android.content.Intent intent,
                    android.content.ServiceConnection connection,
                    int mode)Called by the MPactClient to bind your
MPactClientConsumer to the MPactService.
```

Called by the MPactClient to bind MPactClientConsumer to MPactService. This method is implemented by Service or Activity, and should not be overridden.

Returns:

True is returned if successfully bound to the service. False is returned if the connection is not made and a service object is not returned..

2.9 MPactClientNotifier

```
public interface MPactClientNotifier
```

Optional callback methods for the MPact Client. Use these methods to receive extra information from the MPactClient.

2.9.1 INSIDE

Static variable in interface com.zebra.mpact.mpactclient.**MPactClientNotifier.**

```
static final int INSIDE
```

Indicates the Android device is currently seeing beacons matching the UUID specified by `MPactClient.setiBeaconUUID(String)`.

See Also:

Constant Field Values

2.9.2 OUTSIDE

Static variable in interface com.zebra.mpact.mpactclient.**MPactClientNotifier**

Indicates the Android device is NOT currently seeing beacons matching UUID specified by `MPactClient.setiBeaconUUID(String).`

See Also:

Constant Field Values

2.9.3 didDetermineClosestTag

Method in interface com.zebra.mpact.mpactclient.**MPactClientNotifier**

```
void didDetermineClosestTag(MPactTag mpactTag)
```

Called once per second to estimate the closest beacon

Parameters:

`mpactTag` - The beacon determined closest.

2.9.4 didDetermineState

Method in interface com.zebra.mpact.mpactclient.**MPactClientNotifier**

```
void didDetermineState (int state)
```

Called with a state value of *INSIDE* the first time a beacon matching the UUID specified by `MPactClient.setiBeaconUUID(String)` is seen by the device. Called with a state value of *OUTSIDE* when the device is no longer receiving beacons matching the UUID specified by `MPactClient.setiBeaconUUID(String)`.

Parameters:

state - either *INSIDE* or *OUTSIDE*

See Also:

MPactClient

2.9.5 didDetermineState

Method in interface com.zebra.mpact.mpactclient.**MPactClientNotifier**

```
void didDetermineState(int state,
                       java.lang.Integer major,
                       java.lang.Integer minor)
```

Called with a state value of INSIDE the first time a beacon from any beacon matching the UUID/major/minor combination is seen by the device. Called with a state value of OUTSIDE when the device is no longer receiving beacons from any tag matching the UUID/major/minor combination. Null for both the major and minor indicates the notification is for matching the region defined by MPactClient.setiBeaconUUID(String).

Parameters:

`state` - either *INSIDE* or *OUTSIDE*

`major` - The major value of the region triggering the event. This value is set to null if the region is not specific to a major value.

`minor` - The minor value of the region triggering the event. This value is set to null if the region is not specific to a minor value.

See Also:

MPactClient

2.9.6 didDetermineState

```
void didDetermineState(int state,
                       MPactCategoryValue categoryValue)
```

Called with a state value of INSIDE the first time a beacon from any tag that matches the Category/Value combination is seen by the device. Called with a state value of OUTSIDE when the device is no longer receiving beacons from any tag that matches the Category/Value combination.

Parameters:

`state` - either *INSIDE* or *OUTSIDE*

`categoryValue` - The category value of the region that triggered the event.

See Also:*MPactClient*

2.9.7 didChangeBeaconType

Method in interface com.zebra.mpact.mpactclient.**MPactClientNotifier**

```
void didChangeBeaconType (MPactBeaconType beaconType)
```

Called whenever t`MPactBeaconType` changes occur in the application or a new configuration downloaded from the MPact Server.

Parameters:

`beaconType` - The new value for the beaconType.

See Also:

MPactBeaconType

2.9.8 didChangeIBeaconUUID

Method in interface com.zebra.mpact.mpactclient.**MPactClientNotifier**

```
void didChangeIBeaconUUID(java.lang.String uuid)
```

Called whenever the iBeacon UUID changes due to either application code or a new configuration downloaded from the MPact Server.

Parameters:

`uuid` - The new value for the iBeaconUUID.

2.9.9 didChangeProximityRange

This method runs whenever the proximity range changes due to either the application code or a new configuration downloaded from the MPact Server.

```
void didChangeProximityRange(MPactProximity proximityRange)
```

Parameters:

proximityRange - The new value for the proximityRange.

See Also:*MPactProximity*

2.10 MPactLogNotifier

An interface for receiving logs from the MPactLogger class. Use this interface to receive debug logging messages from the MPact SDK. The app will only receive messages if the message zone matches the zone set in `MPactLogger.setAppLoggingZone(MPactLogZone, MPactLogLevel)` method and the priority of the message is at least as high as the logging level set for that zone.

```
public interface MPactLogNotifier
```

2.11 MPactLogMessage

Method in interface com.zebra.mpact.mpactclient.**MPactLogNotifier**

```
void MPactLogMessage(MPactLogZone zone,
                     MPactLogLevel level,
                     java.lang.String message)
```

A log message generated by the MPactSDK.

Parameters:

`zone` - The zone where the message originated.

`level` - The severity of the message.

`message` - The log message.

2.12 MPactSDKVersion

```
public class MPactSDKVersion
```

Contains versioning information for the Android version client SDK.

2.12.1 version

Static variable in class in com.zebra.mpact.mpactclient.**MPactSDKVersion**

```
public static final java.lang.String version
```

See Also:

Constant Field Values

2.13 MPactServerInfo

Holds network information relevant to the MPact Server.

```
public class MPactServerInfo
```

2.13.1 getHost

Method in class com.zebra.mpact.mpactclient.**MPactServerInfo**

```
public java.lang.String getHost()
```

Returns:

IP address of the MPact Server

2.13.2 setHost

Method in class com.zebra.mpact.mpactclient.**MPactServerInfo**

```
public void setHost(java.lang.String host)
```

Parameters:

`host` - IP address of the MPact Server

2.13.3 getLoginID

Method in class com.zebra.mpact.mpactclient.**MPactServerInfo**

```
public java.lang.String getLoginID()
```

Returns:

User name requesting access to the MPact Server

2.13.4 setLoginID

Method in class com.zebra.mpact.mpactclient.**MPactServerInfo**

```
public void setLoginID(java.lang.String loginID)
```

Parameters:

`loginID` - User name requesting access to the MPact Server

2.13.5 getPassword

Method in class com.zebra.mpact.mpactclient.**MPactServerInfo**

```
public java.lang.String getPassword()
```

Returns:

login password MPact Server

2.13.6 setPassword

Method in class com.zebra.mpact.mpactclient.**MPactServerInfo**

```
public void setPassword(java.lang.String password)
```

Parameters:

`password` - Login password to the MPact Server

2.13.7 getPort

Method in class com.zebra.mpact.mpactclient.**MPactServerInfo**

```
public int getPort()
```

Returns:

Port number of the MPact Server

2.13.8 setPort

Method in class com.zebra.mpact.mpactclient.**MPactServerInfo**

```
public void setPort(int port)
```

Parameters:

`port` - Port number of the MPact Server

2.13.9 getAuthenticate

Method in class com.zebra.mpact.mpactclient.**MPactServerInfo**

```
public boolean getAuthenticate()
```

Returns:

Whether or not authentication is performed on the server's certificate

2.13.10 setAuthenticate

Method in class com.zebra.mpact.mpactclient.**MPactServerInfo**

```
public void setAuthenticate(boolean authenticate)
```

Zebra recommends setting to true, as setting to false makes this app vulnerable to a Man-In-The-Middle attacks. Set to false when a self-signed certificate is used by the server until an official certificate is requested from a CA.

Parameters:

`authenticate` - Whether or not authentication is initiated on the server's certificate.

2.13.11 getUseHTTPS

Method in class com.zebra.mpact.mpactclient.**MPactServerInfo**

```
public boolean getUseHTTPS()
```

The loginID and password are not encrypted if this property is set to false.

Returns:

Whether or not to use HTTPS to secure communications to the MPact Server.

2.13.12 setUseHTTPS

Method in class com.zebra.mpact.mpactclient.**MPactServerInfo**

```
public boolean getUseHTTPS()
```

The loginID and password are not encrypted if this property is set to false.

Returns:

Whether HTTPS is used to secure communications to the MPact Server.

2.13.13 isOverrideSystemProxy

Method in class com.zebra.mpact.mpactclient.**MPactServerInfo**

```
public boolean isOverrideSystemProxy()
```

Returns:

Whether or not system proxy server settings are used. TRUE - Override the system proxy settings; FALSE - Use the system proxy settings.

2.13.14 setOverrideSystemProxy

Method in class com.zebra.mpact.mpactclient.**MPactServerInfo**

```
public void setOverrideSystemProxy(boolean overrideSystemProxy)
```

Parameters:

`overridSystemProxy` - Set this property to TRUE to use custom settings for the proxy server.

2.13.15 getProxyServerAddress

Method in class com.zebra.mpact.mpactclient.**MPactServerInfo**

```
public java.lang.String getProxyServerAddres()
```

Returns:

The IP address or name of the proxy server. Null indicates a proxy server is not used. This property only takes effect if overrideSystemProxy is set to TRUE.

2.13.16 setProxyServerAddress

Method in class com.zebra.mpact.mpactclient.**MPactServerInfo**

```
public void setProxyServerAddress(java.lang.String proxyServerAddress)
```

Parmeters:

`proxyServerAddress` - The IP address or name of the proxy server. Set to null to disable the use of a proxy server. This property only takes effect if overrideSystemProxy is set to TRUE.

2.13.17 getProxyServerPort

Method in class com.zebra.mpact.mpactclient.**MPactServerInfo**

```
public int getProxyServerPort()
```

Returns:

The IP address or name of the proxy server. Null indicates a proxy server is not used. This property only takes effect if overrideSystemProxy is set to TRUE.

2.13.18 setProxyServerPort

Method in class com.zebra.mpact.mpactclient.**MPactServerInfo**

```
public void setProxyServerPort(int proxyServerPort)
```

Parmeters:

`proxyServerPort` - The port number of the proxy server. This property only takes effect if overrideSystemProxy is set to TRUE.

2.14 MPactTag

Holds properties describing a MPact beacon.

