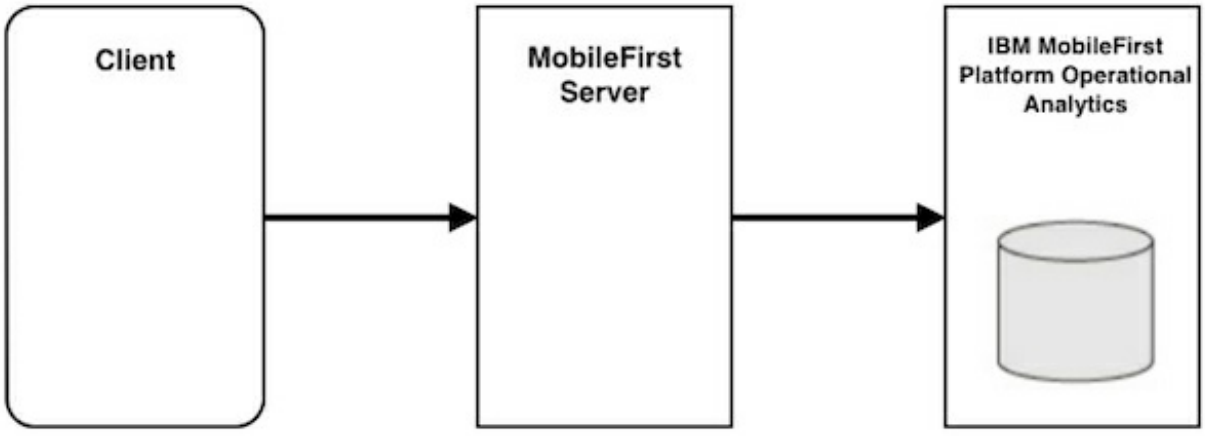
Operational Analytics

This tutorial covers the following topics:

- Introducing IBM MobileFirst Platform Operational Analytics
- Viewing the Analytics Dashboard - Configurations
- Capturing data
- Capturing data - Application Activities
- Capturing data - Notification Activities
- Capturing data - Server logs
- Capturing data - Client logs
- Analytics logs
- Sending data
- Creating a custom chart

Introducing IBM MobileFirst Platform Operational Analytics

MobileFirst Operational Analytics collects data about applications, adapters, devices, logs, and your own custom events to give a high-level view of the client interaction with the IBM MobileFirst Platform Server.



In IBM MobileFirst Platform Foundation V7.0, MobileFirst Operational Analytics is delivered as an EAR file which contains two WAR files: `analytics.war` and `analytics-service.war`. You can deploy the EAR file to the following supported application servers:

- Liberty
- WebSphere® Application Server
- Tomcat

In MobileFirst Studio, the two WAR files are installed and available by default in the embedded Liberty server.

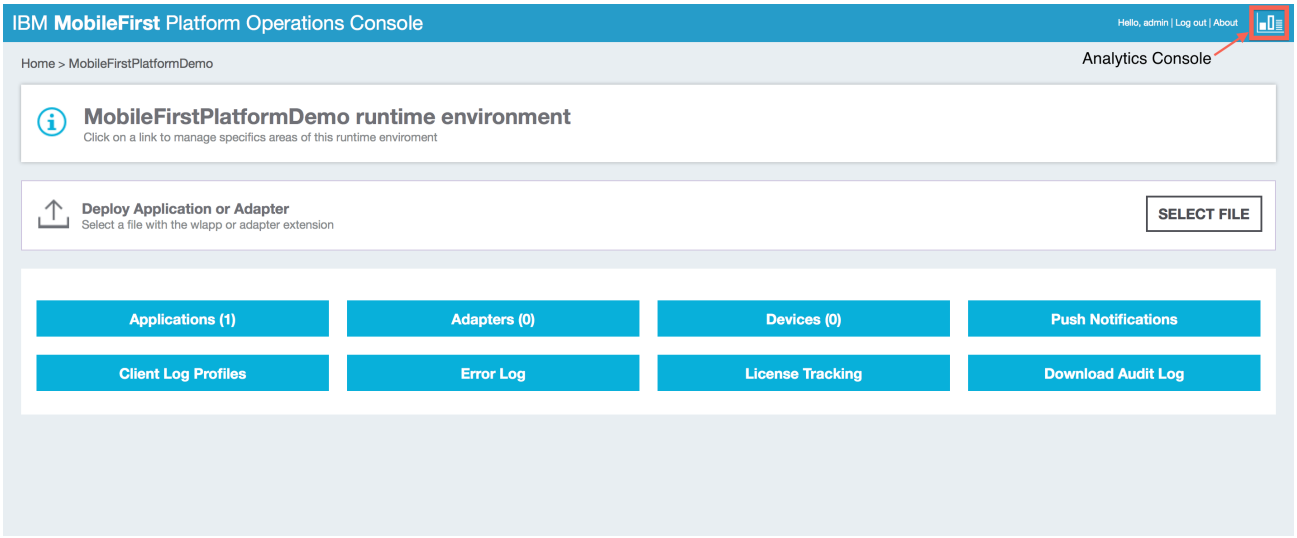
Viewing the Analytics Dashboard - Configurations

