uberAgent 7.3.1

Contents

uberAgent 7.3.1

What is uberAgent?

uberAgent is a User Experience Monitoring (UXM) and Endpoint Security Analytics (ESA) tool that provides in-depth visibility into user experience, system performance, and security across physical and virtual endpoints. It offers detailed insights into installed application usage - including installed web browsers, and network performance. The solution is capable of providing Endpoint Security Analytics (ESA) with its highly efficient and dynamic threat detection engine and rule engine.

Why uberAgent?

uberAgent makes performance and security diagnosis a lot easier by collecting vital KPIs like latency, jitter, and packet loss –per application - including the installed browser One agent, one solution, across all physical and virtual Windows and macOS devices.

The agent is highly efficient looking at the amount of data that it reports, as well as looking at the footprint it adds on top: less than 0.5% CPU load and a minimal amount of RAM. Customers can complement the existing EDR solutions with uberAgent.

Application adoption and licensing reporting, crash analytics on process executable level, or pinpointing performance bottlenecks? You name it, uberAgent provides the metrics and comes with 70+ dashboards that visualize the collected data.

All of this sets us apart from other monitoring vendors in the EUC space, while we supplement existing observability platforms customers have in place today.

What can uberAgent do?

- Makes user experience measurable
- Helps you identify trends that otherwise would have gone unnoticed
- Simplifies troubleshooting by showing you everything you need to know in one place
- Detects threats, risky behavior, and insecure configurations
- Shows you which applications are used when –and where
- Makes Helpdesk and IT operations more effective
- Provides visibility into endpoint performance, reliability, and security
- Provides all the endpoint metrics and KPIs required by the business

Get Started

- Quickstart
- Installation

More Information

- About uberAgent
- Changelog and Release Notes
- New Release Information & Notification
- Products: UXM & ESA
- System Requirements

Quickstart

uberAgent is an innovative Windows and macOS **user experience** monitoring and **endpoint security** analytics product. You have come to the right place if you want to get it up and running in your environment.

Getting Started

1. Start by checking out the system requirements and take a look at the release notes.
2. If you have not yet done so, install Splunk or one of the alternative supported backends.
3. Next, you probably want to know how to install and deploy uberAgent.

Products

uberAgent comes in two flavors: uberAgent UXM (**user experience** monitoring) and uberAgent ESA (**endpoint security** analytics).

Components

uberAgent's main component is a **lightweight agent** that collects data on the endpoints and sends it to a backend of your choice. The other component is a **Splunk dashboard app** that visualizes what has been collected by the agent.

Configuration and Metrics

Metrics

Here is a list of all the metrics collected by uberAgent, neatly organized by category.

Configuration

uberAgent can be configured by way of **config files** (all platforms) or via **Group Policy** (Windows). Details in configuration options.

Here is a list of steps to take in order to reduce uberAgent's data volume.

Troubleshooting

uberAgent's log files help you quickly find out what is going on. They are easy to read and have a wealth of information.

If you have a question, search the documentation, ask a question in the community forum or contact support.

Upgrading

When a new version of uberAgent is released you might want to upgrade your installation. Here's how to do that.

More Information

Take a look at our blog articles or follow us on Twitter.

About uberAgent

uberAgent is an innovative Windows and macOS user experience monitoring (UXM) and endpoint security analytics (ESA) product.

uberAgent UXM highlights include detailed information about **boot and logon duration**, application unresponsiveness detection, **network reliability** drill-downs, process startup duration, **application usage** metering, browser performance, **web app** metrics, and **Citrix** insights. These varied system performance and reliability aspects are smartly brought together in the Experience Score dashboard.

uberAgent ESA excels with a sophisticated **Threat Detection Engine**, endpoint security & compliance rating, the uAQL query language, detection of **risky activity**, DNS query monitoring, hash calculation, registry monitoring, and Authenticode signature verification. uberAgent ESA comes with **Sysmon and Sigma rule converters**, a graphical rule editor, and uses a simple yet powerful query language instead of XML.

Products: UXM & ESA

uberAgent is made up of **two products**: uberAgent **UXM** and uberAgent **ESA**.

uberAgent UXM: Digital Employee Experience (DEX)

uberAgent UXM covers all aspects of digital employee experience (DEX) on any Windows or macOS endpoint. uberAgent works equally well in virtual machines as on physical PCs. It has been optimized to meet the requirements of the world's most demanding organizations, with **proven scalability** to hundreds of thousands of endpoints.

uberAgent UXM determines **application reliability** KPIs, finds issues with network connectivity, collects detailed inventory information, and shows which applications are used when and how often, to name but a small subset of its impressive features. In addition to the above, uberAgent UXM has some of the most **comprehensive Citrix monitoring** capabilities. uberAgent's browser monitoring functionality provides detailed insights into **web app usage** and performance.

More Info on uberAgent UXM

Explore the following for more information on uberAgent UXM:

- Digital employee experience (DEX) monitoring
- Citrix monitoring
- UXM documentation

uberAgent ESA: Endpoint Security Analytics

uberAgent UXM and uberAgent ESA are deeply integrated. uberAgent UXM provides rich context and metadata, while uberAgent ESA adds deep security visibility. With the combination of ESA and UXM, you need **only one agent** for digital employee experience, performance, and security. This guarantees the smallest possible footprint on the endpoint. With its flexible and open architecture, uberAgent ESA is the **perfect complement to EDR/antivirus products**.

uberAgent ESA's powerful Threat Detection Engine helps businesses **identify risky behavior**, unusual communications, suspicious executables, and common vulnerabilities. The product ships with an extensive predefined ruleset covering some of the most significant endpoint security use cases. Extending ESA's rules is easily possible and highly encouraged.

uberAgent ESA's Security & Compliance Inventory is a **testing and rating framework** that checks the attack surface of your operating systems and applications. The test results are used to calculate security scores that pinpoint configuration and security hardening weaknesses.

More Info on uberAgent ESA

Explore the following for more information on uberAgent ESA:

- Endpoint security testing & rating with Security & Compliance Inventory
- How uberAgent ESA complements EDR products
- Sysmon alternative
- ESA documentation

Components: Agents & Apps

uberAgent consists of **two main components**: the actual agent runs on the endpoints you want to be monitored, sending the data it collects to the configured backend either directly or via Splunk's Universal Forwarder. The second main component, implemented as a set of Splunk apps, provides dashboards, visualizations, searches, and reports.

The Agent

uberAgent's data collecting component is a lightweight agent that runs independently of any runtimes or frameworks. The agent's footprint is so small that it is truly unobtrusive on the monitored endpoints and does not affect user density.

Where other products rely on operating system sources such as performance counters, WMI, or similar unfiltered data, uberAgent comes with its own metrics. Instead of raw data uberAgent gives you **information that matters** (see uberAgent's list of metrics).

The agent is highly configurable: metrics can be turned on or off individually, the **data collection frequency** can be chosen freely, denylists and allowlists (supporting regular expressions) enable fine-grained control. This ensures that only data you really need is sent to the backend for indexing.

Browser Extensions

uberAgent's endpoint agent is complemented by browser extensions, which enable uberAgent to collect performance and usage metrics from web apps, too. Extensions are available for all major browsers.

The Splunk Apps

Multiple Splunk apps help process and visualize the data collected by the agent. The **indexer app** lives on Splunk indexers and mainly creates uberAgent's index and data input. **Dashboard apps** implement the user interface, providing different views into the collected data. uberAgent comes with two dashboard apps: one for **uberAgent UXM** and another for **uberAgent ESA**.

uberAgent's main apps are complemented by special-purpose apps that offer a different view of the existing uberAgent data, such as the **uberAgent Helpdesk** app.

Most dashboards are searchable and have extensive filtering capabilities to give you a fast and powerful way of isolating specific data. Time range pickers make it easy to go back to the exact time a problem occurred.

uberAgent makes full use of Splunk's advanced UI components to display a beautiful user interface that fluidly adapts to screen width and device type. As a result, it works equally well on a tablet as on a PC or Mac.

Alternative Backends

uberAgent supports several backends in addition to Splunk: Azure Monitor, Elasticsearch, and Apache Kafka. Please note that at this time support is limited to sending data from the endpoint agent to the backend. Dashboards are only available for Splunk.

System Requirements

Backend

Splunk

Operating system: all that are supported by Splunk

Splunk version: Splunk Enterprise 7.0 or newer or Splunk Cloud (details).

We only support uberAgent on Splunk software that is officially supported by Splunk (details).

Endpoints

Windows

OS Versions

- Windows 11
- Windows 10 1607
- Windows Server 2022
- Windows Server 2019
- Windows Server 2016

Please note that we only support uberAgent on Windows versions that are officially supported by Microsoft.

Processor Architectures

- x64
- x86 (where applicable)
- Arm (Windows 11 only)

Windows Agent Service Account The main component of uberAgent's endpoint agent is a system service that needs to run as the local SYSTEM account because that is required by some APIs.

macOS

Operating system: macOS Monterey or newer

Citrix Virtual Apps and Desktops, Microsoft RDS, VMware Horizon View

Remote desktop session hosts (e.g. Citrix XenApp/CVAD, VMware Horizon View, Microsoft RDS) are explicitly supported. The same applies to any kind of virtual desktop (e.g. Citrix XenDesktop/CVAD, VMware View).

Azure Virtual Desktop (AVD)

uberAgent supports Azure Virtual Desktop (AVD) environments.

IaaS (AWS EC2, Azure VMs)

uberAgent supports cloud infrastructure as a service offerings such as Amazon AWS EC2, Azure VMs, and similar in any region.

Browser Extensions

Chrome

Google Chrome 120 or newer

Edge

Microsoft Edge 120 or newer

Firefox

Mozilla Firefox 120 or newer

Internet Explorer

Microsoft Internet Explorer 11

Endpoint Agent Features

Citrix Virtual Apps and Desktops Site Monitoring

- CVAD delivery controller version 7.6 or newer

Citrix NetScaler Monitoring

- At least PowerShell version 3.0
- At least Citrix NetScaler firmware version 13.1

Citrix Cloud Monitoring

- Citrix DaaS Remote PowerShell SDK

Alternative Backends

Apache Kafka

- Confluent REST Proxy 5.4.x or newer
- Confluent Schema Registry 5.4.x or newer
- Any Apache Kafka version supported by the Confluent components above

Elasticsearch & Kibana

- At least Elasticsearch 7.12.x
- At least Kibana 7.12.x

Microsoft Azure Log Analytics

- No specific requirements

Microsoft Azure Data Explorer Cluster

- No specific requirements

Changelog and Release Notes

Version 7.3.1

Improvements

- **Agent (Windows) [I1241]:** registry monitoring now resolves the selected ControlSet key as `CurrentControlSet`.
- **Configuration [I1254]:** the agent restarts upon changes in Windows Event Log Forwarding configuration.

Bugfixes

- **Agent (macOS) [I1243]:** fixed a rare deadlock condition when the `NetworkTargetPerformanceProce` and `SessionDetail` metrics were configured in two separate timers with the same interval.
- **Agent (Windows) [I1255]:** fixed an issue where helper processes erroneously created configuration log files.

- **Citrix CVAD/DaaS (Windows) [I1262]:** fixed incompatibility with the latest Citrix Remote PowerShell SDK.
- **Citrix site monitoring (Windows) [I1272]:** improved logging for Citrix database query failures.
- **Dashboards [I1292]:** fixed the calculation of time periods that caused incorrect intervals in time charts when timestamps included decimal points.
- **Installer (Windows) [I1269]:** optimized the uberAgent installation scripts.
- **Setup (macOS) [I1279]:** in rare cases the uninstall script did not finish.
- **Setup (Windows) [I1256]:** custom installation directory was not preset during upgrade.

Known issues

- **Agent (Windows) [I1154]:** under heavy load the following message may be logged: `CheckEventRecord,Events were lost. This may affect uberAgent's per-process disk, network, or UI-responsiveness metrics`.
- **Boot monitoring [I1052]:** on Windows 11, no information can be retrieved if there is no active session within the data collection period.
- **Browsers [I1085]:** on systems with many user sessions the URL of the foreground tab might not match the browser's window title.
- **Browsers/Firefox add-on [I626]:** if the option `privacy.resistFingerprinting` is set to true, browser metrics are not available due to invalid data being sent from Firefox.
- **Browsers/IE add-on (Windows):** metrics are collected incompletely for the configured start page.
- **Browsers/IE add-on (Windows):** metrics are not collected on page reload.
- **Browsers/IE add-on (Windows):** monitoring does not work if IE is published from Citrix Virtual Apps. It does work from Citrix Virtual Desktops, however.
- **Citrix CVAD/DaaS (Windows):** data collection issue if the Citrix Remote Powershell SDK (required for Citrix Cloud monitoring) is installed on a CVAD controller.
- **Citrix CVAD/DaaS (Windows):** when running the Citrix VDA on a Citrix delivery controller, some per-machine information is missing.
- **Experience Score [I377]:** scheduled searches generate three warnings in Splunk's `_internal` index every 30 minutes. The messages look like the following: `DateParserVerbose - Failed to parse timestamp in first MAX_TIMESTAMP_LOOKAHEAD (128) characters of event.`. However, there is no impact on uberAgent's functionality.
- **GPU (Windows) [I33]:** values for the fields `ComputeUsagePercentAllEngines`, `ComputeUsagePercentEngine0` and similar can be higher than 100 with Intel Iris GPUs on Windows Server 2016 1607.
- **Kafka [I291]:** in rare cases, sending data to Kafka results in a `SEC_E_BUFFER_TOO_SMALL` error message in the logfile. This should have no effect; the transmission is repeated and suc-

ceeds on the second try.

- **NetScaler:** in very rare cases, the content of the Virtual Server Performance field `vServerName` contains spaces in wrong places.
- **Update inventory (Windows):** not all installed Windows updates may be reported due to API limitations.
- **User input delay (Windows) [I983]:** determining this metric may trigger a handle leak in uberAgent caused by Windows. This was fixed by Microsoft in most OS versions, but still happens on Windows Server 2022 22H2.
- **Volume inventory (macOS):** the encryption status of mounted read-only APFS snapshots may not be reported due to API limitations. This includes the root directory volume in a default installation of macOS.

New Release Information & Notifications

There are many ways to get notified about new uberAgent releases.

Citrix Blog

- Citrix uberAgent blog URL

Citrix Downloads

- Citrix Observability feed URL

Social Media

- Citrix Twitter
- Citrix LinkedIn
- Citrix Facebook
- Citrix Instagram

Email

You can subscribe to Citrix's newsletter on citrix.com/blogs.

Planning

The information in this section helps you plan uberAgent deployments.

Architecture Options

Backend Placement

Applies to: Splunk and alternative backends.

The backend service required by uberAgent can be installed or consumed in any setup that is supported by the backend product vendor. This includes:

- On-premises (physical or virtual machine)
- Backend vendor cloud offering such as Splunk Cloud
- Public cloud IaaS (virtual machine in a cloud such as AWS EC2, Azure VMs)

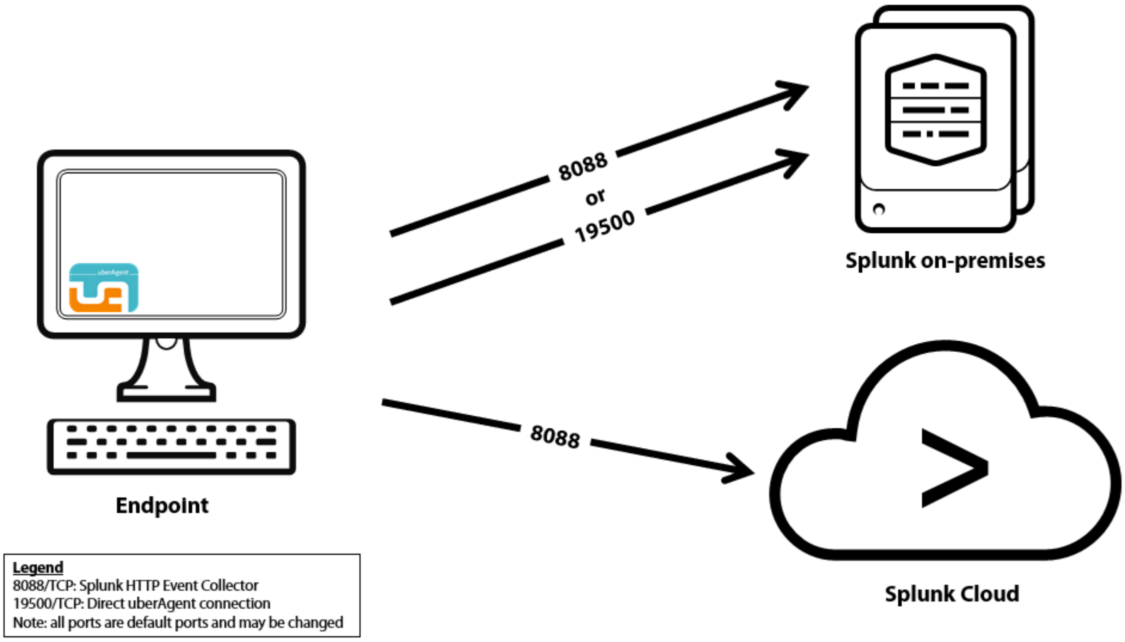
Note: vast limits does not operate backend servers for data storage and visualization. Backend servers such as Splunk are always operated by vast limits'customers or partners.

Endpoint to Backend Communication

Recommended: Direct

Applies to: Splunk and alternative backends.

In this recommended configuration uberAgent talks directly to the backend servers. This has the advantage that the overall footprint on the monitored endpoints is smaller compared to architecture options that require Splunk Universal Forwarder.


8088
or
19500
Splunk on-premises
8088
Splunk Cloud
Endpoint
Legend
8088/TCP: Splunk HTTP Event Collector
19500/TCP: Direct uberAgent connection
Note: all ports are default ports and may be changed

Configuration highlights

- **Communications:** uberAgent sends data directly from the endpoint to the backend servers
- **Splunk backends:** on the indexers, either a TCP port is opened or HTTP Event Collector is configured
- **Alternative backends:** uberAgent makes use of the backend's native REST API

Pros

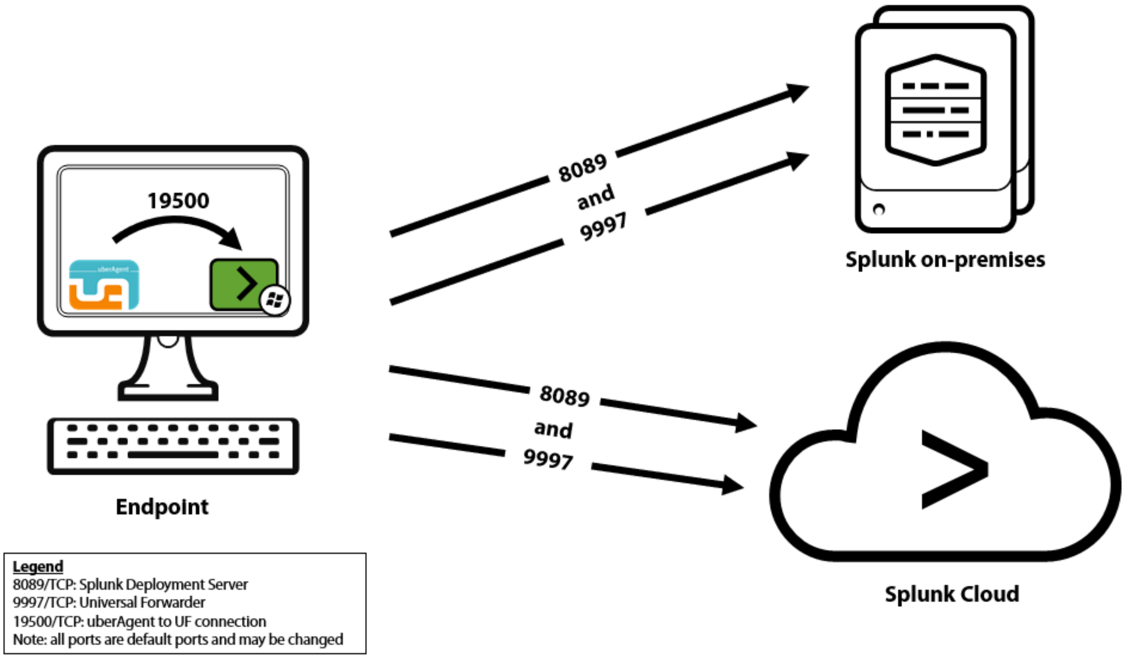
- Smallest footprint
- Data can be transferred encrypted via HTTPS and authenticated with a REST token

Alternative: Via Splunk Universal Forwarder

Applies to: Splunk backends only.

> **Note**
>
> Please note that with this configuration option, data is transmitted unencrypted over TCP from uberAgent to Splunks Universal Forwarder on the local machine. Therefore, we recommend that you use the direct connection as described before, which enables encrypted and authenticated data transmission.

Similar to Standalone mode, but uberAgent sends the data it collects to a locally installed Splunk Universal Forwarder. If you have deployed Universal Forwarder on your monitored endpoints anyway you might want the forwarder to handle all Splunk communications.


19500
Endpoint
8089
and
9997
Splunk on-premises
8089
and
9997
Splunk Cloud
Legend
8089/TCP: Splunk Deployment Server
9997/TCP: Universal Forwarder
19500/TCP: uberAgent to UF connection
Note: all ports are default ports and may be changed

Configuration highlights

- **Communications:** uberAgent sends data to the local forwarder's TCP port which in turn sends data to the Splunk indexers
- **Splunk Universal Forwarder:** a TCP port is opened on each endpoint (for local access only)

Pros

- All Splunk communications are handled by Splunk components
- Additional data can be collected via Universal Forwarder (log files, Windows event logs, scripts)
- Collected data can optionally be persisted to disk before sending off to Splunk

Cons

- Larger footprint than the recommended architecture
- On the local machine, data is sent unencrypted and unauthenticated over TCP

Configuration Options

uberAgent can be configured by means of **configuration files** (all platforms) or via **Group Policy** (Windows). Both variants allow for centralized management and automatic deployment. The **recommended** way to configure uberAgent is via Central Config File Management (CCFM).

Default Configuration

Starting with uberAgent 7.3.1, the agent installer does not store the template configuration files on disk any more. You can download the latest template config files for your installed uberAgent version on GitHub. It is advisable to regularly check the GitHub repository for relevant configuration updates, specifically for uberAgent ESA's threat detections and security inventory checks.

Data Volume Optimized Configuration

A configuration optimized for data volume is available as an alternative to the default configuration.

How to Configure uberAgent

Configuration via Central Config File Management (CCFM)

To configure uberAgent via centrally managed configuration files, please follow these steps (recommended).

Configuration via Local Config Files

To configure uberAgent via configuration files local to the endpoint, please follow these steps.

Configuration via Group Policy

To configure uberAgent via Group Policy, please follow these steps.

Precedence

Windows

1. **CCFM**
2. **Group Policy**

3. **ProgramData:** `%ProgramData%\vast limits\uberAgent\Configuration\uberAgent.conf`

The first configuration found is used exclusively. In other words: CCFM overrides Group Policy, which overrides `%ProgramData%`.

macOS

1. **CCFM**
2. `/Library/Application Support/uberAgent/`

Antivirus Exclusions

It is recommended that specific folders are excluded from scanning by your antivirus product to ensure smooth operation of your uberAgent deployment. Include the following items to reduce the risk of false positives.

Configuration Files & Scripts

Platform	Path	Filetype
Windows	`%ProgramData%\vast limits\uberAgent\Configuration\*`	.conf .ps1
Windows	`%ProgramData%\vast limits\uberAgent\Config cache\*`	.conf .ps1
Windows	`%ProgramFiles%\vast limits\uberAgent\Config Templates\*`	.conf .ps1
macOS	`/Library/Application Support/uberAgent/`	.conf
macOS	`/Library/Application Support/uberAgent/Security inventory/*`	.zsh

Platform	Path	Filetype
macOS	`/Library/Application Support/uberAgent/Config Cache/*`	.conf .zsh
macOS	`/Library/Application Support/uberAgent/Config Templates/*`	.conf .zsh

Persistent Output Queue

Exclude the location where the POQ stores its files. The items below assume the default location. If you choose to use a different path, adjust accordingly.

Platform	Path	Filetype
Windows	`%ProgramData%\vast limits\uberAgent\Output Queue\*`	.db .db-shm .db-wal
macOS	`/Library/Application Support/uberAgent/Output Queue/*`	.db .db-shm .db-wal

Splunk Data Volume Calculation

When uberAgent is used with Splunk as a backend, a Splunk license is required. Splunk is licensed by daily indexed data volume, i.e., you pay for the total amount of data you send to Splunk per day. How long that data is stored does not matter, only how much new data you add. uberAgent is one of the potentially many data sources that put data into Splunk, contributing to the total data volume.

Estimate or Measure

Customers have a vested interest in knowing how much data each Splunk add-on generates, so they can estimate costs before they buy. In the case of uberAgent, the data volume per host depends greatly on the environment, the types of applications used, the desktop configuration, background processes, the type of browser used and many other variables. For that reason, it is not possible to calculate the data volume with any reasonable accuracy without doing an actual proof of concept implementation (see below). However, if you just want some figures for a **very rough first calculation**, use the following values for typical clients and servers:

uberAgent UXM data volume

- Typical data volume per **single-user** client and working day: **25 MB**
- Typical data volume per **multi-user** (Citrix CVAD/RDS) server and working day: **90 MB**

uberAgent ESA data volume

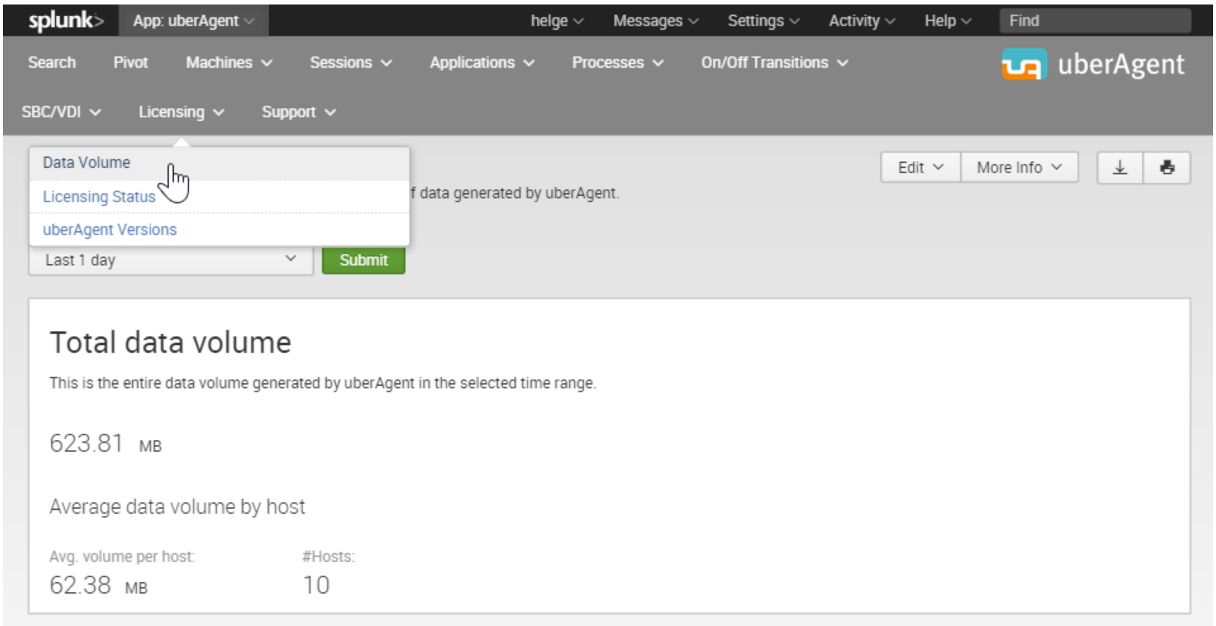
Using uberAgent ESA in addition to uberAgent UXM will approximately double the data volume per client/server and working day.

Accurate data volume numbers

To get accurate numbers install uberAgent and go to the *Data Volume* dashboard (see below). You can significantly reduce the data volume through an optimized configuration.

Data Volume Dashboard

If you already have uberAgent installed, you can simply look up the generated data volume by going to the *Data Volume* dashboard:



f data generated by uberAgent.

Total data volume

This is the entire data volume generated by uberAgent in the selected time range.

623.81 MB

Average data volume by host

Avg. volume per host: 62.38 MB

#Hosts: 10

> **Note**
>
> Make sure you have configured Splunk correctly or the data volume dashboard may not be able to display values for all metrics.
>
> The dashboard uses macros of the UXM and ESA apps. If you only have the UXM app installed, you see the error message **Error in 'Searchparser': the search specifies a macro 'uberAgentESA_score_index'that cannot be found…**. Installing the ESA app solves that issue.

Reducing the Data Volume

Once you have your first installation set up you might want to fine-tune and possibly reduce the data volume. Luckily that is easily possible. Here is how to reduce uberAgent's data volume.

Splunk Sizing Resources and Recommendations

Sizing Splunk is not always trivial, especially if it is used for other use cases in addition to uberAgent. We generally recommend working with one of our partners.

That being said, this page lists some basic recommendations as well as resources that should help with sizing Splunk. Before we start, please keep in mind that the only generic answer any good consultant will give is: "it depends". Because it does.

Splunk Sizing Considerations

Hardware Resources: CPU and Disk

Splunk needs CPU and disk resources, RAM not so much (compared to some other workloads). Make sure you have enough disk space for the planned retention time as well as a disk subsystem that delivers good IOPS numbers.

Accelerated Data Model

uberAgent's Splunk app makes use of an accelerated data model which speeds up searches by about 50-100x. The data model's high-performance analytics store (HPAS) is located on the indexers. Generating the HPAS incurs some additional indexer CPU load and requires additional disk storage.

Heavy Forwarders

Splunk Heavy Forwarders (HFs) can often be a useful third tier, logically situated between the uberAgent endpoints and the Splunk indexers. If you are deploying uberAgent to tens of thousands of endpoints, keep in mind that high numbers of simultaneous network connections may place a significant load on the HFs. Monitor heavy forwarder performance and be prepared to scale out.

Splunk Sizing Recommendations

1. Always **start with a PoC** and closely monitor Splunk performance during that phase.
2. Measure uberAgent's data volume, keeping in mind that optimization is often possible.
3. Due to the accelerated data model, uberAgent's Splunk load profile is somewhat similar to Splunk's Enterprise Security (ES) app. When looking at sizing guides, base your calculations on the ES use case.

Splunk Sizing Resources

- Splunk's Capacity Planning Manual and its chapter on reference hardware and its summary of performance recommendations
- The deployment planning chapter from Splunk's Enterprise Security installation and upgrade manual
- Splunk's unofficial storage sizing calculator
- Hurricane Labs'Splunking Responsibly blog series. Part 1: considerations for managing limited system resources and Part 2: sizing your storage
- Aplura's Splunk best practices

uberAgent Feature Support on macOS

The long-term goal for uberAgent on macOS is to achieve feature-parity with its Windows counterpart. As long as this is not yet the case, this article helps you to see at a glance **which metrics are being collected and which dashboards contain macOS-specific information**.

More information about individual fields is available in the metrics documentation.

User Experience Monitoring

Sourcetypes

- uberAgent:Application:AppNameIdMapping

- uberAgent:Application:ApplicationInventory
- uberAgent:Application:BrowserWebRequests2
- uberAgent:Application:Errors
- uberAgent:Application:NetworkConnectFailure
- uberAgent:Config:ConfigInfo
- uberAgent:License:LicenseInfo
- uberAgent:Process:NetworkTargetPerformance
- uberAgent:Process:ProcessDetail
- uberAgent:Process:ProcessStartup
- uberAgent:Process:ProcessStatistics
- uberAgent:Session:SessionDetail
- uberAgent:System:DiskInventory
- uberAgent:System:MachineInventory
- uberAgent:System:MonitorInventory
- uberAgent:System:NetworkConfigInformation
- uberAgent:System:SystemPerformanceSummary2
- uberAgent:System:VolumeInventory

Dashboards

Experience Score

- Experience Score Overview
- Machine Scores
- Session Scores
- Application Scores

Machines

- Machine Performance
- Machine Network Communication
- Machine Network Configuration
- Machine Network Issues
- Machine Uptime
- Machine Inventory
- Machine Storage

Sessions

- User Session Overview

- User Sessions
- Session 0

Applications

- Application Performance
- Application Errors
- Application Network Communication
- Application Network Issues
- Browser Web App Performance
- Browser Web App Usage
- Application Inventory
- Application Usage

Processes

- Process Performance
- Process Network Communication
- Process Network Issues

Licensing

- Data Volume
- Licensing Status
- uberAgent Versions

Endpoint Security Analytics

Sourcetypes

- uberAgentESA:ActivityMonitoring:ProcessTagging
- uberAgentESA:Process:DnsQuery

Dashboards

Processes

- Threat Detection Events
- DNS Exfiltration & Tunneling
- Process Tree

Installation

Installing uberAgent is quick and easy. This section shows you how to install the two main uberAgent components: the actual agent that runs on the machines you want to monitor as well as the backend that stores and visualizes the collected data.

Backend

Supported Backends

While uberAgent has been designed to use **Splunk** as a backend from day one, it also works well with **Elasticsearch & Kibana**, **Apache Kafka**, **Microsoft Azure Monitor**, and **Microsoft ADX**. Installing instructions for all backends are available in this section.

Installing Splunk

This page explains how to install a Splunk server for uberAgent.

Prerequisites

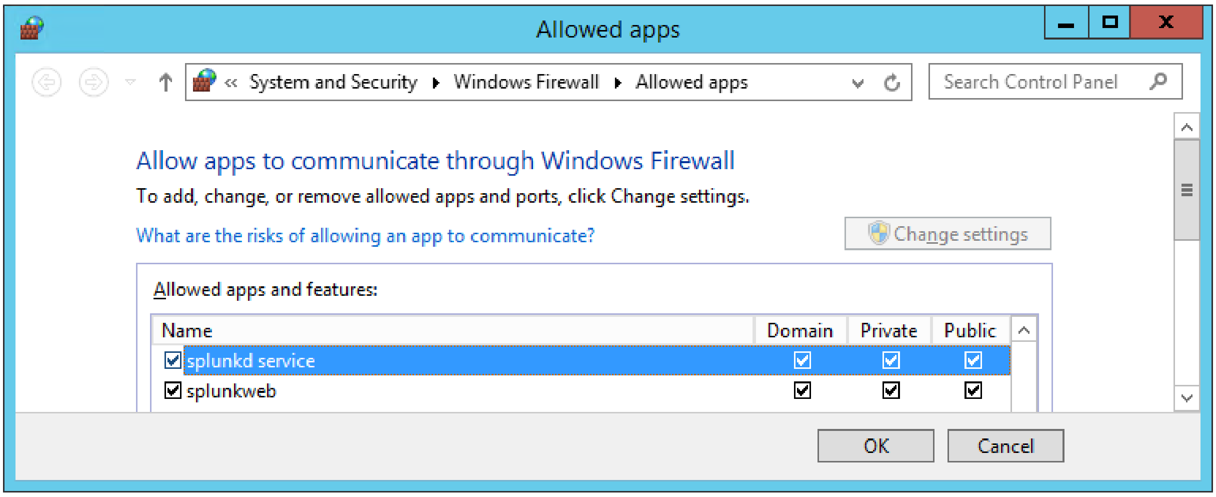
- Download Splunk
- Please see the KB article supported Splunk versions

Setup

Run through Splunk's setup on the designated Splunk server (in this simple tutorial we assume that you only have a single, Windows-based Splunk server). Choose *Local system user* when asked for a Splunk user.

Firewall

Once Splunk is installed: if you have a firewall enabled, make sure that communication is allowed for `splunkd.exe` and `splunkweb.exe` (both normally located in `C:\Program Files\Splunk\bin`). For Windows Firewall the recommended configuration looks like this:



Allow apps to communicate through Windows Firewall

To add, change, or remove allowed apps and ports, click Change settings.

What are the risks of allowing an app to communicate?

Allowed apps and features:

Name	Domain	Private	Public
☑ splunkd service	☑	☑	☑
☑ splunkweb	☑	☑	☑

Log On

Log on to the Splunk console by navigating to `http://servername:8000` in your browser.

License

If you plan to use Splunk Enterprise and already have a license, install it through **Settings** > **Licensing**. If you do not have a license yet: Splunk runs in Enterprise mode with an allowed daily data volume of 500 MB for 60 days. Then it switches to the free version.

Sending to Splunk's HTTP Event Collector

If you plan to have the endpoint agent send the collected data to Splunk's HTTP Event Collector follow the steps in this article.

Install uberAgent

Read on about how to install uberAgent.

Configuring Splunk's HTTP Event Collector

What is HTTP Event Collector

HTTP Event Collector (HEC) is a high-performance REST API data input. It accepts plain text or JSON data sent via HTTP or HTTPS.

Clients must authenticate with a token in order to be able to send data to a HEC input. Multiple tokens can be generated per HEC input if required.

When to Use HTTP Event Collector

HTTP Event Collector (HEC) is the only way to send uberAgent data to Splunk Cloud. But it is useful even with on-premises Splunk Enterprise. HEC forces clients to authenticate before being allowed to send and it can use HTTPS as data transport, which qualifies it for sending data over the internet.

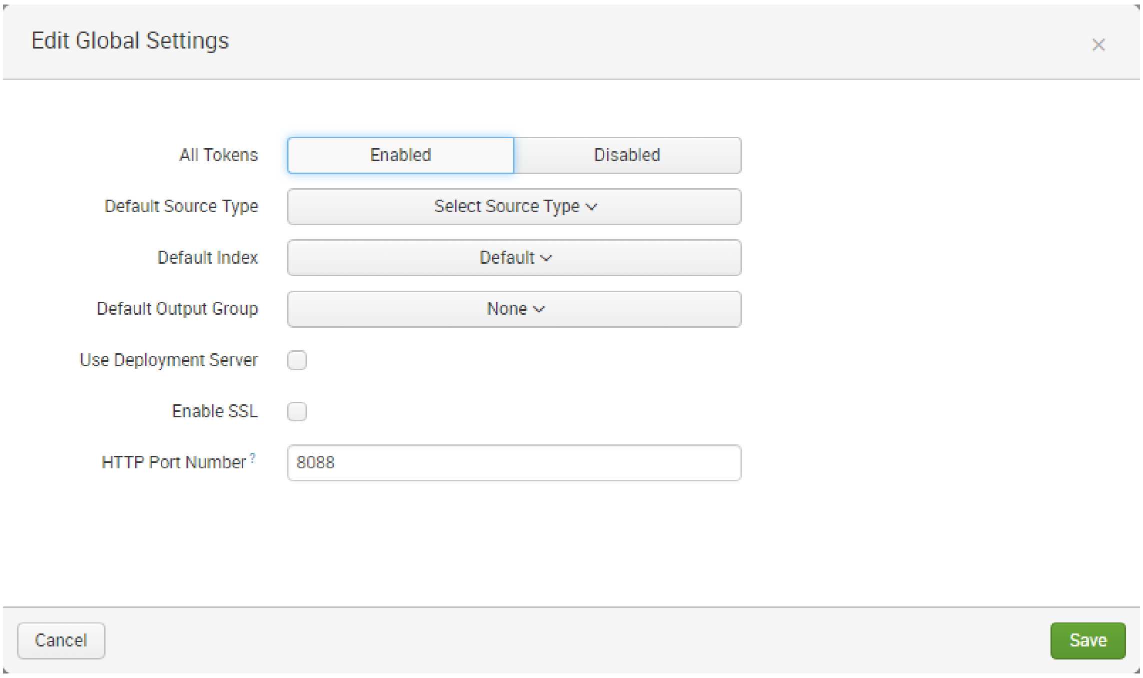
uberAgent natively supports HEC. It can send the data it collects to HEC via HTTP or HTTPS.

Configuring HTTP Event Collector in Splunk Enterprise

Enabling HTTP Event Collector in the UI

To enable HTTP Event Collector (HEC) for uberAgent follow these steps:

- From the system bar, click **Settings** > **Data Inputs**
- On the left side of the page, click **HTTP Event Collector**
- In the upper right corner, click **Global Settings**. The following dialog comes up:



Edit Global Settings

All Tokens: Enabled | Disabled

Default Source Type: Select Source Type

Default Index: Default

Default Output Group: None

Use Deployment Server: □

Enable SSL: □

HTTP Port Number ?: 8088

- In the **All Tokens** toggle button, select **Enabled**
- Optionally change the HEC port or enable SSL/TLS

  - Note that Splunk's default self-signed certificate is not trusted by uberAgent if it is not in the endpoint's operating system certificate store.
  - HTTP Event Collector shares SSL settings with the Splunk management server so check your server.conf for SSL configuration details.

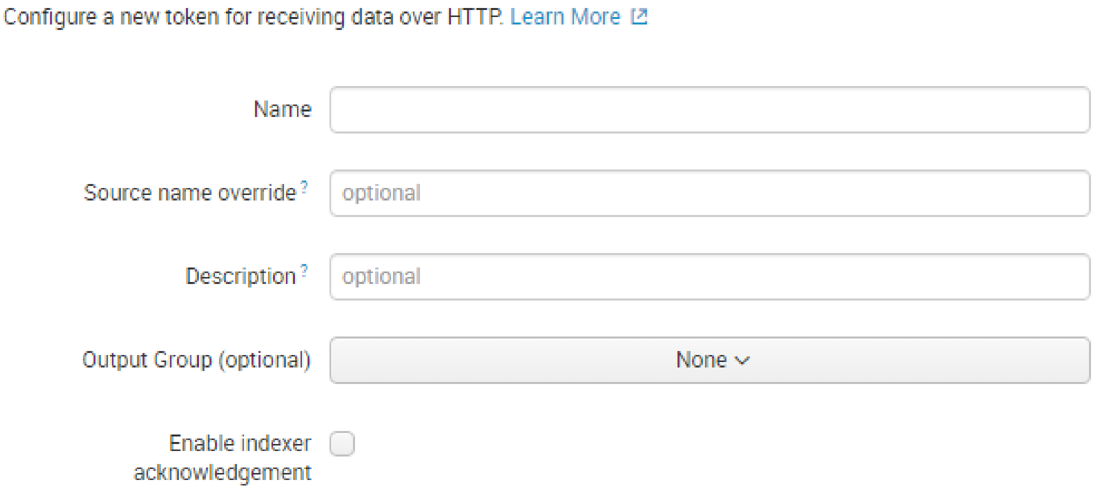
- Click **Save**

Creating an HTTP Event Collector Token in the UI

To use the HTTP Event Collector, you must configure at least one token. The token is what uberAgent uses when it connects to Event Collector to send data.

To create an HEC token for use with uberAgent follow these steps:

- From the system bar, click **Settings** > **Data Inputs**
- On the left side of the page, click **HTTP Event Collector**
- In the upper right corner, click **New Token**. The following dialog comes up:



Configure a new token for receiving data over HTTP. Learn More

Name

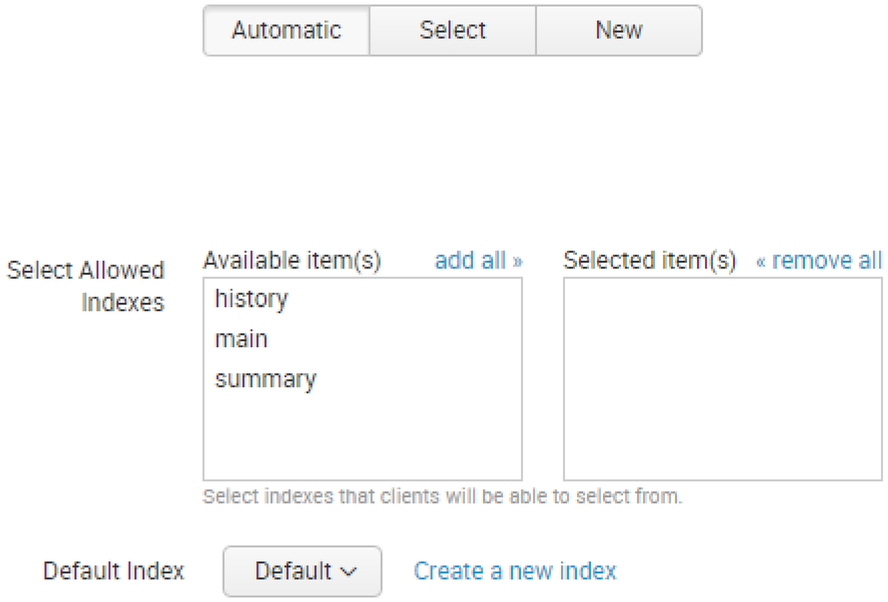
Source name override ? optional

Description ? optional

Output Group (optional) None

Enable indexer acknowledgement □

- Enter a name (e.g. uberAgent) and click **Next**. The following dialog comes up:



Select Allowed Indexes

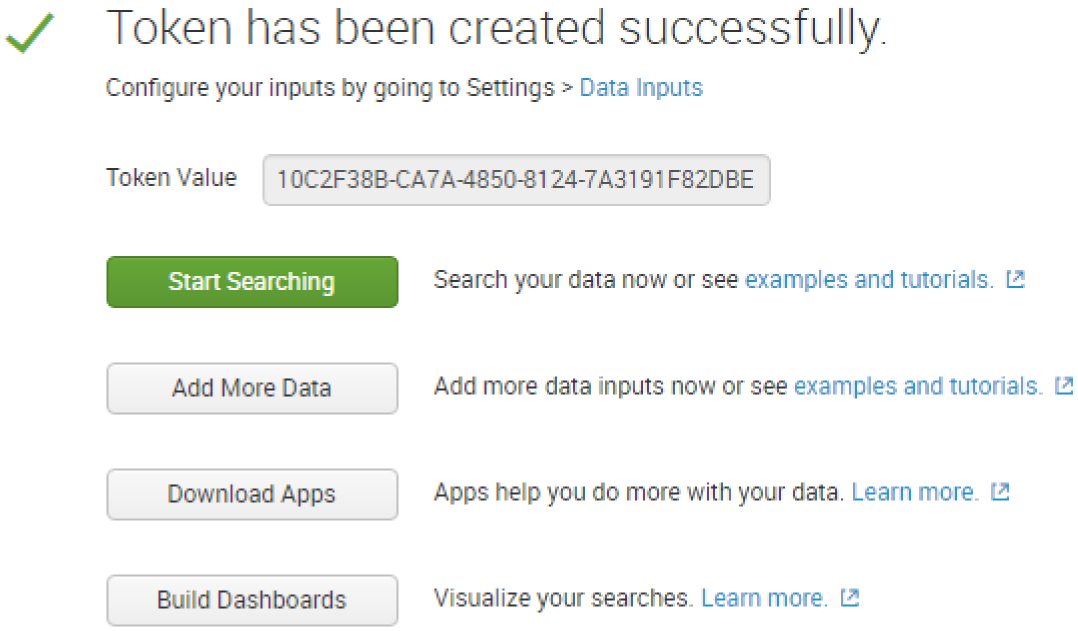
Available item(s): history, main, summary

Selected item(s):

Select indexes that clients will be able to select from.

Default Index: Default

- Leave everything at the defaults and click **Review**
- On the next page click **Submit**
- Copy the token value displayed. In the following screenshot that would be: 10C2F38B-CA7A-4850-8124-7A3191F82DBE



Token has been created successfully.

Configure your inputs by going to Settings > Data Inputs

Token Value: 10C2F38B-CA7A-4850-8124-7A3191F82DBE

Search your data now or see examples and tutorials.

Add more data inputs now or see examples and tutorials.

Apps help you do more with your data. Learn more.

Visualize your searches. Learn more.

HTTP Event Collector Configuration Files

Splunk HTTP Event Collector can be configured through configuration files instead of the UI, which is often required for automation or for configuring settings that are not exposed in the UI. See Splunk's documentation Set up and use HTTP Event Collector with configuration files for details.

Configuring uberAgent to Send to an HTTP Event Collector Input

To configure uberAgent to send its collected data to HEC the following configuration settings are required:

- **Servers:** comma-separated list of URLs starting with http or https, e.g.: `http://server1:8088, https://server2:8088`
- **Protocol:** must be set to `HTTP` (even when sending via HTTPS)
- **RESTToken:** fill in the token created above. The token can optionally be encrypted with the uAEncrypt commandline tool.

Please make sure to review the KB document Reuse of Open HTTP Connections.

Installing and Configuring Elasticsearch & Kibana

This document explains how to install and configure Elasticsearch/Kibana for uberAgent (in this simple tutorial, we assume that you only have a single node with Elasticsearch and Kibana).

Installation

Follow Elastic's installation guide to install Elasticsearch.

Configuration for uberAgent

Elasticsearch 6.x

Elasticsearch 6 is only supported by uberAgent version 6.2 and lower.
Run the following PowerShell command on the Elasticsearch server to create an index template for uberAgent with the required field definitions:

```
Invoke-RestMethod -Uri http://localhost:9200/_template/uberagent -
    Method Put -InFile .\elasticsearch-uberagent.json -ContentType "
    Application/json"
```

On Linux use:

```
curl -XPUT -H 'Content-Type: application/json'http://localhost:9200/_template/uberagent -d@elasticsearch-uberagent.json
```

The file `elasticsearch-uberagent.json` is part of the uberAgent download package.

If you enabled X-Pack security, you need to pass the elastic user (configured during the setup) for the commands above. On Windows use the `-Credential` parameter, for Linux `-u`.

Elasticsearch 7.7 And Lower

Elasticsearch 7.7 and lower are only supported by uberAgent version 6.2 and lower.
Run the following PowerShell command on the Elasticsearch server to create an index template for uberAgent with the required field definitions:

```
Invoke-RestMethod -Uri http://localhost:9200/_template/uberagent?include_type_name=true -Method Put -InFile .\elasticsearch-uberagent.json -ContentType "Application/json"
```

On Linux use:

```
curl -XPUT -H 'Content-Type: application/json'http://localhost:9200/_template/uberagent?include_type_name=true -d@elasticsearch-uberagent.json
```

The file `elasticsearch-uberagent.json` is part of the uberAgent download package.

