# Xbox Live Trace Analyzer Tool Readme

V1512- May 2018

## Introduction

Welcome to the Xbox Live Trace Analyzer program. The purpose of this program is to help developers analyze trace logs of their Xbox One or Windows 10 game applications. Logs containing relevant information about service calls made to Xbox Live Services endpoints will be parsed, analyzed, and evaluated to determine if call patterns are possibly inappropriate, malformed or erroneous. Violations are reported by an outputted text file as either warnings or errors.

To collect logs, please see the section in XDK documentation titled *"Analyze Calls to Xbox Live Services."*

## Command Line Use

This command line application ingests different kind of log files, differentiated by extension type.

* CSV extension type logs are generated from the Xbox Live Services API (v2.0) traces using protocol activation or the xbTrace command line tool from the Xbox XDK package.
* JSON extension type logs are generated from the Xbox Live Services API (v1.0) traces using protocol activation or the xbTrace command line tool from the Xbox XDK package.
* SAZ extension type logs originate from Fiddler captures.

Currently, the application defaults to a data.json input data file, when no explicit -data parameter argument is specified.

In addition to the required input data file, the application expects to ingest a rules file. The rules file is what the application uses as values for the rules engine it uses to determine if calls are at risk of being in violation. The rules file will be provided by Microsoft. Tweaking of the rules.json file is currently not supported, though is planned for later versions.

Currently, the application defaults to a rules.json input data file, when no explicit -rules parameter argument is specified.

The Xbox Live Trace Analyzer will generate a report.txt file which indicates the rule that found violations, along with the details of the violations. Users can specify an alternative output directory where the application will generate the report.txt using the command line parameter -output directory with a valid directory path as an argument.

### Examples of how to use the command line application:

#### Launch with all defaults

XBLTraceAnalyzer.exe

#### Launch with specific rules file

XBLTraceAnalyzer.exe -rules myRules.json

#### Launch with specific data and rules files

XBLTraceAnalyzer.exe -data myData.json -rules myRules.json

## The Rules Engine

The current version of the Xbox Live Trace Analyzer checks against several different rule types to analyze calls and call patterns for potential violations or risks. The Rules Engine initializes itself by what is found as rules parameters inside the ingested rules.json file. Currently, manipulations of this rules file should be avoided as values have been determined as best use-case by the team at Xbox. Tweaking of the rules file is currently not supported.

Rules configurations can be specific to one of the supported [Xbox Live service endpoints](#_Supported_Endpoints), or to all of the supported endpoints. The Rules Engine will prioritize individual rules for a specific endpoint, over configurations specified for all endpoints. When the Rules Engine finds a configuration for all specified endpoints, violations are still determined per endpoint using the rules variables found in the configuration file. This means calls will be analyzed within sets mapped per endpoint, instead of one large chunk of calls.

### Rules

#### Color Coding and XR Compliance

#### As of **April 2018**, titles which exceed the call frequency to Live services at which fine grained rate limiting takes effect by 10x will not pass certification. Additional details can be found under [Xbox Live fine grained rate limiting](https://docs.microsoft.com/en-us/windows/uwp/xbox-live/using-xbox-live/best-practices/fine-grained-rate-limiting#implemented-limits).

**Red** = Indicates that a title **will not pass certification** because it is exceeding the point at which fine grained rate limiting takes effect by 10x.

**Yellow** = Indicates that the indicated service is being rate limited for your title because the title is exceeding the frequency with which calls are allowed to the service, but not exceeding the threshold that would prevent you from passing certification. This something titles should look to resolve. The anticipated title side effects of rate limiting are discussed in the [Xbox Live fine grained rate limiting](https://docs.microsoft.com/en-us/windows/uwp/xbox-live/using-xbox-live/best-practices/fine-grained-rate-limiting) article.

**Green** = Indicates that your title is making calls to Live services below the frequency at which rate limiting would take effect and is not being rate limited.

#### Call Frequency

This rule will trigger as a warning or error if the number of calls to the endpoint exceeds the MaxNumAllowedCalls within a time period of TimePeriodInMilliseconds.

