SOASTA 53.0.4 (mPulse 7419.158.11)

Oct 31, 2014

Table of Contents

SOASTA 53.0.4 (mPulse 7419.158.11)

Bugs Fixed

85950: Scroll bars show up in screenshots

This visual layout issue occurred when the Date/Time filter was too wide at the widget-level filter bar

85739: Conversion rate incorrect in Summary widget

The underlying calculation was not as expected.

85066: Drop down text garbled in widgets

Display text in widget drop-downs was not legible.

SOASTA 53.0.3 (mPulse 7419.158.10)

Bugs Fixed

85838: Czech Republic regional zones not appearing correctly

The CZ regional code was improperly shown.

85801: Update the valueList when updating filters from popup

The possible values weren't populated as expected.

85772: Top N tables not properly sorting child rows

Child rows would not always obey the sort after expanding a parent row or two and applying sort in the Top N widgets.

85708: Enable User Agent Family and OS Family for Measurements Over Time widget

The widget's filter now includes the additional OS Family and User Agent Family.

86793: Remove the "compare to” filter from the UI

The filter is now disabled.

85545: Using a percentile slider on the surface of a widget causes a JS error

Using the Percentile slider resulted in this error.

85918: Attach click handlers even if the mobile carriers table isn't there.

Non-mPulse or non-admins were unable to view/clear their saved dashboard filter preferences. This is resolved.

85066: Drop-down text garbled

The display of text in a drop-down was illegible.

85038: Using timers-metrics in mPulse API return HTTP 500 error

Some mPulse API calls returned HTTP 500 unexpectedly.

83247: mPulse Error dialog says CloudTest

The CloudTest brand unexpectedly appeared in an mPulse Error Dialog.

77394: Dropped beacon logs not consistently uploaded to S3

SOASTA 53.0.2 (mPulse 7419.158.8)

Enhancements

mPulse Aggregate API now includes percentile & timer parameters

The following optional Query Type parameters are now available for use by mPulse users:

percentile	The `percentile` can be any positive numeric value between 1 and 99.
timer	The `timer` parameter can be one of the following: • t_done: Page Load Time • t_page: Frontend Time • t_resp: Backend Time • dns: DNS Time • tcp: TCP connect Time • ssl: SSL negotiation Time • domLoad: Time from navigation Start to domLoading • domReady: Time from navigation Start to domComplete • Also, any custom timers as custom[0-9]

Refer to mPulse Aggregate API for a complete, current reference.

Bugs Fixed

85713: Hide Active Sessions

The Active Sessions feature is disabled until issues are resolved.

85624: Cannot get 90th percentile using API

The new percentile query type parameter allows mPulse Aggregate API users to specify any percentile from 1-99.

85539: PageLoad element does not report value in alert

The Page Load element found in the Alert workflow unexpectedly failed to produce a report value.

85537: PageLoad element causes error in previously defined alerts

The Page Load element's presence in an edited alert caused this error.

85530: TopN Dashboard - Child rows have no session data

Child rows in the Top N widget unexpectedly had no data.

85038: Using timers-metrics in mPulse API return HTTP 500 error

An attempt to use the timers-metrics parameter in an API call unexpectedly resulted in a 500 error.

84854: Uncaught Type Error

An error occurred while trying to filter custom metrics by date.

SOASTA 53.0.1 (mPulse 7419.158)

Features

mPulse Mobile for iOS

SOASTA mPulse™ Mobile combines the power behind SOASTA's real-time analytics with real user measurement (RUM) beacons for native iOS and mobile web sites in iOS.

SOASTA provides the Add MPulse utility to inject the mPulse Library into the mobile app. By injecting the mPulse Library into a mobile app, users can collect beacons and get session analytics and performance data from real users.

mPulse Library injection can be done dynamically (into an IPA or APP) or statically (into an Xcode project). Dynamic instrumentation requires iOS 6 or later. Static support is provided for earlier iOS versions.

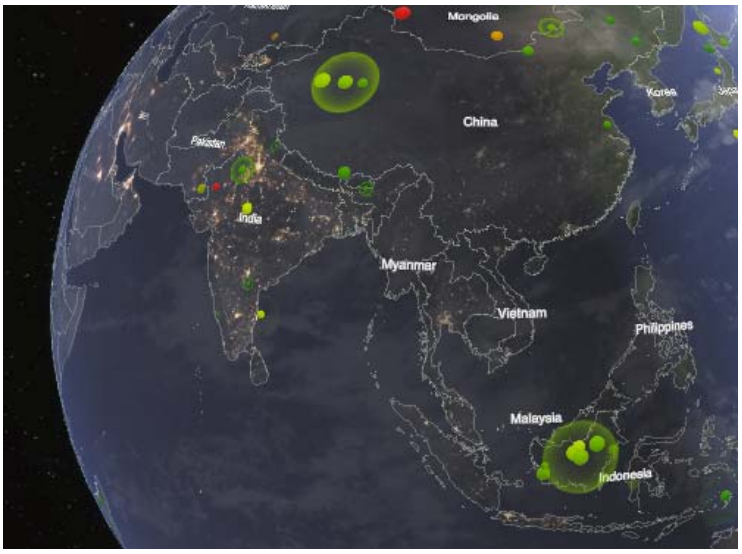
With mPulse Mobile, you control the rate at which your distributed mobile app sends beacon data so you can ensure a light footprint for your users and still collect valuable beacon data. Refer to the "Seed data no more than once every [x] seconds" setting in the Configuring Your Mobile App section below.

In the following sections, we'll cover all the steps necessary to inject the mPulse Library into a mobile app(s).

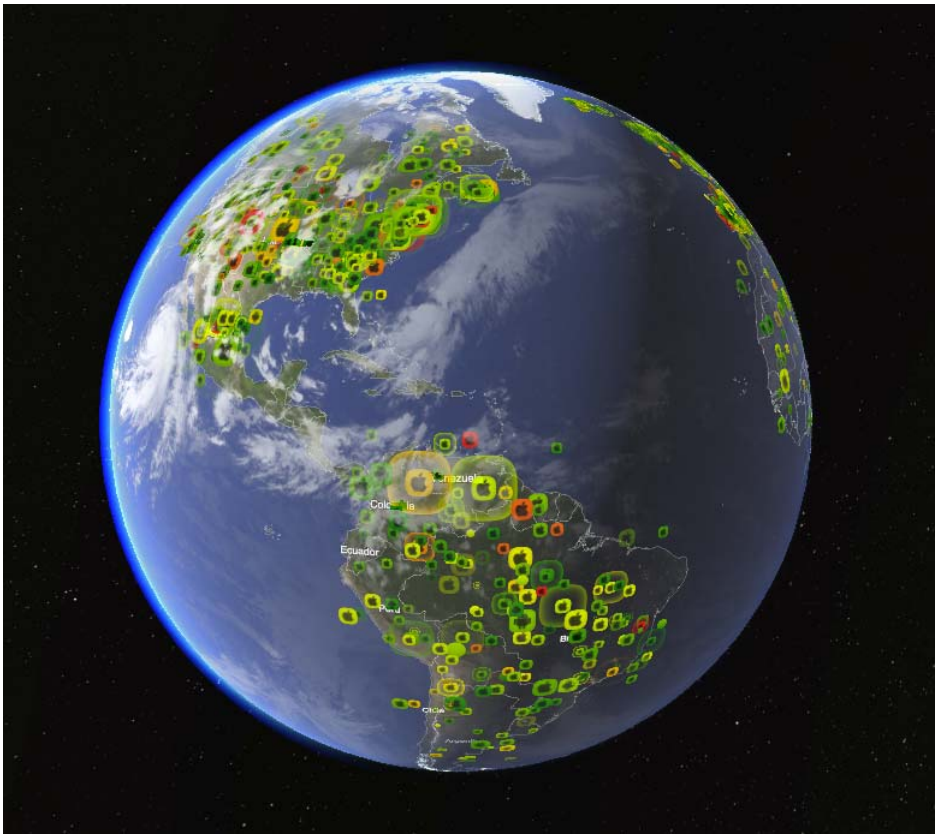
Distinct Look for Mobile Beacons

With the addition of mPulse Mobile, a new mobile beacon type has been introduced to further the visual distinction between the source(s) of your beacons while viewing the Globe Dashboard (or any custom dashboard using the Globe).

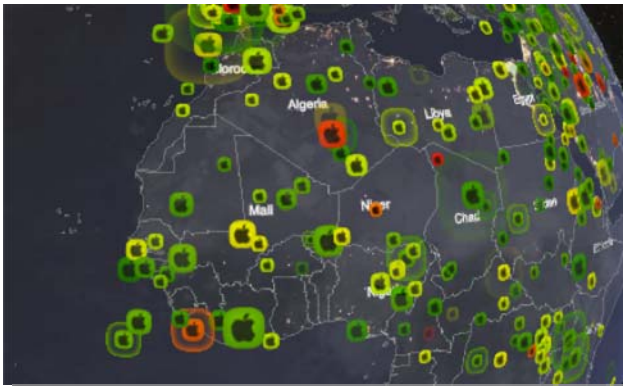
- For web apps, beacons retain the rounded appearance familiar to users of earlier mPulse releases



- For the new mPulse Mobile support, there is also new mobile beacon style (shown below), using a squared off visual style. Intensity is indicated by the size of the beacon pulse just as it is for mobile beacons.



  - Note that iOS beacons are further distinguished by the logo inside the square



Touch Metrics & Touch Timers (iOS)

mPulse Mobile provides Touch Metrics and Touch Timers in order to collect beacons based on specific *elements* that originate in your mobile app.

So, for example, if you define a Touch Metric or Touch Timer as "When user taps on "Add To Cart" icon, send us the value of "Cart Total," mPulse Mobile will scan for those two elements. Scanning is done only when the view changes.

Once you've identified the actions and conditions to track and from which to extract data, both Custom Metrics and Custom Timers will utilize the settings found in the Configure App box at runtime.

Using Touch Metrics and Touch Timers, it is no longer necessary to wait until your mobile app is re-submitted to the App Store for metrics and timers to be updated.

Using the Configure Apps box Custom Metrics and Custom Timers tabs, metrics and timers can be easily assigned to specific actions and conditions in your mobile app. These selections are then added to your mPulse app configuration and are published within your app when that time comes.

Custom Metrics

Customization of the following accessors, based on either their Actions or Conditions, is supported in this release.

All of these apply to both iOS:

- Action-based
    - Tap
    - Double-tap
    - Pan
    - Pinch
    - Rotate
    - Swipe
- Condition-based
    - elementPresent
    - elementNotPresent
    - elementValue
    - elementPropertyValue
    - elementVisible
    - elementNotVisible
    - elementText

    - elementCount
- Fixed metric values (can be assigned by entering the value into a provided field)
- Accessors
    - elementValue
    - elementPropertyValue
    - elementText
    - elementCount
- Page group filtering for action-based and condition-based
- Programmatic metrics, timers, and view groups

Custom Timers

All of these apply to iOS:

- Action-based
    - Tap
    - Double-tap
    - Pan
    - Pinch
    - Rotate
    - Swipe
- Condition-based
    - elementPresent
    - elementNotPresent
    - elementValue
    - elementPropertyValue
    - elementVisible
    - elementNotVisible
    - elementText
    - elementCount
- Page group filtering for action-based and condition-based
- Programmatic metrics, timers, and view groups

As part of mPulse Library injection, your app is registered in mPulse Central using both its URL Scheme field and its bundle ID. Additionally, a unique API Key is created for use by mPulse.

Using the Add mPulse Utility (Developer Only)

The App Admin user injects the mPulse Library into the mobile app using the Add MPulse utility. AMP will configure your app, and create a new Mobile App object in the mPulse repository. The developer can also administer the resulting app(s) object in mPulse Central (or, another App Admin user can be assigned that task).

When you inject the mPulse Library using the provided Add MPulse (AMP) utility, a corresponding Mobile App is created in mPulse Central. This app contains a Launch URL that is created automatically (if not specified by the App Admin), and in the case of static projects, can be located in the Xcode Project's Info tab, URL Scheme field (for the given Xcode project target).

The Mobile App object created will have the auto-created URL Scheme in its Launch URL field unless specified using the `launchurl` flag.

TIP: In the basic example, we do not use AMP's `launchurl` flag to create a launch URL. In which case, AMP will auto-generate the URL for us. If the flag is used, be sure to avoid spaces and underscores, as they will cause an error.

The Mobile App object also has a Mobile Bundle ID (that corresponds to your project's bundle ID).

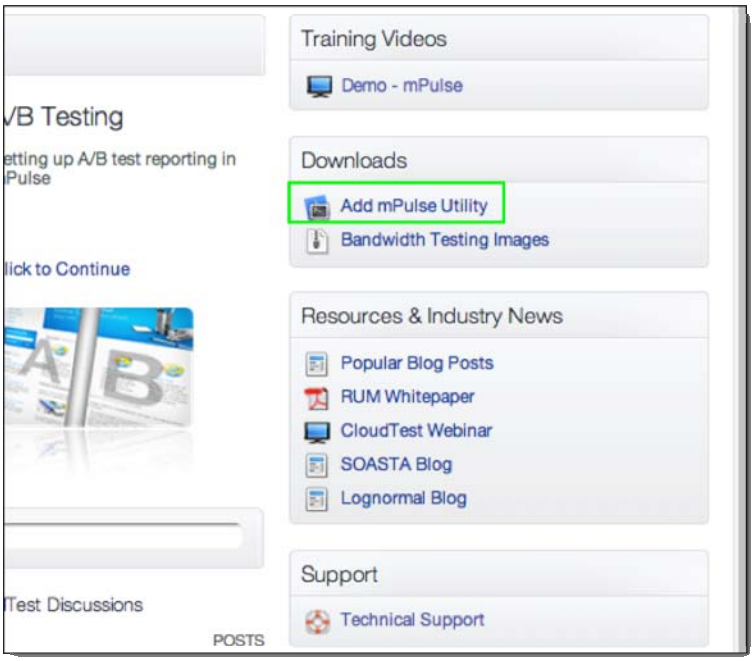
A unique API key is also provided by AMP and is used in the beacon collection process to identify your mobile app.

As noted above, the Add MPulse (AMP) utility supports two instrumentation methods.

- In Dynamic Instrumentation, AMP is used to inject the mPulse Library into an IPA or APP bundle file (for example, Stockfish.ipa or Stockfish.app). As noted above, this dynamic method requires iOS 6 or later.
- In Static Instrumentation, AMP is used to inject the mPulse Library into an iOS project (i.e. Stockfish.xcodeproj, etc.) before you compile it.
- In either case, the injection creates a mobile app, which can be in the mPulse > Central > Apps list.

Downloading Add MPulse

- Login to your mPulse instance and open Central > Welcome.
- In the Welcome Page's Downloads section, locate and click the *Add MPulse Utility*.



- Unarchive the ZIP file. This archive contains the AddMPulse.jar that is used to perform the mPulse Library injection. Add MPulse.jar utility is the script that will make the necessary project modifications and create a mobile app in mPulse®.

Note: Take note of the Add MPulse folder. You'll need its location for use in the command line. SOASTA recommends you maintain a designated Add MPulse folder. Occasionally, you'll need to download a new Add MPulse version, so having one location for this utility is the best practice.

Dynamic Instrumentation of an IPA

To instrument an IPA using the AddMPulse utility, run:

```
java -jar AddMPulse.jar -ipa <compiled IPA> -bundleid <bundle ID> -url
<mPulse URL> -username <mPulse user name> -password <mPulse password>
```

where:

- `<compiled IPA>` is the full path to the compiled iOS IPA file to use

Note: One and only one of the project file, IPA, or app file can be specified when using the AMP utility.

To dynamically instrument an APP using the AddMPulse utility, run:

```
java -jar AMP.jar -appbundle <compiled APP bundle folder > -
bundleid <bundle ID> -url <MPulse URL> -username <mPulse user
name> -password <mPulse password>
```

where:

- `<compiled APP bundle folder>` is the full path to the compiled iOS .app file
- `<bundle ID> is the Xcode project Bundle Identifier`

Required parameters (dynamic and static):

`project <projectpath>`	`Path of the Xcode project directory (e.g. ~/Documents/MyApp/MyApp.xcodeproj)`
`target <name>`	`the name of the Xcode target to modify. Usually, a copy of the`
`url <url>`	`the mPulse URL (e.g. http://ctserver/concerto`
`bundleid <bundleid>`	`the Bundle ID of the application`
`username <username>`	`The mPulse user name`
`password <password>`	`the mPulse password`

IPA/APP wrapping parameters:

`-ipa <ipafilepath>`	`path of the iOS IPA file (e.g. ~/Documents/MyApp.ipa)`
`-appbundle <appbundlepath>`	`: path of the iOS app bundle directory (e.g. ~/Documents/MyApp.app).`

IPA/APP optional parameters:

The following optional parameters are also available for dynamic instrumentation of IPA files (in some cases their use may be necessary to succeed):

`-signingidentity <signingidentityname>`	The name of the signing identity to be used for codesigning the application (e.g. "iOS Distribution: Developer Name").
`-provisioningprofile <profilepath>`	Path of the Provisioning profile to be used for building IPA file.
`-entitlementsfile <entitlementsfilepath>`	Path of the entitlements file to be used for codesigning the application.

Static Instrumentation of an Xcode project

1. From the Add mPulse folder, run:

```
java -jar addmpulse.jar -project <Xcode project directory> -target
<target name> -bundleid <bundleid> -url <MPulse URL> -username
<mPulse user name> -password <mPulse password>
```

   where:

   - `<Xcode project file>` is the path to the ".xcodeproj" file representing your project
   - `<target name>` is the name of the Xcode target you would like to modify. For example, a copy of the main Xcode project target.
   - `<bundle ID>` is the Xcode project Bundle Identifier

   Here is a complete example:

```
java -jar addmpulse.jar -project
~/Documents/Demo/Stockfish/Stockfish.xcodeproj -target "Stockfish
copy" -url http://mpulse.soasta.com/concerto -username
bob@acme.com -password secret
```

Add mPulse will configure your project, and create a new Mobile App object in the mPulse server repository.

The Mobile App object created will have the auto-created URL Scheme in its Launch URL field. You will see a message similar to the following:

```
Mobile App Object representing your Application "Stockfish" has been
created in mPulse Repository.
```

If the mobile app already exists, it is not overwritten.

```
Configuring Target in Xcode Project...

Adding URL Types to Info Plist file of Automation Target...

Already Have an App Object By This Name. Please Change The Name And Try Again.

You should manually create an object corresponding to your Application using Launch URL: touchtest-b38ddb30-a7ad-42b4-86d8-c957d6ac6237://
```

In order for the mPulse Library injection to work, the URL Scheme found in your project must match the Launch URL found in the Configure Domain or App box in mPulse. Be aware of other team members before overwriting an existing app/Launch URL combination.

