USING APPLICARE › USER EXPERIENCE MONITORING

iOS Mobile App Performance Monitoring

Applicare 9.2 · Updated August 23, 2024 · **Pending technical review**

The Applicare Mobile Application Tracker provides real-time monitoring of your mobile applications, including the ability to track business transactions. This powerful service offers features such as real-time request tracking, activity path tracking, geo-tracking, error reporting, and much more.

One of the advantages of mobile app development is the ability to quickly distribute new versions and ensure that all users are using the latest app. If you haven't been monitoring your code previously, you can easily implement Applicare Tracker, push the update, and start collecting metrics within minutes.

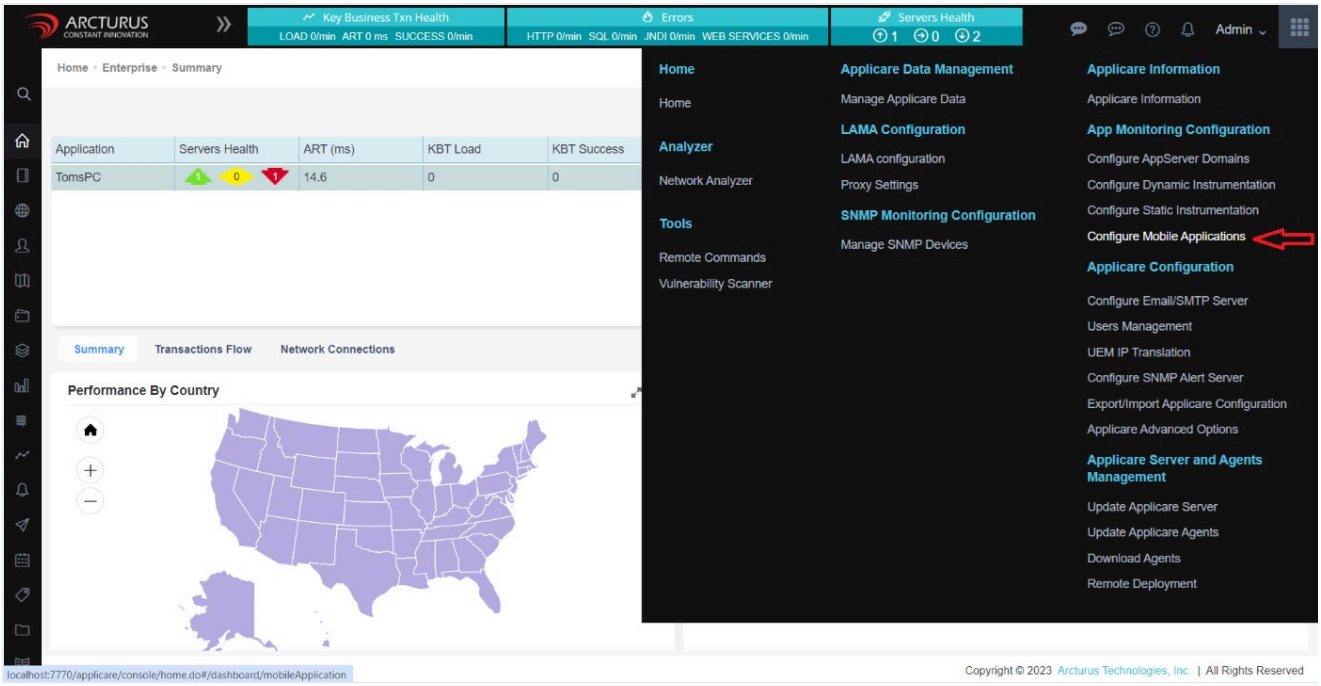
Why Use Applicare Mobile Application Tracker?