If you enabled X-Pack security, you need to pass the elastic user (configured during the setup) for the commands above. On Windows use the `-Credential` parameter, for Linux `-u`.

Elasticsearch 7.8 And Higher (Including 8.x)

Elasticsearch 7.8 and higher as well as 8.x are only supported by uberAgent version 6.3 and higher.
Run the following PowerShell command on the Elasticsearch server to create an index template for uberAgent with the required field definitions:

```
Invoke-RestMethod -Uri http://localhost:9200/_index_template/uberagent -Method Put -InFile .\elasticsearch-uberagent.json -ContentType "Application/json"
```

On Linux use:

```
curl -XPUT -H 'Content-Type: application/json'http://localhost:9200/_index_template/uberagent -d@elasticsearch-uberagent.json
```

The file `elasticsearch-uberagent.json` is part of the uberAgent download package.

If you enabled X-Pack security, you need to pass the elastic user (configured during the setup) for the commands above. X-Pack security is enabled by default since Elasticsearch 8. On Windows use the

`-Credential` parameter, for Linux `-u`.

If you want to use an Elasticsearch data stream instead of a classic index, add `"data_stream": { }` , to the file `elasticsearch-uberagent.json`. Example:

```
{

   "priority": 100,
   "index_patterns": [
      "uberagent*"
   ],
   "data_stream": {
  }
 ,
   "template": {

      "mappings": {

      "dynamic": "strict",
         "properties":
         {

            "@timestamp" : {
 "type" : "date" }

          }

       }

    }

 }
```

Kibana

Follow Elastic's installation guide to install Kibana

Dashboards

uberAgent only comes with Splunk dashboards. Nevertheless, there is a way to visualize uberAgent data in Elasticsearch using XOAP's Kibana dashboard package.
XOAP's GitHub repository has 60+ dashboards that visualize a large part of the data set collected by uberAgent.

Install uberAgent

Read on about how to install uberAgent.

To configure uberAgent to send data to Elasticsearch, a configuration section similar to the following is required:

```
[Receiver]
Name = Default
Type = Elasticsearch
Protocol = HTTP
Servers = http://servername:9200
# The setting RESTToken is required if X-Pack security is enabled. It
    is enabled by default since Elasticsearch version 8.
RESTToken = username:password
```

Configuring Microsoft Azure Monitor

This page explains how to configure uberAgent to send the collected data to Microsoft Azure Monitor.

Support/Status

Support for Microsoft Azure Monitor Logs is **experimental** and **and available exclusively on Windows platforms**.

Creating the Log Analytics Workspace

To use uberAgent with Microsoft Azure Monitor Logs, please follow these steps:

- Navigate to https://portal.azure.com and sign in with your Microsoft account or your Organizational account associated with your Microsoft Azure subscription.
- Navigate to the Azure service **Log Analytics workspaces** and create a new workspace by specifying a unique workspace name, subscription, resource group, location and pricing tier or use an existing workspace linked to your Microsoft Azure subscription.
- Open up the **Advanced settings** > **Connected sources** menu and note the `WORKSPACE-ID` and the `PRIMARY-KEY`.

Configuring uberAgent to Send to Azure Monitor Logs

To configure uberAgent to send its collected data to Azure Monitor Logs the following `Receiver` configuration settings are required:

- **Type:** `OMSLogAnalytics`
- **Protocol:** `HTTP`

- **Servers:** `https://WORKSPACE-ID.ods.opinsights.azure.com`
- **RESTToken:** `PRIMARY-KEY`

Example

```
[Receiver]
Name = Default
Type = OMSLogAnalytics
Protocol = HTTP
Servers = https://WORKSPACE-ID.ods.opinsights.azure.com
RESTToken = SECRET-PRIMARY-KEY
```

Configuring Apache Kafka & Confluent REST Proxy

Introduction

What is Apache Kafka

Apache Kafka is a high-performance distributed message queueing and event stream processing platform. Producers send data to Kafka, which buffers the incoming events to disk for a configurable period of time. Kafka logically groups events into topics, which Consumers subscribe to.

What is Confluent REST Proxy

Confluent is a Kafka distribution targeted at enterprises. It adds monitoring, security, and management components to open source Kafka.

While vanilla Kafka only accepts incoming data via a proprietary binary protocol, Confluent comes with a REST proxy that is similar in nature to Splunk HEC or Elasticsearch REST API. REST Proxy is part of the free community edition of Confluent.

When to Use Kafka with uberAgent

By routing uberAgent's data through Kafka, enterprises can make end-user computing metrics available to individual teams within the organization in a publish/subscribe kind of model. Once access has been granted for a topic, a team can start consuming that topic's metrics. While some teams might be interested in uberAgent's asset and inventory information, others might need access to application performance or browser web app usage data.

Kafka makes it possible to distribute uberAgent's metrics in a highly scalable manner, supporting hundreds of thousands of endpoints (data producers) and thousands of consumers. uberAgent natively supports Kafka via the Confluent REST proxy.

Prerequisites

In order for uberAgent to be able to send data to a Kafka backend the following is required:

- Confluent REST Proxy with Confluent Schema Registry
- Apache Kafka

Please see the system requirements for details.

Implementation Details

Topics

Kafka Topics and uberAgent Sourcetypes uberAgent uses Splunk's concept of sourcetypes to logically group events. uberAgent sourcetypes names always start with `uberAgent` followed by two additional sections concatenated by colons. Example:

```
uberAgent:Application:Errors
```

uberAgent sourcetypes are mapped to Kafka topics in a 1:1 relationship. Since colons are not allowed in topic names, they are replaced by underscores. The sourcetype from the above example maps to the following Kafka topic:

```
uberAgent_Application_Errors
```

Topic Creation uberAgent does not create Kafka topics. For uberAgent to be able to send data to Kafka, either topic auto-creation must be enabled (setting `auto.create.topics.enable`), or the topics needed by uberAgent must be created manually.

uberAgent's sourcetypes are documented here. A listing of all sourcetypes is available in the file `uA_sourcetypes_metrics.csv` which is part of the uberAgent dashboard app (included in the uberAgent download).

Partitions

Kafka stores topic data in partitions that can be distributed across nodes in a Kafka cluster. One partition per consumer group is required.

Kafka producers like uberAgent can either select a specific partition for each event or have events be distributed randomly across partitions. uberAgent combines the two approaches: it specifies the hostname as key for each event. This ensures that all events from a specific host are always stored in the same partition.

Schema and Data Format

uberAgent uses Avro as data format when sending to Kafka. Avro requires a schema definition for every event. This guarantees high data quality and easier processing by downstream consumers.

In order to minimize network data volume, uberAgent sends the full schema definition only for the first event of a topic. All further events of the same topic reference the topic's schema definition by the ID returned from the schema registry in response to the first event.

Avro requires REST Proxy to be configured with a schema registry URL (configuration setting `schema.registry.url`).

Event Metadata Each uberAgent event has the same default metadata fields in Kafka as in Splunk:

- time
- host
- index
- source
- sourcetype

The sourcetype is used as the key for the Kafka partition selection (see above) and added to each event as a dedicated field.

REST Proxy API Details uberAgent sends data to the Confluent REST Proxy API v2 via HTTP POST to `/topics/SOURCETYPE` (docs). The HTTP content type is `application/vnd.kafka.v2+json`.

Configuration

uberAgent

To configure uberAgent to send data to Kafka via Confluent REST Proxy a configuration section similar to the following is required:

```
[Receiver]
Name = Kafka
Type = Kafka
Protocol = HTTP
Servers = https://confluent-rest-proxy.company.com:8083/
TLSClientCertificate = LocalMachine\MY\abcd1234567891234567891234567891234567891
```

The receiver `Name` can be any string. The `Protocol` must be `HTTP` - it stands for a REST endpoint accessed via http or https. The `Servers` parameter accepts a comma-separated list of REST proxies that are contacted in a round-robin fashion for fault tolerance and load sharing. `TLSClientCertificate` optionally specifies a client certificate to use for authentication (see below).

Kafka

Please see the implementation details section above for information on how to configure Confluent and Kafka for receiving data from uberAgent.

Authentication

Confluent REST Proxy supports authentication via an HTTPS client certificate on the endpoint. Make sure to specify the `TLSClientCertificate` option in the uberAgent configuration.

It is not necessary to issue individual certificates to clients. A single certificate with `CN=uberAgent` (or similar) might be best.

Confluent REST Proxy propagates the client certificate to the Kafka broker in one of two ways:

- TLS
- SASL

The following sections show the configuration steps for each option.

Client Certificate Propagation via TLS In order for certificate propagation from REST Proxy to the Kafka brokers to work, the following requirements must be met:

- REST Proxy and the Kafka brokers must be configured for TLS support
- A root CA that issues all required certificates and is trusted by all components is highly recommended
- REST Proxy's client certificate store must contain the certificate (inluding private key) uberAgent authenticates with

REST Proxy Configuration This example shows a simple configuration with all components on one server. In a real-world deployment that would most likely not be the case. Adjust the URLs accordingly.

Add the following to the file `/etc/kafka-rest/kafka-rest.properties`:

```
###################################################
#
# Schema registry
#
###################################################

schema.registry.url=http://localhost:8081

###################################################
#
# REST proxy to broker connection
#
###################################################

bootstrap.servers=localhost:9093
client.security.protocol=SSL

# The client keystore contains the client certificate (including the private key) that is used by uberAgent to authenticate. REST proxy passes it on to the broker.
client.ssl.keystore.location=/opt/vastlimits/certificates/kafka.clientkeystore.jks
client.ssl.keystore.password=PASSWORD3
client.ssl.key.password=PASSWORD3

# The truststore contains the collection of CA certificates trusted by this application
client.ssl.truststore.location=/opt/vastlimits/certificates/kafka.truststore.jks
client.ssl.truststore.password=PASSWORD1

###################################################
#
# Miscellaneous
#
###################################################

# Listeners (port 8083 cannot be used for REST Proxy because it is used by Confluent Connect)
listeners=https://0.0.0.0:8084

###################################################
#
# Security plugin
#
###################################################

```

```
# License for commercial edition (required for the security plugin)
confluent.license=LICENSE_KEY

# Enable the REST proxy security plugin
kafka.rest.resource.extension.class=io.confluent.kafkarest.security.KafkaRestSecurityResourceExtension

# Whether or not to require the HTTPS client to authenticate via the server's trust store (required if the security plugin is enabled)
ssl.client.auth=true

###################################################
#
# HTTPS from clients to REST proxy
#
###################################################

# Web server certificate for REST proxy
ssl.keystore.location=/opt/vastlimits/certificates/kafka.keystore.jks
ssl.keystore.password=PASSWORD2
ssl.key.password=PASSWORD2

# The truststore lists certificates that are allowed to connect. Specify either a single CA certificate or individual client certificates.
ssl.truststore.location=/opt/vastlimits/certificates/kafka.truststore.jks
ssl.truststore.password=PASSWORD1
```

Kafka Broker Configuration Add the following to the file `/etc/kafka/server.properties` :

```
# The truststore lists certificates that are allowed to connect. Specify either a single CA certificate or individual client certificates.
ssl.truststore.location=/opt/vastlimits/certificates/kafka.truststore.jks
ssl.truststore.password=PASSWORD1

# Server certificate for the broker
ssl.keystore.location=/opt/vastlimits/certificates/kafka.keystore.jks
ssl.keystore.password=PASSWORD2
ssl.key.password=PASSWORD2

# Both plaintext and SSL listeners on different ports. Plaintext can be used from the schema registry, for example.
listeners=PLAINTEXT://:9092,SSL://:9093

# Enforce authentication for clients connecting via SSL
ssl.client.auth=required
```

Client Certificate Propagation via SASL We are waiting for information from Confluent on some specifics of this configuration. We will update this section once that is available.

Configuring Microsoft Azure Data Explorer (ADX) & Azure Event Hubs

Introduction

What is Azure Data Explorer (ADX)

Microsoft Azure Data Explorer (ADX) is a data analytics platform designed for querying and analyzing large volumes of data in near real time.

What is Azure Event Hubs

Microsoft Azure Event Hubs is a real-time data ingestion service based on Apache Kafka. It can process millions of events per second which subsequently can be made available to hundreds of individual consumers (applications that read and process incoming data).

Why Combine Azure Data Explorer With Azure Event Hubs for uberAgent Data

In enterprise scenarios, uberAgent's endpoint agents may send events concurrently from hundreds of thousands of devices. This requires a backend service that is capable of ingesting millions of events per second. In the Azure cosmos, Event Hubs is exactly such a service. Once ingested, it buffers uberAgent's events until they've been routed to Azure Data Explorer (ADX), where they can be efficiently queried from various additional services via the Kusto Query Language (KQL).

Advantages of Using Azure Data Explorer & Azure Event Hubs for uberAgent Data

By routing uberAgent's data through Azure Event Hubs to Azure Data Explorer, enterprises can make DEX and endpoint security metrics available to individual teams within the organization in a publish/-subscribe kind of model. Once access has been granted to an ADX database, a team can start working with the metrics it stores.

Azure Event Hubs makes it possible to distribute uberAgent's metrics in a highly scalable manner, supporting hundreds of thousands of endpoints (data producers) and thousands of consumers. uberAgent natively supports Azure Data Explorer via Azure Event Hubs.

Prerequisites

In order for uberAgent to be able to send data to an Azure Data Explorer backend, the following Azure services are required:

- Microsoft Azure Event Hubs
- Microsoft Azure Data Explorer Cluster
- Microsoft Azure Enterprise Application

Please see the system requirements for details.

Setup

Navigate to the Azure portal and sign in with an organizational account associated with your Microsoft Azure subscription.

Creating an Azure Enterprise Application

In order to authenticate to Event Hubs, an Enterprise Application needs to be created. Follow the steps in the Microsoft documentation. Make sure to obtain a **client ID** and a **client secret**.

Creating the Azure Event Hubs Namespace & Instance

1. **Create a new Event Hubs namespace** Azure service by specifying a resource group, a unique namespace name, a location, and a pricing tier.
2. Assign the following permissions to the namespace for the enterprise application you created above:
    - Azure Event Hubs Data Sender
    - Schema Registry Contributor (Preview)
    - Schema Registry Reader (Preview)
3. Open up the newly created Event Hubs namespace and **create an Event Hubs instance** by specifying a unique name, the partition count (1 is sufficient), and the retention time.

Creating the Azure Data Explorer Cluster

1. **Create a new Azure Data Explorer Cluster** by specifying a resource group, a unique cluster name, and the workload.

2. **Create a new database** in the newly created Azure Data Explorer Cluster by specifying a database name, the retention period, and the cache period.
3. Open the newly created database and **create a new data connection**, using the Event Hubs instance created above as the source. Specify a data connection name, and the name of the Event Hubs namespace. Make sure to allow routing the data to other databases and leave the table name empty. Set the data format to `CSV`.

Creating the Azure Data Explorer Tables

The uberAgent endpoint agent does not create the database tables in Azure Data Explorer. The script `uberAgentSchema.AzureDataExplorer.kql` to create the tables is part of the uberAgent download package.

Configuring Azure Event Hubs as Data Receiver in uberAgent

Receiver Stanza The following configuration settings are required in a `Receiver` stanza:

- **Type:** `AzureEventHubs`
- **Protocol:** `HTTP`
- **AzureEventHubsConfigurationId:** The ID of the AzureEventHubsConfiguration stanza (see below).

AzureEventHubsConfiguration Stanza The following configuration settings are required in an `AzureEventHubsConfiguration` stanza:

- **Id:** Unique GUID this stanza is referenced by from a `Receiver` stanza.
- **AuthenticationURL:** The authentication URL (optional). The default value is: `https://login.microsoftonline.com/###UA_TENANTID###/oauth2/token`
- **SendDataURL:** The URL to send the data to (optional). The default value is: `https://###UA_EVENTHUBSNAMESPACE###.servicebus.windows.net/###UA_HUBNAME###/messages`
- **AzureEventHubsNameSpace:** The name of the Azure Event Hubs namespace. Placeholder variable: `###UA_EVENTHUBSNAMESPACE###`
- **AzureHubName:** The name of the Azure Event Hubs instance. Placeholder variable: `###UA_HUBNAME###`
- **AzureTenantID:** The Entra tenant ID. Placeholder variable: `###UA_TENANTID###`
- **AzureClientID:** The Entra enterprise application client ID.
- **AzureClientSecret:** A client secret associated with the client ID.

The fields `###UA_TENANTID###`, `###UA_EVENTHUBSNAMESPACE###`, and `###UA_HUBNAME###` are placeholders and are replaced by the agent with their corresponding value.

Example

```
[Receiver]
Name = vast limits Azure Event Hubs
Type = AzureEventHubs
Protocol = HTTP
AzureEventHubsConfigurationId = 82204bf8-fc99-4d59-8903-7206234dcbc8

[AzureEventHubsConfiguration]
Id = 82204bf8-fc99-4d59-8903-7206234dcbc8
AzureEventHubsNameSpace = vastlimits
AzureHubName = uberagent
AzureTenantID = YOUR_TENANT_TENANT_ID
AzureClientID = YOUR_ENTRA_ENTERPRISE_APPLICATION_CLIENT_ID
AzureClientSecret = YOUR_AZURE_CLIENT_SECRET
```

Implementation Details

Authentication Method

Authentication from the uberAgent endpoint agent to the Event Hubs instance happens by way of Azure AD JSON web tokens (JWT).

Routing Events to the Proper ADX Table

uberAgent sends many different types of events, distinguished by a so-called sourcetype. Each sourcetype has its own field definition and, therefore, needs to be stored in its own ADX table.

While uberAgent sends all types of events to a single Event Hubs ingestion service, from there the incoming events need to routed to the proper ADX tables depending on each event's sourcetype. This routing is automatically accomplished by adding an HTTP header with the name of the ADX table, e.g.:

```
"Table" : "Process_ProcessStartup"
```

Timer scripts

Timers are mostly used to determine metrics (e.g. `SessionDetail`). They can also be used to start custom scripts (see here). The output of the scripts is sent to the backend where the timer name is part of the sourcetype. In most backends the sourcetype does not need to be created in advance, or it is created by uberAgent. In Azure Data Explorer the table needs to be created manually by the customer otherwise ingestion errors occur.
The table name has the following format: `Script_`.

The following gives an example of the KQL script needed to run to ingest the result data of a custom script into the corresponding table in Azure Data Explorer.

```
[Timer]
Name = DetermineDepartment
Interval = 30000
Script = C:\Scripts\DetermineDepartment.cmd
ScriptContext = UserSessionAsUser
```

The output of the script `DetermineDepartment.cmd` is:

`Department=Sales UserName=John`

The KQL script to create the table and the ingestion mapping in Azure Data Explorer for this timer is:

```
.execute database script <|
.create table Script_DetermineDepartment (Timestamp: long, Host: string
    , Department: string, UserName: string);
.create table Script_DetermineDepartment ingestion csv mapping '
    Script_DetermineDepartment_mapping' '[{
 "column":"Timestamp", "Properties":{
 "Ordinal":"0" }
  }
 , {
 "column":"Host", "Properties":{
 "Ordinal":"1" }
  }
 , {
 "column":"Department", "Properties":{
 "Ordinal":"2" }
  }
 , {
 "column":"UserName", "Properties":{
 "Ordinal":"3" }
  }
 ]';
```

The fields `Timestamp` and `Host` are mandatory and are automatically added by uberAgent. The field names in the output do not necessarily have to match the field names in the table.

Installing uberAgent

Installing uberAgent is quick and straightforward. Due to the product's flexibility, there is often more than one way of doing things. In such a case we describe the recommended configuration and link to supported alternatives.

Prerequisites

- Read about the architecture options.
- If you do not have a running Splunk installation yet set up Splunk.
- Download uberAgent and extract the archive. You should have a directory `uberAgent components` with the following content:
  - A subdirectory `uberAgent_endpoint`
  - Packed Splunk app `uberAgent_indexer.tgz`
  - Packed Splunk app `uberAgent_searchhead.tgz`
  - Packed Splunk app `uberAgent_ESA_searchhead.tgz`

Installation Steps

Installing uberAgent consists of the following steps:

1. Installing the **Splunk** apps
2. Installing the **Windows** endpoint agent
3. Installing the **macOS** endpoint agent
4. Installing Splunk **Universal Forwarder** (optional)
5. Installing the **Chrome** browser extension (optional)
6. Installing the **Edge** Browser Extension (optional)
7. Installing the **Firefox** Browser Extension (optional)
8. Installing the **Internet Explorer** Browser Extension (optional)

Installing the Splunk Apps

Upgrade

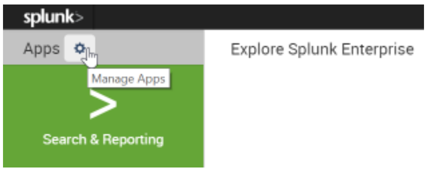
If you are upgrading from an earlier version of the uberAgent Splunk apps please make sure to follow the upgrade instructions.

Please consult the release notes for possible changes in configuration or functionality.

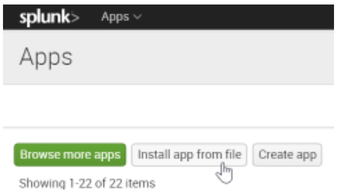
Manual Installation

- Go to the Splunk console's home page by navigating to `http://servername:8000` in your browser.

- Click **Manage apps**:



- Click **Install app from file**:



- Select `uberAgent_indexer.tgz` and click **Upload**
- The uberAgent indexer app is now installed
- Install `uberAgent_searchhead.tgz` in the same way
- Optionally install the uberAgent ESA search head app `uberAgent_ESA_searchhead.tgz`, too
- Click **Settings** > **Server controls** > **Restart Splunk**

Splunk Enterprise Security
uberAgent ESA comes with native support for Splunk Enterprise Security. Install the following apps on your Enterprise Security server to enjoy the benefits:

- `uberAgent_searchhead.tgz`
- `uberAgent_ESA_searchhead.tgz`
- `uberAgent_ESA_ES_companion.tgz`.

Available on Splunkbase.

Using Splunk Cloud
The installation process is different when using uberAgent in combination with Splunk Cloud. This knowledgebase article contains all the necessary details.

Distributed Splunk Deployment

If you have a distributed Splunk deployment with separate search heads and indexers please deploy the **indexer app to all indexers as well as heavy forwarders** and the **search head app to all search heads**.

Alternative Architectures

Note: This is optional and not required for the recommended architecture.

If you decided to have uberAgent send data to Splunk through a locally installed Universal Forwarder on the monitored endpoints you need to enable receiving Universal Forwarder data as described here, i.e., through **Settings** > **Forwarding and receiving** > **Receive data** > **Add new** > **9997** > **Save**.

Sending to Splunk HTTP Event Collector

Note: This is only required when you want uberAgent to send directly to Splunk Cloud, but it can optionally be used with Splunk Enterprise, too.

uberAgent can send the data it collects via HTTP or HTTPS to a Splunk data input called HTTP Event Collector (HEC). Please follow these steps to enable and configure HTTP Event Collector.

Installing the Windows Endpoint Agent

The agent installer is available as an MSI package. The MSI can either be installed manually or unattended through existing software deployment tools or Splunk's Deployment Server.

> **Note**
>
> **Securing the Configuration Directory**
>
> uberAgent can be configured via the `%ProgramData%` directory (details). It is important to secure this directory, or standard users might be able to elevate their privileges to SYSTEM and/or abuse uberAgent.

Starting with version 7.0.2, uberAgent's installer secures the agent's `%ProgramData%` directories automatically. Two new MSI parameters provide control over the process: `PROGRAMDATA_CONFIGDIR_RESETPERMISSIONS` and `PROGRAMDATA_CONFIGDIR_DELETEFILES` (see below).

We recommend disabling those MSI parameters only if you're managing the security of `%ProgramData%\vast limits` and its subdirectories via other means. We recommend the following permissions:

Administrators: full control **SYSTEM:** full control

Optionally provide read access to standard users on scripts that are to be executed in user context:

Users: read

> **Note**
>
> **Changed Startup Behavior After Installation**
>
> Starting with uberAgent 7.2.1, the uberAgent installer does not start the service automatically after a successful installation. Please update any agent deployment procedures to reflect changes.

Manual Installation

Run the batch file `uberAgent_endpoint\bin\manual-install.cmd`.

Configuration

uberAgent can be configured very flexibly. By editing the configuration you can switch metrics on or off, change the data collection frequency and significantly reduce the data volume.

> **Note**
>
> **Template Configuration Files Available on GitHub**
>
> Starting with uberAgent 7.3.1, the agent installer does not store the template configuration files on disk any more. You can download the latest template config files for your installed uberAgent version on GitHub.
>
> **Configuration Directory Change - Windows**
>
> Starting with uberAgent 7.3.0, the agent **no longer** searches for configuration files in the **installation directory**.
> To configure uberAgent, download the template configuration files from GitHub (see above), make your changes, deploy the configuration files to: `%PROGRAMDATA%\vast limits\uberAgent\Configuration`, and start the uberAgent service. For more details, please see: Configuration via Local Config Files.

License File

If you have a license file for uberAgent, copy it to the installation directory (default: `%ProgramFiles%\vast limits\uberAgent`). Without a license file, uberAgent displays a splash screen during logon.

Installation Through a Software Deployment Tool

Install the appropriate MSI file from the directory `uberAgent_endpoint\bin` depending on the bitness of your machine: `uberAgent-32.msi` or `uberAgent-64.msi`.

MSI Parameters

Specify the following MSI parameters:

INSTALLDIR

- **Required:** no
- **Description:** installation directory
- **Valid values:** any local file system path

PROGRAMDATA_CONFIGDIR_RESETPERMISSIONS

- **Required:** no
- **Default:** `1`
- **Description:** Set secure permissions on uberAgent's ProgramData directory (`%ProgramData%\vast limits\uberAgent`).
- **Valid values:**
  - `0`: disabled
  - `1`: enabled

PROGRAMDATA_CONFIGDIR_DELETEFILES

- **Required:** no
- **Default:** `1`
- **Description:** Delete existing config files in uberAgent's ProgramData directory (`%ProgramData%\vast limits\uberAgent`). Disable this setting only if you're removing potentially malicious existing config files as part of your own deployment package logic.
- **Valid values:**
  - `0`: disabled
  - `1`: enabled

License File

If you have a license file for uberAgent, copy it to the installation directory (default: `%ProgramFiles%\vast limits\uberAgent`). Without a license file, uberAgent displays a splash screen during logon.

Installation Through Splunk Deployment Server

Note: Deployment Server can only be used with Splunk Enterprise and requires Splunk Universal Forwarder on the endpoint as deployment client.

uberAgent

Copy the directory `uberAgent_endpoint` from the unzipped uberAgent download package to `$SPLUNK_HOME\etc\deployment-apps` on your deployment server.

Note: `$SPLUNK_HOME` refers to the base directory of the Splunk installation, typically `%ProgramFiles%\Splunk`.

Configuration

To deploy a customized configuration file, copy it into the directory `$SPLUNK_HOME\etc\deployment-apps\uberAgent_endpoint\bin`. This overwrites the default configuration file from the installation package.

License File

If you have a license file for uberAgent, copy it into the directory `$SPLUNK_HOME\etc\deployment-apps\uberAgent_endpoint\bin`.

Serverclass

Create a file called `serverclass.conf` in `$SPLUNK_HOME\etc\system\local` on your deployment server. `Serverclass.conf` defines what to deploy where. For a quick start paste the following content into `Serverclass.conf` to deploy uberAgent to all Windows machines. You may want to fine-tune this to suit your needs.

```
# [global]
# We cannot match by machine type here. We'll do that on the app level below.
whitelist.0 = *

# Define a serverclass
[serverClass:windows]
# Deploy only to Windows machines
machineTypesFilter = windows-*

# Define which apps to deploy to the serverclass
[serverClass:windows:app:uberAgent_endpoint]
stateOnClient = enabled
restartSplunkd = true
```

To make Splunk read the new file `serverclass.conf`, run the following command:

```
$SPLUNK_HOME\splunk.exe reload deploy-server
```

Citrix Site Monitoring

If some or all of your endpoints are running the Citrix Virtual Apps and Desktops (CVAD) VDA, you should install uberAgent on the Citrix delivery controller(s), too. Please see this page for details.

Endpoint to Backend Communication Via Splunk Universal Forwarder

Note: This is optional and not required for the recommended architecture.

If you decided to implement the alternative endpoint to backend communication path via Splunk Universal Forwarder, you need to install Universal Forwarder on each endpoint.

Imaging & Citrix PVS

If you intend to copy the agent installation via an imaging method or Citrix PVS, we recommend you remove instance-specific information. To do that, follow these steps right before capturing the image:

- Stop the service `uberAgent` (but leave the start type at `automatic`).
- Open an administrative command prompt.
- Run the command: `reg delete "HKLM\SOFTWARE\vast limits\uberAgent"/f /reg:64`.
- **Optionally**: delete existing log files.
- **Optionally**: delete existing Persistent Output Queue files/folders.
- Prepare the machine for cloning as necessary, but do not reboot.

If you have Splunk Universal Forwarder installed, please follow the steps listed here, too.

Installing the macOS Endpoint Agent

The agent installer is available as a PKG file. It can either be installed manually or unattended through existing software deployment tools. Also, most device management solutions support the distribution of such packages natively.

> **Note 1**
>
> **Securing the Configuration Directory**
>
> To protect the configuration of uberAgent, **starting with version 6.1.2, the installer automatically sets the permissions for the agent's directory so that only `root` can read or access the installation directory.**

> **Note 2**
>
> **Template Configuration Files Available on GitHub**
>
> Starting with uberAgent 7.3.1, the agent installer does not store the template configuration files on disk any more. You can download the latest template config files for your installed uberAgent version on GitHub.

> **Note 3**
>
> **Running uberAgent after installing the Endpoint Agent**
>
> The uberAgent daemon does not start automatically upon completion of the setup process. It will initiate automatically following a system restart.
> Alternatively, you may start it manually using launchctl, provided you have root privileges: `launchctl bootstrap system /Library/LaunchDaemons/com.vastlimits.uberAgent.plist`

Requirements

uberAgent on macOS requires the entitlement for certain privacy privileges through Apple's Transparency, Consent and Control (TCC) Framework in order to run properly. These can either be assigned manually, as described below or distributed as part of a PPPC profile. The latter can either be created manually by following the documentation provided by Apple or make use of tools like the PPPC-Utility. An example profile is available at the bottom of this page.

Full Disk Access

uberAgent makes use of Apple's EndpointSecurity (ES) framework (available since macOS 10.15), which requires explicit user authorization. Authorization is granted manually, by adding `/Library/uberAgent/uberAgent.app` to **System Preferences** > **Security & Privacy** > **Privacy** > **Full Disk Access**.

Note: Without this permission uberAgent won't be able to start and will terminate itself. When this happens, you'll find an error log message indicating that the ES client could not be started due to a missing TCC approval.

Accessibility Access

uberAgent uses the Accessibility framework on macOS to determine parts of the Session Detail metrics. Authorization is granted manually, by adding `/Library/uberAgent/uberAgent.app` to **System Preferences** > **Security & Privacy** > **Privacy** > **Accessibility**.

Installation method

Interactive

- The installation of uberAgent can be started by opening `uberAgent.pkg` and following the instructions on the screen.
- Since uberAgent is installed as a system-wide daemon, the installer will ask for a password for elevated access rights to install it.

Command Line

- The Installer can also be used from the command line to install uberAgent using the command `installer -pkg uberAgent.pkg -target /`
- This requires root privileges. A command line installation can be executed locally or remotely, e.g., using SSH.

License File

If you have a license file for uberAgent, copy it to the following directory `/Library/Application Support/uberAgent`. Without a license file, uberAgent displays a splash screen during logon. Contact us for an evaluation license.

Installation Through Splunk Deployment Server

Note: Deployment Server can only be used with Splunk Enterprise and requires Splunk Universal Forwarder on the endpoint as deployment client. Please make sure that Splunk Universal Forwarder has **sufficient privileges** to perform system-wide installations, e.g. by enabling boot-start.

uberAgent

Copy the directory `uberAgent_endpoint` from the unzipped uberAgent download package to `$SPLUNK_HOME/etc/deployment-apps` on your deployment server. Please make sure to apply the executable flag to the installation script by executing `chmod +x silent-install.sh`. Besides that, the `uberAgent_endpoint` folder is ready to use for deployment on Windows as well as macOS operating systems.

Note: `$SPLUNK_HOME` refers to the base directory of the Splunk installation, typically `/opt/splunk` on Linux.

Configuration

To deploy customized configuration files, copy them into the directory `$SPLUNK_HOME/etc/deployment-apps/uberAgent_endpoint/bin`. This overwrites existing files under `/Library/Application Support/uberAgent/`.

License File

If you have a license file for uberAgent, copy it into the directory `$SPLUNK_HOME/etc/deployment-apps/uberAgent_endpoint/bin`.

Serverclass

Create a file called `serverclass.conf` in `$SPLUNK_HOME/etc/system/local` on your deployment server. `serverclass.conf` defines what to deploy where. For a quick start, paste the following content into `serverclass.conf` to deploy uberAgent to all macOS machines. You may want to fine-tune this to suit your needs.

```
# [global]
# We cannot match by machine type here. We'll do that on the app level
    below.
whitelist.0 = *

# Define a serverclass
[serverClass:macOS]
# Deploy only to macOS machines
machineTypesFilter = darwin-x86_64

# Define which apps to deploy to the serverclass
[serverClass:macOS:app:uberAgent_endpoint]
stateOnClient = enabled
restartSplunkd = true
```

To make Splunk read the new file `serverclass.conf` run the following command:

```
$SPLUNK_HOME/bin/splunk reload deploy-server
```

Configuration

uberAgent can be configured very flexibly. By editing the configuration, you can switch metrics on or off, change the data collection frequency and significantly reduce the data volume.

Example uberAgent PPPC Profile

This is an exemplary privacy preferences policy control (PPPC) profile to grant uberAgent the necessary TCC privileges.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>PayloadContent</key>
    <array>
        <dict>
            <key>PayloadDescription</key>
            <string>uberAgent</string>
            <key>PayloadDisplayName</key>
            <string>uberAgent</string>
            <key>PayloadIdentifier</key>
            <string>5FD70527-9AC3-4667-82AF-8F8F44C95C94</string>
            <key>PayloadOrganization</key>
            <string>vast limits GmbH</string>
            <key>PayloadType</key>
            <string>com.apple.TCC.configuration-profile-policy</string>
            <key>PayloadUUID</key>
            <string>E86AC250-6BF9-4C91-BB0A-5890F000A48C</string>
            <key>PayloadVersion</key>
            <integer>1</integer>
            <key>Services</key>
            <dict>
                <key>Accessibility</key>
                <array>
                    <dict>
                        <key>Allowed</key>
                        <true/>
                        <key>CodeRequirement</key>
                        <string>anchor apple generic and identifier "com.vastlimits.uberAgent" and (certificate leaf[field.1.2.840.113635.100.6.1.9] /* exists */ or certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] = "64N35HHH3F")</string>
                        <key>Comment</key>
                        <string></string>
                        <key>Identifier</key>
                        <string>com.vastlimits.uberAgent</string>
                        <key>IdentifierType</key>
                        <string>bundleID</string>
                    </dict>
                </array>
                <key>SystemPolicyAllFiles</key>
                <array>
                    <dict>
                        <key>Allowed</key>
                        <true/>
                        <key>CodeRequirement</key>
                        <string>anchor apple generic and identifier "com.vastlimits.uberAgent" and (certificate leaf[field.1.2.840.113635.100.6.1.9] /* exists */ or certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] = "64N35HHH3F")</string>
                        <key>Comment</key>
                        <string></string>
                        <key>Identifier</key>
                        <string>com.vastlimits.uberAgent</string>
                        <key>IdentifierType</key>
                        <string>bundleID</string>
                    </dict>
                </array>
            </dict>
        </dict>
    </array>
    <key>PayloadDescription</key>
    <string>uberAgent</string>
    <key>PayloadDisplayName</key>
    <string>uberAgent</string>
    <key>PayloadIdentifier</key>
    <string>5FD70527-9AC3-4667-82AF-8F8F44C95C94</string>
    <key>PayloadOrganization</key>
    <string>vast limits GmbH</string>
    <key>PayloadScope</key>
    <string>System</string>
    <key>PayloadType</key>
    <string>Configuration</string>
    <key>PayloadUUID</key>
    <string>F53E14C7-D9ED-4933-8A71-87FD74C9870D</string>
    <key>PayloadVersion</key>
    <integer>1</integer>
</dict>
</plist>
```

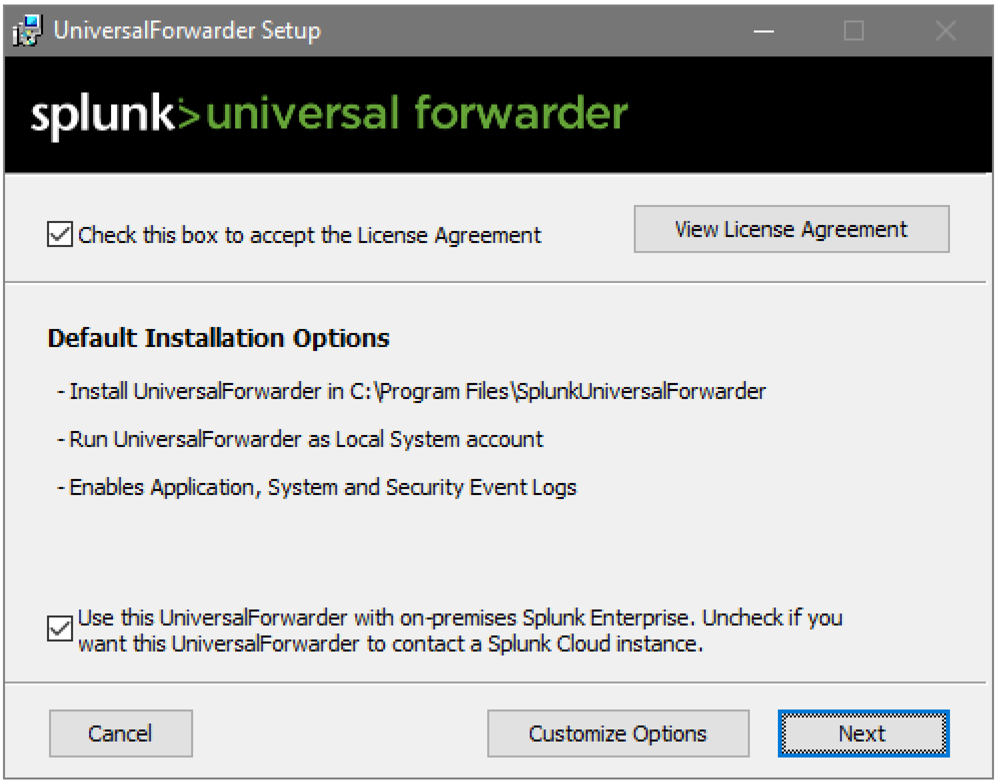
Installing Splunk Universal Forwarder

If you have chosen an architecture option where uberAgent interacts with Splunk through Splunk's Universal Forwarder you need to install Universal Forwarder on every machine along with uberAgent.

The following description shows you how to install and configure Universal Forwarder for uberAgent. In this example, the Splunk server's FQDN is `srv1.hk.test`.

Installation

Launch the Universal Forwarder MSI file. Accept the license agreement and click **Next**.



☑ Check this box to accept the License Agreement

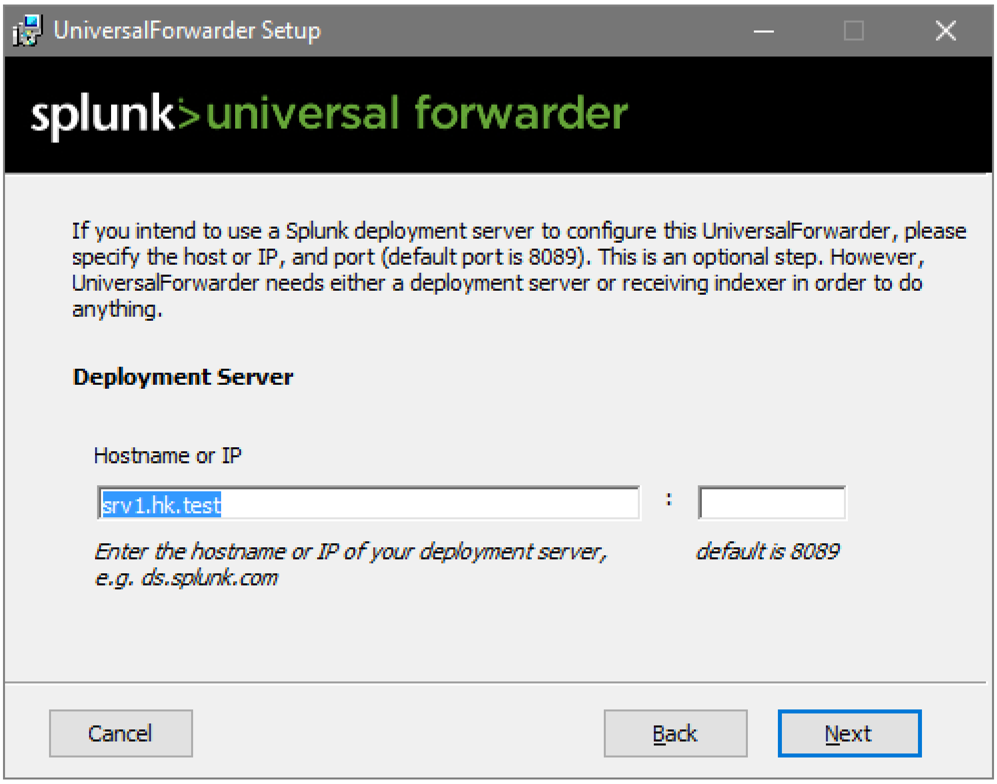
Default Installation Options

- Install UniversalForwarder in C:\Program Files\SplunkUniversalForwarder
- Run UniversalForwarder as Local System account
- Enables Application, System and Security Event Logs

☑ Use this UniversalForwarder with on-premises Splunk Enterprise. Uncheck if you want this UniversalForwarder to contact a Splunk Cloud instance.

Deployment Server

If you have Splunk Enterprise, you can use Splunk's Deployment Server functionality to deploy apps. In that case, specify the Splunk server name. You can leave the port empty to use the default. If you have Splunk Free leave both fields empty.



If you intend to use a Splunk deployment server to configure this UniversalForwarder, please specify the host or IP, and port (default port is 8089). This is an optional step. However, UniversalForwarder needs either a deployment server or receiving indexer in order to do anything.

Deployment Server

Hostname or IP

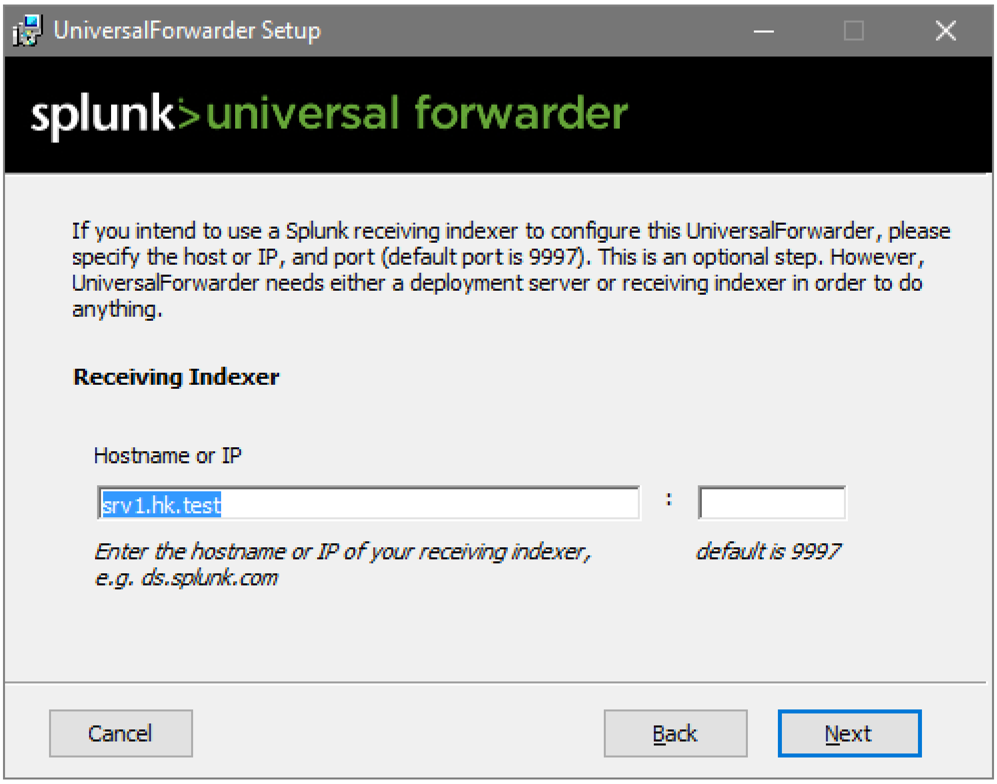
srv1.hk.test :

Enter the hostname or IP of your deployment server, e.g. ds.splunk.com

default is 8089

Indexer

Specify the name of your Splunk Indexer. In this simple walkthrough, our Splunk server also acts as an indexer. Again, the port is left at the default value:



If you intend to use a Splunk receiving indexer to configure this UniversalForwarder, please specify the host or IP, and port (default port is 9997). This is an optional step. However, UniversalForwarder needs either a deployment server or receiving indexer in order to do anything.

Receiving Indexer

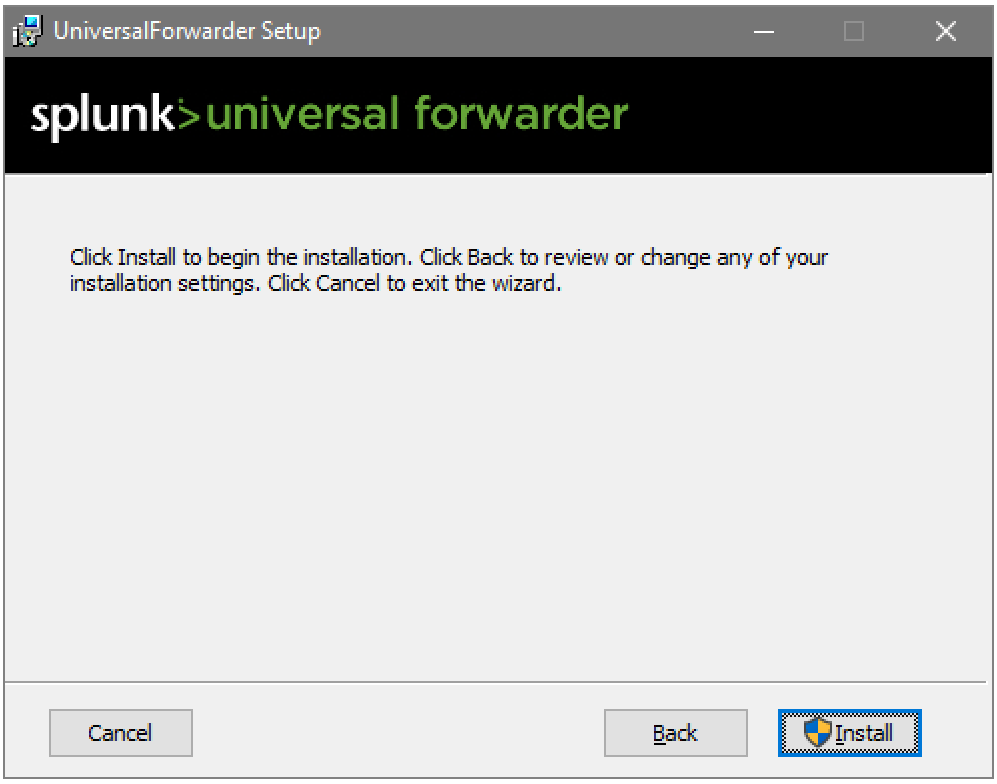
Hostname or IP: srv1.hk.test :

Enter the hostname or IP of your receiving indexer, e.g. ds.splunk.com

default is 9997

Finished

That's all!



Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

Configuration

Receive uberAgent Data via TCP Port

If you want Universal Forwarder to handle all Splunk communications, you need to configure uberAgent to pass its output to Universal Forwarder on the same machine. To do that, open a TCP port uberAgent can send data to by adding the following to `$SPLUNK_HOME\etc\system\local\inputs.conf` on your Universal Forwarders:

```
[tcp://127.0.0.1:19500]
connection_host = none
sourcetype = dummy
listenOnIPv6 = no
acceptFrom = 127.0.0.1
```

Note: `$SPLUNK_HOME` refers to the base directory of the Splunk (Forwarder) installation, typically `C:\Program Files\SplunkUniversalForwarder`.

If you are deploying the `uberAgent_endpoint` app, port 19500 is opened for you automatically (details).

Disable Universal Forwarder Eventlog Data Collection

A default installation of Universal Forwarder sends all data from the Windows Application, Security, and System event logs to Splunk. If you do not need that, edit `$SPLUNK_HOME\etc\apps\Splunk_TA_windows\local\inputs.conf` so that all stanzas are disabled as in the following example:

```
[WinEventLog://Application]
disabled = 1

[WinEventLog://Security]
disabled = 1

[WinEventLog://System]
disabled = 1
```

Unattended (Silent) Installation

You can find all the information required for automating the deployment of Universal Forwarder here.

Preparation for Imaging/Citrix PVS

If you intend to copy the installation via an imaging method or Citrix PVS, you need to remove instance-specific information such as server name and GUID from the Universal Forwarder installation. To do that, follow these steps right before capturing the image:

- Stop the service `SplunkForwarder` (but leave the start type at `automatic`).
- Open an administrative command prompt.
- Run the command: `C:\Program Files\SplunkUniversalForwarder\bin\splunk clone-prep-clear-config`.
- Prepare the machine for cloning as necessary, but do not reboot.

Installing the Chrome Browser Extension

uberAgent's browser web app performance functionality requires a browser extension to be installed. This article lists the necessary steps.

Enabling the Agent Metric

To enable data collection from Chrome make sure uberAgent's `BrowserPerformanceChrome` metric is enabled in the configuration. In the default configuration, `BrowserPerformanceChrome` is enabled.

Installing the Chrome Extension

> **Note**
>
> As of uberAgent 7.2, the version 4.0 of the Google Chrome browser extension is required. This version is based on a V3 manifest file.

Manual Installation

For tests and PoCs, it may be easiest to install uberAgent's Chrome extension manually from the Chrome Web Store.

Install the extension from the Chrome Web Store.

Automated Deployment

Deployment of uberAgent's Chrome extension is easily possible with Group Policy. Follow the steps in the Chromium Documentation to configure the ExtensionInstallForcelist setting.

To configure Chrome to install the uberAgent extension from the standard Chrome Web Store update URL add the following to the `ExtensionInstallForcelist` setting:

```
jghgedlkcoafeakcaepncnlanjkbinpb;https://clients2.google.com/service/update2/crx
```

Offline Deployment

If your endpoints are not connected to the Internet and can't access Chrome's Web Store, you can also pre-install the extensions. Please contact us for the CRX file or use an online CRX extractor.

Note that using the offline deployment you need to push updated CRX files manually.

Relevant Configuration Settings

Native Messaging

uberAgent's Chrome extension uses native messaging to communicate with the agent. In a default configuration of Chrome, native messaging is not restricted.

If native messaging has been restricted, for example via the NativeMessagingBlocklist Group Policy setting, an exception needs to be defined for the uberAgent extension, for example via the NativeMessagingAllowlist Group Policy setting.

Installing the Firefox Browser Extension

uberAgent's browser web app performance functionality requires a browser extension to be installed. This article lists the necessary steps for the Firefox extension. The requirements can be found here.

Enabling the Agent Metric

To enable data collection from Firefox make sure uberAgent's `BrowserPerformanceFirefox` metric is enabled in the configuration. In the default configuration, `BrowserPerformanceFirefox` is enabled.

Installing the Firefox Extension

> **Note**
>
> As of uberAgent 7.2, the version 4.0.1 of the Firefox browser add-on is required. This version is based on a V3 manifest file.

Manual Installation

For tests and PoCs, it may be easiest to install uberAgent's Firefox extension manually.

Install the extension from the Firefox Web Store.

User Consent The extension version 2.6 or newer initially does not collect any data. **To enable data collection click the uberAgent icon in the upper right corner.**

Automated Installation

Add-on Deployment Deployment of uberAgent's Firefox extension is easily possible with Group Policy (Firefox 60+ required). Please follow the steps in Mozilla's documentation.

Below is an example JSON code to block the user from installing any extension but force the uberAgent extension installation through Group Policy. Put it in `Policies - Administrative Templates - Mozilla - Firefox - Extensions - Extension Management`.