```
public class MPactTag
```

2.14.1 getTagID

Method in class com.zebra.mpact.mpactclient.**MPactTag**

```
public java.lang.String getTagID()
```

Returns:

The ID of the tag

2.14.2 setTagID

Method in class com.zebra.mpact.mpactclient.**MPactTag**

```
public void setTagID(java.lang.String tagID)
```

Parameters:

tagID - The ID of the tag

2.14.3 getBatteryLife

Method in class com.zebra.mpact.mpactclient.**MPactTag**

```
public int getBatteryLife()
```

Returns:

Estimated remaining battery life in percentage (0-100). The batteryLife is set to -1 when the beaconType is set to MPactBeaconType.iBeacon.

2.14.4 setBatteryLife

Method in class com.zebra.mpact.mpactclient.**MPactTag**

```
public void setBatteryLife(int batteryLife)
```

Parameters:

`batteryLife` - Estimated remaining battery life in percentage (0-100).

2.14.5 getRssi

Method in class com.zebra.mpact.mpactclient.**MPactTag**

```
public int getRssi()
```

Returns:

RSSI of the beacon

2.14.6 setRssi

Method in class com.zebra.mpact.mpactclient.**MPactTag**

```
public void setRssi(int rssi)
```

Parameters:

`rssi` - The RSSI of the beacon.

2.14.7 getManufacturingID

Method in class com.zebra.mpact.mpactclient.**MPactTag**

```
public java.lang.String getManufacturingID()
```

Returns:

The beacon MAC ID stored in the MPact Server during installation. This property is dependent on information retrieved from the MPact Server. null is returned if the client cannot establish a connection with the MPact Server.

2.14.8 setManufacturingID

Method in class com.zebra.mpact.mpactclient.**MPactTag**

```
public void setManufacturingID(java.lang.String manufacturingID)
```

Parameters:

`manufacturingID` - The beacon MAC stored in the MPact Server during installation.

2.14.9 getCategories

Method in class com.zebra.mpact.mpactclient.**MPactTag**

```
public java.util.Set<MPactCategoryValue> getCategories()
```

Returns:

Retrieved from the MPact Server. Set to null if the client is notable to establish a connection with the MPact Server, or if no categories have been assigned to this beacon.

2.14.10 setCatergories

Method in class com.zebra.mpact.mpactclient.**MPactTag**

```
public void setCategories(java.util.Set<MPactCategoryValue> categories)
```

Parameters:

categories - The categories assigned to this beacon stored in the MPact Server during beacon installation.

2.15 MPactBeaconType

Enum in com.zebra.mpact.mpactclient

```
public enum MPactBeaconType
```

Format of the beacon. Formats include:

`BatterySave`

BatterySave custom format. Battery life is longer than iBeacon.

`iBeacon`

Standard iBeacon format. Major and Minor are combined to form the ID. Battery life is not reported.

`MPact`

iBeacon format is used, however, the Major and Minor fields have specific meanings. Battery life is included in the LSB of the Minor field.

`SecureCast`

SecureCast custom form created by Swirl. Over the air information is encrypted.

2.15.1 valueOf

Static method in enum com.zebra.mpact.mpactclient.**MPactBeaconType**

```
public static MPactBeaconType valueOf(java.lang.String name)
```

Returns the enum constant with the specified name. The string must match exactly an identifier used to declare an enum constant in this type.

Parameters:

Name - The name of the enum constant returned.

Returns:

The enum constant with the specified name

Throws:

```
java.lang.IllegalArgumentException - if this enum type has no constant with
the specified name
```

```
java.lang.NullPointerException - if the argument is null
```

2.15.2 values

Static method in enum com.zebra.mpact.mpactclient.**MPactBeaconType**

```
public static MPactBeaconType[] values()
```

Returns an array containing the constants of this enum type, in the order declared. This method can be used to iterate over the constants as follows:

For `(MPactBeaconType c : MPactBeaconType.values())`

```
   System.out.println(c);
```

Returns:

An array containing the constants of this enum type in the order declared

2.15.3 getValue

Method in enum com.zebra.mpact.mpactclient.**MPactBeaconType**

```
public int getValue()
```

2.16 MPactLogLevel

Enum in com.zebra.mpact.mpactclient

```
public enum MPactLogLevel
```

```
extends java.lang.Enum<MPactLogLevel>
```

Debug logging Levels for the MPact SDK. The *MPactLogZone* combined with the MPactLogLevel determine the logging messages generated.

See Also:

MPactLogger

2.16.1 MPactLogLevelCritical

```
public static final MPactLogLevel MPactLogLevelCritical
```

Highest level. Cannot be filtered.

2.16.2 MPactLogLevelError

```
public static final MPactLogLevel MPactLogLevelError
```

The system did not function properly.

2.16.3 MPactLogLevelWarning

```
public static final MPactLogLevel MPactLogLevelWarning
```

An unexpected event occurred with the potential of an unintended side effect.

2.16.4 MPactLogLevelInfo

```
public static final MPactLogLevel MPactLogLevelInfo
```

Informational messages.

2.16.5 MPactLogLevelDebug

```
public static final MPactLogLevel MPactLogLevelDebug
```

Low level debug messages.

2.16.6 MPactLogLevelTrace

```
public static final MPactLogLevel MPactLogLevelTrace
```

Detailed and high volume logging.

2.16.7 MPactLogLevelInvalid

```
public static final MPactLogLevel MPactLogLevelInvalid
```

Not a valid logging level.

2.16.8 valueOf

Static method com.zebra.mpact.mpactclient.**MPactLogLevel**

```
public static MPactLogLevel valueOf(java.lang.String name)
```

Returns the enum constant of this type with the specified name. The string must match exactly an identifier used to declare an enum constant in this type (extraneous whitespace characters are not permitted).

Parameters:

Name - The name of the enum constant returned.

Returns:

The enum constant with the specified name

Throws:

```
java.lang.IllegalArgumentException - if this enum type has no constant with
the specified name
```

```
java.lang.NullPointerException - if the argument is null
```

2.16.9 values

Static method com.zebra.mpact.mpactclient.**MPactLogLevel**

```
public static MPactLogLevel[] values()
```

Returns an array containing the constants of this enum type in the order declared. This method can be used to iterate over the constants as follows:

```
for (MPactLogLevel c : MPactLogLevel.values())
    System.out.println(c);
```

Returns:

The enum constant with the specified name.

2.17 MPactLogZone

Enum in com.zebra.mpact.mpactclient

```
public enum MPactLogZone
extends java.lang.Enum<MPactZone>
```

Available debug logging zones for the MPact SDK. The *MPactLogZone* combined with the *MPactLogLevel* determine the logging messages generated.

See Also:

MPactLogger

2.17.1 MPactLogZoneAll

```
public static final MPactLogZone MPactLogZoneAll
```

Special enum for applying actions to all debug zones.

2.17.2 MPactLogZoneGeneral

```
public static final MPactLogZone MPactLogZoneGeneral
```

General logs about the SDK.

2.17.3 MPactLogZoneBeacon

```
public static final MPactLogZone MPactLogZoneBeacon
```

Logs specific to beacon tracking.

2.17.4 MPactLogZoneWinner

```
public static final MPactLogZone MPactLogZoneWinner
```

Logs specific to determining the closest beacon.

2.17.5 MPactLogZoneServer

```
public static final MPactLogZone MPactLogZoneServer
```

Logs specific to communications with the MPact Server.

2.17.6 MPactLogZoneInvalid

```
public static final MPactLogLevel MPactLogZoneInvalid
```

Invalid Zone.

2.17.7 values

Static method com.zebra.mpact.mpactclient.**MPactLogLevel**

```
public static MPactLogZone[] values()
```

Returns an array containing the constants of this enum type, in the order declared. This method is used to iterate over the constants as follows:

```
for (MPactLogZone c : MPactLogZone.values())
    System.out.println(c);
```

Returns:

An array containing the constants of this enum type in the order declared.

2.17.8 valueOf

Static method com.zebra.mpact.mpactclient.**MPactLogLevel**

```
public static MPactLogZone valueOf(java.lang.String name)
```

Returns the enum constant of this type with the specified name. The string must match an identifier used to declare an enum constant in this type (extraneous whitespace characters not permitted).

Parameters:

Name - The name of the enum constant returned.

Returns:

The enum constant with the specified name

Throws:

```
java.lang.IllegalArgumentException - if this enum type has no constant with
the specified name
```

```
java.lang.NullPointerException - if the argument is null
```

2.18 MPactProximity

Enum in com.zebra.mpact.mpactclient

```
public enum MPactProximity
```

Format of the beacon. Formats include:

`Far`

100 feet

`Immediate`

3 feet

`Near`

50 feet.

`Invalid`

Not a valid proximity.

NOTE: Distance is a function of measured RSSI, which can change based on environment and deployment scenarios. Therefore, the distances noted above can change.

2.18.1 values

Static method com.zebra.mpact.mpactclient.**MPactProximity**

```
public static MPactProximity[] values()
```

Returns an array containing the constants of this enum type, in the order declared. This method is used to iterate over the constants as follows:

```
for (MPactProximity c : MPactProximity.values())
    System.out.println(c);
```

Returns:

An array containing the constants of this enum type, in the order they are declared.

2.18.2 valueOf

Static method in enum com.zebra.mpact.mpactclient.**MPactProximity**

```
public static MPactProximity valueOf(java.lang.String name)
```

Returns the enum constant of this type with the specified name. The string must match exactly an identifier used to declare an enum constant in this type. (extraneous whitespace characters are not permitted).

Parameters:

Name - The name of the enum constant returned.

Returns:

The enum constant with the specified name.

Throws:

```
java.lang.IllegalArgumentException - if this enum type has no constant with
the specified name
```

```
java.lang.NullPointerException - if the argument is null.
```

2.18.3 getValue

Method in enum com.zebra.mpact.mpactclient.**MPactProximity**

```
public int getValue()
```

CHAPTER 3 IOS API

MPact Client API allows a third party application to scan for MPact BlueTooth Low Energy beacons and send the information about the closest beacon to the MPact Server. The MPact Client API supports an Objective-C interface and should be integrated into a partner application.

The MPact Client API is distributed as part of a framework named *MPactClient*

The MPact Client library makes uses CoreBluetooth and CoreLocation frameworks. A third party application needs to add the CoreBluetooth framework and the CoreLocation.framework libraries to your build settings.

This library uses C++ templates. Add "-lstdc++" to "Other Linker Flags" link options in your project's build settings.

Drag and drop this framework to access the files and objects comprising the MPact Client API.

Class References

- *MPactClient*
- *MPactClient*
- *MPactServerInfo*
- *MPactTag*
- *MPactLogger*

Protocol Reference

- *MPactClientDelegate*

Constant Reference

- *MPactBeaconType*
- *MPactLogZone*
- *MPactLogLevel*
- *MPactBackgroundPolicy*

3.1 MPactCatergoryValue

This class holds the properties that describe the state of an MPact Category Value. Categories and Values are defined on the MPact Server and are assigned to individual beacons.

See *MPactClient*.

3.1.1 Properties

3.1.1.1 category

The category as defined by the MPact Server.

```
@property (nonatomic) NSString *category
```

Declared In:

```
MPactCategoryValue.h
```

3.1.1.2 value

The value associated with the category as defined by the MPact Server.