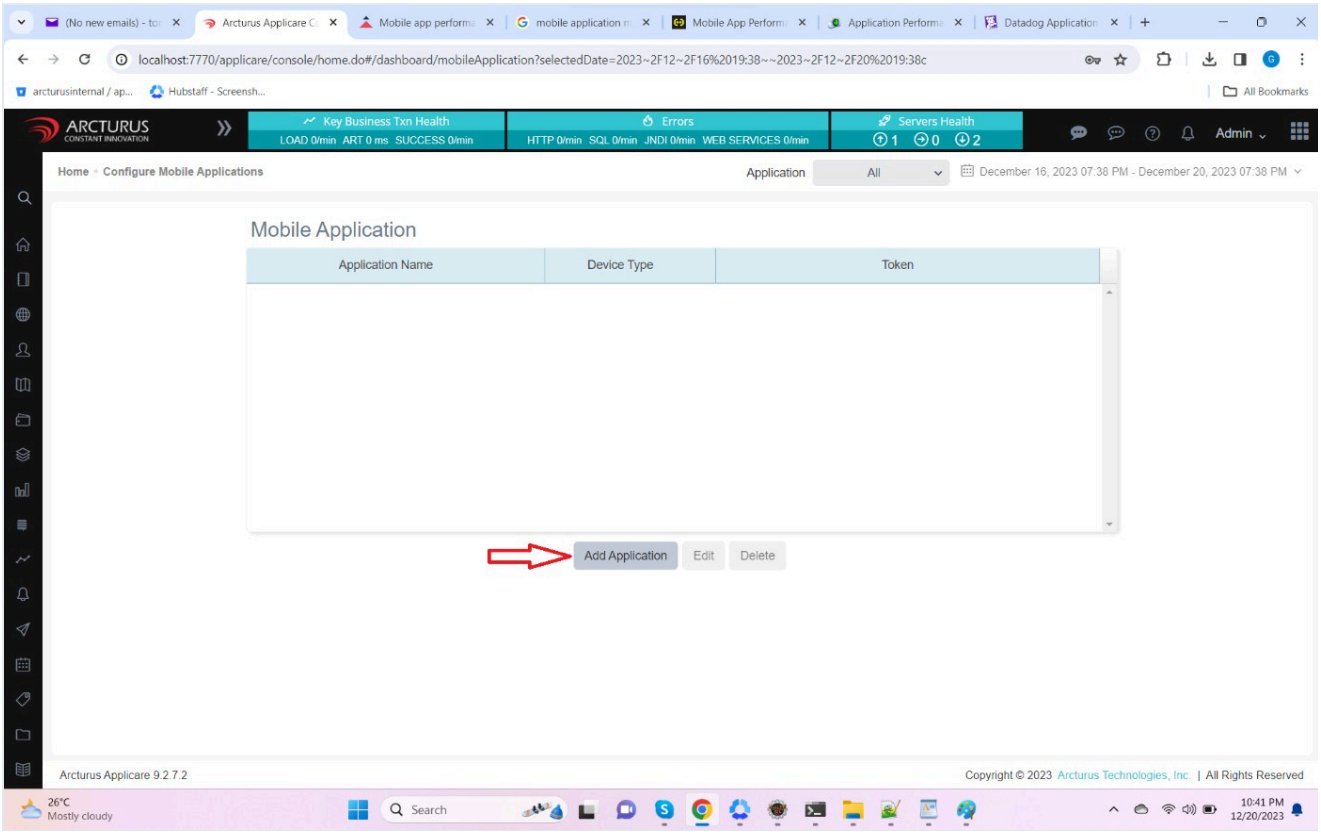
- Quick and straightforward setup process.
- Seamlessly integrates with any Android application.
- Provides insights into real user interactions.
- Capture performance metrics, set SLAs for Android method calls, and track individual transactions with performance statistics and call graphs.
- Gain insights into user characteristics.
- Visualize user locations on a map.
- Detect performance issues and server failures automatically.
- Receive regular reports to stay informed about performance issues.
- Examine detailed historical server data, including conditions before crashes.