The `wl.analytics.url` property must be set to send data to the Analytics server and the `wl.analytics.console.url` must be set to access the Analytics dashboard.

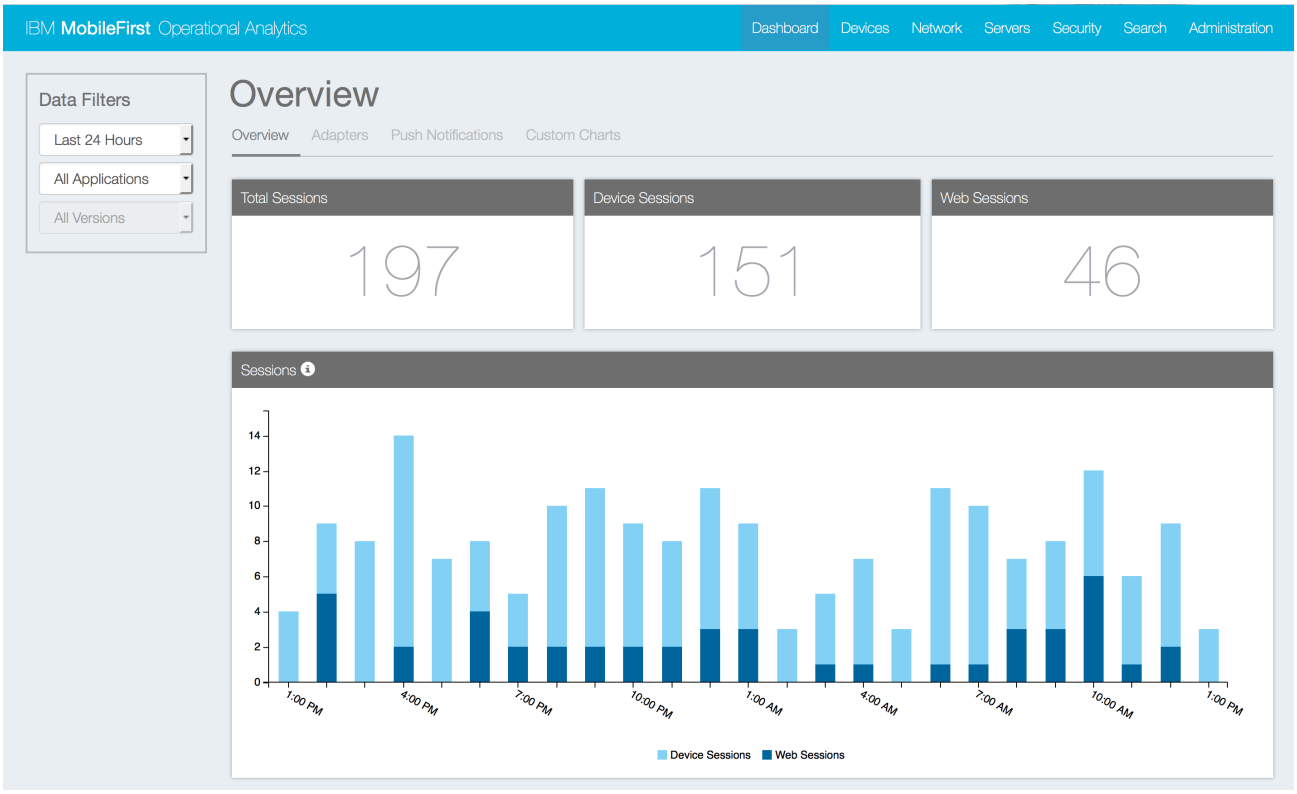
1. You can set these two properties in the `server.xml` file:

   ```
   <jndiEntry jndiName="AppName/wl.analytics.url" value="http://localhost:10080/analytics-service/data"/
   >
   <jndiEntry jndiName="AppName/wl.analytics.console.url" value="http://localhost:10080/analytics/conso
   le"/>
   ```

   After the property is set, the **Analytics Dashboard** link appears in the MobileFirst Operations Console.