Other AMP optional parameters:

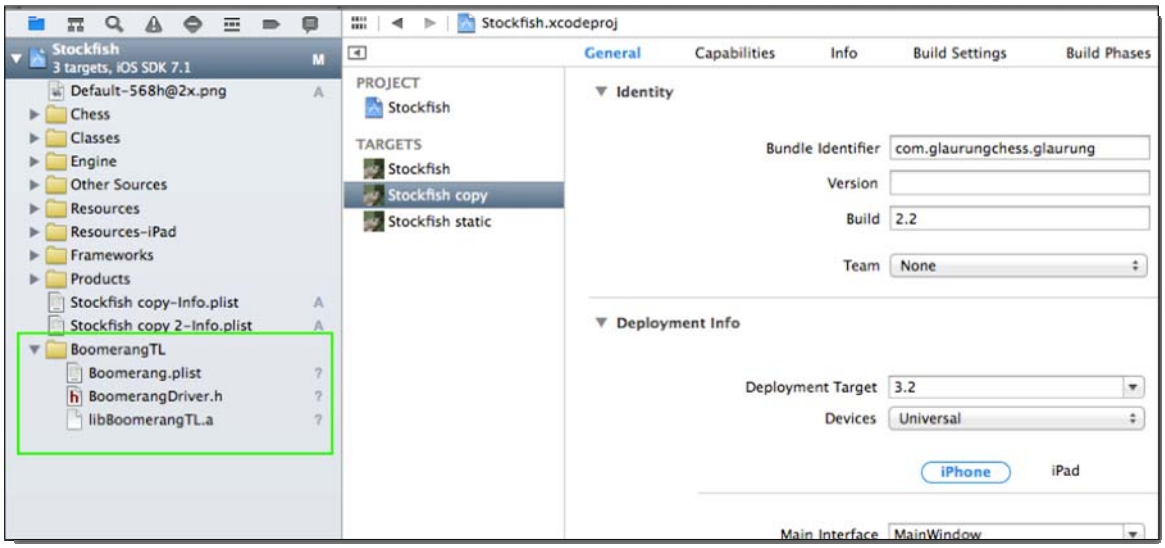
The following additional parameters are also provided for the described purpose.

`-apikey <apikey>`	the mPulse API key for this project
`-tenant> <tenant>  :`	the mPulse tenant the user is associated with
`-infoplistfile`	the project's Info.plist file If this parameter is not included, then AddMPulse will try to automatically locate it.
`-customrulesfile`	The project's custom_rules.xml file If this parameter is not included, then AddMPulse will try to automatically locate it.
`-launchurl <URL Scheme value to use>`	the URL that mPulse should use to start the app. Must use lowercase letters. Avoid spaces and underscores. Example: `my-app://launch` If this parameter is not included, then AddMPulse will auto-generate a URL based on the source project URL Scheme.
`-appobjectname <name>`	The name of the Mobile App object to

	create in the mPulse server.
-donotcreateapp	Don't create a Mobile App object in the mPulse server.
-previewmode	Run in Preview mode to examine the console output for these parameters. If this parameter is not included, then AddMPulse runs in full mode and the project is changed.
-hudenabled	Enable HUD for the mPulse Library
-generatenetworkerrors	Generate random network errors at regular intervals
-proxyserver <server>	HTTP proxy server name
-proxyport <port>	HTTP proxy port number
-proxyusername <username>	HTTP proxy user name (if any)
-proxypassword <password>	HTTP proxy password (if any)
-version	Print the Utility Version
-reporterrors	The utility will automatically report errors to SOASTA along with any relevant files.
-donotreporterrors	The utility will not report any errors to SOASTA.
-addheadersearchpath	Add header files to "Header Search Paths" build setting.
-useforceloadlinkerflag	add -force_load flag to "Other Linker Flags" build setting.
-removelibraryfrombuildphase	if user does not want the library to be added to "Link Binary With Libraries" step of Build Phases, this argument will prevent that.

Xcode Project Changes Made by AMP (Static)

When Add MPulse is used to inject the mPulse Library into your Xcode project via static instrumentation, users will note a new BoomerangTL folder in the refreshed project tree.



This folder contains the necessary mPulse Library info and should not be removed.

Register Your Device to Use mPulse™ (Developer/App Admin)

The mPulse™ Agent is responsible for launching the app while configuring the app in the Configure App. For example, you'll need it when applying actions of conditions using Touch Locator.

The agent is a web application that is run in mobile Safari on iOS devices. To get started, browse to the mPulse Agent URL below on the mobile device and then perform the one-time registration steps that will enable your device for use with mPulse.

Note: If you clear your cookies on the given mobile device after registration, you may need to register your device again so that mPulse can recognize it. This does not consume an additional license.

1. On the mobile device, enter the mPulse URL with “/mpulse” appended. The complete URL form for mPulse is:

   http://mpulse.soasta.com/concerto/mpulse/

TIP: If you're using a Simulator that is prior to iOS 7, use the "Tap here if this is a simulator" link to proceed. This link will appear below the Login button in all configurations that require it. This link doesn't appear in iOS 7 or after (the shot below shows an iOS 5.1 simulator).

If the device is not registered, the Register Device page below appears.

1. Login using your SOASTA mPulse user name and password.

TIP: If you clear your cookies, you may need to register your device again so that mPulse can recognize it. This does not consume an additional license.

2. When prompted, enter an mPulse Agent a name (this name will appear in Central drop-down lists. For example, *SOASTA Demo iPad*.

The Unique Device Identifier (UDID) will be used to register the mobile device for use with mPulse™ in iOS7 or after; for earlier iOS versions, the registration will also use a profile.

2. Click the Connect button to continue.

Once the mPulse Agent is connected, its status will change to *Connected*. If it is not *Connected*, refresh the page, and ensure that you are logged in.

Once connected, login to mPulse Central and open the mobile app object in the repository.

Configuring a Mobile App (App Admin)

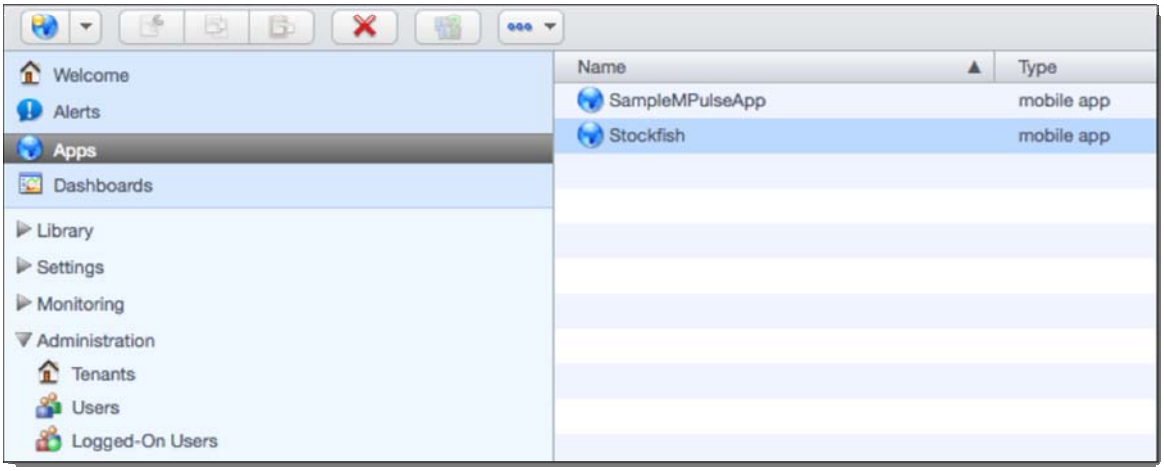
Mobile App creation in mPulse will start with the Add MPulse utility in the vast majority of cases. App Administrators can modify existing mobile app configurations or, in some rare cases, create new mPulse mobile apps from within Central. In most cases, a user with the App Admin privilege will use Add MPulse to create apps in Central while also inject the mPulse Library into the resulting IPA or APP. If you will be creating your mobile app in Central first, refer to the manual app creation instructions in the following section.

By using Add MPulse, the characteristics of the Xcode project settings (URL Scheme, Bundle ID, etc.) are also found in the Add MPulse utility.

If necessary, mobile app administration can also be accomplished in Central (after which, the developer would conform to the prescribed values when injecting the mPulse Library using AMP on the command line).

1. To get started, login as the user with App Administration rights.
2. After the mPulse Dashboard has loaded, click the Central tab and select the Domain or App node (the third node in Central).
3. Click New (top left toolbar) to create a new mobile or web app.

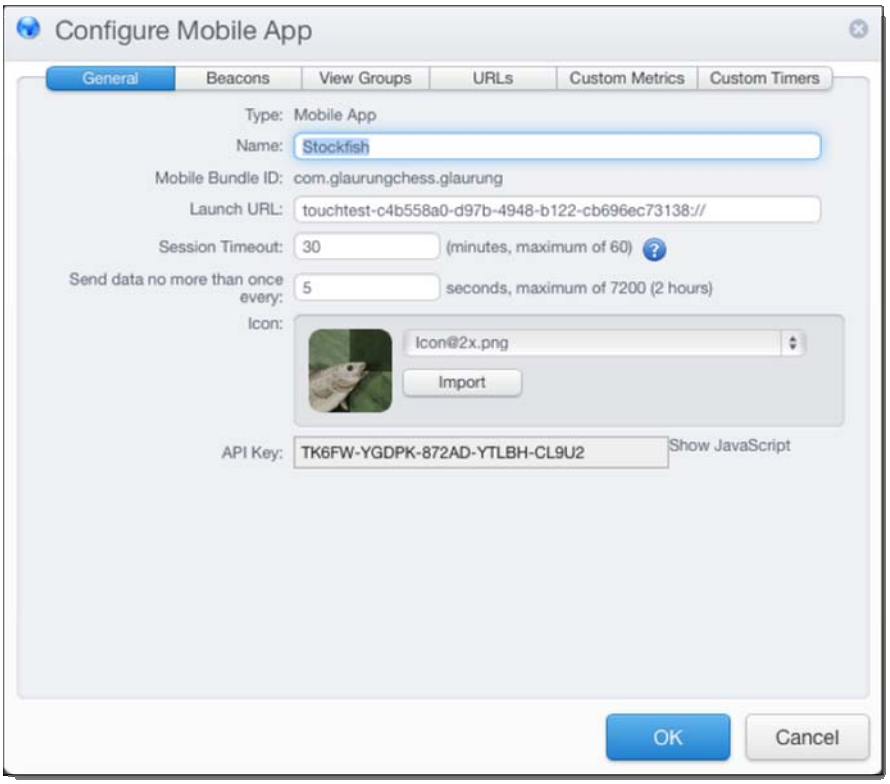
   Or, open an existing app in the Apps list.



When you do so, the Configure App box appears with the General tab shown. This tab contains the essential basic settings for the mobile app entry (and the deployed mobile apps that had mPulse Library injected).

If you're creating a mPulse Mobile App from the server side before using Add MPulse, use the Configure App, General tab to accomplish the necessary tasks: you will need a Name, Mobile Bundle ID, a Launch URL, and an API Key, all of which will need to be specified correctly while using Add MPulse at a later time. Refer to mPulse Setup for additional information.

In the screenshot below, the details of the Stockfish app with mPulse Library (via AMP) applied are shown.

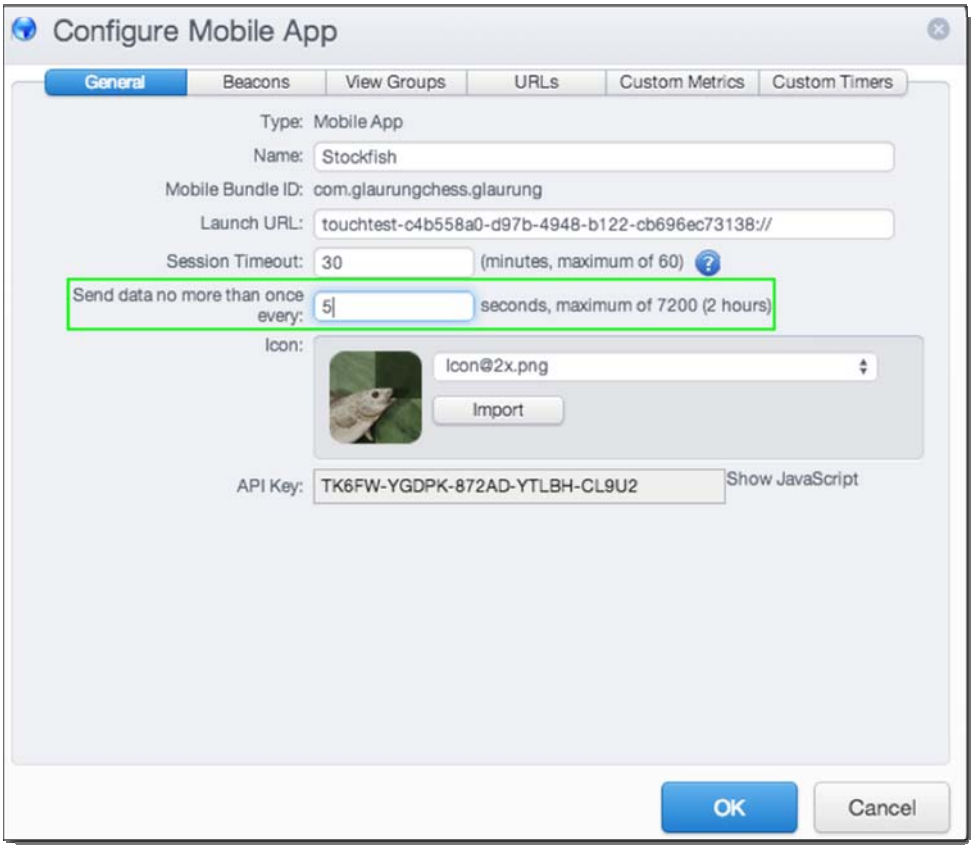


The mPulse mobile app entry we created using Add MPulse has all the required settings (ones that can be specified from the command-line using Add Mpulse):

- The name of the app (The name of the Xcode project, IPA, or APP from which the mPulse mobile app was created)
- The Mobile Bundle ID (specified by the Add Admin while injecting the mPulse Library & also found in the project)
- The Launch URL (specified by the App Admin, or inherited from the Xcode project's URL Scheme, if not specified)

Note: This also depends on whether mPulse already has an entry for the given mobile app name (e.g. created earlier using the same project details), Bundle ID, and LaunchURL combination.

- The Send data no more than once every [x] seconds is used to control the beacon collection footprint in your mobile app. For more about optimal use of mPulse Mobile, including Touch Metrics, Touch Timers, and View Groups, refer to the "Using Touch Timers and Touch Metrics Optimally" section below.



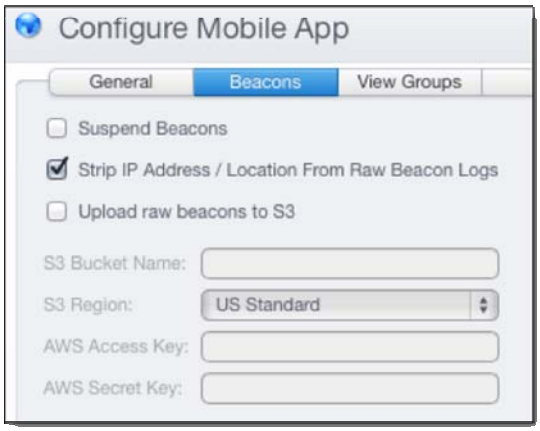
- The API Key is auto-generated by Add MPulse

The other General tab values shown here are important default values (ones that also apply to Domains):

- Session Timeout (this default is an important one). The mPulse default session timeout is 30 minutes but can be tweaked to a maximum of 60 minutes. See Session Metrics and Session Timeouts for more information.

Beacon Settings for Mobile and Web Apps

Beacon settings that apply to your Mobile or Web App are found in the Configure App box, Beacons tab.



- Check the Suspend Beacons box if you'd like to stop beacon collection for this app.
- Note that the Strip IP Addresses from Raw Beacon Logs setting is used to ensure user privacy.

For more information about the other checkboxes, refer to Beacon Settings.

Using Touch Metrics and Touch Timers Optimally

mPulse™ Mobile Touch Metrics and Touch Timers are designed with the pragmatic cost of *using* them in mind. The more elements tracked, the longer the time spent to scan them.

If no elements are tracked, mPulse runs for only 1 millisecond per minute. This processing time is required to analyze Network Requests performed by the Application.

So, for example, if a Touch Metric or Touch Timer is defined as:

When the user taps the "Add To Cart" icon, send us the value "Cart Total"

Then, those 2 elements are scanned at the appropriate time(s). Each additional metric, timer, or view group adds to this baseline.

App Administrators should be cognizant of the following additional details:

- A maximum of 10 Touch Metrics and 10 Touch Timers is allowed per Mobile App.
- For View Groups, no maximum limit is enforced. We tested up to 12 View Groups in use.

mPulse Mobile's scans for Touch Metrics, Touch Timers, and View Groups occur at the following times:

- Scanning is done when the view changes, or,
- When some new elements are introduced and a metric, timer, or view group is tracking such a change
- Periodically, at either the default rate of the Configuration data update, or by a substitute value that is specified in the General tab's "Send data no more than once every x seconds" field, or a user-defined custom value.

  The "Send data…" setting is used to control the beacon collection footprint in your mobile app. The range can be set to any value between the default (5 seconds) and maximum (7200 seconds). Changing this setting should be considered with the number of elements to track in mind.

SOASTA has tested extensively using the maximum configuration of 10 Custom Metrics, 10 Custom Timers, alongside 12 View Groups. In such a configuration, mPulse Mobile tracks 64 unique locators/elements during its scan(s).

While tracking 64 elements, and if the view doesn't change, mPulse Mobile will use the processor for 10ms every 500ms. If the view changes, then mPulse Mobile will consume ~400ms time in this configuration.

We believe this is the practical outer limit of advisable tracking and a figure half or less as much may be prudent for most use cases.

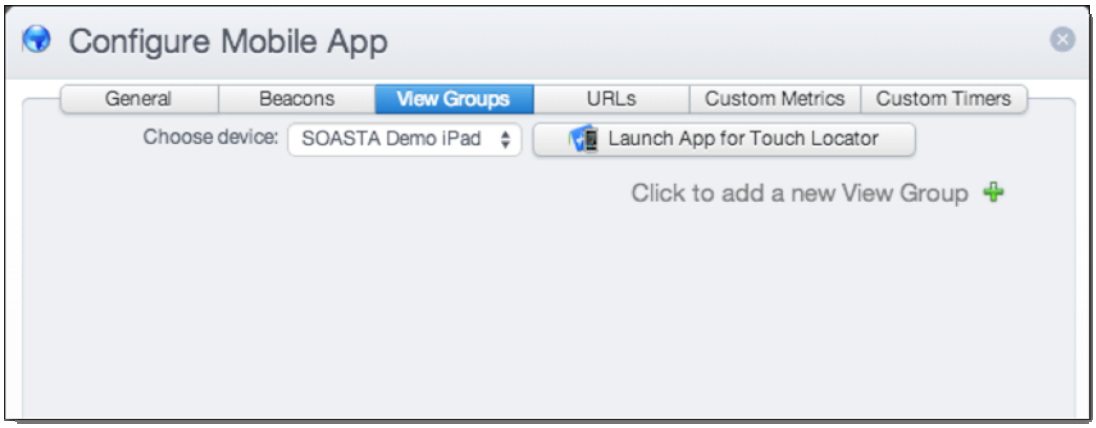
Touch Locator and Mobile App Configuration

Capture and summarize the performance characteristics of your mobile app's View Groups using mPulse Mobile. A View Group is a special view that can contain other views (e.g. children) and is the base class for layouts and Views containers. Beacon collection can be organized by the actions and conditions of a View Group via the Configure App box, View Groups tab.

View Groups Settings for Mobile Apps (App Admin)

The View Group settings that apply to your Mobile App are found in the Configure App box, View Groups tab.

