MobileFirst Platform {dev}

Location services in native Android applications

Relevant to:
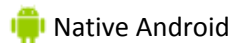
Native Android

Architecture
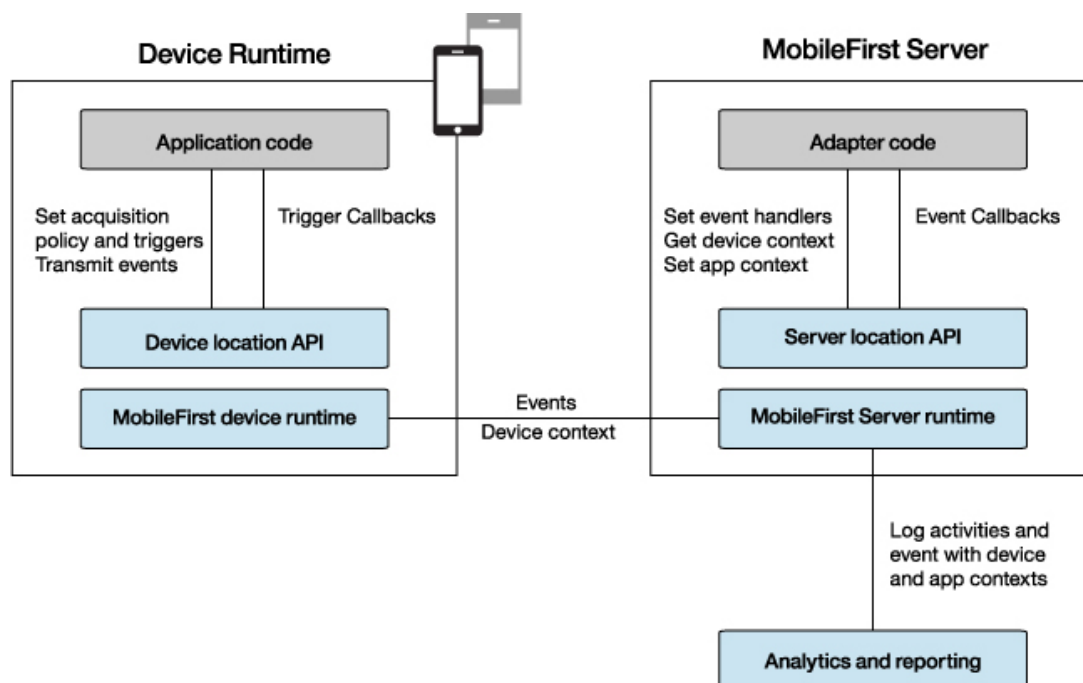
The application code on the mobile device, in the form of an acquisition policy, controls the collection of data from device sensors.

The collected data is referred to as the device context.

When a change occurs in the device context, such as a change in the geolocation of the device or the fact that it entered a Wi-Fi zone, triggers can be activated.

Triggers specify that an action occurs: either a callback function is called, or an event is sent to the server, based on the device context.

Events are created by triggers and application code, and include a snapshot of the device context at the time of their creation.

Events are buffered on the client and are transmitted to the server periodically.

The server might process the event later.

During the event transmission process, the device context is synchronized transparently to the server.

To handle the events, the server uses adapter application code.

This code sets up event handlers on the server. These handlers filter event data and pass matching events to a callback function.

The code also accesses the client device context (its location and Wi-Fi network information) and sets an application context.

Server activities and received events are logged, together with the device and application contexts, for future reporting and analytics.

Acquisition policy

An acquisition policy defines how acquisition takes place.

```
WLAcquisitionPolicy policy = new WLAcquisitionPolicy();
```

Geo acquisition policy

To enable geolocation, make sure to update your `AndroidManifest.xml` file to enable the following permissions:

```
AndroidManifest.xml
<uses-permission
 android:name="com.google.android.c2dm.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission
 android:name="com.google.android.c2dm.permission.ACCESS_FINE_LOCATION"/>
```

Then in your Java code:

```
policy.setGeoPolicy(WLGeoAcquisitionPolicy.getLiveTrackingProfile());
```

LiveTracking is a preset profile that uses the most accurate settings to track the device.
Additional configuration options:

- `RoughTracking`
- `PowerSaving`
- Custom settings

> For more information, see the topic about setting an acquisition policy, in the user documentation.