```
{

   "*": {

      "blocked_install_message": "If you need this browser extension,
         please contact the helpdesk.",
       "installation_mode": "blocked"
    }
 ,
   "{
 2e53b2d2-1bdf-446a-8f9d-9be4173886fe }
 ": {

      "installation_mode": "force_installed",
      "install_url": "https://addons.mozilla.org/firefox/downloads/
         latest/uberagent/latest.xpi"
    }

 }
```

User Consent As of 2019-12-02, Mozilla enforces showing a consent dialog to users if an add-on collects information about visited URLs, even if the add-on was deployed through Group Policy.

This change **forces** us to publish a new version of the add-on that initially does not collect any data. To automatically enable data collection set the new managed storage setting `EnableBrowserDataCollection` to **`true`**. This can be achieved in one of two ways.

Automatically declare consent and enable data collection by running a script to set `EnableBrowserDataColle` to **`true`**:

- Make sure that cookies are enabled and Firefox's setting `dom.storage.enabled` is set to **`true`**
- Download and unzip this script
- Run the script with elevated permissions on every endpoint: `powershell.exe -executionpolicy bypass -file Set-uberAgentFFEnableBrowserD .ps1`

- Only upgrade to or install version 2.6+ of uberAgent's Firefox add-on once the script has been run on all endpoints

If you prefer to change things manually instead of using our script we have you covered, too:

1. Create the file `uAFirefoxSettings.json` in uberAgent's installation directory with the following content: `{ "name": "{ 2e53b2d2-1bdf-446a-8f9d-9be4173886fe } ", "description": "ignored", "type": "storage", "data": { "EnableBrowserDataCollection": true } }`
2. Create the registry key `HKEY_LOCAL_MACHINE\SOFTWARE\Mozilla\ManagedStorage\\{ 2e53b2d2-1bdf-446a-8f9d-9be4173886fe }`
3. Set the key's default value to the full path of the JSON file from step one, e.g. `C:\ProgramFiles\vast limits\uberAgent\uAFirefoxSettings.json`

If you do not set `EnableBrowserDataCollection` to `true` before upgrading to or installing version 2.6+ of uberAgent's Firefox extension users have to opt-in at first run. If they do not opt-in, uberAgent cannot collect any website metrics.

Future versions of uberAgent (> 5.3) will set `EnableBrowserDataCollection` to **`true`** automatically unless this is specifically disabled in uberAgent's configuration.

Note that setting `ShowConsentDialog` to **`false`** has the same effect as setting `EnableBrowserDataColle` to **`true`**.

Installing the Edge Browser Extension

uberAgent's browser web app performance functionality requires a browser extension to be installed. This article lists the necessary steps.

Enabling the Agent Metric

To enable data collection from Edge make sure uberAgent's `BrowserPerformanceEdge` metric is enabled in the configuration. In the default configuration, `BrowserPerformanceEdge` is enabled.

Installing the Edge Extension

> **Note**
>
> As of uberAgent 7.2, the version 4.0 of the Google Chrome browser extension is required. This version is based on a V3 manifest file.

Manual Installation

For tests and PoCs, it may be easiest to install uberAgent's Edge extension manually from the Chrome Web Store. Please note that you need to allow extensions from other stores.

Install the extension from the Chrome Web Store.

Automated Deployment

Deployment of uberAgent's Edge extension is easily possible with Group Policy. Follow the steps in the Edge documentation to configure the ExtensionInstallForcelist setting. Extensions on the `ExtensionInstallForcelist` will also get installed if the Edge setting `Allow extensions from other stores` is not enabled.

To configure Edge to install the uberAgent extension from the standard Chrome Web Store update URL add the following to the `ExtensionInstallForcelist` setting:

```
jghgedlkcoafeakcaepncnlanjkbinpb;https://clients2.google.com/service/update2/crx
```

Offline Deployment

If your endpoints are not connected to the Internet and can't access Chrome's Web Store, you can also pre-install the extensions. Please contact us for the CRX file or use an online CRX extractor.

Note that using the offline deployment you need to push updated CRX files manually.

Relevant Configuration Settings

Native Messaging

uberAgent's Edge extension uses native messaging to communicate with the agent. In a default configuration of Edge, native messaging is not restricted.

If native messaging has been restricted, for example via the NativeMessagingBlocklist Group Policy setting, an exception needs to be defined for the uberAgent extension, for example via the NativeMessagingAllowlist Group Policy setting.

Installing the Citrix Enterprise Browser Extension

uberAgent's browser web app performance functionality requires a browser extension to be installed. This article lists the necessary steps.

Enabling the Agent Metric

To enable data collection from Citrix Enterprise Browser (CEB) make sure uberAgent's `BrowserPerformanceCE` metric is enabled in the configuration. In the default configuration, `BrowserPerformanceCEB` is enabled.

Installing the CEB Extension

> **Note**
>
> As of uberAgent 7.2, the version 4.0 of the Google Chrome browser extension is required. This version is based on a V3 manifest file.

Manual Installation

CEB is managed through Global App Configuration service (GACS). To let users manually install the extension or to test the extension in a proof of concept, one can add the extension to the allow list. The extension then appears as available in the extension manager.

Add the ID `jghgedlkcoafeakcaepncnlanjkbinpb` to the ExtensionInstallAllowlist.

Automated Deployment

Deployment of uberAgent's CEB extension is easily possible with GACS. Follow the steps in the CEB documentation to configure the ExtensionInstallForcelist setting.

Add the ID `jghgedlkcoafeakcaepncnlanjkbinpb` to the ExtensionInstallForcelist.

Offline Deployment

An offline deployment is not supported as CEB is managed by GACS, a cloud service hosted by Citrix.

Installing the Internet Explorer Browser Add-on

uberAgent's browser web app performance functionality requires a browser add-on (aka extension) to be installed. This article lists the necessary steps.

Enabling and Configuring the Agent Metric

To enable data collection from Internet Explorer make sure uberAgent's `BrowserPerformanceIE` metric is enabled in the configuration. In the default configuration, `BrowserPerformanceIE` is enabled.

You can optionally disable the collection of performance data from embedded frames via the registry. The following setting is valid for 32-bit and 64-bit Windows systems:

```
Hive:        HKEY_LOCAL_MACHINE
Key:         SOFTWARE\vast limits\uberAgent
Value name:  IEAddonIgnoreFrames
Value type:  DWORD
Value data:  0 (default) - Collect performance data from each page
    including embedded frames
             1           - Do not collect performance data from embedded
                 frames
```

Installing the IE Add-on

The add-on's files are part of the uberAgent setup and placed in the installation directory automatically.

Note: the add-on is **not registered automatically** to prevent prompts asking the end-users for confirmation.

Registering the IE Add-on

Manual Registration

For tests and PoCs, it may be easiest to register uberAgent's Internet Explorer add-on manually. The installation directory contains the file `Register-uAIEExtension.cmd`. Run this file with elevated rights to register uberAgent's Internet Explorer add-on DLLs.

Alternatively, run the following commands with elevated rights:

```
regsvr32 /s "%ProgramFiles%\vast limits\uberAgent\uAIEExtension32.dll"
regsvr32 /s "%ProgramFiles%\vast limits\uberAgent\uAIEExtension64.dll"
```

Automated Registration - Software Deployment Tool

The automated registration of uberAgent's Internet Explorer add-on is easily possible with your software deployment tool of choice. Make sure to register the two DLLs using the regsvr32 commands (see above) after uberAgent is installed.

Automated Registration - uberAgent Service

uberAgent's Internet Explorer add-on can be registered automatically through uberAgent's system service. Please note that this does not happen by default to prevent end-user prompts (see below).

To configure the uberAgent service to automatically register the IE add-on set the `RegisterIEAddOn` configuration value to `1`.

Enabling the IE Add-on

Enable Manually

Internet Explorer will ask each user to enable the add-on once the DLLs are registered when the browser is started.

Enable Automatically

To avoid end-user prompts, configure the Group Policy setting `Add-on List` for automatic enablement. The setting can be found at **Computer Configuration** > **Policies** > **Administrative Templates** > **Windows Components** > **Internet Explorer** > **Security Features** > **Add-on Management** > **Add-on List**.

Add an entry with the following values:

```
Value name: {
 82004312-5B53-46F1-B179-4FCE28048E6F }

Value data: 1
```

Relevant Configuration Settings

Protected Mode

The uberAgent Internet Explorer add-on has been tested successfully and is supported in protected mode.

Enhanced Protected Mode

The uberAgent Internet Explorer add-on has been tested successfully and is supported in enhanced protected mode.

Enterprise Mode

Internet Explorer enterprise mode allows customers to define URLs that are rendered with older versions of the IE engine (e.g., IE9 or IE10). uberAgent's Internet Explorer add-on requires IE11 and only works for sites rendered with version 11 of the IE engine.

Upgrading uberAgent

This document outlines the procedure for upgrading uberAgent to a newer version.

> **Note**
>
> **Securing the Configuration Directory**
>
> Please review this document for important information on securing the agent's `%ProgramData%` configuration directory.

Understand What's New and Changed

Please read the changelog. Changes might affect uberAgent's configuration, for example.

Depending on the nature of the changes made for a new version, it may be necessary to update custom apps that make use of the data collected by uberAgent.

Endpoint Agents

To upgrade uberAgent's endpoint agents, run the installer for the newer version. It detects the existing installation, uninstalls it, and installs the updated version automatically. Please refer to the endpoint agent installation documentation for more details (Windows, macOS).

Note: Any custom configuration files in the installation directory are not preserved during the upgrade. As of uberAgent 7.3, configuration files must be stored outside the installation directory in `%ProgramData%`, as described here.

User & Host Tags

Starting with uberAgent 7.0, host and user tags are collected through dedicated timers. If you are upgrading from an earlier version, adjust your configuration to implement a timer for user and host tags as described here.

Upgrading to uberAgent 7.3.1

Upgrading to uberAgent 7.3.1 follows the standard installation instructions. However, there are important changes regarding the configuration directory for Windows. Please note the following when upgrading:

- **Configuration Directory Change**: Starting with uberAgent 7.3, configuration files can no longer be deployed in the installation folder due to security concerns.

  Beginning with version 7.3, uberAgent only searches for configuration files in the `%ProgramData%\vast limits\uberAgent\Configuration\` directory. Customers who previously stored their configurations in the installation folder need to move them to `%ProgramData%\vast limits\uberAgent\Configuration\`. Please refer to the endpoint agent installation documentation for more details (Windows).

- **Template Configuration Source Change**: Starting with uberAgent 7.3.1, the agent installer does not store the template configuration files on disk any more. You can download the latest template config files for your installed uberAgent version on GitHub.
- **License File**: While configuration files must be deployed to `%ProgramData%`, the license file should still be placed in the installation folder.
- **GPO and Central Configuration File Management**: uberAgent can continue to be configured using Group Policy Objects (GPO) or Central Configuration File Management. Ensure that these methods reflect the new `%ProgramData%` directory requirement.
- **Securing the Configuration Directory**: Please review this document for important information on securing the agent's `%ProgramData%` configuration directory.

Upgrade Steps to uberAgent 7.3.1

1. **Install the Update**: Run the uberAgent 7.3.1 installer. It automatically removes any previous installations and set the correct permissions for the configuration folder in `%ProgramData%`.
2. **Dowload Configuration from GitHub**: Download the latest template config files for your installed uberAgent version on GitHub.
3. **Deploy Configuration**: After installation, move your configuration files to `%ProgramData%`.

4. **Deploy License**: Place the license file in the installation folder, as this remains the default location in uberAgent 7.3.

Splunk Apps

Upgrade Procedure

1. **Delete** the following Splunk app directories:

   ```
   $SPLUNK_HOME\etc\apps\uberAgent
   $SPLUNK_HOME\etc\apps\uberAgent_ESA
   $SPLUNK_HOME\etc\apps\uberAgent_indexer
   ```

2. **Install** the new versions of the Splunk apps as described here.

3. **Restart** Splunk.

Upgrade from uberAgent 6.0

With uberAgent 6.1, the experience score data was moved from the KV store to a dedicated index. If you're upgrading from 6.0 to a newer version, delete the Splunk KV store lookup `lookup_hostinfo2` and associated data by running the following Splunk searches:

```
| outputlookup lookup_hostinfo2

| outputlookup lookup_score_per_machine

| outputlookup lookup_score_historic_per_machine

| outputlookup lookup_score_per_session

| outputlookup lookup_score_historic_per_session

| outputlookup lookup_score_per_application

| outputlookup lookup_score_historic_per_application
```

Elasticsearch

New versions of uberAgent often include additional fields, which may require updating the Elasticsearch index template. To do so, overwrite the existing index template with the new version as described here. Note that index templates are applied only to new indexes; existing indexes retain their original templates.

Configuration via Group Policy

Platforms: Windows only

Configuration via Group Policy is an alternative to configuring uberAgent via central config file management or local config files.

Overview

To configure uberAgent via Group Policy, follow these steps:

- Copy the Group Policy AMDL/ADMX templates included in the uberAgent download package to your `PolicyDefinitions` directory.
- Create a new GPO that applies to the computers uberAgent is installed on.
- Import the default configuration from the directory `uberAgent components\Group Policy\GPO backup` of the download package, choosing either the default configuration or the version optimized for data volume.
- Adjust the imported default configuration as required.
- Local configuration files on the endpoints are ignored, and policy settings are used exclusively. If an additional configuration file is set via Group Policy, the settings of that file overwrite Group Policy settings.

Detailed Description

Preparation

In order to have the uberAgent settings show up in Group Policy Management Editor, you need to add uberAgent's Group Policy template (ADMX and ADML files) to your `PolicyDefinitions` directory. If you are using a Central Store for Group Policy administrative template files, then the location of the `PolicyDefinitions` directory is:

`\\FQDN\SYSVOL\FQDN\policies\PolicyDefinitions`

If you are not using a Central Store, administrative templates reside on the machines that are used to manage Group Policy in:

`%Systemroot%\PolicyDefinitions`

In any case, copy all files from this location in uberAgent's download package to your `PolicyDefinitions` directory:

`uberAgent components\Group Policy\Administrative template (ADMX)`

That directory should contain at least:

- `uberAgent.admx`
- `en\uberAgent.adml`

Configuration

Once the administrative template files are in the correct location, choose a Group Policy Object (GPO) you want to configure uberAgent's settings in. uberAgent's configuration settings are located in **Computer Configuration** > **Policies** > **Administrative Templates** > **uberAgent**.

Settings Import The easiest way to get a working configuration is to import the settings from the GPO backup included in the uberAgent download package.

To import the default configuration, follow these steps:

- In Group Policy Management Console (GPMC), navigate to the node **Group Policy Objects** (import is not available on a GPO link).
- Right-click the GPO you want to import the settings into.
- Select **Import Settings...**.
- When asked for the **Backup location** from which settings will be imported select the directory `uberAgent components\Group Policy\GPO backup` from uberAgent's download package.
- Complete the **Import Settings** wizard.
- Open the GPO you imported the settings into in Group Policy Management Editor.
- Navigate to **Computer Configuration** > **Policies** > **Administrative Templates** > **uberAgent**> **Receivers** > **Receiver 01**.
- Adjust the setting **Servers (required)** according to your requirements.

Manual Configuration As an alternative to importing the default configuration (as described above), you can configure uberAgent's settings from scratch. Here are some recommendations:

- Enable on-demand metrics: **On-demand metrics** > **On-demand metrics**.
- Configure and enable a receiver: **Receivers** > **Receiver 01**.
- Configure and enable at least one timer: **Timers** > **Timer 01**.
- Do not forget to enable the setting **Configuration through Group Policy**.

Troubleshooting

Verify in the registry that the machines running uberAgent have received the correct Group Policy settings. uberAgent's settings are stored in `HKEY_LOCAL_MACHINE\SOFTWARE\Policies\vast limits`.

Verify that uberAgent is processing the settings correctly:

- Restart the uberAgent service.
- Open the log file.
- Right after the service starts uberAgent logs in detail how it processes its configuration and what source (Group Policy, config file) is being used.

Central Config File Management

The ability to retrieve configuration sets from an SMB file share is called Central Config File Management (CCFM). It has the advantage of **automatic deployment** via an agent-based pull mechanism and full support for uberAgent ESA Threat Detection rules and inventory tests (which cannot be distributed by Group Policy alone). With Central Config File Management, either a single configuration can be used for all endpoints, or different types of agents can be configured to pull different configurations.

Central Config File Management is an alternative to configuring uberAgent via local configuration files or Group Policy.

Requirements

File Share Access

uberAgent must have read permissions on the file share to copy the central configuration to the local cache.

Windows The uberAgent service runs as `SYSTEM`. Hence, in an Active Directory domain environment, the computer account must have read permissions on the central file share.

macOS Mount the file share as SMB share.

Azure Files You can use Azure Files to store the central configuration in the same way that you can use it for the license file.

Availability Requirements

There is no need for the configured file share to be available all of the time. Endpoints always use the local configuration cache.

Enabling CCFM

If a base path is specified via one of the methods described in this section, Central Config File Management is enabled.

Windows

uberAgent checks the following registry values at startup. If one of them exists, the agent uses the specified value as the base path to the central configuration file share.

Via Group Policy Specify the base path to the configuration file share in the registry value `HKLM\Software\Policies\vast limits\uberAgent\Config\ConfigFilePath` (`REG_SZ`). This setting can be deployed via uberAgent's Group Policy settings.

Via the Registry Specify the base path to the configuration file share in the registry value `HKLM\Software\vast limits\uberAgent\Config\ConfigFilePath` (`REG_SZ`).

macOS

1. Add the path to the config file share in the file `/Library/Application Support/uberAgent/Config Templates/uberAgent-remote-config-macOS.conf`:
   Section: `RemoteConfiguration`
   Key: `ConfigFilePath`
2. Copy the updated file to `/Library/Application Support/uberAgent`.

How CCFM Works

If Central Config File Management is enabled, uberAgent tries to retrieve its configuration from the file share specified in the `ConfigFilePath` setting. It does so by looking for the configuration matching its own version the closest.

If a central configuration is found, and its timestamp differs from the previously cached configuration on the endpoint, uberAgent downloads the central configuration. It then compares the currently applied configuration with the downloaded central configuration for meaningful changes. If any are detected, the newly downloaded central configuration is applied.

Configuration Archive (uberAgent.uAConfig)

uberAgent expects the central configuration in a single file, `uberAgent.uAConfig`. Technically, this file is a zipped archive of all of uberAgent's configuration files.

Base Path Subdirectory Precedence

Assuming the agent's version is `7.1.0.5050` and the file share path is `\\server\share`, uberAgent looks in the following directories for the configuration archive `uberAgent.uAConfig` and uses the first config archive it finds:

1. `\\server\share\7.1.0.5050`
2. `\\server\share\7.1.0`
3. `\\server\share\7.1`
4. `\\server\share\7`
5. `\\server\share`

CCFM Application Matrix

The following matrix helps to understand how Central Config File Management updates the local agent configuration.

File share accessible	Central configuration found	Local cache valid	Meaningful change detected	uberAgent behavior
yes	yes	no	n/a	The configuration is downloaded and applied; the local cache folder is updated.
no	n/a	no	n/a	uberAgent pauses for 60 s, after which it retries, repeating this until `uberAgent.uAConfig` is found.

File share accessible	Central configuration found	Local cache valid	Meaningful change detected	uberAgent behavior
yes	yes	yes	yes	The configuration is downloaded and applied; the local cache folder is updated.
yes	no	yes	n/a	uberAgent uses the cached configuration.
no	n/a	yes	n/a	uberAgent uses the cached configuration.

Refresh Interval & Auto-Update

Each agent regularly checks the CCFM file share for updates and applies new configurations automatically. See this document for details.

Creating or Updating a Custom CCFM Archive (uberAgent.uAConfig)

Note: the CCFM archive of the default configuration can always be found in the `config-dist` subdirectory of the uberAgent Configuration GitHub repository. Creating your own CCFM archive is only necessary if **customizations** are to be applied to the default configuration.

Downloading the Latest Files from GitHub

We provide a PowerShell script to help automate the process of downloading a subset of uberAgent' s configuration files from GitHub.

With the help of this script, download those files that you want to update, leaving your customized config files unchanged. Then proceed to create the CCFM archive as explained below.

Creating the CCFM Archive (uberAgent.uAConfig)

Create an archive of all uberAgent configuration files you want to be part of your central configuration and name it `uberAgent.uAConfig`. This can either be done in Windows Explorer, macOS Finder, or on the command line. The steps to produce a configuration archive on the terminal are as follows:

1. Copy all configuration files into a folder.
2. Change to this folder on the command line.
3. Execute one of the following commands, depending on the platform you are creating the archive on.

Windows

```
Compress-Archive * -DestinationPath config-temp.zip
Move-Item config-temp.zip uberAgent.uAConfig
```

macOS

```
zip -r uberAgent.uAConfig ./ -x "*.DS_Store"
```

Switching Between Configuration Options (CCFM ↔ Local ↔ Group Policy)

Switching from one configuration option to another at runtime is possible anytime.

From Central to Local Configuration

Windows Remove the value `ConfigFilePath` where you configured it (registry or Group Policy, see above).

macOS Remove the file `uberAgent-remote-config-macOS.conf` from the folder `/Library/Application Support/uberAgent`.

From Local to Central Configuration

Windows Add the value `ConfigFilePath` either via the registry or via Group Policy (see above).

macOS Add the path to the file share in file `uberAgent-remote-config-macOS.conf` (see above).

CCFM Reporting & Monitoring

uberAgent reports configuration metadata in the sourcetype `uberAgent:Config:ConfigInfo` (docs). The transmitted data is visualized in the **uberAgent Versions** Splunk dashboard.

Configuration via Local Config Files

Platforms: all

Configuration via local config files is an alternative to configuring uberAgent via central config file management or Group Policy.

Config File Location

Windows

Config File Templates Starting with uberAgent 7.3.1, the agent installer does not store the template configuration files on disk any more. You can download the latest template config files for your installed uberAgent version on GitHub.

As the agent searches for valid configurations only in the `%PROGRAMDATA%\vast limits\uberAgent\Configuration\` directory, perform the following steps:

1. Download the config files from GitHub to `%PROGRAMDATA%\vast limits\uberAgent\Configuration\`
2. Modify the `%PROGRAMDATA%\vast limits\uberAgent\Configuration\uberAgent.conf` file.

Configuration Optimized for Data Volume To switch to the configuration optimized for data volume back up `uberAgent.conf` and rename `uberAgent-data-volume-optimized.conf` to `uberAgent.conf`.

macOS

The configuration file `uberAgent.conf` for the uberAgent daemon is located in `/Library/Application Support/uberAgent/`. During upgrades, files in this directory are not overwritten to preserve customizations. It is recommended to update the default configuration from the config file templates (see below) after an upgrade.

Config File Templates Starting with uberAgent 7.3.1, the agent installer does not store the template configuration files on disk any more. You can download the latest template config files for your installed uberAgent version on GitHub.

Applying Configuration Changes

The agent reads the configuration upon startup of the `uberAgent` service/daemon. To apply configuration changes, restart the `uberAgent` service/daemon.

Config File Syntax

uberAgent's configuration file supports several advanced syntax features.

Windows

PATH_REGEX

`PATH_REGEX` stands for a **regex** in which **environment variables** can be used. It is matched against a **file system** path. The following rules apply:

- Environment variables (enclosed in percent signs, e.g., `%ProgramFiles%`) are expanded first, regex matching happens second.
- Paths are specified as regular expressions.
- Backslashes must be escaped by prepending a second backslash.
- Matching is case-insensitive.
- Specifying only the file name without the full path works but is not recommended.

macOS

PATH_REGEX

Due to macOS security policies, environment variables for daemons must be declared within the daemon's property list file. As future macOS releases will mandate that the plist resides within the application bundle, enforcing code signature integrity through directory hash validation, any post-installation modification to the plist, such as dynamic environment variable expansion, would disrupt the code signature, rendering the daemon inoperable. Consequently, support for environment variable expansion under macOS is not supported.

Agents on All Operating Systems

ConfigInclude

A `@ConfigInclude` directive includes another configuration file. The other file is identified by name (without path) and must reside in the same directory as the configuration file including it. The additional `platform` parameter may be used to conditionally include a file on the selected platform (`Windows` or `macOS`). Platform matching is case-insensitive.

Examples:

```
#Include on all platforms
@ConfigInclude other-file.conf
```

```
# Include on Windows only
@ConfigInclude windows-specific.conf platform=Windows
```

PROCESS_NAME_REGEX

`PROCESS_NAME_REGEX` stands for a **regex** that is matched against a **process name**. The following rules apply:

- Backslashes must be escaped by prepending a second backslash.
- Matching is case-insensitive.
- Only the file name is used for matching, not the full path.

URL_REGEX

`URL_REGEX` stands for a **regex** that is matched against a **URL**. The following rules apply:

- Backslashes must be escaped by prepending a second backslash.
- Matching is case-sensitive.

Format of the URLs matched against `URL_REGEX`:

- Origin and path (without query segment), e.g.: `https://uberagent.com/download/`
- Port numbers are stripped from the URL if they match the default port number.

platform=

All configuration stanzas can optionally be limited to a specific platform using the `platform=` attribute.

- The platform is either **Windows** or **macOS**
- Stanzas without `platform=` are active on all supported platforms.

Example

```
[Timer platform=Windows]
...

[Timer platform=macOS]
...
```

Change configured setting values dynamically with uAQL

Some configuration settings are changeable using prefixed uAQL queries. This allows use cases where configuration values depend on environment variables, registry values or require string prefixes/suffixes. For now, the following configuration settings are changeable:

[Receiver]

- RESTToken
- TLSClientCertificate
- ElasticIngestPipeline
- Index
- Host
- Source
- PersistentOutputQueuePathWindows
- PersistentOutputQueuePathMacOS
- KafkaTopicName

Consider the following **example** to configure a prefix string in any *KafkaTopicName* setting.

```
KafkaTopicName = uaql(set(KafkaTopicName, concat('vastlimits_',
    KafkaTopicName)))
```

UXM Features & Configuration

uberAgent uses sophisticated methods to collect **relevant data** from the operating system and applications. It then presents the data in an **easily understandable** and human-friendly way to give IT the information they actually need to improve **digital employee experience** and maintain **security**.

This section lists important features with their respective configuration options.

Experience Score

uberAgent's experience score is a rating system that calculates and visualizes the current and past status of all applications, devices, and user sessions monitored by uberAgent.

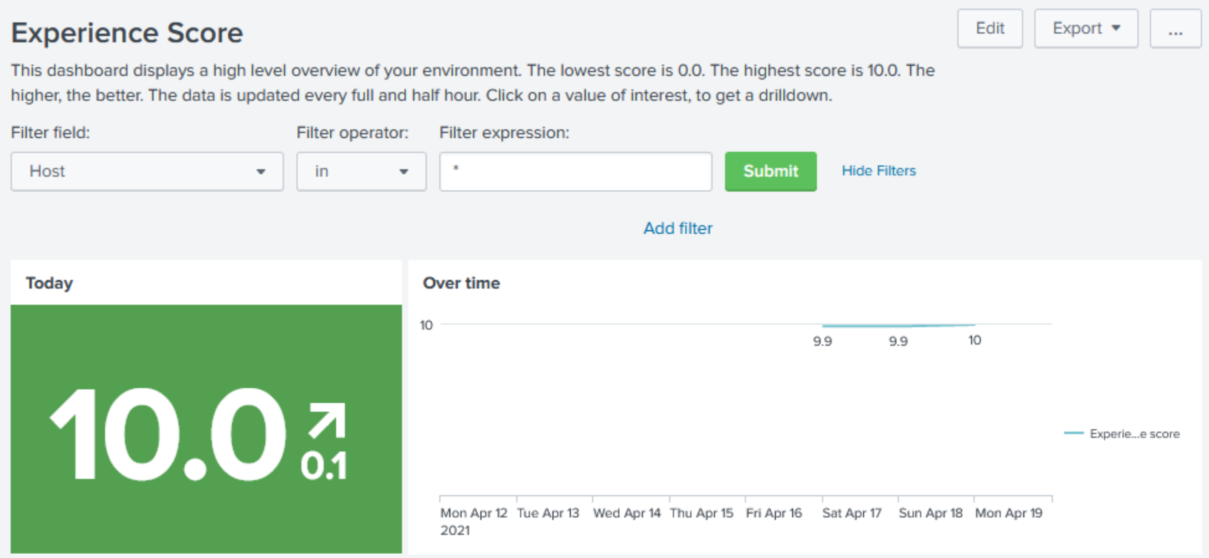
Experience scores range **from zero to ten**. Scores from zero to four are considered **bad**, scores from four to seven are considered **medium**, and scores from seven to ten are considered **good**.

Experience Score Dashboard

The experience score dashboard is the entry point of the uberAgent UXM Splunk app. It visualizes experience scores for the entire estate, breaking the data down by category and component, highlighting components where potential issues originate from.

The dashboard also provides quick access to important **KPIs** like logon duration, application responsiveness, or application errors.

Overall Score



Experience Score

This dashboard displays a high level overview of your environment. The lowest score is 0.0. The highest score is 10.0. The higher, the better. The data is updated every full and half hour. Click on a value of interest, to get a drilldown.

Filter field: Host

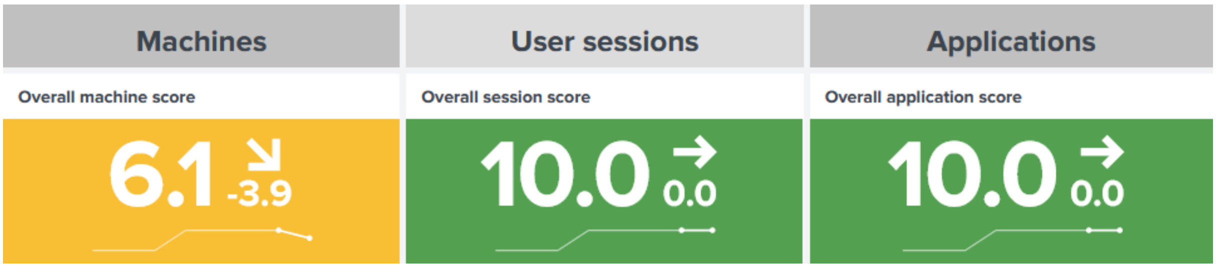
Filter operator: in

Filter expression: *

The filter at the top allows filtering the dashboard to a subset of machines. By default, all machines are shown.

On the left in the second row, one can see the overall score and the trend compared to yesterday. On the right, the score development over time is visualized.

Machine, User Session, and Application Scores

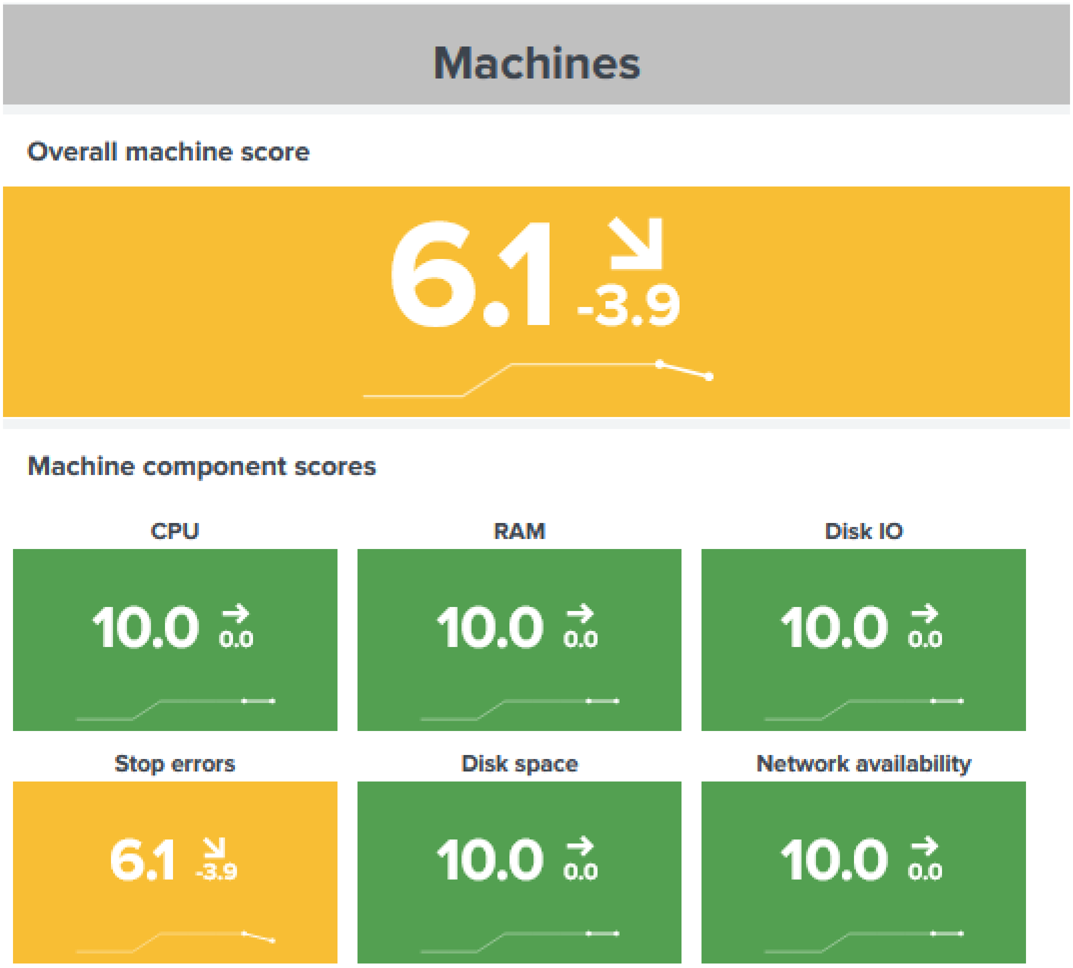


The overall score derives from three categories:

- **Machine score**: quality indicator for machine performance and health
- **User session score**: quality indicator for user session performance and health
- **Application score**: quality indicator for application performance and health

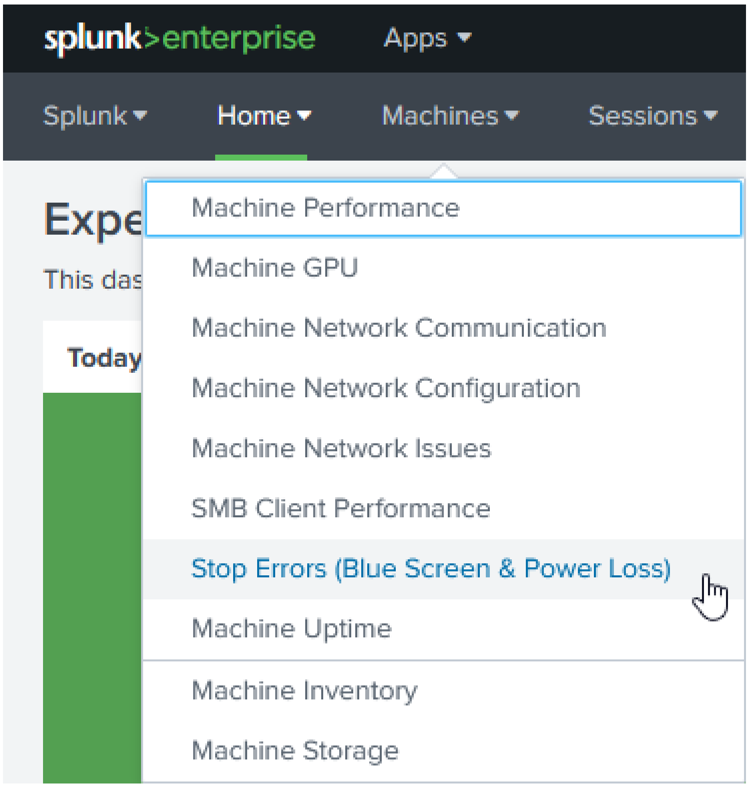
The charts show a trend indicator for the last day as well as a sparkline for the last seven days.

Score Components



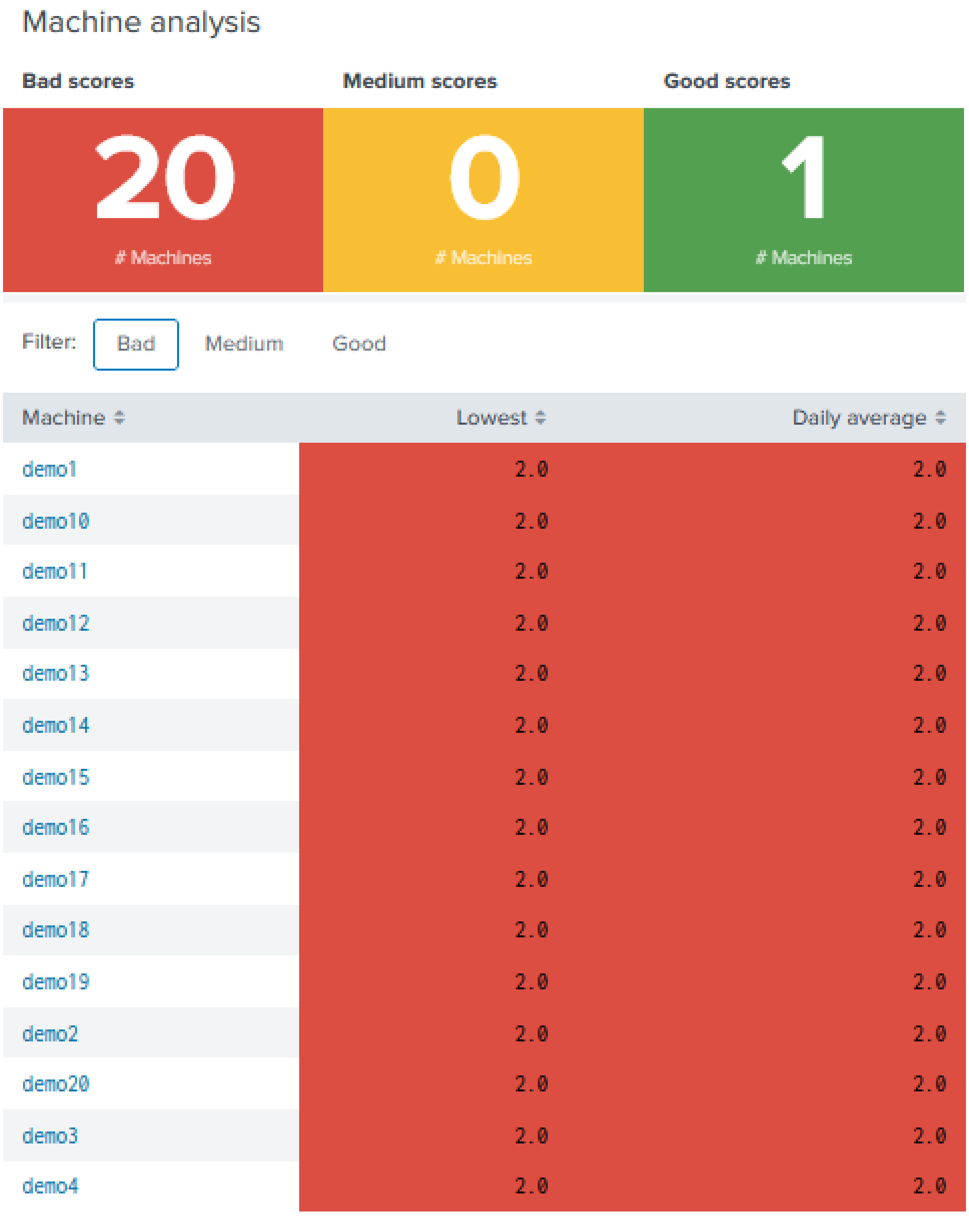
Each category is calculated by different components. Components differ per category. For example, *Stop errors* is a component solely for the machine category, while the *Protocol latency* component is only part of the user session category. There are also common components, like *CPU* or *RAM*.

The categories allow us to see issues in an environment, and the components unveil the cause (or causes). In the screenshot above, the low machine score is caused by a lot of stop errors (BSODs).



In that case, checking the *Stop Errors (Blue Screen & Power Loss)* dashboard in the *Machine* menu shows the problematic machines.

Analyzing Individual Machines, User Sessions, and Applications



Machine analysis

Bad scores: 20 # Machines

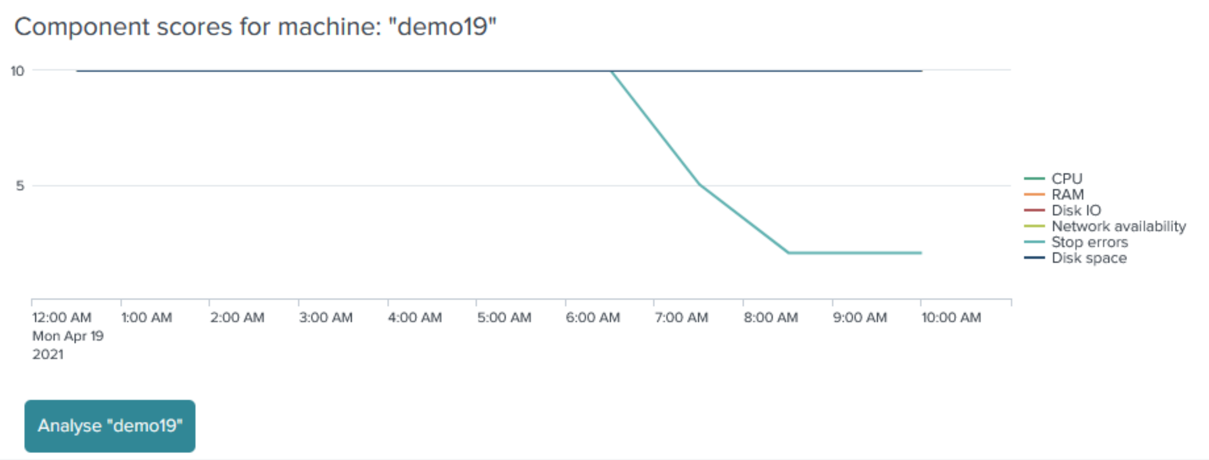
Medium scores: 0 # Machines

Good scores: 1 # Machines

Filter: Bad Medium Good

Machine	Lowest	Daily average
demo1	2.0	2.0
demo10	2.0	2.0
demo11	2.0	2.0
demo12	2.0	2.0
demo13	2.0	2.0
demo14	2.0	2.0
demo15	2.0	2.0
demo16	2.0	2.0
demo17	2.0	2.0
demo18	2.0	2.0
demo19	2.0	2.0
demo2	2.0	2.0
demo20	2.0	2.0
demo3	2.0	2.0
demo4	2.0	2.0

The charts below show scores for individual machines, user sessions, and applications for today. Items with bad (red) or medium (yellow) scores may need attention the most. Click on an item in the table to get a drilldown.



A new chart opens showing the components over time for that item. That allows us to see when the issue or the issues started. To analyze the item in detail, click on the **analyze button**, which redirects to a new page.

More Details

Scores might not be enough to get an overview. One may want to see real numbers, for example, login times, to get a better understanding of the performance.

Click on the plus sign next to the *More details* title to reveal charts with more details. Click an item of interest to get a drilldown.

Score Calculation

Component scores are evaluated at every full and half hour for the past 30 minutes. Calculations span a time interval of three minutes, resulting in 10 sections (30 minutes/3 minutes = 10). If a section is above a threshold, a threshold counter is incremented.

Each score has two thresholds. One for low severity, the other for high severity. Each threshold has a weight.

A score is calculated as follows: `10 - (Low severity threshold counter x low severity threshold weight + high severity threshold counter x high severity threshold weight)`

- **Example 1:** Three sections above the low severity threshold as well as a weight of 0.5. The score would be: `10 - (3 x 0.5)= 8.5`

- **Example 2:** Three sections above the low severity threshold (weight = 0.5) and two sections above the high severity threshold (weight = 1). The score would be: `10 - (3 x 0.5 + 2 x 1)= 6.5`

Note: The higher the weight, the lower the score.

Below is a list of default thresholds and weights. To modify the defaults, see Modifying the Score Calculation.

Machine

Threshold	Setting	Default value	Unit	Default weight
CPU usage. Low severity.	ThresholdMachineCPUPercentLowerBound	80	%	0.5
CPU usage. High severity.	ThresholdMachineCPUPercentHigherBound	90	%	1
RAM usage. Low severity.	ThresholdMachineRAMPercentLowerBound	80	%	0.5
RAM usage. High severity.	ThresholdMachineRAMPercentHigherBound	90	%	1
DIsk IO usage. Low severity.	ThresholdMachineIOPercentLowerBound	80	%	0.5
Disk IO usage. High severity.	ThresholdMachineIOPercentHigherBound	90	%	1
Stop errors. Low severity.	ThresholdStopErrorCountLowerBound	1	Count	0.7
Stop errors. High severity.	ThresholdStopErrorCountHigherBound	2	Count	1
Disk usage. Low severity.	ThresholdMachineDiskUsagePercentLowerBound	80	%	0.2
Disk usage. High severity.	ThresholdMachineDiskUsagePercentHigherBound	90	%	0.5
Network availability. Low severity. *Note: higher is better*	ThresholdMachineNetworkAvailabilityPercentLowerBound	95	%	0.2

Threshold	Setting	Default value	Unit	Default weight
Network availability. High severity. *Note: higher is better*	ThresholdMachineNetworkAvailabilityPercentHigherBound	90	%	0.5

User session

Threshold	Setting	Default value	Unit	Default weight
CPU usage. Low severity.	ThresholdSessionCPUPercentLowerBound	80	%	0.5
CPU usage. High severity.	ThresholdSessionCPUPercentHigherBound	90	%	1
RAM usage. Low severity.	ThresholdSessionRAMPercentLowerBound	80	%	0.5
RAM usage. High severity.	ThresholdSessionRAMPercentHigherBound	90	%	1
Disk IO latency. Low severity.	ThresholdIOLatencyLowerBound	20	ms	0.5
Disk IO latency. High severity.	ThresholdIOLatencyHigherBound	30	ms	0.7
Logon duration. Low severity.	ThresholdLogonDurationLowerBound	30	s	0.2
Logon duration. High severity.	ThresholdLogonDurationHigherBound	60	s	0.4
Protocol latency. Low severity.	ThresholdSessionRpLatencyMsLowerBound	100	ms	0.2
Protocol latency. High severity.	ThresholdSessionRpLatencyMsHigherBound	200	ms	0.5

Application

Threshold	Setting	Default value	Unit	Default weight
CPU usage. Low severity.	ThresholdAppCPUPercentLowerBound	80	%	0.5
CPU usage. High severity.	ThresholdAppCPUPercentHigherBound	90	%	1
RAM usage. Low severity.	ThresholdAppRAMMBLowerBound	1024	MB	0.1
RAM usage. High severity.	ThresholdAppRAMMBHigherBound	2048	MB	0.3
Disk IO. Low severity.	ThresholdAppIOCountLowerBound	200	Count	0.1
Disk IO. High severity.	ThresholdAppIOCountHigherBound	400	Count	0.3
Network availability. Low severity. *Note: higher is better*	ThresholdAppNetworkAvailabilityPercentLowerBound	95	%	0.2
Network availability. High severity. *Note: higher is better*	ThresholdAppNetworkAvailabilityPercentHigherBound	90	%	0.5
Network latency. Low severity.	ThresholdAppSendLatencyMsLowerBound	100	ms	0.2
Network latency. High severity.	ThresholdAppSendLatencyMsHigherBound	300	ms	0.5
Application UI delay. Low severity.	ThresholdAppUIDelaySLowerBound	5	s	0.2
Application UI delay. High severity.	ThresholdAppUIDelaySHigherBound	10	s	0.5
Application errors. Low severity.	ThresholdApplicationErrorCountLowerBound	1	Count	0.5

Threshold	Setting	Default value	Unit	Default weight
Application errors. High severity.	ThresholdApplicationErrorCountHigherBound	2	Count	1

Modifying the Score Calculation

The scores'default calculation settings have been chosen in a careful selection and tuning process. They should work well in most environments. Please read on if you find that the calculations should be optimized for your organization's requirements.

Before Modifying

Before making changes, note the following:

- The lowest weight possible is 0.
- The highest weight possible is 1.
- The sum of all weights doesn't need to be 1. Each component is calculated separately.
- All components together form a total machine/user session/application score. The total score is always equal to the lowest component score.

Modifying

To modify the score calculation, change the following three input lookup files in `$SPLUNK_HOME/etc/apps/uberAgent/lookups`. See Score Calculation for calculations and settings.

- **Machine:** `score_machine_configuration.csv`
- **User session:** `score_session_configuration.csv`
- **Application:** `score_application_configuration.csv`

After Modifiying

- Distribute the changed input lookup files to all search heads
- It is best to delete all previous scores as they cannot be compared to the new ones. See Deleting Scores for instructions.

New versions of uberAgent may introduce new scores or changes to calculations for existing scores. Hence your score modifications will be overridden when updating uberAgent.

Score Storage

Scores are stored in the Splunk index `score_uberagent_uxm`. The scores for the current date get aggregated at midnight (average per day) and then stored in the index as well.

If you want to delete the scores, see Deleting Scores.

Deleting Scores

Scores are stored in the Splunk index `score_uberagent_uxm` and can be deleted via the CLI.

