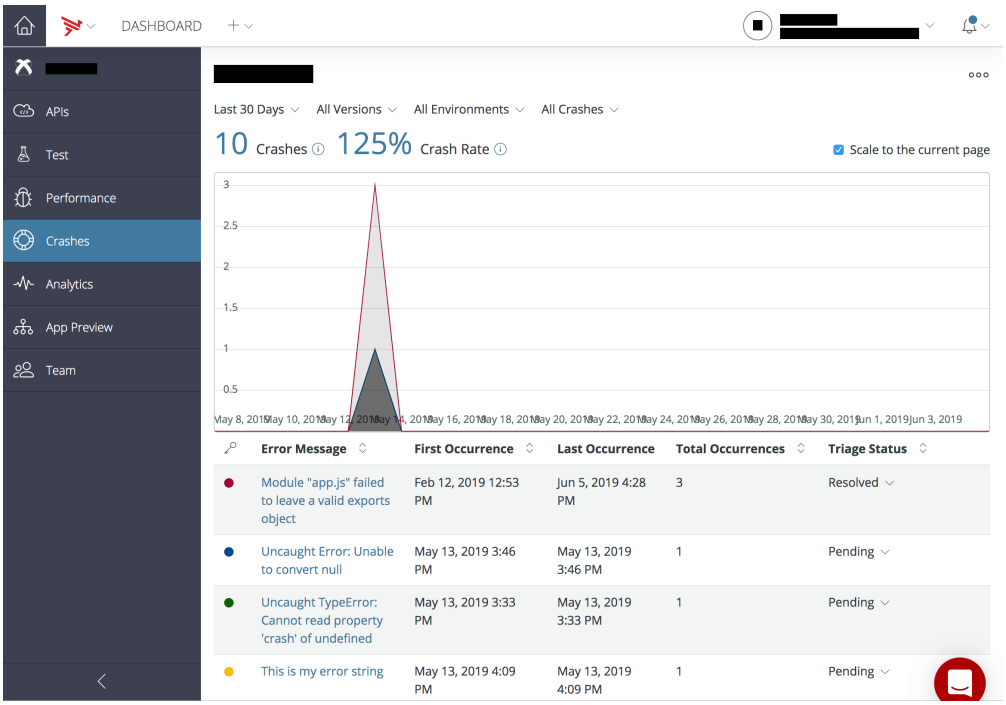
AMPLIFY Crash Analytics

Subscription Required!
This AMPLIFY Service feature requires a Pro, Enterprise, or purchased subscription.

Introduction

The AMPLIFY Crash Analytics (ACA) module provides you with real-time crash reporting and app monitoring for your Titanium powered mobile apps. This module enables the enterprise level support for monitoring critical errors and provides valuable diagnostic information to rapidly debug issues that occur in production from the Appcelerator Dashboard (Dashboard).



Requirements

- Titanium SDK 8.0.1 or above is required to use this feature
- Appcelerator Studio (Studio)

Note: ACA only supports the Android and iOS platforms.

Installation and setup

The ACA module is downloaded if you have either an Pro or Enterprise account or purchased it separately.

Setup Your Project

To activate it when creating a new project using Studio, click the **Enable Axway Appcelerator Platform Services** checkbox. Once the app has been created, you can see that analytics is set in the `tiapp.xml` file. Use the ACA module API to add breadcrumbs, user metadata, and error handling code to log events leading up to a crash. Login to the Dashboard and use the ACA dashboard to analyze crash reports.

For a previously created project, if Appcelerator Services were not previously enabled, open your `tiapp.xml` file, then click the **Enable Services** button under the *Appcelerator Service* section.

Appcelerator Studio injects code into the `tiapp.xml` file to set up the ACA module (`com.appcelerator.aca`).

If you wish to disable, remove the module from the `tiapp.xml` file and any references to the module.

To confirm that the ACA module works, try out the sample app listed in Modules.ACA.

Initialize the Module

Once `com.appcelerator.aca` has been added to the `tiapp.xml` file, the module will initialize automatically on startup.