2. Click the **Analytics Dashboard** link to open up the dashboard in a new window.



Capturing data

Different types of analytics events are captured by the MobileFirst Operational Analytics server: app activities, notification activities, server logs, and client logs.

Application activities

- Client initializations with the server
- Adapter calls

Notification activities

- Push notifications

Server Logs

- Server events
- Server stack traces

Client Logs

- Debug logs
- Crashes
- Custom events
- Network latency information

Capturing data - Application activities

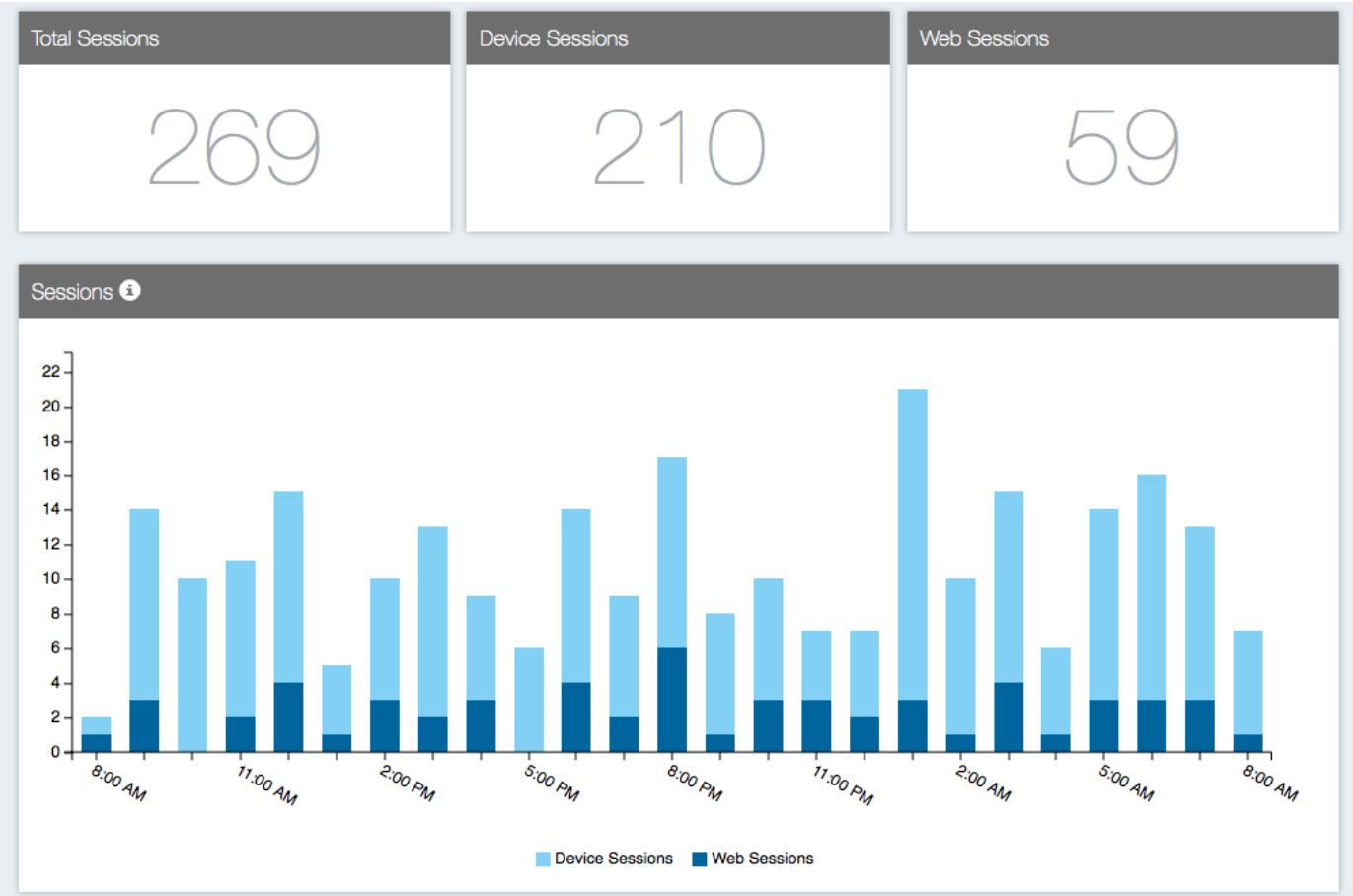
When an application activity occurs, the event is captured automatically and forwarded to the MobileFirst Operational Analytics server.