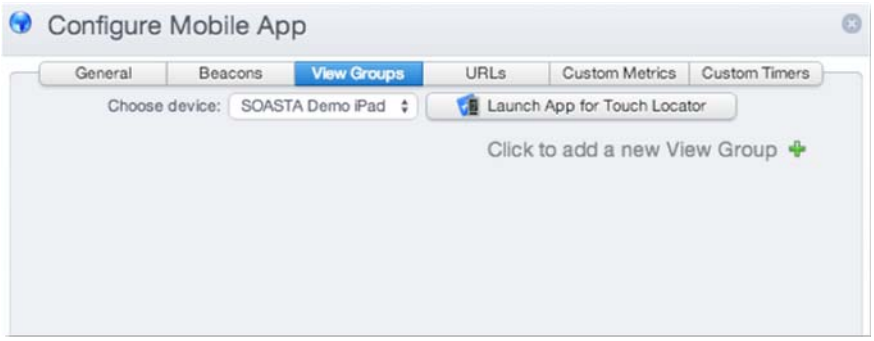
- The mPulse Agent must be active on the mobile device (shown below) that you'll use to add actions and conditions to this View Group.
- The mobile app to use exists on the mobile device (e.g. with the necessary mPulse Library code injected).



Launch App for Touch Locator

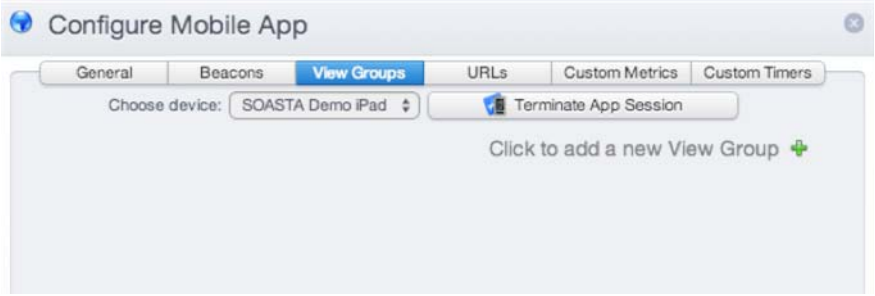
Ensure that the mobile app that has mPulse Library injected is installed on the mobile device used in the following steps.

1. In the View Groups tab, click the Launch App for Touch Locator button to initiate Touch Locator mode on the device.



The mobile app will launch on the device in Touch Locator mode, which is signified by a blue border.

In the Configure Apps box, the state also changes to indicate that Touch Locator mode is on.



The Terminate App Session button appears whenever Touch Locator mode is engaged.

Configuring a New View Group

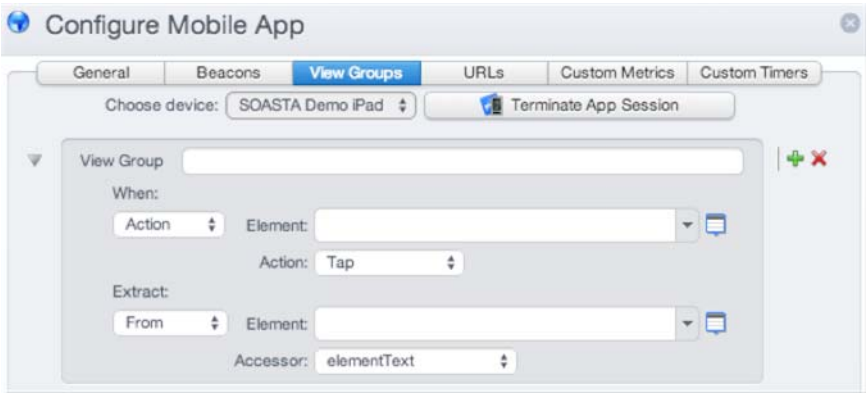
View Groups let you configure beacon data to collect from the specific parts of your mobile app by specifying those elements or accessors to track when actions or conditions are met.

Values are set on individual fields using the View Group form Element field (e.g. for either the When or Extract settings). Ensure that Touch Locator mode is on whenever you'd like to add an element.

- The mPulse Agent must be active on the mobile device (shown below) that you'll use to add actions and conditions to this View Group.
- The mobile app to use exists on the mobile device (e.g. with the necessary mPulse Library code injected).

1. Open the Configure App box, View Groups tab.
2. Click the green Plus (+) icon to add the View Group or navigate to an existing one you'd like to modify (e.g. by scrolling, if necessary).

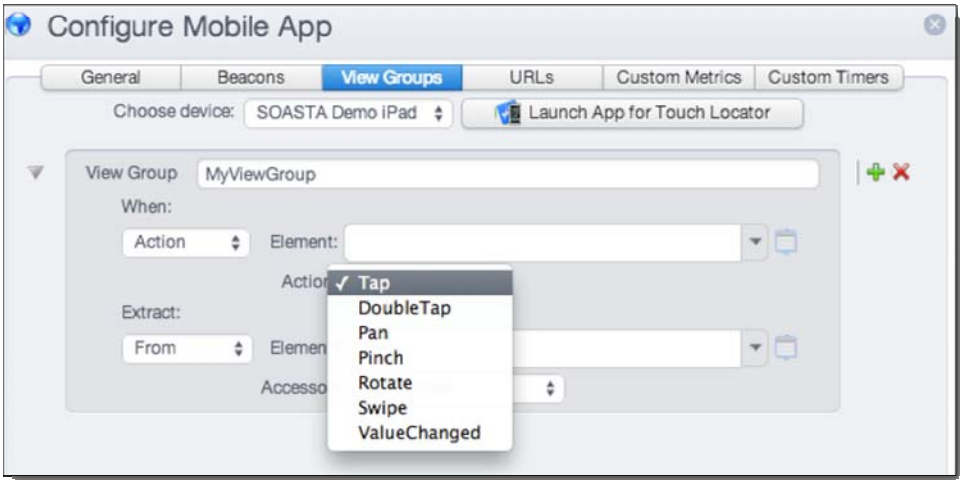
   The View Group form is added to the tab. The View Group tab can be used to configure as many groups as you'd like.



3. Give the new View Group a name.

4. In the subsequent steps, we will set *When* and *Extract* settings that will collect beacon data from the mobile app.

5. In the When dropdown, select either an Action or Condition and then specify the corresponding Action or Accessor to use. In the example below, we selected the Action, *ValueChanged*.



- An *Action* is a user action on some UI element.

  The following iOS Actions are supported:

  - Tap
  - Double Tap
  - Pan
  - Pinch
  - Rotate
  - Swipe
  - ValueChanged

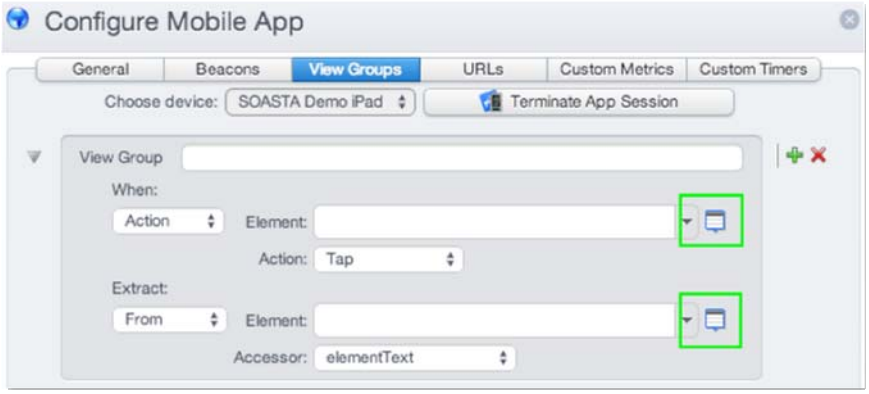
- A *Condition* is a state in the UI, such as the appearance of a UI element (following on some user action or workflow).

  Conditions can be set on the following iOS elements:

  - elementPresent
  - elementNotPresent
  - elementPropertyValue
  - elementVisible
  - elementNotVisible

- elementText
- elementCount

6. If the Element drop-down is already populated you can select from among the values. Otherwise, Click the active Touch Locator icon and capture the locator for the element to use.



7. Launch App for Touch Locator to open the app.

The mobile app will launch on the device in Touch Locator mode, which is signified by a blue border.

8. Select the UI element to identify its locator. The selection is highlighted blue and the available locators are listed in the Locator box.

When you do so, the Locator box appears with the item's view as its title and lists the currently selected object's locator(s).

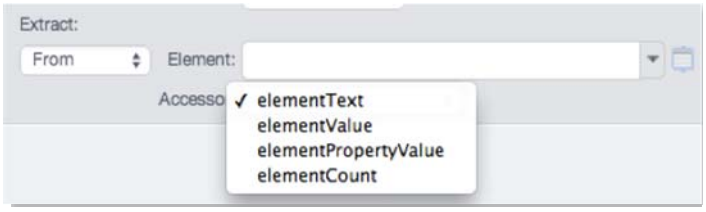
9. Tap the Use button (up arrow) in the top right of the box to select the current element or tap a different UI element to do the same..

- If the selection is the same as one that previously existed, then Locator mode is ended.
- If it is different or none existed before, a replacement is made and Locator mode is ended.
- You can select from among the populated values using the Elements drop-down (e.g. in Configure Apps) once all the possible values have been populated into the clip.

10. Next, define an *Extract* for the View Group form using the same techniques.

    Choose either *From* or *Fixed* for the Extract.

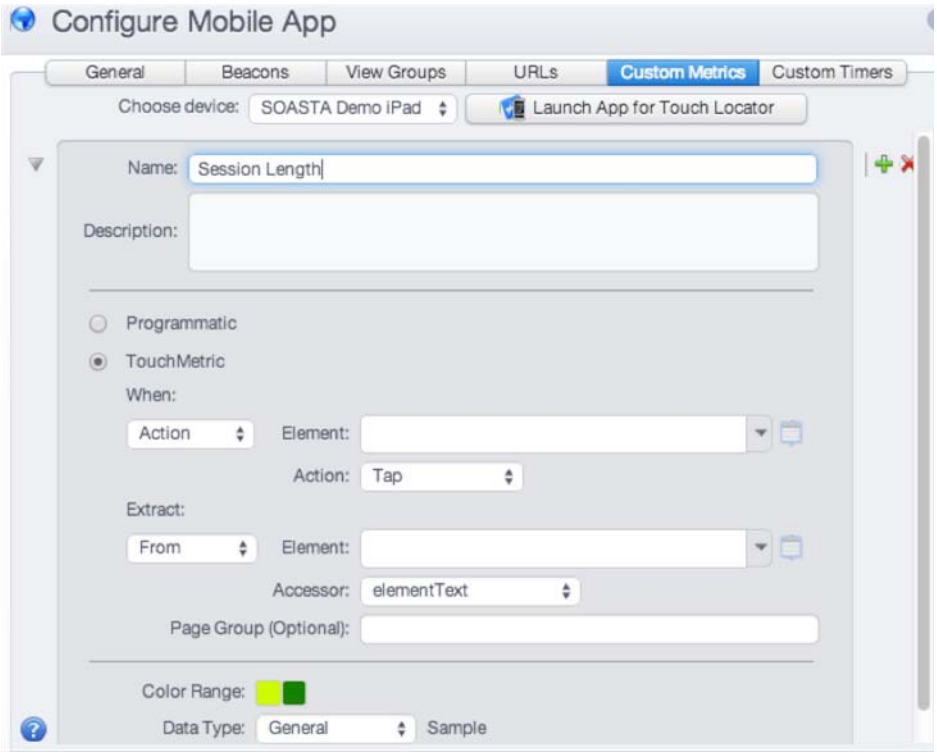
    - If From is selected, you must then specify the element to extract from using the Touch Locator (or by entering the correct value manually)



        - If necessary, click the Touch Locator icon in the given Element row a second time to re-enter Touch Locator mode on the device.
        - Specify the accessor to use to extract from among the following supported values:
            - elementText
            - elementValue
            - elementPropertyValue
            - elementCount
    - If Fixed is selected, enter the fixed value to extract in the provided field.
11. Click OK to complete creation of the View Group.

Custom Metric Settings for Mobile Apps

Custom metrics are user-defined values that refer to a business goal, or to a Key Performance Indicators (KPIs) such as revenue, conversion, orders per minute, widgets sold, etc. In mPulse Mobile, these values are defined using the Configure App, Custom Metrics tab.



- The mPulse Agent must be active on the mobile device (shown below) that you'll use to add actions and conditions to this View Group.
- The mobile app to use exists on the mobile device (e.g. with the necessary mPulse Library code injected).

1. Give the Custom Metric a name (i.e. Session Length)
2. Select either Programmatic or Touch Metric.
   - For Programmatic, select the radio button for the custom metric. Once you've added each of the metrics to use, add any additional programmatic metrics to use, and then click OK to complete the steps.

Note: When you create a new Touch Metric or Touch Timer with a programmatic one, ensure that the name matches the value you've used in your code. Refer to the "Developing for Custom Metrics and Customer Timers" section below for programmatic guidelines in your source project.

   - For Touch Metric, use the Touch Locator method to define the Action or Condition, Element, and Action or Accessor, etcetera to use.

TIP: SOASTA recommends using Touch Metrics because mPulse can match the current value found here at runtime. If programmatic is used, then that value can only be revised by re-submitting your app to the App Store.

3. In the When: field, set either an Action or Condition.
   - An *Action* is a user action on some UI element.

     The following iOS Actions are supported:
     - Tap
     - Double Tap
     - Pan
     - Pinch
     - Rotate
     - Swipe
     - ValueChanged
   - A *Condition* is a state in the UI, such as the appearance of a UI element (following on some user action or workflow).

     Conditions can be set on the following iOS elements:
     - elementPresent
     - elementNotPresent
     - elementPropertyValue
     - elementVisible
     - elementNotVisible
     - elementText
     - elementCount

12. For an action, set the action type to use and the Element to use from among the Element drop-down values.
4. Click the Touch Locator icon to enter Touch Locator mode on the device.

The mobile app will launch on the device in Touch Locator mode, which is signified by a blue border.

5. Select the UI element to identify its locator. The selection is highlighted blue and the available locators are listed in the Locator box.

When you do so, the Locator box appears with the item's view as its title and lists the currently selected object's locator(s).

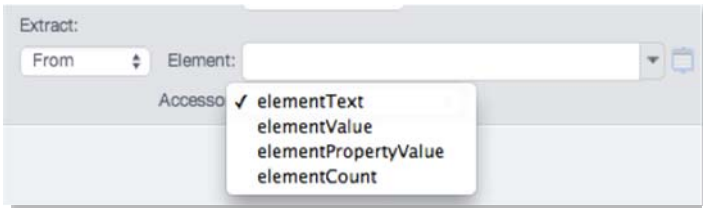
6. Tap the Use button in the top right of the box to complete the selection or tap a different UI element to list its locators.

- You can select from among the populated values using the Extract drop-down once all the possible values have been populated into the clip.

7. Next, define an Extract for the Custom Metric form using the same techniques.

   Choose either *From* or *Fixed*.

13. If From is selected, you must then specify the element to extract from using the Touch Locator (or by entering the correct value manually)



   a. If necessary, click the Touch Locator button in the Custom Metrics tab a second time to enter Locator mode on the device.
   b. Specify the accessor to use to extract from among the following supported values:

- elementText
- elementValue
- elementPropertyValue
- elementCount

14. If Fixed is selected, enter the value to extract in the provided field.

8. Optionally, add a Page Group by entering the text in the provided field. Refer to Configuring mPulse to use Page Groups for more about this important feature.
9. Click OK to complete creation of the Custom Metric.

Custom Timer Settings for Mobile Apps

mPulse supports the use of custom timers configured using either Programmatic or Touch Timer methods. Custom Timer settings that apply to your Mobile App are found in the Configure App box, Custom Timers tab.

1. Select the mobile app in Central > Apps and then click the Custom Timers tab.
2. Give the Custom Timer a name (i.e. SessionLength, etc.)
3. For Programmatic, select the radio button for the custom timer. Once you've added each of the timers to use, add any additional programmatic metrics to use, and then click OK to complete the steps.

Note: When you create a new, programmatic Touch Metric or Touch Timer, ensure that the name matches the value you've used in your code. Refer to the "Developing for Custom Metrics and Customer Timers" section below for programmatic guidelines in your source project.

4. For Touch Metric, use the Touch Locator method to define the Action or Condition, Element, and Action or Accessor, etcetera to use.

TIP: SOASTA recommends using Touch Timers because mPulse can match the current value found here at runtime. If programmatic is used, then that value can only be revised by re-submitting your app to the App Store.

5. In the Start Timer: section, set either an Action or Condition and then set the corresponding action or accessor.
    - An *Action* is a user action on some UI element.

      The following iOS Actions are supported:
        - Tap
        - Double Tap
        - Pan
        - Pinch
        - Rotate
        - Swipe
        - ValueChanged
    - A *Condition* is a state in the UI, such as the appearance of a UI element (following on some user action or workflow).

      Conditions can be set on the following iOS elements:
        - elementPresent
        - elementNotPresent

- elementPropertyValue
- elementVisible
- elementNotVisible
- elementText
- elementCount

6. For an action, set the action type to use or for a Condition set the accessor to use.
7. Click the Touch Locator icon for the Start Timer to enter Touch Locator mode on the device.

The mobile app will launch on the device in Touch Locator mode, which is signified by a blue border.

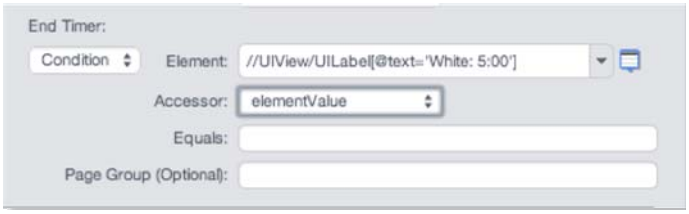
8. Select the UI element to identify its locator. The selection is highlighted blue and the available locators are listed in the Locator box.
9. Tap the Use button in the top right of the box to complete the selection or tap a different UI element to list its locators.

10. If necessary, click the Touch Locator button in the Custom Timers tab a second time to re-enter Locator mode on the device.

- You can select from among the populated value using the Elements drop-down once all the possible values have been populated into the clip.

11. Next, define an End Timer for the Custom Timer form using the same techniques.

Note: If you specified the Accessor elementValue in an End Timer, enter the expected value in the Equals field.



12. Click OK to complete creation of the Custom Timer.

Developing for Custom Metrics and Custom Timers

A programmatic metric or timer requires only a name match in the mPulse Configure Apps UI to go along with your project source code modifications for this feature. Unlike Touch Metrics and Timers, changes can only be applied by re-submitting your app to the App Store.

If you selected a Programmatic option for either a metric or timer, use the following steps to ensure a match between the named value here and that of your source code for the given timer or metric.