#### Burst Detection

This rule will trigger as a warning or error if for all the calls within a time window (BurstDetectionWindowMs), a series of calls size BurstSizeToDetect or greater exceeds the standard deviation of time deltas between calls by a factor of two. Bursts are atypical groupings of calls determined by the time in-between calls throughout the BurstDetectionWindowMs value.

#### Repeated Calls

This rule will trigger as a warning or error when two or more calls are detected as made to the same supported endpoint within the time span given by MinAllowedRepeatIntervalMs. Calls are deemed identical when their URL and Request Body Content values are equivalent. Request Body Content hashes are compared against each other to determine equivalency.

#### Small Batch Detection Rule

This rule will trigger as a warning or error when a batched endpoint is called too few XUIDs. Or when an unbatched endpoint is called repeatedly over a short time window when it would be possible to use a batch API.

#### Throttling Rule

This rule will trigger as a warning or error when calls to an endpoint have been throttled. If any throttled calls are detected (via 429 errors), the number of throttled calls per endpoint will be displayed.

## Supported Endpoints

The Rules Engine will only process call data from traces with regards to the following Xbox Live Services endpoints:

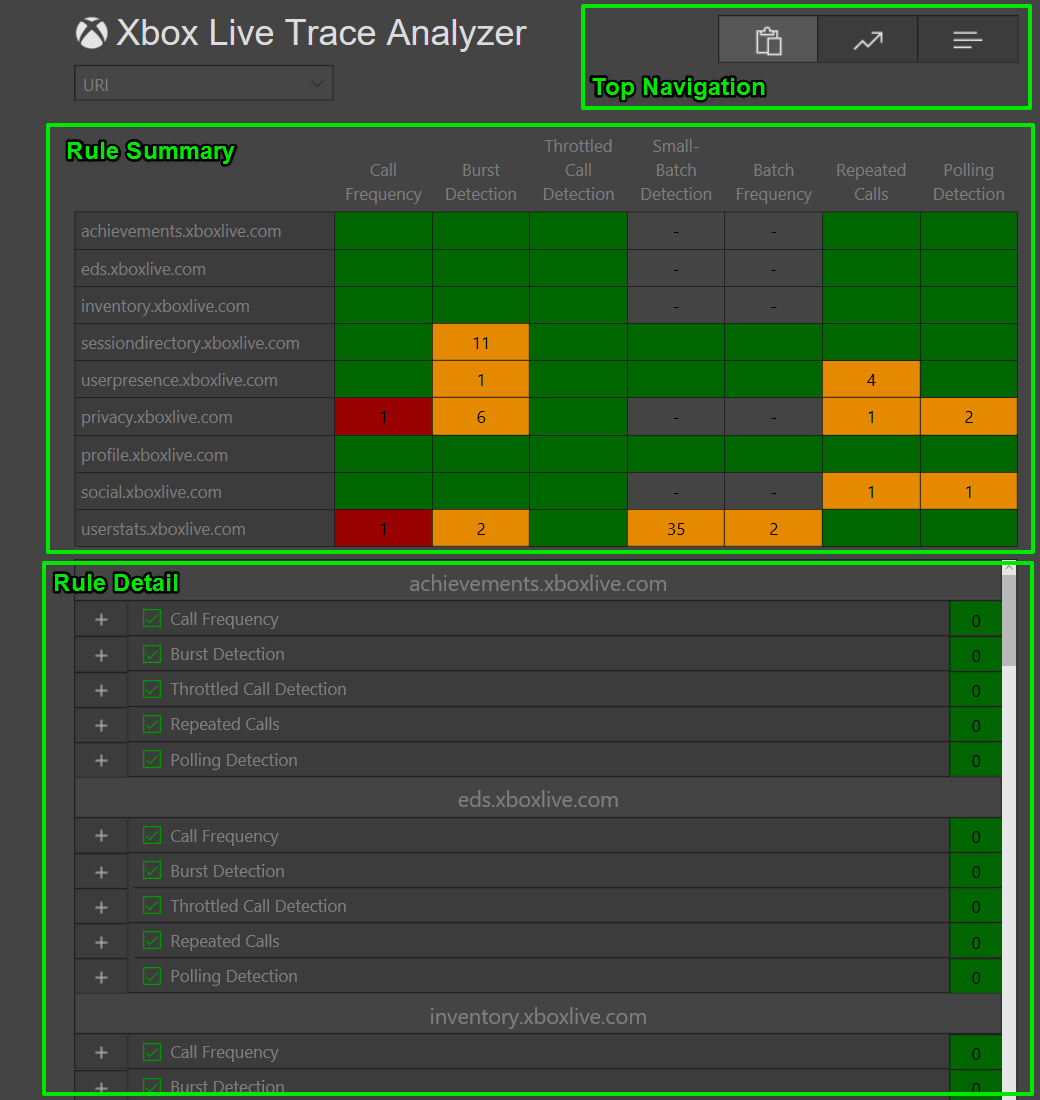
* profile.xboxlive.com
* achievements.xboxlive.com
* eplists.xboxlive.com
* leaderboards.xboxlive.com
* eds.xboxlive.com
* inventory.xboxlive.com
* momatch.xboxlive.com
* userstats.xboxlive.com
* reputation.xboxlive.com
* client-strings.xboxlive.com
* privacy.xboxlive.com
* sessiondirectory.xboxlive.com
* social.xboxlive.com
* titlestorage.xboxlive.com
* presence.xboxlive.com
* data-vef.xboxlive.com
* clubhub.xboxlive.com
* userposts.xboxlive.com
* statswrite.xboxlive.com