```
@property (nonatomic) NSString *value
```

Declared In:

```
MPactCategoryValue.hvalue
```

3.2 MPactClient

Sends MPact position updates to the MPact Server. Partner applications should integrate this interface into their customer applications.

3.2.1 Properties

3.2.1.1 beaconType

The location format used by MPact beacons. For more information, see *MPactClientDelegate*.

```
@property (nonatomic, assign) MPactBeaconType beaconType
```

Declared In:

```
MPactClient.h
```

3.2.1.2 clientId

The identity of this device. The clientId is sent to the MPact Server with the position updates.

```
@property (nonatomic, readonly) NSString *clientId
```

Discussion

This is a UUID generated when the third party application is installed and cannot be changed.

Declared In:

```
MPactClient.h
```

3.2.1.3 clientName

Human readable name for the device. The client name is sent to the MPact Server with the position updates.

```
@property (nonatomic) NSString *clientName
```

Declared In:

```
MPactClient.h
```

3.2.1.4 closestTag

The attributes of the closest tag.

```
@property (nonatomic, readonly) MPactTag *closestTag
```

Discussion

The property is nil when there is no closest beacon identified. The battery life is set to -1 when the beaconType is set to MPactBeaconTypeiBeacon.

Declared In:

```
MPactClient.h
```

3.2.1.5 delegate

Delegate for MPactClient.

```
@property (nonatomic, weak) id<MPactClientDelegate> delegate
```

3.2.1.6 iBeaconUUID

The UUID of the iBeacon tags. This field is not used when *beaconType* is set to MPactBeaconTypeBatterySave.

```
@property (nonatomic) NSUUID *iBeaconUUID
```

Declared In:

```
MPactClient.h
```

NOTE: A universally unique identifier (UUID) is an identifier standard used in software construction. A UUID is simply a 128-bit value. The meaning of each bit is defined by any of several variants. For human-readable display, many systems use a canonical format using hexadecimal text with inserted hyphen characters. For example: de305d54-75b4-431b-adb2-eb6b9e546014

3.2.1.7 server

Network information about the MPact Server such as server IP address, port, user name, password, protocol, authenticate, and proxy info, etc.

```
@property MPactServerInfo *server
```

Declared In:

```
MPactClient.h
```

3.2.1.8 proximityRange

The maximum distance considered for beacons.

```
@property (nonatomic) CLPProxity proximityrange
```

Declared In:

```
MPactClient.h
```

3.2.1.9 backgroundTime

The time in seconds the app should continues to request client updates to the server when processing an event in the background.

```
@property (nonatomic) NSTimeInterval backgroundTime
```

Discussion

This value is only relevant when the backgroundPolicy is set to MPactBackgroundPolicyTimed.

Declared In

```
MPactClient.h
```

3.2.1.10 backgroundPolicy

Execution policy used by the SDK when the user is not actively using your app.

```
@property (nonatomic) MPactBackgroundPolicy backgroundPolicy
```

Discussion

These policies only apply when the beaconType is set to either MPactBeaconTypeiBeacon or MPactBeaconTypeMPact.

Declared In

`MPactClient.h`

3.2.2 Class Methods

3.2.2.1 initClient

Factory function for creating the MPactClient object.

```
+ (id)initClient
```

Return Value

The MPactClient object

Discussion

Function required to create the MPactClient object. Directly calling alloc and init in the MPactClient class results in a non-functional object.

Declared In:

`MPactClient.h`

3.2.2.2 version

Version of the MPact API

```
+ (NSString *)version
```

Declared In:

`MPactClient.h`

3.2.3 Instance Methods

3.2.3.1 start

Starts scanning for MPact beacons and sends updates to the *server*.

```
- (void)Start
```

Return Value

void

Declared In

`MPactClient.h`

3.2.3.2 stop

Stops scanning for MPact beacons and stops the updates to the *server*.

```
- (void)Stop
```

Return Value

void

Declared In

`MPactClient.h`

3.3 MPactServerInfo

Holds network information relevant to the *server*. For more information, see *MPactClient*.

3.3.1 Properties

3.3.1.1 authenticate

Whether or not authentication is performed on the MPact *server*'s certificate

```
@property BOOL authenticate
```

Discussion

Set to TRUE, as leaving this property in the default setting of FALSE makes the application vulnerable to a Man-In-The-Middle attack. It is useful to set this value to FALSE during development so a self-signed certificate is used by the *server* until an official certificate is requested from CA.

Declared In:

```
MPactServerInfo.h
```

3.3.1.2 host

IP address of MPact Server

```
@property NSString *host
```

Declared In:

```
MPactServerInfo.h
```

3.3.1.3 loginID

User name required to access the *server*

```
@property NSString *loginID
```

Declared In:

```
MPactServerInfo.h
```

3.3.1.4 password

MPact Server's login password

```
@property NSString *password
```

Declared In:

```
MPactServerInfo.h
```

3.3.1.5 port

MPact Server's connection port

```
@property short port
```

Declared In:

```
MPactServerInfo.h
```

3.3.1.6 useHTTPS

Determines whether HTTPS is used to secure communications to the *server*.

```
@property BOOL useHTTPS
```

Discussion

The *loginID* and *password* are not encrypted if this property is set to FALSE.

Declared In:

```
MPactServerInfo.h
```

3.3.1.7 overrideSystemProxy

Set this property to TRUE to use custom settings for the proxy server.

```
@property BOOL overrideSystemProxy
```

Declared In:

```
MPactServerInfo.h
```

3.3.1.8 proxyServerAddress

The IP address or name of the proxy server.

```
@property BOOL proxyServerAddress
```

Discussion

Set this property to nil to disable the proxy server. This property only takes effect if overrideSystemProxy is set to TRUE.

Discussion

```
MPactServerInfo.h
```

3.3.1.9 proxyServerPort

The port number of the proxy server.

```
@property BOOL proxyServerPort
```

Discussion

This property only takes effect if overrideSystemProxy is set to TRUE.

Discussion

```
MPactServerInfo.h
```

3.3.2 Instance Methods

3.3.2.1 initWithNetwork:AndPort:

Convenience method for creating an MPactServerInfo object

```
- (id)initWithNetwork:(NSString *)host AndPort:(unsigned short)port
```

Paramters

host -IP address of the MPact Server

port - Port number of the MPact Server.

Return Value

An MPactServerInfo object initialized with the input *host* and *port*

Declared In:

```
MPactServerInfo.h
```

3.4 MPactTag

Holds the properties describing the state of an MPact beacon. For more information, see *MPactClient*.

3.4.1 Properties

3.4.1.1 batteryLife

Estimated percentage of remaining life in the beacon's battery.

```
@property NSInteger batteryLife
```

Discussion

The batteryLife is set to -1 when the beaconType is set to MPactBeaconTypeiBeacon.

3.4.2 categories

The categories assigned to this beacon.

```
@property NSSet *categories
```

Discussion

This property is dependent upon information retrieved from the MPact Server. The set is filled in with a MPactCategoryValue object for every Category-Value pair assigned. This property is set to nil if the client cannot establish a connection with the MPact Server, or if no categories are assigned to this beacon.

Declared In

`MPactTag.h`

3.4.2.1 rssi

RSSI of the closest beacon

```
@property NSInteger rssi
```

Declared In:

`MPactTag.h`

3.4.2.2 manufacturingID

The beacon MAC ID stored in the MPact Server during installation.

```
@property NSInteger manufacturingID
```

Discussion

This property is dependent on information retrieved from the MPact Server. This property is set to nil if the client cannot establish a connection with the MPact Server.

Declared In:

`MPactTag.h`

3.4.2.3 tagID

The ID of the closest beacon.

```
@property NSString *tagID
```

Declared In:

`MPactTag.h`

3.5 MPactLogger

Enables the generation of debug log messages by the MPact SDK.

3.5.1 Properties

3.5.1.1 delegate

Delegate for MPactLogger

```
@property (nonatomic, weak) id<MPactLoggerDelegate> delegate
```

Discussion

See also *MPactLoggerDelegate*.

Declared In:

`MPactLogger.h`

3.5.2 Class Methods

3.5.2.1 sharedInstance

The MPactLogger object.

```
+ (MPactLogger *)sharedInstance
```

Discussions

Use this property to get an instance of the MPactLogger. Do not use alloc or init methods.

Declared In:

`MPactLogger.h`

3.5.3 Instance Methods

3.5.3.1 getAppLoggingZone:

Gets the *MPactLoggerDelegate* logging level for the specified zone.

```
- (MPactLogLevel)getAppLoggingZone:(MPactLogZone)zone
```

Paramters

zone - The logging zone to change. See *Declared In* `MPactLogger.h`.

Return Value

The current logging level for the requested zone. See *MPactLogZone*.

Declared In:

`MPactLogger.h`

3.5.3.2 getConsoleLoggingZone:

Get the system console logging level for the specified zone.

```
- (MPactLogLevel)getConsoleLoggingZone:(MPactLogZone)zone
```

Paramters

zone - The logging zone to change. See *MPactLogZone*.

Return Value

The current logging level for the requested zone. See *MPactLogZone*.

Declared In:

`MPactLogger.h`

3.5.3.3 setAppLoggingZone:Level

Sets the zones and levels used for logging to the *MPactLoggerDelegate*.

```
- (void)setAppLoggingZone:(MPactLogZone)zone Level:(MPactLogLevel)level
```

Paramters

zone - The logging zone to change. See *Declared In* `MPactLogger.h`.

level - The logging level requested for the zone. See *MPactLogZone*.

Discussion

Messages can only be received by the app if the *MPactLoggerDelegate* is implemented.

Declared In:

`MPactLogger.h`

3.5.3.4 setConsoleLoggingZone:Level

Sets the zones and levels used for logging to the system console.