To access module methods, you will need to require the module:

```
const aca = require('com.appcelerator.aca');
```

If you are upgrading from APM, you no longer need the initialization call:

```
const apm = require('com.appcelerator.apm');
apm.init(); // Remove this line.
```

Note: You may need to do a thorough code update to reference *aca* instead of *apm*.

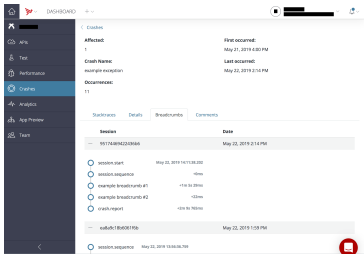
Using with Alloy

When using ACA with Alloy, you will want to maintain a reference in `Alloy.Globals`

```
Alloy.Globals.aca = require('com.appcelerator.aca');

// Throughout your project, access ACA from Alloy.Globals
const aca = Alloy.Globals.aca;
```

Create a Breadcrumb Trail



To make it easier to track the events leading up to a crash, use the `leaveBreadcrumb` method to add breadcrumbs in your code. Place breadcrumbs near events and application state changes to track problematic paths that can lead to an application crash. Append variables to your breadcrumbs to track their state. For example:

```
function alphaCB (args) {
    aca.leaveBreadcrumb('enter alphaCB:' +
JSON.stringify(args));
    //do some stuff...
    aca.leaveBreadcrumb('exit alphaCB:' + result);
    return result;
}

function betaCB (args) {
    aca.leaveBreadcrumb('enter betaCB:' +
JSON.stringify(args));
    // do some stuff...
    aca.leaveBreadcrumb('exit betaCB:' + result);
    return result;
}

switch (state) {
    aca.leaveBreadcrumb('switch:' + state);
    case x :
        alphaCB({foo: 1});
        break;
    case y :
        alphaCB({foo: 2});
        betaCB({foo: 1});
        break;
    default :
        alphaCB({foobar: 0});
        betaCB({foobar: 0});
}
```

These breadcrumbs are collected and passed to the ACA. Note: breadcrumbs will only be reported in the case of a handled exception or a crash (they won't be found in the Dashboard).

Add User Metadata

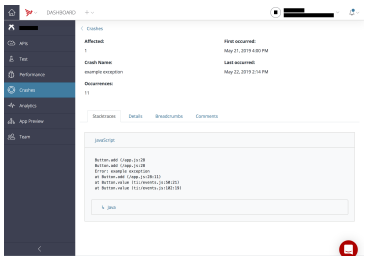
Use the `setUsername` and `setMetadata` methods to differentiate users of your application when viewing reports on the AMPLIFY Crash Analytics dashboard. For example:

```
var status = login(username);
if (status) {
    // Sets the username
    aca.setUsername(username);
    // Sets some user state metadata for tracking errors
    aca.setMetadata('lastLogin', datetime);
}

// Track the user's state
aca.setMetadata('gameLevel', 0);
// do some stuff...
aca.setMetadata('gameLevel', 1);
// do some stuff...
aca.setMetadata('gameLevel', 2);
```

By default, a username is not included. The username appears with the crash or error reports.

Log An Error



You can track handled errors in your application by passing a JavaScript Error object to the `logHandledException` method, which can help identify and analyze potential errors and hot spots. For example:

```
// If Alloy project
// const aca = Alloy.Globals.aca;

try {
    var err = new Error('FATAL ERROR: Value cannot be
null or undefined!');
    if (value === null || value === undefined) throw
err;
} catch (err) {
    aca.logHandledException(err);
}
```

Error logs are useful for tracking crashes in third-party SDKs, code that syncs data between services, or detecting bad data that is returned from a server.

Allow the User to Opt-Out

Use the `setOptOutStatus` method to allow the user NOT to send any information to the ACA. Passing `true` to this method disables sending data to ACA.

```
// Disable sending data to Crash Analytics
aca.setOptOutStatus(true);
```