- `startCustomTimer` – provide the `timerName` value to match
- `stopCustomTimer` – stops the timer
- `setCustomMetric` – provide the `metricName` value to match

For example:

```
-(NSString *) startCustomTimer:(NSString *)timerName;
-(void) stopCustomTimer:(NSString *) customTimerID;
-(void) setCustomMetric:(NSString *)metricName value:(NSNumber *)value;
```

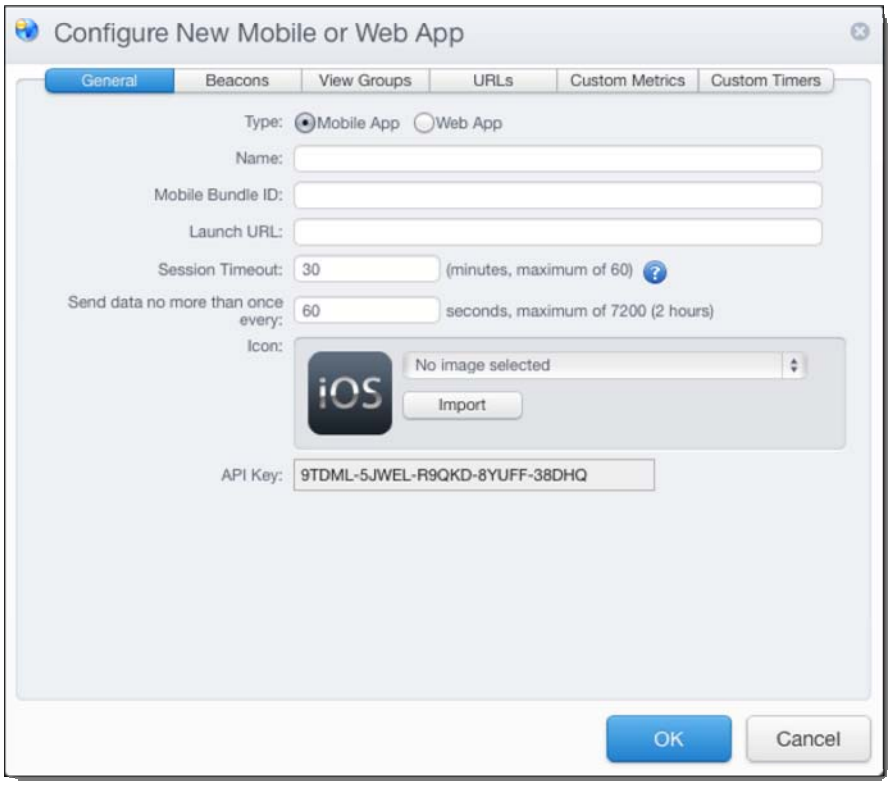
Both the `startCustomTimer` and `setCustomMetric` methods take a name as an input that must match the value you define in the respective Configure App tabs.

Manually Creating a Mobile App

TIP: If, like the vast majority of users, have created your mobile app via the Add MPulse (AMP) utility, you can skip this section.

As noted elsewhere, in *most cases* the App Admin user will create an app in the mPulse Repository using the Add MPulse utility. However, both types can also be created through the mPulse Central > Apps node. Use the following instructions to manually create a mobile app in mPulse that represents your mobile app. Then, when you inject the mPulse Library into your project, IPA, or APP bundle folder be sure to conform to the values that you set here.

1. To get started, login as the user with App Administration rights.
2. In Central, select Apps and then click New. The Configure New Mobile or Web App box appears.
3. Give the app a name. Usually, this will be the name of the project, IPA or APP bundle folder.



4. Provide the Mobile Bundle ID from the project.
5. Provide the Launch URL (must end in "://").

- The API Key is auto-generated by Add MPulse
- Accept the default Session Timeout

  The mPulse default session timeout is 30 minutes but can be tweaked to a maximum of 60 minutes. See Session Metrics and Session Timeouts for more information about this setting.

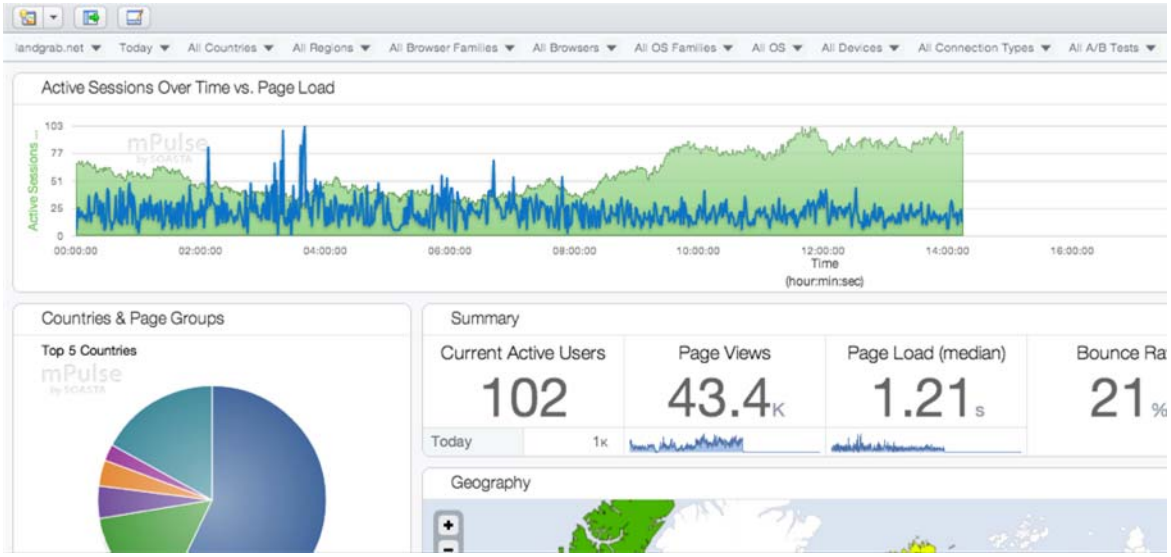
6. Click OK to exit the Configure Apps box after making any or all of the necessary changes.

Analytics

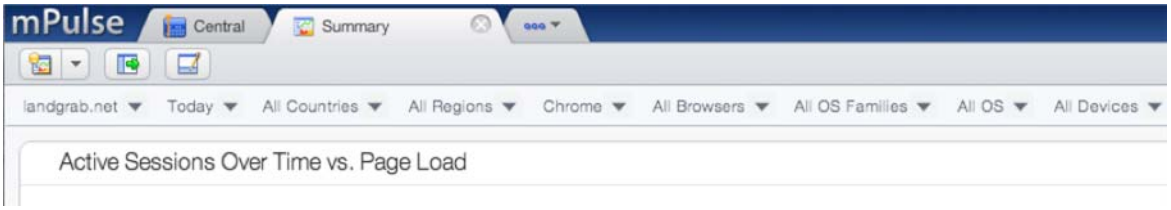
SOASTA 53 introduces many visual improvements, including a new, sleek Dashboard Filter Toolbar, filtering by Browser and OS Family, and crucial new filtering metrics that track Active Sessions as well as Current Active Users, as well as a new Flip Book Mode that allows as many as 8 mPulse tab to be displayed and toured easily. These and additional general dashboard improvements are detailed in the following sections.

New Dashboard Filter Toolbar

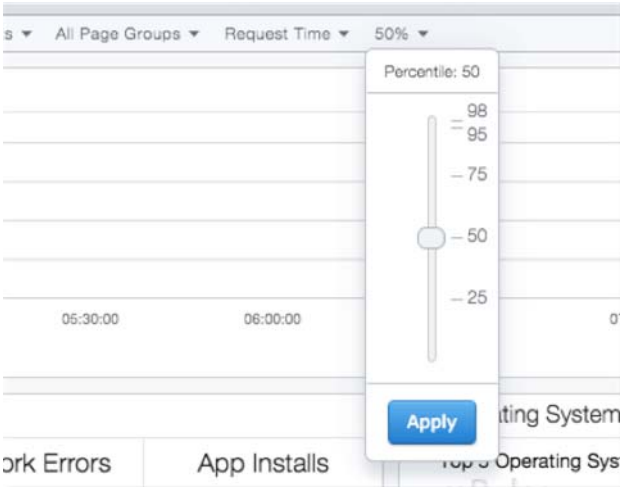
The new Dashboard Filter Toolbar is less obtrusive, & considerably more streamline than the one found in prior releases. The new Filter Toolbar occupies less of the vertical space between the top-level toolbar and dashboard widgets in whichever layout is in use.



Drop-down labels and the down-arrow icon also reflect a newer, more streamline design.



The Percentile drop-down also reflects the new, streamline look of the Dashboard Filter Toolbar.



New Browser Families and OS Families Toolbar Filters

This release introduces two new toolbar filters, Browser Families, and OS Families. These two new filters modify the existing Browser and OS filters respectively.

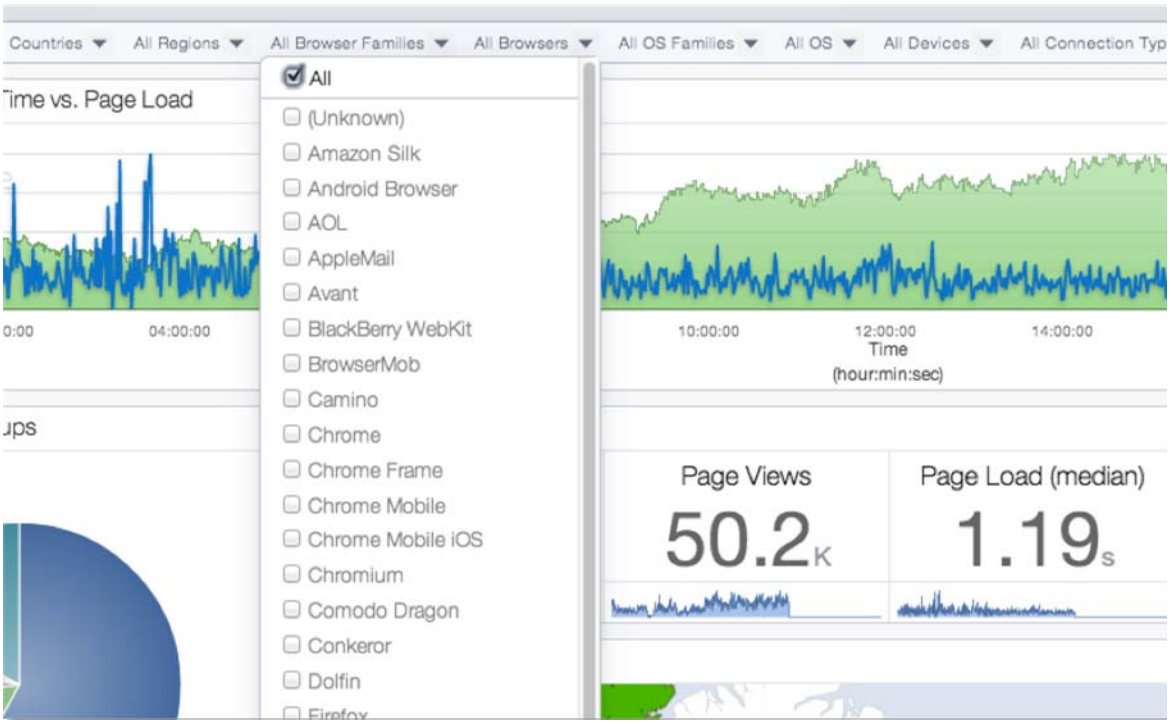
For this reason, Browser Family is placed to the left of Browser on the toolbar, while OS Families is placed to the left.

Users can now filter by Browser Family (left selection above) as well as by Browser, and by OS Family (right selection above) as well as by OS itself.

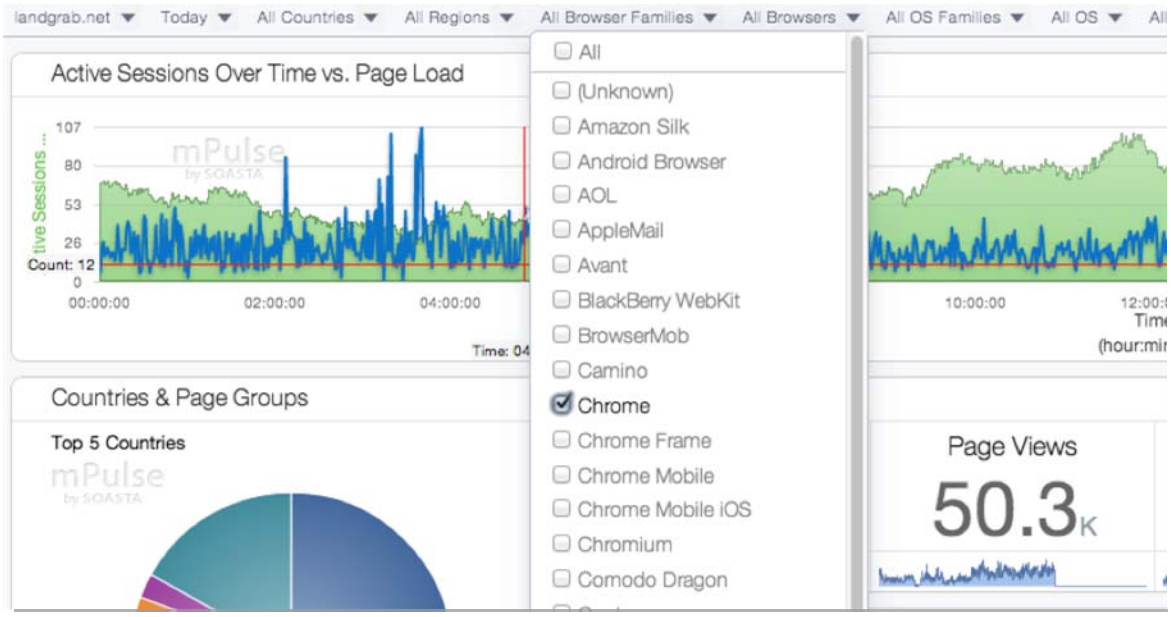
The addition of these two new filters accommodates an increasing number of variations of user agent and OS as mPulse Mobile beacons are added..

Filter by Browser Family

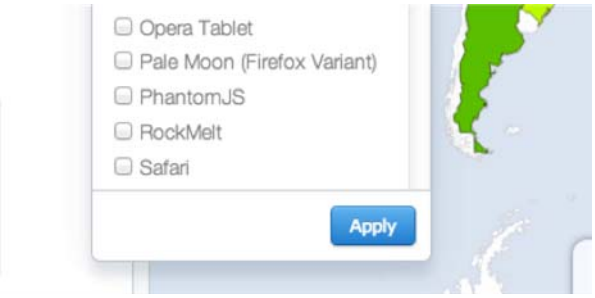
The new All Browser Families drop-down is shown below with the default *All* selected. All Browser Families are shown using this default. There are many more Browser Families than can be shown.



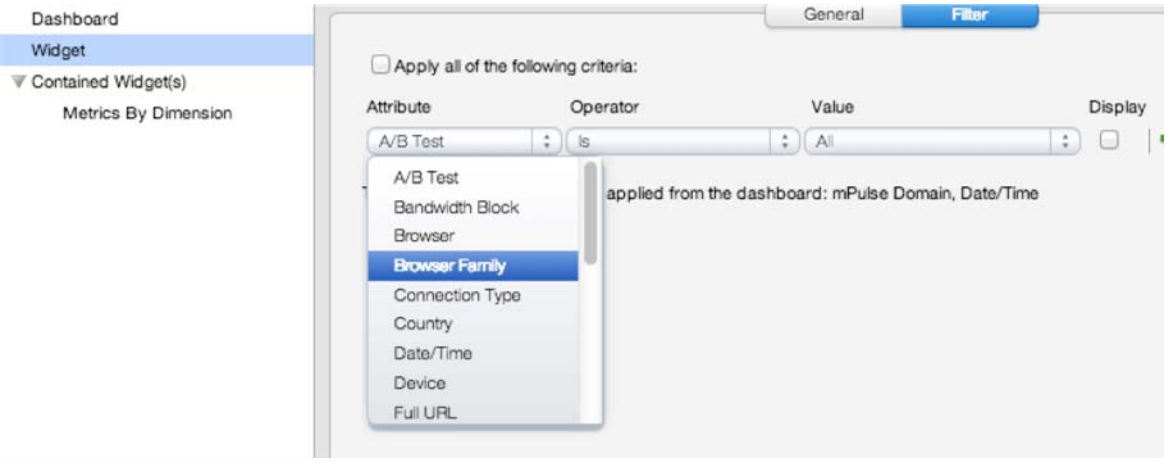
- To filter by a single Browser Family, check its value in the drop-down. Check more than one Browser Family, if desired, and then click Apply.



Once the filter is applied, only the matching Browsers are presented in the dashboard.

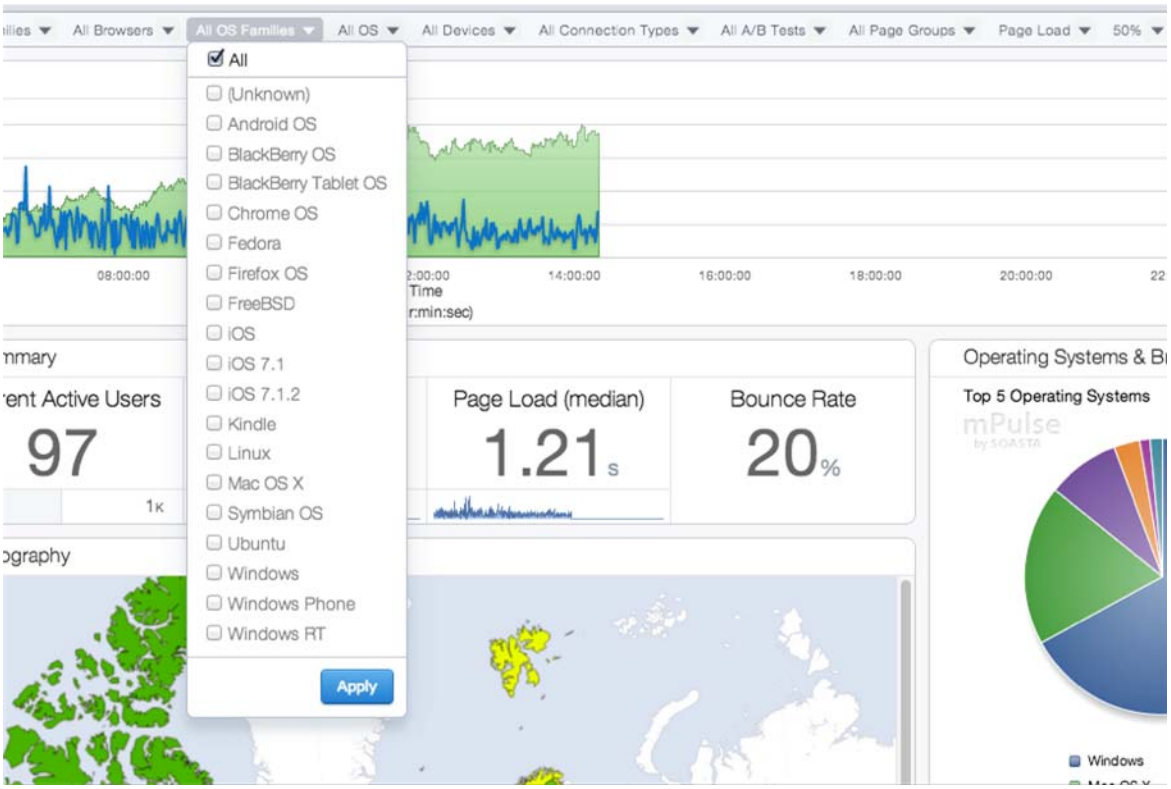


Browser Family is also available as a filtering attribute at the widget level (where applicable).

