Debugging Applications

- Edit
Collaborate with us

Edit this page on GitHub

Debug your app in Service Studio by pausing the execution at breakpoints, specific points in a module, and then running the logic step-by-step. This lets you find any issues in your logic design.

The Debugger tab shows app information like variable and runtime values. It also shows the current debugging context (current thread, event name, UI flow, screen and action, when applicable. Use the debugger commands available in the Debugger Toolbar and in the Debugger menu.
How to debug your app

To debug your app, do the following in Service Studio:

1. Click the 1-Click Publish button to save the latest changes in the module before debugging.

2. Set one or more breakpoints in the module you’re debugging.

3. Before debugging a native mobile app, choose a debugging target in the Debugger tab: Android, iOS, or Google Chrome which emulates a device. The section Mobile Debugging Scenarios includes further details about the different targets. If you’re debugging a mobile app distributed as a PWA, select Emulate using Google Chrome in Debugger > Debug Setup.

4. Start debugger by clicking the Start Debugging button in the Debugger tab or by selecting Debug in the Public Area in the Debugger menu. When you’re debugging mobile apps using the Google Chrome target, Service Studio opens a dedicated Chrome browser instance for debugging only.

Note: To also open a browser when starting a debug session in a web application, check the Open in new browser window option in the Debug Setup tab.

5. Do some tasks in the module, up to a point when the execution runs into a breakpoint and suspends.

6. When you switch to the Service Studio window, the flow or screen containing the element with the breakpoint shows on the canvas. Service Studio selects the element with the breakpoint and marks it with the debug icon.

7. The execution context shows in the Threads tab of the Debugger tab, marked with the
current thread icon, showing the current execution stack of the module elements. The Debugger tab also shows additional information you can explore.

8. After analyzing the runtime values at that execution point, you can continue running the app by:

   - Selecting one of the commands available for advancing the execution of the application logic:
     
     ![](Continue.png)
     Continue,

     ![Step Over](Step_Over.png)
     Step Over,

     ![Step Into](Step_Into.png)
     Step Into or

     ![Step Out](Step_Out.png)
     Step Out. The execution point advances according to the command you run.

   - Right-clicking an element on the canvas (or in the module tree) and selecting the Continue To Here option in the context menu. The execution continues until it reaches that element on the canvas.

In some scenarios you need to debug some functionality exposed by another module (called a producer module).

While developing Traditional Web apps you can also debug modules in your Personal Area. This lets you test your changes separately from other developer's changes in the same module.

Mobile debugging scenarios

There are different ways of debugging a mobile app that help you discover, understand, and fix issues. You can debug your mobile app in one of the following ways:

**Emulate the mobile app using the Google Chrome browser in your PC**

Use the Chrome browser in your PC to debug your mobile app if you don't need to execute native plugins, as the native plugins can't run in PC. This option is convenient to test the logic of the app. However, to check the performance or experience of the mobile app, test your app on a real device. Also consider this scenario if all the native plugins in the mobile app have action wrappers defined that return mock data when the plugin isn't available. For more information, check the Best Practices topic on creating wrapper actions for native plugins.

https://success.outsystems.com/Documentation/11/Developing_an_Application/Troubleshooting_Applications/Debugging_Ap...
Install the Mobile App on a Device

Test the mobile app directly on a device as your users would run it. It's the best place to test the performance and experience of your app. You can do it in an iOS or Android device. Generate the native app package for your app in Service Studio using the Debug (Android) or Development (iOS) build type, install it in the device, and follow the steps below according to your mobile device platform.

To test a mobile app on an iOS device:
1. On your PC, install iTunes.
2. In your device, turn the "Web Inspector" option on. For detailed instructions see Troubleshoot Debugger Connection Issues.
3. Connect your mobile device to the PC through a USB cable.
4. In your device, allow the PC to debug on the device.

To test a mobile app on an Android device:
1. In your device, turn USB debugging ON.
2. Connect your mobile device to the PC through a USB cable.
3. In your device, allow the PC to debug on the device.

For more help, check Troubleshoot Debugger Connection Issues.

If you need to troubleshoot app crashes, a plugin or check the native code of apps, debug your apps with the mobile platform's native tools, such as Android Studio for Android and Xcode for iOS. Before debugging using the native tools, you must generate a mobile package with Debug (Android) or Development (iOS) build type.

See also recommendations in Solve Common Mobile App Development Issues and Best Practices. These resources have useful tips that might save you some troubleshooting time.

Articles in this Section

- **Breakpoints**
  Breakpoints are used to suspend the execution of your application while troubleshooting and debugging issues.

- **Watches**
  Use watches to examine module elements while debugging threads in your module.

- **Threads**
  Learn more about threads in OutSystems.
• **Public and Personal Areas**
  Learn more about the two areas where modules can be executed and debugged.

• **Debugging Producer Modules**
  Check how to debug functionality exposed by a producer Module being consumed in a different module.

• **Debugger Tab Reference**
  UI reference for the Debugger Tab in Service Studio.

• **Troubleshoot Debugger Connection Issues**
  Troubleshoot issues that can arise while connecting your mobile device to your PC for debugging purposes.