Steps to Integrate the Applicare Mobile Tracking Agent

Register your Mobile App and Generate a Token

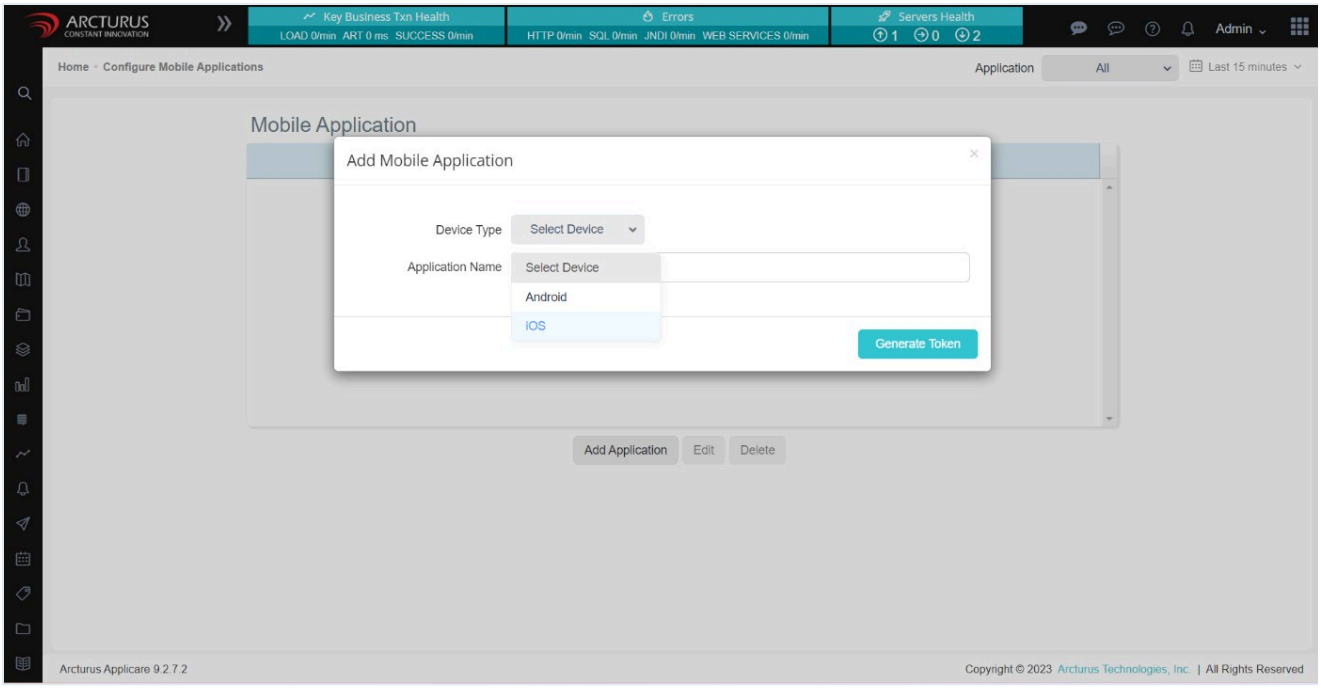
- Sign in to your Applicare Admin Console.
- Access the Menu in top right corner and navigate to the "Configure Mobile Application" screen.

