SOASTA 55.0.1 (mPulse 7950.23.4)

May 24, 2015

Table of Contents

SOASTA 55.0.1 (mPulse 7950.23.4) 1
This update deployed improvements for stability and performance 1
SOASTA 55 (mPulse 7950.23) 2
Features 2
Custom Dimensions 2
Value Sources Used to Define Dimensions 3
URL Patterns 4
Dimension Types 4
Alias Maps 5
Defining a Dimension Using Query Parameter 8
Defining a Dimension Using Cookie 9
Defining a Dimension Using User Agent 10
Defining a Dimension Using JavaScript Variable 11
New Dashboard Dimensions 12
Beacon Type Dimension 12
Device Type Dimension 15
Device Manufacturer Dimension 16
Connection Type Dimension 18
Internet Service Provider (ISP) Dimension 19
Angular Support 20
Implement the Plugin Code 21
Setting and Refreshing the ResourceTimings Buffer 22
Set the Resource Timings Buffer 22
Clear the Resource Timings Buffer 22
Metrics and Sessions in SPA Apps 23
New mPulse APIs 24

mPulse Beacon API 24
mPulse JavaScript API 24
SOASTA Repository API 24
Enhancements 25
Update sun, clouds, lighting and Starfield on a timely basis (82043) 25
Use 8k / 3 hour clouds (82042) 25
Top 10 Browser widget is showing Browser family instead of specific browser version in the rank (87582) 25
Bugs Fixed 26

SOASTA 55.0.1 (mPulse 7950.23.4)

This update deployed improvements for stability and performance

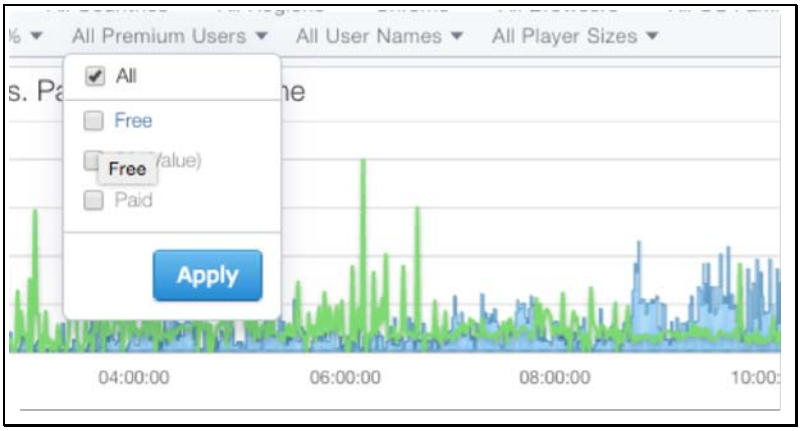
SOASTA 55 (mPulse 7950.23)

Features

Custom Dimensions

App Admins can define dimensions to track on each beacon. Beacon-level dimensions can be defined using one of six supported pattern-matching techniques, or value types found in the Configure Apps box, Advanced view, Dimensions tab.

SOASTA recommends defining dimensions on value types that have relatively few possible values. With that in mind, a good example is a dimension used to track free and paid users on a domain.



Dashboard performance may decrease if a custom dimension returns more than 100 unique values. So, categorical dimensions work well because they can only return a limited number of possible values.

Identity-tracking dimensions, however, are riskier for performance because they can return millions of unique values. Displaying a list of 1 million values in a dashboard will take longer to load and analyze than a list of 100 values. Examples include: user ID, session ID, and username.

Tenants are each limited to 10 custom dimensions.

For performance reasons, this feature should be used according to best practices. Best practices will prevent data explosions that are counter-productive and could result in performance hits against your app.

Within individual dimensions, an alias map can be used to create user-friendly values for display in dashboards.

Each of the six supported value types can be either **text** (any value), **numeric** (Boomerang will convert it to an integer or, if it fails, it will not persist), or **Boolean** (follows JavaScript truthiness).

> Boolean will do JavaScript truthiness except for in the JavaScript value source itself. mPulse's server side has been tweaked for JavaScript truthiness (e.g. if it's there, then its value is 1, and if not there, then 0).

Value Sources Used to Define Dimensions

The App Administrator can use any of the following Value Source techniques to define a dimension:

- **XPath** – extracts the value of a DOM Element, located via XPath.

Note: If you require compatibility with Internet Explorer, you can only use a limited subset of the XPath language.

- **URL**– extracts a portion of the URL, using a regular expression
- **Query Parameter** – extracts the value of a query string parameter
- **Cookie** – extracts a portion of a cookie value, using a regular expression
- **User Agent** – extracts a portion of the “User-Agent” header, using a regular expression
- **JavaScript variable** – extracts the value of a JavaScript variable

No matter which type you choose, the value is extracted on the client side (inside the browser).

URL Patterns

Some custom dimensions only apply to a portion of your website. In those cases, you can limit the set of URLs for which the dimension value will be extracted, using the "URL Pattern" input. For example, if a custom dimension only applies to the online store portion of your site, the URL pattern might be `http://www.mywebsite.com/store/*`.

Note that URL patterns are **not** regular expressions. Their syntax is limited to asterisk wildcards

Dimension Types

Dimensions may be of three (user facing) types:

- **Text** – Some text on a page; can be any value
- **Numeric** – A count on a page such as the number of advertisements
- **Boolean** - A 0 or 1 for Boolean (implemented as a Numeric)

To define a Text type Dimension by Alias Map, define some token value on the left and an English-readable string on the right. This will match a value defined in the web or mobile app. Similarly, a numeric value from a JavaScript can be aliased to have a text value.

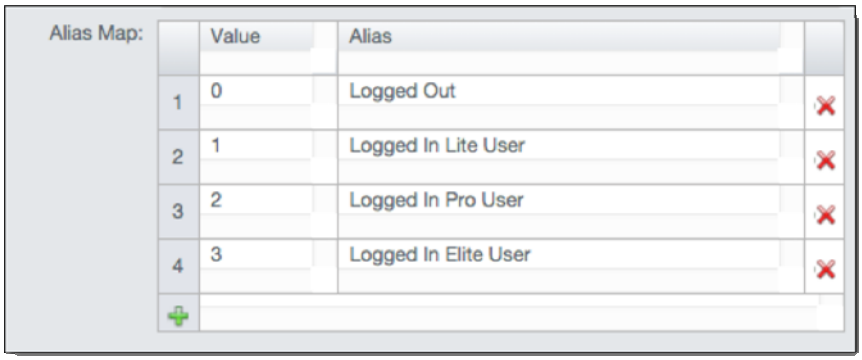
Alias Maps

In some cases, the actual value extracted at runtime is not the value that you want to display in dashboards. For example, let's say you're setting up an "Account Type" dimension, whose values are "Free", "Pro", or "Enterprise". The value will be extracted from a cookie, but it turns out that the cookie uses internal codes, like "f", "p" and "e".

You can define an **Alias Map** to translate the internal values collected in the browser to "user-friendly" values that will be shown in the mPulse dashboard and alert UIs.

For example, a simple Alias Map for user authentication status plus user type might be defined using the following user-defined values and aliases (with the numeric dimension type):

0 = Logged Out
1 = Logged In Lite User
2 = Logged In Pro User
3 = Logged In Elite User

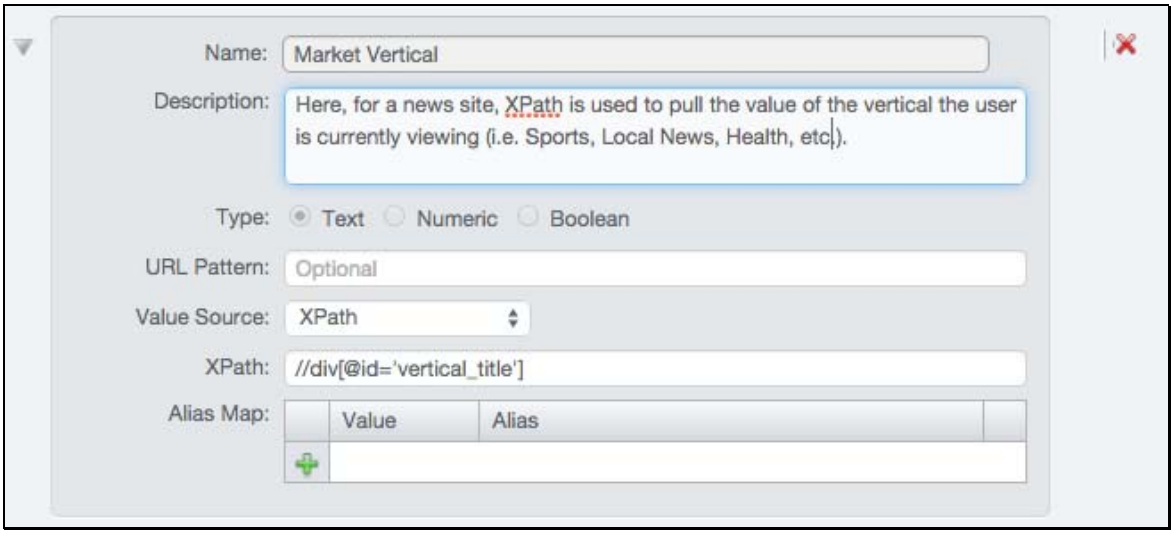


For the map defined above, mPulse will get the numeric value and place it on the beacon, save that value to the database, and then display the user-friendly alias in dashboards (e.g. as Logged Out, Logged In Lite User, etc.).

Defining a Dimension using XPath

Xpath matching works similarly to the matching presented in mPulse via the Metrics tab and can also be filtered by URL pattern. When this value source is selected, define an Xpath from which to match.

For example, a news site might create a custom dimension called "Market Vertical," which uses XPath to pull the type of news content that the user is currently viewing.



1. Enter the dimension name and (optionally) description.
2. Select the desired data type (Text, Numeric, or Boolean).
3. Optionally, if the dimension only applies to certain pages within your site, set the URL Pattern to match those pages.
4. Set the Value Source drop-down to *XPath*.
5. Enter the XPath of the DOM element that will contain the dimension value.

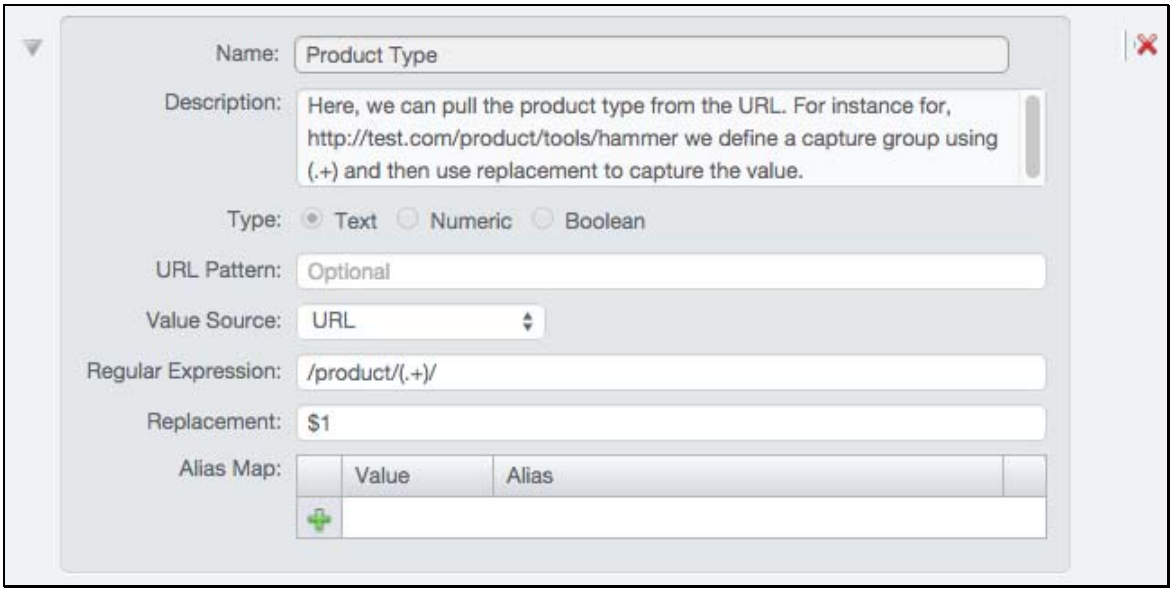
Note: If you require compatibility with Internet Explorer, you can only use a limited subset of the XPath language.

6. Optionally, configure an Alias Map to translate the DOM element values to user-friendly names.

Defining a Dimension using URL

URL matching works similarly to the matching presented in mPulse via the Page Groups tab and can also be filtered by pattern.

For example, an ecommerce site might create a custom dimension called "Product Type" that uses a Regex capture group to pull the product type from a URL, and a replacement value to capture the specific product value, such as *hammer*.

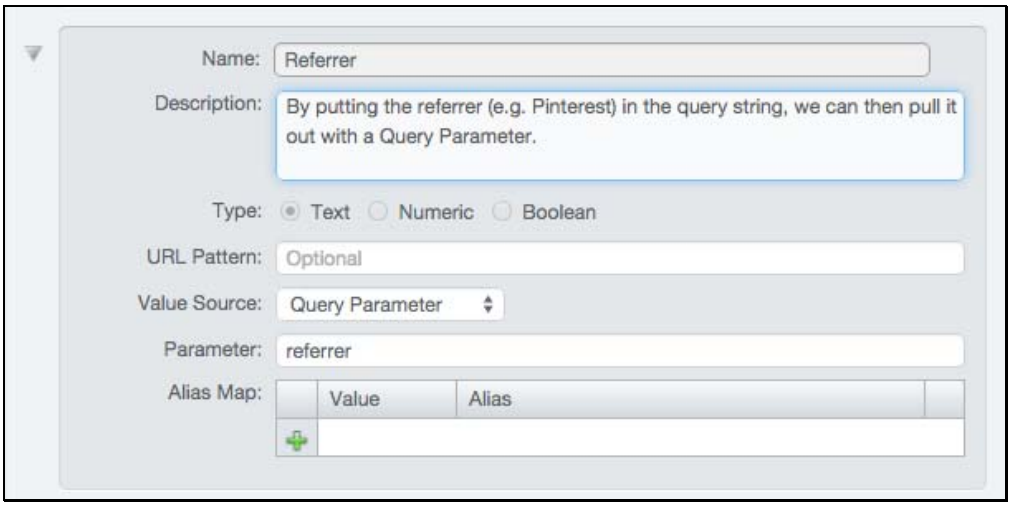


1. Enter the dimension name and (optionally) description.
2. Select the desired data type (Text, Numeric, or Boolean).
3. If the dimension only applies to certain pages within your site, set the URL Pattern to match those pages.
4. Set the Value Source drop-down to *URL*.
5. Enter the regular expression that captures the portion of the URL you want to extract.
6. Optionally, configure an Alias Map to translate the extracted strings to user-friendly names.

Defining a Dimension Using Query Parameter

Query Parameter matching works similarly to the matching presented in mPulse via the Page Groups tab. When this value source is selected, define a query string parameter from which to match. The URL used here must have a query string (e.g., a "?" that precedes the non-hierarchical data that follows).

For example, the Query Parameter value source can be used to pull the value of the common query parameter, *referrer*.

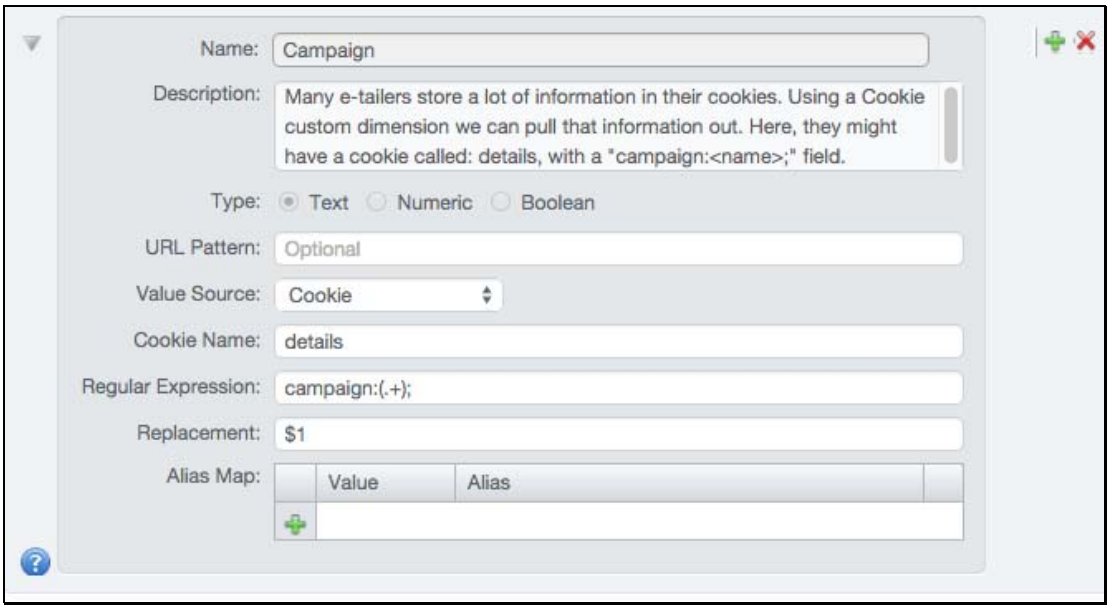


1. Enter the dimension name and (optionally) description.
2. Select the desired data type (Text, Numeric, or Boolean).
3. If the dimension only applies to certain pages within your site, set the URL Pattern to match those pages. **This is strongly recommended for the Query Parameter value source**, because most query parameters only apply to a subset of URLs.
4. Set the Value Source drop-down to *Query Parameter*.
5. Enter the name of the query parameter that contains the dimension value.
6. Optionally, configure an Alias Map to translate the extracted strings to user-friendly names.

Defining a Dimension Using Cookie

Cookie matching works similarly to the matching presented in mPulse via the Page Groups tab.

For example, many e-tailers store the marketing campaign value(s) that resulted in a web site hit. Using the Cookie value source to define a custom dimension, we can extract that information from a cookie named *details*, and then get the campaign name using a replacement value.

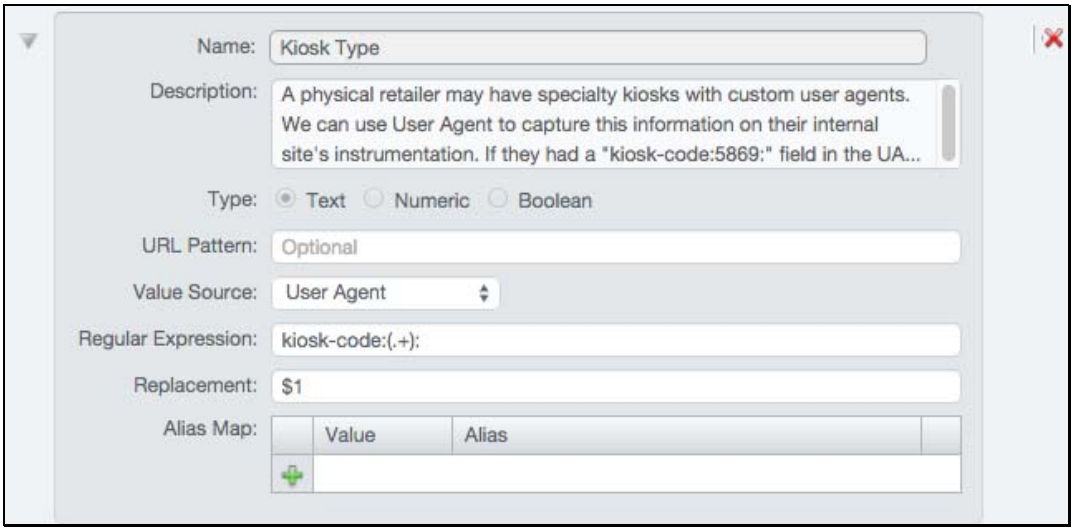


1. Enter the dimension name and (optionally) description.
2. Select the desired data type (Text, Numeric, or Boolean).
3. If the dimension only applies to certain pages within your site, set the URL Pattern to match those pages.
4. Set the Value Source drop-down to *Cookie*.
5. Enter the cookie name.
6. Enter the regular expression that will determine which portion of the cookie value is used.
7. If you have Regular Expression replacement strings (e.g. from a reference) then they can be used in the Replacement field. The string entered here should be the text that will replace a regex match after a search.
8. Optionally, configure an Alias Map to translate the extracted strings to user-friendly names.

Defining a Dimension Using User Agent

User Agent matching works similarly to the matching presented in mPulse via the Page Groups tab.

For example, a physical retailer may have specialty in-store kiosks whose user agents (e.g. browsers) are identified by a kiosk-code such as "kiosk-code:5869:". In such a case, mPulse can capture this user-agent string from their internal site's instrumentation.

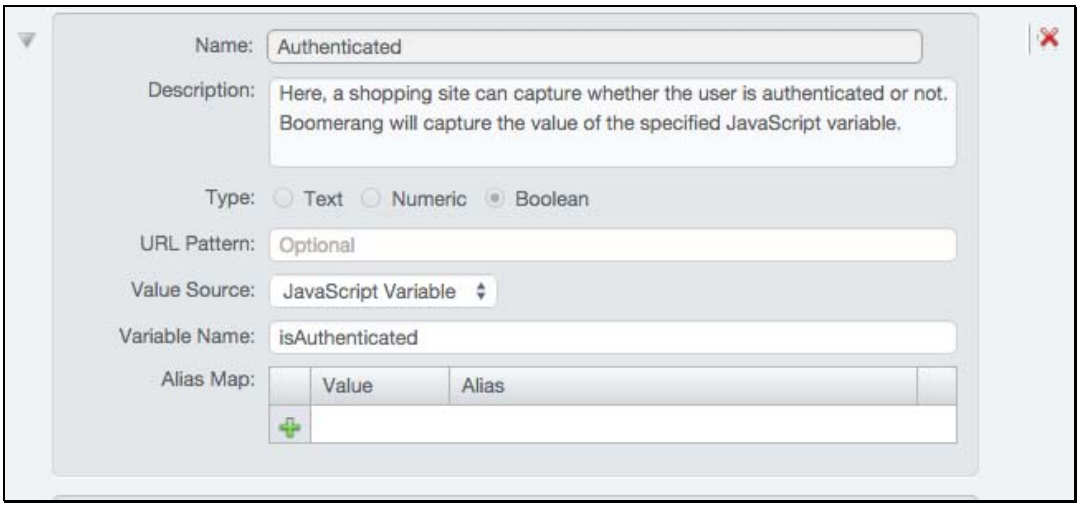


1. Enter the dimension name and (optionally) description.
2. Select the desired data type (Text, Numeric, or Boolean).
3. If the dimension only applies to certain pages within your site, set the URL Pattern to match those pages.
4. Set the Value Source drop-down to *User Agent*.
5. Enter the regular expression that will determine which portion of the User-Agent header is used.
6. If you have Regular Expression replacement strings (e.g. from a reference) then they can be used in the Replacement field. The string entered here should be the text that will replace a regex match after a search.
7. Optionally, configure an Alias Map to translate the extracted strings to user-friendly names.