- The following API call results in a session hit that is visualized on MobileFirst Operational Analytics:

```
// a 'session hit' will be recorded upon a successful connection
WL.Client.connect();
```

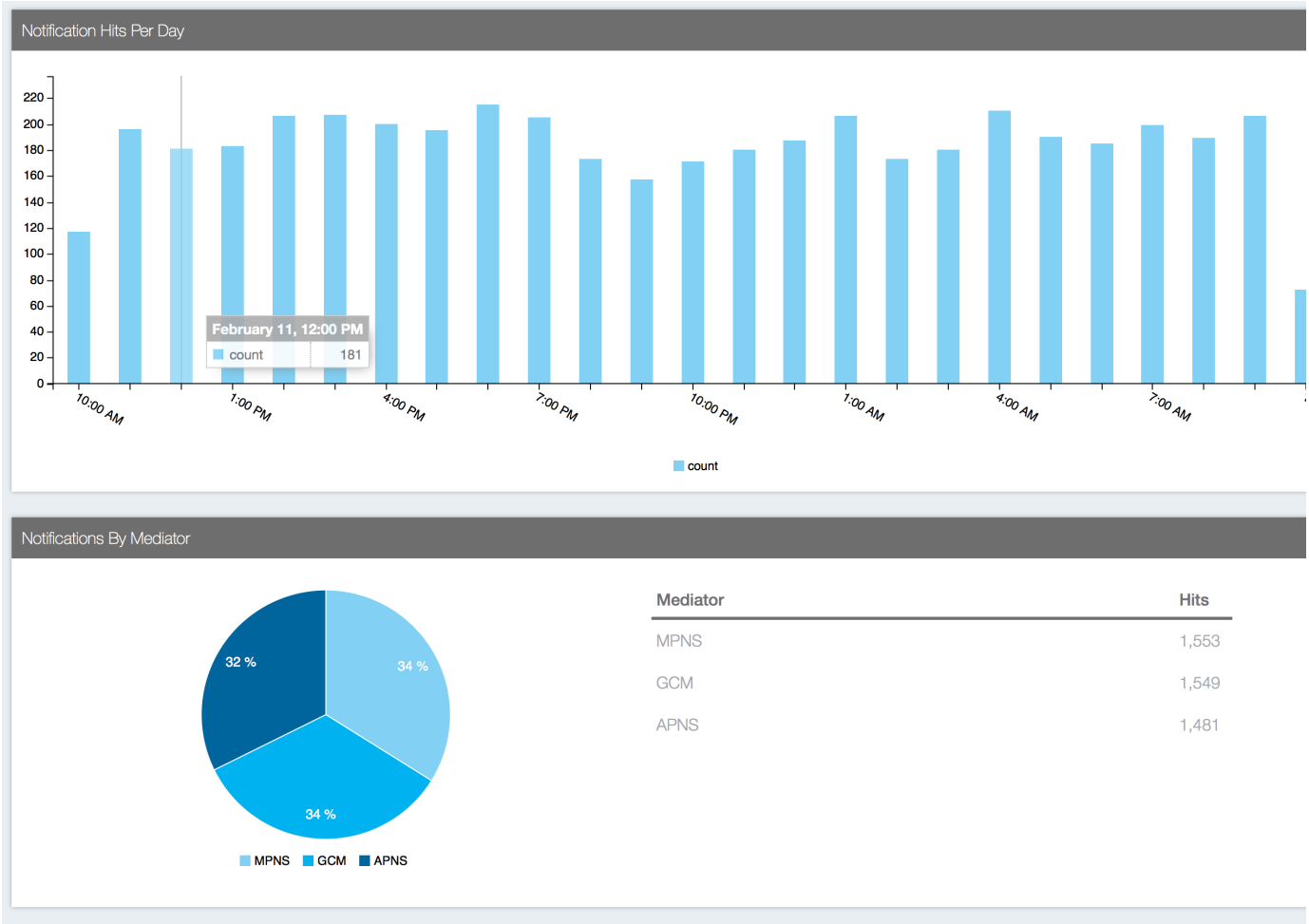
- The following API call results in an adapter hit that is visualized on the MobileFirst Operational Analytics dashboard:

```
// an 'adapter hit' will be recorded upon a successful adapter invocation
WL.Client.invokeProcedure({...});
```