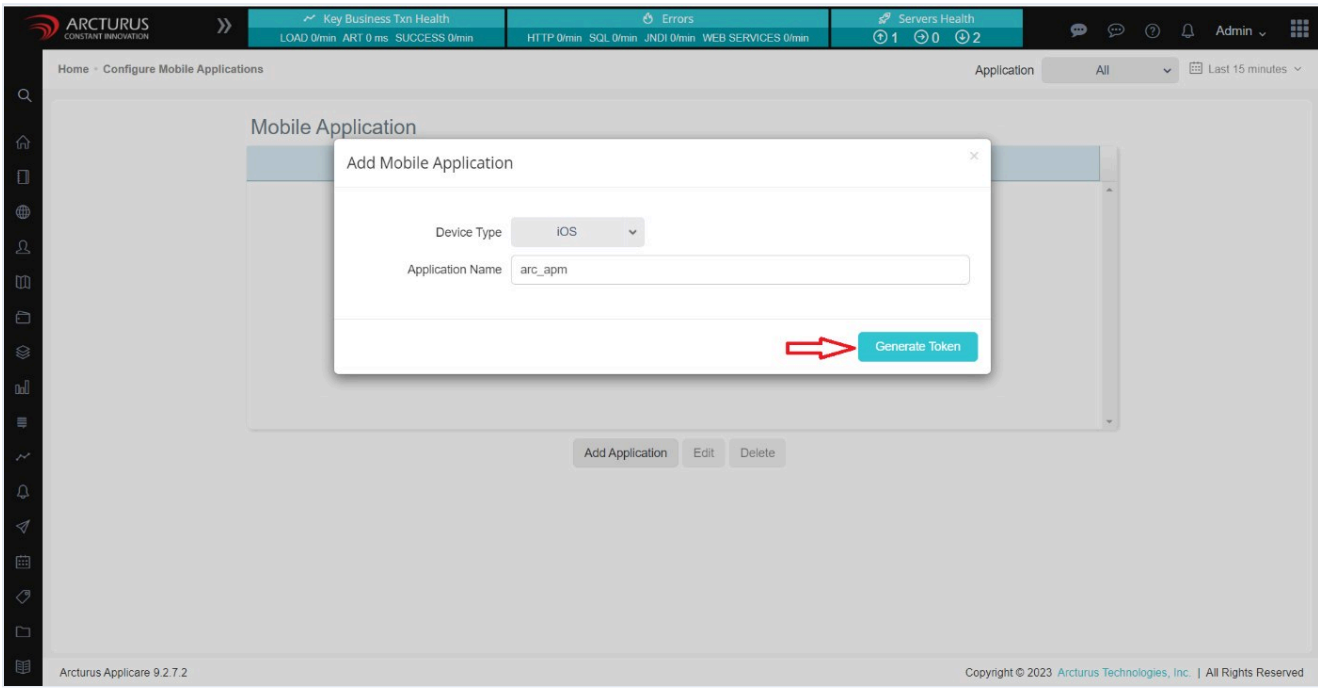


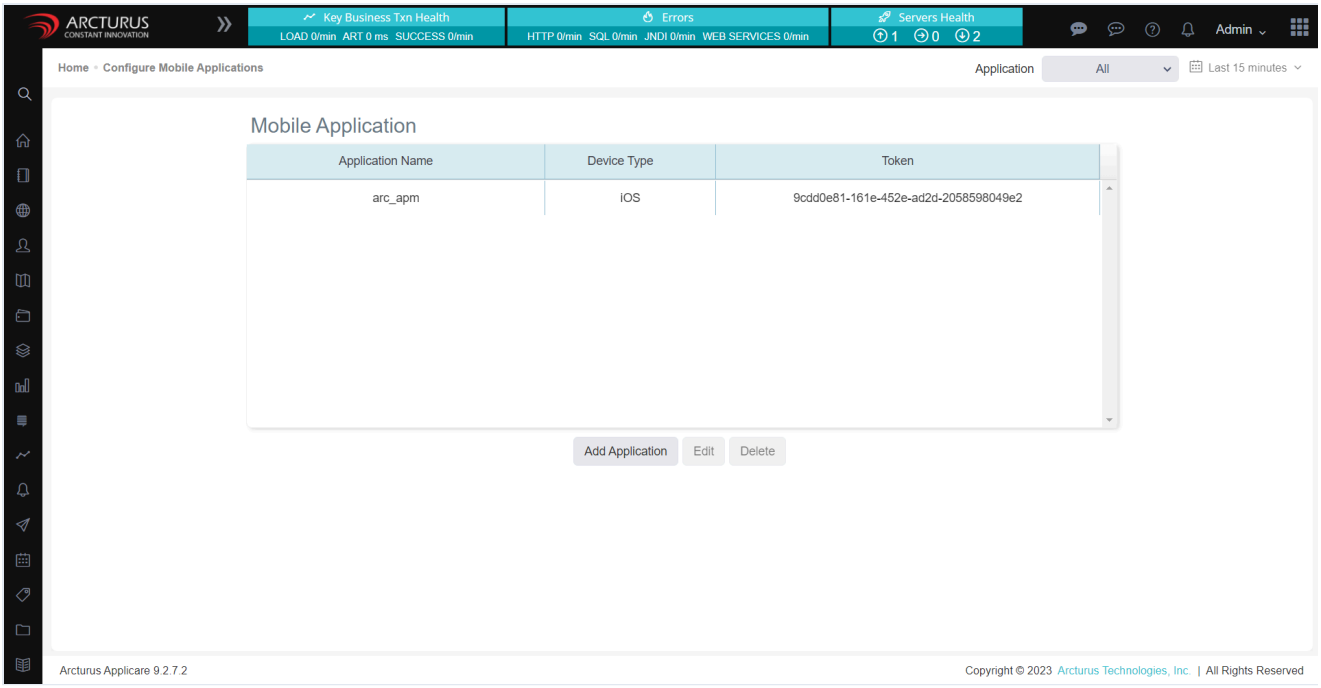


- On the "Configure Mobile Application" screen, select "iOS" from the dropdown menu. Enter your application's name and click the “Generate Token” button. A token will be generated; copy and save this token for future use.

  .