The wildcard indicator '\*' means that rules will process for all supported endpoints, though still segmenting calls to individual endpoints. There currently is no support to evaluate all calls as one large set or to evaluate calls made to endpoints outside of the above list.

## Report Output

The report itself is a webpage that is generated by default into the same directory you ran the Live Trace Analyzer in. You can use the -outputdir parameter to specify an output directory.

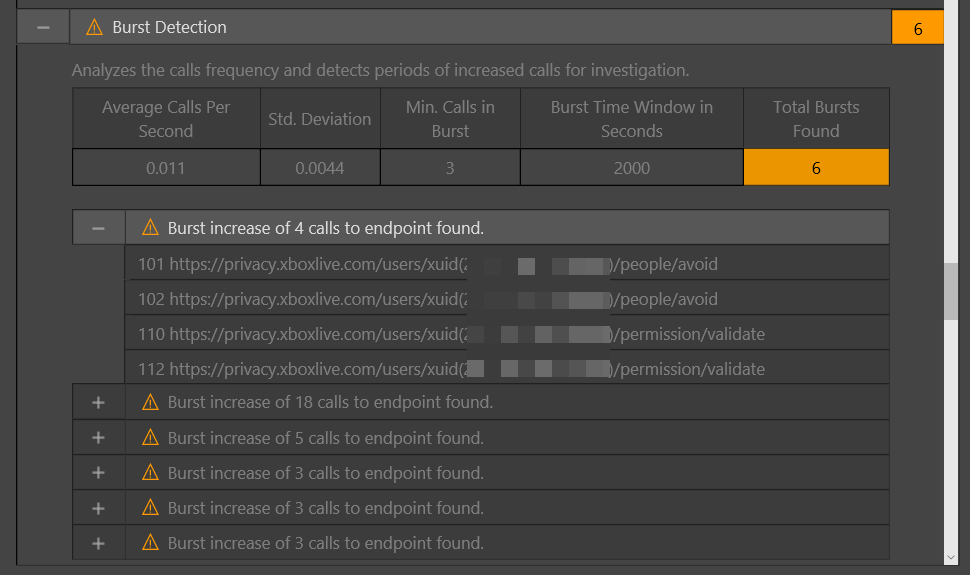
Open index.html and you will a web page resembling the following.



The matrix at the top will show a summary of the violations found for each rule configuration per endpoint. Violations will be deemed as errors is call limits are being exceeded, as this might be cause you to be rate limited. Other violations will be shown as warnings.