Capturing data - Notification Activities

When a push notification occurs, the event is captured automatically and forwarded to the MobileFirst Operational Analytics server.



Capturing data - Server Logs

The log data that is generated by MobileFirst Server is automatically forwarded to the MobileFirst Operational Analytics server, where the data can be searched and downloaded.

To disable this behavior, set the `wl.analytics.logs.forward` property to `false`.

Capturing data - Client Logs

You can instrument a MobileFirst application with client logs to record client debugging information and events.

You can use the following APIs to create client logs which are then forwarded to the MobileFirst Operational Analytics server, where they can be searched and downloaded.

```
// Set the log level to trace so that all logs are captured
WL.Logger.config({"level": "TRACE"});

// Create a client side log that is persisted locally until it is sent to the server
WL.Logger.trace("Create a client log at the TRACE level.");
WL.Logger.debug("Create a client log at the DEBUG level.");
WL.Logger.info("Create a client log at the INFO level.");
WL.Logger.warn("Create a client log at the WARN level.");
WL.Logger.error("Create a client log at the ERROR level.");
WL.Logger.fatal("Create a client log at the FATAL level.");
```

Analytics Logs

Client-side logs are captured based on the logging level that is set on the client. If you want to create analytics logs that are always captured regardless of the logging level, you can use the `WL.Analytics` API.

```
// Create an analytics log message
WL.Analytics.log("Analytics log message");

// Create a custom activity
WL.Analytics.log({_activity: "customActivity"});

// Create a custom activity with a log message
WL.Analytics.log({_activity: "customActivity"}, "Analytics log message");
```

Sending data

Logs that are captured by the client-side logging APIs and the `WL.Analytics` APIs are sent to the server automatically upon a successful server connection or a successful adapter call.

```
<p>// Logs sent upon successful connection
WL.Client.connect();

// Logs sent upon successful adapter invocation
WL.Client.invokeProcedure({...});
```

You can disable this automatic behavior by using the following call:

```
// Disable automatic sending of client and analytics logs
WL.Logger.config({autoSendLogs: false});
```

If you want to send this data more frequently, you can use the following API calls:

```
// Send client debug logs
WL.Logger.send();

// Send analytics logs
WL.Analytics.send();
```

Custom charts

Custom charts are a new feature of MobileFirst Platform Operational Analytics 7.0. By using custom charts, you can take data that is already collected, like device ID, device model, device OS, etc., or log your own custom data and create charts. To understand how to log and send data, see Analytics logs and Sending data.

Chart types

- Bar Graph
- Flow Chart
- Line Graph

- Metric Group
- Pie Chart
- Table

Creating a custom chart

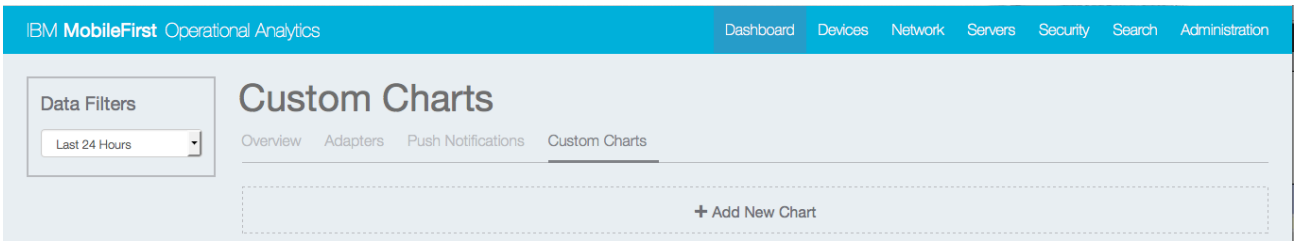
Creating a custom chart is simple. The following example walks you through creating a pie chart, based on your user location.