```
splunk clean eventdata -index score_uberagent_uxm
```

Troubleshooting

If the Experience Score dashboard stays empty, try the following troubleshooting steps.

1. Scores are stored in an additional Splunk index `score_uberagent_uxm` since version 6.1. Check whether the index exists.
2. Scores are calculated through saved searches running on Splunk search heads. In a distributed Splunk environment, configure the search heads to forward data to the indexer layer.

Per-Application Network Monitoring

This feature is supported on: Windows, macOS

What Is Per-Application Network Monitoring

uberAgent network monitoring keeps track of all incoming and outgoing network connections. uberAgent associates every network connection with the application handling it and determines metrics like **latency, packet loss, data volume**, and more. uberAgent network monitoring also records **failed connections** that may have been blocked by firewalls, for example.

uberAgent per-application network monitoring does not inspect packets and does not break TLS or other types of encryption.

Use Cases for Per-Application Network Monitoring

By providing insights into network activity per application, uberAgent enables the following use cases:

- Network **availability** scoring
- Network **quality** monitoring
- Identification of network **issues** (jitter, packet loss, blocked ports)
- Mapping of network **targets** (who talks to whom)
- **Data volume** analytics (how much traffic is going where)

Which Data is Collected by Per-Application Network Monitoring

The data collected as part of uberAgent per-application network monitoring is sent to the backend via the sourcetypes listed in the network metrics documentation.

Name Resolution (IP Address to DNS Name)

Name resolution is supported on: Windows

Network packets contain IP addresses, but not DNS names. In order to be able to associate each network target IP address with the corresponding DNS name, uberAgent also monitors DNS queries. By enriching network monitoring data with DNS query data, uberAgent ensures that each network event has the target's IP address as well as its DNS name.

Please note that uberAgent does not perform reverse DNS lookups, nor does it send its own DNS queries over the wire. uberAgent only generates network traffic to send the collected data from the endpoint to the backend servers.

Protocol Notes

IPv6

IPv6 is fully supported by uberAgent per-application network monitoring.

UDP

UDP traffic is fully supported by uberAgent per-application network monitoring. However, due to the protocol's connectionless nature, latency cannot be determined.

TCP

TCP traffic is fully supported by uberAgent per-application network monitoring. Since TCP is a connection-based protocol, latency, jitter, and packet loss can be determined.

Please note that latency detection is limited to send operations because otherwise a cooperating agent at the receiving side would be required.

QUIC

QUIC traffic is treated as UDP by uberAgent per-application network monitoring.

ICMP

ICMP traffic is ignored by uberAgent per-application network monitoring.

Configuration

Enabling or Disabling Per-Application Network Monitoring

uberAgent per-application network monitoring is enabled or disabled via the `NetworkTargetPerformancePr` timer metric in the configuration. In the default configuration, network monitoring is enabled.

Configuration Options for Per-Application Network Monitoring

Certain aspects of per-application network monitoring can be configured via the stanza [`NetworkTargetPerformanceProcess_Config`].

Key *This setting is supported on: Windows, macOS*

Internally, uberAgent records network activity by process instance. However, that level of detail is rarely required. In most cases, it is sufficient to visualize network activity per process name. This is an optimization that reduces the event count in the backend and the data volume. Optionally, the agent can be configured to record network activity by process ID for full granularity by switching to grouping per ID instead of per name.

```
Setting name: Key
Description: What to group by: process name or ID
Valid values: name | id
Default: name
Required: no
```

IgnoreLowActivity *This setting is supported on: Windows, macOS*

This is another setting targeted at reducing the event count and the data volume. By default, a threshold is applied below which per-application network activity is dropped (not sent to the backend).

```
Setting name: IgnoreLowActivity
Description: Whether to ignore processes with very low activity during a collection interval
Valid values: true | false
Default: true
Required: no
```

IgnoreLoopbackTraffic *This setting is supported on: Windows*

This setting aims to reduce event count and data volume. By default, uberAgent ignores loopback traffic and doesn't send its metrics to the backend.

```
Setting name: IgnoreLoopbackTraffic
Description: Whether to ignore processes with loopback traffic or not
Valid values: true | false
Default: true
Required: no
```

Remarks: If you set `IgnoreLoopbackTraffic = false`, you should also set `NetworkDriverLegacyAPI = true` to view all metrics related to loopback traffic. By default, only network connections are collected without additional traffic details. This restriction stems from the inherent Windows API. To access all loopback traffic metrics, it's necessary to change the provider as well.

NetworkDriverEnabled *This setting is supported on: Windows*

Starting with uberAgent 6.0 (Windows), uberAgent monitors network activity with a kernel driver. This configuration option can be used to switch back to the older network data collection source ETW. ETW network monitoring has several limitations. Most notably, latency cannot be determined accurately.

```
Setting name: NetworkDriverEnabled
Description: Enables processing all network metrics by a driver instead of ETW.
Valid values: true | false
Default: true
Required: no
```

NetworkDriverLegacyAPI *This setting is supported on: Windows*

This setting is intended for internal use by vast limits. Only enable it if instructed by support.

```
Setting name: NetworkDriverLegacyAPI
Description: Use legacy WFP API to process packet streams in kernel
    mode.
Valid values: true | false
Default: false
Required: no
```

TestCompareNetworkDriverAndETW *This setting is supported on: Windows*

This setting is intended for internal use by vast limits. Only enable it if instructed by support.

```
Setting name: TestCompareNetworkDriverAndETW
Description: Collect network metrics using Windows ETW interfaces and
    the uberAgent network driver.
             The ProcUser field of the metric
                 NetworkTargetPerformanceProcess is extended by a suffix
                  ETW or DRV to differentiate the type of network event.
             Because of that, the network events are sent two times to
                 the configured backend.
             Use this feature to test unusual behavior in test
                 environments. This is not intended to be used in a
                 production environment.
Valid values: true | false
Default: false
Required: no
```

Web App Monitoring

What Is Web App Monitoring

uberAgent web app monitoring records page loads and XmlHttpRequests. For every such event, uberAgent collects the duration, the user, and the HTTP status code (plus some additional information). Web app monitoring works on every website and in all major browsers.

In the default configuration, uberAgent does not send full URLs to the backend. Instead, events are summarized per webserver (the host displayed in the browser's address bar). **Full URL monitoring** can optionally be enabled for specific domains or sites of interest.

Use Cases for Web App Monitoring

By providing insights into browser and website usage, uberAgent web app monitoring enables the following use cases:

- Measuring web **page load performance**

- Calculating **web app usage**
- Collecting an **inventory of web apps**
- Tracking the **user journey** on a website

Which Data is Collected by Web App Monitoring

The data collected as part of uberAgent web app monitoring is sent to the backend via the sourcetypes listed in the browser metrics documentation.

Additionally, some fields in the `SessionDetail` sourcetype (docs) are populated through web app metrics.

Requirements & Installation

Web App monitoring requires the uberAgent browser extensions to be installed in the end user's web browser. Please see the installation instructions for details on how to install the browser extensions.

Enabling or Disabling Web App Monitoring

uberAgent web app monitoring is enabled or disabled via the `BrowserPerformance*` timer metrics in the configuration. In the default configuration, web app monitoring is enabled for all supported browsers.

Full URL Monitoring

What Is Full URL Monitoring

Full URL monitoring optionally includes additional URL components in the following browser web app events or fields:

- the sourcetype `uberAgent:Application:BrowserWebRequests2` (docs)
- the field `SessionFgBrowserActiveTabHost` of the sourcetype `uberAgent:Session:SessionDetail` (docs)

By default, full URL monitoring is disabled. The path and query components are stripped from URLs sent to the backend, i.e., uberAgent only records the scheme (e.g., `https`), the host (e.g., `www.domain.com`) and the port (unless it's a standard port, e.g., 443 for https).

Example of the default behavior:

- Full URL in the browser: `https://splunk1.domain.com:8000/en-US/app/uberAgent/session_user_detail?earliest=-1h&latest=now`

- Shortened URL in uberAgent event: `https://splunk1.domain.com:8000`

Configuring Full URL Monitoring

The default level of URL component detail can be changed by adding items in the `[BrowserWebAppURL_Compo`
`]` stanza of the configuration.

Please note that URL **allowlists and denylists** (`[BrowserWebAppURL_Filter]` stanza) have precedence over `[BrowserWebAppURL_ComponentDetail]`. In other words, the level of detail configuration only applies to URLs that are not filtered by a deny or allow list.

BrowserWebAppURL_ComponentDetail Format of entries in the `[BrowserWebAppURL_ComponentDeta`
`]` stanza:

```
URL_REGEX = COMPONENT_DETAIL_SPEC
or:
URL_REGEX = COMPONENT_DETAIL_SPEC_TAB;COMPONENT_DETAIL_SPEC_REQUEST
```

Note: see the definition of URL_REGEX.

Tab URLs are matched against the regular expressions (`URL_REGEX`) of the items in the `[BrowserWebAppURL_ComponentDetail]` stanza. For matching URLs, `COMPONENT_DETAIL_SPEC` specifies which level of detail is recorded of the tab and request URLs.

Note:

- **Tab URLs** are URLs in the browser's address bar
- **Request URLs** are URLs the browser communicates with

Optionally, the detail level of the request URL can be set independently of the detail level of the tab URL by using the second format shown above.

Multiple URL component detail specs can be specified in a single `[BrowserWebAppURL_ComponentDetail`
`]` stanza. Matching is performed in the order in which items are listed. The first matching item wins.

COMPONENT_DETAIL_SPEC Format of the URL component detail specification (`COMPONENT_DETAIL_SPEC`
):

`host[:path[(DEPTH)]] [:query[(PARAMS)]]`

with:

- **DEPTH:** the number of path elements to include. Without `DEPTH`, the entire path is included.
- **PARAMS:** comma-separated list of regular expressions specifying query parameters to include. Without `PARAMS`, all query parameters are included. Details:

  - Query parameters to be included are specified as regular expressions.
  - Matching is case-sensitive.

Examples: Same Level of Detail for Tab URL and Request URL

The following examples show the effect of different configurations in a scenario where the tab URL and the request URL are identical:

- Tab URL: `https://splunk1.domain.com:8000/en-US/app/uberAgent/session_user_detail?earliest=-1h&latest=now`
- Request URL: `https://splunk1.domain.com:8000/en-US/app/uberAgent/session_user_detail?earliest=-1h&latest=now`

Host and Path Capture URLs on `splunk1.domain.com` including the path, but excluding the query:

```
.*\.?splunk1\.domain\.com/.*$ = host:path
```

Result:

```
https://splunk1.domain.com:8000/en-US/app/uberAgent/session_user_detail
```

Host and 2 Path Elements Capture URLs on `splunk1.domain.com` including the path, but only the first two elements:

```
.*\.?splunk1\.domain\.com/.*$ = host:path(2)
```

Result:

```
https://splunk1.domain.com:8000/en-US/app
```

Host, Path, and Query Capture URLs on `splunk1.domain.com` including the path and all query parameters:

```
.*\.?splunk1\.domain\.com/.*$ = host:path:query
```

Result:

```
https://splunk1.domain.com:8000/en-US/app/uberAgent/session_user_detail
?earliest=-1h&latest=now
```

Host, 1 Path Element, and 1 Query Parameter Capture URLs on `splunk1.domain.com` including the first path element and the query parameter `earliest`:

```
.*\.?splunk1\.domain\.com/.*$ = host:path(1):query(earliest)
```

Result:

```
https://splunk1.domain.com:8000/en-US?earliest=-1h
```

Host, 1 Path Element, and 1 Query Parameter Regex Capture URLs on `splunk1.domain.com` including the first path element and all query parameters that match the regular expression `.+est`:

```
.*\.?splunk1\.domain\.com/.*$ = host:path(1):query(.+est)
```

Result:

```
https://splunk1.domain.com:8000/en-US?earliest=-1h&latest=now
```

Examples: Different Level of Detail for Tab URL and Request URL

The following examples show the effect of different configurations in a scenario where the tab URL and the request URL are different:

- Tab URL: `https://splunk1.domain.com:8000/en-US/app/uberAgent/session_user_detail?earliest=-1h&latest=now`
- Request URL: `https://www.example.org/images/uberAgent`

Tab URL: Host Only, Request URL: Host and Path Capture only the host of the tab URL, but the host and the path of the request URL:

```
.*\.?splunk1\.domain\.com/.*$ = host;host:path
```

Result:

```
Tab URL:     https://splunk1.domain.com:8000/
Request URL: https://www.example.org/images/uberAgent
```

Tab URL: Host and Path, Request URL: Host Only Capture the host and the path of the tab URL, but only the host of the request URL:

```
.*\.?splunk1\.domain\.com/.*$ = host:path;host
```

Result:

```
Tab URL:     https://splunk1.domain.com:8000/en-US/app/uberAgent/
    session_user_detail
Request URL: https://www.example.org
```

Automatic Application Identification

Applications

While it is pretty clear where machine and process data come from, this is not so obvious with applications. In fact, there is a tiny little bit of magic involved here - the operating system does not have a concept of applications, it only knows about processes. The fact that `receiver.exe`, `concentr.exe` and `wfcrun32.exe` form a logical application entity called `Citrix Receiver` is totally irrelevant to the OS. For that reason, other monitoring products do not bother with applications.

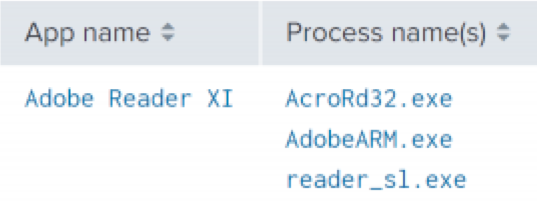
uberAgent is different. It automatically groups related processes to applications, because humans think in applications, not in processes - and uberAgent is designed for humans. When troubleshooting performance problems it is not enough to know that, say, `SelfService.exe` has a problem. You need to know that `Citrix Receiver` generates all those IOs. uberAgent shows you exactly that and makes it easy to analyze the impact your applications have on overall system performance.

How It Works (Windows)

Automatic application identification works for classic Win32 apps just as well as it does for Java applications, Universal Windows Platform (UWP) apps, and Windows services.

Win32

uberAgent determines the application names for all executables from properties of the MSI package the executable was installed from. Where that is not possible, uberAgent uses the information embedded in the executable.

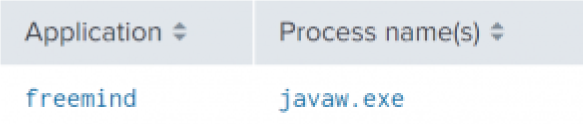


App name ↕	Process name(s) ↕

`Adobe Reader XI`	`AcroRd32.exe`
	`AdobeARM.exe`
	`reader_sl.exe`

Java

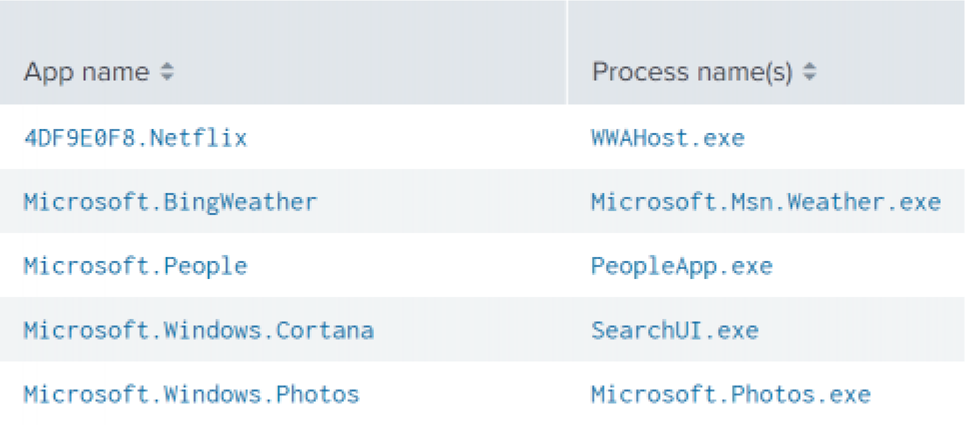
uberAgent extracts the class name or filename of a Java command line. This works for Java Web Start processes, too.



Application ↕	Process name(s) ↕
`freemind`	`javaw.exe`

UWP

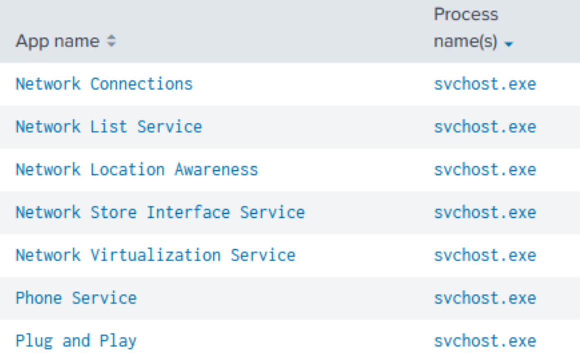
uberAgent retrieves the information from the UWP package.



App name ↕	Process name(s) ↕
4DF9E0F8.Netflix	WWAHost.exe
Microsoft.BingWeather	Microsoft.Msn.Weather.exe
Microsoft.People	PeopleApp.exe
Microsoft.Windows.Cortana	SearchUI.exe
Microsoft.Windows.Photos	Microsoft.Photos.exe

Windows services

Starting with Windows 10 1703 each service gets a dedicated `svchost.exe` instance if certain minimum RAM conditions are met. uberAgent extracts the (localized) display name for services that run in a dedicated `svchost.exe` instance.



App name ↕	Process name(s) ▾
Network Connections	svchost.exe
Network List Service	svchost.exe
Network Location Awareness	svchost.exe
Network Store Interface Service	svchost.exe
Network Virtualization Service	svchost.exe
Phone Service	svchost.exe
Plug and Play	svchost.exe

Requirements:

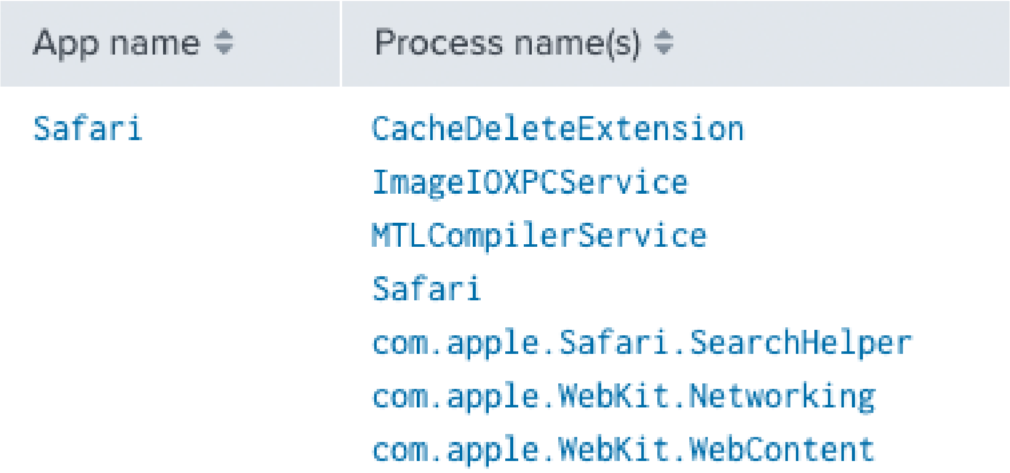
- Windows 10 1703 or higher
- uberAgent 5.2 or higher

How It Works (macOS)

Bundles

uberAgent determines application names for executables that are part of a bundle folder structure by retrieving either bundle name, bundle identifier, or bundle executable name from the bundle’s

`Info.plist` file. Executables located in nested app bundles are identified as their top-level host app bundle. Executables located in nested framework bundles are identified as their nested framework bundle, except for XPC Services which are always identified as the application that initiated their launch.



App name ⇕	Process name(s) ⇕

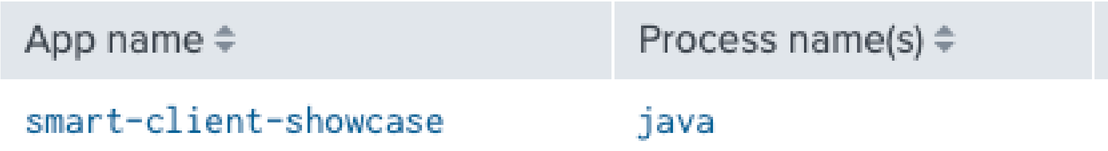
`Safari`	`CacheDeleteExtension`
	`ImageIOXPCService`
	`MTLCompilerService`
	`Safari`
	`com.apple.Safari.SearchHelper`
	`com.apple.WebKit.Networking`
	`com.apple.WebKit.WebContent`

Executables

uberAgent extracts application names from executables that are not part of a bundle folder structure by retrieving either bundle name, bundle identifier, or bundle executable name from the embedded binary info plist. If an executable can not be identified this way but is located in a system directory it is mapped to app name `macOS`.

Java

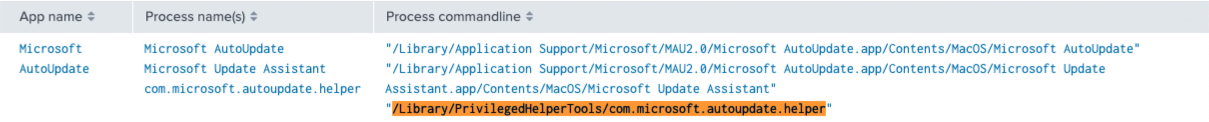
In case a Java based application is not bundled, uberAgent extracts the class name or filename of a Java command line. This works for Java Web Start processes, too.



App name ⇕	Process name(s) ⇕
`smart-client-showcase`	`java`

Privileged Helper Tools

uberAgent determines application names for executables located in `/Library/PrivilegedHelperTools` by inspecting their respective host app bundles as described above.



App name ⇕	Process name(s) ⇕	Process commandline ⇕
Microsoft AutoUpdate	Microsoft AutoUpdate Microsoft Update Assistant com.microsoft.autoupdate.helper	"/Library/Application Support/Microsoft/MAU2.0/Microsoft AutoUpdate.app/Contents/MacOS/Microsoft AutoUpdate" "/Library/Application Support/Microsoft/MAU2.0/Microsoft AutoUpdate.app/Contents/MacOS/Microsoft Update Assistant.app/Contents/MacOS/Microsoft Update Assistant" "/Library/PrivilegedHelperTools/com.microsoft.autoupdate.helper"

Configuration

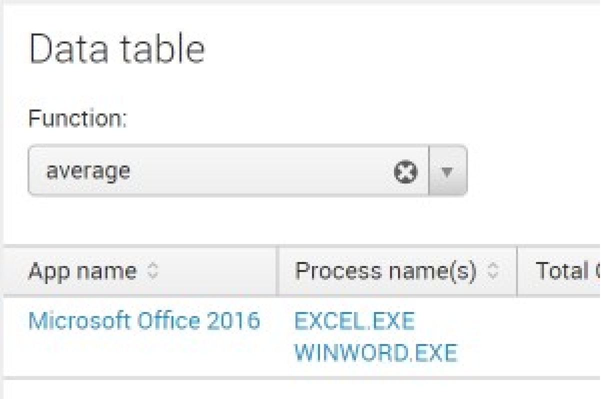
Adding Your Own Mappings

uberAgent's sophisticated method to map processes to applications works nearly every time for every application. When it does not, you can adjust it in the configuration section [ProcessToApplicationMapping]. Enter the full path to the process as a regular expression. Environment variables can also be used (enclosed in percent signs: %ProgramFiles%). Then specify the application name. Specifying only the file name without the full path works but is not recommended.

```
[ProcessToApplicationMapping]
# App name for C:\Dir\my.exe is "MyApp"
^C:\\DIR\\my\.exe$ = MyApp

# App name for all executables in "C:\Program Files\Windows Defender"
    is "Windows Defender"
^%ProgramFiles%\\Windows Defender\\.+\.exe$ = Windows Defender
```

Overriding Default Mappings

This configuration option also allows you to override uberAgent's default mapping. Let us take Microsoft Office as an example. uberAgent, by default, groups all Office processes like WINWORD.EXE and EXCEL.EXE into the application Microsoft Office.



Data table

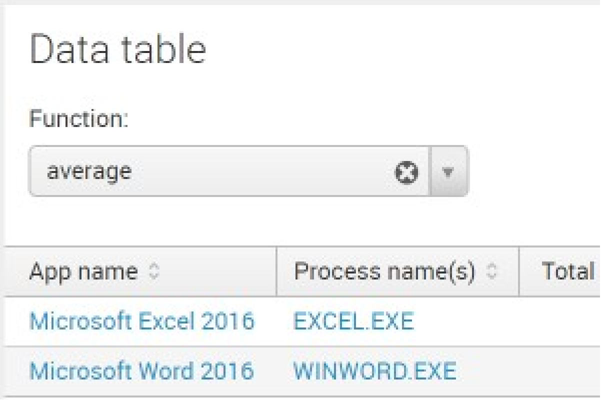
Function:

average

App name ⇕	Process name(s) ⇕	Total (
Microsoft Office 2016	EXCEL.EXE WINWORD.EXE	

By changing uberAgent's [ProcessToApplicationMapping] configuration settings as follows, you get individual applications per process.

```
[ProcessToApplicationMapping]
^%ProgramFiles(x86)%\\Microsoft Office\\root\\Office16\\EXCEL\.EXE$ = Microsoft Excel 2016
^%ProgramFiles(x86)%\\Microsoft Office\\root\\Office16\\WINWORD\.EXE$ = Microsoft Word 2016
```



Data table

Function: average

App name	Process name(s)	Total
Microsoft Excel 2016	EXCEL.EXE	
Microsoft Word 2016	WINWORD.EXE	

Ignoring Certain Processes

If you want uberAgent to ignore certain processes during application lookup, you can configure this with the option [ApplicationMappingIgnoredProcesses].

Citrix NetScaler (ADC) Monitoring

What is Citrix NetScaler (ADC) Monitoring

With Citrix NetScaler (ADC) monitoring, uberAgent collects appliance & gateway performance, utilization, and inventory data from Citrix NetScaler Application Delivery Controllers (metrics docs).

The uberAgent endpoint agent needs to be installed on a machine from where it can reach the physical or virtual Citrix NetScaler (ADC) appliance(s). On Citrix Delivery Controllers (DDCs), the collection of NetScaler (ADC) metrics is enabled by default. Of course, NetScaler (ADC) metrics can also be retrieved from any other machine instead.

Enabling or Disabling Citrix NetScaler (ADC) Monitoring

Citrix NetScaler (ADC) monitoring is activated by default only on Citrix Delivery Controllers, as explained above. You can modify this behavior through the configuration setting [CitrixADC_Config] in uberAgent's configuration.

To completely disable collecting Citrix ADC metrics, disable the following metrics in the configuration:

- CitrixADCPerformance
- CitrixADCvServer
- CitrixADCGateways
- CitrixADCInventory

Installation and Configuration

If you have uberAgent already installed on one or more DDCs for Citrix site monitoring, it makes sense to also collect Citrix NetScaler (ADC) metrics there. Make sure that the DDCs can communicate with the Citrix NetScaler (ADC) management interface (NSIP) either through port 80 or, if you enabled `Secure Access Only` for the NSIP, through port 443.

While DDCs may be the ideal place to capture Citrix NetScaler (ADC) data for most customers, that may not be the case for everybody. As explained above, you can overwrite the setting `CollectADCInformation` in the stanza `[CitrixADC_Config]` to collect Citrix NetScaler (ADC) metrics from any installed uberAgent endpoint. Make sure that this setting is only distributed to the machines that are intended for this purpose. If you distribute it to all machines where uberAgent is installed, every endpoint communicates with the Citrix NetScaler (ADC) appliance(s), which could overload the appliance(s) and generate unnecessary data.

`CollectADCInformation=MachineList` in combination with a set of machines configured in `CollectADCInformationMachines`, defines on which machines the collection of ADC metrics will be performed.

Citrix NetScaler (ADC) Configuration

Besides network access, uberAgent also needs credentials to be able to login into the Citrix ADC appliances. It only needs to read information; hence using a user with the command policy *read-only* bound is the recommended option. Users with more privileges are also supported, but not recommended from a security perspective.

Create a user with the command policy *read-only* bound from CLI:

```
add system user nsread Password -externalAuth DISABLED -timeout 900 -maxsession 20
bind system user nsread read-only 100
```

uberAgent Configuration

This section explains the configuration via uberAgent's configuration file. The configuration via GPO works accordingly.

Use the stanza [CitrixADC_Config] to specify how uberAgent should connect to your appliance(s). If multiple [CitrixADC_Config] stanzas are specified, the configured metrics are determined for each of them. Use one [CitrixADC_Config] stanza per Citrix NetScaler (ADC) pair.

Following are some examples of valid Citrix ADC configurations. More information on the parameters is available in the configuration file itself. Note that passwords can be encrypted.

Example 1: One Citrix NetScaler (ADC) appliance manageable through HTTP (port 80). Data collection happens on DDCs only

```
[CitrixADC_Config]
Server = 10.1.1.21
Username = nsread
Password = Password
Https = false
CollectADCInformation = DDCOnly
```

Example 2: A Citrix ADC high-availability pair manageable through HTTPS (port 443). Data collection happens not on a DDC.

- If HTTPS is used, the entries in the setting Server must match those in the certificate bound to the NSIP.
- uberAgent collects performance information for the primary appliance only. So for best collection performance, list the primary appliance first.

```
[CitrixADC_Config]
Server = CitrixADC1.domain.local,CitrixADC2.domain.local
Username = nsread
Password = Password
Https = True
CollectADCInformation = True
```

Example 3: A Citrix ADC high-availability pair manageable through HTTPS (port 443). Data collection happens on a set of machines.

```
[CitrixADC_Config]
Server = CitrixADC1.domain.local,CitrixADC2.domain.local
Username = nsread
Password = Password
Https = True
CollectADCInformation = MachineList
CollectADCInformationMachines = DDC01, DDC02
```

Deployment Considerations

uberAgent's NetScaler integration sequentially queries the configured NetScalers with PowerShell through the NITRO API. This implementation method has pros and cons.

This section describes if the uberAgent implementation is for you, or if you should go with NetScaler's builtin Observability integrations.

When to Use uberAgent's Netscaler Integration

uberAgent's NetScaler integration is easy to use, comes with beautiful Splunk dashboards out-of-the-box, and is tailored to the Citrix CVAD/DaaS administrator who focuses on the quality of user sessions.

The NITRO API offers plenty of information about NetScaler, but comes with limitations. For example, it does not offer insights into failed AAA logins. In addition, not all existing NITRO API fields are queried by uberAgent, but only the ones that you see in the Splunk dashboards. Additional fields can be requested via feature requests.

uberAgent queries NetScaler sequentially. Querying lots of NetScaler boxes from one uberAgent endpoint can lead to performance issues and requires distributing the load to multiple endpoints.

Use uberAgent's NetScaler integration, when you have only a few NetScaler to monitor and want to benefit from pre-built Splunk dashboards and easy configuration.

When to Use NetScaler's Built-in Observability Integrations

NetScaler can send metrics and logs to various backends natively. This includes Splunk with example dashboards.

NetScaler Console, the monitoring and configuration tool for NetScaler from Citrix, can send data to Observability platforms like Splunk, too.

Sending data off the NetScaler instead of querying it externally through an API is more efficient. Also, NetScaler can send, more or less, data for every NetScaler setting that you see in the UI or via the Shell. It can also send log files which include information on failed AAA logins.

While NetScaler can send a lot to Observability platforms, you have to define which settings you would like to see. Also, you have to create your own visualizations or change the example dashboards.

Use NetScaler's and NetScaler Console's built-in Observability integrations when you require monitoring more than a handful NetScaler, when you need management logs or more data that is not available through uberAgent and the NITRO API.

System Requirements

Please see the system requirements page for details.

Citrix Cloud Monitoring

Disclaimer: Citrix Cloud monitoring makes use of the Citrix Virtual Apps and Desktops OData APIs. These APIs are currently in Tech Preview state!

What Is Citrix Cloud Monitoring

Citrix Cloud monitoring refers to uberAgent's capability to monitor the Citrix Virtual Apps and Desktops (CVAD) control plane in Citrix Cloud. This is an exciting feature for customers who do not host delivery controllers on-premises but leverage Citrix Cloud instead.

Which Metrics Are Collected By Citrix Cloud Monitoring

Citrix Cloud monitoring collects a rich set of metrics about many aspects of a Citrix Cloud deployment (see the metrics documentation for details):

- Published **applications**
- **Desktops**
- Desktop groups
- Machines
- Machine Catalogs

Enabling or Disabling Citrix Cloud Monitoring

Citrix Cloud monitoring is **disabled by default**. You can enable the data collection through the configuration setting `[CitrixCloud_Config]` in uberAgent's configuration.

To completely disable Citrix Cloud monitoring, disable the following metrics in the configuration:

- CitrixDCDesktopGroup
- CitrixDCCatalog
- CitrixDCMachine
- CitrixDCGeneralInformation
- CitrixDCApplication
- CitrixDCPublishedDesktops

Installation and Configuration

To enable the Citrix Cloud data collection, please configure the stanza `[CitrixCloud_Config]` in uberAgent's configuration. Make sure that this setting is only distributed to the machines that are intended for this purpose. If you distribute it with `CollectCitrixCloudInformation=True` to all machines where uberAgent is installed, every endpoint communicates with the Citrix Cloud APIs, which could overload the APIs and generate unnecessary data.

`CollectCitrixCloudInformation=MachineList` in combination with a set of machines configured in `CollectCitrixCloudInformationMachines`, defines on which machines the collection of Citrix Cloud metrics will be performed.

uberAgent Configuration

This section explains the configuration via uberAgent's configuration file. The configuration via GPO works accordingly.

Use the stanza `[CitrixCloud_Config]` to specify how uberAgent should connect to the Citrix Cloud infrastructure. If multiple `[CitrixCloud_Config]` stanzas are specified, the configured metrics are determined for each of them. Use one `[CitrixCloud_Config]` stanza per Citrix Cloud tenant. The API endpoint option must contain to full URL to the Citrix Cloud server (e.g., `https://api-eu.cloud.com`). The appendix `/monitordata` is optional.

Following are some examples of valid Citrix Cloud monitoring configurations. More information on the parameters is available in the configuration file itself.

Example 1: one Citrix Cloud tenant. Data collection happens on all uberAgent endpoints.

```
[CitrixCloud_Config]
API endpoint = https://api-eu.cloud.com
CustomerId = <CustomerId>
ClientId = <ClientId>
ClientSecret = <ClientSecret>
CollectCitrixCloudInformation=True
```

Example 2: one Citrix Cloud tenant. Data collection happens on specific endpoints only.

```
[CitrixCloud_Config]
API endpoint = https://api-eu.cloud.com
CustomerId = <CustomerId>
ClientId = <ClientId>
ClientSecret = <ClientSecret>
CollectCitrixCloudInformation=MachineList
CollectCitrixCloudInformationMachines = MachineA,MachineB
```

Requirements

- Create a Citrix Cloud API client as described here.
- The Citrix Virtual Apps and Desktops Remote PowerShell SDK must be installed on the endpoint.
- PowerShell script execution is required by the Remote PowerShell SDK. Set the execution policy (machine scope) accordingly, for example, `RemoteSigned`.

Note: the Remote PowerShell SDK must **not** be installed on:

- Citrix Virtual Apps and Desktops **delivery controllers**
- Citrix **Cloud Connector** machines

Please see the system requirements page for details.

Citrix Session Monitoring

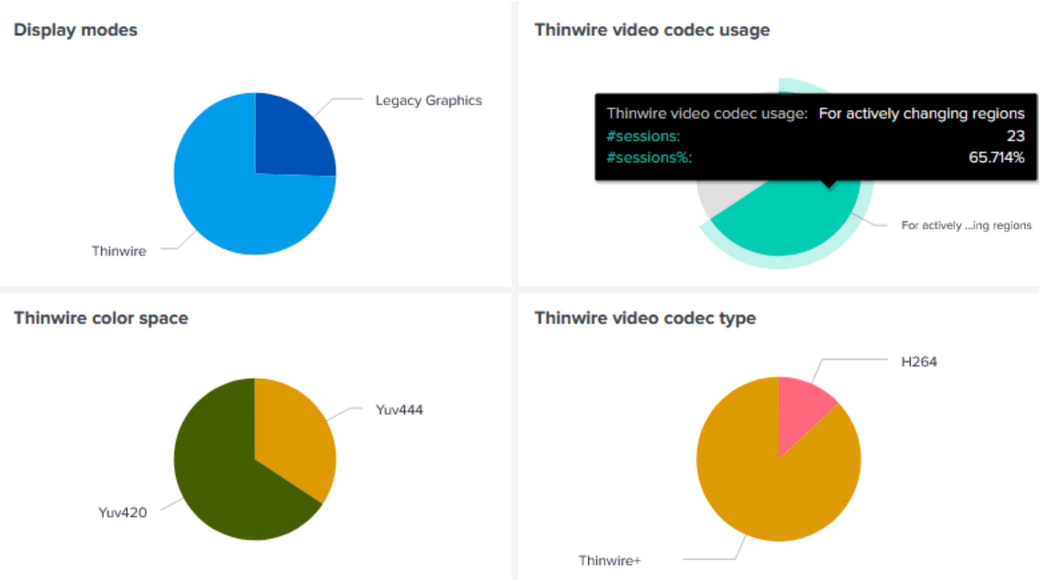
uberAgent provides unprecedented visibility into what's happening on the session level: bandwidth usage, protocol latency, Citrix policies, video encoding settings, and much more.

Citrix Session Configuration Details

The session configuration dashboard lets you analyze almost any aspect of how users connect and what policies are being applied to their sessions. The dashboard pulls its data from live sessions, so it always reflects the current state. Historical values are available, too, of course.

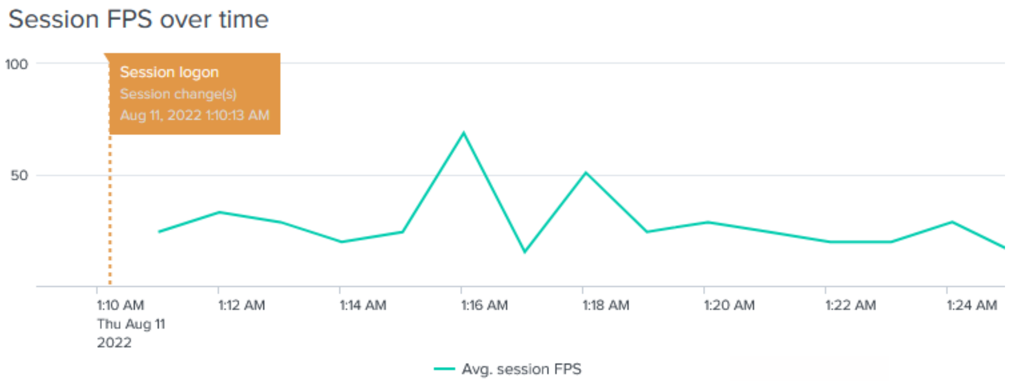
Video Codec, Display Mode, Color Space

Many things need to be configured just right to achieve optimal visual quality in a remoting session. These include the video codec and color space, to pick just one set of essential session parameters. As you can see in the screenshot below, uberAgent shows you exactly which codecs and rendering algorithms are configured for your sessions:



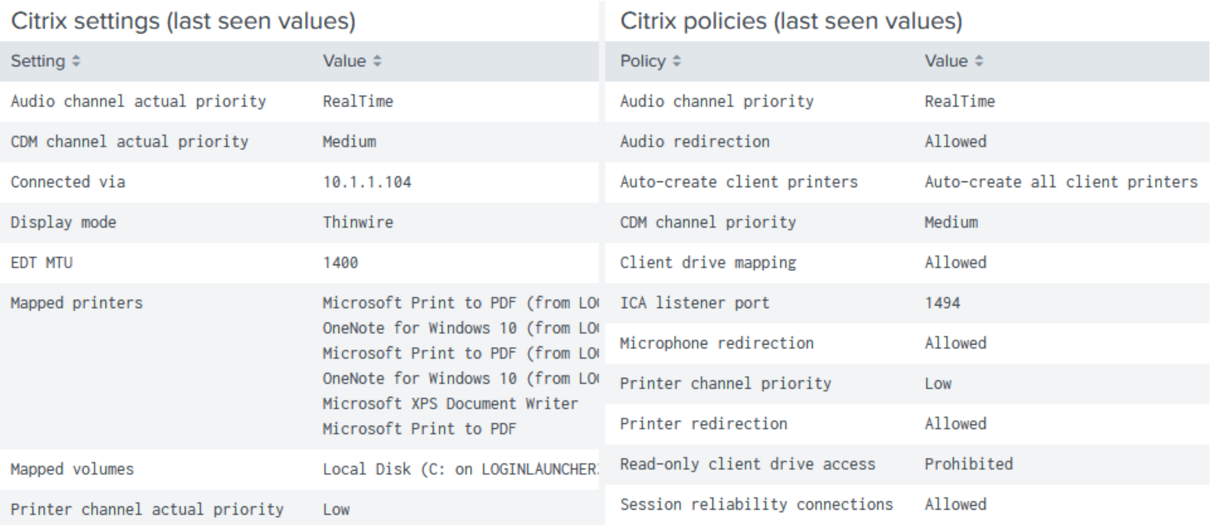
Frame Rate (FPS)

Another important metric that greatly influences rendering fidelity as experienced by end users is the frame rate, measured in frames per second. uberAgent continuously collects the frame rate for the entire duration of any session. This allows IT staff to proactively optimize the user experience without a single end-user having to call the help desk.



Citrix Policies and Settings

The Citrix session configuration dashboard offers drilldown capabilities that allow you to inspect the policies and settings of every session individually. The following screenshot shows a small excerpt of what's available:



Citrix settings (last seen values)

Setting ⇕	Value ⇕
Audio channel actual priority	RealTime
CDM channel actual priority	Medium
Connected via	10.1.1.104
Display mode	Thinwire
EDT MTU	1400
Mapped printers	Microsoft Print to PDF (from LO OneNote for Windows 10 (from LO Microsoft Print to PDF (from LO OneNote for Windows 10 (from LO Microsoft XPS Document Writer Microsoft Print to PDF
Mapped volumes	Local Disk (C: on LOGINLAUNCHER:
Printer channel actual priority	Low

Citrix policies (last seen values)

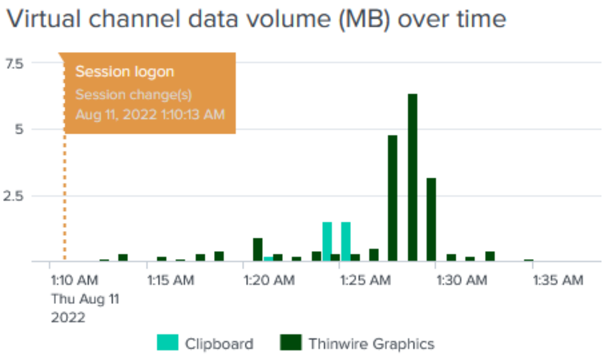
Policy ⇕	Value ⇕
Audio channel priority	RealTime
Audio redirection	Allowed
Auto-create client printers	Auto-create all client printers
CDM channel priority	Medium
Client drive mapping	Allowed
ICA listener port	1494
Microphone redirection	Allowed
Printer channel priority	Low
Printer redirection	Allowed
Read-only client drive access	Prohibited
Session reliability connections	Allowed

Citrix Session Protocol Insights

The session protocol insights dashboard is focused on ICA/HDX remoting protocol analytics.

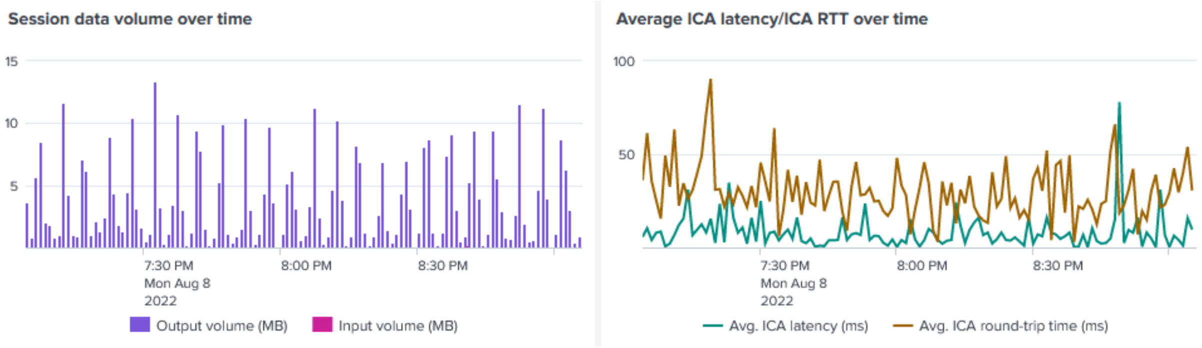
Virtual Channel Bandwidth Usage

One of the dashboard's highlights: it visualizes session bandwidth usage per virtual channel. This is crucial information for any Citrix admin, as it explains which kind of data is transferred in the encrypted Citrix HDX protocol. In the screenshot below, for example, the user seems to have copied a lot of data to or from the clipboard (virtual channel: `Clipboard`), followed by what may have been scrolling in a long document (virtual channel: `Thinwire Graphics`).



ICA RTT

Another cool metric helps to get to the bottom of many networking issues: ICA round-trip time (RTT). A high round-trip time may be experienced by end users as screen lag. The session protocol insights dashboard visualizes ICA RTT along with ICA latency, which offers a lower-level view of network connectivity (think: an ICA/HDX version of ping). uberAgent also determines the total incoming and outgoing HDX data volume, of course:



Configuration

Citrix session configuration details as well as policies and settings are configured through two metrics. The metrics are collected by default if uberAgent is running on a Citrix Virtual Apps and Desktops session host, but this behavior is customizable.

- `CitrixSessionVirtualChannelDetail`
- `CitrixSessionConfig`

Requirements

- Citrix Virtual Apps and Desktops 7.6 LTSR and higher
- Citrix Cloud

> **Note**
>
> **Verify That Registry Is Configured Correctly for Citrix Virtual Channel Metrics**
>
> Starting with uberAgent 7.3, the agent uses the following registry values to automatically determine the installation location of the Citrix HDX binaries that are part of a Citrix deployment. Make sure that at least one of these values points to the correct Citrix HDX binaries folder.
>
> **1.**
>
> - **Hive**: HKLM\Software\Citrix\Install\HDX
> - **Value name**: BinFolder
> - **Type**: REG_SZ
> - **Data**: *Path to HDX binary folder* (e.g., `C:\Program Files\Citrix\HDX\bin\`)
>
> **2.**

> - **Hive**: HKLM\Software\Citrix\Install
> - **Value name**: Location
> - **Type**: REG_SZ
> - **Data**: *Path to HDX binary folder* (e.g., `C:\Program Files\Citrix\HDX\bin\`)
>
> **3.**
>
> If neither of the values listed above is present in your deployment, please create the following registry value and set it to the installation location of the Citrix HDX binaries. This value takes precedence over those mentioned previously.
>
> - **Hive**: HKLM\Software\vast limits\uberAgent
> - **Value name**: CitrixHDXBinFolder
> - **Type**: REG_SZ
> - **Data**: *Path to HDX binary folder* (e.g., `C:\Program Files\Citrix\HDX\bin\`)

Please see the system requirements page for supported product versions.

Citrix Site Monitoring

What is Citrix Virtual Apps and Desktops Site Monitoring

uberAgent detects if it is running on a Citrix Delivery Controller (DDC) or a Citrix Virtual Desktop Agent (VDA). On DDCs, **uberAgent automatically activates additional metrics** like machine registration status, license usage, and published application inventory.

Which Metrics Are Collected By Citrix Site Monitoring

Citrix site monitoring collects a rich set of metrics about many aspects of Citrix CVAD sites (see the metrics documentation for details):

- Published **applications**
- Databases
- **Desktops**
- Desktop groups
- Hypervisors
- **Licenses**
- Machines
- Machine Catalogs

Enabling or Disabling Site Monitoring

Citrix site monitoring is **enabled by default** but is only **activated on Citrix delivery controllers** as explained above. To disable Citrix site monitoring disable the following metrics in the configuration:

- CitrixDCDesktopGroup
- CitrixDCCatalog
- CitrixDCMachine
- CitrixDCHypervisor
- CitrixDCGeneralInformation
- CitrixDCLicenseInformation
- CitrixDCApplication
- CitrixDCPublishedDesktops

Installing & Configuring Citrix Site Monitoring

Install the uberAgent endpoint agent on at least one delivery controller per site. Installation on more than one controller is optional - it helps ensure uninterrupted data collection even when individual DDCs are unavailable.

Required Permissions

uberAgent requires read-only administrator permissions for:

- each delivery controller's computer account
- the local SYSTEM account

Permissions Script Template

The following script template can be used to grant the required permissions. Fill in the names of your domain and of your Citrix DDCs before running it in an elevated PowerShell console.