ARCTURUS
CONSTANT INNOVATION
Key Business Txn Health
LOAD 0/min ART 0 ms SUCCESS 0/min
Errors
HTTP 0/min SQL 0/min JNDI 0/min WEB SERVICES 0/min
Servers Health
1 0 2
Admin
Home • Configure Mobile Applications
Application All
Last 15 minutes
Mobile Application
Add Mobile Application
Device Type
Select Device
Application Name
Select Device
Android
iOS
Generate Token
Add Application
Edit
Delete
Arcturus Applicare 9.2.7.2
Copyright © 2023 Arcturus Technologies, Inc. | All Rights Reserved


ARCTURUS
CONSTANT INNOVATION
Key Business Txn Health
LOAD 0/min ART 0 ms SUCCESS 0/min
Errors
HTTP 0/min SQL 0/min JNDI 0/min WEB SERVICES 0/min
Servers Health
1 0 2
Admin
Home • Configure Mobile Applications
Application All
Last 15 minutes
Mobile Application
Add Mobile Application
Device Type
iOS
Application Name
arc_apm
Generate Token
Add Application
Edit
Delete
Arcturus Applicare 9.2.7.2
Copyright © 2023 Arcturus Technologies, Inc. | All Rights Reserved



Adding Applicare Agent to your app

Add the framework applicare_agent.framework file into your Objective-C or Swift project.

For Objective-C

- Inside the AppDelegate.m file import the header by adding the following line.

```
TEXT
#import <applicare_agent/ApplicationTracker.h>
```

- Now inside the didFinishLaunchingWithOptions method, add the following lines

If your app is using server add the below code.

```
TEXT

NSString *serverPort = YourServerPort;
NSString *serverIp = YourServerIp;
NSString *token = YourToken;
NSString *agentType = @"agent";
```

If your app is not using server, add the below code.

```
TEXT
NSString *serverPort = ApplicareServerPort;
NSString *serverIp = ApplicareServerIp;
NSString *token = YourToken;
NSString *agentType = @"none";
```

- Now initialize the framework by adding the following line.

```
TEXT
[ApplicationTracker initialize:serverPort arg2: serverIp arg3: token arg4: agentType];
```

- It should finally looks like below for apps with server.

```
18
19 - (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD
20     // Override point for customization after application launch.
21     NSString *serverPort = @"8080";
22     NSString *serverIp = @"192.168.2.100";
23     NSString *token = @"4db94134-bc29-4eba-b972-038d0145ce87";
24     NSString *agentType = @"agent";
25     [ApplicationTracker initialize:serverPort arg2: serverIp arg3: token arg4: agent
26     return YES;
27 }
28
```

For Swift : Here are steps to add an Objective-C frame to a Swift project.

- **Import the Framework**: Drag and drop the Objective-C framework into your Xcode Swift project.
- **Create a New Objective-C File**: Go to File -> New -> File, and select Objective-C for iOS.
- **Create Bridging Header**: Accept the prompt to create a bridging header file between Objective-C and Swift.
- **Delete the Objective-C File**: You can now delete the newly created Objective-C file, but keep the bridging header file ${YOURPROJ}-Bridging-Header.h.
- **Import the Framework**: In the bridging header file, use the standard Objective-C import syntax to include your framework.

```
TEXT
#import <applicare_agent/ApplicationTracker.h>
```

- This approach eliminates the need to perform an ***import ApplicationTracker.h*** directly in your ***AppDelegate.swift*** file. By using the bridging header, you can access the framework across all Swift files in your project.
- Inside the didFinishLaunchingWithOptions method in the AppDelegate.swift file add the below lines.

If your app is not using server add the below lines.

```
TEXT
let serverPort = ApplicareServerPort
let serverIp = ApplicareServerIp
let token = YourToken
let agentType = "none"
```

If your app is using server then add the below lines.

```
TEXT
let serverPort = YourServerPort
let serverIp = YourServerIp
let token = YourToken
let agentType = "agent"
```

- Now initialize the framework by adding the following lines.

```
TEXT
ApplicationTracker.initialize(serverPort, arg2: serverIp, arg3: token, arg4: agentType)
```

- It should finally looks like below for apps without server.

```
func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject
    // Override point for customization after application launch.
    let serverPort = "8880"
    let serverIp = "192.168.2.100"
    let token = "4db94134-bc29-4eba-b972-038d0145ce87"
    let agentType = "none"
    ApplicationTracker.initialize(serverPort, arg2: serverIp, arg3: token, arg4: agentType)
    return true;
}
```