Defining a Dimension Using JavaScript Variable

JavaScript Variable matching works similarly to the matching presented in mPulse via the Page Groups tab.

This option allows you to handle more complex extraction scenarios, by writing JavaScript that captures the value and then makes it available to mPulse.



1. Enter the dimension name and (optionally) description.
2. Select the desired data type (Text, Numeric, or Boolean).
3. If the dimension only applies to certain pages within your site, set the URL Pattern to match those pages.
4. Set the Value Source drop-down to *JavaScript Variable*.
5. Enter the JavaScript variable name.
6. Optionally, configure an Alias Map to translate the extracted strings to user-friendly names.

New Dashboard Dimensions

The following new mPulse Dashboard dimensions are available for the Dashboard Filter toolbar: Beacon Type, Device Type, Device Manufacturer (Pro and Enterprise only), Device Model (Enterprise only), Connection Type (Pro and Enterprise only), and ISP (Enterprise only).

Dashboard dimensions are included in the mPulse System Dashboards and can be added to custom dashboards as well using the lower panel Dashboard Editor.

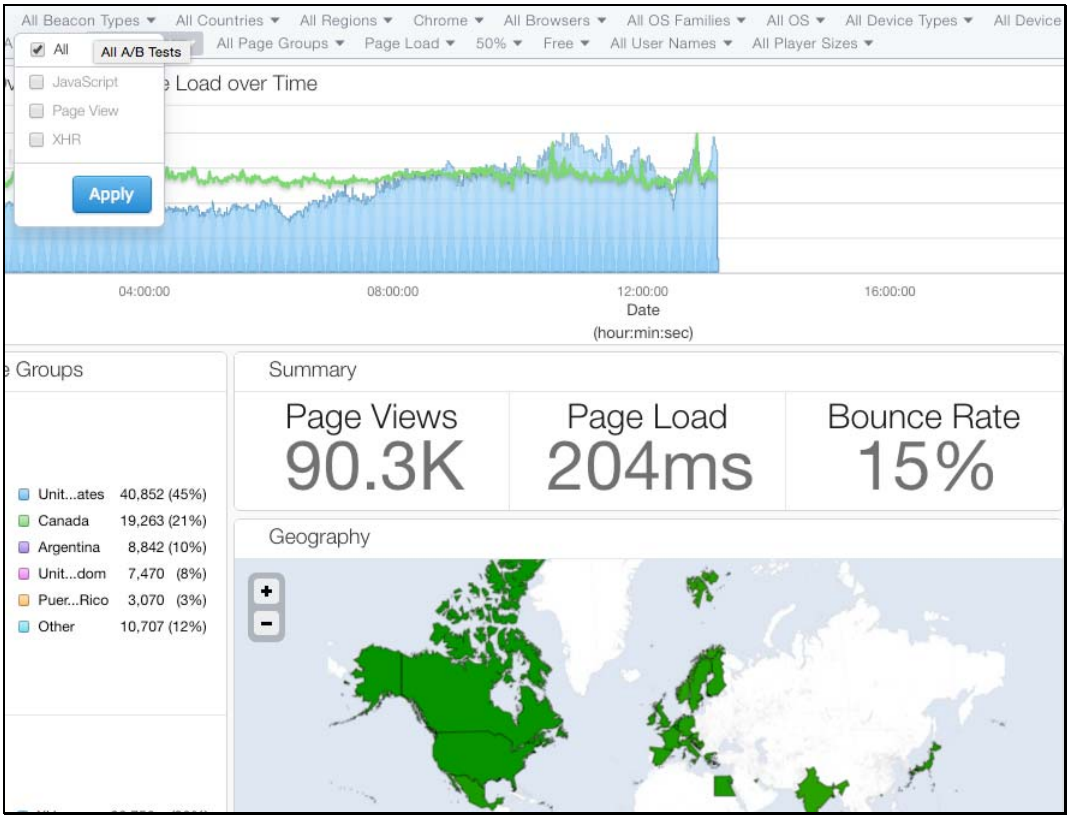
As in all prior releases, to filter by a dimension, check the desired value in the filter and click Apply. The dashboard filter toolbar's dimension label reflects the current selection.

Beacon Type Dimension

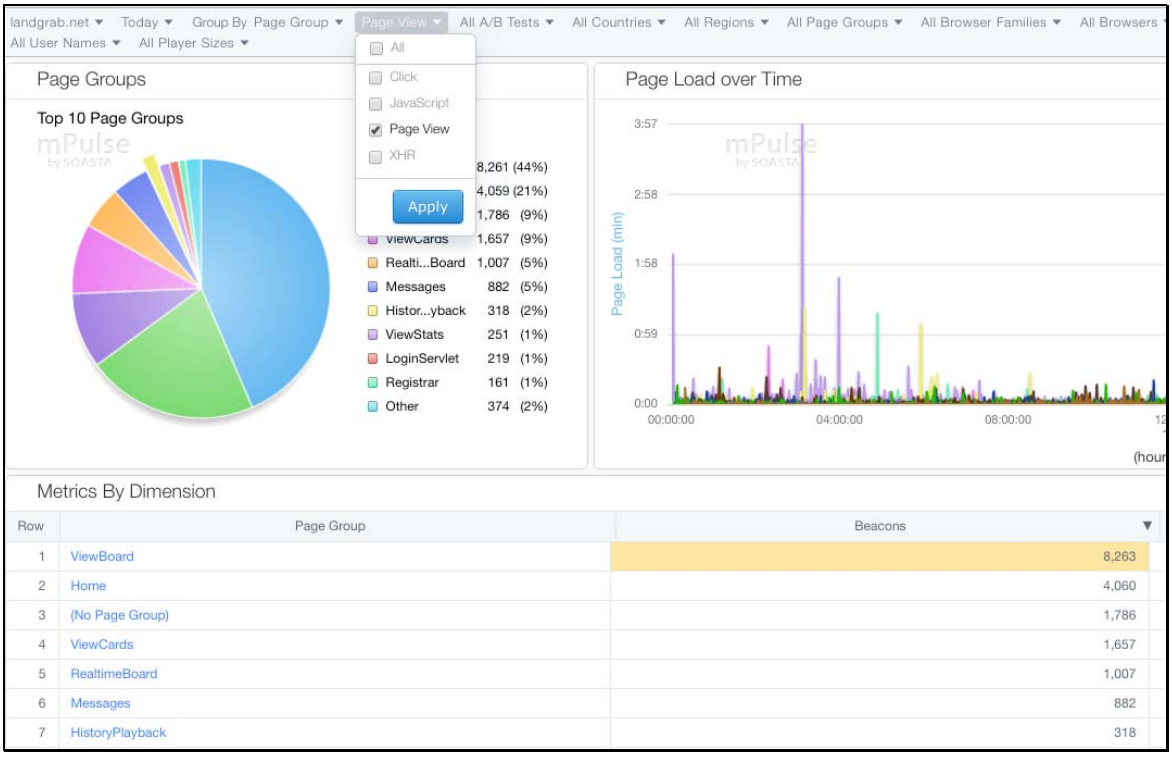
The new Beacon Type filter is now available to all mPulse users. It allows you to analyze the performance of “regular” page views separately from Ajax requests (XHR).