```
Add-PSSnapin Citrix.DelegatedAdmin.Admin.V1
New-AdminAdministrator -Sid S-1-5-18 -Enabled $true
Add-AdminRight -Role 0a05f0c6-0153-4852-a55a-989d6a95c0eb -Administrator S-1-5-18 -All
New-AdminAdministrator -Name
```

Permissions Script Example

The following example script demonstrates how to grant the required permissions in an environment with two DDCs named `CTX-DDC1` and `CTX-DDC2` that are members of the `CONTOSO` domain. Note that you need to append a dollar sign (`$`) to the computername.

```
Add-PSSnapin Citrix.DelegatedAdmin.Admin.V1
New-AdminAdministrator -Sid S-1-5-18 -Enabled $true
Add-AdminRight -Role 0a05f0c6-0153-4852-a55a-989d6a95c0eb -Administrator S-1-5-18 -All
New-AdminAdministrator -Name CONTOSO\CTX-DDC1$ -Enabled $true
Add-AdminRight -Role 0a05f0c6-0153-4852-a55a-989d6a95c0eb -Administrator CONTOSO\CTX-DDC1$ -All
New-AdminAdministrator -Name CONTOSO\CTX-DDC2$ -Enabled $true
Add-AdminRight -Role 0a05f0c6-0153-4852-a55a-989d6a95c0eb -Administrator CONTOSO\CTX-DDC2$ -All
```

Sizing Guidelines

Sizing recommendations for delivery controllers with uberAgent:

- **1,000 machines:** minimal CPU usage, 30 MB RAM
- **25,000 machines:** 0.5 CPU cores, 500 MB RAM

System Requirements

The following services need to be started on delivery controllers with uberAgent:

- Citrix Delegated Administration Service
- Citrix Monitor Service

Please see the system requirements page for supported product versions.

Central License File Management

Central License File Share

License file management is easy with uberAgent. All you have to do is set up a file share with read permissions for computer accounts, drop your uberAgent license file(s) there and configure the share path in uberAgent's configuration option `LicenseFilePath`. Endpoints periodically check `LicenseFilePath` for new licenses. If any are found that are not yet cached locally, the new license files are copied to the local license cache directory in `%ProgramData%\vast limits\uberAgent\License cache` (Windows) or in `/Library/Application Support/uberAgent/License Cache` (macOS) respectively.

Storing uberAgent's License in Azure Files

uberAgent can also validate a license file stored in an Azure Files share. Our knowledge base article describes the necessary steps to set this up.

Availability Requirements

There is **no need** for the configured license file path to be available all of the time. Endpoints always use the local license cache for license validation.

Alternative: Local License Storage

Of course, there is no requirement to set up a central license file path. You can also distribute the license file to each endpoint. If the license file path is not specified uberAgent falls back to the installation directory (Windows) or to `/Library/Application Support/uberAgent` (macOS).

Event Data Filtering

What is Event Data Filtering

Event data filtering allows defining rules with conditions that are evaluated for every event before it is sent to the backend. This feature applies to all built-in metrics. With each matching rule, a pre-defined action is executed that controls whether the event is sent to the backend or not. Additionally, it allows clearing the contents of certain fields before the event is sent to the backend.

Use Cases for Event Data Filtering

Data Volume Reduction

Depending on the requirements, one might only need a subset of the events generated by uberAgent for certain sourcetypes. Filtering out unnecessary data at the endpoint may reduce the data volume significantly (see the documentation for other ways to reduce the data volume).

Example: every time a `cmd.exe` process is started on Windows, an accompanying console host process is started too. These `conhost.exe` processes are rarely of interest and can be safely excluded.

```
[EventDataFilter]
# Exclude "conhost.exe" (typically started from the path: \??\C:\
   WINDOWS\system32\conhost.exe)
```

```
Action = deny
Sourcetype = Process:ProcessStartup
Query = regex_match_path(ProcPath, r"^(\\\?\?\\)?%SystemRoot%\\System32\\conhost\.exe$")
```

Example: compared to Windows, macOS runs significantly more concurrent processes. which can also lead to an increased data volume. One way to adjust this is to exclude the processes that belong to the operating system.

```
[EventDataFilter]
# Exclude processes belonging to the operating system
Action = deny
Sourcetype = Process:ProcessDetail
Sourcetype = Process:ProcessStartup
Sourcetype = Process:ProcessStop
Sourcetype = Process:ProcessStatistics
Sourcetype = Process:NetworkTargetPerformance
Query = AppId == "mcOS"
```

Sensitive Data Removal

Some sourcetypes have fields with data that may be considered sensitive in nature, such as window titles. Event data filtering allows clearing such fields, on the endpoint, before the data is sent to the backend for indexing and searching.

Example: clear the contents of session detail window title metric.

```
[EventDataFilter]
# Clear the contents of window titles.
Action = clear
Sourcetype = Session:SessionDetail
Field = SessionFgWindowTitle
Query = true
```

The query above always evaluates to **`true`** and therefore the action `clear` is executed for this particular event and the field `SessionFgWindowTitle` is cleared to an empty value.

Actions: allow, clear and deny

allow

When an event matches a rule with the allow action, it is immediately sent to the backend. No further filtering rules are applied to the event.

deny

When an event matches a rule with the deny action, it is immediately discarded and not sent to the backend. Like allow, no further rules are applied.

clear

The clear action is used to overwrite the value of a specified field with an empty value. Filtering continues with subsequent rules after clear.

Configuring Event Data Filtering

An event data filter is configured using the configuration file. The stanza `[EventDataFilter]` starts a new filter configuration, followed by multiple settings.

It's essential to note that filter rules are processed sequentially. Once a rule matches, the filtering process for that particular event stops. Therefore, the order of filters can be crucial for the desired outcome.

Setting	Description	Valid Values
Action	Specifies the action to be taken if a query rule is matched.	`allow`, `deny` or `clear`
Sourcetype	The category and name of the sourcetype.	Please refer to the metrics documentation for a list of available sourcetypes.
Query	The query rule to filter using uAQL.	Please refer to the uAQL documentation.
Field	Optional setting to specify the field to be cleared if the current Action is `clear`. This setting can be configured multiple times.	This can be any field of the given metric.

Example: Exclude DNS Monitoring Events Caused by Browsers

A browser can generate a significant number of DNS monitoring events that are rarely of interest. Such events can be easily ignored using the following event data filter.

```
[EventDataFilter]
# Deny any DNS event caused by browsers.
Action = deny
Sourcetype = Process:DnsEvent
Query = ProcName in ["chrome.exe", "iexplore.exe", "firefox.exe", "
    msedge.exe", "opera.exe"]
```

Example: Exclude Process Detail Metrics For Certain Processes

```
[EventDataFilter]
# Exclude processes whose name is exactly one of the given names.
Action = deny
Sourcetype = Process:ProcessDetail
Query = ProcName in ["cmd.exe", "conhost.exe", "csrss.exe", "lsm.exe",
    "smss.exe", "wininit.exe", "winlogon.exe"]
```

Example: Exclude Everything Except..

Consider a use case where one wants to exclude an entire metric, except for specific data in events. This is achievable by explicitly allowing certain processes but denying all others. The stanza ordering is important because filter processing stops after the first match.

```
[EventDataFilter]
# Allow network events caused by John Doe and Jane Doe.
Action = allow
Sourcetype = Process:NetworkTargetPerformance
Query = ProcUser in ["John Doe", "Jane Doe"]

[EventDataFilter]
# Deny any network event that was not allowed in a previous filter.
Action = deny
Sourcetype = Process:NetworkTargetPerformance
Query = true
```

Per-Receiver Configuration

In addition to the fields of a sourcetype, certain built-in receiver fields may also be used. This allows creating event data filters that are active for certain receivers, only. Check the documentation on routing to different backends to learn more on the use case.

The available fields are `Receiver.Name` and `Receiver.Index`.

Example: Clear Fields Per Receiver

For this example, we take a look at the Session Detail metric. Consider there are two receivers configured, one for most metrics and an extra receiver for sensitive content with restricted access.

```
[EventDataFilter]
# Clear the contents of window titles for any receiver except "
   uberagent_sensitive"
Action = clear
Sourcetype = Session:SessionDetail
Field = SessionFgWindowTitle
Query = Receiver.Name != "uberagent_sensitive"
```

This configuration will clear the contents of `SessionFgWindowTitle` in all receivers, except the receiver `uberagent_sensitive`.

Example: Filter Events Per Receiver

We can also use almost the same rule to achieve a different use case: exclude events from all receivers, except `uberagent_sensitive`.

```
[EventDataFilter]
# Exclude this event from all receivers except "uberagent_sensitive"
Action = deny
Sourcetype = Session:SessionDetail
Query = Receiver.Name != "uberagent_sensitive"
```

Instead of clearing the field content for the non-matching receiver, we simply deny sending the event to all receivers, except `uberagent_sensitive`.

More Examples

More examples including rules created by our support for our customers can be found in the knowledge base article here.

Username and Configuration Setting Encryption

uberAgent optionally encrypts user and domain names in the agent and has support to encrypt sensitive settings in its configuration.

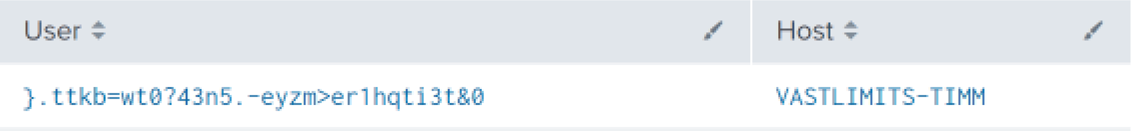
User and Domain Names

Encryption

User and domain names can be encrypted in the agent before being sent off to the backend. This can be useful for compliance with privacy regulations.

User and domain name encryption is disabled by default. If required, enable it via the configuration setting `EncryptUserNames`.

With encryption enabled, user and domain names show up in the backend and in dashboards like this:



User ⇕	Host ⇕
}.ttkb=wt0?43n5.-eyzm>er1hqti3t&0	VASTLIMITS-TIMM

Decryption

uberAgent's download package comes with the command line-tool `uAEncrypt.exe`, which can be used to decrypt usernames as shown in the following example:

```
uAEncrypt.exe -decrypt -keyId 101 -data
```

Configuration Settings

Configuration settings can optionally be obfuscated or retrieved from the OS-specific credential store. Obfuscation is the older option, but less secure. The OS credential store has been added in uberAgent 7.2 and is the recommended way to store passwords and other sensitive information that is required by uberAgent.

Encryption

Some configuration setting may optionally be retrieved from the OS-specific credential store. To indicate to uberAgent that a setting should be read from the OS credential store, specify its value in the following format: `###UA_CREDENTIAL_SOMENAME###`. When uberAgent encounters the above format in its configuration, it reads the actual value from the setting `SOMENAME` in the OS credential store.

Example:

To secure a backend receiver's REST token, specify it in uberAgent's configuration as follows:

```
RESTToken = ###UA_CREDENTIAL_SplunkRESTToken###
```

Distribute the actual REST token value to the OS credential store (see below) of all endpoints where uberAgent's configuration references it.

Supported Settings Encryption is supported for the following configuration settings:

```
[AzureEventHubsConfiguration]
AzureClientSecret

[CitrixADC_Config]
Password

[CitrixCloud_Config]
ClientSecret

[Receiver]
RESTToken
TLSClientCertificate
```

Deployment to the OS Credential Store uberAgent doesn't handle the deployment of secrets to the endpoint's OS credential store.

Windows Implementation The credentials are read from the SYSTEM user's credential store (more information).

macOS Implementation Under macOS the credentials are read from the keychain (more information). The keychain items
must be stored in the system keychain, and uberAgent must be exempt from the option `Confirm before allowing access` under the tab `Access Control`.

Example:

To create a keychain item securing the REST token for Splunk, you can run the following command line:

```
sudo security add-generic-password -a "Splunk"-s "uberAgent"-w "TOKEN_TO_BE_USED"-T "/Library/uberAgent/uberAgent.app"-U /Library/Keychains/System.keychain
```

where `Splunk` is the keychain item's account name, `uberAgent` is the service name and `TOKEN_TO_BE_USED` the REST token. The account name can be chosen freely. If you want to change the default service name `uberAgent`, you can do so by adding the configuration option `CredentialStoreServiceName` in stanza `Miscellaneous`, and passing the name with option `-s` as seen in the example above.

Example:

```
[Miscellaneous]
ConfigFlags = CredentialStoreServiceName:MY_CUSTOM_SERVICE_NAME
```

Obfuscation

Sensitive settings like passwords can be obfuscated with the command-line tool `uAEncrypt.exe`, which is available in the uberAgent download package. The syntax is viewable by running `uAEncrypt.exe -?`.

To encrypt `MySecretPassword`:

```
uAEncrypt.exe -encrypt -keyId 1 -data PlaintextData
```

De-Obfuscation De-obfuscation is possible with:

```
uAEncrypt.exe -decrypt -keyId 1 -data ObfuscatedData
```

Supported Settings Obfuscation is supported for all configuration settings.

ESA Features & Configuration

This section describes features that are part of the uberAgent ESA endpoint security analytics product. Please take a look at the UXM and ESA products page for an explanation what the two products are about and how they work together.

Enabling ESA

uberAgent ESA is enabled or disabled through the configuration setting `EnableESA`. In the default configuration, `EnableESA` is set to **`true`**. However, the endpoint agent enables ESA only if one of the following is also true:

- Valid UXM and ESA licenses are available (i.e., the product is properly licensed)
- Neither UXM nor ESA licenses are available (demo mode)

ESA Enables EnableExtendedInfo

When ESA is enabled, `EnableExtendedInfo` is automatically enabled, too, irrespective of the configuration setting of the same name. `EnableExtendedInfo` ensures that GUIDs are added to process events so that individual process instances can be identified and tracked over time.

Threat Detection Engine

uberAgent ESA Threat Detection **makes system activity traceable** and searchable.

When a Threat Detection rule matches a risky process, an unusual network connection, or similar activity, uberAgent ESA creates an event in your SIEM (e.g., Splunk). Threat Detection's comprehensive, extensible ruleset is powered by uAQL, a feature-rich query language that is both easy to read by humans and fast to process by computers.

uberAgent ESA comes with hundreds of predefined rules for many common attack vectors and converters for Sysmon rules and Sigma signatures. Customizing and extending ESA's ruleset is explicitly encouraged.

Rule Sources

uberAgent ESA ships with rules from two different sources: vast limits rules and third-party rules. The former are curated by vast limits, while the latter are converted from sources such as the Sigma project.

Rule Storage

uberAgent ESA Threat Detection rules are part of uberAgent's configuration, which is maintained in the uberAgent Configuration GitHub repository.

Metadata

Annotation (MITRE ATT&CK Technique ID)

Annotations add supplementary data to Threat Detection rules, notably MITRE ATT&CK technique IDs.

Tag & Risk Score

Every ESA Threat Detection rule comes with a tag and a risk score that are assigned to matching events.

Sourcetype

ESA Threat Detection events are assigned the sourcetype `uberAgentESA:ActivityMonitoring:ProcessTagging` (see the metrics documentation for a description of the fields).

Visualization

ESA Threat Detection events are visualized in the *Threat Detection Events* dashboard, which is part of the `uberAgent_ESA` Splunk searchhead app.

File System Activity Monitoring

File system activity monitoring is uberAgent ESA's capability to detect changes to objects in the file system. The monitored object types include files, of course, but also named pipes. uberAgent ESA can identify and react many different types of file system events, e.g., read, write, delete. Please see the event types documentation for details.

Configuration

File system activity monitoring can be configured via the stanza `[FilesystemMonitoring]`. By default, file system activity monitoring is enabled. To disable file system activity monitoring set `Enabled = false`.

Monitored Volumes

New Volumes Added to the System In the default configuration, uberAgent only monitors volumes that were present when the agent was started; newly added volumes are ignored. This can be changed by setting `Hotplug = true`.

Disabling Monitoring by Volume Type To monitor only certain volume types, use the setting `Monitor`. The following table shows available values.

Volume type	Description	Platform
`Disks`	Monitors all hard drives. This includes removable media such as USB thumb drives.	all
`Mailslots`	Monitors all mailslots, a mechanism for one-way interprocess communication (IPC) using pseudofiles that reside in memory.	Windows
`NamedPipes`	Monitors all named pipes, a mechanism for interprocess communication (IPC) that is accessed much like a file.	Windows
`NetworkShares`	Monitors all remote file systems (access to network shares). Supported protocols depend on the OS: **Windows:** all protocols handled by the multiple UNC provider (MUP), typically: SMBRDPNP (remote desktop services client drives)WebDAV **macOS:** SMBAFPNFS	all

Example Configurations

With the following example configuration, uberAgent monitors disks only, i.e., hard drives and removable media. As `Hotplug` is set to **`true`**, uberAgent monitors volumes that were present when the agent was started as well as volumes mounted at runtime. Monitoring of network shares, mailslots, and named pipes is disabled.

```
[ProductComponents]
EnableESA = true

[FilesystemMonitoring]
Enabled = true
Hotplug = true
Monitor = Disks
```

With the following example configuration, uberAgent monitors disks and network shares. As `Hotplug` is not enabled, uberAgent only monitors volumes that were present when the agent was started. Monitoring of mailslots and named pipes is disabled.

```
[ProductComponents]
EnableESA = true

[FilesystemMonitoring]
Enabled = true
Monitor = Disks
Monitor = NetworkShares
```

File System Events in Threat Detection Engine (TDE)

File system events are available in TDE rules, whose queries are powered by uAQL. uAQL queries for file system events can make use of common event properties as well as file system event properties.

Example Rule for Windows

The following example detects writes to files in the directory `C:\Windows` (Windows):

```
[ActivityMonitoringRule]
# Detect writes to files in the directory C:\Windows.
RuleId = 2b635c8b-3c2b-4c51-b073-40a796d4ab9e-write
RuleName = Any file write
EventType = File.Write
Tag = file-write
RiskScore = 100
Query = File.Path like "%C:\\Windows\\%"
GenericProperty1 = File.Name
GenericProperty2 = File.CreationDate
GenericProperty3 = File.IsExecutable
GenericProperty4 = File.Path
```

Example Rule for macOS

The following example detects writes to files in the directory `/Library/LaunchAgents` (macOS):

```
[ActivityMonitoringRule]
# Detect writes to files in the directory /Library/LaunchAgents.
RuleId = 2b635c8b-3c2b-4c51-b073-40a796d4ab9e-write
RuleName = Any file write
EventType = File.Write
Tag = file-write
RiskScore = 100
```

```
Query = File.Path like "%/Library/LaunchAgents/%"
GenericProperty1 = File.Name
GenericProperty2 = File.CreationDate
GenericProperty3 = File.IsExecutable
GenericProperty4 = File.Path
```

Platform-Specific Notes

Windows

On Windows, uberAgent uses a minifilter driver to monitor the file system. Only successful file system activity is reported. Attempts that are not permitted due to missing permissions and attempts that are blocked by security products are not reported.

macOS

On macOS, uberAgent leverages the Endpoint Security Framework to monitor the file system. The Endpoint Security Framework provides the necessary APIs to capture a comprehensive range of events, including both successful and unsuccessful events. Different from the Windows implementation, on macOS uberAgent may capture unsuccessful events, too. Such events might result from permissions issues, for example. Specific results depend on the altitude at which the framework hooks the list of syscalls needed for file system activity.

Network Monitoring

The ESA Threat Detection rules for monitoring network activity are vast limits vendor rules.

Network Rules

The rules in this section detect suspicious behavior related to network operations.

- Suspicious network target names
- PowerShell outbound network connections
- Suspicious outbound Kerberos connections
- PowerShell remoting
- Detect network connects from suspicious sources
- Detect network connects from Windows processes
- Detect network connects from third-party tools
- RDP connects from non-RDP software, indicating lateral movement

- Detect network connects to suspicious ports
- Detect network connects to 80 and 443 from non-browser applications

MS Office & Acrobat Reader Monitoring

The ESA Threat Detection rules for monitoring Microsoft Office and Adobe Acrobat Reader are vast limits vendor rules.

Microsoft Office Rules

The rules in this section detect suspicious behavior with MS Office applications.

- Detect macro execution with the default Office security configuration applied (”disable all macros with notification”)
- Detect child processes of Microsoft Office applications (dedicated rules for scripts and other types of child processes)
- Detect Microsoft Office download operations
- Detect Microsoft Office applications executing macros that access WMI to create child processes
- Suspicious DLL load by Office
- Detect loading of MAPI DLLs from processes other than Outlook

Adobe Acrobat Reader Rules

The rules in this section detect suspicious behavior with Adobe Acrobat Reader.

- Detect child processes of Adobe Reader

LOLBAS Monitoring

LOLBAS stands for Living Off the Land Binaries And Scripts, a type of activity that misuses tools and executables that are already there because they are part of the operating system. To cite the LOLBAS’ project’s criteria, a LOLBin/lib/script must:

- Be a Microsoft-signed file, either native to the OS or downloaded from Microsoft.
- Have extra ”unexpected” functionality. It is not interesting to document intended use cases.
- Have functionality that would be useful to an APT or red team.

The ESA Threat Detection rules for monitoring LOLBAS activity are vast limits vendor rules.

LOLBAS Rules

The rules in this section detect suspicious behavior related to operating system binaries.

- Unusual child processes and DLL loads
- Detect starts from non-default locations
- Detect proxy execution
- Detect UAC bypass
- Detect csc/jsc compile
- Detect execute from alternate data streams
- Detect AWL bypass
- Detect encode and decode operations
- Detect copy operations
- Detect download operations

Process Tampering Monitoring

uberAgent ESA detects several malicious attack techniques such as Process Herpaderping and Process Hollowing. We name these attack techniques Process Tampering events.

Configuration

uberAgent ESA Process Tampering Monitoring is enabled or disabled through a configuration option. The related configuration Stanza is `[ProcessStartupSettings]`.

Configure the setting `EnableProcessTampering = false` to disable process tampering monitoring.

By default, this option is enabled (requires ESA enabled, too).

Detecting Process Tampering Events

Any process tampering action is queryable with uAQL and its Threat Detection Engine rules.

Example Rule

The following example detects any Process Tampering event and forwards it to your backend, once triggered.

```
[ActivityMonitoringRule]
# Detects any Process Tampering action
RuleName = Detects any Process Tampering action
EventType = Process.TamperingEvent
Tag = process-tampering
RiskScore = 75
Query = true
```

This example rule forwards any tampering event. You may filter this with more advanced conditions using Common Event Properties.

Introduction to Process Tampering

While there are a couple of different techniques the outcome is most likely the same. A malicious process is running in the context of a non-malicious process and tries to hide malicious actions in the context of this good process. There are many good resources available that explain these techniques in detail. To get you a short summary please refer to the notes below.

What is Process Hollowing

A process is launched in a suspended state and executable code is unmapped and replaced with malicious code and resumed.

What is Process Herpaderping

This technique requires replacing an executable binary with a malicious file and launching it. Then, the original file is restored and the malicious executable pretends to be the original one.

Remote Thread Monitoring

uberAgent ESA detects remote thread creation that may be used in malicious attack techniques or suspicious activities such as DLL injections or malicious code execution in remote processes.

Configuration

uberAgent ESA Remote Thread Monitoring is enabled or disabled through a configuration option. The related configuration Stanza is `[Miscellaneous]`.

Configure the setting `RemoteThreadMonitoring = false` to disable remote thread monitoring.

By default, this option is enabled (requires ESA enabled and Process Startup metrics enabled, too).

Detecting Remote Thread Events

Any remote thread action is queryable with uAQL and its Threat Detection Engine rules.

Example Rule

The following example detects any Remote Thread event and forwards it to your backend, once triggered.

```
[ActivityMonitoringRule]
# Detect any remote thread creation
RuleName = Detect remote thread creations
EventType = Process.CreateRemoteThread
Query = true
Tag = process-create-remote-thread
RiskScore = 75
GenericProperty1 = Thread.Id
GenericProperty2 = Thread.Timestamp
GenericProperty3 = Thread.StartAddress
GenericProperty4 = Thread.StartModule
GenericProperty5 = Thread.StartFunctionName
```

However, this general rule may include false positives. There are many cases where remote threads are used that are absolutely not malicious or suspicious. (e.g: debugging applications or OS remote threads)

Therefore it is recommended to filter this with more advanced conditions using Common Event Properties and Remote Event Properties.

uberAgent ESA is shipped with many automatically converted rules from Sigma. This ruleset already includes several useful Remote Thread detection rules.

Root CA certificate monitoring

The ESA Threat Detection rules for monitoring changes to root CA certificates are vast limits vendor rules.

The rules detect certificate chain cloning and cloned root trust attacks by monitoring writes to user and machine registry keys. For details, check the following rules:

- `Detect AuthRoot, CA and Root certificate changes per machine`
- `Detect AuthRoot, CA and Root certificate changes per user`

Security Descriptor & ACL Monitoring

The ESA Threat Detection rules for permissions (security descriptors and ACLs) are vast limits vendor rules.

File System ACL Rules

The rules in this section detect suspicious behavior related to file system permissions (ACLs).

- Detect processes started from directories that are user-writeable
- Detect process starts from directories with a low mandatory integrity label

Security Descriptor Monitoring Capabilities

uberAgent ESA has sophisticated features that make security descriptors, which can be a bit obscure and difficult to work with, much more accessible:

- SID to name lookup
- Conversion of hex access masks to permission strings

Please see this document for details.

Sigma Rules & Converter

uberAgent ESA ships with Threat Detection rules derived from Sigma signatures. These rules are grouped by severity: critical, high, medium, and low.

By their nature, Sigma rules are pretty dynamic and may change quickly. Our Sigma rule coverage explorer web app lets you browse and explore which Sigma rules are supported in which uberAgent version - and why. See below for instructions on how to convert Sigma rules yourself with our **Sigma converter**.

Sigma Rules

Following is an excerpt of some Sigma rules that ship with uberAgent ESA:

- Detect Ryuk ransomware command lines
- Detect DNS tunnel activity for Muddywater actor
- Detect a suspicious PowerShell command-line combination as used by APT29 in a campaign against US think tanks

- Detect Russian group activity as described in Global Threat Report 2019 by Crowdstrike
- Detect a suspicious DLL loading from `AppData\Local` as described in BlueMashroom report
- Detect Chafer activity attributed to OilRig as reported in Nyotron report in March 2018
- Detect CrackMapExecWin activity as described by NCSC
- Detect Elise backdoor activity as used by APT32
- Detect the execution of DLL side-loading malware used by threat group Emissary Panda aka APT27
- Detect a specific tool and export used by EquationGroup
- Detects Golden Chickens deployment method as used by Evilnum in a report published in July 2020
- Detect tools and process executions as observed in a Greenbug campaign in May 2020
- Detect Judgement Panda activity as described in Global Threat Report 2019 by Crowdstrike
- Detect registry modifications performed by Ke3chang malware in campaigns running in 2019 and 2020
- Detects Trojan loader activity as used by APT28
- ...and hundreds more

Not all Sigma rules are enabled by default.

Sigma Converter

vast limits maintains a Sigma to uberAgent rule converter as part of the Sigma project. The converter is implemented as a Sigma backend. Please see the header of uberAgent's Sigma rule files for instructions on how to invoke the conversion.

Sysmon Rule Converter

The Sysmon to uberAgent rule converter **translates Sysmon rules into the format used by uberAgent ESA**.

Sysmon is one of the most popular endpoint detection tools. Numerous quality rulesets are maintained by the security community. The Sysmon converter makes those rulesets available for use with uberAgent ESA.

uberAgent ESA as Sysmon Alternative

In addition to the Sysmon converter, vast limits also provides a converter for Sigma rules. This makes uberAgent one of the most versatile detection tools on the market. Read on to find out why you should consider uberAgent ESA as an alternative to Sysmon.

Getting Started

Download

To download the Sysmon converter, head over to the releases section of its GitHub repository.

Usage

Please see the Sysmon converter's readme for usage information.

Sysmon Converter Details

Development Info

The Sysmon converter is developed as an open-source project on GitHub. Contributions are always welcome!

The Sysmon converter is written in C# .NET. Release builds are available for Windows.

Threat Detection Engine

uberAgent ESA Threat Detection **makes system activity traceable** and searchable.

When a Threat Detection rule matches a risky process, an unusual network connection, or similar activity, uberAgent ESA creates an event in your SIEM (e.g., Splunk). Threat Detection's comprehensive, extensible ruleset is powered by uAQL, a feature-rich query language that is both easy to read by humans and fast to process by computers.

uberAgent ESA comes with hundreds of predefined rules for many common attack vectors and converters for Sysmon rules and Sigma signatures. Customizing and extending ESA's ruleset is explicitly encouraged.

Rule Sources

uberAgent ESA ships with rules from two different sources: vast limits rules and third-party rules. The former are curated by vast limits, while the latter are converted from sources such as the Sigma project.

Rule Storage

uberAgent ESA Threat Detection rules are part of uberAgent's configuration, which is maintained in the uberAgent Configuration GitHub repository.

Metadata

Annotation (MITRE ATT&CK Technique ID)

Annotations add supplementary data to Threat Detection rules, notably MITRE ATT&CK technique IDs.

Tag & Risk Score

Every ESA Threat Detection rule comes with a tag and a risk score that are assigned to matching events.

Sourcetype

ESA Threat Detection events are assigned the sourcetype `uberAgentESA:ActivityMonitoring:ProcessTagging` (see the metrics documentation for a description of the fields).

Visualization

ESA Threat Detection events are visualized in the *Threat Detection Events* dashboard, which is part of the `uberAgent_ESA` Splunk searchhead app.

Rule Syntax

uberAgent ESA's Threat Detection rules are part of the configuration. This page documents the rule syntax.

Example

The following example shows a simple rule that is triggered whenever a process is started (`EventType = Process.Start`). The rule's query checks if the started process'name is `wmiprvse.exe`. If that is the case, the rule matches, and an event with the tag `proc-start-wmiservice-child` is sent to the backend.

```
[ThreatDetectionRule]
RuleId = 0a1bbfbc-e0d9-4c49-953e-e31c3aa3fc91
RuleName = Detect child processes of the WMI service
EventType = Process.Start
Annotation = {
 "mitre_attack": ["T1071", "T1071.004"] }

Tag = proc-start-wmiservice-child
Query = Parent.Name == "wmiprvse.exe"
```

Rule Stanzas

There can be any number of `[ThreatDetectionRule]` stanzas, each defining one rule. Rules are processed in the order in which they are defined in the configuration. uberAgent ESA always processes all rules for every activity. This means that multiple events may be generated per activity.

Rule Components

An `[ThreatDetectionRule]` stanza may contain the following components.

RuleName

- **Setting name:** `RuleName`
- **Description:** any name to more easily identify a rule. Not used by uberAgent.
- **Valid values:** any string
- **Default:** empty
- **Required:** yes

EventType

- **Setting name:** `EventType`
- **Description:** the type of event this rule applies to.
- **Valid values:** see the event types page
- **Default:** empty
- **Required:** yes

Annotation

- **Setting name:** `Annotation`

- **Description:** one or more annotations for the event in JSON format
- **Valid values:**
  - Supported in Splunk Enterprise Security: `nist`, `kill_chain_phases`, `cis20`, `mitre_attack`, or any custom cyber security framework
  - Supported in uberAgent ESA: `mitre_attack`
- **Default:** empty
- **Required:** no

Hive

This setting is only valid for registry events.

- **Setting name:** `Hive`
- **Description:** a comma-separated list of registry hives that are matched against before evaluating the query. For best performance, only the necessary hives should be specified.
- **Valid values:**
  - `HKLM`, matches `HKEY_LOCAL_MACHINE`
  - `HKU`, matches `HKEY_USERS`
  - `A`, matches application hives
  - `WC`, matches App-V packages and UWP apps
  - `*`, matches events in any hive. Note that `*` has a high performance impact and should be avoided if possible.
- **Default:** empty
- **Required:** only if `EventType` is a registry event type.

Query

- **Setting name:** `Query`
- **Description:** a uAQL query string that is matched against the properties of the event. A rule is considered matching if the query returns **`true`**.
- **Valid values:** any uAQL query string
- **Default:** empty
- **Required:** yes

Tag

- **Setting name:** `Tag`
- **Description:** a tag assigned to events matching this rule.
- **Valid values:** any string
- **Default:** empty
- **Required:** yes

RuleId

- **Setting name:** `RuleId`
- **Description:** a unique id assigned to this rule.
- **Valid values:** any string, e.g.: 7098a059-4191-4a9e-973c-8976d61cddc0
- **Default:** empty
- **Required:** yes

RiskScore

- **Setting name:** `RiskScore`
- **Description:** a risk score assigned to events matching this rule.
- **Valid values:** any number from 0 to 100. If an invalid value is set, the rule is ignored.
- **Default:** 50
- **Required:** no

VerboseLogging

- **Setting name:** `VerboseLogging`
- **Description:** if enabled, more detail is added to the log file, e.g., the fully evaluated security descriptor if an SDDL rule is configured.
- **Valid values:** **`true`** or **`false`**
- **Default:** **`false`**
- **Required:** no

Rule Evaluation

Rules are evaluated by running the rule's query with the event properties as input.

Reusable uAQL Queries

Commonly used queries can be defined as expressions, a functionality that is similar to macros or functions in other languages. This gives you the flexibility to write query code only once and use it multiple times.

Example

The following declares the Threat Detection expression **ParentIsMsOffice** to identify Microsoft Office as a parent application.

```
[AddThreatDetectionExpression name=ParentIsMsOffice]
Query = istartswith(Parent.Company, "Microsoft") and Parent.Name in ["excel.exe", "msaccess.exe", "onenote.exe", "outlook.exe", "powerpnt.exe", "winword.exe"]
```

The expression **ParentIsMsOffice** is used in the following Threat Detection rule to identify child processes of Microsoft Office.

```
[ThreatDetectionRule]
Query = ParentIsMsOffice and (Process.Name in ["cmd.exe", "cscript.exe", "wscript.exe", "ftp.exe"] or ProcessIsPowerShell)
```

Please note that expressions must be defined *before* usage. Ideally, expressions are defined at the top of the configuration file. Also, please make sure not to use reserved keywords as expression names.

Disabling Default Rules

To disable any of the default rules, create a post-processing rule as follows:

```
[ThreatDetectionRuleExtension RuleId=RULE_ID]
Name = Disable the rule with ID RULE_ID
Query = false
```

Example

To disable the rule with the ID `7098a059-4191-4a9e-973c-8976d61cddc0`, add the following stanza to any uberAgent configuration file:

```
[ThreatDetectionRuleExtension RuleId=7098a059-4191-4a9e-973c-8976d61cddc0]
Name = Disable the rule with ID 7098a059-4191-4a9e-973c-8976d61cddc0
Query = false
```

Handle False Positives with Post-Processing

Consider the following default rule, which, unfortunately, also matches certain Splunk processes:

```
[ThreatDetectionRule]
RuleId = bdc64095-d59a-42a2-8588-71fd9c9d9abc
RuleName = Suspicious Unsigned Dbghelp/Dbgcore DLL Loaded
EventType = Image.Load
Tag = suspicious-unsigned-dbghelp/dbgcore-dll-loaded
RiskScore = 75
Annotation = {
 " mitre_attack" : [" T1003.001" ] }

Query = ((Image.Path like r" %\\dbghelp.dll"  or Image.Path like r" %\\dbgcore.dll" ) and Image.IsSigned == false)
```

To prevent the above rule from matching your Splunk processes, add the following post-processing stanza to any uberAgent configuration file:

```
[ThreatDetectionRuleExtension RuleId=bdc64095-d59a-42a2-8588-71fd9c9d9abc]
Query = Rule.Result and Parent.Name != "Splunkd.exe"
```

The above rule extension does what its name implies: it extends the original rule's uAQL query with additional statements, which is ideal for adding exclusions.

An important feature of how we've set up rule post-processing is its flexibility. Extension sections can be used not just for one rule at a time, but they can also be applied to groups of rules together. For example, a certain extension section could be set up to affect all rules that deal with network events:

```
[ThreatDetectionRuleExtension EventType=Net.Any]
Query = Rule.Result and Parent.Name != "Splunkd.exe"
```

Rule Editor: uAQL Studio

uAQL Studio is a free online tool to learn, build and test uberAgent ESA Threat Detection rules.

Quickstart

Follow these steps to get started with uAQL Studio quickly:

1. Open uAQL Studio in a new browser tab
2. Select an event type
3. Add events to test the uAQL query your going to write
4. Work on your uAQL query until it matches your test events correctly

5. Fill out the fields `Rule name`, `Tag`, `Risk score`
6. Copy the rule definition from the preview pane to the clipboard and paste it into your uberAgent ESA configuration

Walkthrough

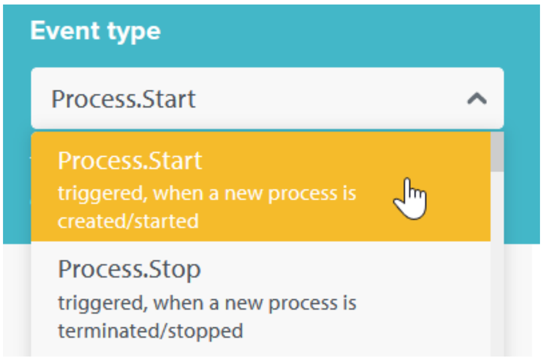
In this walkthrough, we're building a Threat Detection rule that detects script child processes of Microsoft Office applications.

Launch uAQL Studio

Click the following link to launch uAQL Studio in a new browser tab: https://uberagent.com/uaql-studio/.

Select an Event Type

uberAgent ESA Threat Detection supports many different types of events, including process, network, registry, and DNS events. Select the **Process.Start** event type:



Event type

Process.Start

Process.Start
triggered, when a new process is created/started

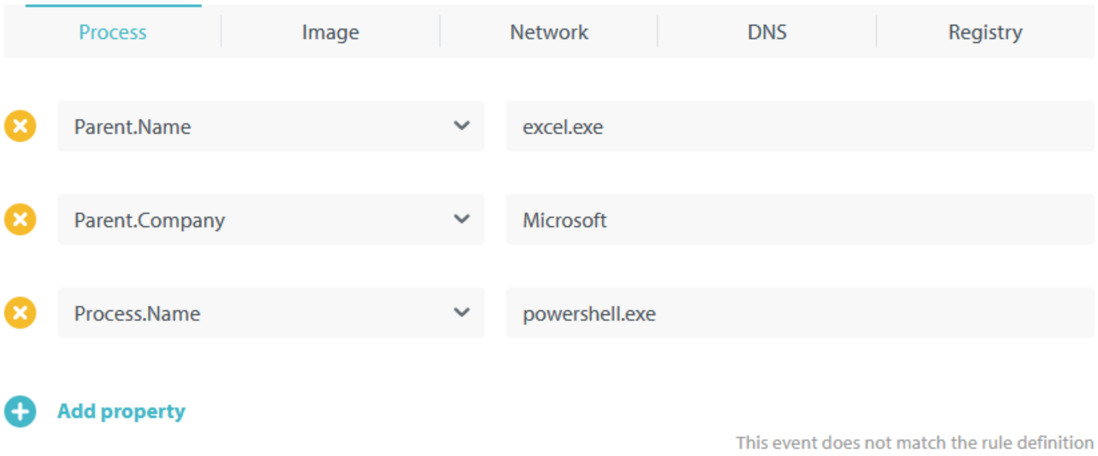
Process.Stop
triggered, when a new process is terminated/stopped

Add a Test Event

Click **Add event** and specify the properties of an event you want uAQL Studio to match against the uAQL query you're going to write. Since we're creating a rule to detect script child processes of MS Office apps we're adding the properties of just such an event:

- **Parent.Name:** `excel.exe`
- **Parent.Company:** `Microsoft`
- **Process.Name:** `powershell.exe`

It should look similar to this:



Parent.Name: excel.exe

Parent.Company: Microsoft

Process.Name: powershell.exe

This event does not match the rule definition

As you can see in the lower right corner of the above screenshot, at this point the test event is not matched by the rule definition (aka, the query). That is correct, of course. After all, the query is still empty!

Define the uAQL Query

Detecting Child Processes of MS Office Apps Let's start to work on our detection query by first creating a snippet that identifies child processes of MS Office apps. Please see the documentation for details. The following properties should work:

1. the company name of the parent process needs to start with `Microsoft` (which also covers "Microsoft Corporation" and variants like "Microsoft Corp.", all of which can be found in the wild)
2. the parent process name must be one of the following: `excel.exe`, `msaccess.exe`, `onenote.exe`, `outlook.exe`, `powerpnt.exe`, `winword.exe`

In uAQL code the above looks as follows:

```
istartswith(Parent.Company, "Microsoft") and Parent.Name in ["excel.exe", "msaccess.exe", "onenote.exe", "outlook.exe", "powerpnt.exe", "winword.exe"]
```

Detecting Script Processes We define the following executables as script runtimes:

1. **PowerShell:** `powershell.exe`, `pwsh.exe`
2. **Batch, VBS, miscellaneous:** `cmd.exe`, `cscript.exe`, `wscript.exe`, `ftp.exe`

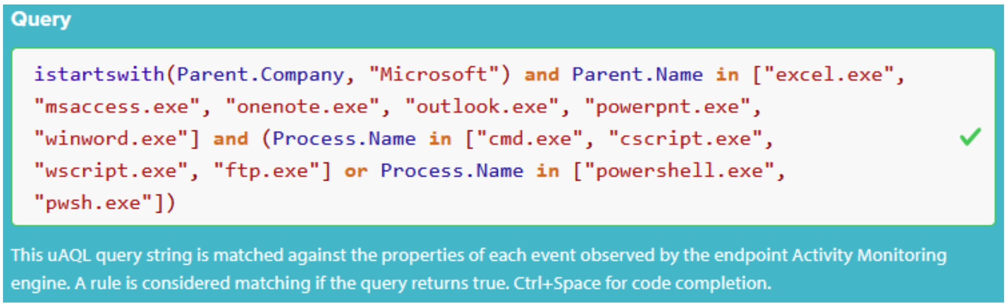
In uAQL code the above looks as follows:

```
Process.Name in ["cmd.exe", "cscript.exe", "wscript.exe", "ftp.exe"] or Process.Name in ["powershell.exe", "pwsh.exe"]
```

Putting the Query Together We can now combine the two snippets into the complete uAQL query:

```
istartswith(Parent.Company, "Microsoft") and Parent.Name in ["excel.exe", "msaccess.exe", "onenote.exe", "outlook.exe", "powerpnt.exe", "winword.exe"] and (Process.Name in ["cmd.exe", "cscript.exe", "wscript.exe", "ftp.exe"] or Process.Name in ["powershell.exe", "pwsh.exe"])
```

Paste the Query Into uAQL Studio Copy the completed query from above and paste it into the **Query** field in uAQL Studio. The built-in syntax checking mechanism should show a green checkmark similar to the following:

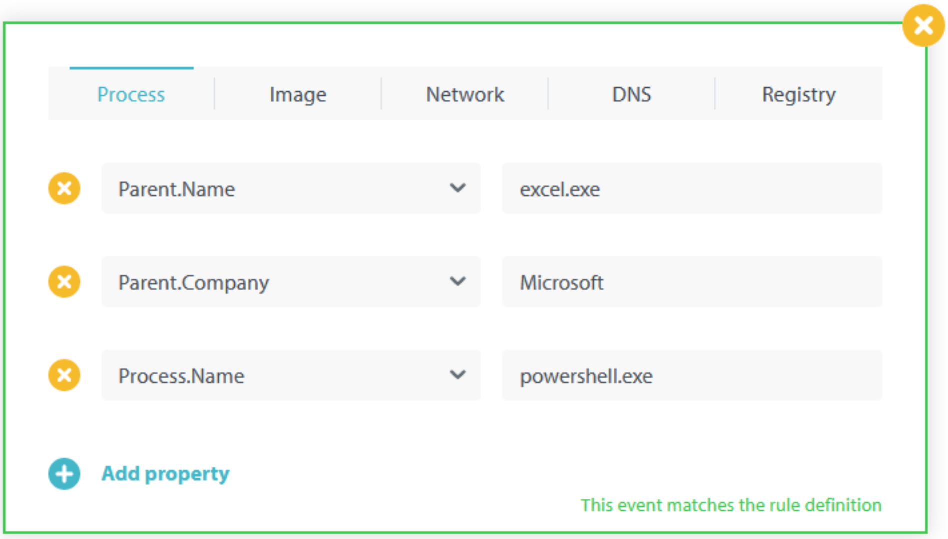


```
istartswith(Parent.Company, "Microsoft") and Parent.Name in ["excel.exe",
"msaccess.exe", "onenote.exe", "outlook.exe", "powerpnt.exe",
"winword.exe"] and (Process.Name in ["cmd.exe", "cscript.exe",
"wscript.exe", "ftp.exe"] or Process.Name in ["powershell.exe",
"pwsh.exe"])
```

✓

This uAQL query string is matched against the properties of each event observed by the endpoint Activity Monitoring engine. A rule is considered matching if the query returns true. Ctrl+Space for code completion.

Verify the Test Event Matches Take a look at the test event we created earlier. The green border indicates that it matches the query. The text in the lower right corner has changed to *This event matches the rule definition*:



Parent.Name	excel.exe
Parent.Company	Microsoft
Process.Name	powershell.exe

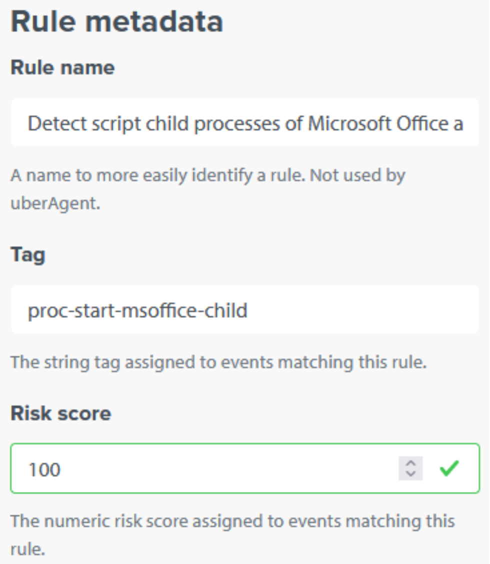
This event matches the rule definition

Add Rule Metadata

The rule definition is now complete but we need to add some metadata to make it usable with uberAgent and Splunk. Please see the documentation for details. The following values work well:

- **Rule name:** `Detect script child processes of Microsoft Office applications`
- **Tag:** `proc-start-msoffice-child`
- **Risk score:** `100`

The UI should now look similar to the following:



Rule metadata

Rule name

Detect script child processes of Microsoft Office a

A name to more easily identify a rule. Not used by uberAgent.

Tag

proc-start-msoffice-child

The string tag assigned to events matching this rule.

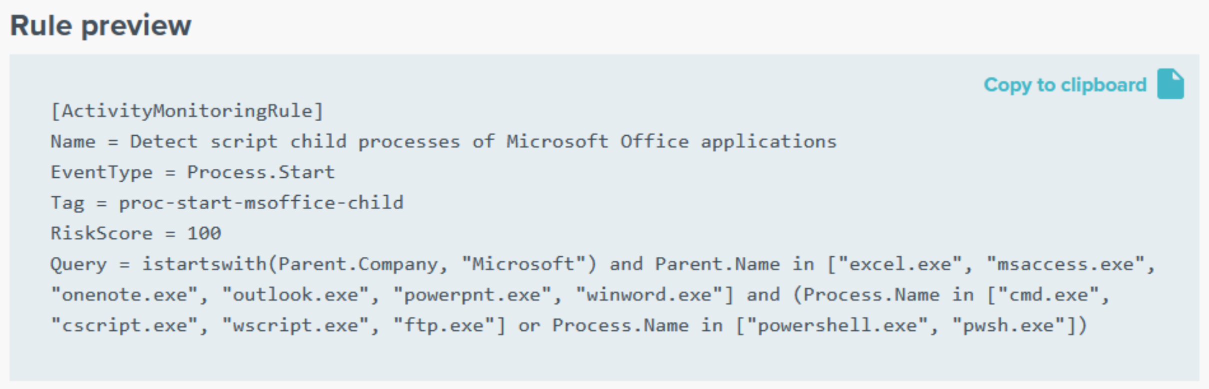
Risk score

100 ✓

The numeric risk score assigned to events matching this rule.

Copy the Rule Definition

With this, the rule definition is complete. We can copy it from the preview pane and use it with uberAgent ESA!



Rule preview