```
- (void)setConsoleLoggingZone:(MPactLogZone)zone Level:(MPactLogLevel)level
```

Paramters

zone - The logging zone to change. See *Declared In* `MPactLogger.h`..

level - The logging level requested for the zone. See *MPactLogZone*.

Discussion

Messages can only be received by the app if the *MPactLoggerDelegate* is implemented.

Declared In:

`MPactLogger.h`

3.6 MPactClientDelegate

Optional delegate methods for the MPact Client. Use these methods to receive extra information from the MPact Client.

3.6.1 MPactClient:ClosestTag:

@method *MPactClient*:ClosestTag:

```
- (void)MPactClient:(id)client ClosestTag:(MPactTag *)tag
```

Parameters

client - The object sending the message

tag - The closest beacon.

Discussion

This method is called at periodic intervals with the identity of the closest beacon. The method stops when there is nothing identified. For more information, see *MPactTag*.

Declared In

`MPactClient.h`

3.6.2 MPactClient:didChangeBeaconType:

@method *MPactClient*:didChangeBeaconType

```
- (void)MPactClient:(id)client
didChangeBeaconType:(MPactBeaconType)beaconType
```

Parameters

client - The object sending the message

beaconType - The new value for the beaconType. For a list of possible values, see *MPactBeaconType*.

Discussion

This method is called whenever the beacon type changes due to either the code in the third party application or a new configuration downloaded from the MPact Server.

Declared In

`MPactClient.h`

3.6.3 MPactClient:didChangeIBeaconUUID:

@method *MPactClient*:didChangeIBeaconUUID:

```
- (void)MPactClient:(id)client didChangeIBeaconUUID:(NSUUID *)iBeaconUUID
```

Parameters

client - The object sending the message

iBeaconUUID - The new value for the iBeaconUUID.

Discussion

This method is called whenever the iBeacon UUID changes due to either the code in the third party application or a new configuration downloaded from the MPact Server.

Declared In

`MPactClient.h`

3.6.4 MPactClient:didDetermineState:

@method *MPactClient*:didDetermineState:.

`-(void)MPactClient:(id)client didDetermineState:(CLRegionState)state`

Parameters

client - The object sending the message

state - This method is invoked whenever the third party application crosses a border transition of an iBeacon region. Even applications in the background become active for a few seconds when a transition occurs.

Discussion

This method is active only if the *MPactClient* property beaconType is set to either MPactBeaconTypeiBeacon or MPactBeaconTypeMPact.

Declared In

`MPactClient.h`

3.6.5 MPactClient:didDetermineState:CategoryValue:

`@method MPactClient:didDetermineState:CategoryValue:`

```
- (void)MPactClient:(id)client didDetermineState:(CLRegionState)state
CategoryValue:(MPactCategoryValue *)categoryValue
```

Parameters

client

The object sending the message.

state

This method is invoked whenever the application crosses a border transition of a Category-Value region. Even applications in the background become active for a few seconds when a transition occurs (except when using BatterySave mode). For a list of possible values, see the CLRegionState type.

categoryValue

The category value of the region triggering the event.

Discussion

Enables the Background Mode of “Uses Bluetooth LE Accessories” needs to be enabled when beaconType is set to MPactBeaconTypeSecureCast to receive notifications in the background.

Declared In

`MPactClient.h`

3.6.6 MPactClient:didDetermineState:Major:Minor:

@method *MPactClient*:didDetermineState:.

```
- (void)MPactClient:(id)client didDetermineState:(CLRegionState)state
Major:(NSNumber *)major Minor:(NSNumber *)minor
```

Parameters

client - The object sending the message

state - This method runs whenever the third party application crosses a border transition of an iBeacon region. Even applications in the background become active for a few seconds when a transition occurs.

`Minor` - The minor value of the region triggering the event. This value is set to nil if the region is not specific to a minor value.

`Major` - The major value of the region triggering the event. This value is set to nil if the region is not specific to a major value.

Discussion

This method is active only if the *MPactClient* property beaconType is set to either MPactBeaconTypeiBeacon or MPactBeaconTypeMPact. A value of nil for both the Major and Minor indicates that the notification is for the region defined by the iBeaconUUID.

Declared In

`MPactClient.h`

3.7 MPactLoggerDelegate

MPactLoggerDelegate methods receive logs from the MPactLogger. Use these methods to receive debug logging messages from the MPact SDK. The delegate only receives messages if the setAppLoggingZone method of the MPactLogger class is set to a high enough level for one of the logging zones.

3.7.1 Instance Methods

3.7.1.1 MPactLogger:Zone:Level:LogMessage:

```
@method MPactLogger:LogMessage- (void)MPactLogger:(id)logger
Zone:(MPactLogZone)zone Level:(MPactLogLevel)level LogMessage:(NSString
*)message
```

Parameters

logger - The object sending the message.

zone - The zone where the message originated.

level - The severity of the message.

message - The log message.

Discussion

Low level debugging messages from the MPact library.

Declared In

`MPactClient.h`

3.8 MPactBeaconType

Format of the beacon.

Definition:

```
typedef NS_ENUM(NSInteger, MPactBeaconType ) {
  MPactBeaconTypeBatterySave = 1,
  MPactBeaconTypeiBeacon = 2,
  MPactBeaconTypeMPact = 3,
};
```

Constants

`MPactBeaconTypeBatterySave`

MPact custom format. Battery life of the beacon is better than iBeacon.

Declared In `MPactClient.h`.

`MPactBeaconTypeiBeacon`

Standard iBeacon format. Major and Minor are combined to form the tag ID. Battery life is not reported.

Declared In `MPactClient.h`.

`MPactBeaconTypeMPact`

iBeacon format is used, however, the Major and Minor fields have specific meanings. Battery life is included in the Minor field.

Declared In `MPactClient.h`.

Definition:

```
typedef NS_ENUM(NSInteger, MPactBeaconType ) {
  MPactBeaconTypeBatterySave = 1,
  MPactBeaconTypeiBeacon = 2,
  MPactBeaconTypeMPact = 3,
};
```

3.9 MPactLogZone

Available debug logging zones for the MPact SDK. MPactLogZone is combined with MPactLogLevel to determine the logging messages generated.

Definition:

typedef NS_ENUM(NSInteger, MPactLogZone) {

MPactLogZoneAll,

MPactLogZoneGeneral,

MPactLogZoneBeacon,

MPactLogZoneWinner,

MPactLogZoneServer,

MPactLogZoneInvalid,

};

Constants

`MPactLogZoneAll`

Special enum for applying actions to all debug zones

Declared In `MPactLogger.h`.

`MPactLogZoneGeneral`

General logs about the SDK

Declared In `MPactLogger.h`.

`MPactLogZoneBeacon`

Logs specific to beacon tracking

Declared In `MPactLogger.h`.

`MPactLogZoneWinner`

Logs specific to determining the closest beacon

Declared In `MPactLogger.h`.

`MPactLogZoneServer`

Logs specific to communications with the MPact Server

Declared In `MPactLogger.h`.

`MPactLogZoneInvalid`

Invalid Zone

Declared In `MPactLogger.h`.

3.10 MPactLogLevel

Debug logging Levels for the MPact SDK. MPactLogZone is combined with MPactLogLevel to determine the logging messages generated.

Definition:

```
ttypedef NS_ENUM(NSInteger, MPactLogLevel ) {
   MPactLogLevelCritical,
   MPactLogLevelError,
   MPactLogLevelWarning,
   MPactLogLevelInfo,
   MPactLogLevelDebug,
   MPactLogLevelTrace,
   MPactLogLevelInvalid,
};
```

Constants

`MPactLogLevelCritical`

Highest Level. Can't be filtered

Declared In `MPactLogger.h`.

`MPactLogLevelError`

The system did not function properly.

Declared In `MPactLogger.h`.

`MPactLogLevelWarning`

An unexpected event occurred with the potential of an unintended side effect.

Declared In `MPactLogger.h`.

`MPactLogLevelInfo`

Informational messages

Declared In `MPactLogger.h`

`MPactLogLevelDebug`

Low level debug messages

Declared In `MPactLogger.h`.

`MPactLogLevelTrace`

Extremely detailed and potentially high volume logging.

Declared In `MPactLogger.h`.

`MPactLogLevelInvalid`

Not a valid logging level.

Declared In `MPactLogger.h`.

3.11 MPactBackgroundPolicy

Execution policy used by the SDK when the user is not actively using your app. iOS sends a notification for transitions between iBeacon Regions, even if your app is no longer in the foreground. The app is assigned an interval to process these notifications, and there is the opportunity to request extra time for processing. One possible reason for requesting extra time would be to allow the MPact SDK to calculate the closest beacon and send an update to the MPact Server.

These policies only apply when the MPactBeaconType is set to either MPactBeaconTypeiBeacon or MPactBeaconTypeMPact.

Definition

```
typedef NS_ENUM(NSUInteger, MPactBackgroundPolicy ) {
   MPactBackgroundPolicySystem,
   MPactBackgroundPolicyKeepRunning,
   MPactBackgroundPolicyTimed,
};
```

Constants

`MPactBackgroundPolicySystem`

Use the background policy of the operating system. On iOS 8 the app may run for an additional 10 seconds.

Declared In `MPactClient.h`.

`MPactBackgroundPolicyKeepRunning`

The SDK attempts to keep the app running for as long as the OS allows. On iOS 8 the app runs for up to 3 additional minutes.

Declared In `MPactClient.h.`

`MPactBackgroundPolicyTimed`

The SDK attempts to keep the app running for the time specified in the backgroundTime field. This setting is useful if the app developer wants the app running for less than the maximum.

Declared In MPactClient.h.

CHAPTER 4 WINDOWS CE API

The MPact Windows CE Client API library is used to develop third party applications for Windows CE devices. The CE API supports C++ and allows applications to scan for MPact BlueTooth Low Energy beacons and send information about the closest beacon initializing to the MPact Server.

4.1 Function Summary

MPACT_InitLibrary	Initializes the MPact WinCE ClientSDK library and specifies the callback function.
MPACT_DeinitLibrary	De-initializes the MPact WinCE ClientSDK library and stops the processing of any BLE beacons. All the callback functions are immediately stopped.
MPACT_BEACONCALLBACK	The callback function from the MPact driver. It returns beacon information to the third party application. It must be registered with the MPact driver during `MPACT_InitLibrary`.
MPACT_StartScanning	Starts BLE beacon scanning. The beacon is reported through the `BEACONCALLBACK` function.
MPACT_StopScanning	Stops BLE beacon scanning. There is no report generated after scanning is stopped.
MPACT_SetBeaconMode	Allows a third party application to pass in Beacon Mode, – which identifies beacon payloads. This ensures the MPact ClientSDK correctly process beacon callbacks for the appropriate beacons.
MPACT_GetBeaconMode	Returns the `BeaconMode` registered in the MPact ClientSDK for beacon processing. This function is only for third party applications.
MPACT_SetReportMode	Allows customer applications to pass in the Report Mode, – which includes three different reporting behaviors. If the API is called after scanning is started, it takes immediate effect.
MPACT_GetReportMode	Returns the ReportMode registered in the MPact ClientSDK for beacon reporting. This function is only for third party application information purposes.
MPACT_SetParameter	Allows customer applications to set parameters. This function is optional.
MPACT_GetParameter	Returns the parameter value registered in MPact ClientSDK. This function is only for third party applications.
MPACT_SetBeaconUUID	Allows third party customer applications to pass in the UUID of the registered beacons. This ensures the MPact ClientSDK is processing only the beacons with correct UUIDs, and triggers their callbacks.
MPACT_GetBeaconUUID	Returns the UUID registered for beacon processing.
MPACT_GetClosestBeacon	Allows the third party application to actively fetch the closest beacon from the MPact ClientSDK.
MPACT_SetServerInfo	Allows applications to configure server information to MPact ClientSDK. Then the ClientSDK can exchange information with the server.
MPACT_GetServerInfo	Allows third party applications set server information in the MPact ClientSDK.
MPACT_GetLibVersion	Returns the version number of the MPact ClientSDK library.
MPACT_ConfigureBeacon	Starts the configuration for the list of beacons with specific parameter settings.
MPACT_UpgradeBeacon	Starts the upgrade process for beacons with a specific firmware image.
MPACT_RebootBeacon	Reboots the list of beacons into broadcasting mode.

4.1.1 Enum Summary

TagBeaconMode	The mode defining how signals are emitted from MPact beacons.
TagReportMode	The modes defining the beacon reports generated via MPact or whether no report is generated.
TagParmeter	The optional parameters third party application can configure.
TagProtocol	Defines which protocol is used by MPact beacons.
TagBeaconState	Defines the MPact beacon state as either: UnknownState, Discovered, Connected, Disconnected, ServiceFound, ParameterRead, Configuring, Upgrading, Rebooting, Configured, Upgraded, or Rebooted.

4.2 Sample Code

The following example captures how a customer uses the MPact ClientSDK to leverage BLE micro-locationing.