The following additional Beacon Types are presented in the Beacon Types drop-down:

- **All** – This dashboard default filter shows every beacon type and is the dashboard default

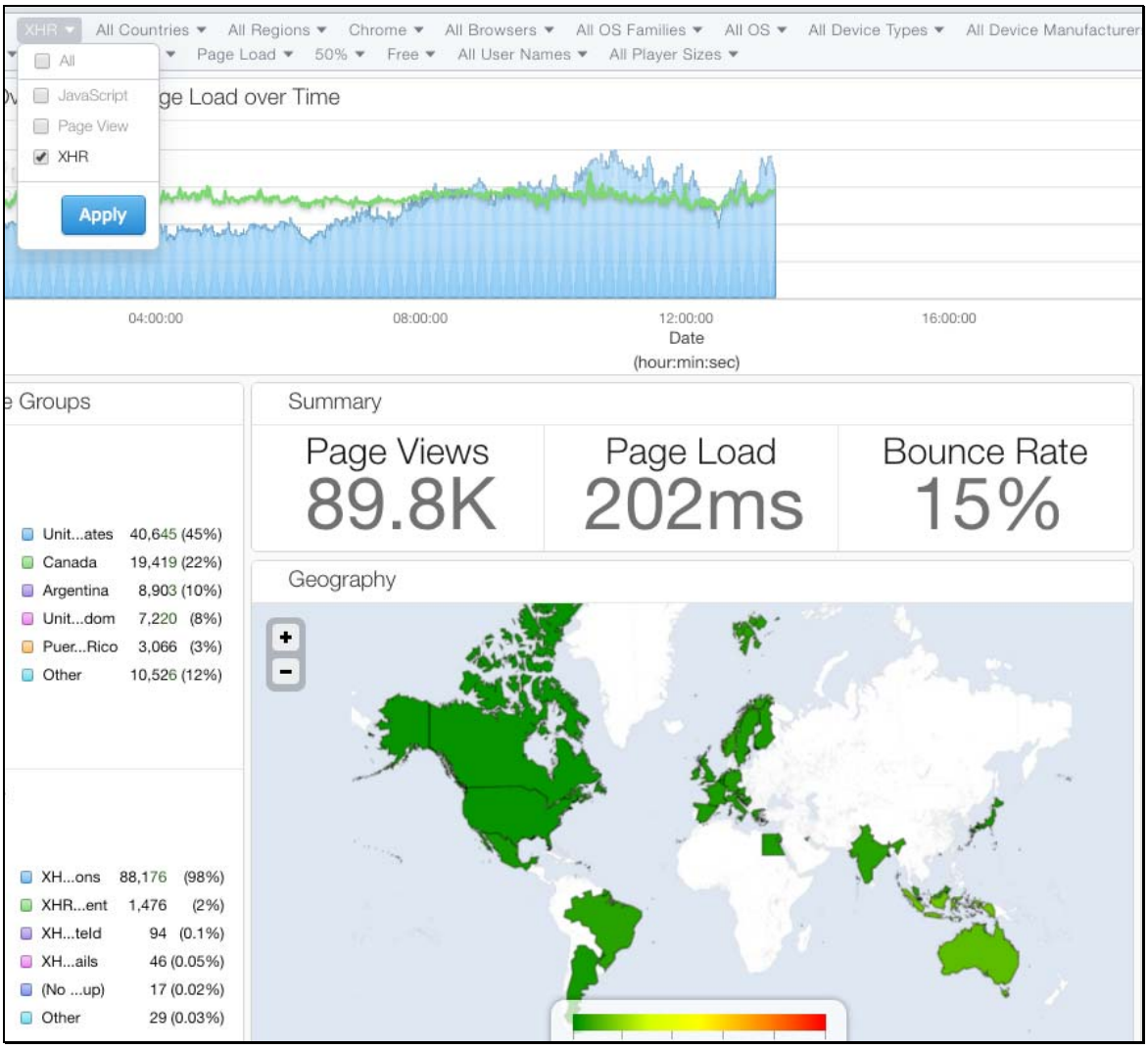


- **Page View** – This filter shows only beacons that pertain to page load beacons (excluding XHRs and other beacon types). A page load involves an interaction with the HTTP server (such as getting a page, posting from a form, etc.).



- **XHR** – This filter shows only Ajax or other XHR beacons

Note: The App Admin must have Auto-XHR enabled for the given app via the Configure App box, General tab. Contact your SOASTA representative about enabling this Limited Availability feature.



Other less common beacon types include:

- **Click** – This filter shows only beacons generated by a user clicking an element within the page.
- **JavaScript** – This filter shows only beacons generated by calling the Boomerang API from JavaScript.

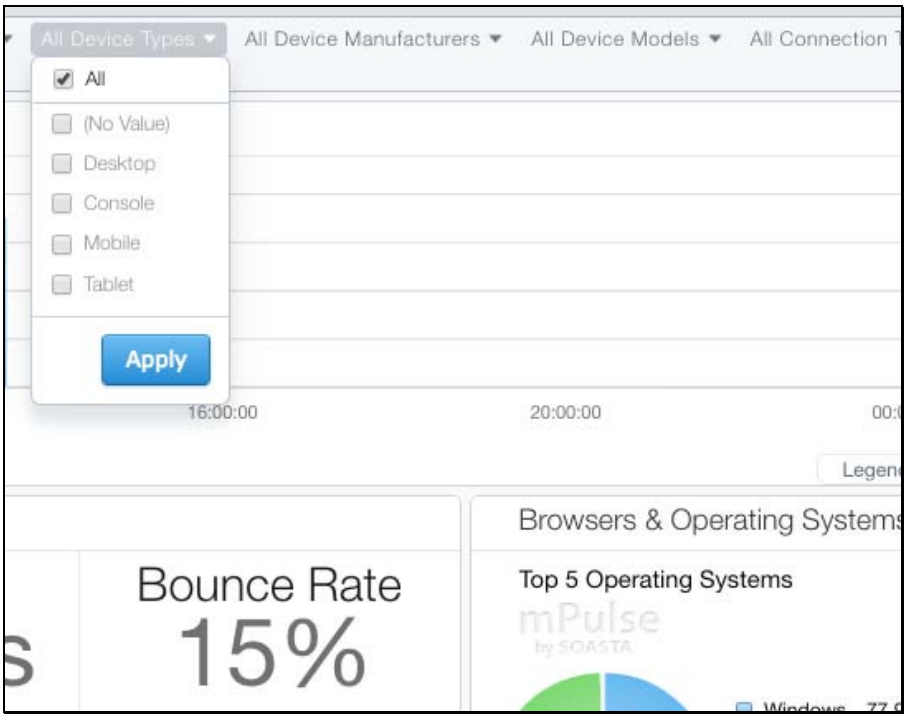
Device Type Dimension

The new Device Type dimension is now available to all mPulse users. It allows you to analyze the performance of your site based on whether the end-user is browsing via a desktop computer, tablet, or smartphone.

The specific values are:

- Desktop – includes Windows, Mac, Linux, and any other type of laptop or desktop computer.
- Mobile – includes all iOS and Android phones, as well as devices like iPod Touch.
- Tablet – includes all iOS and Android tablets.