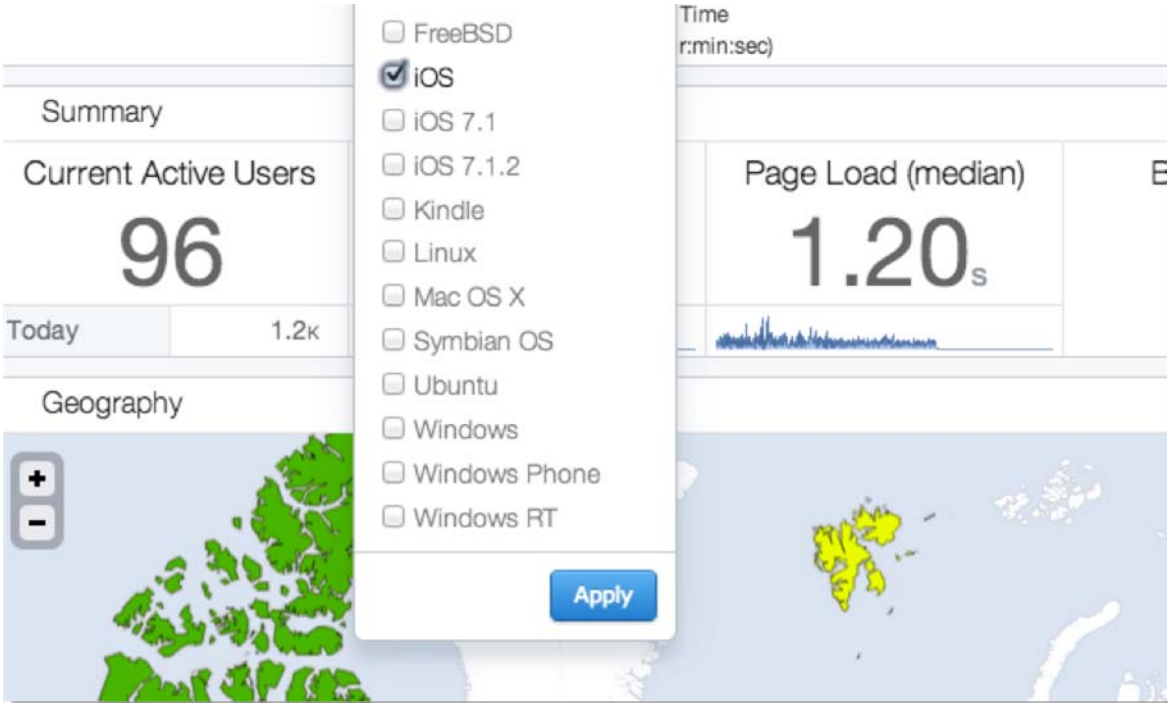


Filter by OS Family

The new All OS Families drop-down is shown below with the default *All* selected. All Operating Systems are shown using this default.

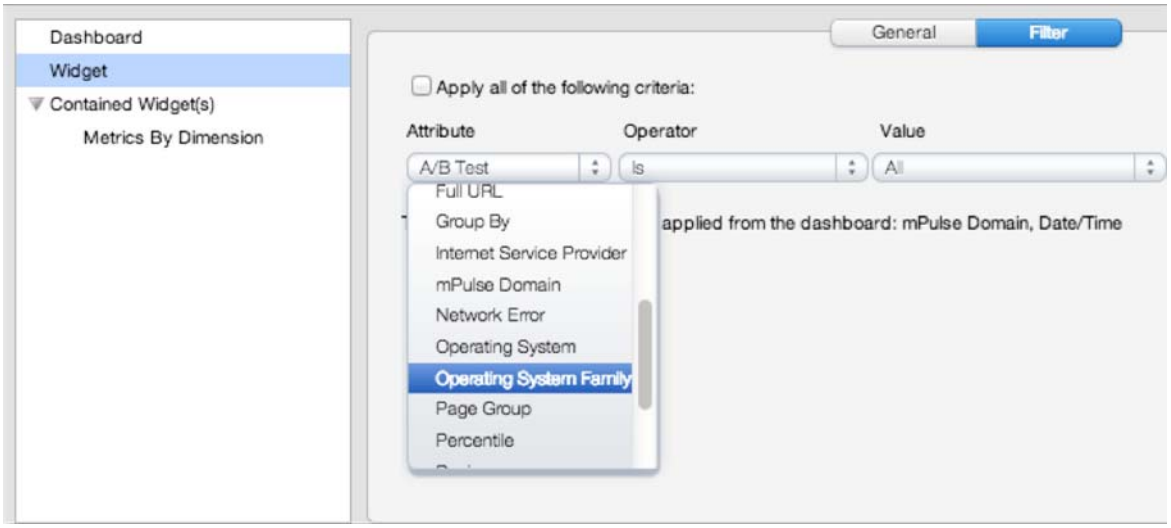


- To filter by a single Operating System(s) in the dashboard display, check one of the OS Family values. Check more than one OS, if desired, and then click Apply.



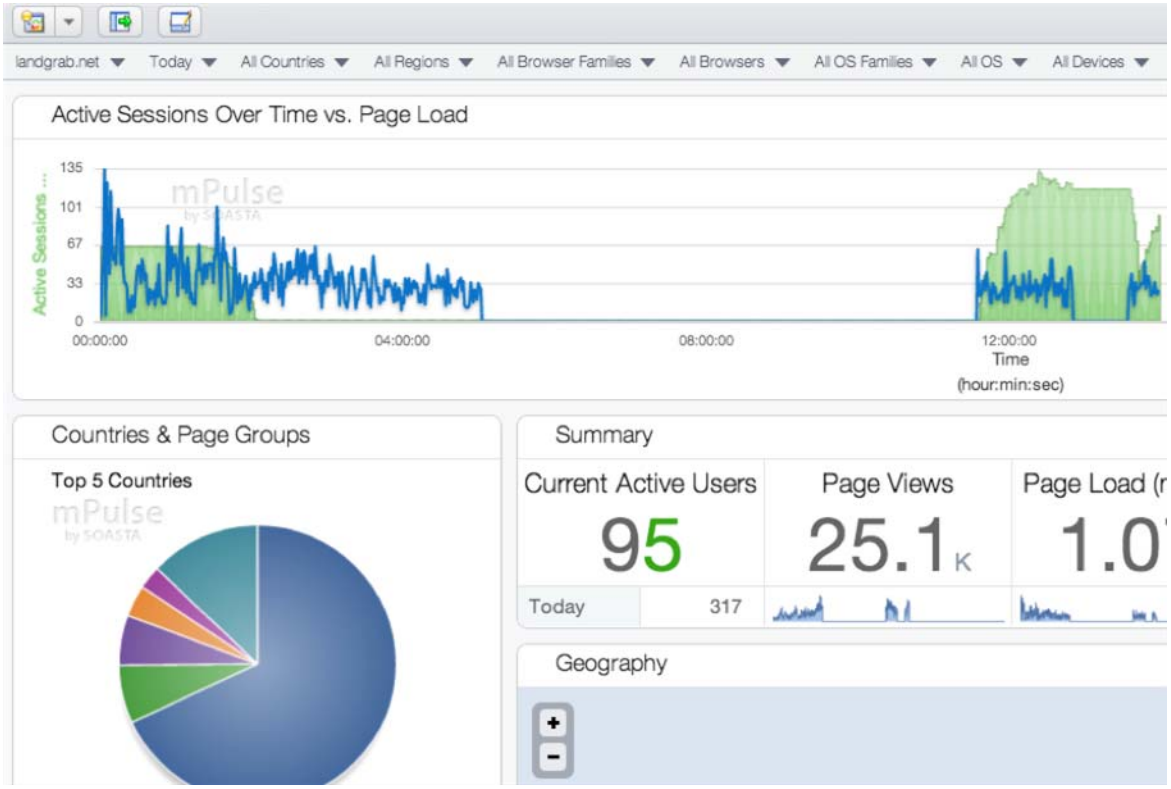
When the filter is applied, only the matching Operating Systems are presented in the dashboard.

Operating System Family also appears as a widget-level attribute to filter by. For example, Operating System Family is shown below as a filtering attribute in the Metrics By Dimensions widget (shown below).

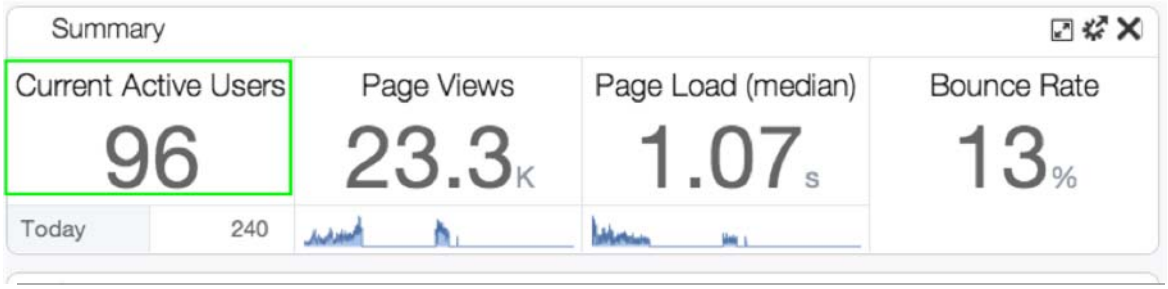


Active Sessions and Current Active Users

This mPulse release implements a new *active session* metric that complements the *completed session* metric already in use. This new metric is featured prominently in mPulse's Summary Dashboard, in the combined Active Sessions Over Time vs. Page Load widget, as well as in the Summary, Current Active Users section, which displays an up-to-the-minute user count. Current Active Users also features a running Today count.



When we calculate active sessions we count unique sessions (e.g. by *sessionID*) for that point in time. A completed session is recorded only once the session timeout is reached. This distinction, between *active* sessions and *completed* sessions (e.g. those that have ended) is an important one. The active sessions are the *current* sessions.



Additionally, a running count for a moving time windows is shown at the bottom of the Current Active Users section (the default is Today).

Filtering Current Active Users / Last-X Active Users sub-row

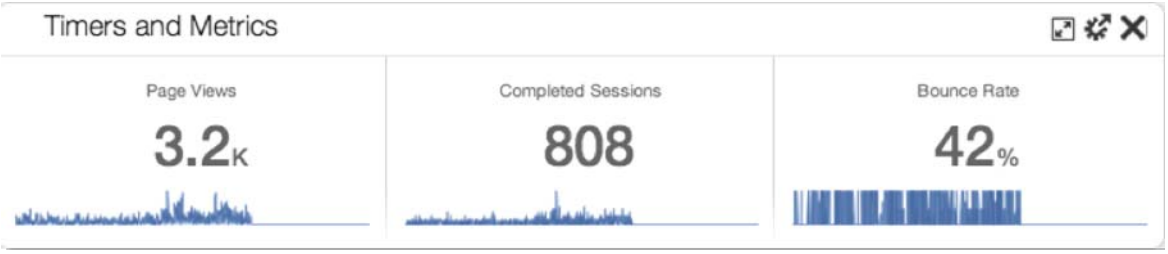
The Current Active Users metric never varies, and always applies to the active sessions *now*. The sub-row value (shown here as "Today) displays a running count of active users (e.g. users who were active during the selected time window).

Refer to Time Windows for a brief overview of applying time windows to a dashboard or widget.

Timers and Metrics widget merged into Summary widget

As noted above, the Summary Dashboard (formerly: mPulse Summary Dashboard) was renamed for this release. This name change includes, of course, the Summary widget from which the dashboard takes its name.

Additionally, users should note that Timers and Metrics (shown below in its SOASTA 52 version) is now embedded into the Summary widget.

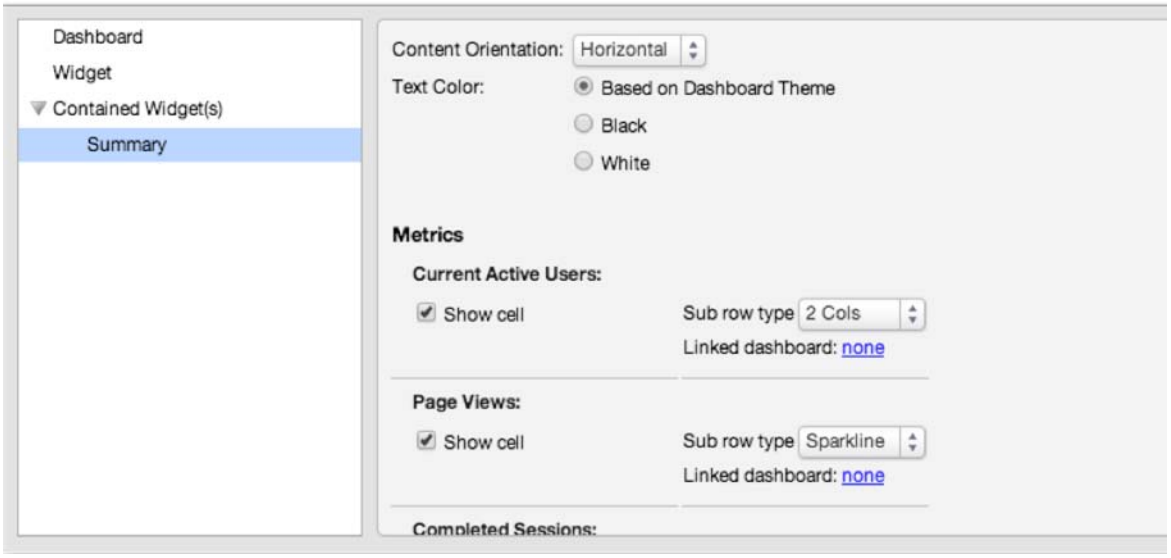


As a result of this change, Timers and Metrics no longer appears in the Widget Selection Panel's Widget Type > RUM category as a separate widget.

By default, the Summary widget now includes the 3 metrics above (as well as the ones already detailed in the Current Active Users section above, which is new in SOASTA 53). The following section details how to change the widget in a custom dashboard to display only a sub-set of these defaults.

Changing the Summary Widget Display

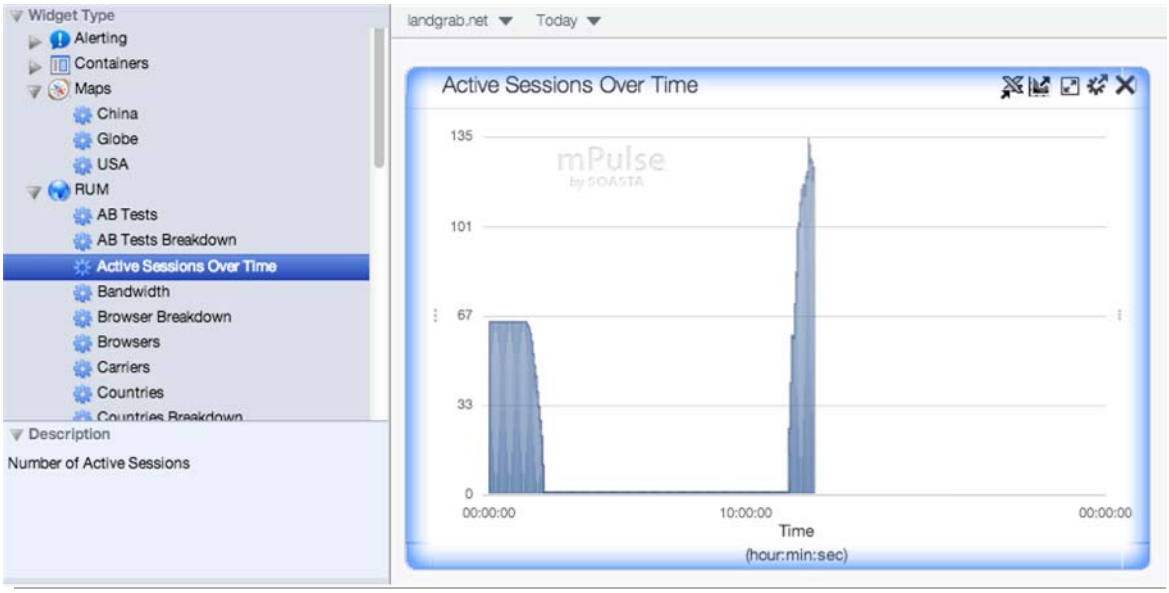
Summary widget metrics, including Current Active Users, are modified in the Edit Panel. The 2-column sub-row can be turned off, or changed to Sparkline. . Turn on/off a metric, including Current Active Users, by unchecking it in the Summary widget's edit panel.



Active Sessions Over Time Widget

The Active Sessions Over Time widget is featured prominently, as a combined widget (e.g. Active Sessions Over Time vs. Page Load) on mPulse's default Summary Dashboard.

For custom dashboards, this widget is available for addition via the RUM list (e.g. under Widget Type Selection Panel, Widget Type category).

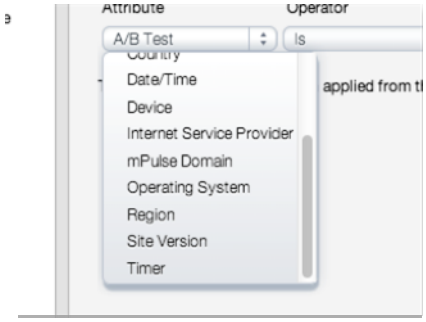


Filtering the Active Sessions Over Time Widget

The Active Sessions Over Time widget inherits dashboard-level filters. In custom dashboards, widget-level filtering can also be applied.

Any or all of the following filters can be applied to the Active Sessions Over Time widget:

- A/B Test
- Bandwidth Block
- Connection Type
- Country
- Date/Time
- Device
- Internet Service Provider



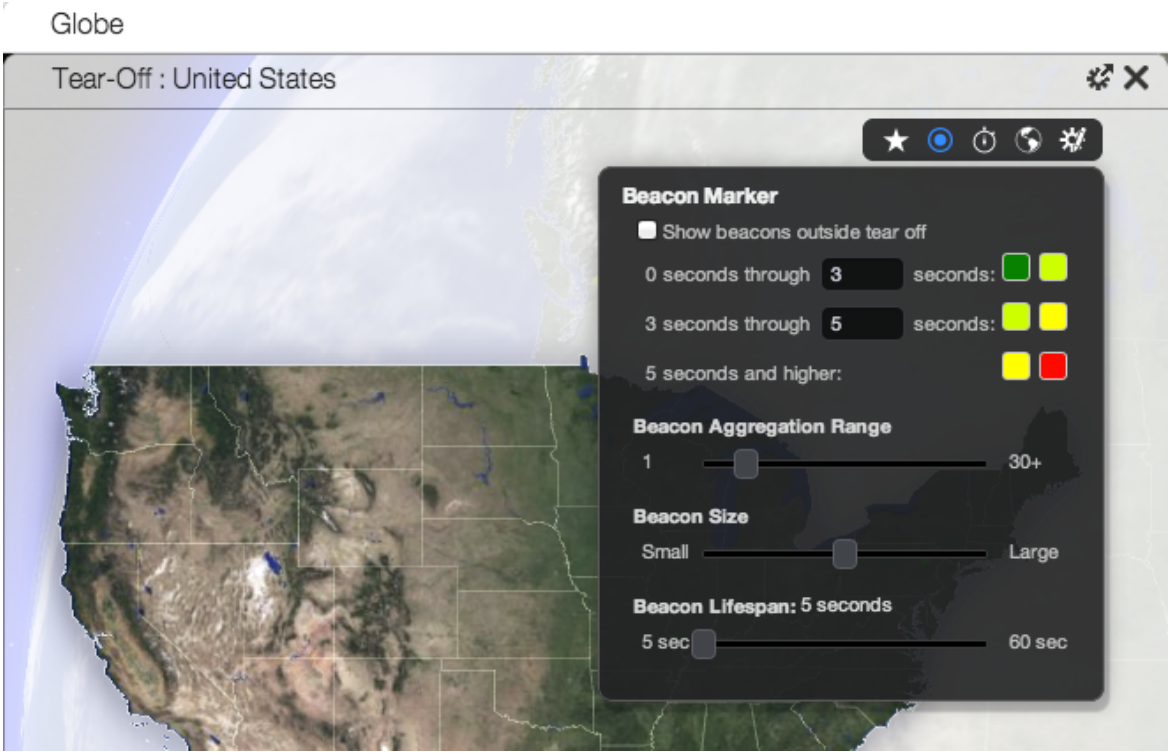
- Domain (e.g. App)
- Operating System
- Region
- Site Version
- Timer

Refer to Filtering mPulse Dashboards for a general overview of applying filters to mPulse dashboards and widgets.