```
#define BEACON_ID_LEN 8
#define MAC_LEN 12

#define SERVER_IP_LEN 100
#define USERNAME_LEN 30
#define PASSWORD_LEN 30
#define NICKNAME_LEN 30

#define LIB_VERSION_LEN 30

#define ASSIGN_UUID(_dest, _a, _b, _c, _d, _e, _f, _g, _h, _i, _j, _k, _l,
_m, _n, _o, _p) \
{
\
   (_dest).UUID_Byte0  = (_a); (_dest).UUID_Byte1  = (_b); (_dest).UUID_Byte2
= (_c);             \
   (_dest).UUID_Byte3  = (_d); (_dest).UUID_Byte4  = (_e); (_dest).UUID_Byte5
= (_f);             \
   (_dest).UUID_Byte6  = (_g); (_dest).UUID_Byte7  = (_h); (_dest).UUID_Byte8
= (_i);             \
    (_dest).UUID_Byte9  = (_j); (_dest).UUID_Byte10 = (_k);
(_dest).UUID_Byte11 = (_l);             \
    (_dest).UUID_Byte12 = (_m); (_dest).UUID_Byte13 = (_n);
(_dest).UUID_Byte14 = (_o);             \
   (_dest).UUID_Byte15 = (_p);
\
}

typedef struct
{
   STRUCT_INFO StructInfo;
   char cBeaconID[BEACON_ID_LEN+1];
   char cMAC[MAC_LEN+1];
   int nRSSI;
   int nBatteryLife;
}BEACON_INFO_t;
```

```
void __stdcall GetNotify(BEACON_INFO_t* pBeaconInfo)
{
       char buffer[512];
       sprintf(buffer,"GetNotify, ID %s, MAC %s, Battery %d%%, RSSI
        %d\n",pBeaconInfo->cBeaconID,pBeaconInfo->cMAC,pBeaconInfo-
       >nBatteryLife,pBeaconInfo->nRSSI);
}

void Start_MPact()
{
       MPACT_RETURN_VALUE ret = MPACT_ERROR_FAILED;
       UUID_BEACON_t uuid;

       ret = MPACT_InitLibrary(GetNotify);

       if(ret == MPACT_SUCCESS)
       {
           ret = MPACT_SetBeaconMode(MPact);
           if(ret != MPACT_SUCCESS)
              goto Error;

          // Set beacon UUID, default in Mpact library is all 0s
          // User can skip this if using Battery Save mode
          memset(&uuid,0,sizeof(uuid));
          uuid.StructInfo.dwAllocated = sizeof(uuid);
          uuid.StructInfo.dwUsed = sizeof(uuid);

          ASSIGN_UUID(uuid,0xFE,0x91,0x32,0x13,0xB3,0x11,0x4A,0x42,0x8C,0x1
          6,0x47,0xFA,0xEA,0xC9,0x38,0xDB);
          ret = MPACT_SetBeaconUUID(&uuid);
          if(ret != MPACT_SUCCESS)
             goto Error;

          // Set report mode, default in Mpact library is ClosestBeacon mode
          ret = MPACT_SetReportMode(ClosestBeacon);
          if(ret != MPACT_SUCCESS)
            goto Error;
```

```
            // Optional: Set server info to work with MPact server
            SERVER_INFO_t serverInfo;

            memset(&serverInfo,0,sizeof(serverInfo));
            serverInfo.StructInfo.dwAllocated = sizeof(serverInfo);
            serverInfo.StructInfo.dwUsed = sizeof(serverInfo);

            strncpy(serverInfo.cServerIP, "server ip or domain",
SERVER_IP_LEN);
            strncpy(serverInfo.cNickname, "Sam", NICKNAME_LEN);
            strncpy(serverInfo.cUsername, "user1", USERNAME_LEN);
            strncpy(serverInfo.cPassword, "MyPassword ", PASSWORD_LEN);

            serverInfo.Protocol = HTTP;
            serverInfo.nPort = 80;

            ret = MPACT_SetServerInfo(&serverInfo);
            if(ret != MPACT_SUCCESS)
               goto Error;

            ret = MPACT_StartScanning();
            if(ret != MPACT_SUCCESS)
               goto Error;
   }

   return;
   Error:
   MPACT_DeinitLibrary();
}
```

4.3 Return Values

```
typedef enum tagMPACT_RETURN_VALUE {
  MPACT_SUCCESS
= 0,/**< Function completed successfully */
  MPACT_ERROR_IN_INITIALIZATION,   /**< Error happens during BLE stack initilization
*/
   MPACT_ERROR_NOT_INITIALIZED,    /**< MPact Driver is not initialized */
   MPACT_ERROR_ALREADY_INITIALIZED,    /**< MPact Driver is already initialized */
  MPACT_ERROR_IN_START_SCANNING,                          /**< BLE stack is already
in BLE scanning state */
   MPACT_ERROR_NOT_IN_SCANNING,                                                  /**<
MPact Driver is not in BLE beacon scanning state */
   MPACT_ERROR_ALREADY_IN_SCANNING,  /**< MPact Driver is already in BLE scanning
state */
   MPACT_ERROR_IN_STOP_SCANNING,                              /**< Error happens in
BLE stack when stop scanning */
   MPACT_ERROR_IN_REGISTER_CALLBACK,        /**< Error happens in BLE stack when
register callback */
   MPACT_ERROR_CALLBACK_ALREADY_REGISTERED,     /**< BLE stack already has the
callback registered */
   MPACT_ERROR_CALLBACK_IS_NULL,                 /**< MPact Driver receives a NULL
callback function pointer */
   MPACT_ERROR_INVALIDPARAM,    /**< A parameter or argument is out of range */
  MPACT_ERROR_INSUFFICIENT_BUFFER, /**< Application provided buffer is smaller than
required. Increase buffer size. */
   MPACT_ERROR_BUFFER_IS_NULL,                   /**< Application provided buffer
is NULL */
  MPACT_ERROR_INSUFFICIENT_MEM,  /**< Insufficient memory to create class instance
in MPact driver */
   MPACT_ERROR_PARAMETER_FROM_SERVER,  /**< The parameter is configured through
server. User can't change it */
   MPACT_ERROR_BAD_STRUCT_INFO,  /**< Error when StructInfo.dwUsed >
StructInfo.dwAllocated */
  MPACT_ERROR_IN_START_RECEIVE_MODE,  /**< When there is error in starting receiving
mode */
   MPACT_ERROR_ALREADY_IN_RECEIVE_MODE,  /**< MPact Driver is already in receiving
mode */
  MPACT_ERROR_IN_CONNECT_TO_BEACON,  /**< When there is error in connecting to beacon
*/
  MPACT_ERROR_IN_READ_BEACON_PARAMETER,  /**< When there is error in reading beacon
parameters */
  MPACT_ERROR_IN_STOP_RECEIVE_MODE,  /**< When there is error in stopping receiving
mode */
   MPACT_ERROR_NOT_IN_RECEIVE_MODE,  /**< MPact Driver is not in receiving mode */
   MPACT_ERROR_IN_CONFIGURE_BEACON,  /**< When there is error during configuring
beacon parameters */
  MPACT_ERROR_IN_UPGRADE_FIRMWARE,  /**< When there is error during upgrading beacon
firmware */
   MPACT_ERROR_IN_REBOOT_BEACON,  /**< When there is error in rebooting beacons */
   MPACT_ERROR_IMAGE_NOT_EXIST,  /**< The Image file does not exist */
   MPACT_ERROR_WRONG_IMAGE_FORMAT,  /**< The Image file for upgrading is wrong */
   MPACT_ERROR_STILL_PROCESSING_BEACON,  /**< When SDK is processing beacons,
additional command is not allowed*/
   MPACT_ERROR_NO_BEACON_TO_PROCESS,  /**< The beacon to be configured/upgraded/
rebooted does not find*/

  MPACT_ERROR_FAILED                                                             =
99,           /**< General error*/
```

```
} MPACT_RETURN_VALUE;
```

4.4 Functions

Requirements:

OS Versions: Windows CE 5.0 and later

BLE Stack: StoneStree One Bluetopia

Header: MPactBLEapi.h

Library: MPactBLEapi.lib

4.4.1 MPACT_InitLibrary

Initializes the MPact WinCE ClientSDK library and specifies the callback function.

This function must be called first. An MPact class instance is created inside the MPact library. At anytime, only one class instance is created at most. If the third party application calls this function multiple times, only one MPact class instance is created, while the callback function is updated according to the latest function call. Therefore, if the third party application switches the callback function, it can use this function with a new callback function pointer. The MPact library does not support multiple instances. If an application is using the MPact library, and another application calls the MPACT_InitLibrary, the return value is `MPACT_ERROR_ALREADY_INITIALIZED`.

Parameters:

`BEACONCALLBACK cbCallBack`: The function to call back with the beacon details.

Returns:

Returns values from the enum `MPACT_RETURN_VALUE`.

`MPACT_SUCCESS:` Initialization is successful

`MPACT_ERROR_IN_INITIALIZATION:` An error occurs during BLE Stack initialization.

`MPACT_ERROR_INSUFFICIENT_MEM`: Insufficient memory.

`MPACT_ERROR_ALREADY_INITIALIZED`: The MPact driver is already initialized.

`MPACT_ERROR_CALLBACK_ALREADY_REGISTERED`: The BLE Stack already has the callback registered.

`MPACT_ERROR_CALLBACK_IS_NULL`: The MPact driver receives a NULL callback function pointer.

`MPACT_ERROR_IN_REGISTER_CALLBACK`: An error occurs in the BLE stack when the register calls back.

`MPACT_ERROR_FAILED`: A general error.

Example:

```
MPACT_InitLibrary(GetNotify);
```

4.4.2 MPACT_DeinitLibrary

This function de-initializes the MPact WinCE ClientSDK library and stops processing any BLE beacons. All the callback functions are immediately stopped. This is the last function to call before exiting the third party application.

Parameters:

None.

Returns:

Returns value from the enum `MPACT_RETURN_VALUE`.

MPACT_SUCCESS: De-initialization is successful.

MPACT_ERROR_NOT_INITIALIZED: The MPact driver is not initialized.

MPACT_ERROR_IN_STOP_SCANNING: An error occurs when stopping BLE Stack scanning.

MPACT_ERROR_FAILED: A general error.

Example

```
MPACT_DeinitLibrary();
```

4.4.3 MPACT_BEACONCALLBACK

The callback function from the MPact driver returns beacon information to the third party application. It must be registered with the MPact driver during MPACT_InitLibrary. The exact beacon information contained in the function is based on the beacon mode and report mode.

Parameters:

BEACON_INFO_t* pBeaconInfo: The beacon information captured during BLE scanning

Returns:

None

Requirements:

BLE Stack: StoneStree One Bluetopia

Header: MPactBLEapi.h

Library: MPactBLEapi.lib

Example

```
MPACT_InitLibrary(GetNotify);
void __stdcall GetNotify (BEACON_INFO_t* pBeaconInfo)
{
// Process the beacon information
}
```

4.4.4 MPACT_StartScanning

Starts BLE beacon scanning. The beacon is reported through the MPACT_BEACONCALLBACK function. Beacon mode, Report mode and UUID must be set before scanning is started. The server information setting is optional. If the function MPACT_InitLibrary is not called before this function, the function fails and returns MPACT_ERROR_NOT_INITIALIZED. If the MPact SDK is already in the scanning state, the function fails and returns MPACT_ERROR_ALREADY_IN_SCANNING.

Parameters:

None

Returns:

Returns value from enum MPACT_RETURN_VALUE.

MPACT_SUCCESS: Scanning was successful

MPACT_ERROR_IN_START_SCANNING: An error occurs during BLE Stack starts scanning.

MPACT_ERROR_NOT_INITIALIZED: The MPact driver is not initialized.

MPACT_ERROR_ALREADY_IN_SCANNING: The MPact driver is already in scanning state.

MPACT_ERROR_FAILED: A general error.

Example:

```
MPACT_StartScanning();
```

4.4.5 MPACT_StopScanning

Stops the BLE beacon from scanning. There is no report generated after BLE beacon scanning is stopped. If the function MPACT_InitLibrary is not called before this function, the function fails and the return value is MPACT_ERROR_NOT_INITIALIZED. If the MPact SDK is not in the scanning state, the function fails and the return is MPACT_ERROR_NOT_IN_SCANNING.

Parameters:

None

Returns

Returns value from the enum MPACT_RETURN_VALUE.

MPACT_SUCCESS: When scanning was successful

MPACT_ERROR_IN_STOP_SCANNING: Error during BLE Stack stop scanning

MPACT_ERROR_NOT_INITIALIZED: The MPact driver is not initialized.

MPACT_ERROR_NOT_IN_SCANNING: The MPact driver is already in scanning state.

MPACT_ERROR_FAILED: A general error.

Example:

```
MPACT_StopScanning();
```

4.4.6 MPACT_SetBeaconMode

Allows a third party application to pass in Beacon Mode, which identifies the beacon payloads. This ensures the MPact ClientSDK correctly processes beacon payloads and triggers callbacks for the appropriate beacons. Make sure the beacon is in the specific beacon mode set in the third party application. Then the MPact ClientSDK can capture the beacon.

BatterySave: Beacon packet format which uses less bytes than iBeacon. Batteries last longer when set to BatterySave mode. This packet contains the battery information.

IBeacon: Follows the Apple iBeacon packet format. Set UUID in the third party application with MPACT_SetBeaconUUID first.

MPact: Follows Apple iBeacon packet format. The last byte of the packet contains the battery information. Set the UUID with function MPACT_SetBeaconUUID first.

If the API is called after scanning is started, it takes immediate effect. The beacon callback changes based on new setting.

Parameters:

BeaconMode_t beaconMode: The third party application can set up to three different modes.

Returns:

Returns value from the enum MPACT_RETURN_VALUE.

MPACT_SUCCESS: Beacon mode successfully set.

MPACT_ERROR_INVALIDPARAM: The parameter is out of range.

MPACT_ERROR_PARAMETER_FROM_SERVER: The parameter is configured by the server.

MPACT_ERROR_FAILED: A general error.

Example:

```
MPACT_SetBeaconMode(MPact);
```

4.4.7 MPACT_GetBeaconMode

Returns the BeaconMode registered in the MPact ClientSDK for beacon processing. This function is only for third party application purposes.

Parameters:

BeaconMode_t* pBeaconMode: A pointer to the buffer.

Returns:

Returns value from the enum MPACT_RETURN_VALUE.

MPACT_SUCCESS: When the beacon mode was read successfully

MPACT_ERROR_BUFFER_IS_NULL: The pBeaconMode buffer is NULL

MPACT_ERROR_FAILED: A general error.

Example:

```
BeaconMode_t beaconMode = Unknown;
MPACT_GetBeaconMode(&beaconMode);
```

4.4.8 MPACT_SetReportMode

Allows customer applications to pass in the Report Mode, which includes three different reporting behaviors. If the API is called after scanning is started, it takes immediate effect. The beacon callbacks change according to the new setting.

ClosestBeacon: The MPact ClientSDK reports the closest beacon among all the beacons or out-of-range beacons. The report time is set in the function MPACT_SetParameter.

RealtimeBeacon: The MPact ClientSDK reports all beacons in real time, in the sequence received.

ChangeOfBeacon: The MPact ClientSDK reports the closest beacon or out-of-range only when the event occurs. There is no duplicated report for the same closest beacon and out-of-range event.

Parameters:

ReportMode_t reportMode: Three different report modes are set in the MPact.

Returns:

Returns value from the enum MPACT_RETURN_VALUE.

MPACT_SUCCESS: When the report mode was set successfully

MPACT_ERROR_INVALIDPARAM: The parameter is out of range

MPACT_ERROR_FAILED: A general error.

NoReport: There is no report of an event from the MPact ClientSDK.

Example:

```
MPACT_SetReportMode(ClosestBeacon);
```

4.4.9 MPACT_GetReportMode

Returns the `ReportMode` registered in MPact ClientSDK for different reporting behaviors. This function is only for third party applications.

Parameters:

`ReportMode_t* pReportMode`: The pointer to the buffer.

Returns:

Returns value from the enum `MPACT_RETURN_VALUE`.

`MPACT_SUCCESS`: Report mode read successfully.

`MPACT_ERROR_BUFFER_IS_NULL`: The `pReportMode` buffer is NULL.

`MPACT_ERROR_FAILED`: A general error.

Example:

```
ReportMode_t reportMode = NoReport;
MPACT_GetReportMode(&reportMode);
```

4.4.10 MPACT_SetParameter

ConnectionTimeout: This parameter defines how long the beacon waits in connection mode if not currently connected. Once the timeout is triggered, the beacon switches into broadcast mode. The value unit is in minutes. The minimum value is 1 minute. The default value in the MPact ClientSDK is 10 minutes.

Parameters:

`Parameter_t parameter`: The name of the parameter.

`int value`: The value of the parameter.

Returns:

Return values from the enum `MPACT_RETURN_VALUE`.

`MPACT_SUCCESS`: When the parameter was set successfully.

`MPACT_ERROR_INVALIDPARAM`: The parameter is out of range.

`MPACT_ERROR_FAILED`: A general error.

Parameters:

`ReportTime`**:** This parameter defines how often the MPact ClientSDK triggers the callback event if the report mode is set to `ClosestBeacon`. The value unit is in milliseconds. The minimum value is 250ms. The default report time in the MPact ClientSDK is 250ms.

If the `ReportTime` is set after scanning is started, it takes immediate effect. All of the following beacon callbacks change according to the new setting.

`BeaconInterval:` This parameter defines how often the MPact ClientSDK reports beacons from the BLE stack. The value unit is in milliseconds. The minimum value is 100ms. The default Beacon Interval in the MPact ClientSDK is 250ms. The smaller the value, the more beacons reported from the BLE stack. If the `BeaconInterval` is set after scanning is started, the third party application must be stopped and re-started for scanning to take effect.

Example:

```
MPACT_SetParameter(ReportTime, 250);
```

```
MPACT_SetParameter(BeaconInterval, 150);
```

4.4.11 MPACT_GetParameter

ConnectionTimeout: This parameter defines how long a beacon waits in connection mode if not currently connected. The value unit is in minutes.

Parameters:

`Parameter_t parameter`: The name of the parameter.

`int* pValue`: The pointer to the buffer.

Returns:

Returns value from the enum `MPACT_RETURN_VALUE`.

`MPACT_SUCCESS`: Report mode read successfully.

`MPACT_ERROR_BUFFER_IS_NULL`: The `pValue` buffer is NULL.

`MPACT_ERROR_INVALIDPARAM`: The `parameter` is invalid.

`MPACT_ERROR_FAILED`: A general error.

`ReportTime`: This parameter defines how often the MPact ClientSDK triggers the callback event if the report mode is set to `ClosestBeacon`. The value unit is in milliseconds.

`BeaconInterval`: This parameter defines how often the MPact ClientSDK gets beacons from BLE stack. The value unit is in milliseconds.

Example:

```
int nReportTime = 0;
int nBeaconInterval = 0;
int nConnectionTimeout = 0;

MPACT_GetParameter(ReportTime,&nReportTime);
MPACT_GetParameter(BeaconInterval,&nBeaconInterval);
MPACT_GetParameter(ConnectionTimeout,&nConnectionTimeout)
```

4.4.12 MPACT_SetBeaconUUID

Allows third party customer applications to pass in the UUID of the beacons pending registration. This ensures the MPact ClientSDK is processing only beacons with correct UUIDs and triggers callbacks for them. This function is required only when beacon mode is `IBeacon` or `MPact.` Only one UUID must be specified for these two modes. The library does not support scan for all UUIDs. If the API is called after scanning is started, it takes immediate effect. All of the following beacon callbacks change according to the new setting.

Parameters:

`const UUID_Beacon_t *pUUID`: The UUID the third party application wants to use.

Returns:

Returns value from the enum `MPACT_RETURN_VALUE`.

`MPACT_SUCCESS`: UUID read successfully

`MPACT_ERROR_BUFFER_IS_NULL`: The `pUUID` buffer is NULL.

`MPACT_ERROR_PARAMETER_FROM_SERVER:` The parameter is configured by the server.

MPACT_ERROR_INSUFFICIENT_BUFFER: The pUUID does not have enough space.
MPACT_ERROR_BAD_STRUCT_INFO: StructInfo.dwUsed is larger than StructInfo.dwAllocated

MPACT_ERROR_FAILED: A general error.

Example:

```
UUID_BEACON_t uuid;


memset(&uuid,0,sizeof(uuid));
uuid.StructInfo.dwAllocated = sizeof(uuid);
uuid.StructInfo.dwUsed = sizeof(uuid);