Users can select more than one type whenever *All* is not selected.



Device Manufacturer Dimension

The new Device Manufacturer dimension is now available to mPulse Pro and Enterprise users.

To use this filter, you must first select a Device Type value. Here is an example, after setting the device type to “Mobile”:

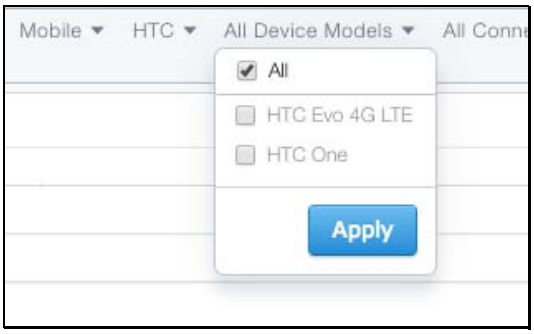
Users can select more than one manufacturer whenever *All* is not selected.

The Device Manufacturer dimension is included in the default Summary System Dashboard (for Enterprise and Pro users only).

Device Model

The new Device Model dimension is now available, by request, to mPulse Enterprise users.

To use this filter, you must first select a Device Type and a Device Manufacturer. Here is an example, after setting the device type to “Mobile” and the Device Manufacturer to “HTC”:

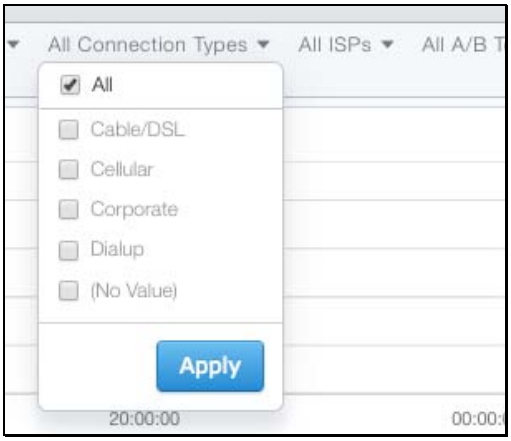


Users can select more than one model whenever *All* is not selected and the Device Model list will reflect relevant models for that selection (or multi-selection).

Connection Type Dimension

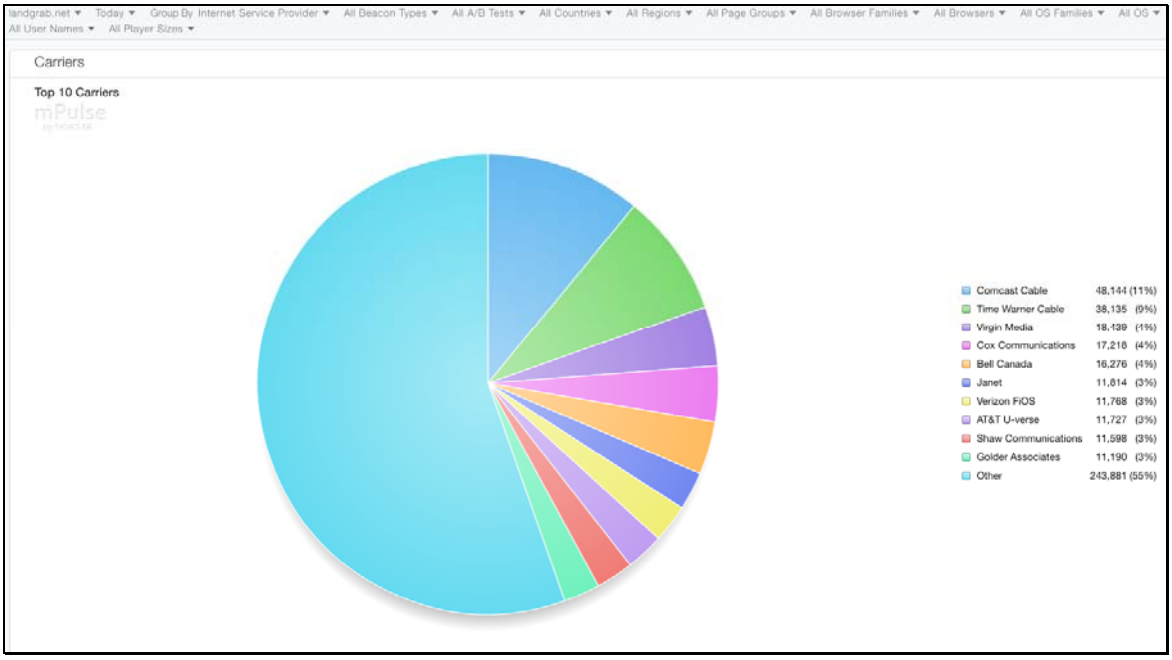
The new Connection Type dimension is now available to mPulse Pro and Enterprise users.

This dimension includes types for Cable/DSL, Cellular, Corporate, and Dialup.

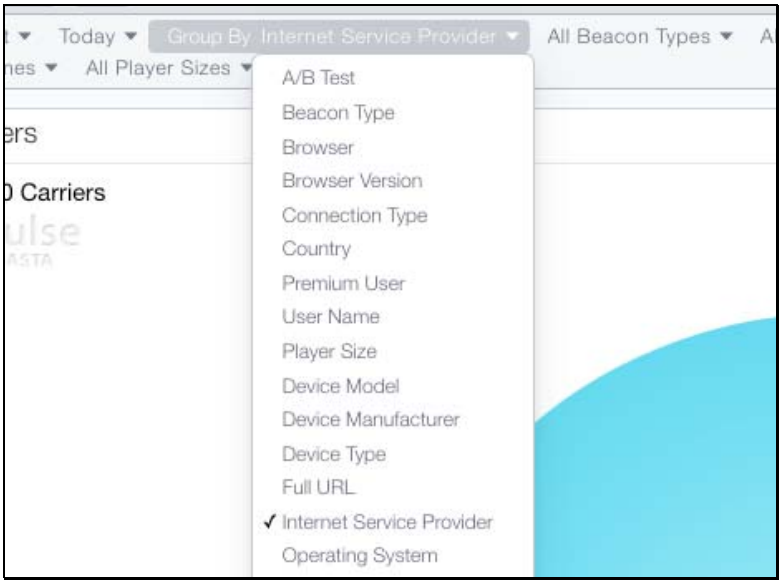


Internet Service Provider (ISP) Dimension

The new ISP dimension is now available, by request, to mPulse Enterprise users.



In addition to filters, ISP is also a “Group By” option in the Dimensions dashboard, and any custom dashboard that includes a “Group By” option.



Angular Support

The mPulse™ Boomerang AngularJS plugin allows users to automatically monitor their Single Page App's (SPA) navigations beyond the initial page load.

In non-SPA scenarios, your visitors do a full navigation with every new page they visit. The browser requests the new page's HTML from the server, and the CSS and JavaScript is re-executed once all of the components have been loaded.

In SPA apps, only the first page that the visitor loads is a full navigation. All subsequent navigations are handled by frameworks like AngularJS, where it dynamically pulls in the content it needs to render the new page, without doing a full navigation.

Boomerang works well in the traditional, non-SPA scenario, where a full navigation occurs. On each page load, it tracks performance information about the entire page load experience. However, in SPA scenarios like AngularJS, only the first page triggers a full navigation. Thus, any subsequent pages your AngularJS visitors see to do not get tracked by Boomerang.

With the Boomerang AngularJS Plugin, Boomerang is able to track all of the SPA navigations beyond the first, initial navigation.