Show Beacons Outside Tear Off Checkbox

Globe, Beacon settings now include an additional checkbox (e.g. Show beacons outside tear off).

Since, for tear offs widgets, the default beacons display only within the tearoff's geographical shape, this new setting is provided to override that and show all beacons in the tear off. When this checkbox is ticked, beacons will render at all geographic for which locations data is given.



Widget-on-Widget Layout and Edge Constraints

This release resolves some issues with Widget-on-Widget layout that would result when the display was switched between monitors of varying resolutions. Additionally, This release adds the ability to edge constrain Widget-on-Widget (WoW) layouts in dashboards.

Edge constraint provides the ability to fix the widget's position relative to the left or right edge, or to the corner(s) of a dashboard.

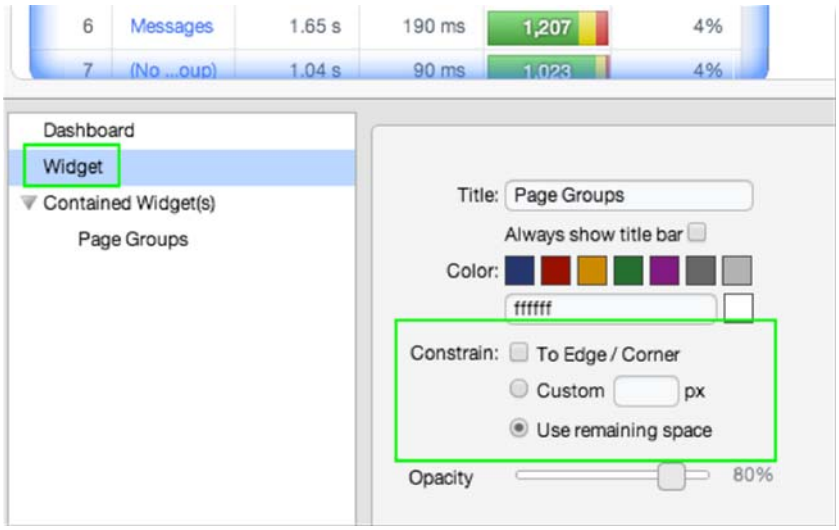
1. To add edge constraints to a WoW in the dashboard, first select it.

   If you don't already have a WoW in the current dashboard, toggle the Widget Selection Panel on and add it now.

TIP: Refer to Widget on Widget Layout for more about this existing dashboard feature.

2. Click the Properties (Toggle Dashboard Edit Panel) button. The Dashboard Edit panel opens with the Widget node selected.

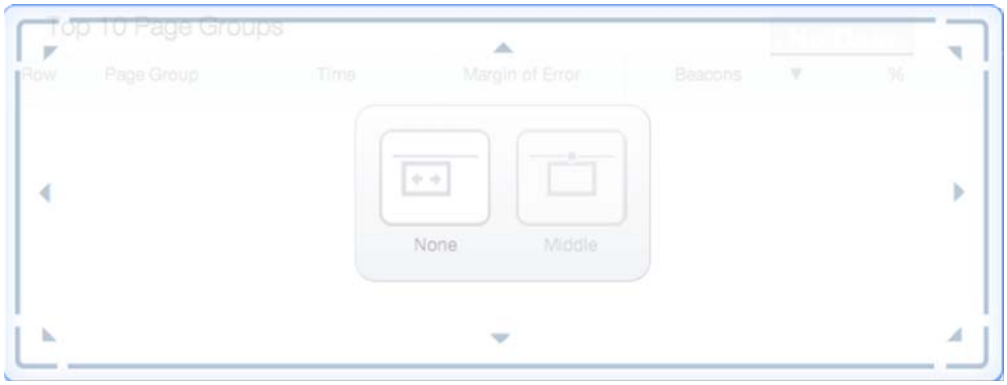
The new Constrain settings section is shown below.



The other two settings above existed in the prior release:

- *Custom Pixel Height* – Specify a custom pixel height for the selected widget (overrides To Edge/Corner)
- *Use Remaining Space* – If no pixel height is specified, a WoW will expand and shrink to fit, it can also be drag resized or resized on Dashboard load (overrides To Edge/Corner)

3. Check the To Edge/Corner box.
4. Click Apply (on the right of the panel). When you do so, the selected widget displays its Constrain Position Along Edge settings.



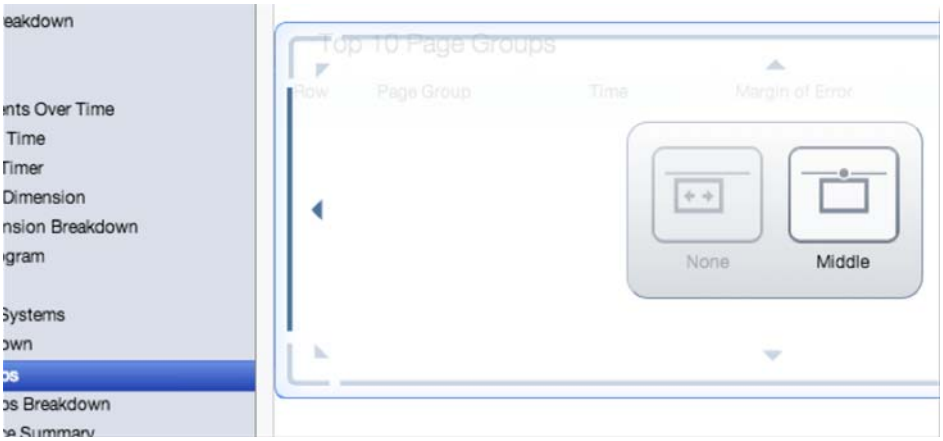
5. Choose a constraint mode.

- Click *Middle* to constrain widget to use a 50% vertical position (e.g. the widget will align to the vertical middle of the dashboard (e.g. from top to bottom).
- Or, click *None*. In which case the widget is aligned to the position it was in prior to applying the edge constraint.

6. If None, was clicked, you can use the directional arrows in the Constrain Position box to snap-to any of the four directions: left, right, top, or bottom or to any of the corner positions.

   When a given direction is selected a blue border appears along the selected widget's edge. For example, top is selected below.

Once you've selected one of the four directional positions (e.g. left, right, top, or bottom), you can additionally then click Middle again to adjust to the middle of the given direction.



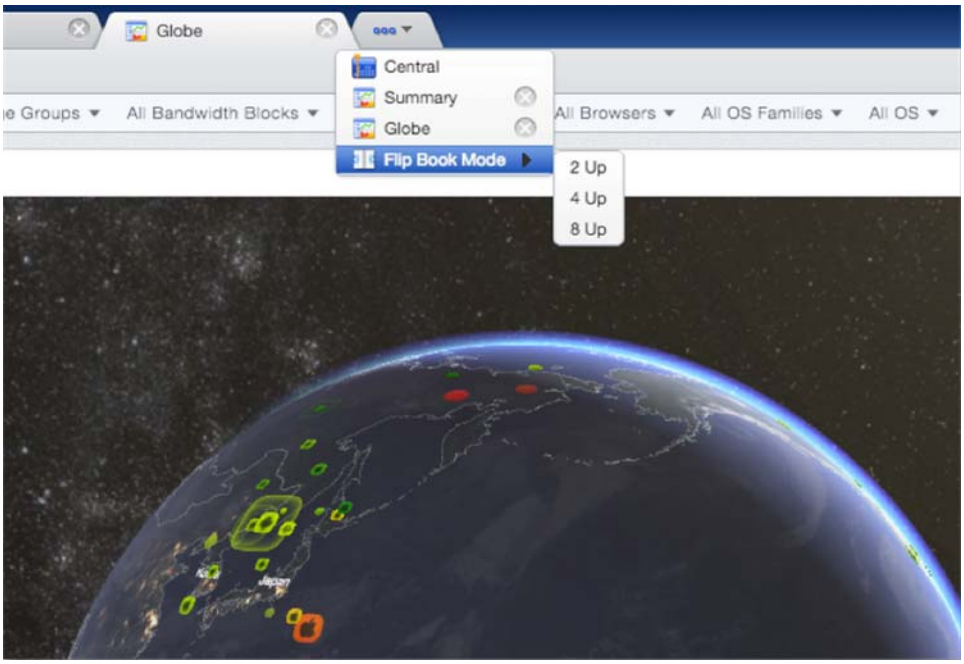
Note that if the corner position is applied that the widget is then in an absolute position (e.g. it is not relative since the widget cannot move from there) so the subsequent Middle click doesn't apply.

7. Additionally, you can specify a Custom pixel position.
8. Or, click Use Remaining Space.

Flip Book Mode

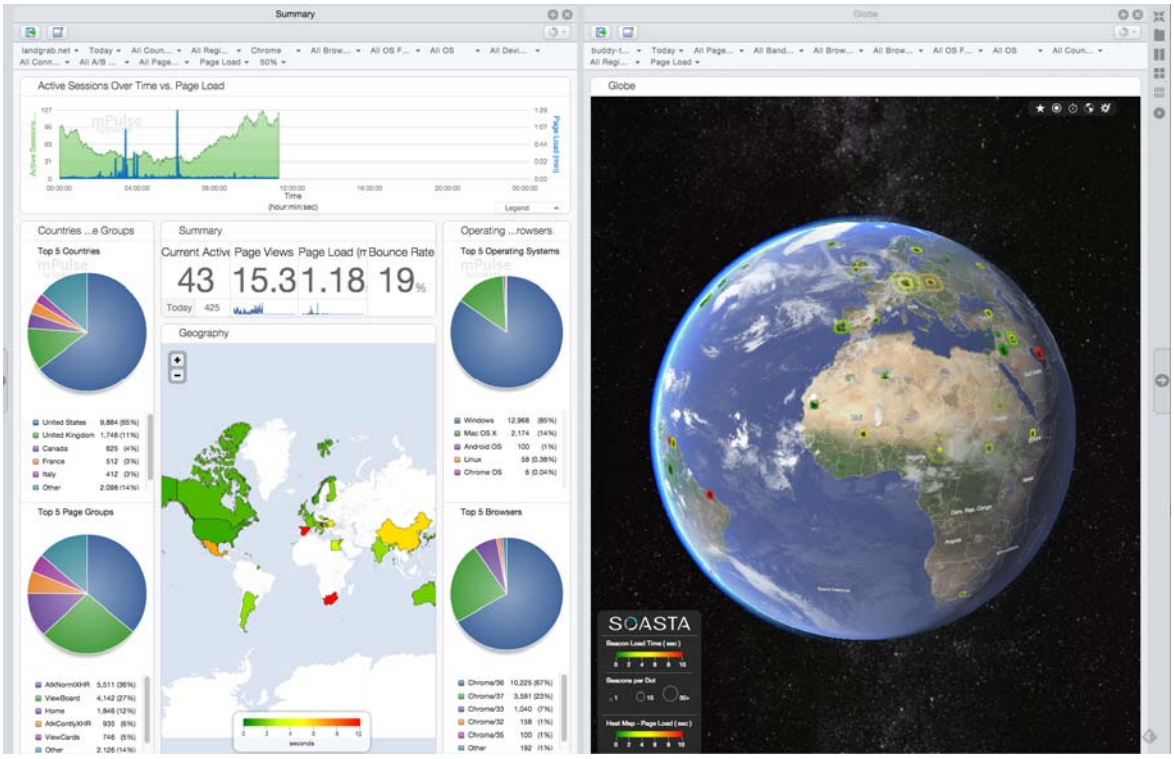
This release introduces Flip Book Mode as a new tab management feature. This mode allows the user to easily view multiple tabs at a time to take advantage of side-by-side comparisons on larger displays. Flip Book Mode is available whenever the Etc. (…) tab appears in Central (below, right of image).

To invoke Flip Book Mode, click the Etc. drop-down menu on the main toolbar and then select the Flip Book Mode command (shown below).



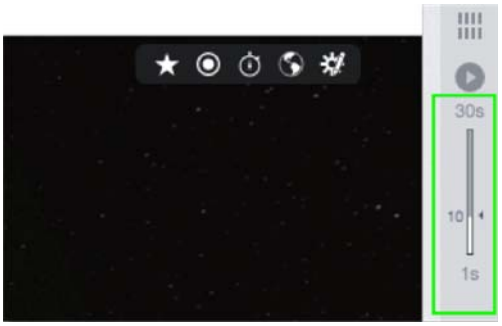
When you do so, the sub-menu options appear. As shown above, Flip Book Mode allows the user to switch the display from a single tab at a time to a layout of 2 Up (Alt-2), 4 Up (Alt-4), or 8 Up (Alt-8) per page.

For example, if we select 2 Up, the following page layout is invoked. The pages in Tile Mode work as a carousel so there is no start or end page.



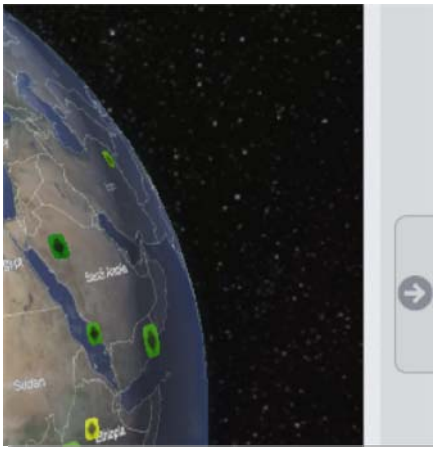
While in Flip Book Mode, use the top right controls, from top to bottom, to enter Full Screen mode (Alt-F), to Exit Flip Book Mode (Alt-T), or to switch between 2, 4 (Alt-4), or 8 Up (Alt-8).

Click the Auto-Page button, which is found just below the five buttons described above. When you hover over it, a slider also appears to set the Auto-Paging interval.

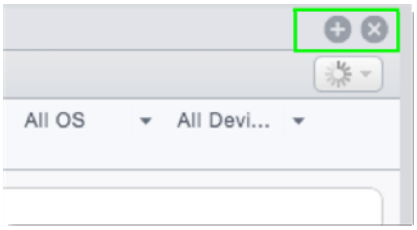


The slider can be set before or while Auto-Paging mode is set to on, and to any value from 1 (as if one were flipping through a book) to 30 seconds.

To browse pages, click the Left (Alt-Left) or Right (Alt-Right) arrow (in the vertical middle of either side of the browser page).



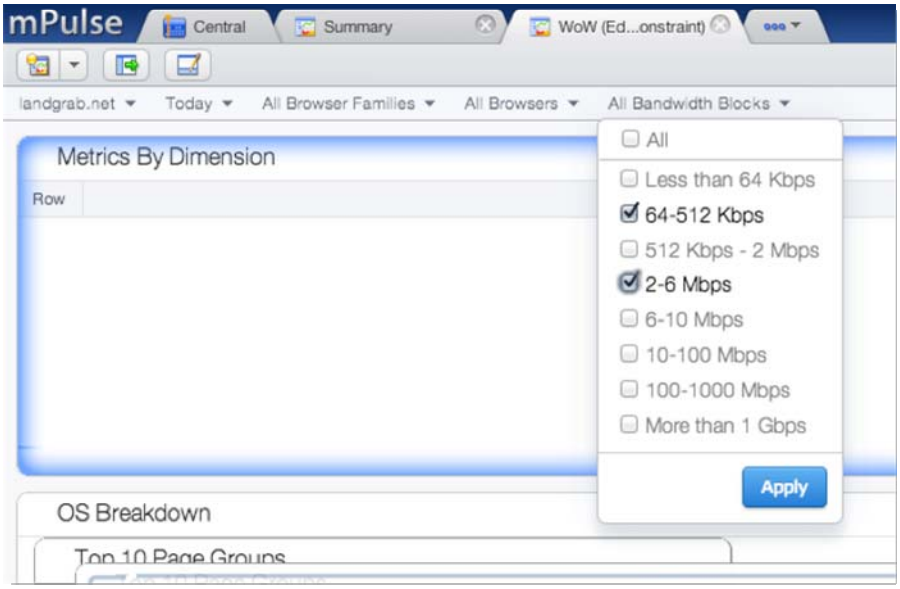
Use the Close (X) icon on a given tab to close that tab just as you would in normal mode.



Use the Plus (+) icon to maximize the tab to focus more closely on a particular task in a particular tile.

Multi-Select in Dropdown Selections (64833)

As of this release, checkboxes appear in Filter dropdowns in mPulse.

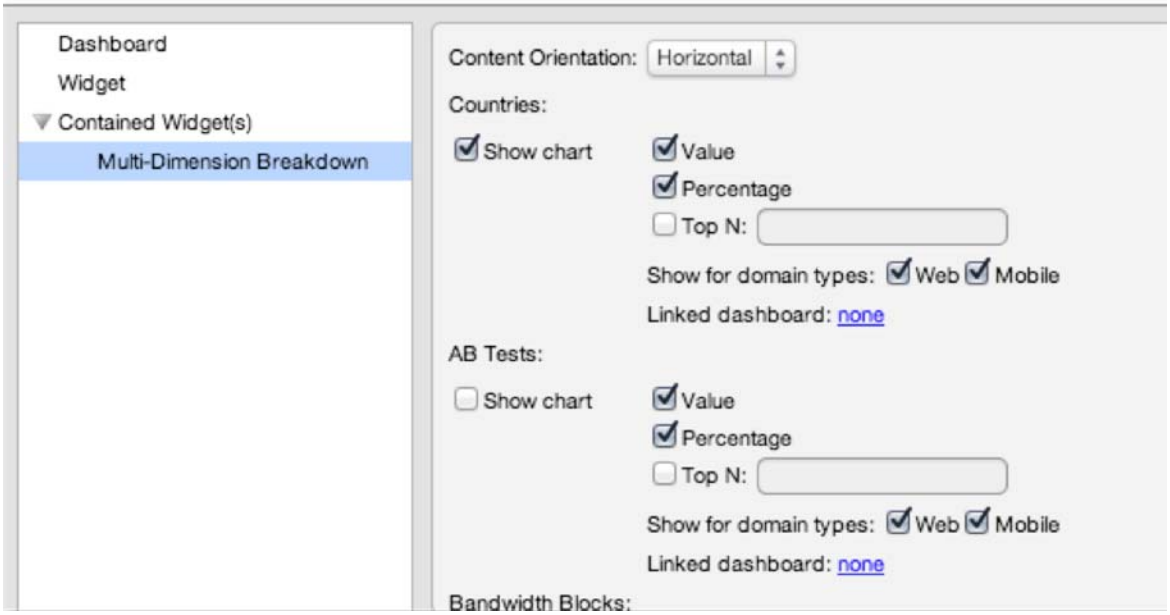


Multi-selection allows non-sequential filtering for the first time in mPulse and is provided for both new and existing metrics.

As noted above, this is particularly useful in the context of Browser Families and OS Families filtering, both of which are new to mPulse in SOASTA 53.

Toggling Mobile and Web Dimensions in the Multi-Dimension Breakdown

Dimensions can be turned on or off for Mobile or Web per dimension in the Multi-Dimension Breakdown widget's Edit Panel.



- Check or uncheck the Show for domain types for Web or Mobile to turn off the dimension for that domain type.