ASSIGN_UUID(uuid,0xFE,0x91,0x32,0x13,0xB3,0x11,0x4A,0x42,0x8C,0x16,0x47,0xFA,0xEA,0xC9,0x38,0xDB);
MPACT_SetBeaconUUID(&uuid);
```

4.4.13 MPACT_GetBeaconUUID

Returns the UUID registered for beacon processing.

Parameters:

UUID_Beacon_t *pUUID: The UUID buffer provided by the third party application.

Returns

Returns value from the enum MPACT_RETURN_VALUE.

MPACT_SUCCESS: UUID copied successfully

MPACT_ERROR_BUFFER_IS_NULL: The pUUID buffer is NULL

MPACT_ERROR_INSUFFICIENT_BUFFER: The pUUID does not have enough space

MPACT_ERROR_BAD_STRUCT_INFO: StructInfo.dwUsed is larger than StructInfo.dwAllocated

MPACT_ERROR_FAILED: A general error.

This function is only for third party application information purposes.

Example:

```
UUID_BEACON_t uuid;
memset(&uuid,0,sizeof(uuid));
uuid.StructInfo.dwAllocated = sizeof(uuid);
uuid.StructInfo.dwUsed = sizeof(STRUCT_INFO);


MPACT_GetBeaconUUID(&uuid);
```

4.4.14 MPACT_GetClosestBeacon

Allows the third party application to actively fetch the current closest beacon from MPact ClientSDK.

Parameters:

`BEACON_INFO_t* pBeaconInfo`: The beacon information buffer provided by the third party application.

Returns:

Returns value from the enum `MPACT_RETURN_VALUE`.

`MPACT_SUCCESS`: Function executed successfully.

`MPACT_ERROR_BUFFER_IS_NULL`: The `pBeaconInfo` buffer is NULL or the `cBeaconID` buffer is NULL

`MPACT_ERROR_NOT_INITIALIZED`: The MPact driver is not initialized.

`MPACT_ERROR_NOT_IN_SCANNING`: The MPact driver is already in scanning state.

`MPACT_ERROR_INSUFFICIENT_BUFFER`: The `cBeaconID` does not have enough space.

`MPACT_ERROR_BAD_STRUCT_INFO: StructInfo.dwUsed` is larger than `StructInfo.dwAllocated`

`MPACT_ERROR_FAILED`: A general error.

When the report mode is set to `NoReport`, the third party application can use this function to retrieve the current closest beacon. It can also be used with other report modes.

Example:

```
BEACON_INFO_t beaconInfo;
memset(&beaconInfo, 0, sizeof(beaconInfo));
beaconInfo.StructInfo.dwAllocated = sizeof(beaconInfo);
beaconInfo.StructInfo.dwUsed = sizeof(STRUCT_INFO);

MPACT_GetClosestBeacon(&beaconInfo);
```

4.4.15 MPACT_SetServerInfo

Allows applications to set server information so it can be exchanged with the MPact ClientSDK. Then the ClientSDK can exchange information with the server. If the server is properly set up, the MPact Client SDK downloads the settings (beacon mode and UUID) from the server and override user settings. This function is optional.

If the server information is not set, the MPact ClientSDK functions like it is offline.

Only the HTTP protocol is supported. Even if a user sets the protocol to HTTPS, the actual connection is HTTP.

If the server information is set, the server settings (beacon mode and UUID, if exist) is downloaded and configured in the MPact ClientSDK. The server settings (beacon mode and UUID, if it exists) have higher priority than user settings and they overrides user settings. If the server information is set up properly and the report mode is `ClosestBeacon` or `ChangeOfBeacon`, the MPact library regularly updates the client visit information (the closest beacon MAC, user nickname, battery life of the beacon) to the server. Then in the server user interface, a user icon displays beside the corresponding beacon.

If the third party application wants to remove server settings and makes use of user settings, it must call `MPACT_SetServerInfo` again with `cServerIP` set to empty (`""`), and followed by the `MPACT_SetBeaconMode` function and `MPACT_SetBeaconUUID`. If the third party application does not set the beacon mode and UUID, the server settings is retained. The MPact library does not send the client visit information to the server any more.

If the API is called after scanning is started, it takes immediate effect after the server settings are downloaded. All the following beacons callbacks change according to the server settings. The new settings takes immediate effect If the `cServerIP` is set to empty and followed by `MPACT_SetBeaconMode` and `MPACT_SetBeaconUUID`.

Parameters:

`const SERVER_INFO_t* pServerInfo`: The server information buffer provided by the third party application.

Returns:

Returns value from the enum `MPACT_RETURN_VALUE`.

`MPACT_SUCCESS`: Server information set successfully

`MPACT_ERROR_BUFFER_IS_NULL`: The `pServerInfo` buffer is NULL or the `cServerIP` buffer is NULL.

`MPACT_ERROR_INSUFFICIENT_BUFFER`: The `pServerInfo` does not have enough space.

`MPACT_ERROR_BAD_STRUCT_INFO: StructInfo.dwUsed` is larger than `StructInfo.dwAllocated`

`MPACT_ERROR_FAILED`: A general error.

Example:

```
SERVER_INFO_t serverInfo;
memset(&serverInfo,0,sizeof(serverInfo));
serverInfo.StructInfo.dwAllocated = sizeof(serverInfo);
serverInfo.StructInfo.dwUsed = sizeof(serverInfo);

strncpy(serverInfo.cServerIP, "server ip or domain", SERVER_IP_LEN);
strncpy(serverInfo.cNickname, "Sam", NICKNAME_LEN);
strncpy(serverInfo.cUsername, "user1", USERNAME_LEN);
strncpy(serverInfo.cPassword, "MyPassword ", PASSWORD_LEN);

serverInfo.Protocol = HTTP;
serverInfo.nPort = 80;

MPACT_SetServerInfo(&serverInfo);
```

4.4.16 MPACT_GetServerInfo

Allows third party applications to retrieve the server information from the MPact ClientSDK.

Parameters

`SERVER_INFO_t* pServerInfo`: The server information buffer provided by the third party application.

Returns:

Returns value from the enum `MPACT_RETURN_VALUE`.

`MPACT_SUCCESS`: Server information set successfully

`MPACT_ERROR_BUFFER_IS_NULL`: Any buffer is NULL

`MPACT_ERROR_INSUFFICIENT_BUFFER`: The `pServerInfo` does not have enough space

`MPACT_ERROR_BAD_STRUCT_INFO: StructInfo.dwUsed` is larger than `StructInfo.dwAllocated`

`MPACT_ERROR_FAILED`: A general error.

The return buffer is empty if the corresponding field is not set yet in the MPact SDK.

Example:

```
SERVER_INFO_t serverInfo;
memset(&serverInfo,0,sizeof(serverInfo));
serverInfo.StructInfo.dwAllocated = sizeof(serverInfo);
serverInfo.StructInfo.dwUsed = sizeof(STRUCT_INFO);

MPACT_GetServerInfo(&serverInfo);
```

4.4.17 MPACT_GetLibVersion

Returns the version number of the MPact ClientSDK library.

Parameters:

`VERSION_INFO_t* pVersionInfo:` Third party application provided buffer address.

Returns:

Returns value from the enum `MPACT_RETURN_VALUE`.

`MPACT_SUCCESS`: Function executed successfully.

`MPACT_ERROR_BUFFER_IS_NULL`: The input buffer is NULL.

`MPACT_ERROR_INSUFFICIENT_BUFFER`: The `pVersionInfo` does not have enough space.

`MPACT_ERROR_BAD_STRUCT_INFO: StructInfo.dwUsed` is larger than `StructInfo.dwAllocated.`

`MPACT_ERROR_FAILED`: A general error.

This function is only for information purposes.

Example

```
VERSION_INFO_t versionInfo;
memset(&versionInfo,0,sizeof(versionInfo));
versionInfo.StructInfo.dwAllocated = sizeof(versionInfo);
versionInfo.StructInfo.dwUsed = sizeof(STRUCT_INFO);

MPACT_GetLibVersion(&versionInfo);
```

4.4.18 MPACT_BEACONRECEIVINGCALLBACK

The callback function from the MPact driver. It returns results, beacon status and parameters to third party applications. It must be registered with MPact driver during `MPACT_StartReceiveModeDetection`.

Parameters:

`MPACT_RETURN_VALUE MPactReturn`: The specific beacon processing result.

`BEACON_STATUS_t* pBeaconStatus`: The current beacon status.

`BEACON_PARAMETER_t* pBeaconParameter`: The current beacon parameters.

Returns:

None

The exact beacon status and parameters contained in the function may be different for different beacons.

`MPactReturn` tells the processing result of the current beacon. If it is not `MPACT_SUCCESS`, there must be some error happened.

The `pBeaconStatus` provides the current status of the beacon processed. The `nPercent` is only used during firmware image upgrading. It lets the third party application know the upgrade progress. The default value of `nPercent` is -1 if it is not used.

The `pBeaconParameter` is only available when the beacon in connected and parameters are read. Therefore, check for a NULL value before reading any parameter from it. The `cFirmwareVersionA` and `cFirmwareVersionB` are the versions of the current firmware image A and B of the beacon. These 2 parameters are information-only and read-only.

Example:

```
MPACT_StartReceiveModeDetection(ReceivingNotify);
void __stdcall ReceivingNotify(MPACT_RETURN_VALUE MPactReturn,
BEACON_STATUS_t* pBeaconStatus, BEACON_PARAMETER_t* pBeaconParameter)
{
// Process the information
}
```

4.4.19 MPACT_StopReceiveModeDetection

Stops the receiving mode detections for beacons in MPact ClientSDK library. Stops processing any remaining beacons from configuration and upgrading. All the callback functions are immediately stopped.

Parameters:

None.

Returns:

Returns value from the enum `MPACT_RETURN_VALUE`.

`MPACT_SUCCESS`: Stop action successful

`MPACT_ERROR_IN_STOP_RECEIVE_MODE`: The MPact driver is not initialized.

`MPACT_ERROR_NOT_IN_RECEIVE_MODE`: MPact driver is not in receiving mode

`MPACT_ERROR_FAILED`: A general error.

This is the last function called before the third party application exits.

Example:

MPACT_StopReceiveModeDetection();

4.4.20 MPACT_ConfigureBeacon

Starts the configuration for the list of beacons with specific parameter settings.

Parameters:

`BEACON_PARAMETER_t *pBeaconParameter`: The list of beacon parameter structures. The third party client applications can set different values for parameters in different beacons.

`int nNumberOfBeacon`: Number of beacons configured. This is also the number of beacon parameter structures.

Returns:

Return values from the enum `MPACT_RETURN_VALUE`.

`MPACT_SUCCESS:` Function successful.

`MPACT_ERROR_NOT_IN_RECEIVE_MODE`: The MPact driver is not in receiving mode.

`MPACT_ERROR_FAILED`: A general error.

Third party client applications must call `MPACT_StartReceiveModeDetection` with the `MPACT_BEACONRECEIVINGCALLBACK` function before this function.

The configuration result is sent to third party applications through the callback function.

The possible returns in the callback function are:

`MPACT_SUCCESS:` When the beacon is successfully configured with given parameters

`MPACT_ERROR_IN_CONFIGURE_BEACON`: An error logged during configuring beacon parameters.

`MPACT_ERROR_IN_CONNECT_TO_BEACON`: The MPact driver cannot connect to the beacon.

`MPACT_ERROR_IN_READ_BEACON_PARAMETER`: The MPact driver cannot read the parameters from the beacon.

`MPACT_ERROR_STOPPED_BY_APP`: Third party client application called `MPACT_StopReceiveModeDetection` during configuration.

`MPACT_ERROR_FAILED`: A general error.

After successful configuration, beacons are rebooted automatically.

Example:

```
// App plans to configure 10 beacons.
BEACON_PARAMETER_t BeaconParameter[10];

// Set parameters for these 10 beacons

MPACT_ConfigureBeacon(BeaconParameter, 10);
```

4.4.21 MPACT_UpgradeBeacon

Starts the upgrade process for a list of beacons with a specific firmware image.

Parameters:

`BEACON_LIST_t *pBeaconList`: The list of beacons upgraded.

`int nNumberOfBeacon`: The number of beacons upgraded. This is also the number of beacon list structures.

`IMAGE_FILE_t *pImageFile`: The path to the firmware image file. The file should have the extension *.bin.

Returns:

Returns values from the enum MPACT_RETURN_VALUE.

MPACT_SUCCESS: Function successful.

MPACT_ERROR_NOT_IN_RECEIVE_MODE: The MPact driver is not in receiving mode.

MPACT_ERROR_IMAGE_NOT_EXIST: The image file does not exist.

MPACT_ERROR_WRONG_IMAGE_FORMAT: The Image file for upgrading is wrong.

MPACT_ERROR_FAILED: A general error.

The third party client applications must call MPACT_StartReceiveModeDetection with MPACT_BEACONRECEIVINGCALLBACK function before this function.

The upgrade result of each beacon is sent to the third party application through the callback function.

The possible returns in the callback function are:

MPACT_SUCCESS: Beacon is successfully upgraded with a given firmware image.

MPACT_ERROR_IN_UPGRADE_FIRMWARE: There is an error when upgrading beacons.

MPACT_ERROR_IN_CONNECT_TO_BEACON: The MPact driver cannot connect to the beacon.

MPACT_ERROR_IN_READ_BEACON_PARAMETER: The MPact driver cannot read the parameters from the beacon.

MPACT_ERROR_STOPPED_BY_APP: Third party application called MPACT_StopReceiveModeDetection during upgrading.

MPACT_ERROR_FAILED: A general error.

The nPercent of the structure BEACON_STATUS_t shows the current upgrade progress of the specific beacon.

The firmware upgrade only upgrades the firmware image B inside the beacon.

After successful upgrading, the beacons are rebooted automatically.

Example:

```
// App plans to upgrade 10 beacons.
BEACON_LIST_t BeaconList[10];
IMAGE_FILE_t Image;

// Set BeaconList

// Set Image file
memset(&Image,0,sizeof(Image));
Image.StructInfo.dwAllocated = sizeof(Image);
Image.StructInfo.dwUsed = sizeof(Image);

_snwprintf(Image.szImageFile, MAX_PATH, TEXT("\\Application\\broadcaster-1.0.1.0-014R.bin"));

MPACT_UpgradeBeacon(BeaconList, 10, &Image);
```

4.4.22 MPACT_RebootBeacon

Reboots the list of beacons into broadcasting mode.

Parameters:

`BEACON_LIST_t *pBeaconList`: The list of beacons rebooted.

`int nNumberOfBeacon`: The number of beacons rebooted. This is also the number of beacon list structures.

Returns:

Returns values from the enum `MPACT_RETURN_VALUE`.

`MPACT_SUCCESS:` Displays upon a successful beacon reboot.

`MPACT_ERROR_NOT_IN_RECEIVE_MODE`: The MPact driver is not in receiving mode.

`MPACT_ERROR_FAILED`: A general error.

The third party applications must call `MPACT_StartReceiveModeDetection` with the `MPACT_BEACONRECEIVINGCALLBACK` function before this function.

The reboot result of each beacon is sent to the third party application through the callback function.

The possible returns in the callback function are:

`MPACT_SUCCESS:` When the beacon is successfully rebooted.

`MPACT_ERROR_IN_REBOOT_BEACON`: There is an error when beacons are rebooted.

`MPACT_ERROR_IN_CONNECT_TO_BEACON`: The MPact driver cannot connect to the beacon.

`MPACT_ERROR_STOPPED_BY_APP`: The third party application called `MPACT_StopReceiveModeDetection` during rebooting.

`MPACT_ERROR_FAILED`: A general error.

Example:

```
// App plans to reboot 10 beacons.
BEACON_LIST_t BeaconList[10];

// Set BeaconList

MPACT_RebootBeacon(BeaconList, 10);
```

4.5 Enums

4.5.1 TagBeaconMode

Defines the mode of the MPact beacon.

Modes include:

`BatterySave`

The beacon format using less bytes than iBeacon for longer battery life. This packet contains the battery information.

`iBeacon`

Standard iBeacon mode. Major and Minor are combined to form the tag ID. Battery life is not reported.

`MPact`

iBeacon mode is used, however, the Major and Minor fields have specific meanings. Battery life is included in the LSB of the Minor field.

4.5.2 TagReportMode

TagReportMode is data type that defines the set of beacon reports.

Modes include:

`No Report`

There is no report of an event from MPact ClientSDK.

`ClosestBeacon`

The MPact ClientSDK reports the closest beacon among all or out-of-range beacons.

`RealTimeBeacon`

The MPact ClientSDK reports all beacons in real time, in the sequence received.

`ChangeOfBeacon`

The MPact ClientSDK reports the closest beacon or out-of-range only when the event happens. There is no duplicated report for the same closest beacon and out-of-range event.

4.5.3 TagParmeter

TagParmeter is data type that defines the tag parameters.

Parameters include:

`ReportTime`: This parameter defines how often the MPact ClientSDK triggers the callback event if the report mode is set to ClosestBeacon. The value unit is in milliseconds. The minimum value is 250ms. The default report time in MPact ClientSDK is 250ms.

If the ReportTime is set after scanning is started, it takes immediate effect. The following beacon callbacks change according to the new setting:

`BeaconInterval:` Defines how often the MPact ClientSDK gets beacons from BLE stack. The value unit is in milliseconds.

`ConnectionTimeout`: Defines how long the beacon waits in the connection mode if not currently connected. The value unit is in minutes.

4.5.4 TagProtocol

TagProtocal defines the set of protocol types used by the MPact Client library to communicate with the MPact Server.

Protocols include:

`http`

HyperText Transfer protocol

`https`

4.5.5 TagBeaconState

Defines the beacon state.

BeaconState includes

`UnknownState` = 0

The beacon state is unknown.

`Discovered`

The beacon is discovered.

`Connected`

The beacon is connected to the MPact ClientSDK.

`Disconnected`

The beacon is disconnected from the MPact ClientSDK.

`ServiceFound`

The beacon's service profiles are found.

`ParameterRead`

The beacon's parameters are read.

`Configuring`

The beacon is being configured.

`Upgrading`

The beacon firmware is being upgraded.

`Rebooting`

The beacon is being rebooted.

`Configured`

The beacon has been configured.

`Upgraded`

The beacon has upgraded.

`Rebooted`

The beacon has rebooted.

4.6 Structures

Captures all the structures and enums used within MPACT Client SDK library.

```
typedef enum tagParameter {
   ReportTime = 0,
   BeaconInterval,
   ConnectionTimeout,
} Parameter_t;


typedef enum tagBeaconState {
   UnknownState = 0,
   Discovered,
   Connected,
   Disconnected,
   ServiceFound,
   ParameterRead,
   Configuring,
   Upgrading,
   Rebooting,
   Configured,
   Upgraded,
   Rebooted,
} BeaconState_t;

typedef struct
{
   STRUCT_INFO StructInfo;
   char cMAC[MAC_LEN+1];
}BEACON_LIST_t;

typedef struct
{
   STRUCT_INFO StructInfo;
   char cMAC[MAC_LEN+1];
   BeaconState_t State;
   int nPercent;
}BEACON_STATUS_t;


typedef struct
{
   STRUCT_INFO StructInfo;
   char cMAC[MAC_LEN+1];
   int nPower;
   int nChannel;
   int nInterval;
   BeaconMode_t Mode;
   UUID_BEACON_t UUID;
   int nMajor;
   int nMinor;
   char cFirmwareVersionA[IMAGE_VERSION_LEN+1];
   char cFirmwareVersionB[IMAGE_VERSION_LEN+1];
}BEACON_PARAMETER_t;
```

4.7 Registry Settings

There is one registry value to control the logging level in the MPact ClientSDK.

The log file is stored in the device: `\Logs\MPactLog.txt`

[HKEY_CURRENT_USER\MPact]

"DebugZone" = dword:xxx

Set DebugZone to 0 to disable all logging.

Set DebugZone to 0xFF to enable all logging.

Set DebugZone between 0 to 0xFF to enable different logging levels, according to the table below:

typedef enum tagMPactLogLevel

{

MPACTLOG_LEVEL_TRACE = 0,

MPACTLOG_LEVEL_DEBUG,

MPACTLOG_LEVEL_INFO,

MPACTLOG_LEVEL_WARNING,

MPACTLOG_LEVEL_ERROR,

MPACTLOG_LEVEL_CRITICAL,

MPACTLOG_LEVEL_REALTIME,

MPACTLOG_LEVEL_USER,

} enumMPactLogLevel;

NOTE: If MPACTLOG_LEVEL_REALTIME (0x40) bit is set, there is huge log message.

APPENDIXCUSTOMER SUPPORT

If you have a problem with your equipment, contact Support for your region. Support and issue resolution is provided for products under warranty or that are covered by a services agreement. Contact information and Web self-service is available by visiting *www.zebra.com/support*.

When contacting Support, please provide the following information:

- *MAC ID of the unit*
- *Model number or product name*
- *Software type and version number*

Support responds to calls by email or telephone within the time limits set forth in support agreements. If you purchased your product from a business partner, contact that business partner for support.

Customer Support Web Site

The support site, located at *www.zebra.com/support* provides information and online assistance including developer tools, software downloads, product manuals, support contact information and online repair requests.

Manuals

To see manuals, go to: *www.zebra.com/support*.

ZEBRA

MN002604A01 Revision A February, 2016