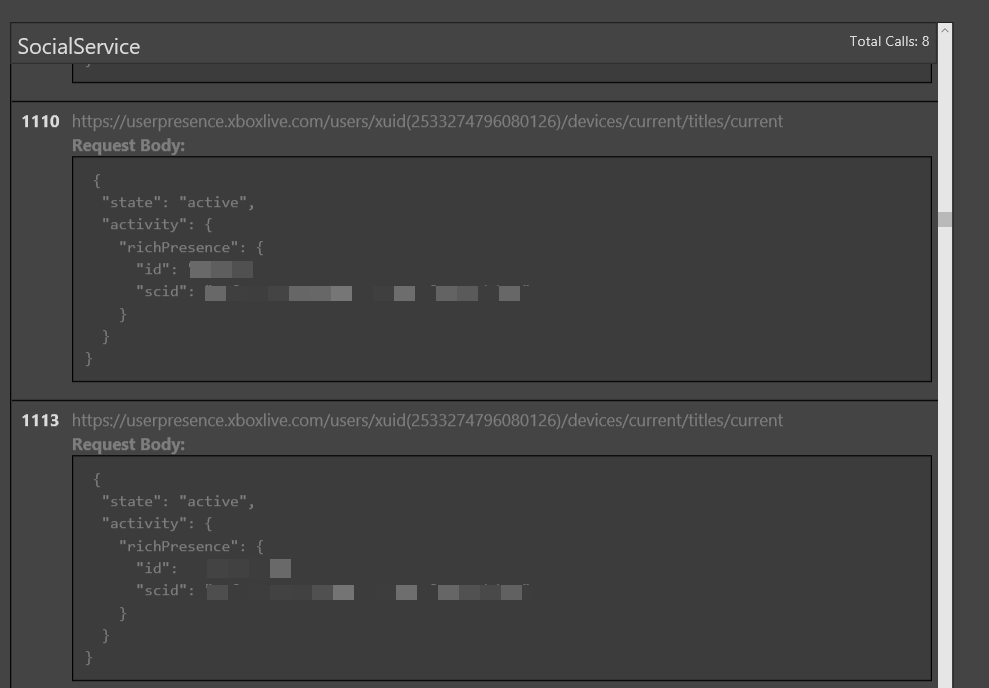
The details about each rule will be shown near the bottom. The list will be sorted by endpoint, and within each endpoint, the details on each rule can be seen. You can also click on a cell in the matrix to automatically jump to the calls corresponding to that rule violation.

You can click on a particular rule within an endpoint to expand out the rule and get some more information as in the below screenshot.



In the above example it can be seen that one endpoint has violated the burst rule, with 6 violations. The expanded burst is showing 4 calls to the privacy endpoint over the course of 2 seconds.

You can click on an individual call to get the request body.



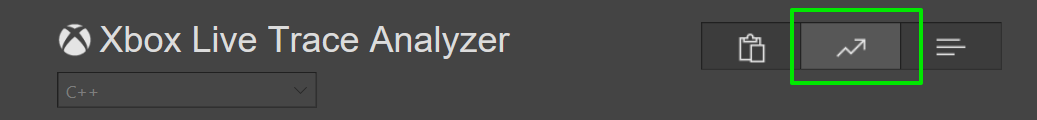
The IDs shown originate from those generated by the source log file.

* CSV IDs originate from either XSAPI 1.0 breadcrumb IDs
* JSON IDs originate from either XSAPI 1.0 breadcrumb IDs or Xbox Live Service CorrelationIDs.
* SAZ IDs originate from Fiddler frame cell data. \*PLEASE NOTE\* If Fiddler is run with "check for updates" turned on, Fiddler captures will have frame numbers off by one. We apologize for this inconvenience.