General Dashboard Improvements

Renamed dashboard objects in this release:

- The System Dashboard, mPulse Summary dashboard is now simply the Summary dashboard. The Timers and Metrics widget has been merged into the new Summary widget as an adjustment for mPulse Mobile. The new Summary widget is comprised of the combined metrics with only a few changes (most notably, the new Current Active Users section, and Median Request Time, which is now just Request Time)
- The System Dashboard, Operational Dashboard, has been renamed to OPS.

Dashboard improvements in this release include:

- Multi-series charts (e.g. Top N charts) are now *Top 10* by default
- Globe Widget defaults have been updated for this release to the following default(s):
    - Country, State and City Labels OFF
    - Country Borders ON
    - USA State Borders ON
    - Night ON
    - Clouds ON
    - Rotation ON
    - Sun ON
- Additionally, the Globe's visual display of man-made light (e.g. where it's night time) has been improved.

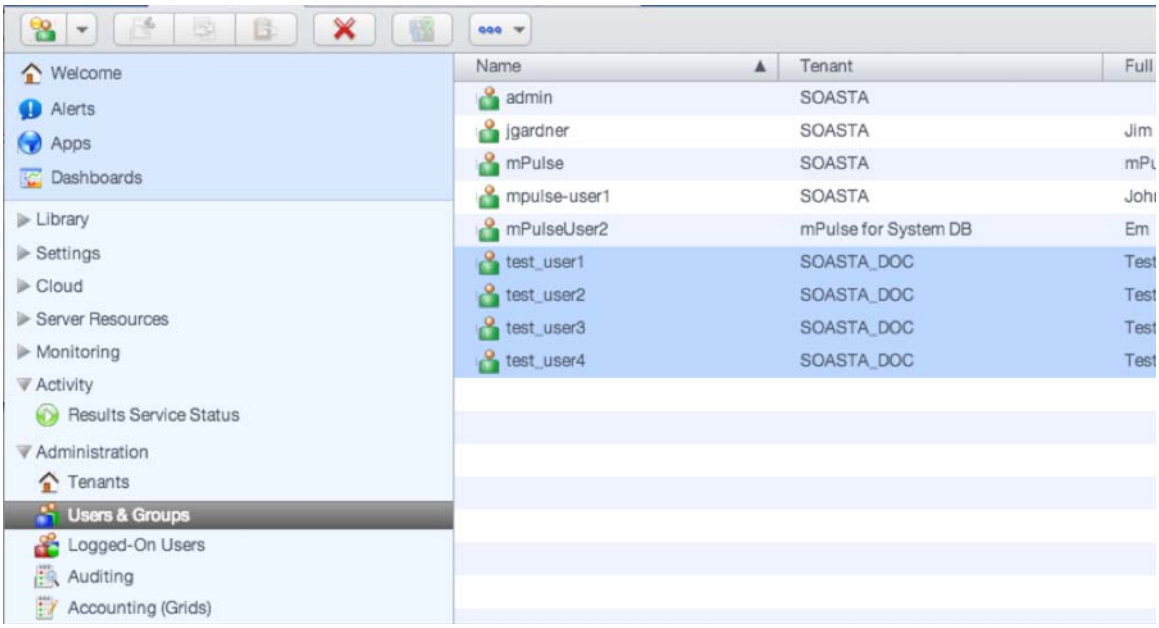
Enhancements

User and Permission Management Enhancements in SOASTA 53

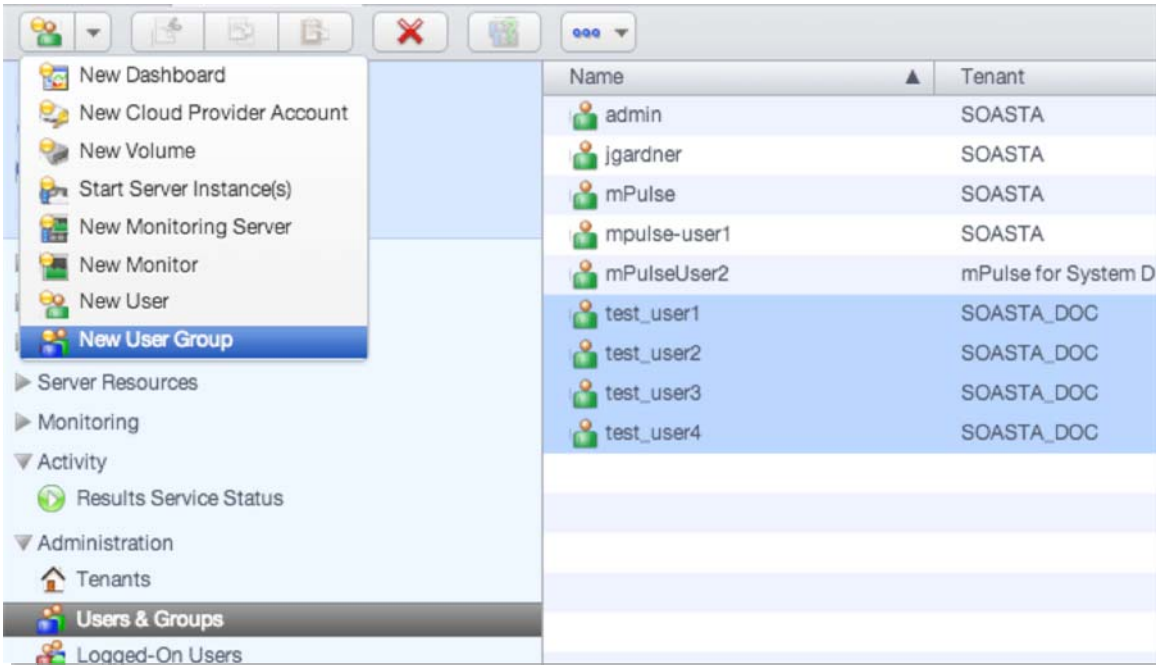
User Groups

In previous releases, permissions (access control lists) were managed by adding lists of individual users to an ACL. As of SOASTA 53, administrators can create User Groups as well. User Groups are simply collections of users that can be used when building access control lists.

To this end, the Central > Users node has been renamed to Users & Groups. Users are created as they have been in SOASTA since the beginning.

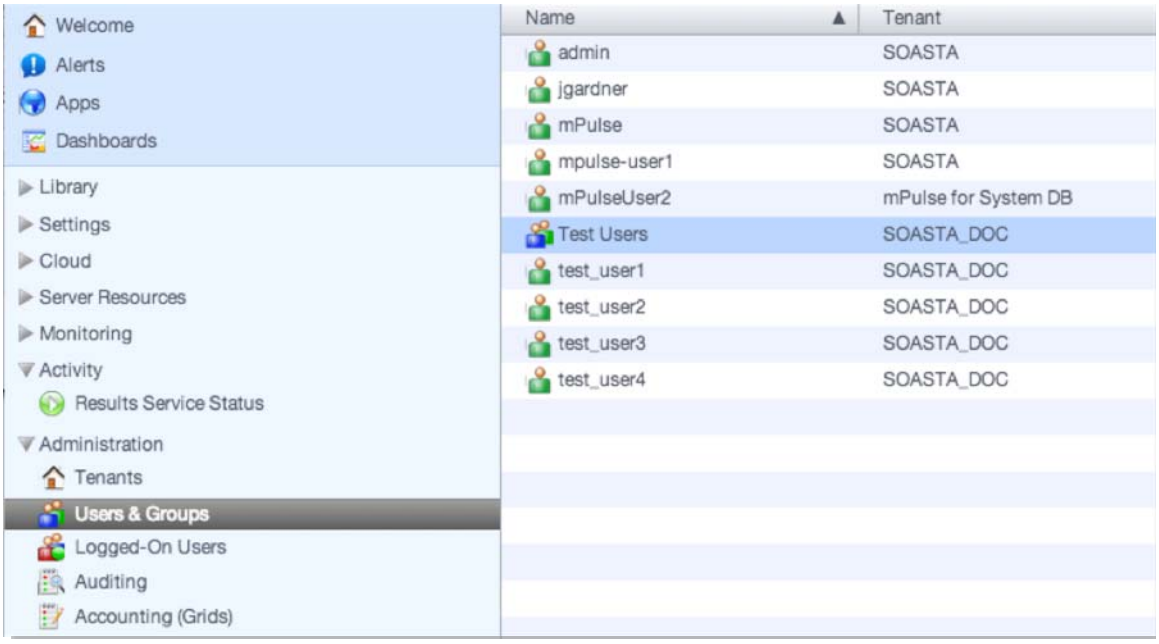


1. To create a User Group, select Users & Groups and then choose New User Group from the New drop-down.



2. In the New User Group, give the new User Group a name and then check all the users to include.
3. Click OK to complete the User Group creation.

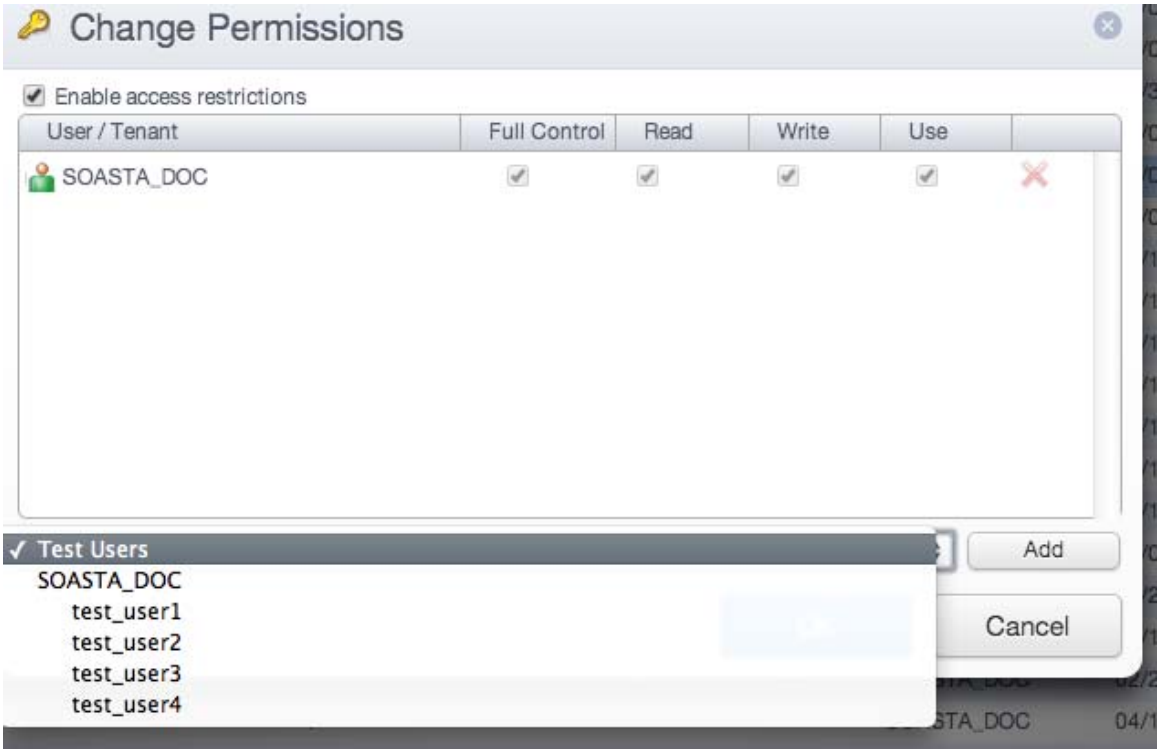
The New User Group appears in the Users & Groups list.



To edit an existing group, double-click it or right-click to open it in the User Group Editor. This box is precisely the same as the New User Group box. Un-check the user(s) to remove from the group.

Users can be included in multiple groups if desired. For example, the user joe@acme.com might be included in a New Hires group, but also in a Performance Engineers group. This allows a high level of precision when building access control lists.

Note that User Groups also appear in the Change Permissions box at the same logical level as User Admins. For example, in the drop-down (shown below) Test Users is a user group, while SOASTA_DOC is a User Admin.



Change or Transfer Ownership of Resource(s)

Ownership of a SOASTA repository object can now be transferred from one to another user by any user with User Admin privileges.

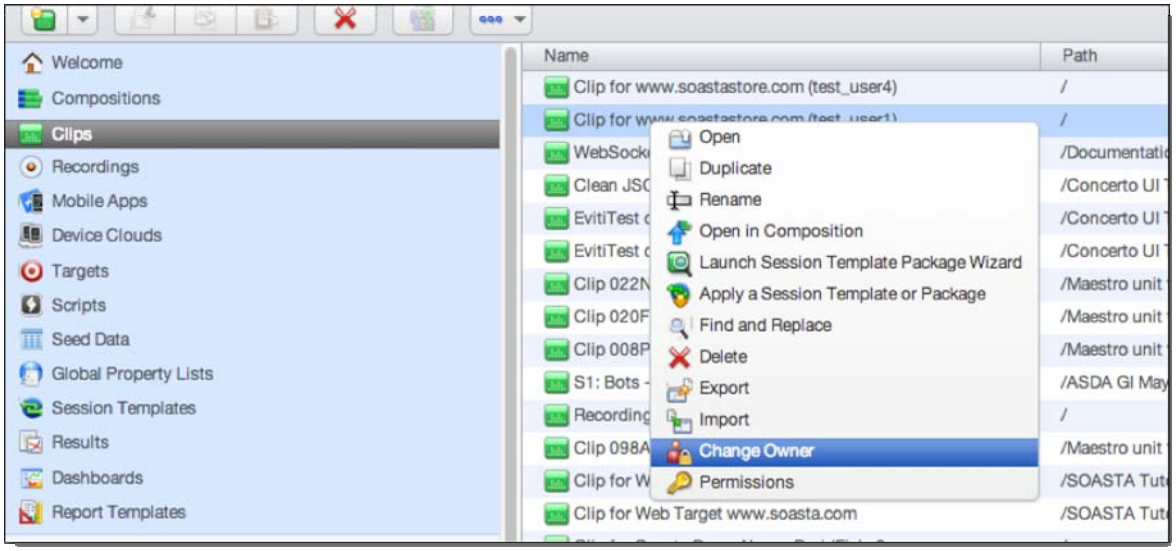
Change Owner allows a User Admin to clear a user's ownership of repository objects by moving ownership to a second user. This is, for example, necessary if that user has left the organization or team.

TIP: The User Admin can also take Full Control of any user object by selecting and choosing the Permissions command. Refer to the Full Control command also included in this release.

In prior releases, lacking this feature, the departed user couldn't be deleted until ownership of objects was removed, thus blocking the re-use of organizational equity.

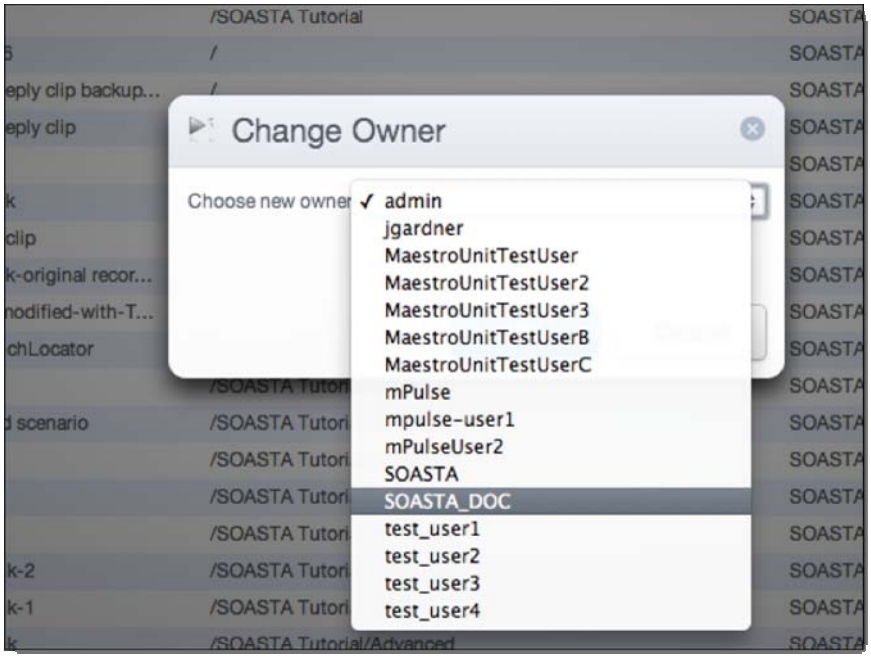
Now, the User Admin can simply transfer ownership of a directory or item in order to have the ability to delete a user, but still retain all of the items they have created for different projects.

To change the owner of one of more items, first select, and then right-click to select the Change Owner command from the menu.



When you do so, the Change Owner box appears.

Use this box to select an alternate user to whom ownership will be transferred.



Note: The Change Owner command will only appear for the User Admin of the (current) Owner.

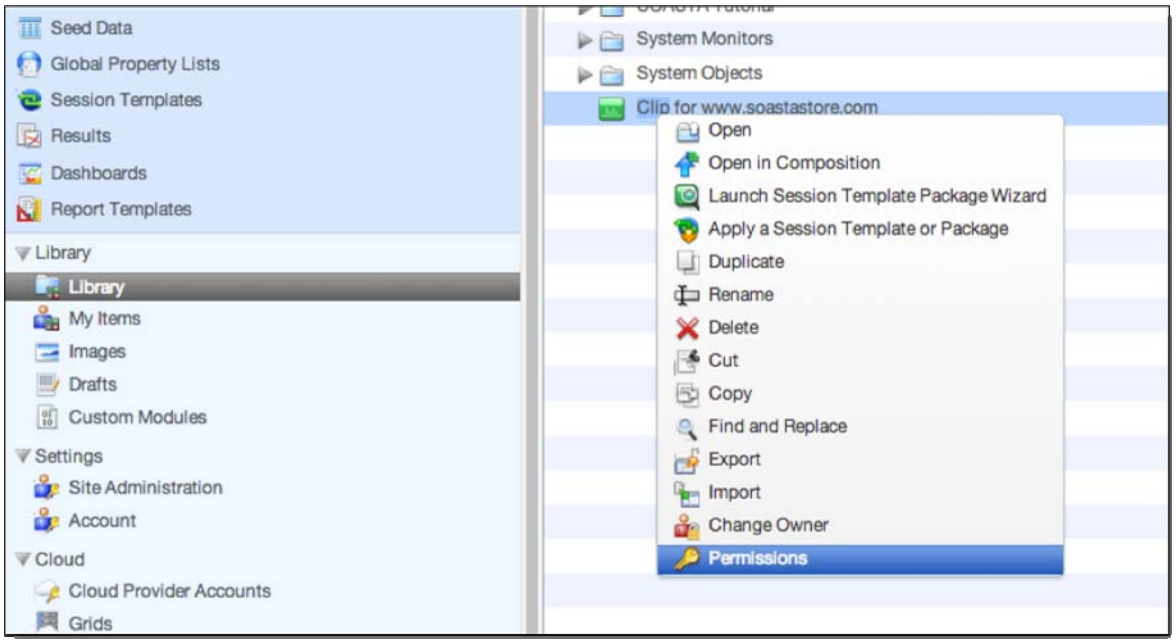
Full Control of Object(s) (User Admin)

This release introduces an enhanced ability for a User Administrator to take full control—using the existing Permissions box—of the underlying permissions formerly restricted to the original object owner. This ability is crucial to team collaboration in test compositions and with respect to other SOASTA repository objects, and is also useful if a team member is absent or leaves the team.