```
[ActivityMonitoringRule]
Name = Detect script child processes of Microsoft Office applications
EventType = Process.Start
Tag = proc-start-msoffice-child
RiskScore = 100
Query = istartswith(Parent.Company, "Microsoft") and Parent.Name in ["excel.exe", "msaccess.exe",
"onenote.exe", "outlook.exe", "powerpnt.exe", "winword.exe"] and (Process.Name in ["cmd.exe",
"cscript.exe", "wscript.exe", "ftp.exe"] or Process.Name in ["powershell.exe", "pwsh.exe"])
```

Event Types

uberAgent ESA's Threat Detection rules can be triggered by many different types of events.

Event types are specified in the `EventType` component of `[ActivityMonitoringRule]` stanzas (rule syntax).

Process & Image Events

Event Types

The following process event types are available:

- `Process.Start`: a new process is created/started.
- `Process.Stop`: a process is terminated/stopped.
- `Process.CreateRemoteThread`: a process is starting a thread in another process.
- `Process.TamperingEvent`: a process tampering event is detected.
- `Image.Load`: an executable image (e.g., a DLL) is loaded.
- `Driver.Load`: a kernel image (e.g., a driver) is loaded.

Event Properties

Common event properties are available with all types of events. Remote thread creation events and image load events have additional properties.

Network Events

Event Types

The following network event types are available:

- `Net.Send`: a network packet is sent.
- `Net.Receive`: a network packet is received.
- `Net.Connect`: a network connection is established.
- `Net.Reconnect`: a network connection is re-established.
- `Net.Retransmit`: a network packet is retransmitted (sent again).

Event Properties

Please see the documentation for the properties of network events.

Registry Events

Event Types

The following registry event types are available:

- `Reg.Key.Create`: a registry key is created.
- `Reg.Value.Write`: a registry value is written. This includes registry value creation as well as changes to the value's name and data.
- `Reg.Delete`: a registry key or value is deleted.
- `Reg.Key.Delete`: a registry key is deleted.
- `Reg.Value.Delete`: a registry value is deleted.
- `Reg.Key.SecurityChange`: a registry key's security descriptor is changed.
- `Reg.Key.Rename`: a registry key is renamed.
- `Reg.Key.SetInformation`: a registry key metadata is changed (e.g. last-write time, tags, virtualization, etc.).
- `Reg.Key.Load`: a registry hive is loaded.
- `Reg.Key.Unload`: a registry hive is unloaded.
- `Reg.Key.Save`: a registry key is saved.
- `Reg.Key.Restore`: a registry key is restored.
- `Reg.Key.Replace`: a registry key is replaced.
- `Reg.Any`: any of the above.

Event Properties

Please see the documentation for the properties of registry events.

DNS Query Events

Event Types

The following DNS query event types are available:

- `DNS.Query`: an outgoing DNS query request has completed, and a response has been received.

Event Properties

Please see the documentation for the properties of DNS query events.

File System Events

Event Types

The following file system activity event types are available:

- `File.ChangeCreationTime`: a file's original creation timestamp is changed (available on Windows only).
- `File.Create`: a file is created.
- `File.CreateStream`: an alternate data stream (ADS) is created (available on Windows only).
- `File.Delete`: a file is deleted (all platforms)
- `File.PipeCreate`: a named pipe is created.
- `File.PipeConnected`: a client connects to a named pipe.
- `File.RawAccessRead`: raw read access to a target device, bypassing ACLs (available on Windows only).
- `File.Rename`: a file is renamed.
- `File.Write`: a file is being written to.
- `File.Read`: a file is being read from.

Event Properties

Please see the documentation for the properties of file system activity events.

Common Event Properties

The following event properties can be used with all types of events in uAQL queries.

Property name	uAQL Data Type	Description	Platform
`Process.Id`	String	The process'id (e.g., `148`)	all
`Parent.Id`	String	The process'parent's id (e.g., 4)	all
`Process.Name`	String	The process'image file name (e.g., `Winword.exe`)	all
`Parent.Name`	String	The process'parent's image file name (e.g., `Winword.exe`)	all

Property name	uAQL Data Type	Description	Platform
Process.User.Sid	String	The process'user SID	Win
Process.User	String	The process'user name in the format domain\account	all
Parent.User.Sid	String	The process'parent's user SID	Win
Parent.User	String	The process'parent's user name. Format on Windows: domain\account	all
Process.Path	String	The process'full path including the image file name	all
Parent.Path	String	The process'parent's full path including the image file name	all
Process.CommandLine	String	The process'command line	all
Parent.CommandLine	String	The process'parent's command line	all
Process.AppName	String	The process' application name (e.g., Microsoft Office)	all
Parent.AppName	String	The process'parent's application name (e.g., Microsoft Office)	all
Process.AppVersion	String	The process' application version	all
Parent.AppVersion	String	The process'parent's application version	all
Process.Company	String	The process'company (as stored in the PE image resources)	Win

Property name	uAQL Data Type	Description	Platform
Parent.Company	String	The process'parent's company (as stored in the PE image resources)	Win
Process.IsElevated	Boolean	Is the process elevated?	all
Parent.IsElevated	Boolean	Is the parent process elevated?	all
Process.IsProtected	Boolean	Is the process protected?	Win
Parent.IsProtected	Boolean	Is the parent process protected?	Win
Process.SessionId	Integer	The process'session ID	all
Parent.SessionId	Integer	The process'parent's session ID	all
Process.DirectorySdSddl	String	The security descriptor (SD) of the process' directory. The SD is converted to the security descriptor string format (SDDL) for the match. NULL SDs, which grant full access to everyone, are represented as [UA_NULL_SD]. SIDs in the SD are looked up and replaced with names. Hex access masks are replaced with their string representations in SetACL's format (details).	Win

Property name	uAQL Data Type	Description	Platform
`Process.DirectoryUserWriteable`	Boolean	Is the process' directory writeable by the user that is logged on the session the process is started in? Ignores processes in session 0.	Win
`Process.Hash.MD5`	String	MD5 hash of the process executable	Win
`Process.Hash.SHA1`	String	SHA1 hash of the process executable	Win
`Process.Hash.SHA256`	String	SHA256 hash of the process executable	Win
`Process.Hash.IMP`	String	Import-table hash of the process executable	Win
`Process.Hashes`	String	All enabled hashes for process are output comma-separated, e.g.: `MD5=CFCD208495D565EF66E7DFF9F98764DA,SHA1=B6589FC6AB0DC82CF12099D1C2D40AB994E8410C`	Win
`Parent.Hash.MD5`	String	MD5 hash of the parent process executable	Win
`Parent.Hash.SHA1`	String	SHA1 hash of the parent process executable	Win
`Parent.Hash.SHA256`	String	SHA256 hash of the parent process executable	Win
`Parent.Hash.IMP`	String	Import-table hash of the parent process executable	Win

Property name	uAQL Data Type	Description	Platform
`Parent.Hashes`	String	All enabled hashes for parent process are output comma-separated, e.g.: `MD5=CFCD208495D565EF66E7DFF9F98764DA,SHA1=B6589FC6AB0DC82CF12099D1C2D40AB994E8410C`	Win
`Process.IsSigned`	Boolean	Is the process signed? This evaluates to **`true`** even if the certificate was revoked or is expired.	Win
`Process.IsSignedByOSVendor`	Boolean	Is the process signed by the vendor of the operating system (e.g. Microsoft)? This evaluates to **`true`** even if the certificate was revoked or is expired.	all
`Process.Signature`	String	The signer name.	Win
`Process.SignatureStatus`	String	Evaluates to `Valid` for a valid certificate and, under Windows, `Invalid` for an invalid certificate. Furthermore, it evaluates to `SelfSigned` under macOS if the binary is ad-hoc signed. It is empty if the process is not signed.	all

Property name	uAQL Data Type	Description	Platform
`Process.SigningId`	String	The unique identifier associated with the developer's certificate used for signing the bundle or binary.	macOS
`Process.TeamId`	String	A unique identifier assigned by Apple to a specific development team.	macOS
`Process.CdHash`	String	The process's code directory hash.	macOS
`Parent.IsSigned`	Boolean	Is the parent process signed? This evaluates to **`true`** even if the certificate was revoked or is expired.	Win
`Parent.IsSignedByOSVendor`	Boolean	Is the parent process signed by the vendor of the operating system (e.g. Microsoft)? This evaluates to **`true`** even if the certificate was revoked or is expired.	all
`Parent.Signature`	String	The signer name.	Win

Property name	uAQL Data Type	Description	Platform
Parent.SignatureStatus	String	Evaluates to Valid for a valid certificate and, under Windows, Invalid for an invalid certificate. Furthermore, it evaluates to SelfSigned under macOS if the binary is ad-hoc signed. It is empty if the parent process is not signed.	all
Parent.SigningId	String	The parent process's unique identifier associated with the developer's certificate used for signing the bundle or binary.	macOS
Parent.TeamId	String	The parent process's unique identifier assigned by Apple to a specific development team.	macOS
Parent.CdHash	String	The parent process's code directory hash.	macOS

Note for macOS

As all binaries for macOS on Apple Silicon are signed, Process.IsSigned and Parent.IsSigned are always true. To reflect if a binary is ad-hoc signed (i.e. there is no valid certificate included) Process.SignatureStatus, respectively Parent.SignatureStatus, are set to SelfSigned. If the binary is not ad-hoc signed those fields are set to valid. If the binary is not signed at all (e.g. because it is an Intel binary running under Rosetta 2) those fields are empty.

In case a process is already running before uberAgent is started, the following fields might be unavailable:

- *.CdHash
- *.IsSigned

- *.IsSignedByOSVendor
- *.SignatureStatus
- *.SigningId
- *.TeamId

As soon as the affected process calls fork or exec the values are available.

DNS Query Event Properties

The following event properties can be used with DNS query events in uAQL queries (event type DNS.Query). In addition to the properties listed here, the common properties are applicable, too.

Property name	uAQL Data Type	Description	Platform
Dns.QueryRequest	String	The requested DNS name.	all
Dns.QueryResponse	String	Responses to the request. May be a semicolon ; separated list of multiple responses.	all

File System Activity Event Properties

You can use the following event properties with file system events in uAQL queries (event type File.*). In addition to the properties listed here, the common properties are applicable, too.

Property name	uAQL Data Type	Description	Platform
File.CreationDate	String	The file creation date (format: Unix epoch timestamp in ms).	all
File.HasExecPermissions	Boolean	Determined by checking the file's executable bits. Available for all event types.	macOS

Property name	uAQL Data Type	Description	Platform
`File.IsExecutable`	Boolean	Indicates whether the file is an executable. **Windows:** determined by inspecting the file's first two bytes. Available for the following event types: `File.Create`, `File.Delete`, `File.ChangeCreationTime`.**macOS:** determined by inspecting the file's first four bytes. Available for the following event types: `File.ChangeCreationTime`, `File.Create`, `File.Rename`, `File.Write`.	all
`File.Name`	String	The file name only, without the path.	all
`File.Path`	String	The full path including the file name.	all
`File.PreviousCreationDate`	String	In case of `File.ChangeCreationTime` events: the original creation date.	Windows
`File.PreviousName`	String	In case of `File.Rename` events: the original file name.	all

`File.PreviousPath`	String	In case of `File.Rename` events: the original file path including the file name.	all

Image Load Event Properties

The following event properties can be used with image load events in uAQL queries (event type `Image.Load` and `Driver.Load`). In addition to the properties listed here, the common properties are applicable, too.

Property name	uAQL Data Type	Description	Platform
`Image.Name`	String	The image’s file name (e.g., `userenv.dll`)	Win
`Image.Path`	String	The image’s full path including the image file name	Win
`Image.Hash.MD5`	String	MD5 hash of the image	Win
`Image.Hash.SHA1`	String	SHA1 hash of the image	Win
`Image.Hash.SHA256`	String	SHA256 hash of the image	Win
`Image.Hash.IMP`	String	Import-table hash of the image	Win
`Image.Hashes`	String	All enabled hashes for image are output comma-separated, e.g.: `MD5=CFCD208495D565EF66E7DFF9F98764DA,SHA1=B6589FC6AB0DC82CF12099D1C2D40AB994E8410C`	Win

Property name	uAQL Data Type	Description	Platform
`Image.IsSigned`	Boolean	Is the image signed? This evaluates to **`true`** even if the certificate was revoked or is expired.	Win
`Image.IsSignedByOSVendor`	Boolean	Is the image signed by the vendor of the operating system (e.g. Microsoft)? This evaluates to **`true`** even if the certificate was revoked or is expired.	Win
`Image.Signature`	String	The signer name.	Win
`Image.SignatureStatus`	String	Evaluates to `Valid` for a valid certificate and `Invalid` for an invalid certificate. It is empty if the image is not signed.	Win

Network Event Properties

The following event properties can be used with network events in uAQL queries (event type `Net.*`). In addition to the properties listed here, the common properties are applicable, too.

Property name	uAQL Data Type	Description	Platform
`Net.Target.Ip`	String	The network target's IP address (IPv4 or IPv6)	all
`Net.Target.IpIsV6`	Boolean	Indicates whether the network target's IP address is IPv6 (true) or not (false).	all

Property name	uAQL Data Type	Description	Platform
`Net.Target.Name`	String	The network target's name. This is only available if it can be resolved from the local DNS cache.	all
`Net.Target.Port`	Integer	The network target's port	all
`Net.Target.PortName`	String	A textual representation of the network target port, e.g: https	all
`Net.Target.Protocol`	String	Either TCP or UDP	all
`Net.Source.Ip`	String	The network source IP address (IPv4 or IPv6)	all
`Net.Source.IpIsV6`	Boolean	Indicates whether the network source IP address is IPv6 (true) or not (false).	all
`Net.Source.Name`	String	The network source name. This is only available if it can be resolved from the local DNS cache.	all
`Net.Source.Port`	Integer	The network source port	all
`Net.Source.PortName`	String	A textual representation of the network source port, e.g: https	all

Registry Event Properties

The following event properties can be used with registry events in uAQL queries (event type `Reg.*`). In addition to the properties listed here, the common properties are applicable, too.

Property name	uAQL Data Type	Description	Platform
`Reg.Key.Path`	String	The absolute path of the registry key (e.g., `^HKLM\\SYSTEM\\.*ControlSet.*\\Services\\lmhosts$`). Not supported for `Reg.Key.Rename`.	Win
`Reg.Key.Name`	String	The name of the registry key - the last path element of the full path (e.g., `^lmhosts$`). Not supported for `Reg.Key.Rename`.	Win
`Reg.Parent.Key.Path`	String	The absolute path to the parent key (e.g., `^HKLM\\SYSTEM\\.*ControlSet.*\\Services$`). Not supported for `Reg.Key.Rename`.	Win
`Reg.Key.Path.New`	String	The new absolute path of the registry key (e.g., `^HKLM\\SYSTEM\\.*ControlSet.*\\Services\\lmhosts$`). Only supported for `Reg.Key.Rename`.	Win

Property name	uAQL Data Type	Description	Platform
`Reg.Key.Path.Old`	String	The old absolute path of the registry key (e.g., `^HKLM\\SYSTEM\\.*ControlSet.*\\Services\\lmhosts$`). Only supported for `Reg.Key.Rename`.	Win
`Reg.Value.Name`	String	The name of a key property (e.g., `RequiredPrivileges`).	Win

Property name	uAQL Data Type	Description	Platform
`Reg.Value.Data`	String	The value is formatted to be compatible with Sysmon. **DWORD values** are formatted with a hexadecimal representation, for example: `DWORD (0x00000001)`. **QWORD values** are shown in a range format, such as: `QWORD (0x00000001-0x00000002)`. **Empty Strings** are denoted as: `(Empty)`. **Binary Data** and **Multiline Strings**, including **Empty Multiline Strings**, are all represented as: `Binary Data`. **Regular Strings** remain unchanged. **Expandable Strings** have any percent (%) characters escaped, so `%PATH%` becomes `%%PATH%%`.	Win
`Reg.Value.Data.Number`	Number	Access to the non-formatted `DWORD` and `QWORD` registry values as number.	Win
`Reg.Value.Data.String`	String	Access to the non-formatted registry value strings.	Win

Property name	uAQL Data Type	Description	Platform
`Reg.Value.Type`	Number	The numeric value represents the data-type of the content written to the registry value. Possible values include: `0` = REG_NONE 1 = REG_SZ 2 = REG_EXPAND_SZ 3 = REG_BINARY 4 = REG_DWORD 4 = REG_DWORD_LITTLE_ENDIAN 5 = REG_DWORD_BIG_ENDIAN 6 = REG_LINK 7 = REG_MULTI_SZ 8 = REG_RESOURCE_LIST 9 = REG_FULL_RESOURCE_DESCRIPTOR `10` = REG_RESOURCE_REQUIREMENTS_LIST `11` = REG_QWORD `11` = REG_QWORD_LITTLE_ENDIAN For more details, see the Microsoft documentation.	Win
`Reg.EventType`	String	The Event Type identifies the actual registry event. Possible values include: SetValue DeleteValue RenameKey DeleteKey CreateKey	Win

Property name	uAQL Data Type	Description	Platform
`Reg.File.Name`	String	A file path (e.g., `C:\TempHive.hiv`). Supported for `Reg.Key.Load`, `Reg.Key.Restore`, `Reg.Key.Save`, or `Reg.Key.Replace`.	Win
`Reg.Key.Sddl`	String	The security descriptor (SD) of a registry key.	Win
`Reg.Key.Hive`	String	The name of the Hive (e.g., `HKLM`).	Win
`Reg.Key.Target`	String	The absolute path of the registry key. Takes `Reg.Key.Path.Old` or `Reg.Key.Path` and is thus never empty.	Win
`Reg.TargetObject`	String	This property is either the full path to the registry key or the full path to the registry value.	Win

Remote Thread Event Properties

The following event properties can be used with create remote thread events in uAQL queries (event type `Process.CreateRemoteThread`). In addition to the properties listed here, the common properties are applicable, too.

Property name	uAQL Data Type	Description	Platform
`Thread.Id`	Integer	The thread identifier of the newly created thread.	Win

Property name	uAQL Data Type	Description	Platform
`Thread.Timestamp`	Integer	Event Timestamp	Win
`Thread.Process.Id`	Integer	The process identifier of the process that runs the newly created thread.	Win
`Thread.Parent.Id`	Integer	The process identifier of the process that has initiated the remote thread.	Win
`Thread.StartAddress`	Integer	The absolute address in virtual memory where the function is located.	Win
`Thread.StartModule`	String	The name of the library where the function that was started is located in.	Win
`Thread.StartFunctionName`	String	The name of the function that was started as entry point for the new thread.	Win

Process.DirectorySdSddl

The event property `Process.DirectorySdSddl` is a powerful tool: it makes the **file system permissions** of a process'directory available for regex matching and rule evaluation.

Overview

Whenever an event occurs, uberAgent ESA checks if the event property `Process.DirectorySdSddl` is configured in at least one Threat Detection rule. If that is the case, uberAgent ESA does the following:

1. determine the **directory** of the process executable
2. read the directory's **security descriptor** (SD)

3. convert the SD to the **SDDL string** format
4. in the SDDL string, **replace SIDs** with user/group names
5. in the SDDL string, replace hex access masks with **readable permissions** strings
6. **match** the resulting string against the rule's regex

Security Descriptor Components

Security Descriptors are structures that contain multiple components, some of which are optional:

- Owner
- Primary group (rarely used, if at all)
- DACL (permissions)
- SACL (auditing configuration)
- Attributes (claims)
- Mandatory integrity label
- Scoped policy ID

uberAgent ESA retrieves all of the SD components shown above.

How It Works in Detail

Converting the SD to the SDDL String Format

Security descriptors are binary structures. In order for humans to read or regular expressions to match their contents, SDs must be converted to strings. Microsoft established a common format for that purpose, the Security Descriptor Definition Language.

uberAgent ESA converts all the security descriptor components to SDDL, but it does not stop there because **SDDL has shortcomings.**

Converting User/Group SIDs to Names

SDDL strings are more or less 1:1 representations of the binary SD structure. This means that, with very few exceptions, users and groups are not shown by their names, but by their SIDs, for example `S-1-5-21-3803133166-2955000686-238773884-1029`. Such a SID string is not very useful for regex matching, so uberAgent goes ahead and converts it to the well-known `domain\user` format before performing the regex matching.

Converting Hex Access Masks to Permission Strings

The same is true for access masks, which store the actual permission in a 32-bit unsigned integer. In a raw SDDL string, an access mask might look like this: `0x1200a9`. That is not very useful for regex matching because multiple permissions can be combined in one access mask through bitwise OR. Again, uberAgent does the heavy lifting by converting access masks to a string format that is processed easily: SetACL's. With this conversion, the cryptic access mask `0x1200a9` becomes the easily understandable string `read_execute`.

If an access mask contains a combination of multiple individual permissions, uberAgent's SetACL string lists all the individual permission names separated by commas.

Example

SDDL string for `C:\Windows\System32` **as obtained by the Windows API** before uberAgent ESA's simplifications:

```
O:S-1-5-80-956008885-3418522649-1831038044-1853292631-2271478464G:S-1-5-80-956008885-3418522649-1831038044-1853292631-2271478464D:PAI(A;;FA;;;S-1-5-80-956008885-3418522649-1831038044-1853292631-2271478464)(A;CIIO;GA;;;S-1-5-80-956008885-3418522649-1831038044-1853292631-2271478464)(A;;0x1301bf;;;SY)(A;OICIIO;GA;;;SY)(A;;0x1301bf;;;BA)(A;OICIIO;GA;;;BA)(A;;0x1200a9;;;BU)(A;OICIIO;GXGR;;;BU)(A;OICIIO;GA;;;CO)(A;;0x1200a9;;;AC)(A;OICIIO;GXGR;;;AC)(A;;0x1200a9;;;S-1-15-2-2)(A;OICIIO;GXGR;;;S-1-15-2-2)S:AINO_ACCESS_CONTROL
```

The same string **after uberAgent ESA replaced SIDs with names:**

```
O:NT SERVICE\TrustedInstallerG:NT SERVICE\TrustedInstallerD:PAI(A;;FA;;;NT SERVICE\TrustedInstaller)(A;CIIO;GA;;;NT SERVICE\TrustedInstaller)(A;;0x1301bf;;;SY)(A;OICIIO;GA;;;SY)(A;;0x1301bf;;;BA)(A;OICIIO;GA;;;BA)(A;;0x1200a9;;;BU)(A;OICIIO;GXGR;;;BU)(A;OICIIO;GA;;;CO)(A;;0x1200a9;;;AC)(A;OICIIO;GXGR;;;AC)(A;;0x1200a9;;;APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES)(A;OICIIO;GXGR;;;APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES)S:AINO_ACCESS_CONTROL
```

The same string after uberAgent ESA additionally **replaced hex access masks with permission strings:**

```
O:NT SERVICE\TrustedInstallerG:NT SERVICE\TrustedInstallerD:PAI(A;;full;;;NT SERVICE\TrustedInstaller)(A;CIIO;full;;;NT SERVICE\TrustedInstaller)(A;;change;;;SY)(A;OICIIO;full;;;SY)(A;;change;;;BA)(A;OICIIO;full;;;BA)(A;;read_execute;;;BU)(A;OICIIO;read_execute;;;BU)(A;OICIIO;full;;;CO)(A;;read_execute;;;AC)(A;OICIIO;read_execute;;;AC)(A;;read_execute;;;APPLICATION PACKAGE AUTHORITY\ALL
RESTRICTED APPLICATION PACKAGES)(A;OICIIO;read_execute;;;APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES)S:AINO_ACCESS_CONTROL
```

Deconstructing the SDDL String

Here is a quick explanation of the security descriptor string format. For the full specifications please see Microsoft's documentation.

Split into the SD's components, the SDDL string from the example above is already much more readable:

```
O:NT SERVICE\TrustedInstaller
G:NT SERVICE\TrustedInstaller
D:PAI(A;;full;;;NT SERVICE\TrustedInstaller)(A;CIIO;full;;;NT SERVICE\TrustedInstaller)(A;;change;;;SY)(A;OICIIO;full;;;SY)(A;;change;;;BA)(A;OICIIO;full;;;BA)(A;;read_execute;;;BU)(A;OICIIO;read_execute;;;BU)(A;OICIIO;full;;;CO)(A;;read_execute;;;AC)(A;OICIIO;read_execute;;;AC)(A;;read_execute;;;APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES)(A;OICIIO;read_execute;;;APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES)
S:AINO_ACCESS_CONTROL
```

As you can see, the four main components of a security descriptor are prepended by the following:

- `O:`: owner
- `G:`: primary group
- `D:`: DACL
- `S:`: SACL

The DACL part of an SDDL string is a concatenation of **access control entries (ACEs)**, each wrapped in parentheses. In this example, there are 13 ACEs in the ACL:

```
(A;;full;;;NT SERVICE\TrustedInstaller)
(A;CIIO;full;;;NT SERVICE\TrustedInstaller)
(A;;change;;;SY)
(A;OICIIO;full;;;SY)
(A;;change;;;BA)
(A;OICIIO;full;;;BA)
(A;;read_execute;;;BU)
(A;OICIIO;read_execute;;;BU)
(A;OICIIO;full;;;CO)
(A;;read_execute;;;AC)
(A;OICIIO;read_execute;;;AC)
(A;;read_execute;;;APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES)
(A;OICIIO;read_execute;;;APPLICATION PACKAGE AUTHORITY\ALL RESTRICTED APPLICATION PACKAGES)
```

Please see Microsoft's documentation for an explanation of the ACE properties.

Logging

As you saw above, SDDL strings can be complex. When writing regular expressions to match them, you need to know what you are dealing with. In other words, you need log samples.

Tip: while working on a Threat Detection rule involving the security descriptor, set the rule's VerboseLogging config element to `true`. With verbose logging enabled, uberAgent's write messages like the following to its log file:

```
GetPermissionsSddl,Read the SD of <\\?\C:\WINDOWS\System32>:
```

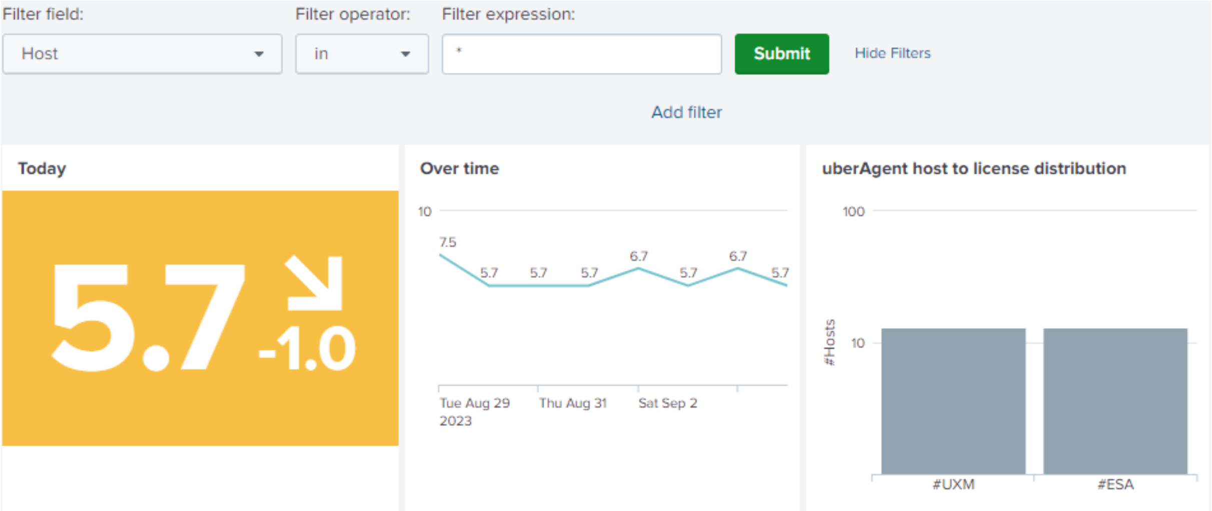
Security Score

uberAgent ESA's security score is a scoring system that calculates and visualizes the current and past status of ESA's two primary data sources, Security and Compliance Inventory (SCI) and Threat Detection Engine (TDE). In short, SCI is a script-based test engine that detects insecure configurations, while TDE is a rules-based engine that continuously monitors endpoints for unusual or risky behavior. Security scores are assigned to the results of the two data sources and range from zero to ten. Scores of zero to four are considered **high risk**, scores of four to seven are considered **medium risk**, and scores of seven to ten are considered **low risk**.

Security Score Dashboard

The security score dashboard is the entry point to uberAgent ESA's Splunk app. The dashboard visualizes security scores for the entire estate monitored by uberAgent ESA, breaking down data by data source, category, test, and rule tag to highlight potential security risks.

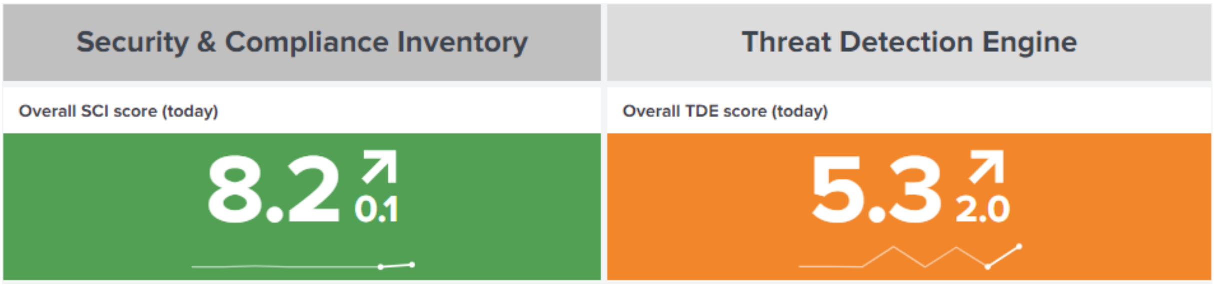
Overall Score



The filter at the top allows filtering the dashboard to a subset of machines. By default, all devices are shown.

On the left, one can see the **overall score** and the trend compared to yesterday. In the middle, the **score development** over time is visualized. On the right, one can see the **host to license distribution** panel, which compares and highlights uberAgent UXM to ESA deployment mismatches.

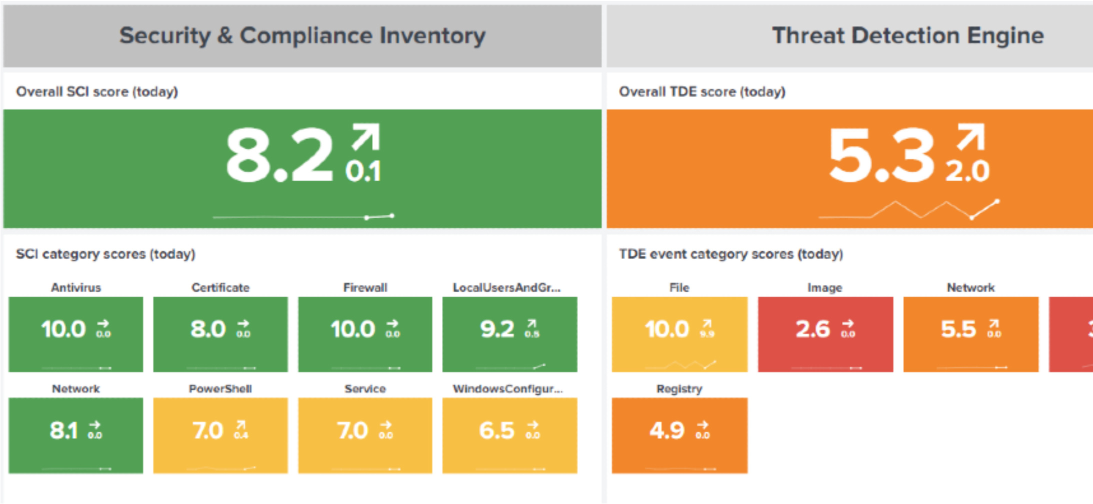
Security & Compliance Inventory and Threat Detection Engine Scores


Security & Compliance Inventory
Threat Detection Engine
Overall SCI score (today)
8.2
0.1
Overall TDE score (today)
5.3
2.0

The overall score is derived from two data sources, SCI and TDE:

- **SCI score**: indicator for *insecure or non-compliant* configured devices, user accounts, and applications.
- **TDE score**: indicator for *risky or unusual behavior* of devices, user accounts, and applications.

The charts show a trend indicator for the last day and a sparkline for the last seven days.



Score Components

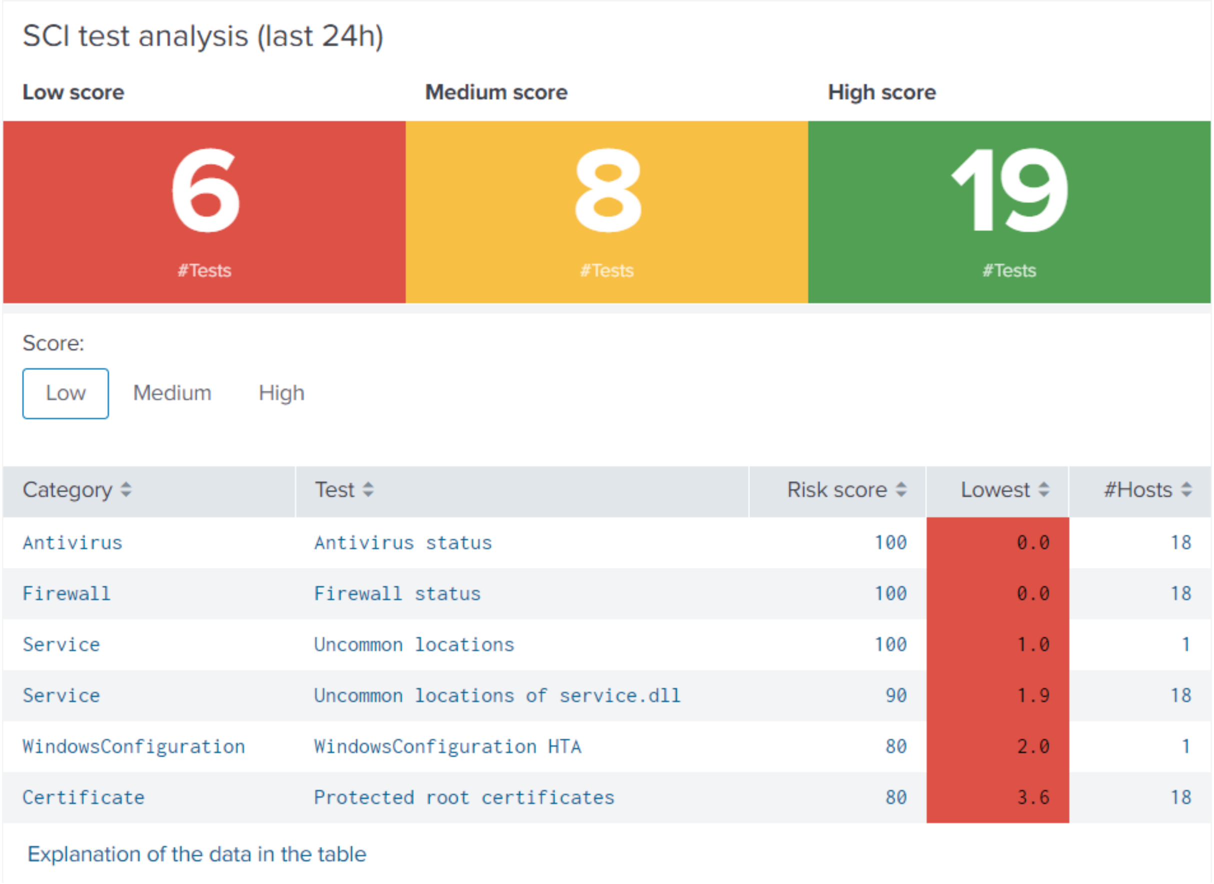
Each data source score is calculated from the underlying test category scores for the past seven days. See score calculation for further details. Each category score is displayed in a single-value visualization along with a sparkline and trend indicator.

uberAgent ESA ships with an extensive list of Security and Compliance Inventory tests to monitor various aspects of the endpoints. Each test belongs to a category. For each category, a security score is calculated and visualized in the dashboard panel. As customers may add their own tests, with their own categories, the dashboard dynamically displays test categories as test results are received by the (Splunk) backend.

Threat Detection Events can originate from various event types. Each event type is associated with a TDE event category, such as file, image, or network. The Security Score dashboard displays and calculates a security score for each TDE event category found in the events processed by the backend.

SCI Test Analysis

The charts below show scores for tests executed within the last 24 hours. Tests with low (red) or medium (yellow) security scores may warrant your attention.



SCI test analysis (last 24h)

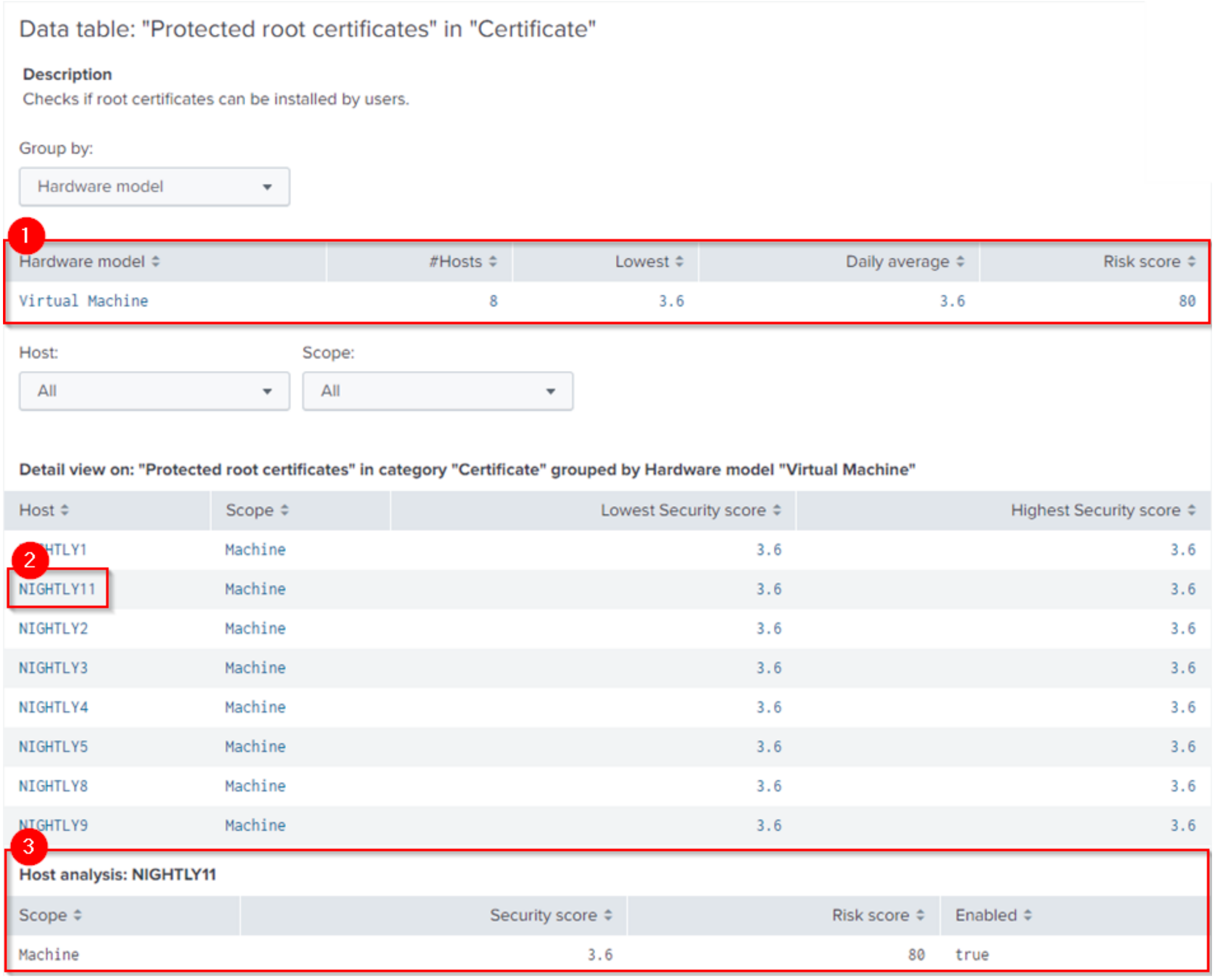
Low score	Medium score	High score
6 #Tests	8 #Tests	19 #Tests

Score:

Category ⇕	Test ⇕	Risk score ⇕	Lowest ⇕	#Hosts ⇕
Antivirus	Antivirus status	100	0.0	18
Firewall	Firewall status	100	0.0	18
Service	Uncommon locations	100	1.0	1
Service	Uncommon locations of service.dll	90	1.9	18
WindowsConfiguration	WindowsConfiguration HTA	80	2.0	1
Certificate	Protected root certificates	80	3.6	18

Explanation of the data in the table

Click on an item in the table to perform a drilldown. If you do so, a new data table displays the results for a particular test. The table shows which endpoints the test was performed on. By default, hits are grouped by *hardware model*. Other groupings are available. To drilldown further, click the first column to display all hosts in the selected group by field (1). The new drilldown table displays the lowest and highest security score, as well as the scope of the test. To analyze a specific host, click the host (2). Another drilldown table displays the scope, security score, risk score, and the actual test result (3).



Data table: "Protected root certificates" in "Certificate"

Description
Checks if root certificates can be installed by users.

Group by: Hardware model

Hardware model	#Hosts	Lowest	Daily average	Risk score
Virtual Machine	8	3.6	3.6	80

Host: All
Scope: All

Detail view on: "Protected root certificates" in category "Certificate" grouped by Hardware model "Virtual Machine"

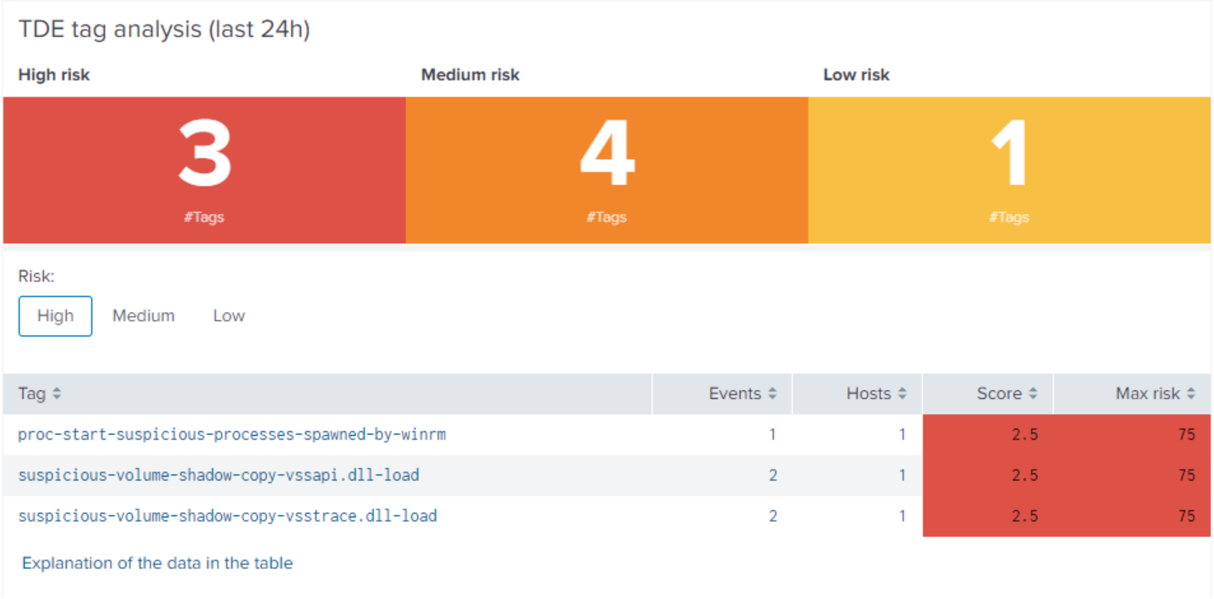
Host	Scope	Lowest Security score	Highest Security score
HTLY1	Machine	3.6	3.6
NIGHTLY11	Machine	3.6	3.6
NIGHTLY2	Machine	3.6	3.6
NIGHTLY3	Machine	3.6	3.6
NIGHTLY4	Machine	3.6	3.6
NIGHTLY5	Machine	3.6	3.6
NIGHTLY8	Machine	3.6	3.6
NIGHTLY9	Machine	3.6	3.6

Host analysis: NIGHTLY11

Scope	Security score	Risk score	Enabled
Machine	3.6	80	true

TDE Tag Analysis

The charts below show scores for TDE rule tags found within the last 24 hours. Low security scores are considered high risk (red), medium scores medium risk (orange) and high security scores low risk (yellow). Therefore, rule tags with low or medium risk scores may require the most attention. Click on an item in the table to analyze the tag in more detail on the **Threat Detection Events** dashboard.



TDE tag analysis (last 24h)

High risk	Medium risk	Low risk
3 #Tags	4 #Tags	1 #Tags

Risk:

Tag	Events	Hosts	Score	Max risk
proc-start-suspicious-processes-spawned-by-winrm	1	1	2.5	75
suspicious-volume-shadow-copy-vssapi.dll-load	2	1	2.5	75
suspicious-volume-shadow-copy-vsstrace.dll-load	2	1	2.5	75

Explanation of the data in the table

Failed SCI Tests

Failed SCI tests

Test executions can fail on endpoints for various reasons. If a test script runs into an error, uberAgent sends the errors message to the configured backend. The dashboard dynamically creates a button at the bottom of the page to display information about the failed execution. Click on the plus sign next to *Failed SCI tests* to reveal information about affected devices or users and error messages received in the last 24 hours.

Score Calculation

The security score is calculated separately for the two data sources as they contain different elements to be considered in the calculation.

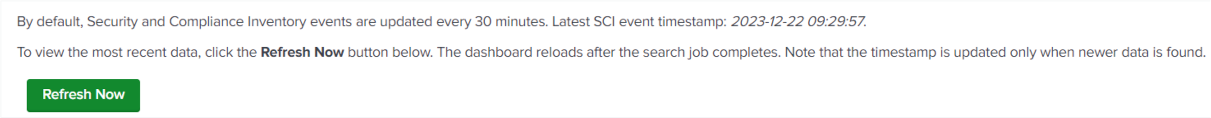
SCI Score Calculation

Each test result is sent to the backend with a result score (`SecurityInventoryScore`) as well as a risk score (`SecurityInventoryRiskScore`). The result score is contains the actual test result, whereas the risk score describes the test's importance (criticality). The following formula combines the result score with the risk score into the actual *security score*.

Each security score for SCI events is calculated as follows: *(10 - (10 - SecurityInventoryScore) x SecurityInventoryRiskScore) / 100)*

Automatic Score Calculation SCI security scores are automatically evaluated every 30 minutes for the past 60 minutes. Results are stored in a separate score index. Within the calculation, the corresponding search makes sure to avoid evaluating duplicates by comparing event timestamps with the timestamps of the last calculated events in the score index.

Trigger Score Calculation Manually The Security Score dashboard's last panel displays a timestamp indicating the time of the latest calculation search results within the score index. To update the results, users can click the *Refresh Now* button. After successfully triggering the calculation, users see a spinner animation within the button. Once the dashboard reloads, the search job is finished. If the timestamp remains unchanged after clicking the button, it may indicate that no new SCI events were found. As a result, no new calculation events were written to the score index.



By default, Security and Compliance Inventory events are updated every 30 minutes. Latest SCI event timestamp: *2023-12-22 09:29:57.*

To view the most recent data, click the **Refresh Now** button below. The dashboard reloads after the search job completes. Note that the timestamp is updated only when newer data is found.

TDE Score Calculation

Events generated by the Threat Detection Engine do not come with a *result score*. Therefore, the following formula calculates a security score for TDE events.

Security score = ((101 - RiskScore)/10) x (1 - (#Events / #AllEvents) x 0.05) x (1 - (#Devices / #AllDevices) x 0.05)

The formula uses the following variables:

- **RiskScore**: The risk score that was configured for the specific TDE rule.
- **#Events**: The number of events for the specific TDE rule.
- **#AllEvents**: The number of all events produced by any TDE rule.
- **#Devices**: The number of devices affected by the specific TDE rule.
- **#AllDevices**: The number of all devices affected by any TDE rule.

The formula is designed to take into account three factors: **risk score**, **number of events**, and the **number of devices**. The first part of the formula subtracts the risk score from 101 and divides the result by 10. This is done to invert the risk score scale so that a higher risk score marks a lower security score. The second and third parts of the formula are based on the concept of exponential decay. They calculate the proportion of events and devices relative to their respective maximum values and then convert these proportions into a decay using the exponential function. The decay factors are then multiplied and scaled by a factor of 0.05. Multiplying with that factor ensures that the impact of counts (events and devices) on the security score is not too strong. Finally, the three factors are combined by multiplying them to result in the final security score for TDE events.

Score Ranges

Security Score

The security score combines uberAgent ESA's main data sources, Security and Compliance Inventory and Threat Detection Engine. The score an indicator for insecurely configured devices, or risky behavior on endpoints. The table below shows the security score ranges, where their categorization changes and which colors are applied in the dashboard panels.

Type	Threshold	SCI color	TDE color
High security score	= 8	green	yellow
Medium security score	= 4 & <8	yellow	orange
Low security score	< 4	red	red

Risk Score

The risk score is used to describe the importance of a SCI test or a TDE rule. As risk scores are used in the security score calculation, they are used to add weight to the actual result. Risk scores range from zero (low risk) to one hundred (high risk).

Risk score	Threshold
High	=80
Medium	= 40 & <80
Low	< 40

Hash Calculation of PE Images

uberAgent ESA calculates hashes of executables (e.g., `.exe`, `.dll` or `.sys` files). Whenever a process is started or a DLL is loaded, uberAgent calculates the hash of the file located on disk. uberAgent supports the hash variants `MD5`, `SHA-1`, `SHA-256`, and `ImpHash` both individually and simultaneously.

Configuration

The uberAgent ESA hash calculation feature is configured through the process startup setting `EnableCalculateHash`. In the default configuration, `MD5` hash calculation is enabled.

Process and library (DLL) hashes are cached to reduce CPU load. Cache lifetime is managed automatically. The cache's size can be configured (or disabled altogether) via the configuration setting `HashesCacheMaxSize`.

Metadata

Sourcetype

Process hashes are part of the sourcetypes `uberAgent:Process:ProcessStartup` and `uberAgentESA:Process:ProcessStop`. Please see the metrics documentation for a description of the fields.

Due to the huge amount of data being produced, hash values for image load events are not sent to the backend but can be used with the Threat Detection Engine.

Authenticode Signature Verification

uberAgent ESA verifies the Authenticode signature for every process that is started.

The following information is collected:

- Is the executable signed by the OS manufacturer, e.g., Microsoft?
- Is the Authenticode signature valid?
- The Authenticode signer's name

Configuration

uberAgent ESA Authenticode verification is configured through the process startup setting `EnableAuthenticode`. In the default configuration, Authenticode verification is enabled.

uberAgent ESA caches the results of Authenticode verifications. The number of cached results can be set via `AuthenticodeCacheMaxSize`, which is preset to 500 entries in the default configuration.

Metadata

Sourcetype

Authenticode signature information is part of the sourcetype `uberAgent:Process:ProcessStartup`. Please see the metrics documentation for a description of the fields.

Security & Compliance Inventory

uberAgent ESA Security & Compliance Inventory (SCI) is a testing and rating framework that **checks the attack surface** of your operating systems and applications. The test results are used to calculate **security scores** that pinpoint configuration and security hardening weaknesses.

Security & Compliance Inventory tests are scheduled to run at regular intervals to always provide a complete and accurate status of the security configuration of all devices in the fleet. Test results are reported to uberAgent's backend along with the other metrics and KPIs collected by uberAgent's endpoint agent. uberAgent ESA's Security Score dashboard intuitively visualizes the test scores, highlighting insecure device or application configurations.

uberAgent ESA comes with a comprehensive suite of ready-to-use SCI tests for many common attack vectors. Customizing and extending ESA's SCI tests are explicitly encouraged. By adding tests of their own, customers and partners can automate the verification of an organization's security policy or check the configuration of line-of-business applications.

Test Storage

uberAgent ESA Security & Compliance Inventory tests are part of uberAgent's configuration, which is maintained in the uberAgent Configuration GitHub repository.

Visualization

ESA Security & Compliance Inventory test results are visualized in the *Security Score* dashboard, which is part of the `uberAgent_ESA` Splunk searchhead app.

Test Configuration

uberAgent ESA's Security & Compliance Inventory tests are part of the configuration. This page documents the configuration syntax.

Example

Timer

In the following example, uberAgent runs all Security & Compliance Inventory tests from the category `Antivirus` once every 24 hours:

```
[Timer]
Name                 = Security & Compliance Inventory timer
Interval             = 86400000
Start delay          = 600000
Persist interval     = true
UA metric            = SecurityInventory.Antivirus
```

SecurityInventoryTest

Each Security & Compliance Inventory test is defined in a stanza such as the following. Note the `Category` assignment, which corresponds to `SecurityInventory.Antivirus` from the `Timer` stanze above.