The messages that are logged to the Operational Analytics server in this example are hard-coded locations. Those messages look like this:

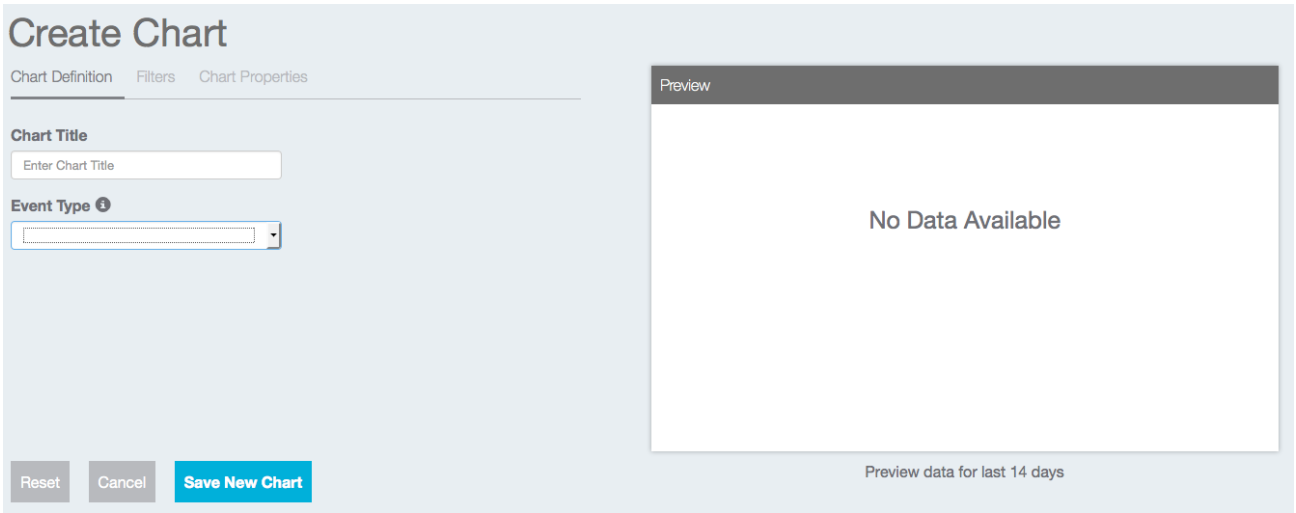
```
WL.Analytics.log({location: "USA"}, "visit");
WL.Analytics.log({location: "Mexico"}, "visit");
WL.Analytics.log({location: "Canada"}, "visit");
```

To create a chart, follow these steps.