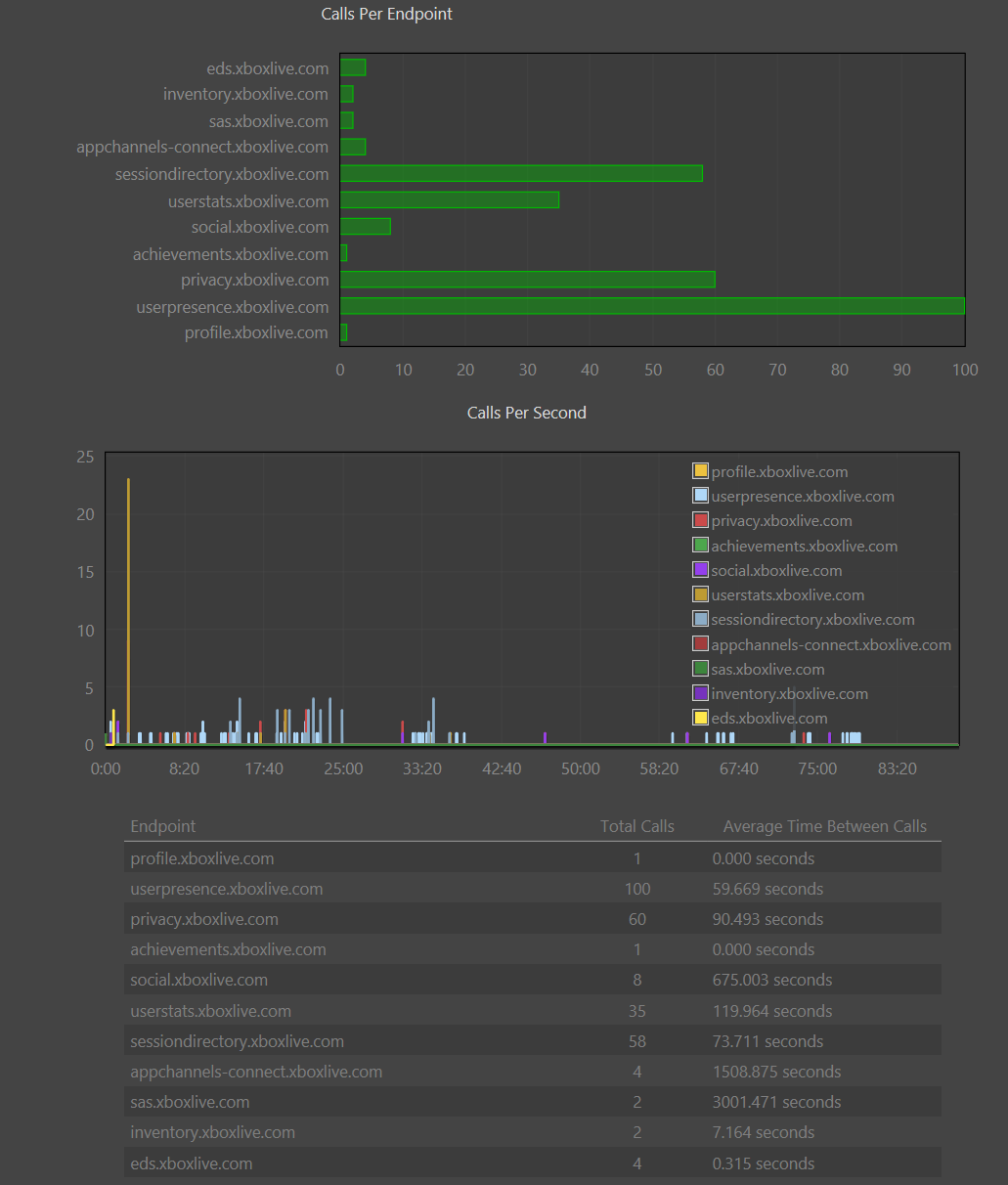
One other option on the main page is you can select whether you want to view endpoint information, or APIs. For example, the default URI option would show <https://userstats.xboxlive.com/batch?operation=read> and the C++ option would show user\_statistics\_service::get\_multiple\_user\_statistics(\_for\_multiple\_service\_configurations) which is the API that calls that URI.

## Graphs

A visual overview of your title’s traffic per endpoint, can be seen by clicking on the graphs button in the top navigation buttons.



Once you’re on the graphs page, there are a few key pieces of information



On the top, you see the average number of calls per endpoint over the entirety of the trace.

In the middle is a timeline view of the average calls per second per endpoint. You can use the scroll wheel on your mouse to zoom in, and drag the graph to see a certain subset of the timeline.

On the bottom are statistics on the average time between calls.