**MOBILE SDKs**

A software development kit (SDK) also known as a “devkit” is a set of programming tools to help developers create apps for a specific platform – whether it’s for computer systems, video game consoles or mobile devices.

More specifically, mobile SDKs equip app developers so that they can easily build a variety of high-performing mobile apps for smartphones and tablets to be published in Google Play and App Store marketplaces.

Not every developer is equipped with full coding expertise and extensive software development skills. Even those who are, save an ample amount of time by incorporating the programming tools of a mobile SDK.

By integrating well-written, quality code with a variety of support materials, you can incorporate new features to your mobile app to make sure it’s robust and up-to-date with your competition.

In addition, a mobile SDK helps your app to be more stable and perform better. If it crashes, you have something to fall back on.

**ADOBE ANALYTICS SDKs**

You can download an ADBMobileConfig.json file that is customized with the app settings you have selected. For example, the configuration file is pre-populated with your report suite ID, tracking server, and the HTTPS, default privacy setting, and session timeout you selected on this screen.

If you download the mobile SDK, the customized ADBMobileConfig.json file is included automatically.

Configure SDK Acquisition Options

You can configure the SDK Acquisition options on the Manage App Settings page while creating a new app or editing an existing app.

Type information in the following fields under **SDK Acquisition Options**:

| **Setting** | **Description** |
| --- | --- |
| **Enable** | Create app store links that allow you to download applications directly from the Apple App Store and Google Play. The links you create help you attribute your success events to the downloads.  For more information, see [Acquisition](https://marketing.adobe.com/resources/help/en_US/mobile/acquisition_main.html#concept_542D3F9599614CB89ACF558683E9D34B). |
| **Referrer Timeout (Seconds)** | Specify the referrer timeout value. The default is 5 seconds.  This value specifies the number of seconds to wait for acquisition information before sending the First Launch hit. |
| **More Details** | Click the **More Details**link to view the app's Tracking ID. |

Packet Analyzers

Packet analyzers let you view the data sent by your implementation to Adobe Data Collection Servers.

Simliar to the DigitalPulse Debugger, a packet monitor shows what data parameters are being passed in an image request; however, packet monitors provide added functionality:

* View custom link tracking image requests
* View image requests using implementation methods other than JavaScript, such as hard-coded image requests or Appmeasurement

To view Analytics requests, filter outgoing requests using "b/ss".

In very rare cases, the debugger will report an image request although no request makes it to Adobe's Analytics processing servers. Using a packet monitor is a great way to be 100% sure that a specific image request is being fired successfully.

While Adobe does not provide an official packet monitor, there are a wide range of them on the internet. The following are some packet monitors others have found useful.

**Note:** These lists are not meant to be comprehensive, but rather information on frequently used monitors. If you have a packet monitor you successfully use and find useful, feel free to provide feedback using the Feedback button on the right side of this window.

| **Firefox** | **Internet Explorer** | **Chrome** | **Standalone Programs** |
| --- | --- | --- | --- |
| [Observe Point](http://www.observepoint.com/product#plugin) (tag viewer) | [HttpWatch](http://www.httpwatch.com/) | [Observe Point](http://www.observepoint.com/product#plugin) (tag viewer) | [Charles](http://www.charlesproxy.com/) |
| [HttpFox](https://addons.mozilla.org/en-US/firefox/addon/httpfox/) |  | [Chrome Developer Tools](http://code.google.com/chrome/devtools/docs/overview.html) | [Fiddler](http://www.fiddler2.com/fiddler2/) |
| [Tamper Data](https://addons.mozilla.org/en-us/firefox/addon/tamper-data/) |  | [Firebug Lite](https://chrome.google.com/webstore/detail/bmagokdooijbeehmkpknfglimnifench) | [Wireshark](http://www.wireshark.org/) |
| [HttpWatch](http://www.httpwatch.com/) |  |  |  |
| [Firebug](http://getfirebug.com/) |  |  |  |

**Note:** Adobe does NOT support or troubleshoot any issues you may experience with these packet monitors. Consult the packet monitor's originating site for assistance instead.

# NS\_Binding\_Aborted in Packet Monitors

The **NS\_BINDING\_ABORTED** message is often seen on custom link tracking image requests with packet sniffers that sit on top of the browser, such as Tamper Data or HTTPFox .

This error occurs because the link tracking image request is designed to let the browser proceed to the next page before waiting for a response from the Adobe data collection servers.

Adobe's response to the image request is simply a blank 1x1 transparent image, which is not relevant to the content of the page. If you see a line item in your packet monitor from Adobe, either with a **200 OK** response or an **NS\_BINDING\_ABORTED** response, the data has reached our servers. There is no need to have the page wait any longer.

Packet monitors integrated as a plug-in rarely see the full response. They tend to see the request as aborted because the full response was not received. These monitors also rarely make a distinction between whether it was the request or response that was aborted. A stand alone packet monitor typically has more detailed messages and reports the status more accurately. For example, a user may get a message inCharles saying "Client closed connection before receiving entire response." This means the data did reach our servers, just the browser moved on to the next page before the 1x1 pixel was received.

If an external packet sniffer is reporting that the data collection request is aborted, rather than the response, this is a cause for concern. Adobe Customer Care can provide help in troubleshooting.

# Getting Started with Fiddler

## **First, Install Fiddler.**

## **Next, Configure the Fiddler Server.**

The **Fiddler Server** is the machine on which Fiddler is installed. Some scenarios may require specific steps for Fiddler to receive and send web traffic. This includes:

* **Types of traffic**, like decrypting HTTPS and authenticating with channel-binding tokens
* **Operating systems**, like Windows 8 and Mac OSX
* **Network configurations**, like monitoring a remote machine, chaining to an upstream proxy, using Fiddler as a Reverse Proxy, monitoring local traffic or monitoring dial-up and VPN connections

## **Last, Configure the Client.**

The **client** is the source of the web traffic that Fiddler monitors. Some client applications, operating systems, and devices may require specific steps to send and receive traffic to and from Fiddler. This includes:

* **Browsers**, like Firefox, Opera, or IE (when sending traffic to localhost)
* **Applications**, like .NET apps, WinHTTP Apps, Java Apps, and PHP/cURL apps
* **Devices**, like Android, iOS, Windows Phone 7, and PocketPC devices