To do so, the Boomerang AngularJS plugin listens for several life cycle events from AngularJS, such as `$routeChangeStart.` Once it gets notified of these events, the Boomerang AngularJS plugin starts monitoring the page's markup (DOM) for changes. If any of these changes trigger a download, such as an XHR, a new image (IMG) getting inserted, or new CSS (LINK) being fetched, then the Boomerang AngularJS plugin monitors those resources as well. Only once all of these new resources have been fetched does Boomerang AngularJS consider the SPA navigation to be complete.

Note: Users should note that any XHR outside of a route change (e.g. outside of a navigation) will not be captured as a page load (e.g. as a beacon).

Implement the Plugin Code

The following code snippet initializes the plugin and should be placed in your Angular app or module startup.

The Boomerang AngularJS plugin does not require any additional instrumentation of your app, its clicks or URLs, nor does it require any additional directives or markup changes.

```
/*
* Installation:
*
* Somewhere in your Angular app or module startup, call
BOOMR.plugins.Angular.hook($rootScope).
*
* For example:
*/
 angular.module('app')
   .run(['$rootScope', function($rootScope) {
     var hadRouteChange = false;
      $rootScope.$on("$routeChangeStart", function() {
       hadRouteChange = true;
     }
     function hookAngularBoomerang() {
       if (window.BOOMR && BOOMR.version) {
         if (BOOMR.plugins && BOOMR.plugins.Angular) {
           BOOMR.plugins.Angular.hook($rootScope, hadRouteChange);
         }
         return true;
       }
     }
     if (!hookAngularBoomerang()) {
       if (document.addEventListener) {
         document.addEventListener("onBoomerangLoaded", hookAngularBoomerang);
       }
       else if (document.attachEvent) {
         document.attachEvent("onpropertychange", function(e) {
           e = e || window.event;
           if (e && e.propertyName === "onBoomerangLoaded") {
             hookAngularBoomerang();
           }
         });
       }
     }
   });
```

Setting and Refreshing the ResourceTimings Buffer

The Boomerang AngularJS plugin gathers ResourceTiming data for all SPA navigations. ResourceTiming data contains performance metrics about all of the resources (IMG, CSS, etc) that were downloaded to construct the page (in supporting browsers). This data is used in the Waterfall dashboards.

Users should note that the Resource Timing API in use in browsers enforces an upper limit of 150 resources tracked. While this upper limit can be manipulated by the developer in order to track more resources (`window.performance.setResourceTimingBufferSize()`), there are likely performance tradeoffs, and SOASTA doesn't attempt to make such impactful changes for the user.

For this reason, SOASTA provides the following additional code examples, which can be used to set and/or refresh the ResourceTiming buffer after each Boomerang beacon in order to ensure that beacons beyond the initial 150 are retrieved.

Set the Resource Timings Buffer

```
var setResourceTimingBufferSize == window && window.performance &&
(performance.setResourceTimingBufferSize ||
performance.webkitSetResourceTimingBufferSize);
if (setResourceTimingBufferSize) {
    setResourceTimingBufferSize(<size>);
}
```

Clear the Resource Timings Buffer

```
(function(w){
  if (!w || !w.performance) {
    return;
  }

  var clearResourceTimings = performance.clearResourceTimings ||
performance.webkitClearResourceTimings;
  if (clearResourceTimings) {
    BOOMR.subscribe("onbeacon", clearResourceTimings);
  }
})(window);
```

Metrics and Sessions in SPA Apps

Because mPulse doesn't evaluate beacons until right before the beacon is set, Page Groups, Timers, and Metrics are tracked in SPAs using the same methods available in non-SPA apps (e.g. XPath, Regex, etc.).

Similarly, sessions are tracked the same in SPA apps as in non-SPA apps. The session length will increase every time the route changes and the session will be kept alive as long as user actions occur.

New mPulse APIs

In addition to the mPulse Boomerang Angular API (also discussed above), this release debuts three additional APIs that expand mPulse accessibility for web and native app developers.

mPulse Beacon API

This new API provides integration support via REST for both web and mobile apps, and also includes native integration for both iOS and Android apps.

The mPulse Beacon API (Version 2) exposes a simple way to send and log beacon data from configured web and native apps.

mPulse JavaScript API

The mPulse JavaScript API (Version 1) exposes a simple way to send and log beacon data from JavaScript. This API is a standalone API but also is a consumer of the mPulse Beacon API, REST API.

The mPulse JavaScript API provides an initialization method, and a method for each of the custom types supported within mPulse: Metrics, Timers, Page (or View) Groups, and for user-defined Dimensions.

SOASTA Repository API

The SOASTA Repository API is a collection of REST APIs that allows users to:

1. Authenticate and obtain a security token.
2. Interact with objects (CRUD).
3. Read and write Seed Data content in CSV form.

Enhancements

Update sun, clouds, lighting and Starfield on a timely basis (82043)

This release includes various improvements to the timely display of the Sun, clouds, lighting, and the Starfield background that users will find to be more responsive during Globe rotation, and particularly so on larger displays.

Use 8k / 3 hour clouds (82042)

This release also includes improvements to the use of clouds in the Globe widget display.

Top 10 Browser widget is showing Browser family instead of specific browser version in the rank (87582)

This release adds support for Browser version, OS version, and Full URL to the Multi-Dimension widget (except in the Full URL metric), Metrics by Dimension, Multi-Histogram, and Group by Filter widgets.

Bugs Fixed

93364: Alert configured to fire an email every 10 minutes, fired every minute, but sent no email

An Alert configured to fire an email every 10 minutes was triggered each minute but no email was sent.

93008: External Data Service fault: 'Error obtaining external data. The AppDynamics HTTP request failed (302 Found).'

This error occurred while attempting to configure and connect to an External Data Source.

92901: Alert dashboard widget images were blank in emails

Alert dashboard Images received via email were blank in some cases, whereas in mPulse dashboards those same widgets were shown with the expected values.

92630: java.lang.NullPointerException

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

92459: All System Dashboards should have the new Dimension filters by default

System Dashboards in mPulse unexpectedly failed to show some available dimensions.

92361: Date picker layout issues in the widget (buttons not visible)

While using the DateTime picker on a widget at the end of a dashboard page, the "Apply" button disappeared.

92306: Metric Histogram X-axis does not align with rollovers

The metric histogram rollovers did not align as expected with the X-axis values in display for rollovers.

91990: Custom combined widgets should go under right category

Custom combined widgets in mPulse will now appear in the correct category (e.g. RUM) in the Widget Selection Panel.

91796: java.lang.NullPointerException

This null pointer exception occurred in an mPulse dashboard.

91699: Date/Time Filter 'On' Selection Not Customizable at Widget Level

After setting the attribute 'Date/Time' and Operator to 'On', the dashboard filter toolbar's date field (e.g. to the right of the 'On' operator) could not be set to any specific date.

91661: Cannot read property 'data' of undefined; JS line 8162

This error occurred in mPulse Central.

91543: java.lang.NullPointerException at com.soasta.web.concerto.reporting.fb.a (fb.java:522)

An error occurred when the Globe dashboard started polling for data before the start time was set.

91212: Waterfall: Beacons list stops after 11 beacons

The Beacons list was unexpectedly concatenated.

91128: boomerang.responseEnd is not a function; JS line 1399

This error occurred in an mPulse dashboard.

91060: java.lang.NullPointerException at com.soasta.web.concerto.command.rum.h.a (h.java:1438)

mPulse was sending null as a dimension filter value. We already had a default of "" for this but the code was assuming that a value couldn't be null. I now check for null and fall through to the default.