```
[SecurityInventoryTest]
Name                 = Antivirus
ScriptId             = 71270F6B-7160-4629-90C4-F36E621D43E1
Category             = Antivirus
ScriptCommandline    = "###UA_SI_LOCALPATH###\Antivirus\Antivirus.ps1"
Interpreter          = PowerShell
ScriptTimeoutMs      = 600000
OutputFormat         = JSON
ScriptContext        = Session0AsSystem
IntegrityLevel       = High
```

ScriptInterpreter

The method of execution is configured in `ScriptInterpreter` stanzas. PowerShell scripts are invoked as defined in the following example:

```
[ScriptInterpreter]
Name         = PowerShell
Executable   = C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Parameter    = -ExecutionPolicy Bypass -file ###UA_SCRIPT###
```

Test Definition

There can be any number of `[SecurityInventoryTest]` stanzas, each defining one test script. Tests are processed in the order in which they are defined in the configuration. uberAgent ESA always processes all scripts for every timer run containing the metric `SecurityInventory.CATEGORY` where the timer stanza's `CATEGORY` matches the category from the `SecurityInventoryTest` stanza.

A `[SecurityInventoryTest]` stanza may contain the following settings.

ScriptId

- **Setting name:** `ScriptId`
- **Description:** a GUID that represents the script. Must be unique.
- **Valid values:** a GUID
- **Default:** empty
- **Required:** yes

Category

- **Setting name:** `Category`
- **Description:** a category name for the test. Multiple tests can have the same category name (e.g., `Antivirus`).
- **Valid values:** Any string
- **Default:** empty
- **Required:** no

Name

- **Setting name:** `Name`
- **Description:** a name for the test which is logged in the uberAgent log file in case of an error.
- **Valid values:** Any string
- **Default:** empty
- **Required:** no

ScriptCommandline

- **Setting name:** `ScriptCommandline`
- **Description:** the full path to the script.
- **Valid values:** any valid file system object
- **Default:** empty
- **Required:** yes

Interpreter

- **Setting name:** `Interpreter`
- **Description:** the interpreter that starts the script. See stanza `[ScriptInterpreter]`. If left empty, the `ScriptCommandline` is run ”as is” without prepending the invocation of an interpreter.
- **Valid values:** any string
- **Default:** empty
- **Required:** no

ScriptTimeoutMs

- **Setting name:** `ScriptTimeoutMs`
- **Description:** timeout in milliseconds.
- **Valid values:** any number
- **Default:** `90000`
- **Required:** no

OutputFormat

- **Setting name:** `OutputFormat`
- **Description:** defines the output format of the script.
- **Valid values:** `JSON`
- **Default:** `JSON`
- **Required:** no

ScriptContext

- **Setting name:** `ScriptContext`
- **Description:** defines the context in which the script is started.
- **Valid values:**
  - `Session0AsSystem`: runs the script once per timer run in the context of the local SYSTEM user.
  - `UserSessionAsSystem`: runs the script for each user logged in during the timer run in the context of the local SYSTEM user.
  - `UserSessionAsUser`: runs the script for each user logged in during the timer run in the context of the user.

- **Default:** `Session0AsSystem`
- **Required:** no

IntegrityLevel

This setting is available on Windows only.

- **Setting name:** `IntegrityLevel`
- **Description:** defines the integrity level at which the script is started.
  Format: `LEVEL [+Mitigation1] [-Mitigation2] [+Mitigation3] [...]`
  See the sandbox documentation for details on the available mitigations.
- **Valid values for `LEVEL`:**
  - `Low`: integrity level is set to low.
  - `Medium`: integrity level is set to medium.
  - `High`: integrity level is set to high.
- **Default:** `medium`
- **Required:** no

Script Interpreter Definition

A `[ScriptInterpreter]` stanza may contain the following settings.

Name

- **Setting name:** `Name`
- **Description:** a unique name for the interpreter. This name must be referred to in the `Interpreter` setting of the `[SecurityInventoryTest]` setting.
- **Valid values:** any string
- **Default:** empty
- **Required:** yes

Executable

- **Setting name:** `Executable`
- **Description:** the full path to the interpreter that executes the script.
- **Valid values:** full path to the script.

- **Default:** empty
- **Required:** yes

Parameter

- **Setting name:** `Parameter`
- **Description:** parameters for the interpreter. The script name must be specified here via the placeholder `UA_SCRIPT` (see below).
- **Valid values:** any string
- **Default:** empty
- **Required:** yes

UA_SCRIPT Placeholder A script interpreter stanza specifies how to execute scripts from a given language. The script to be executed is specified via the placeholder `###UA_SCRIPT###`, a variable that is replaced with the `ScriptCommandline` from the `SecurityInventoryTest` stanza before running a script.

Test Scripts

uberAgent ESA's Security & Compliance Inventory (SCI) test scripts can be authored in any interpreted or compiled programming language. Typically, an interpreted language such as PowerShell or Python is used. This page documents conventions and requirements for SCI test scripts.

Test Script Execution

Tests are scheduled by uberAgent's endpoint agent according to the test configuration.

Tests are run sandboxed according to the principle of least privilege. Please see the sandbox docs for details.

If a test is executed in user context, it is checked whether the user has read permissions. If not, the built-in group `Users` is added to the parent directory. If this is not desired, it can be disabled via the ConfigFlag `DisableSetFilePermissionsOnExec`.

Test Script Output

Multiple Tests per Script

A Security & Compliance Inventory test script may perform multiple independent tests per script invocation.

Console Output

uberAgent expects a test script's output on the console (`stdout`). The script must only print test output to the console. Any other console output must be suppressed.

Output Encoding Set the encoding to Unicode. When using PowerShell scripts, the following line should be specified at the beginning of each script:

```
[Console]::OutputEncoding = [System.Text.Encoding]::UTF8
```

Output Format

Format Specification The expected output format is configured in the `OutputFormat` setting of the `[SecurityInventoryTest]` stanza (docs).

JSON Format A test result that uses JSON output format looks like this:

```
[
   {

      "Name": "PowerShell v2 disabled",
      "Score": 2,
      "RiskScore": 0,
      "ResultData": "Yes",
      "ErrorCode": 0,
      "ErrorMessage": ""
    }
 ,
   {

      "Name": "PowerShell remoting allowed",
      "Score": 2,
      "RiskScore": 80,
      "ResultData": "Yes",
      "ErrorCode": 0,
      "ErrorMessage": ""
    }

]
```

Please note that the dashboard is designed to handle Security and Compliance Inventory test results as JSON objects only. Adding tests that do not use JSON output format may result in faulty dashboard panels or misleading information.

Modify Test Names and Descriptions

uberAgent ESA ships with an extensive set of pre-defined tests than can be executed through its Security and Compliance Inventory functionality. Each test is displayed with a human readable name and a short description. As customers may add their own tests, **test display names** and **descriptions** have to be added as well.

To modify existing or add new tests to this table, change the following input lookup file in `$SPLUNK_HOME/etc/apps/uberAgent_ESA/lookups/security_inventory_checknames.csv`. Please make sure to distribute the changed input lookup file to all search heads.

The lookup file contains three columns.

- **SecurityInventoryName**: this field contains the test name that was used in the script. To save data volume, this name may be abbreviated. Example: *CMProtRoot*
- **SecurityInventoryDisplayName**: this field contains the display name of the test to improve readability. Example: *protected root certificates*
- **SecurityInventoryNameDescription**: this field contains a description of the test to be displayed in the data table after selecting a test in the SCI test analysis chart. Example: *Checks if root certificates can be installed by users.*

Output Fields

The following table describes the fields that may be part of a test script's output:

Field name	Format	Valid values	Description
Name	String	Any string	The name of the test. Must be quoted to avoid parsing errors.
Score	Number	0-10	The resulting test score on a scale from 0 (very bad) to 10 (excellent).
RiskScore	Number	0-100	The severity of the test (how risky is the tested thing).
ResultData	String	Any string	Configuration information determined by the test. Should be quoted to avoid parsing errors.

Field name	Format	Valid values	Description
ErrorCode	Number	Any number	Indicates success or failure. 0 = success.
ErrorMessage	String	Any string	Optional error message returned by the test. This message is logged to the agent log.

Script Sandbox

uberAgent ESA's Security & Compliance Inventory test scripts are run sandboxed according to the principle of least privilege. This page documents the sandbox security levels and all available mitigations. A script's sandbox is configured via the `IntegrityLevel` setting of the `SecurityInventoryTest` stanza in the test configuration.

Mitigations & UI Limits

The following tables list the mitigations and UI limits that are available to limit the capabilities of a script sandbox.

Please note that the three integrity levels are just convenient aliases for common sets of mitigations. As described on the test configuration page, any mitigation and limit can be explicitly enabled or disabled.

Mitigations

Mitigation name	Description	Integrity Level low	Integrity Level medium	Integrity Level high	Supported OS
ControlFlowGuard	Enables Control Flow Guard for the started executable.	✓	✓	-	Windows 8.1 and newer

Mitigation name	Description	Integrity Level low	Integrity Level medium	Integrity Level high	Supported OS
DataExecutionPrevention	Prevents code execution from code inside a memory page marked as non-executable.	✓	✓	-	Windows 8.0 and newer
DataExecutionForChildProcessesFromATL	Prevents code execution from code inside a memory page marked as non-executable (active template library thunk layer).	✓	✓	-	Windows 8.0 and newer
DisableVariousUncommonDllExtensionPoints	Prevents some built-in third-party extension points from being turned on, which prevents legacy extension point DLLs from being loaded into the process.	✓	-	-	Windows 8.0 and newer

Mitigation name	Description	Integrity Level low	Integrity Level medium	Integrity Level high	Supported OS
DynamicCodeMustBeSigned	A Process must be either signed by Microsoft, by the Windows Store, or by the Windows Hardware Quality Labs (WHQL).	✓	-	-	Windows 8.1 and newer
ForceRelocateImagesRequireRelocationSection	Activates the “Address space layout randomization” technique, which loads the address space of key data areas of a process into random positions.	✓	✓	-	Windows 8.0 and newer
HeapTerminateProcess	Causes the heap to terminate if it becomes corrupt.	✓	✓	-	Windows 8.0 and newer
AllowChildProcesses=	Sets the number of allowed child processes.	2	2	∞	Windows 8.1 and newer

Mitigation name	Description	Integrity Level low	Integrity Level medium	Integrity Level high	Supported OS
NonRemoteImagesBeingLoaded	Sets the types of executable images that are allowed to be mapped into the process. When turned on, images cannot be loaded from some locations, such as remote devices or files that have a low mandatory integrity label.	✓	✓	-	Windows 10 and newer
NonSystemFontsBlocked	Blocks loading of fonts outside of the standard Windows fonts directory.	✓	-	-	Windows 10 and newer
PreferSystem32InsteadOfApplicationDirectory	DLLs are preferably loaded from the `System32` directory instead of the application directory.	✓	✓	-	Windows 10 and newer

Mitigation name	Description	Integrity Level low	Integrity Level medium	Integrity Level high	Supported OS
StrictHandleChecks	The process will receive a fatal error if it manipulates a handle that is not valid.	✓	-	-	Windows 8.0 and newer
StructuredExceptionHandlerOverwriteProtection	Prevents the code from overwriting the structured exception handler.	✓	-	-	Windows 8.0 and newer

UI Limits

UI limit name	Description	Integrity Level low	Integrity Level medium	Integrity Level high	Supported OS
UiLimitHandles	Prevents processes from using user handles owned by processes not associated with the same job.	✓	✓	-	Windows
UiLimitReadClipboard	Prevents processes from reading data from the clipboard	✓	✓	-	Windows
UiLimitWriteClipboard	Prevents processes from writing data to the clipboard.	✓	✓	-	Windows

UI limit name	Description	Integrity Level low	Integrity Level medium	Integrity Level high	Supported OS
UiLimitSystemParameters	Prevents processes from changing system parameters by using the `SystemParametersInfo` function.	✓	✓	-	Windows
UiLimitDisplaySettings	Prevents processes from calling the `ChangeDisplaySettings` function.	✓	✓	-	Windows
UiLimitGlobalAtoms	Prevents processes from accessing global atoms. When this flag is used, each job has its own atom table.	✓	✓	-	Windows
UiLimitDesktop	Prevents processes from creating desktops and switching desktops using the `CreateDesktop` and `SwitchDesktop` functions.	✓	✓	-	Windows

UI limit name	Description	Integrity Level low	Integrity Level medium	Integrity Level high	Supported OS
UiLimitExitWindows	Prevents processes from calling the `ExitWindows` and `ExitWindowsEx` functions.	✓	✓	-	Windows

DNS Query Monitoring

uberAgent ESA monitors outgoing DNS queries and their responses. This includes all DNS queries that are processed by the operating system.

Functionality

The implementation and functionality of DNS query monitoring differ between operating systems.

Windows

uberAgent uses the local DNS machine’s DNS resolver as the data source. uberAgent sees all DNS requests that are handled by the OS, including requests via encrypted DNS (DoT or DoH). uberAgent doesn’t see requests that bypass the OS resolver, e.g., requests from tools such as `nslookup`.

uberAgent cannot differentiate between requests that go over the wire and requests that can be answered from the local DNS cache. Both types of requests are collected by uberAgent.

macOS

uberAgent monitors network traffic at the packet level. uberAgent sees any and all DNS requests, but it cannot process encrypted DNS (DoT or DoH). uberAgent only collects requests that go over the wire, it doesn’t process requests that can be answered from the local DNS cache.

Configuration

uberAgent ESA DNS query monitoring is enabled or disabled through the timer metric `DnsMonitoring` . For Windows clients, this feature requires Windows 8.1 or newer versions of Windows since the data source for this event is not available in previous Windows versions, such as Windows 7. Enabling the metric on unsupported Windows versions has no effect. For macOS, all versions are supported.

Metadata

Sourcetype

DNS query monitoring events are sent with the sourcetype `uberAgentESA:Process:DnsQuery` (documentation). This feature may significantly increase the data volume and can be limited using event data filtering. uberAgent's configuration includes a comprehensive set of rules to exclude known and harmless DNS requests.

Threat Detection

DNS monitoring events are also available via uberAgent's Threat Detection engine. This allows the creation of rules to report specific DNS events (e.g., malicious events or threat detection).

Visualization

DNS query monitoring events are visualized in the *DNS Exfiltration and Tunneling* dashboard, which is part of the `uberAgent_ESA` Splunk searchhead app.

MITRE ATT&CK Integration

MITRE ATT&CK

The MITRE ATT&CK® framework is a knowledge base of adversary tactics and techniques based on real-world observations of cybersecurity threats. With the help of ATT&CK, different stakeholders (offense, defense, SOCs, third-party vendors, and so on) can speak the same language to describe attacks on enterprise IT and mobile devices. While there are other similar frameworks available, ATT&CK is the de-facto industry standard.

Integration With uberAgent ESA

When an uberAgent ESA Threat Detection rule matches suspicious activity, uberAgent's endpoint agent sends an event to its configured SIEM backend. uberAgent annotates Threat Detection events with ATT&CK technique IDs.

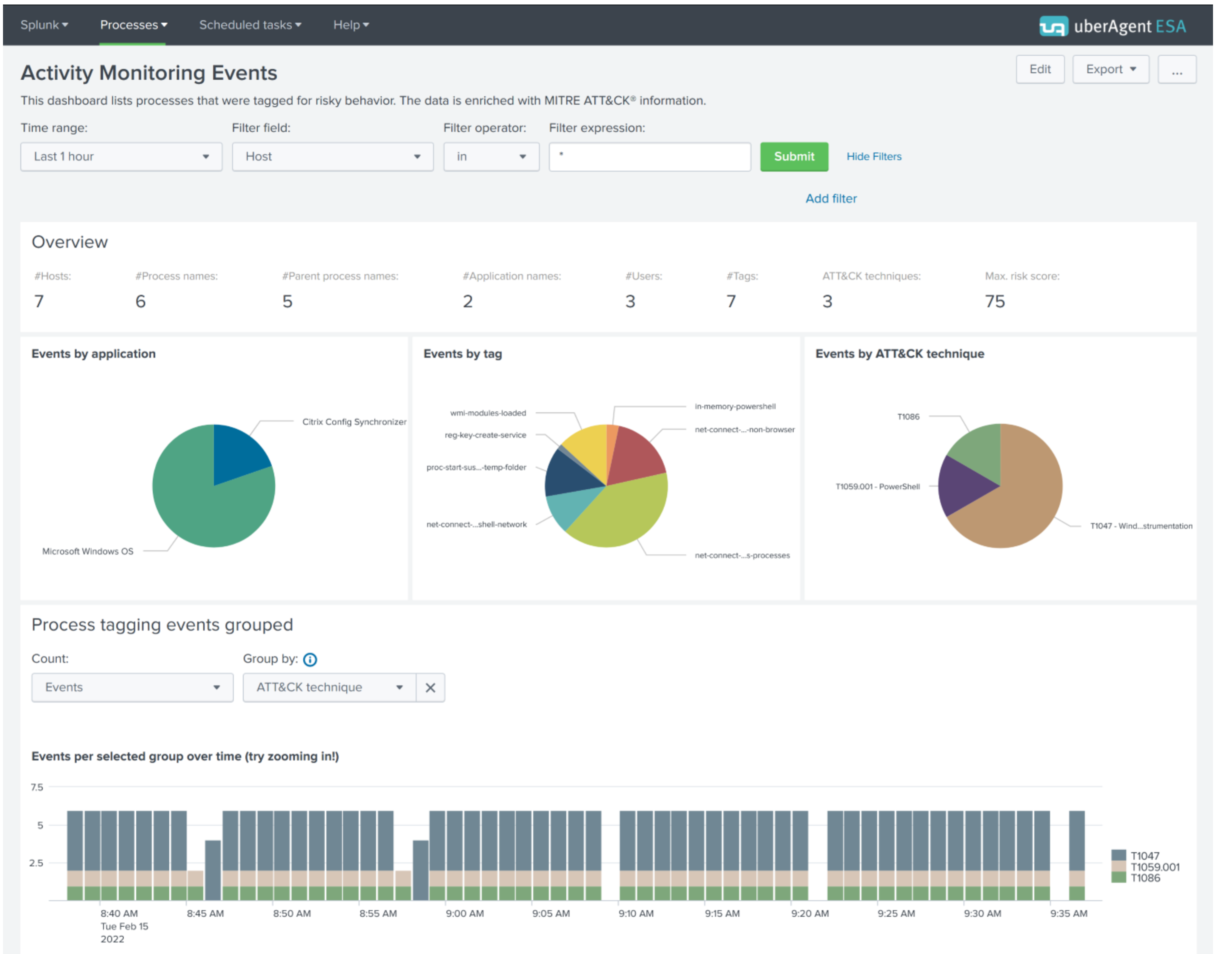
Annotations are part of the ESA Threat Detection rule specification. Implementation details can be found here. Below is a rule sample:

```
[ActivityMonitoringRule]
# Detects suspicious DNS queries known from Cobalt Strike beacons
RuleName = Cobalt Strike DNS Beaconing
EventType = Dns.Query
Tag = cobalt-strike-dns-beaconing
RiskScore = 100
Annotation = {
 "mitre_attack": ["T1071", "T1071.004"] }

Query = (Dns.QueryRequest like r"aaa.stage.%" or Dns.QueryRequest like r"post.1%")
GenericProperty1 = Dns.QueryRequest
GenericProperty2 = Dns.QueryResponse
```

Visualization

The uberAgent ESA Splunk app processes the Threat Detection events with technique ID annotations and adds context like the technique's name, its description and detection, and the URL to the technique on the MITRE website.

This processed information is visualized in the **Threat Detection Events** dashboard. In the screenshot below you can see the number of events by technique ID, and the distribution of events by ID over time.



Activity Monitoring Events

This dashboard lists processes that were tagged for risky behavior. The data is enriched with MITRE ATT&CK® information.

Time range: Last 1 hour

Filter field: Host

Filter operator: in

Filter expression: *

Overview

#Hosts:	#Process names:	#Parent process names:	#Application names:	#Users:	#Tags:	ATT&CK techniques:	Max. risk score:
7	6	5	2	3	7	3	75

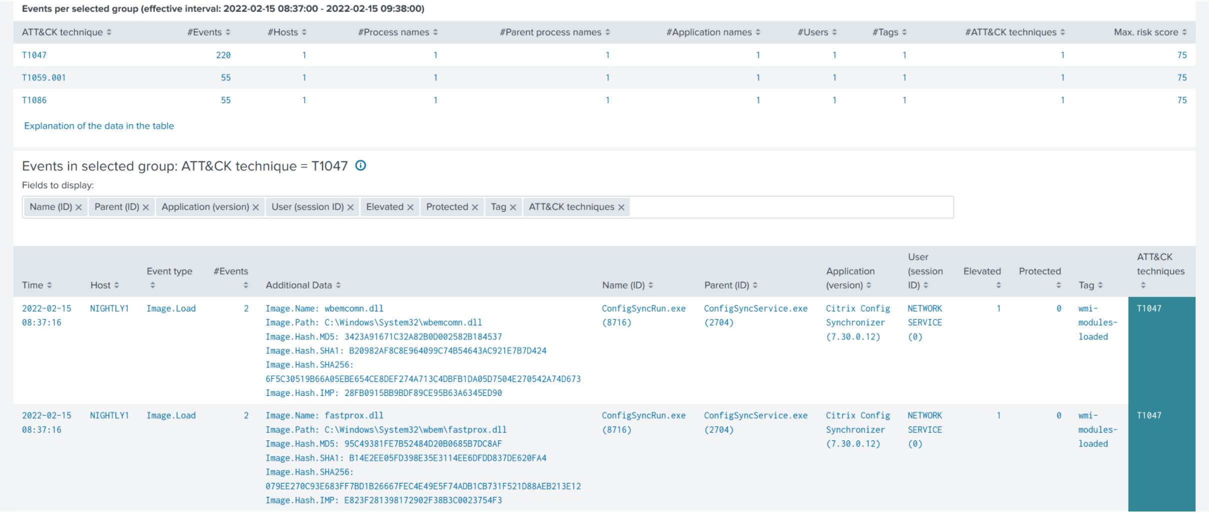
Events by application
Citrix Config Synchronizer
Microsoft Windows OS
Events by tag
wmi-modules-loaded
reg-key-create-service
proc-start-sus...-temp-folder
net-connect-...shell-network
in-memory-powershell
net-connect-...-non-browser
net-connect-...s-processes
Events by ATT&CK technique
T1086
T1059.001 - PowerShell
T1047 - Wind...strumentation

Process tagging events grouped

Count: Events

Group by: ATT&CK technique

The further you scroll down the more details you get. The first table in the screenshot below shows details for each technique. A click on a technique row lists all events where the technique was detected in the second table in the screenshot below.



Events per selected group (effective interval: 2022-02-15 08:37:00 - 2022-02-15 09:38:00)

ATT&CK technique ⇕	#Events ⇕	#Hosts ⇕	#Process names ⇕	#Parent process names ⇕	#Application names ⇕	#Users ⇕	#Tags ⇕	#ATT&CK techniques ⇕	Max. risk score ⇕
T1047	220	1	1	1	1	1	1	1	75
T1059.001	55	1	1	1	1	1	1	1	75
T1086	55	1	1	1	1	1	1	1	75

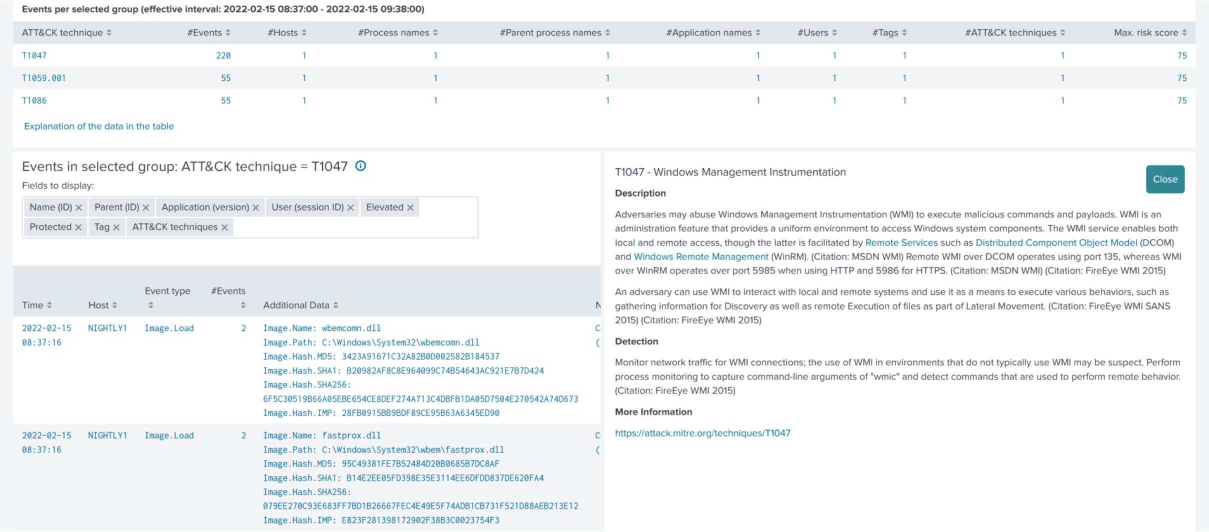
Explanation of the data in the table

Events in selected group: ATT&CK technique = T1047

Fields to display: Name (ID) × Parent (ID) × Application (version) × User (session ID) × Elevated × Protected × Tag × ATT&CK techniques ×

Time ⇕	Host ⇕	Event type ⇕	#Events ⇕	Additional Data ⇕	Name (ID) ⇕	Parent (ID) ⇕	Application (version) ⇕	User (session ID) ⇕	Elevated ⇕	Protected ⇕	Tag ⇕	ATT&CK techniques ⇕
2022-02-15 08:37:16	NIGHTLY1	Image.Load	2	Image.Name: wbemcomn.dll Image.Path: C:\Windows\System32\wbemcomn.dll Image.Hash.MD5: 3423A91671C32A82B0D002582B184537 Image.Hash.SHA1: B20982AF8C8E964099C74B54643AC921E7B7D424 Image.Hash.SHA256: 6F5C30519B66A05EBE654CE8DEF274A713C4DBFB1DA05D7504E270542A74D673 Image.Hash.IMP: 28FB0915BB9BDF89CE95B63A6345ED90	ConfigSyncRun.exe (8716)	ConfigSyncService.exe (2704)	Citrix Config Synchronizer (7.30.0.12)	NETWORK SERVICE (0)	1	0	wmi-modules-loaded	T1047
2022-02-15 08:37:16	NIGHTLY1	Image.Load	2	Image.Name: fastprox.dll Image.Path: C:\Windows\System32\wbem\fastprox.dll Image.Hash.MD5: 95C49381FE7B52484D20B0685B7DC8AF Image.Hash.SHA1: B14E2EE05FD398E35E3114EE6DFDD837DE620FA4 Image.Hash.SHA256: 079EE270C93E683FF7BD1B26667FEC4E49E5F74ADB1CB731F521D88AEB213E12 Image.Hash.IMP: E823F281398172902F38B3C0023754F3	ConfigSyncRun.exe (8716)	ConfigSyncService.exe (2704)	Citrix Config Synchronizer (7.30.0.12)	NETWORK SERVICE (0)	1	0	wmi-modules-loaded	T1047

Note that the rightmost column **ATT&CK techniques** is highlighted in blue. When you click on it you get detailed ATT&CK information on the technique ID in a pop-up.



Events per selected group (effective interval: 2022-02-15 08:37:00 - 2022-02-15 09:38:00)

ATT&CK technique	#Events	#Hosts	#Process names	#Parent process names	#Application names	#Users	#Tags	#ATT&CK techniques	Max. risk score
T1047	220	1	1	1	1	1	1	1	75
T1059.001	55	1	1	1	1	1	1	1	75
T1086	55	1	1	1	1	1	1	1	75

Explanation of the data in the table

Events in selected group: ATT&CK technique = T1047

Fields to display:

Name (ID) × Parent (ID) × Application (version) × User (session ID) × Elevated × Protected × Tag × ATT&CK techniques ×

Time	Host	Event type	#Events	Additional Data
2022-02-15 08:37:16	NIGHTLY1	Image.Load	2	Image.Name: wbemcomn.dll Image.Path: C:\Windows\System32\wbemcomn.dll Image.Hash.MD5: 3423A91671C32A82B0D002582B184537 Image.Hash.SHA1: B20982AF8C8E964099C74B54643AC921E7B7D424 Image.Hash.SHA256: 6F5C30519B66A05EBE654CE8DEF274A713C4DBFB1DA05D7504E270542A74D673 Image.Hash.IMP: 28FB0915BB9BDF89CE95B63A6345ED90
2022-02-15 08:37:16	NIGHTLY1	Image.Load	2	Image.Name: fastprox.dll Image.Path: C:\Windows\System32\wbem\fastprox.dll Image.Hash.MD5: 95C49381FE7B52484D20B0685B7DC8AF Image.Hash.SHA1: B14E2EE05FD398E35E3114EE6DFDD837DE620FA4 Image.Hash.SHA256: 079EE270C93E683FF7BD1B26667FEC4E49E5F74ADB1CB731F521D88AEB213E12 Image.Hash.IMP: E823F281398172902F38B3C0023754F3

T1047 - Windows Management Instrumentation

Description

Adversaries may abuse Windows Management Instrumentation (WMI) to execute malicious commands and payloads. WMI is an administration feature that provides a uniform environment to access Windows system components. The WMI service enables both local and remote access, though the latter is facilitated by Remote Services such as Distributed Component Object Model (DCOM) and Windows Remote Management (WinRM). (Citation: MSDN WMI) Remote WMI over DCOM operates using port 135, whereas WMI over WinRM operates over port 5985 when using HTTP and 5986 for HTTPS. (Citation: MSDN WMI) (Citation: FireEye WMI 2015)

An adversary can use WMI to interact with local and remote systems and use it as a means to execute various behaviors, such as gathering information for Discovery as well as remote Execution of files as part of Lateral Movement. (Citation: FireEye WMI SANS 2015) (Citation: FireEye WMI 2015)

Detection

Monitor network traffic for WMI connections; the use of WMI in environments that do not typically use WMI may be suspect. Perform process monitoring to capture command-line arguments of "wmic" and detect commands that are used to perform remote behavior. (Citation: FireEye WMI 2015)

More Information

https://attack.mitre.org/techniques/T1047

Installation

The ATT&CK integration is enabled by default. You don't need to do anything except install the uberAgent ESA Splunk app on your Splunk search heads. The installation is documented here.

Keeping ATT&CK Up-to-Date

Every uberAgent release includes the latest ATT&CK information. If you notice that MITRE made changes to their ATT&CK framework and want the updated data in Splunk, you have two options:

1. Wait for the next uberAgent version and update the ESA Splunk app
2. Update the CSV file for the ATT&CK Splunk lookup manually
   a) The current CSV is always accessible in our GitHub repository. Download the file `annotation_mitre_attack.csv` from there.
   b) Replace the file in `$SPLUNK_HOME/etc/apps/uberAgent_ESA/lookups`
   c) Restart Splunk
3. Context: in the year 2021, MITRE made five changes to the source

uberAgent ESA and Splunk Enterprise Security

Splunk Enterprise Security also has an ATT&CK integration and uberAgent integrates with ES, too. Below is a short summary of both integrations to avoid confusion.

uberAgent MITRE ATT&CK Data in uberAgent ESA Dashboards

The uberAgent ESA Splunk app visualizes the data collected by uberAgent's endpoint agents. The uberAgent ESA Splunk app is compatible with Splunk Enterprise and Cloud. It has an ATT&CK integration built-in that can be used without the need to purchase Splunk Enterprise Security.

Use it when:

- You don't have Splunk Enterprise Security
- You only need ATT&CK and no other cybersecurity framework

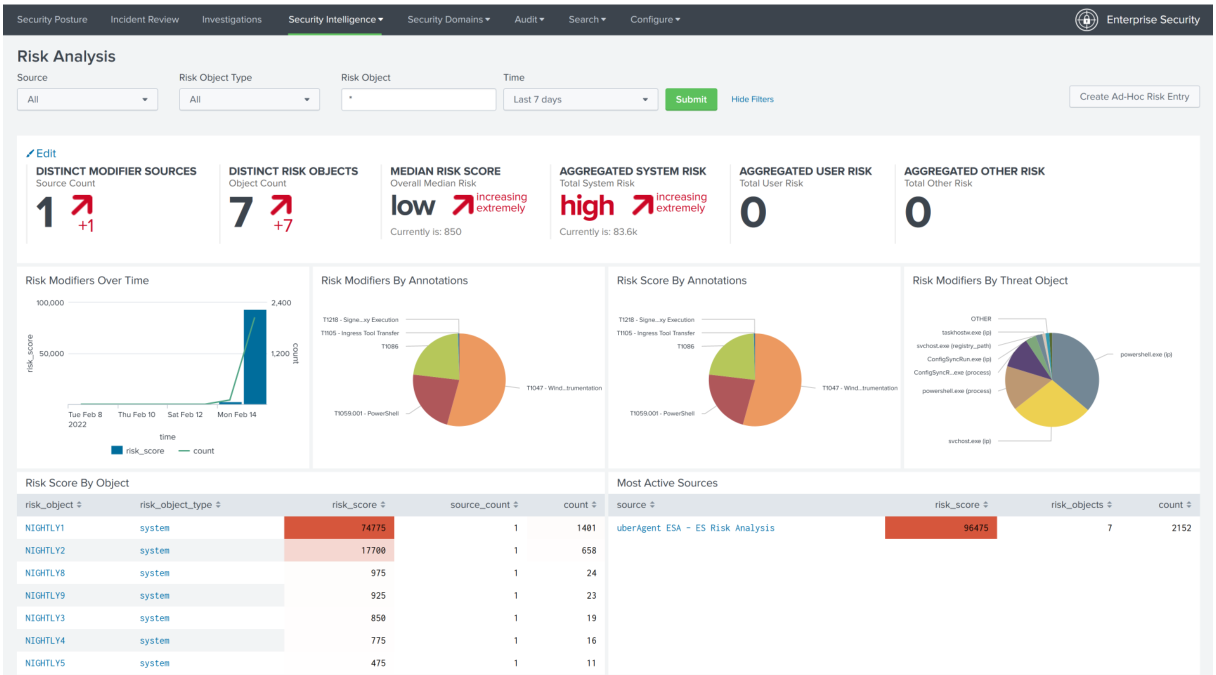
uberAgent MITRE ATT&CK Data in Splunk Enterprise Security

Splunk Enterprise Security is a paid premium app developed by Splunk that needs to be licensed on top of Splunk Enterprise or Cloud. It comes with support for multiple cybersecurity frameworks, ATT&CK is one of them. If you are an ES customer you may want to install the *uberAgent ESA ES companion* app to enjoy the benefits of uberAgent ESA data in Enterprise Security, enriched with data from cybersecurity frameworks (more information).

Use it when:

- You have Splunk Enterprise Security
- You need more cybersecurity frameworks than ATT&CK

Note: in this scenario, you can use both the uberAgent ESA app and Splunk's Enterprise Security dashboards.



Risk Analysis

Source: All
Risk Object Type: All
Risk Object: *
Time: Last 7 days

DISTINCT MODIFIER SOURCES
Source Count
1 ↗ +1

DISTINCT RISK OBJECTS
Object Count
7 ↗ +7

MEDIAN RISK SCORE
Overall Median Risk
low ↗ increasing extremely
Currently is: 850

AGGREGATED SYSTEM RISK
Total System Risk
high ↗ increasing extremely
Currently is: 83.6k

AGGREGATED USER RISK
Total User Risk
0

AGGREGATED OTHER RISK
Total Other Risk
0

Risk Score By Object

risk_object	risk_object_type	risk_score	source_count	count
NIGHTLY1	system	74775	1	1401
NIGHTLY2	system	17700	1	658
NIGHTLY8	system	975	1	24
NIGHTLY9	system	925	1	23
NIGHTLY3	system	850	1	19
NIGHTLY4	system	775	1	16
NIGHTLY5	system	475	1	11

Most Active Sources

source	risk_score	risk_objects	count
uberAgent ESA - ES Risk Analysis	96475	7	2152

Event Log Forwarding

uberAgent can collect Windows system logs like Application, Security, System, etc. Events can be filtered based on their provider, event ID, and level (Information, Error, etc.). Additionally, more fine-grained event-filtering is possible with the XPath-querying functionality.

Requirements

- Eventlog forwarding is supported on Windows
- uberAgent 7.3 or newer

Configuring Eventlog Forwarding

Eventlog forwarding is configured with a configuration file. By default, no logs are collected. Eventlog collection needs to be enabled by an administrator.

The `[EventLog]` Stanza

The stanza `[EventLog]` starts a new log configuration, followed by multiple settings.

Setting	Description	Required	Values
EventLog	The name of the event log to query.	Yes	Any valid log name, like `System`.
LevelName	The event's level. Specify multiple levels separated by commas.	Yes	`Information`, `Warning`, `Error`, `Critical`, `Verbose`
EventID	Limit the collection to specific event IDs. Specify multiple IDs separated by commas.	No	Any valid event IDs

Setting	Description	Required	Values
EventFilterXPath	Limit the collection with a XPath filter. XPath uses path expressions to select nodes or node-sets in an XML document. The node is selected by following a path or steps.	No	Any valid XPath filter.
Provider	Limit the collection to specific providers.	No	Any valid provider name.

Collection Frequency

Eventlogs can be queried on a configurable schedule, or on-demand.

Schedule To schedule the log collection, first create a new `[EventLog]` stanza.

```
[EventLog Name=TimerBased-System-Errors-Warnings]
EventLog = System
LevelName = Error,Warning
```

Second, create a new timer and reference the name the Eventlog stanza's name.

```
[Timer]
Name = System Eventlog collection
EventLog = TimerBased-System-Errors-Warnings
Interval = 60000
```

On-demand Create a new `[EventLog]` stanza without a corresponding timer. When uberAgent can't find the Eventlog stanza's name in any timer, it collects events as they occur.