1. Go to the **Custom Charts** tab and click **Add New Chart**.



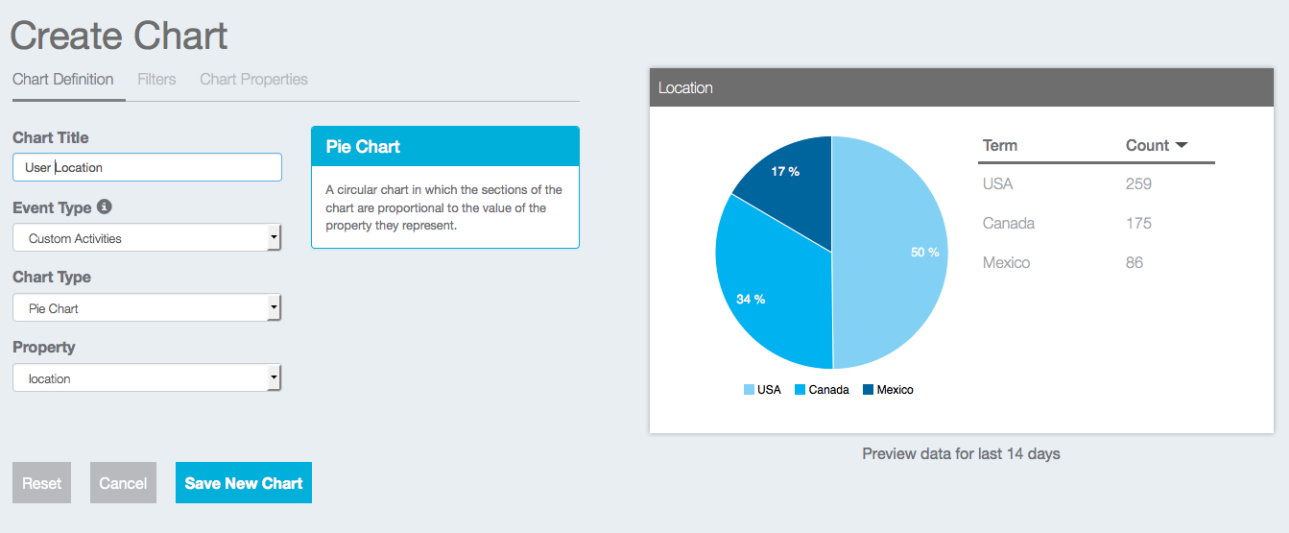
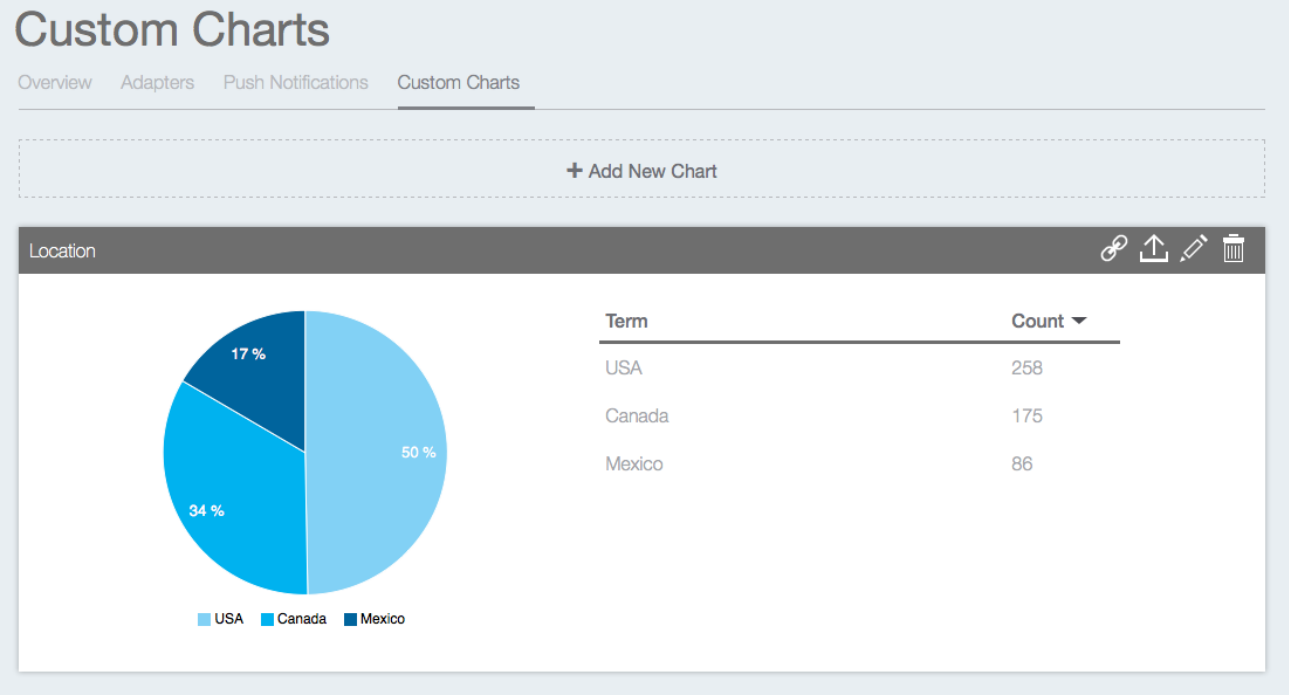
   The following dialog opens:



   As you fill out your information, more input fields are displayed.

2. Type a chart title.
3. From the **Event Type** menu, select **Custom Activities**. No other option creates charts from the custom data sent to the server.

4. Select a chart type. This example uses a pie chart.
   The last menu that becomes available depends on the chart type that you selected. In this example, the **Property** menu contains a list of the custom data that you created to send to the server, plus a few more. In this example, the **location** property is selected, as shown below.



5. Click **Save New Chart**. The chart is saved under the **Custom Charts** tab in the main dashboard.