91052: java.lang.NullPointerException

This null pointer error occurred in an mPulse dashboard.

90982: com.caucho.hessian.client.HessianConnectionException: HTTP request failed. at com.soasta.common.k.a.invoke (a.java:262)

An error occurred while getting Globe data from the underlying API.

90426: mPulse on mPulse: Dashboard event doesn't fire until someone looks at it

Allow all widgets to receive initial poll data, even if they're hidden.

90420: java.lang.NullPointerException

This null pointer exception occurred in mPulse Central.

90407: Shape of Michigan is not very mitten-like

The Michigan peninsula landform was not realistic.

90248: By-minute query returns a 500 internal error for the tenant

A valid by-minute query unexpectedly resulted in an HTTP 500 error.

90140: Uncaught TypeError: Cannot read property 'style' of null; JS line

This error occurred in the mPulse Globe dashboard.

90052: java.lang.NullPointerException at com.soasta.web.concerto.command.rum.s.a (s.java:414)

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

89900: java.lang.NullPointerException

This null error occurred in an mPulse dashboard.

89763: java.lang.NullPointerException at com.soasta.web.concerto.command.rum.h.a (h.java:910)

This error was due to a null domain filter.

89729: DataService alert action caches token. Should refresh it from ServerContext.

Alerts stopped firing at the change of a Calendar month, exposing an underlying token caching issue.

89664: java.lang.NullPointerException

This null error occurred in mPulse Central.

89052: java.lang.NullPointerException at com.soasta.web.concerto.command.rum.h.a (h.java:1433)

This null error occurred in mPulse Central.

89763: java.lang.NullPointerException at com.soasta.web.concerto.command.rum.h.a (h.java:910)

This error was due to a null domain filter.

89729: DataService alert action caches token. Should refresh it from ServerContext.

An alert caching issue resulted in non-configured alert events firing.

89584: Uncaught TypeError: Cannot read property 'CustomTimers19015' of undefined; JS line 92

This error occurred in an mPulse dashboard.

88665: Export to CSV not working for Metrics by Dimension widget in Firefox

The CSV export function unexpectedly failed when applied to the given widget.

88648: Cannot set property 'innerHTML' of undefined; JS line 25

This error occurred in an mPulse dashboard.

88341: com.snowflake.client.jdbc.SnowflakeSQLException: SQL compilation error

This error was caused by there being no OS (or OS Families) for the specified query.

88175: Alerts using time over time widget have y-axis capped to value below data points

An email alert sent using the "Time over Time" widget showed no data points, and the y-axis was capped at 100 ms, while data points were higher.

88135: Title cannot be overwritten for multi-dimension breakdown widget

The title bar of the Multi-Dimension Breakdown widget was unexpectedly read only.

87997: Missing region data for India

mPulse unexpectedly missed data for regions within India that were getting traffic.

87992: Error during update alert incident email when there were Japanese characters in the email subject

This error occurred during update of an alert incident email if there were Japanese characters in the email subject.

87943: URL shows up in Dimension Dashboard, Group By without being enabled

The Dimension dashboard had URL as a choice in the group-by when the domain does not have URL enabled.

87492: RumWidgetCommand.getTimeWindow does not handle the "On" comparator

This fix adds the expected support for the ON date picker.

87331: Cannot read property 'alert' of undefined

This error occurred in an mPulse dashboard.

87304: Cannot read property 'alert' of undefined; JS line 184

This error occurred in an mPulse dashboard.

87107: Uncaught TypeError: Cannot read property 'appendChild' of undefined; JS line 5767:

The Date Time Picker was attempting to access the Dashboard element when it didn't exist.

87053: inFilter is not defined

An underlying dashboard filter error occurred.

86980: For input string: "11111111111"

mPulse Central objects were incorrectly reporting their own locations resulting in this and similar errors.

86950: Uncaught TypeError: Cannot read property 'style' of undefined; JS line 682

This error occurred in an mPulse dashboard.

86937: Alerts with over the past "1 minute" condition does not fire

A configured alert didn't fire as expected when over the past "1 minute" was used.

86907: Uncaught TypeError: Cannot read property 'x' of undefined; JS line 1979

This error occurred in an mPulse dashboard.

86890: this.m_oSupplementalData is null; JS line 89

This error occurred in an mPulse dashboard.

86831: Invalid expired Ids data: ['Côte d'Ivoire',]

Incorrectly escaping IDs resulted in this error.

86803: TypeError: msg is undefined; JS line 44

An error occurred in mPulse Central.

86763: undefined is not a function; JS line 1230

This probable timing error occurred in mPulse Central.

86745: TypeError: null is not an object (evaluating 'indexedDB.open'); JS line 14

This error occurred in an mPulse dashboard.

86631: Uncaught TypeError: Cannot read property 'replace' of undefined; JS line 2117

This error occurred in an mPulse dashboard.

86963: Dashboard: Combining timers and metrics causes bad behavior

This fix updated underlying setTitle methods in order to make Combined Widget titles work as expected. Now, if widgets are combined in the chart, their various titles are combined.

88386: java.lang.NullPointerException

This null pointer happened in an mPulse dashboard.

86779: Development servers in list is not excluded from beacons

Adding a server in the Development list didn't stop beacons from flowing **from that server URL.**

86771: TypeError: this.updateVisitor is undefined; JS line 6652

This error occurred in an mPulse dashboard.

86626: TypeError: this.groupContainer is undefined; JS line 112

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

86583: Cannot read property 'widgetTypeID' of undefined

This error occurred in an mPulse dashboard.

86534: Uncaught TypeError: Cannot set property 'horizontal' of undefined; JS line 2099

This error occurred in an mPulse dashboard.

86523: Cannot read property 'substring' of undefined

This error occurred in an mPulse dashboard.

86292: com.soasta.common.resultsservice.ResultsServiceFault: The "from" timestamp must be less than the "to" timestamp. at java.lang.

This timestamp error occurred in mPulse Central.

86152: Uncaught TypeError: Cannot read property 'addLayer' of undefined; JS line 759

This error occurred in an mPulse dashboard.

85942: Uncaught TypeError: Cannot read property 'customMetricsFilteredByDate' of null; JS line 89

An invalid domain, or no domain filter causes a JS error in the mobile Summary widget.

85723: Object doesn't support property or method 'debounce'

This error occurred in the Globe widget.

85555: TypeError: this.groupContainer is undefined; JS line 92

The open method was being called in an mPulse dashboard before a group had been defined.

85248: Cannot read property 'FrameOverlay' of undefined

This error occurred in mPulse Central.

84369: Cannot read property 'Event' of undefined; JS line 78

An mPulse Alerts error occurred during configuration.

84128: ResultsServiceFault: A result with ID '6560' does not exist in the Results Server database - Waterfall Chart

The Waterfall widget had one method that did not handle routine errors such as result does not exist.

82978: this.m_oDomainSpecificInfo is undefined; JS line 174

A domain-related Alerts error occurred.

82826: Uncaught TypeError: Cannot read property 'fire' of undefined; JS line 1050

This error occurred in an mPulse dashboard.

81520: Global user-preference (in the User Preferences screen) for dark theme / light theme

This fix updates themes in the Globe Dashboard.

80831: Cannot read property 'pageId' of undefined

This error occurred in mPulse Central.

80691: NS_ERROR_FAILURE:; JS line 16828

This error occurred in an mPulse dashboard.

66715: mPulse Query API - Removing Backslashes in Response

Some of the mPulse Query APIs, "geography", "page-groups", "browsers", "timers-metrics", and "metrics-by-dimension" unexpectedly returned the response in JSON format with backslashes.