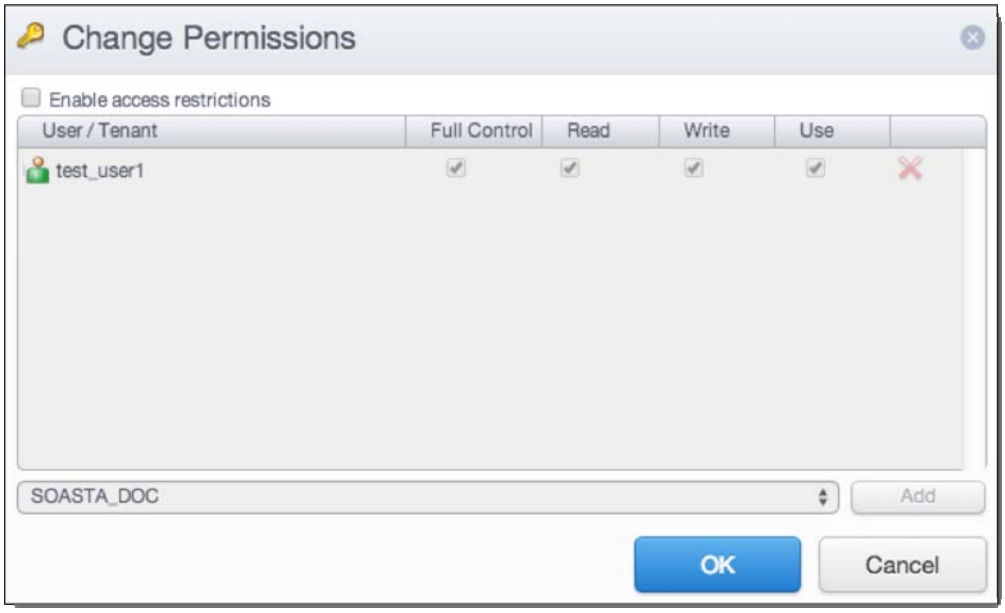
The User Admin can grant such full control per user()s, to herself, or to the Tenant. Any permissions granted to the Tenant applies to all the users in that tenant for the selection. However, the User Admin can supersede any tenant-level restrictions with additional user-level restrictions (or by applying Permissions one at a time per user—in either case, lower granularity means more specific control.

This new feature covers some edge cases. For example, if User A is not the owner of an object, but they have the ability to set permissions, and they apply permissions to User B, then the originating User A (a User Admin in this case) also gets those permissions. CloudTest always applies full-control permissions to User A when they change permissions.

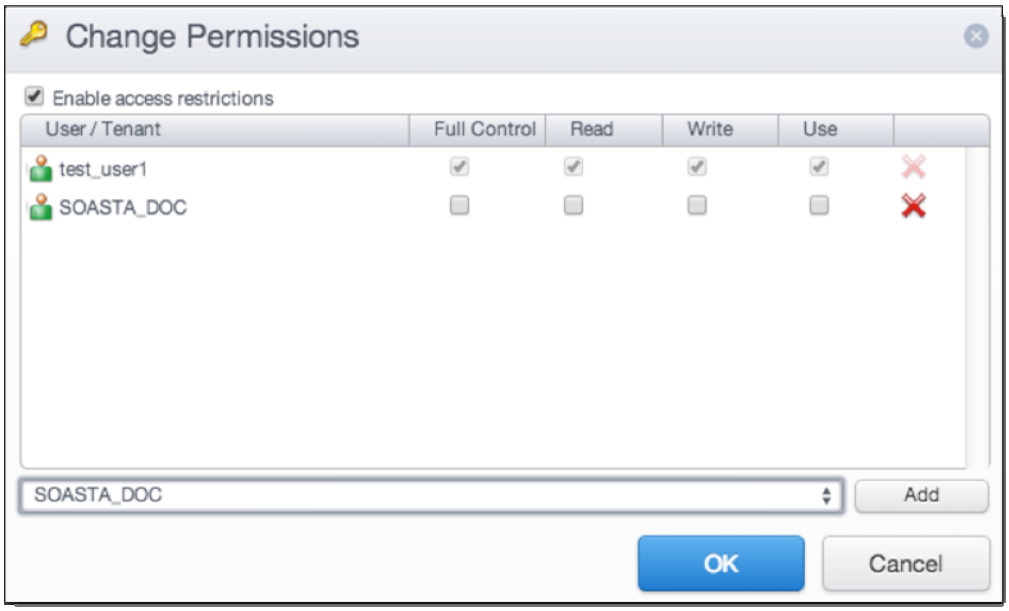
To take control of an object, the User Admin will select the object(s) and then access the Permissions from the right-click menu as in prior releases.



When you do so, the Change Permissions box appears displaying the current Permissions of the selection.

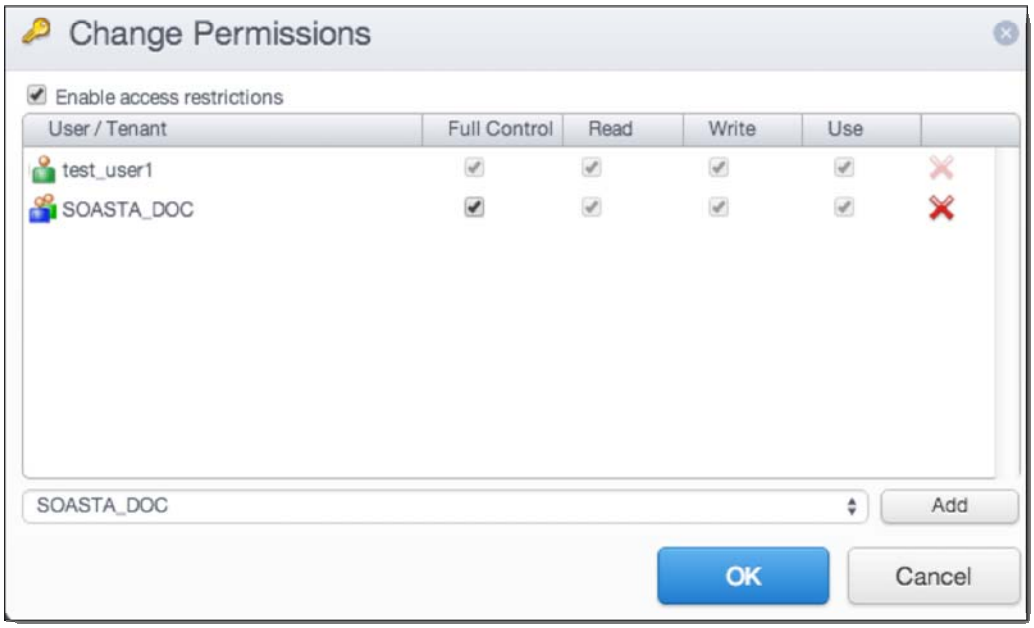


Select the tenant or user to add for whom Full Control (or other permissions) will be added or subtracted and then click Add.

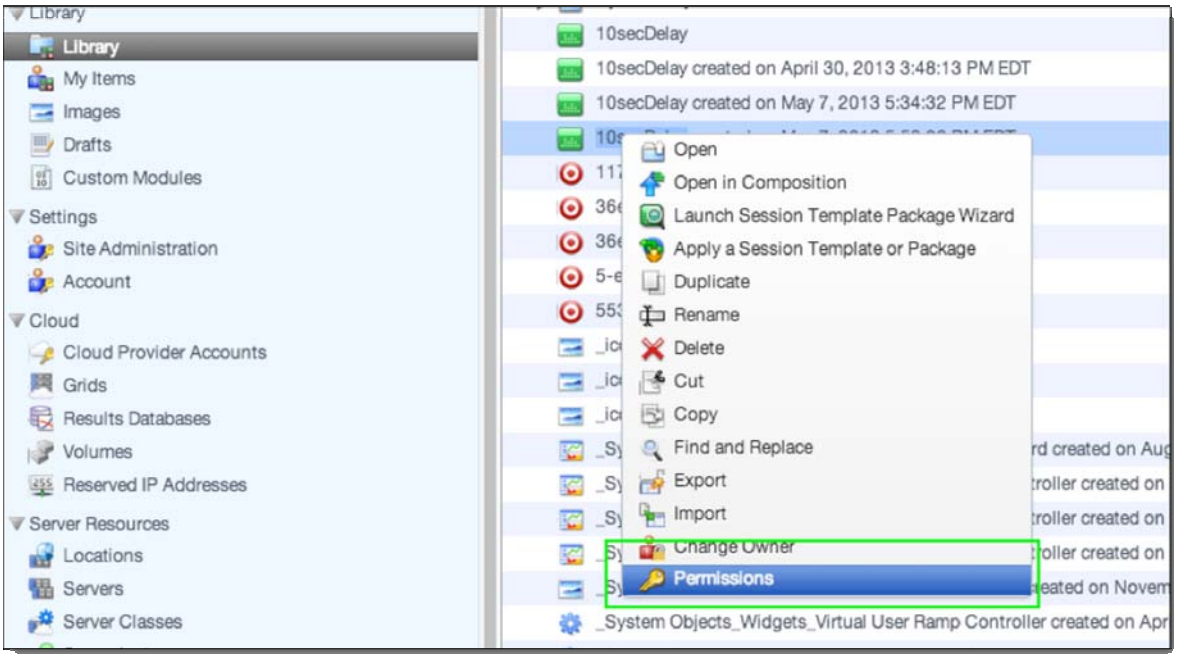


TIP: Full Control cannot be revoked for the given user. Additionally, Full Control should be granted to another user before removing a user from any given tenant.

After adding the selected user to the Access Control List in the workspace, check the Full Control box. When Full Control is checked, the Read, Write, and Use fields are grayed since Full Control includes all of these permissions.

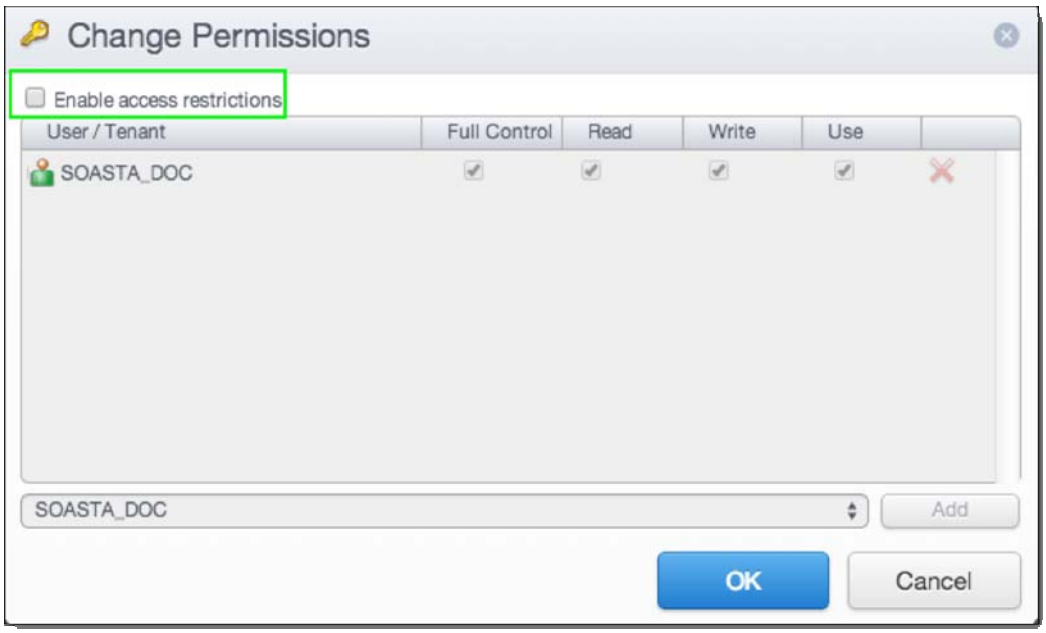


Click OK to complete granting Full Control to the listed user.



Once this command is selected, the Permissions dialog box appears. Access restrictions are disabled by default.

1. Check the Enable access restrictions box to start adding restrictions.



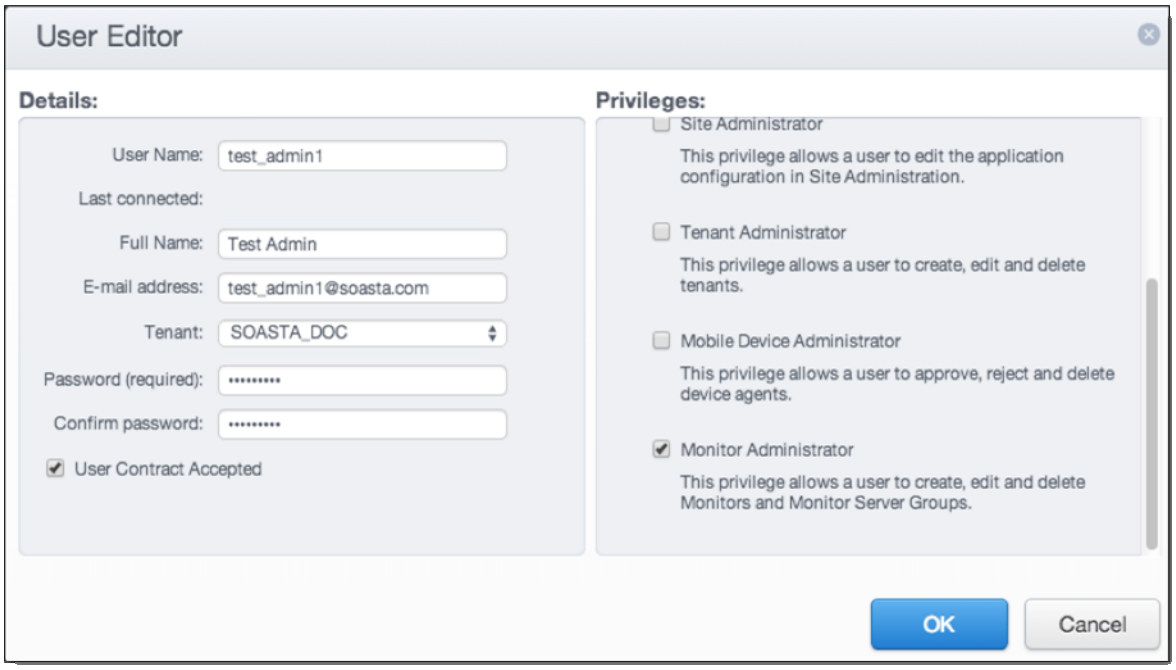
New Monitor Administrator Privilege

This enhancement permits the mPulse User Administrator to disable Monitor and Monitor Server Group creation per user in mPulse production environments.

Since Monitors can place additional stress on the read/write capabilities of the server, and significantly impact performance, Monitor creation is now reserved for those users with this privilege.

As of this release, the New User dialog box contains a new checkbox that the User Administrator can use to assign a new privilege, the Monitor Administrator.

To grant or disable the Monitor Administrator privilege per user, ensure that the box is either checked (enabled) or unchecked (disabled) for the given user.



Accounting records can prevent deleting a user (60818)

Grid and User account records are now no longer dependent on the User table., which in some cases in the prior release could prevent user deletion.

In the current release, a character string will be used (e.g. instead of an owner ID) to represent an owner of the record, and the end result is easy user deletion.

Night lights on mPulse Marble (81307)

The mPulse Globe, Marble mode now more accurately presents human-made light (e.g. from cities) in this release.

Bugs Fixed

85240: Cannot view dashboard data

mPulse will no longer save filter changes to dashboard objects that are marked read only. This fix also adds a "View" and "Clear" option to see and clear persisted filter preferences.

84231: rename the "RUM" category to "mPulse" in widget sidebar

The Widget Selection Panel, Widget Type, RUM category is now known as the mPulse category.

83386: Labels don't match between pie chart and multi-series widgets

Some mismatched labels in either pie chart or multi-series were resolved by this fix.

83553: NaN in dataless geowidget region popups

When clicking on the border of a dateless region, you got a popup with NaN due to no data. Now, regions are painted with a 0 opacity fill so that whole region is clickable, and now popups display with "No Data" instead of NaN.

83521: "getUrlData" doesn't return updates for error bar rows

DeltaDecorator updates rows based on pageLoadTimeTotal, which doesn't change for URLs when there is error. Setting it to beaconTotal in case of errors.

83410: Metrics by Dimension - No URL Data

No URLs would show due to how the dimension of a url request was being set.

83386: Labels don't match between pie chart and multi series widgets

Some formatting issues with labels were found and resolved.

83327: Change 'Other' to something else for OS and UserAgent

The "Other" option in mPulse dashboard drop-downs has been renamed to "(Unknown)."

83208: Geo Widget multi-select

Multi-select filtering is now in effect for the Geography widget.

83174: Firing Test alerts doesn’t appear to be working

An underlying issue with alert firing was resolved.

83125: Some beacons from Tokyo are showing up as region 40, which our FIPS to HASC table is putting into Fukushima

Static FIPS codes for Nagasaki and Okinawa have been added.

82806: Set a limit for the number of Alert incidents to display

Too many alerts in Alert History cause browser to hang, alerts do not appear.

82775: Add column name for percent column

The column heading was missing.

82645: Max User limit doesn’t apply

Applying the Max User Limit in the tenant editor didn't produce the expected result.

82423: Trailing space in domain config causes rejected beacon

Trailing spaces are now trimmed.

82332: URL Pattern match for Query String Parameter does not correctly work when clicking "Test"

An issue with in-box testing of query strings was resolved.

82281: Fiji bounding box is incorrect

The boundaries of Fiji were not accurately represented in the Globe or Geographic widget.

82097: Alert with page load does not get triggered with additional criteria added

Adding additional criteria proved to cause an underlying failure in a given alert.

81988: Dashboard throwing Java error when no filter applied

The underlying source code didn't handle a null filter.

81872: URL aggregation definition is opposite of what it should be

The aggregation code was unexpectedly trying to resolve query strings as part of the URL. mPulse will no longer capture query parameter values in URLs where it is not expected.

81505: Uncaught TypeError: Cannot read property 'length' of null; JS line 11825

Additional null checking has been added to detect further occurrences of this error.

81209: Cannot read property 'name' of undefined

Additional null checking has been added to detect further occurrences of this error.

81010: Remove spark line if there is no data

Where there is no data an array of all zeroes was being returned. This is changed to return null.

81265: mPulse Decimal Values for Color Thresholds

Setting a widget's Color Threshold value from an integer to an unsupported number with a decimal value would freeze the dashboard, requiring the widget to be deleted. Now, decimal values are supported.

80788: Uncaught TypeError: Cannot read property 'style' of undefined; JS line 2157

Additional null checking has been added to detect further occurrences of this error.

80784: Uncaught TypeError: Cannot read property 'substring' of undefined; JS line 603

This error occurred if an undefined metricID was encountered. Now, this is handled.

80666: TypeError: this.widget.filter is null; JS line 1192

Additional null checking has been added to detect further occurrences of this error.

80447: Cannot read property 'GetAllIDs' of null

This likely timing error occurred in mPulse Central.

78644: java.lang.NullPointerException

This null exception occurred in the Recording Editor or during clip conversion.

75992: mPulse Timers: sometimes MAX is assigned, but MIN is not

While read/restoring data, MIN gets set to `Integer.MAXVALUE`, which in this case resulted in large spikes in dashboard cycles. Now, if MAX exists then MIN is set to zero.

74395: java.lang.NumberFormatException exception

An emptiness check has been added to null check to prevent or detect further occurrences of this dashboard creation error.

62549: Fixing Kosovo on Globe

Because Kosovo wasn't listed as an official an attempt to filter by Kosovo failed. Now, ISO2 overrides will treat Slovenia as country ISO code XK.

53764: Region codes for Slovenia not working

Applying the Slovenian region code didn't work as expected.