Wi-Fi acquisition policy

To enable Wi-Fi tracking, make sure to update your `AndroidManifest.xml` file to enable the following permissions:

```
AndroidManifest.xml
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
```

Then, in your Java code:

```
policy.setWifiPolicy(new WLWifiAcquisitionPolicy()
                .setInterval(10000)
                .setAccessPointFilters(
                        Arrays.asList(
                                new WLWifiAccessPointFilter("Net1"),
                                new WLWifiAccessPointFilter("Net2","*")
                                )
                        )
                );
```
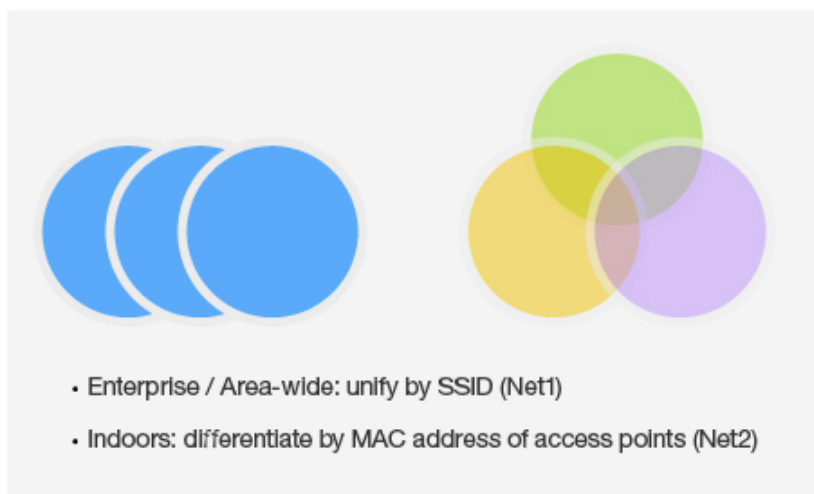
The `interval` parameter is the polling interval, in milliseconds. Wi-Fi polling is performed at each interval.

The `accessPointFilters` parameter lists access points of interest.
In the above example, the acquisition policy:

- Ignores everything except `Net1` and `Net2` – Helpful in dynamic environments, such as mobile hotspots.
- Considers all `Net1` access points as one single access point.
- Differentiates `Net2` access points by MAC address.

- Enterprise / Area-wide: unify by SSID (Net1)
- Indoors: differentiate by MAC address of access points (Net2)

> For more information, see the topic about setting an acquisition policy, in the user documentation.

Triggers

You can set up triggers as follows:

- Geo/Wi-Fi fences: `Enter`, `Exit`, `Dwell Inside`, `Dwell Outside`)
- Movement: Geo: `PositionChange`, Wi-Fi: `VisibleAccessPointsChange`
- Wi-Fi: `Connect` / `Disconnect`

```
WLTriggersConfiguration triggers = new WLTriggersConfiguration();
triggers.getGeoTriggers().put(
        "trigger1",
        new WLGeoEnterTrigger()
        .setArea(new WLCircle(new WLCoordinate(40.68917,-74.04444),100))
        .setCallback(libertyAtLast)
        .setEvent(new JSONObject().put("bring","me").put("your","huddledMasses"))
        );
```

In the above example, `trigger1` is an `Enter` trigger which is activated when the device enters the defined circle (longitude, latitude, and radius).

When a trigger activates, it can call a callback function and/or create an event to be sent to the server.

> For more information, see the topic about triggers, in the user documentation.

Start acquisition

Use the `policy` and `triggers` defined above to start the acquisition.

```
WLLocationServicesConfiguration config = new WLLocationServicesConfiguration();
config.setPolicy(policy);
config.setTriggers(triggers);

WLClient.getInstance().getWLDevice().startAcquisition(config);
```

Events

Client side
Events are generated by triggers, as explained in section [Triggers](#).

```
.setEvent(new JSONObject().put("bring","me").put("your","huddledMasses"))
```

Events can also be generated manually by calls to the API:

```
WLClient.getInstance().transmitEvent(event,immediate)
```

Where the `event` parameter is a `JSONObject` and the `immediate` parameter is a `boolean` value.

Server side
In the adapter code, create event handler:

```
WL.Server.createEventHandler(filter, handlerFunction);
```

Events that match the `filter` value are passed to `handlerFunction`.
Filter examples:

- `{status: "platinum"}` – Handle platinum members only
- `{hotel: { country: "USA" } }` – Hotels in the USA
- `{}` – All events

Register the event handlers:

```
WL.Server.setEventHandlers([...]);
```

> For more information, see the topic about working with geofences and triggers, in the user documentation.

Sample application

[Click to download](#) the MobileFirst project.

[Click to download](#) the Native project.

The LocationServices sample demonstrates:

- Acquiring an initial position
- Using a Geo profile
- Using Geo triggers for DwellInside, Exit area, and PositionChange
- Transmitting event to the server on DwellInside and Exit area
- Ongoing acquisition

Location Services

Your position will appear below. Errors will be displayed in alerts.

After 3 seconds, if you haven't moved more than 50 meters, an alert will be displayed and an event will be sent to the server.
Once you have moved more than 200 meters, an alert will be displayed and ant event will be sent to the server.

Longitude:
Latitude:
Timestamp: