

Data collection, retention, and storage in Application Insights

06/30/2020 • 15 minutes to read •  +9

In this article

[What is Application Insights?](#)
[What data does it collect?](#)
[How can I verify what's being collected?](#)
[How long is the data kept?](#)
[Who can access the data?](#)
[Where is the data held?](#)
[How secure is my data?](#)
[Is the data encrypted?](#)
[Does the SDK create temporary local storage?](#)
[How do I send data to Application Insights using TLS 1.2?](#)
[Personal data stored in Application Insights](#)
[Data sent by Application Insights](#)
[Credits](#)

When you install [Azure Application Insights](#) SDK in your app, it sends telemetry about your app to the Cloud. Naturally, responsible developers want to know exactly what data is sent, what happens to the data, and how they can keep control of it. In particular, could sensitive data be sent, where is it stored, and how secure is it?

First, the short answer:

- The standard telemetry modules that run "out of the box" are unlikely to send sensitive data to the service. The telemetry is concerned with load, performance and usage metrics, exception reports, and other diagnostic data. The main user data visible in the diagnostic reports are URLs; but your app shouldn't in any case put sensitive data in plain text in a URL.
- You can write code that sends additional custom telemetry to help you with diagnostics and monitoring usage. (This extensibility is a great feature of Application Insights.) It would be possible, by mistake, to write this code so that it includes personal and other sensitive data. If your application works with such data, you should apply a thorough review process to all the code you write.
- While developing and testing your app, it's easy to inspect what's being sent by the SDK. The data appears in the debugging output windows of the IDE and browser.

- You can select the location when you create a new Application Insights resource. Know more about Application Insights availability per region [here](#).
- Review the collected data, as this may include data that is allowed in some circumstances but not others. A good example of this is Device Name. The device name from a server has no privacy impact and is useful, but a device name from a phone or laptop may have a privacy impact and be less useful. An SDK developed primarily to target servers, would collect device name by default, and this may need to be overwritten in both normal events and exceptions.

The rest of this article elaborates more fully on these answers. It's designed to be self-contained, so that you can show it to colleagues who aren't part of your immediate team.

What is Application Insights?

[Azure Application Insights](#) is a service provided by Microsoft that helps you improve the performance and usability of your live application. It monitors your application all the time it's running, both during testing and after you've published or deployed it. Application Insights creates charts and tables that show you, for example, what times of day you get most users, how responsive the app is, and how well it is served by any external services that it depends on. If there are crashes, failures or performance issues, you can search through the telemetry data in detail to diagnose the cause. And the service will send you emails if there are any changes in the availability and performance of your app.

In order to get this functionality, you install an Application Insights SDK in your application, which becomes part of its code. When your app is running, the SDK monitors its operation and sends telemetry to the Application Insights service. This is a cloud service hosted by [Microsoft Azure](#). (But Application Insights works for any applications, not just applications that are hosted in Azure.)

The Application Insights service stores and analyzes the telemetry. To see the analysis or search through the stored telemetry, you sign in to your Azure account and open the Application Insights resource for your application. You can also share access to the data with other members of your team, or with specified Azure subscribers.

You can have data exported from the Application Insights service, for example to a database or to external tools. You provide each tool with a special key that you obtain from the service. The key can be revoked if necessary.

Application Insights SDKs are available for a range of application types: web services hosted in your own Java EE or ASP.NET servers, or in Azure; web clients - that is, the

code running in a web page; desktop apps and services; device apps such as Windows Phone, iOS, and Android. They all send telemetry to the same service.

What data does it collect?

There are three sources of data:

- The SDK, which you integrate with your app either [in development](#) or [at run time](#). There are different SDKs for different application types. There's also an [SDK for web pages](#), which loads into the end user's browser along with the page.
 - Each SDK has a number of [modules](#), which use different techniques to collect different types of telemetry.
 - If you install the SDK in development, you can use its API to send your own telemetry, in addition to the standard modules. This custom telemetry can include any data you want to send.
- In some web servers, there are also agents that run alongside the app and send telemetry about CPU, memory, and network occupancy. For example, Azure VMs, Docker hosts, and [Java EE servers](#) can have such agents.
- [Availability tests](#) are processes run by Microsoft that send requests to your web app at regular intervals. The results are sent to the Application Insights service.

What kinds of data are collected?

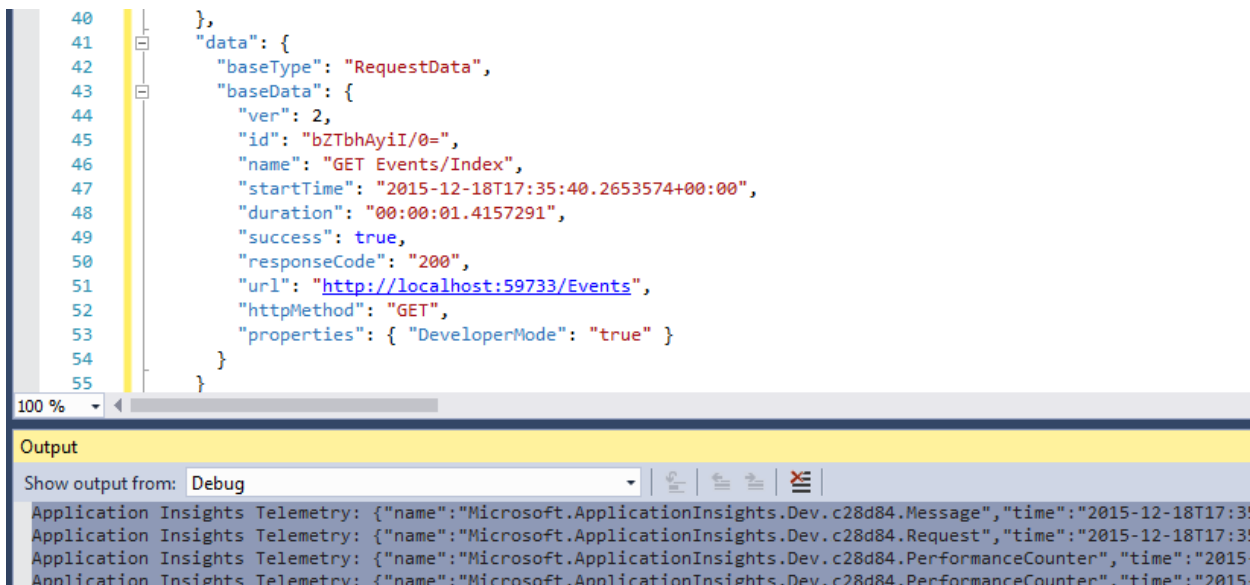
The main categories are:

- [Web server telemetry](#) - HTTP requests. Uri, time taken to process the request, response code, client IP address. *Session id*.
- [Web pages](#) - Page, user and session counts. Page load times. Exceptions. Ajax calls.
- Performance counters - Memory, CPU, IO, Network occupancy.
- Client and server context - OS, locale, device type, browser, screen resolution.
- [Exceptions](#) and crashes - **stack dumps**, build id, CPU type.
- [Dependencies](#) - calls to external services such as REST, SQL, AJAX. URI or connection string, duration, success, command.
- [Availability tests](#) - duration of test and steps, responses.
- [Trace logs](#) and [custom telemetry](#) - **anything you code into your logs or telemetry**.

[More detail.](#)

How can I verify what's being collected?

If you're developing the app using Visual Studio, run the app in debug mode (F5). The telemetry appears in the Output window. From there, you can copy it and format it as JSON for easy inspection.



```

40     },
41     "data": {
42       "baseType": "RequestData",
43       "baseData": {
44         "ver": 2,
45         "id": "bZTbhAyII/0=",
46         "name": "GET Events/Index",
47         "startTime": "2015-12-18T17:35:40.2653574+00:00",
48         "duration": "00:00:01.4157291",
49         "success": true,
50         "responseCode": "200",
51         "url": "http://localhost:59733/Events",
52         "httpMethod": "GET",
53         "properties": { "DeveloperMode": "true" }
54       }
55     }
  
```

Output

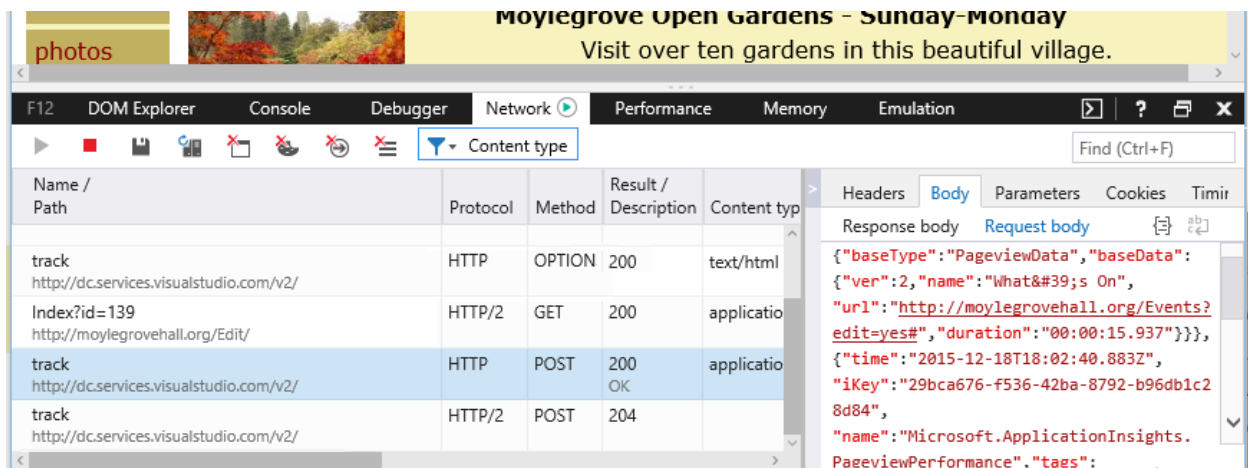
Show output from: Debug

```

Application Insights Telemetry: {"name":"Microsoft.ApplicationInsights.Dev.c28d84.Message","time":"2015-12-18T17:3
Application Insights Telemetry: {"name":"Microsoft.ApplicationInsights.Dev.c28d84.Request","time":"2015-12-18T17:3
Application Insights Telemetry: {"name":"Microsoft.ApplicationInsights.Dev.c28d84.PerformanceCounter","time":"2015
Application Insights Telemetry: {"name":"Microsoft.ApplicationInsights.Dev.c28d84.PerformanceCounter","time":"2015
  
```

There's also a more readable view in the Diagnostics window.

For web pages, open your browser's debugging window.



Moylegrove Open Gardens - Sunday-Monday
Visit over ten gardens in this beautiful village.

photos

F12 DOM Explorer Console Debugger Network Performance Memory Emulation

Content type

Name / Path	Protocol	Method	Result / Description	Content type
track http://dc.services.visualstudio.com/v2/	HTTP	OPTION	200	text/html
Index?id=139 http://moylegrovehall.org/Edit/	HTTP/2	GET	200	application
track http://dc.services.visualstudio.com/v2/	HTTP	POST	200 OK	application
track http://dc.services.visualstudio.com/v2/	HTTP/2	POST	204	

Headers Body Parameters Cookies Timers

Response body Request body

```

{"baseType":"PageviewData","baseData":
{"ver":2,"name":"What&#39;s On",
"url":"http://moylegrovehall.org/Events?
edit=yes#","duration":"00:00:15.937"}},
{"time":"2015-12-18T18:02:40.883Z",
"iKey":"29bca676-f536-42ba-8792-b96db1c2
8d84",
"name":"Microsoft.ApplicationInsights.
PageviewPerformance","tags":
  
```

Can I write code to filter the telemetry before it is sent?

This would be possible by writing a [telemetry processor plugin](#).

How long is the data kept?

Raw data points (that is, items that you can query in Analytics and inspect in Search) are kept for up to 730 days. You can [select a retention duration](#) of 30, 60, 90, 120, 180, 270, 365, 550 or 730 days. If you need to keep data longer than 730 days, you can use [Continuous Export](#) to copy it to a storage account during data ingestion.

Data kept longer than 90 days will incur addition charges. Learn more about Application Insights pricing on the [Azure Monitor pricing page](#).

Aggregated data (that is, counts, averages and other statistical data that you see in Metric Explorer) are retained at a grain of 1 minute for 90 days.

[Debug snapshots](#) are stored for 15 days. This retention policy is set on a per-application basis. If you need to increase this value, you can request an increase by opening a support case in the Azure portal.

Who can access the data?

The data is visible to you and, if you have an organization account, your team members.

It can be exported by you and your team members and could be copied to other locations and passed on to other people.

What does Microsoft do with the information my app sends to Application Insights?

Microsoft uses the data only in order to provide the service to you.

Where is the data held?

- You can select the location when you create a new Application Insights resource. Know more about Application Insights availability per region [here](#).

Does that mean my app has to be hosted in the USA, Europe, or Southeast Asia?

- No. Your application can run anywhere, either in your own on-premises hosts or in the cloud.

How secure is my data?

Application Insights is an Azure Service. Security policies are described in the [Azure Security, Privacy, and Compliance white paper](#).

The data is stored in Microsoft Azure servers. For accounts in the Azure portal, account restrictions are described in the [Azure Security, Privacy, and Compliance document](#).

Access to your data by Microsoft personnel is restricted. We access your data only with your permission and if it is necessary to support your use of Application Insights.

Data in aggregate across all our customers' applications (such as data rates and average size of traces) is used to improve Application Insights.

Could someone else's telemetry interfere with my Application Insights data?

They could send additional telemetry to your account by using the instrumentation key, which can be found in the code of your web pages. With enough additional data, your metrics would not correctly represent your app's performance and usage.

If you share code with other projects, remember to remove your instrumentation key.

Is the data encrypted?

All data is encrypted at rest and as it moves between data centers.

Is the data encrypted in transit from my application to Application Insights servers?

Yes, we use https to send data to the portal from nearly all SDKs, including web servers, devices, and HTTPS web pages.

Does the SDK create temporary local storage?

Yes, certain Telemetry Channels will persist data locally if an endpoint cannot be reached. Please review below to see which frameworks and telemetry channels are affected.

Telemetry channels that utilize local storage create temp files in the TEMP or APPDATA directories, which are restricted to the specific account running your application. This

may happen when an endpoint was temporarily unavailable or you hit the throttling limit. Once this issue is resolved, the telemetry channel will resume sending all the new and persisted data.

This persisted data is not encrypted locally. If this is a concern, review the data and restrict the collection of private data. (For more information, see [How to export and delete private data.](#))

If a customer needs to configure this directory with specific security requirements, it can be configured per framework. Please make sure that the process running your application has write access to this directory, but also make sure this directory is protected to avoid telemetry being read by unintended users.


Java

C:\Users\username\AppData\Local\Temp is used for persisting data. This location isn't configurable from the config directory and the permissions to access this folder are restricted to the specific user with required credentials. (For more information, see [implementation.](#))

.Net

By default `ServerTelemetryChannel` uses the current user's local app data folder `%localAppData%\Microsoft\ApplicationInsights` or temp folder `%TMP%`. (See [implementation](#) here.)

Via configuration file:

XML	 Copy
<pre><TelemetryChannel Type="Microsoft.ApplicationInsights.WindowsServer.TelemetryChannel.ServerTel emetryChannel, Microsoft.AI.ServerTelemetryChannel"> <StorageFolder>D:\NewTestFolder</StorageFolder> </TelemetryChannel></pre>	

Via code:

- Remove `ServerTelemetryChannel` from configuration file
- Add this snippet to your configuration:

C#	 Copy
----	--

```
ServerTelemetryChannel channel = new ServerTelemetryChannel();  
channel.StorageFolder = @"D:\NewTestFolder";  
channel.Initialize(TelemetryConfiguration.Active);  
TelemetryConfiguration.Active.TelemetryChannel = channel;
```

NetCore

By default `ServerTelemetryChannel` uses the current user's local app data folder `%localAppData%\Microsoft\ApplicationInsights` or temp folder `%TMP%`. (See [implementation](#) here.) In a Linux environment, local storage will be disabled unless a storage folder is specified.

The following code snippet shows how to set `ServerTelemetryChannel.StorageFolder` in the `ConfigureServices()` method of your `Startup.cs` class:

C#

 Copy

```
services.AddSingleton(typeof(ITelemetryChannel), new ServerTelemetryChannel  
( ) {StorageFolder = "/tmp/myfolder"});
```

(For more information, see [AspNetCore Custom Configuration](#).)

Node.js

By default `%TEMP%/appInsights-node{INSTRUMENTATION KEY}` is used for persisting data. Permissions to access this folder are restricted to the current user and Administrators. (See [implementation](#) here.)

The folder prefix `appInsights-node` can be overridden by changing the runtime value of the static variable `Sender.TEMPDIR_PREFIX` found in [Sender.ts](#).

JavaScript (browser)

[HTML5 Session Storage](#) is used to persist data. Two separate buffers are used: `AI_buffer` and `AI_sent_buffer`. Telemetry that is batched and waiting to be sent is stored in `AI_buffer`. Telemetry that was just sent is placed in `AI_sent_buffer` until the ingestion server responds that it was successfully received. When telemetry is successfully received, it's removed from all buffers. On transient failures (for example, a user loses network connectivity), telemetry remains in `AI_buffer` until it is successfully

received or the ingestion server responds that the telemetry is invalid (bad schema or too old, for example).

Telemetry buffers can be disabled by setting `enableSessionStorageBuffer` to `false`.

When session storage is turned off, a local array is instead used as persistent storage. Because the JavaScript SDK runs on a client device, the user has access to this storage location via their browser's developer tools.

OpenCensus Python

By default OpenCensus Python SDK uses the current user folder

`%username%/.opencensus/.azure/`. Permissions to access this folder are restricted to the current user and Administrators. (See [implementation](#) here.) The folder with your persisted data will be named after the Python file that generated the telemetry.

You may change the location of your storage file by passing in the `storage_path` parameter in the constructor of the exporter you are using.

Python

 Copy

```
AzureLogHandler(  
    connection_string='InstrumentationKey=00000000-0000-0000-0000-  
000000000000',  
    storage_path='<your-path-here>',  
)
```

How do I send data to Application Insights using TLS 1.2?

To insure the security of data in transit to the Application Insights endpoints, we strongly encourage customers to configure their application to use at least Transport Layer Security (TLS) 1.2. Older versions of TLS/Secure Sockets Layer (SSL) have been found to be vulnerable and while they still currently work to allow backwards compatibility, they are **not recommended**, and the industry is quickly moving to abandon support for these older protocols.

The [PCI Security Standards Council](#) has set a [deadline of June 30, 2018](#) to disable older versions of TLS/SSL and upgrade to more secure protocols. Once Azure drops legacy support, if your application/clients cannot communicate over at least TLS 1.2 you would not be able to send data to Application Insights. The approach you take to test and validate your application's TLS support will vary depending on the operating system/platform as well as the language/framework your application uses.

We do not recommend explicitly setting your application to only use TLS 1.2 unless necessary as this can break platform level security features that allow you to automatically detect and take advantage of newer more secure protocols as they become available such as TLS 1.3. We recommend performing a thorough audit of your application's code to check for hardcoding of specific TLS/SSL versions.


Platform/Language specific guidance

Platform/Language	Support	More Information
Azure App Services	Supported, configuration may be required.	Support was announced in April 2018. Read the announcement for configuration details .
Azure Function Apps	Supported, configuration may be required.	Support was announced in April 2018. Read the announcement for configuration details .
.NET	Supported, configuration varies by version.	For detailed configuration info for .NET 4.7 and earlier versions, refer to these instructions .
Status Monitor	Supported, configuration required	Status Monitor relies on OS Configuration + .NET Configuration to support TLS 1.2.
Node.js	Supported, in v10.5.0, configuration may be required.	Use the official Node.js TLS/SSL documentation for any application-specific configuration.
Java	Supported, JDK support for TLS 1.2 was added in JDK 6 update 121 and JDK 7 .	JDK 8 uses TLS 1.2 by default .
Linux	Linux distributions tend to rely on OpenSSL for TLS 1.2 support.	Check the OpenSSL Changelog to confirm your version of OpenSSL is supported.
Windows 8.0 - 10	Supported, and enabled by default.	To confirm that you are still using the default settings .
Windows Server 2012 - 2016	Supported, and enabled by default.	To confirm that you are still using the default settings

Platform/Language	Support	More Information
Windows 7 SP1 and Windows Server 2008 R2 SP1	Supported, but not enabled by default.	See the Transport Layer Security (TLS) registry settings page for details on how to enable.
Windows Server 2008 SP2	Support for TLS 1.2 requires an update.	See Update to add support for TLS 1.2 in Windows Server 2008 SP2.
Windows Vista	Not Supported.	N/A


Check what version of OpenSSL your Linux distribution is running

To check what version of OpenSSL you have installed, open the terminal and run:

terminal	 Copy
<pre>openssl version -a</pre>	

Run a test TLS 1.2 transaction on Linux

To run a preliminary test to see if your Linux system can communicate over TLS 1.2., open the terminal and run:

terminal	 Copy
<pre>openssl s_client -connect bing.com:443 -tls1_2</pre>	

Personal data stored in Application Insights

Our [Application Insights personal data article](#) discusses this issue in-depth.

Can my users turn off Application Insights?

Not directly. We don't provide a switch that your users can operate to turn off Application Insights.

However, you can implement such a feature in your application. All the SDKs include an API setting that turns off telemetry collection.

Data sent by Application Insights

The SDKs vary between platforms, and there are several components that you can install. (Refer to [Application Insights - overview](#).) Each component sends different data.

Classes of data sent in different scenarios

Your action	Data classes collected (see next table)
Add Application Insights SDK to a .NET web project	ServerContext Inferred Perf counters Requests Exceptions Session users
Install Status Monitor on IIS	Dependencies ServerContext Inferred Perf counters
Add Application Insights SDK to a Java web app	ServerContext Inferred Request Session users
Add JavaScript SDK to web page	ClientContext Inferred Page ClientPerf Ajax
Define default properties	Properties on all standard and custom events
Call TrackMetric	Numeric values Properties

Your action	Data classes collected (see next table)
Call Track*	Event name Properties
Call TrackException	Exceptions Stack dump Properties
SDK can't collect data. For example: - can't access perf counters - exception in telemetry initializer	SDK diagnostics

For [SDKs for other platforms](#), see their documents.

The classes of collected data

Collected data class	Includes (not an exhaustive list)
Properties	Any data - determined by your code
DeviceContext	Id, IP, Locale, Device model, network, network type, OEM name, screen resolution, Role Instance, Role Name, Device Type
ClientContext	OS, locale, language, network, window resolution
Session	session id
ServerContext	Machine name, locale, OS, device, user session, user context, operation
Inferred	geo location from IP address, timestamp, OS, browser
Metrics	Metric name and value
Events	Event name and value
PageViews	URL and page name or screen name
Client perf	URL/page name, browser load time
Ajax	HTTP calls from web page to server
Requests	URL, duration, response code

Collected data class	Includes (not an exhaustive list)
Dependencies	Type(SQL, HTTP, ...), connection string, or URI, sync/async, duration, success, SQL statement (with Status Monitor)
Exceptions	Type, message , call stacks, source file, line number, thread id
Crashes	Process id, parent process id, crash thread id; application patch, id, build; exception type, address, reason; obfuscated symbols and registers, binary start and end addresses, binary name and path, cpu type
Trace	Message and severity level
Perf counters	Processor time, available memory, request rate, exception rate, process private bytes, IO rate, request duration, request queue length
Availability	Web test response code, duration of each test step, test name, timestamp, success, response time, test location
SDK diagnostics	Trace message or Exception

You can [switch off some of the data by editing ApplicationInsights.config](#)

ⓘ Note

Client IP is used to infer geographic location, but by default IP data is no longer stored and all zeroes are written to the associated field. To understand more about personal data handling we recommend this [article](#). If you need to store IP address data our [IP address collection article](#) will walk you through your options.

Credits

This product includes GeoLite2 data created by MaxMind, available from <https://www.maxmind.com>.

Is this page helpful?

 Yes  No