```
[EventLog Name=OnDemand-Security-Events]
EventLog = Security
LevelName=Information
```

Collecting events as they occur can be resource intensive. Hence, collecting on a schedule is the preferred option.

Default Configuration

uberAgent does not collect any logs by default. Event log collection needs to be enabled by an administrator. However, the infrastructure is already prepared. uberAgent comes with the `uberAgent-ESA-eventlog-windows.conf` file, which holds a few examples that can be activated by uncommenting the lines.

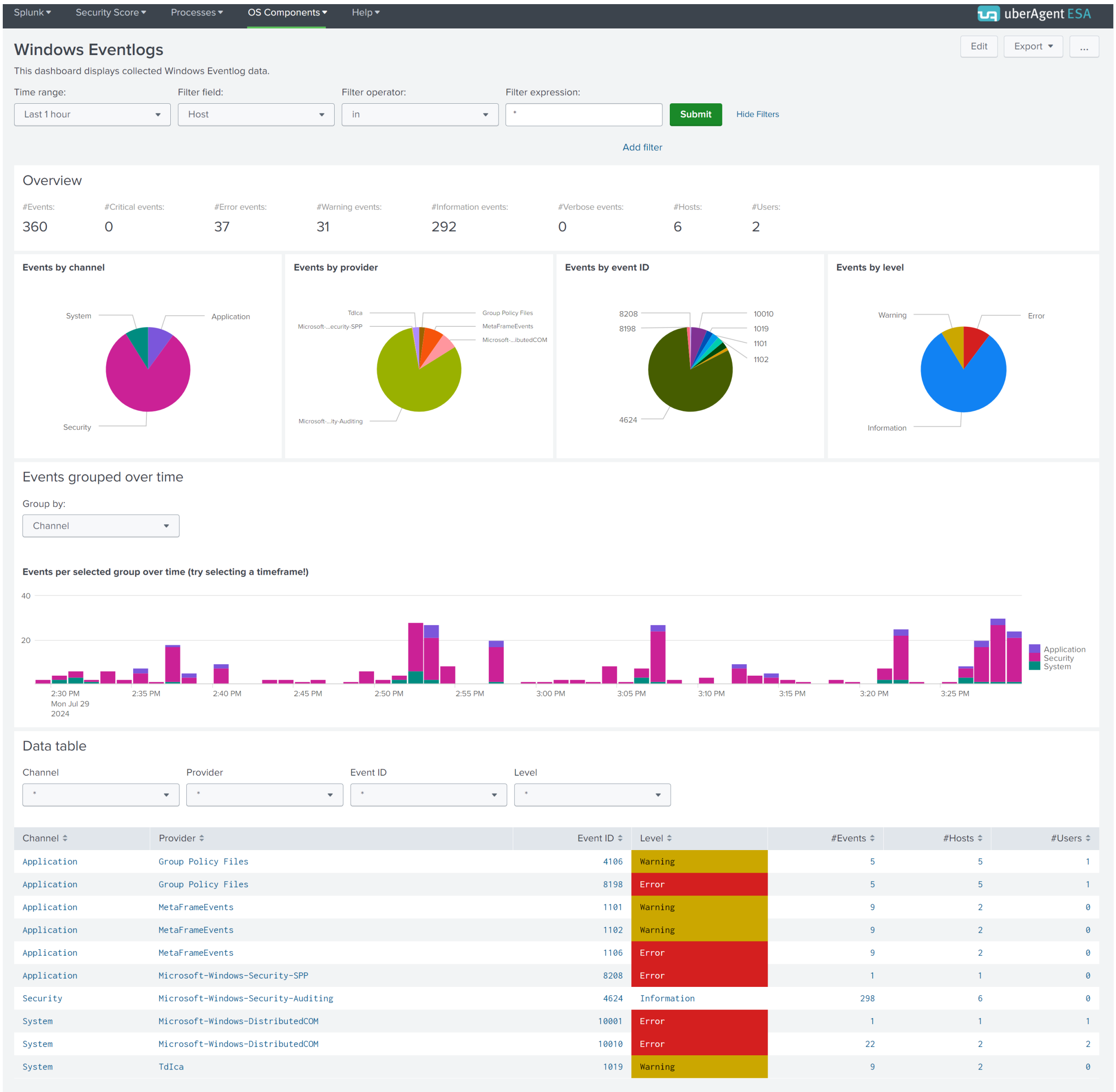
Metadata

Sourcetype

Windows Eventlog events are sent with the sourcetype `uberAgentESA:System:WinEvtLogForwarding` (documentation).

Visualization

The collected Eventlogs are visualized on the **Windows Eventlogs** dashboard in the uberAgent ESA Splunk app.



Windows Eventlogs

This dashboard displays collected Windows Eventlog data.

Time range: Last 1 hour

Filter field: Host

Filter operator: in

Filter expression: *

Overview

#Events:	#Critical events:	#Error events:	#Warning events:	#Information events:	#Verbose events:	#Hosts:	#Users:
360	0	37	31	292	0	6	2

Events grouped over time

Group by: Channel

Data table

Channel: *

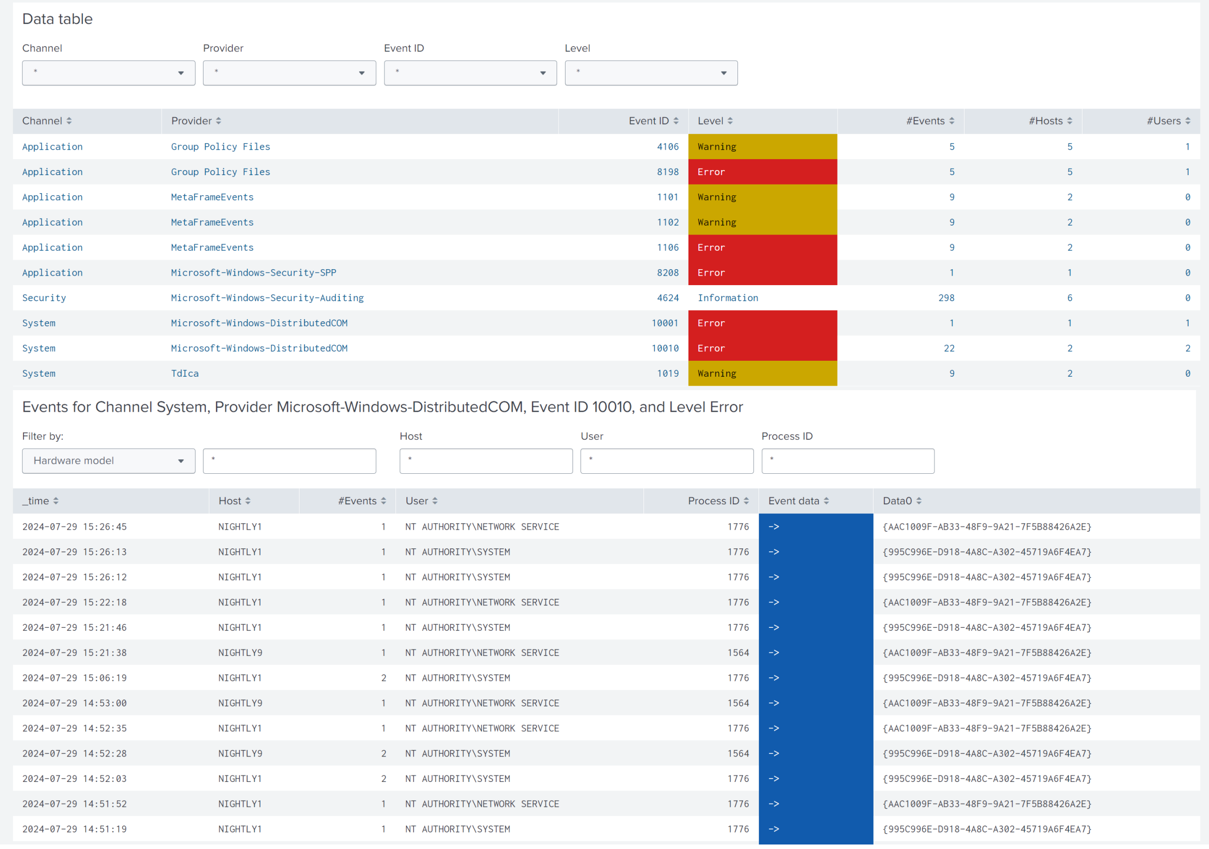
Provider: *

Event ID: *

Level: *

Channel ⇕	Provider ⇕	Event ID ⇕	Level ⇕	#Events ⇕	#Hosts ⇕	#Users ⇕
Application	Group Policy Files	4106	Warning	5	5	1
Application	Group Policy Files	8198	Error	5	5	1
Application	MetaFrameEvents	1101	Warning	9	2	0
Application	MetaFrameEvents	1102	Warning	9	2	0
Application	MetaFrameEvents	1106	Error	9	2	0
Application	Microsoft-Windows-Security-SPP	8208	Error	1	1	0
Security	Microsoft-Windows-Security-Auditing	4624	Information	298	6	0
System	Microsoft-Windows-DistributedCOM	10001	Error	1	1	1
System	Microsoft-Windows-DistributedCOM	10010	Error	22	2	2
System	TdIca	1019	Warning	9	2	0

While the first charts give an overview of the collected Eventlogs, the data table at the bottom lists all of them grouped by channel, provider, ID, and level. Click a row in the table and a new chart with all details opens below. You can use the provided filter panels to limit the output to specific hosts, users, or process IDs.



Data table

Channel: * | Provider: * | Event ID: * | Level: *

Channel	Provider	Event ID	Level	#Events	#Hosts	#Users
Application	Group Policy Files	4106	Warning	5	5	1
Application	Group Policy Files	8198	Error	5	5	1
Application	MetaFrameEvents	1101	Warning	9	2	0
Application	MetaFrameEvents	1102	Warning	9	2	0
Application	MetaFrameEvents	1106	Error	9	2	0
Application	Microsoft-Windows-Security-SPP	8208	Error	1	1	0
Security	Microsoft-Windows-Security-Auditing	4624	Information	298	6	0
System	Microsoft-Windows-DistributedCOM	10001	Error	1	1	1
System	Microsoft-Windows-DistributedCOM	10010	Error	22	2	2
System	TdIca	1019	Warning	9	2	0

Events for Channel System, Provider Microsoft-Windows-DistributedCOM, Event ID 10010, and Level Error

Filter by: Hardware model * | Host: * | User: * | Process ID: *

_time	Host	#Events	User	Process ID	Event data	Data0
2024-07-29 15:26:45	NIGHTLY1	1	NT AUTHORITY\NETWORK SERVICE	1776	->	{AAC1009F-AB33-48F9-9A21-7F5B88426A2E}
2024-07-29 15:26:13	NIGHTLY1	1	NT AUTHORITY\SYSTEM	1776	->	{995C996E-D918-4A8C-A302-45719A6F4EA7}
2024-07-29 15:26:12	NIGHTLY1	1	NT AUTHORITY\SYSTEM	1776	->	{995C996E-D918-4A8C-A302-45719A6F4EA7}
2024-07-29 15:22:18	NIGHTLY1	1	NT AUTHORITY\NETWORK SERVICE	1776	->	{AAC1009F-AB33-48F9-9A21-7F5B88426A2E}
2024-07-29 15:21:46	NIGHTLY1	1	NT AUTHORITY\SYSTEM	1776	->	{995C996E-D918-4A8C-A302-45719A6F4EA7}
2024-07-29 15:21:38	NIGHTLY9	1	NT AUTHORITY\NETWORK SERVICE	1564	->	{AAC1009F-AB33-48F9-9A21-7F5B88426A2E}
2024-07-29 15:06:19	NIGHTLY1	2	NT AUTHORITY\SYSTEM	1776	->	{995C996E-D918-4A8C-A302-45719A6F4EA7}
2024-07-29 14:53:00	NIGHTLY9	1	NT AUTHORITY\NETWORK SERVICE	1564	->	{AAC1009F-AB33-48F9-9A21-7F5B88426A2E}
2024-07-29 14:52:35	NIGHTLY1	1	NT AUTHORITY\NETWORK SERVICE	1776	->	{AAC1009F-AB33-48F9-9A21-7F5B88426A2E}
2024-07-29 14:52:28	NIGHTLY9	2	NT AUTHORITY\SYSTEM	1564	->	{995C996E-D918-4A8C-A302-45719A6F4EA7}
2024-07-29 14:52:03	NIGHTLY1	2	NT AUTHORITY\SYSTEM	1776	->	{995C996E-D918-4A8C-A302-45719A6F4EA7}
2024-07-29 14:51:52	NIGHTLY1	1	NT AUTHORITY\NETWORK SERVICE	1776	->	{AAC1009F-AB33-48F9-9A21-7F5B88426A2E}
2024-07-29 14:51:19	NIGHTLY1	1	NT AUTHORITY\SYSTEM	1776	->	{995C996E-D918-4A8C-A302-45719A6F4EA7}

Process Tree Dashboard

Tracking Process Lifetime & Child Creation

Process GUID

uberAgent ESA assigns each process a GUID. Such a unique ID is necessary because the operating system's process IDs are reused. By leveraging process GUIDs, uberAgent can track processes throughout their lifetime, from the start (sourcetype `uberAgent:Process:ProcessStartup`) to the end (sourcetype `uberAgentESA:Process:ProcessStop`) as well as during the runtime (sourcetype `uberAgent:Process:ProcessDetail`).

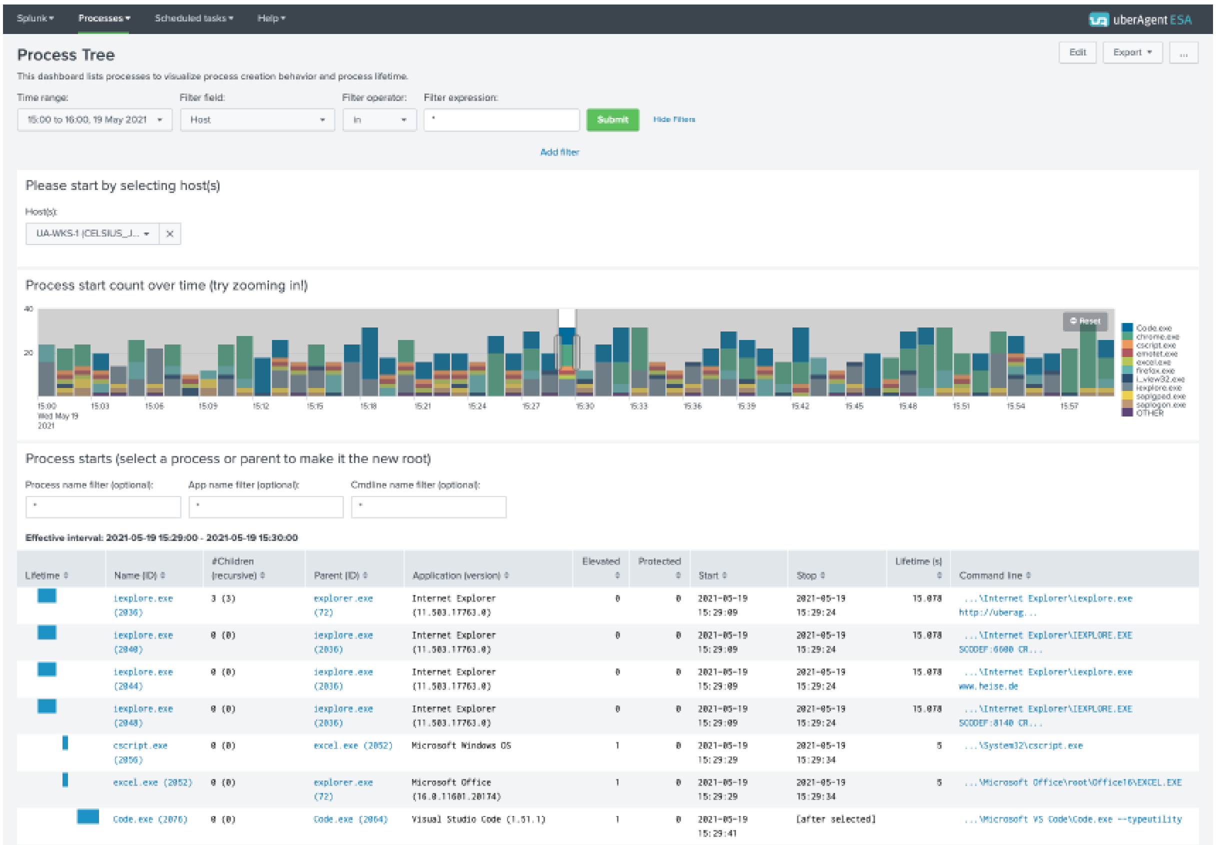
Process Parent

uberAgent ESA not only identifies unique process instances; it also keeps track of parent-child relationships. All process start and stop events include names and GUIDs of the parent process.

Process Tree

A process tree is an essential tool for understanding process creation behavior. uberAgent ESA comes with a powerful Process Tree dashboard that makes it easy to identify a process'descendants, listing important process properties such as the process **lifetime**, the **command line**, the **elevation** status, or the name and **version** of the application the process is a part of. Additionally, the number of child processes is calculated, both direct children as well as their children recursively.

By selecting any process in the tree, it becomes the new root, and the table updates to show its child processes. This makes it possible to browse through process hierarchies.



Process Tree

This dashboard lists processes to visualize process creation behavior and process lifetime.

Time range: 15:00 to 16:00, 19 May 2021

Filter field: Host

Filter operator: in

Filter expression: *

Please start by selecting host(s)

Host(s): UA-WKS-1 (CELSIUS_J...

Process start count over time (try zooming in!)

Process starts (select a process or parent to make it the new root)

Process name filter (optional): *

App name filter (optional): *

Cmdline name filter (optional): *

Effective interval: 2021-05-19 15:29:00 - 2021-05-19 15:30:00

Lifetime	Name (ID)	#Children (recursive)	Parent (ID)	Application (version)	Elevated	Protected	Start	Stop	Lifetime (s)	Command line
▬	iexplore.exe (2036)	3 (3)	explorer.exe (72)	Internet Explorer (11.503.17763.0)	0	0	2021-05-19 15:29:09	2021-05-19 15:29:24	15.078	...\Internet Explorer\iexplore.exe http://uberag...
▬	iexplore.exe (2040)	0 (0)	iexplore.exe (2036)	Internet Explorer (11.503.17763.0)	0	0	2021-05-19 15:29:09	2021-05-19 15:29:24	15.078	...\Internet Explorer\IEXPLORE.EXE SCODEF:6600 CR...
▬	iexplore.exe (2044)	0 (0)	iexplore.exe (2036)	Internet Explorer (11.503.17763.0)	0	0	2021-05-19 15:29:09	2021-05-19 15:29:24	15.078	...\Internet Explorer\iexplore.exe www.heise.de
▬	iexplore.exe (2048)	0 (0)	iexplore.exe (2036)	Internet Explorer (11.503.17763.0)	0	0	2021-05-19 15:29:09	2021-05-19 15:29:24	15.078	...\Internet Explorer\IEXPLORE.EXE SCODEF:8140 CR...
▮	cscript.exe (2056)	0 (0)	excel.exe (2052)	Microsoft Windows OS	1	0	2021-05-19 15:29:29	2021-05-19 15:29:34	5	...\System32\cscript.exe
▮	excel.exe (2052)	0 (0)	explorer.exe (72)	Microsoft Office (16.0.11601.20174)	1	0	2021-05-19 15:29:29	2021-05-19 15:29:34	5	...\Microsoft Office\root\Office16\EXCEL.EXE
▬	Code.exe (2076)	0 (0)	Code.exe (2064)	Visual Studio Code (1.51.1)	1	0	2021-05-19 15:29:41	[after selected]		...\Microsoft VS Code\Code.exe --typeutility

Scheduled Task Monitoring

uberAgent ESA monitors changes to Windows scheduled tasks. Whenever a task is created, updated, or deleted, uberAgent generates an event with all available details. This includes properties that are not displayed in the Windows Task Scheduler UI, such as COM actions or custom triggers.

Configuration

uberAgent ESA scheduled task monitoring is enabled or disabled through the on-demand metric `ScheduledTaskMonitoring`. In the default configuration, scheduled task monitoring is enabled.

Metadata

Sourcetype

ESA scheduled task monitoring events are assigned the sourcetypes:

- `uberAgentESA:System:ScheduledTasks`
- `uberAgentESA:System:ScheduledTaskActions`
- `uberAgentESA:System:ScheduledTaskTriggers`

Please see the metrics documentation for a description of the fields.

Visualization

ESA scheduled task monitoring events are visualized in the *Scheduled Tasks* dashboard which is part of the `uberAgent_ESA` Splunk searchhead app.

Splunk Enterprise Security Integration

Splunk Enterprise Security

> Splunk Enterprise Security (ES) solves a wide range of security analytics and operations use cases including continuous security monitoring, advanced threat detection, compliance, incident investigation, forensics, and incident response. Splunk ES delivers an end-to-end view of organizations'security postures with flexible investigations, unmatched performance, and the most flexible deployment options offered in the cloud, on-premises, or hybrid deployment models.

Source: Splunk

Integrating uberAgent ESA

uberAgent ESA comes with native support for Splunk Enterprise Security. This article summarizes all related information in one place.

Installation

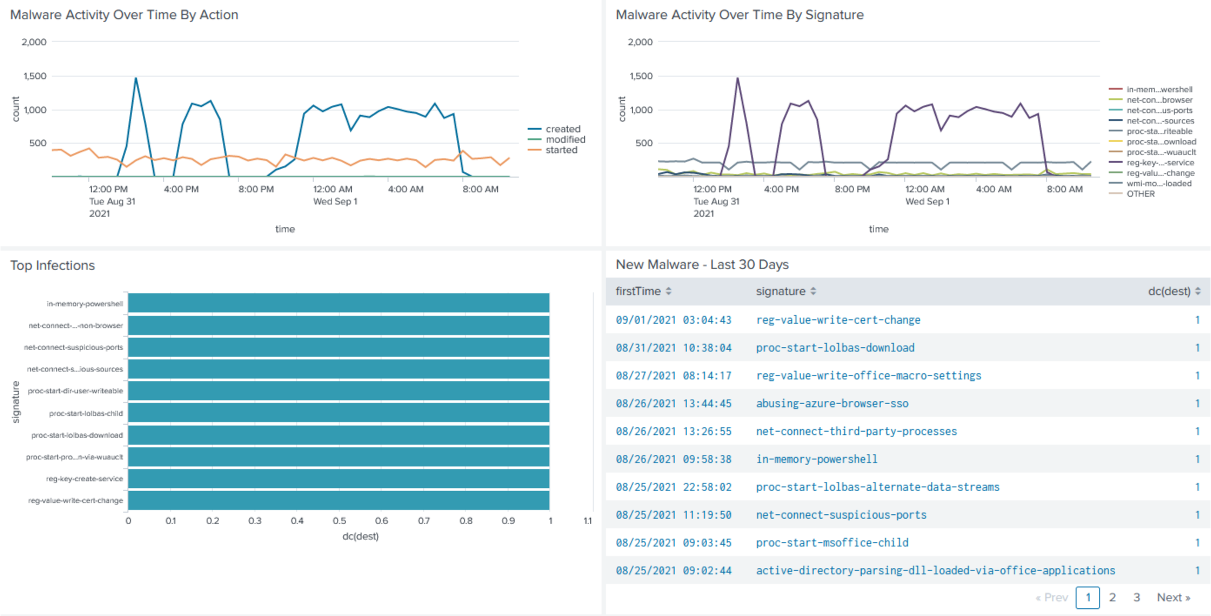
Detailed installation instructions can be found here. In short, you need to install the following uberAgent apps on your ES server:

- **uberAgent_searchhead.tgz**: contains uberAgent UXM CIM information
- **uberAgent_ESA_searchhead.tgz**: contains uberAgent ESA CIM information
- **uberAgent_ESA_ES_companion.tgz**: required for Splunk's risk-based alerting. Available on Splunkbase.

CIM Support

Splunk Enterprise Security works with data sources from many different vendors. This diverse data set needs to be normalized to allow users to search and interact with it in a standardized way. Splunk uses the CIM data model to achieve that.

While uberAgent had CIM support for a long time, we have extended the integration greatly with uberAgent ESA. If you are used to working with Sysmon data in ES, you will notice no difference when switching to uberAgent. **uberAgent supports all CIM fields populated by popular Sysmon add-ons found in Splunkbase, and more!**

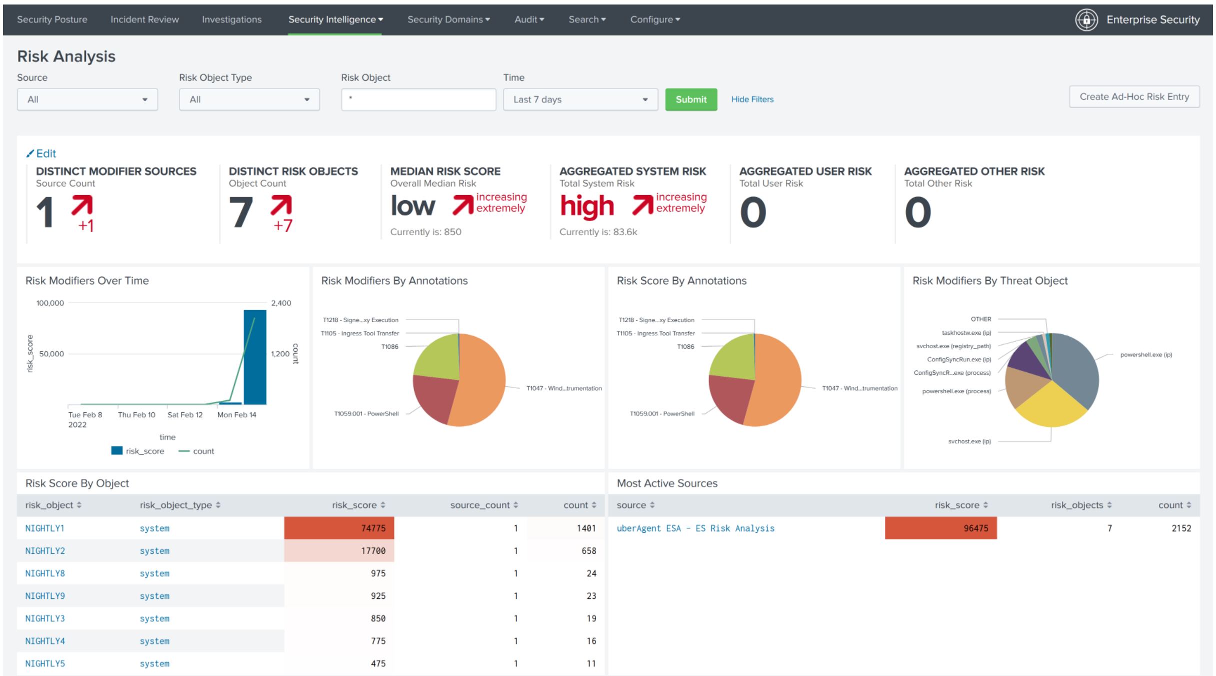


firstTime ⇕	signature ⇕	dc(dest) ⇕
09/01/2021 03:04:43	reg-value-write-cert-change	1
08/31/2021 10:38:04	proc-start-lolbas-download	1
08/27/2021 08:14:17	reg-value-write-office-macro-settings	1
08/26/2021 13:44:45	abusing-azure-browser-sso	1
08/26/2021 13:26:55	net-connect-third-party-processes	1
08/26/2021 09:58:38	in-memory-powershell	1
08/25/2021 22:58:02	proc-start-lolbas-alternate-data-streams	1
08/25/2021 11:19:50	net-connect-suspicious-ports	1
08/25/2021 09:03:45	proc-start-msoffice-child	1
08/25/2021 09:02:44	active-directory-parsing-dll-loaded-via-office-applications	1

Visit our blog post about the CIM integration for details.

Risk-Based Alerting

Splunk introduced **risk-based alerting (RBA)** in ES to reduce alert volume and enhance security operations. uberAgent supports RBA through the **uberAgent ESA ES companion** app. See the installation section above for installation instructions. When all requirements are met, the Splunk Enterprise Security dashboard **Risk Analysis** is populated automatically with uberAgent data:



Risk Analysis

Source: All
Risk Object Type: All
Risk Object: *
Time: Last 7 days

DISTINCT MODIFIER SOURCES
Source Count
1 ↗ +1

DISTINCT RISK OBJECTS
Object Count
7 ↗ +7

MEDIAN RISK SCORE
Overall Median Risk
low ↗ increasing extremely
Currently is: 850

AGGREGATED SYSTEM RISK
Total System Risk
high ↗ increasing extremely
Currently is: 83.6k

AGGREGATED USER RISK
Total User Risk
0

AGGREGATED OTHER RISK
Total Other Risk
0

Risk Score By Object

risk_object ⇕	risk_object_type ⇕	risk_score ⇕	source_count ⇕	count ⇕
NIGHTLY1	system	74775	1	1401
NIGHTLY2	system	17700	1	658
NIGHTLY8	system	975	1	24
NIGHTLY9	system	925	1	23
NIGHTLY3	system	850	1	19
NIGHTLY4	system	775	1	16
NIGHTLY5	system	475	1	11

Most Active Sources

source ⇕	risk_score ⇕	risk_objects ⇕	count ⇕
uberAgent ESA - ES Risk Analysis	96475	7	2152

Security Frameworks RBA supports annotations with cyber security frameworks like MITRE ATT&CK®, CIS 20, NIST Controls, and more. uberAgent does, too, and comes with ATT&CK annotation out-of-the-box. You may add other cyber security frameworks anytime.

Annotations are part of the ESA Threat Detection rule specification. Implementation details can be found here. Below is a rule sample:

```
[ActivityMonitoringRule]
# Detects suspicious DNS queries known from Cobalt Strike beacons
RuleName = Cobalt Strike DNS Beaconing
EventType = Dns.Query
Tag = cobalt-strike-dns-beaconing
RiskScore = 100
Annotation = {
 "mitre_attack": ["T1071", "T1071.004"] }

Query = (Dns.QueryRequest like r"aaa.stage.%" or Dns.QueryRequest like r"post.1%")
GenericProperty1 = Dns.QueryRequest
GenericProperty2 = Dns.QueryResponse
```

UXM Metrics

uberAgent UXM collects many different metrics, logically grouped into sourcetypes. All metrics are explained in detail in this chapter, including:

- Name and description
- Supported platform
- Unit and example
- Related source type and where to find it in the dashboards
- Calculated fields and and their availability in Splunk's data model and/or "Search Processing Language" (SPL)
- The uberAgent template configuration files can be found at our Github uberAgent-config repository.

Application Metrics

Application Crashes & Hangs Metrics

Application Errors

For every application error uberAgent collects metrics like application name and version, process lifetime as well as the exception code.

Notes:

- Processes are auto-grouped into applications, i.e. the application name is determined automatically without requiring any configuration. Information on how this works are available here.
- Field: `AppVersion` - uberAgent has an internal filter to minimize data volume by suppressing version information for system processes and system services. As a result, the `AppVersion` field is typically empty for most system processes and services.
- macOS: The fields `Module*` are not available for `ErrorTypeName Hang`.

Details

- Source type: `uberAgent:Application:Errors`
- Used in dashboards: *Application Errors*, *Single Machine Detail*, *Single Application Detail*, *Single User Detail*
- Enabled through configuration setting: `ApplicationErrors`
- Related configuration settings: *n/a*
- Supported platform: *all*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Measurement type	Example	Platform
ErrorType	Error type code. Possible values: 1, 2. Please take a look at the field `ErrorTypeName`, too.	Number		Snapshot	1	all
ProcName	Process name.	String		Snapshot	chrome.exe	all
ProcPath	Process path.	String		Snapshot	C:\Program Files (x86)\Google\Chrome\Application\chro	all
ProcVersion	Process version.	String		Snapshot	67.0.3396.99	all
ProcTimestamp	Build timestamp of the process binary, embedded PE header information.	String		Snapshot	2017-06-02 11:43:43.000 +0200	Windows
ModuleName	Faulting module name.	String		Snapshot	msvcrt.dll	all
ModulePath	Faulting module path.	String		Snapshot	C:\Windows\system32\msvcrt.dll	all
ModuleVersion	Faulting module version.	String		Snapshot	7.0.9600.17415	all

Field	Description	Data type	Unit	Measurement type	Example	Platform
ModuleTimestamp	Build timestamp of the faulting module, embedded PE header information.	String		Snapshot	2014-10-29 04:04:30.000 +0200	Windows
ProcID	Process ID.	Number		Snapshot	456	all
ProcLifetimeMs	Process lifetime.	Number	ms	Sum	82744165	all
ExceptionCode	Exception code.	String		Snapshot	0xc0000005	all
FaultOffset	An address offset relative to the base address of the crashed image.	String		Snapshot	0x0000b5f0	all
AppPackageFullName	Microsoft UWP app full name. More information.	String		Snapshot	Microsoft.XboxGameCallableUI_1000.1	Windows
AppPackageRelativeId	Microsoft UWP app relative ID. More information.	String		Snapshot	xbox	Windows

Field	Description	Data type	Unit	Measurement type	Example	Platform
AppId	Associated application ID. Used by uberAgent to lookup application names and populate field `AppName`.	String		Snapshot	GglChrm	all
ProcUser	Process user.	String		Snapshot	Domain\JohnDoe	all
SessionGUID	Unique identifier that is generated by uberAgent when the session is created. Valid for this session only.	String		Snapshot	00000002-f295-9109-e7c7-c964011dd401	all
ProcGUID	Unique identifier that is generated by uberAgent when the process is started.	String		Snapshot	00000000-0599-6955-eb63-a00bb127d401	all
AppVersion	Application version.	String		Snapshot	67.0.3396.99	all

Field	Description	Data type	Unit	Measurement type	Example	Platform
HangType	Hang type. Requires Windows 10 1903. Possible values: `Cross-process`, `Cross-thread`, `Quiesce`, `Top level window is idle`, `Unknown`.	String		Snapshot	Cross-process	Windows
ProcImageUUID	A unique identifier of the binary image that was executed.	String		Snapshot	D61D3713-69F9-32C3-B001-E509C0A8EE0A	macOS
ModuleImageUUID	A unique identifier of the binary image in which the crash occurred.	String		Snapshot	D61D3713-69F9-32C3-B001-E509C0A8EE0A	macOS

List of Calculated Fields

Field	Description	Data type	Unit	Measurement type	Where available	Example
ErrorTypeName	Display name of the `ErrorType` field. Possible values: `Crash`, `Hang`.	String		Snapshot	Splunk data model, Splunk SPL	Hang
User	User name.	String		Snapshot	Splunk data model	Domain\JohnDoe
AppName	Associated application name.	String		Snapshot	Splunk data model, Splunk SPL	Google Chrome

Application & Process Performance Metrics

Process Detail

uberAgent collects rich metrics per process & per application. This includes the user and domain, CPU & RAM usage as well as network latency & throughput, to name just a few.

Notes:

- Processes are auto-grouped into applications, i.e., the application name is determined automatically. Information on how automatic application identification works is available here.
- Field: `AppVersion` - uberAgent has an internal filter to minimize data volume by suppressing version information for system processes and system services. As a result, the `AppVersion` field is typically empty for most system processes and services.

Details

- Source type: `uberAgent:Process:ProcessDetail`
- Used in dashboards: *Process Performance*, *Application Performance*, *Machine Performance*, *Application Usage*, *Process GPU*, *Single Machine Detail*, *Single Application Detail*, *Single User Detail*, *Analyze data over time*
- Enabled through configuration setting: `ProcessDetailTop5` or `ProcessDetailFull`

- Related configuration settings: [`ProcessToApplicationMapping`], [`ApplicationMappingIgno`], [`ProcessDetail_SendCommandline`], [`ProcessStartupSettings`]

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Measurement type	Platform	Example
ProcName	Process name.	String		Snapshot	all	chrome.exe
ProcCPUTimeMs	Process CPU usage (absolute usage in milliseconds).	Number	ms	Sum	all	5000
ProcCPUPercent	Process CPU usage (relative usage in percent).	Number	%	Average	all	12
ProcIOPSRead	Process I/O read operations per second.	Number		Average	Win	200
ProcIOPSWrite	Process I/O write operations per second.	Number		Average	Win	200
ProcIOReadCount	Process I/O read operation count.	Number		Count	Win	100
ProcIOWriteCount	Process I/O write operation count.	Number		Count	Win	100
ProcIOReadMB	Process I/O read data volume.	Number	MB	Sum	all	150

Field	Description	Data type	Unit	Measurement type	Platform	Example
ProcIOWriteMB	Process I/O write data volume.	Number	MB	Sum	all	150
ProcIOLatencyReadMs2	Latency of process I/O read operations.	Number	ms	Average	Win	300
ProcIOLatencyWriteMs2	Latency of process I/O write operations.	Number	ms	Average	Win	300
ProcWorkingSetMB	Process RAM usage (working set).	Number	MB	Snapshot	all	100
ProcNetKBPS	Process network traffic data volume per second.	Number	KB	Sum	all	500
ProcUser	Process user.	String		Snapshot	all	Domain\JohnDoe
ProcGpuComputePercent	Process GPU compute usage (relative usage in percent).	Number	%	Average	Win	20
ProcGpuMemMB	Process GPU memory usage.	Number	MB	Average	Win	150

Field	Description	Data type	Unit	Measurement type	Platform	Example
AppId	Associated application ID. Used by uberAgent to lookup application names and populate field `AppName`.	String		Snapshot	all	GglChrm
AppVersion	Associated application version.	String		Snapshot	all	67.0.3396.99
ProcID	Process ID generated by the OS. Process IDs are reused and cannot be used to uniquely identify a process. Use `ProcGUID` for that purpose instead.	Number		Snapshot	all	456
ProcCmdline	The process' command line.	String		Snapshot	all	C:\Program Files (x86)\Google\Chrome\Ap –url http://vastlimits.com

Field	Description	Data type	Unit	Measurement type	Platform	Example
ProcGUID	Unique identifier for a process instance that is generated by uberAgent.	String		Snapshot	all	00000000-ebe5-469c-63ae-f5a1de28d401
ProcGpuEngineMostUsed	The ID of the most used GPU engine (requires at least Windows 10 1709).	Number		Snapshot	Win	1
ProcGpuEngineMostUsedDisplayName	The display name of the most used GPU engine (requires at least Windows 10 1709).	String		Snapshot	Win	3D

Field	Description	Data type	Unit	Measurement type	Platform	Example
SessionID	Session ID generated by the OS. Session IDs are reused and cannot be used to uniquely identify a session. Use `SessionGUID` for that purpose instead.	Number		Snapshot	all	1

Notes

- The following fields are empty unless `EnableExtendedInfo` is set to true: `ProcGUID`.
- The following field is empty unless `EnableExtendedInfo` and `[ProcessDetail_SendCommandlin` `]` are configured: `ProcCmdline`.

List of Calculated Fields

Field	Description	Data type	Unit	Measurement type	Where available	Example
ProcCPUTimeS	Process CPU usage (absolute usage in seconds).	Number	s	Sum	Splunk data model	5
ProcIOCount	`ProcIOReadCount` + `ProcIOWriteCount`.	String		Sum	Splunk data model	200

Field	Description	Data type	Unit	Measurement type	Where available	Example
ProcIOPS	`ProcIOPSRead` + `ProcIOPSWrite`.	Number		Sum	Splunk data model	400
ProcIOMB	`ProcIOReadMB` + `ProcIOWriteMB`.	Number	MB	Sum	Splunk data model	300
ProcIOMBPS	`ProcIOMB` / `ProcIOCount` x `ProcIOPS`.	Number	MB	Sum	Splunk data model	600
ProcIOLatencyMs	`ProcIOLatencyReadMs` + `ProcIOLatencyWriteMs`.	Number	ms	Sum	Splunk data model	600
ProcIODurationReadMS	`ProcIOLatencyRead` x `ProcIOCountRead`.	Number	ms	Sum	Splunk data model	30000
ProcIODurationWriteMS	`ProcIOLatencyWrite` x `ProcIOCountWrite`.	Number	ms	Sum	Splunk data model	30000
ProcIODurationMS	`ProcIODurationReadMS` + `ProcIODurationWriteMS`.	Number	ms	Sum	Splunk data model	60000
User	Alias for `ProcUser`.	String		Snapshot	Splunk data model	Domain\JohnDoe

Field	Description	Data type	Unit	Measurement type	Where available	Example
AppName	Associated application name.	String		Snapshot	Splunk data model, Splunk SPL	Google Chrome
time	Alias for `_time`.	Number		Snapshot	Splunk data model	2018-07-31T18:34:32.451+02:00

Process Statistics

uberAgent collects rich metrics per process. This includes the handle count, page faults, and priority, to name just a few.

Notes:

- Processes are auto-grouped into applications, i.e., the application name is determined automatically. Information on how automatic application identification works is available here.

Details

- Source type: `uberAgent:Process:ProcessStatistics`
- Used in dashboards: *Process Performance*, *Application Performance*, *Single Application Detail*, , *Analyze data over time*
- Enabled through configuration setting: `ProcessStatistics`
- Related configuration settings:

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Measurement type	Platform	Example
ProcName	Process name.	String		Snapshot	all	chrome.exe

Field	Description	Data type	Unit	Measurement type	Platform	Example
ProcID	Process ID generated by the OS. Process IDs are reused and cannot be used to uniquely identify a process. Use `ProcGUID` for that purpose instead.	Number		Snapshot	all	456
ProcGUID	Unique identifier for a process instance that is generated by uberAgent.	String		Snapshot	all	00000000-ebe5-469c-63ae-f5a1de28d401
ProcUser	Process user.	String		Snapshot	all	Domain\JohnDoe
AppId	Associated application ID. Used by uberAgent to lookup application names and populate field `AppName`.	String		Snapshot	all	GglChrm

Field	Description	Data type	Unit	Measurement type	Platform	Example
ProcHandleCount	On Windows: Handle count. On macOS: Number of open file descriptors.	Number		Snapshot	all	810
ProcThreadCount	Process thread count.	Number		Snapshot	all	20
ProcPriority	Process priority. Valid values on Windows: `0`, `1`, `2`, `3`, `4`, `5` or `6`. The numerical priority is used by uberAgent to look up and populate the field `ProcPriorityDisplayName`. Valid values on macOS: `0-127` the higher the number, the higher the current process priority is.	Number		Snapshot	all	4

Field	Description	Data type	Unit	Measurement type	Platform	Example
ProcPrivateMB	Process allocated private memory.	Number	MB	Snapshot	Win	512
ProcVirtualSizeMB	Process allocated virtual memory.	Number	MB	Snapshot	Win	128
ProcPageFaultsPS	Process page faults per second.	Number	s	Average	all	10
ProcPageFileMB	Process page file usage.	Number	MB	Snapshot	Win	256

List of Calculated Fields

Field	Description	Data type	Unit	Measurement type	Where available	Example
ProcPriorityDisplayName	Process priority display name (see also `ProcPriority`). Valid values (Windows): `Above normal`, `Below normal`, `High`, `Idle`, `Normal`, `Realtime` and `Unknown`.	String		Snapshot	Splunk data model, Splunk SPL	Above normal

Application Inventory & Usage Metrics

Application Inventory

uberAgent collects application inventory information like name, publisher, version, and install date.

Notes:

- Processes are auto-grouped into applications, i.e. the application name is determined automatically without requiring any configuration. Information on how this works are available here.
- Field: `AppVersion` - uberAgent has an internal filter to minimize data volume by suppressing version information for system processes and system services. As a result, the `AppVersion` field is typically empty for most system processes and services.

Details

- Source type: `uberAgent:Application:ApplicationInventory`

- Used in dashboards: *Application Inventory*, *Single Application Inventory*, *Single Machine Inventory*
- Enabled through configuration setting: `ApplicationInventory`
- Related configuration settings: *n/a*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Measurement type	Platform	Example
DisplayName	Application name	String		Snapshot	all	Google Chrome
DisplayVersion	Application version	String		Snapshot	all	67.0.3396.99
Publisher	Application publisher	String		Snapshot	all	Google
InstallDate	Application installation date	String		Snapshot	all	2018-03-19
Language	Application language	String		Snapshot	Win	en-US
IsMsiPackage	Indicates if application was installed through a MSI	String		Snapshot	Win	1
IsMachineInstall	Indicates if application is installed for all users	String		Snapshot	all	1
InstallLocation	Application installation location	String		Snapshot	all	C:\Program Files (x86)\Google\Chrome\Ap

Application Usage

DEPRECATED: this sourcetype is deprecated and will be removed in a future release. uberAgent's dashboards use the ProcessDetail sourcetype instead.

uberAgent collects application usage information like name (even for App-V, Java, and UWP apps), version, and number of concurrent users.

Details

- Source type: `uberAgent:Application:ApplicationUsage`
- Used in dashboards: *Application Usage*
- Enabled through configuration setting: `ApplicationUsage`
- Related configuration settings: *n/a*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Measurement type	Platform	Example
AppName	Application name	String		Snapshot	all	Google Chrome
UserName	User name	String		Snapshot	all	Domain\JohnDoe
AppVersion	Application version	String		Snapshot	all	67.0.3396.99
RemotingClientName	Remoting client name	String		Snapshot	Win	ThinClient1

Application Name to ID Mapping Metrics

Application Name to ID Mapping

uberAgent maps application names to IDs to save data volume sent over the network.

Note: processes are auto-grouped into applications, i.e. the application name is determined automatically without requiring any configuration. Information on how this works are available here.

Details

- Source type: `uberAgent:Application:AppNameIdMapping`
- Used in dashboards: *n/a*
- Enabled through configuration setting: *AppNameIdMapping*
- Related configuration settings: *n/a*
- Supported platform: *all*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Measurement type	Example
AppName	Application name.	String		Snapshot	Google Chrome
AppId	Associated application ID. Used by uberAgent to lookup application names and populate field `AppName`.	Number		Snapshot	GlgChrm

Application & Process Startup Metrics

Process Startup

For each application or process that is being launched, uberAgent collects metrics like startup performance (duration, IOPS), as well as process properties (e.g., elevation status). If the configuration setting `EnableExtendedInfo` is enabled, uberAgent also collects metrics like the full path to the process executable in the file system as well the full command line the process was launched with.

Notes:

- As with all other metrics, process startup duration is recorded automatically without requiring any configuration. uberAgent optionally only shows new processes never seen before in the Splunk dashboards.
- Processes are auto-grouped into applications, i.e., the application name is determined automatically. Information on how automatic application identification works is available here.
- A `fork` or `exec` system call can trigger process startup events on macOS. To distinguish between both, see the `StartupEventSource` field.
- Field: `AppVersion` - uberAgent has an internal filter to minimize data volume by suppressing version information for system processes and system services. As a result, the `AppVersion` field is typically empty for most system processes and services.

Details

- Source type: uberAgent:Process:ProcessStartup
- Used in dashboards: *Application Startup*, *Process Startup*, *Single Application Detail*, *Analyze data over time*
- Enabled through configuration setting: ProcessStartup
- Related configuration settings: [ProcessStartupSettings], [ProcessStartupDurationWaitI
  ]

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Measurement type	Platform	Example
ProcName	Process name.	String		Snapshot	all	chrome.exe
ProcUser	Process user.	String		Snapshot	all	Domain\JohnDoe
StartupTimeMs	Startup time duration (how long it took the process to initialize).	Number	ms	Sum	Win	300
StartupIOPS	I/O operations per second generated during the process’ startup phase (see StartupTimeMs.	Number		Count	Win	150

Field	Description	Data type	Unit	Measurement type	Platform	Example
AppId	Associated application ID. Used by uberAgent to look up the application name and populate the field `AppName`.	String		Snapshot	all	GglChrm
ProcID	Process ID generated by the OS. Process IDs are reused and cannot be used to uniquely identify a process. Use `ProcGUID` for that purpose instead.	Number		Snapshot	all	456
ProcParentID	Parent process ID (also see `ProcID`).	Number		Snapshot	all	789

Field	Description	Data type	Unit	Measurement type	Platform	Example
SessionID	Session ID generated by the OS. Session IDs are reused and cannot be used to uniquely identify a session. Use `SessionGUID` for that purpose instead. macOS: for consistency with Windows all non-user sessions are assigned to a fictitious system session 0.	Number		Snapshot	all	3
ProcGUID	Unique identifier for a process instance that is generated by uberAgent.	String		Snapshot	all	00000000-ebe5-469c-63ae-f5a1de28d401

Field	Description	Data type	Unit	Measurement type	Platform	Example
SessionGUID	Unique identifier for a session that is generated by uberAgent.	String		Snapshot	Win	00000002-f295-9109-e7c7-c964011dd401
ProcParentName	Parent process name.	String		Snapshot	all	powershell.exe
ProcPath	Full path to the process executable in the file system.	String		Snapshot	all	C:\Windows\System32\W
ProcCmdline	The process' command line.	String		Snapshot	all	C:\Program Files (x86)\Google\Chrome\Ap –url http://vastlimits.com
StartupEventSource	Indicates if the startup event was generated by a `fork` (1) or `exec` (2) call.	String		Snapshot	macOS	1
IsElevated	Indicates if the process is elevated (has admin rights).	String		Snapshot	all	1
AppVersion	Associated application version.	String		Snapshot	all	67.0.3396.99

Field	Description	Data type	Unit	Measurement type	Platform	Example
ProcParentGUID	Unique identifier of the parent process (the parent’s `ProcGUID`).	String		Snapshot	all	00000000-ebe5-469c-54ae-f5a1de28d401
IsProtected	Indicates whether the process is protected (`PsProtectedTypeProtected`) or protected light (`PsProtectedTypeProtectedLight`, PPL).	String		Snapshot	Win	1
HashMD5	MD5 hash of the process executable (requires ESA). Configurable via settings `EnableCalculateHash` and `HashAlgorithm`.	String		Snapshot	Win	7FFE122B109F1B586DEA

Field	Description	Data type	Unit	Measurement type	Platform	Example
HashSHA1	SHA1 hash of the process executable (requires ESA). Configurable via settings `EnableCalculateHash` and `HashAlgorithm`.	String		Snapshot	Win	26DBC241A37881072689
HashSHA256	SHA256 hash of the process executable (requires ESA). Configurable via settings `EnableCalculateHash` and `HashAlgorithm`.	String		Snapshot	Win	95F0FBBAEF2899923859
HashIMP	Import-table hash of the process executable (requires ESA). Configurable via settings `EnableCalculateHash` and `HashAlgorithm`.	String		Snapshot	Win	188392D5FBCC485811B

Field	Description	Data type	Unit	Measurement type	Platform	Example
SignatureStatus	Authenticode signature status. Can be `0`, `1`, `2`, `3`, `4`, `5`, `6` or `7`. See also `SignatureStatusDisplayName` (requires ESA).	String		Snapshot	Win	1
IsSignedByOSVendor	Indicates whether the Authenticode signer is the OS manufacturer (e.g., Microsoft). Requires ESA.	String		Snapshot	All	1
SignerName	Authenticode signer name (requires ESA).	String		Snapshot	Win	Microsoft Windows
CdHash	Hash of the code directory of a signed executable (requires ESA). Configurable via setting `EnableCdHash` .	String		Snapshot	macOS	24e4b80198b220e4a0ea8

Notes

- The following fields are empty unless `EnableExtendedInfo` is set to true: `ProcID`, `ProcParentID`, `SessionID`, `ProcGUID`, `SessionGUID`, `ProcParentName`, `ProcPath`, `ProcCmdline`, `ProcParentGUID`.
- The maximum supported timer `Interval` for the `ProcessStartup` metric is `300000` (5 minutes).

List of Calculated Fields

Field	Description	Data type	Unit	Measurement type	Where available	Example
User	Alias for `ProcUser`.	String		Snapshot	Splunk data model	Domain\JohnDoe
StartupTimeS	Startup time duration.	Number	s	Sum	Splunk data model	0.3
StartupIOCount	`StartupIOPS` * `StartupTimeMs` / 1000.	Number		Sum	Splunk data model	45
AppName	Associated application name.	String		Snapshot	Splunk data model, Splunk SPL	Google Chrome
SignatureStatusDisplayName	Possible values: `Unknown`, `Ok`, `Revoked`, `Expired`, `InvalidHash`, `UntrustedRoot`, `TrustedRootNotInCA` and `Error`.	String		Snapshot	Splunk data model	Ok

Application UI Unresponsiveness Metrics

UI Unresponsiveness

Whenever an application's user interface is unresponsive for more than a few hundred milliseconds uberAgent collects information like the application name (even for App-V, Java, and UWP apps), application version and the unresponsiveness duration.

Notes:

- Processes are auto-grouped into applications, i.e. the application name is determined automatically without requiring any configuration. Information on how this works are available here.
- Field: `AppVersion` - uberAgent has an internal filter to minimize data volume by suppressing version information for system processes and system services. As a result, the `AppVersion` field is typically empty for most system processes and services.

Details

- Source type: `uberAgent:Application:UIDelay`
- Used in dashboards: *Application UI Unresponsiveness*, *Process UI Unresponsiveness,__Single Machine Detail*, *Single Application Detail*, *Single User Detail*
- Enabled through configuration setting: `ApplicationUIDelay`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Measurement type	Example
AppId	Associated application ID. Used by uberAgent to lookup application names and populate field `AppName`.	String		Snapshot	GglChrm

Field	Description	Data type	Unit	Measurement type	Example
AppVersion	Application version.	String		Snapshot	67.0.3396.99
ProcessName	Process name.	String		Snapshot	chrome.exe
ProcessId	Process ID.	Number		Snapshot	456
UIDelayMs	UI delay/unresponsiveness.	Number	ms	Sum	5000
User	User name.	String		Snapshot	Domain\JohnDoe
SessionGUID	Unique identifier that is generated by uberAgent when the session is created. Valid for this session only.	String		Snapshot	00000002-f295-9109-e7c7-c964011dd401
HasFocus	Indicates if the application is in the foreground (is being used actively). Possible values: `0`, `1`.	String		Snapshot	1

List of Calculated Fields

Field	Description	Data type	Unit	Measurement type	Where available	Example
AppName	Associated application name.	String		Snapshot	Splunk data model, Splunk SPL	Google Chrome

Field	Description	Data type	Unit	Measurement type	Where available	Example
UIDelayS	UI delay/unresponsiveness.	Number	s	Sum	Splunk data model	5
HasFocusNumber	Indicates if the application is in the foreground (is being used actively). Possible values: `0`, `1`.	Number		Snapshot	Splunk data model	1
ProcName	Process name.	String		Snapshot	Splunk data model	chrome.exe

Software Update Metrics

Software Update

uberAgent collects the name and install date per update.

Details

- Source type: `uberAgent:Application:SoftwareUpdateInventory`
- Used in dashboards: *Update inventory*, *Single Machine Detail*, *Single Update Detail*
- Enabled through configuration setting: `SoftwareUpdateInventory`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Measurement type	Example
Guid	GUID.	String		Snapshot	AC76BA86-7AD7-0000-2550-AC120B4E4800
DisplayName	Update name.	String		Snapshot	KB2565063
ProductName	Product name.	String		Snapshot	Microsoft Visual C++ 2010 x86 Redistributable - 10.0.40219
State	Update state. Possible values: `applied`, `superseded`, `obsolete`.	String		Snapshot	applied
InstallDate	Installation date.	String		Snapshot	2018-04-13

Browser Metrics

Google Chrome Metrics

Google Chrome

uberAgent collects performance data per Chrome (Edge, Citrix Enterprise Browser) process type like Browser (main process) or GPU (graphics acceleration).

Details

- Source type: `uberAgent:Application:BrowserPerformanceChrome`
- Used in dashboards: *Browser Performance: Chrome*, *Browser Performance: Edge*, *Browser Performance: CEB*, *Analyse data over time*

- Enabled through configuration setting: `BrowserPerformanceChrome`,`BrowserPerformanceEdge`,`BrowserPerformanceCEB`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
ProcUser	User who ran the process.	String		Domain\JohnDoe
ProcType	Process type. Possible values: `Browser`, `Extension`, `Flash`, `GPU`, `Java`, `Tab`.	String		Tab
CPUTimeMs	Consumed CPU time.	Number	ms	11859
CPUPercent	Consumed CPU performance.	Number		5.05
IOPS	I/O operations per second.	Number		2000
IOCount	Count of I/O operations.	Number		500
IOMB	Amount of data I/O operation volume.	Number	MB	200
IOLatencyMs	I/O operation duration divided by count of I/O operations.	Number	ms	1.20
WorkingSetMB	Consumed RAM.	Number	MB	68.08
NetKBPS	Generated network traffic.	Number	KB	19.06

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
User	User who ran the process.	String		Domain\JohnDoe	Splunk data model
CPUTimeS	Consumed CPU time.	Number	s	11.859	Splunk data model
IOMBPS	Amount of data I/O operation volume per second.	Number	MB	2	Splunk data model
IODurationMs	I/O operation duration.	Number	ms	600	Splunk data model

Internet Explorer Metrics

Internet Explorer

Contains metrics per website (URL) like CPU and RAM usage.

Details

- Source type: `uberAgent:Application:BrowserPerformanceIE`
- Used in dashboards: *Browser Performance: Internet Explorer*, *Analyze data over time*
- Enabled through configuration setting: `BrowserPerformanceIE`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
ProcID	Process ID.	Number		456

Field	Description	Data type	Unit	Example
ProcType	Process type. Possible values: `frame`, `undefined`, `worker`.	String		frame
URL	URL.	String		https://www.vastlimits.com
CPUTimeMs	Consumed CPU time.	Number	ms	11859
CPUPercent	Consumed CPU performance.	Number	%	5.05
IOPS	I/O operations per second.	Number		2000
IOCount	Count of I/O operations.	Number		500
IOMB	Amount of data I/O operation volume.	Number	MB	200
IOLatencyMs	I/O operation duration divided by count of I/O operations.	Number	ms	1.20
WorkingSetMB	Consumed RAM.	Number	MB	68.08
NetKBPS	Generated network traffic.	Number	KB	19.06

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
URLHost	URL host address.	String		www.vastlimits.com	Splunk data model
CPUTimeS	Consumed CPU time.	Number	s	11.859	Splunk data model

Field	Description	Data type	Unit	Example	Where available
IOMBPS	Amount of data I/O operation volume per second.	Number	MB	2	Splunk data model
IODurationMs	I/O operation duration.	Number	ms	600	Splunk data model

Web App Metrics

Browser Web Requests

uberAgent determines UX metrics for SaaS and web apps like page load and XMLHttpRequest duration for every web request made.

Details

- Source type: `uberAgent:Application:BrowserWebRequests2`
- Used in dashboards: *Browser Web App Performance*
- Enabled through configuration setting: `BrowserPerformanceChrome`, `BrowserPerformanceEdge`, `BrowserPerformanceFirefox`, `BrowserPerformanceIE`, `BrowserPerformanceCEB`
- Related configuration settings: `[BrowserWebAppURL_Filter]`
- Supported platforms: *all*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
Browser	Browser ID. Possible values: 1, 2, 3,4,5. Also see the field BrowserDisplayName below.	Number		1
BrowserVersion	Browser version.	String		69.0.3497.81
IsFrame	Indicates if a page is a frame or not. Possible values: 0, 1.	String		1
ProcUser	User who ran the process.	String		Domain\JohnDoe
TabHost	Host address of the entered URL.	String		vastlimits.com
TabUriScheme	Uri scheme ID of the entered URL. Possible values: 0, 1, 2, 3, 4. Also see the field TabUriSchemeDisplayName below.	Number		3
RequestType	Type of the requested URL. Possible values: 0, 1, 2, 3. Also see the field RequestTypeDisplayName below.	String		1
RequestHost	Host address which was requested by the TabHost address.	String		uberagent.com

Field	Description	Data type	Unit	Example
RequestUriScheme	Uri scheme ID of the requested URL. Possible values: `0`, `1`, `2`, `3`, `4`. Also see the field `RequestUriSchemeDisplayName` below.	Number		3
RequestCount	Total count of requests	Number		40
WebRequestDurationMs	Duration for all web requests. Does not work with Internet Explorer.	Number	ms	140
PageLoadTotalDurationMs	Total duration for page load.	Number	ms	300
PageLoadNetworkDurationMs	Network duration for page load.	Number	ms	150
PageLoadRenderDurationMs	Render duration for page load.	Number	ms	150
HttpStatusCount2xx	Count of 2xx http errors.	Number		2
HttpStatusCount3xx	Count of 3xx http errors.	Number		3
HttpStatusCount4xx	Count of 4xx http errors.	Number		4
HttpStatusCount5xx	Count of 5xx http errors.	Number		5

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
BrowserDisplayName	Browser name. Possible values: `Chrome` ,`Internet Explorer` ,`Firefox` ,`Edge` ,`Citrix Enterprise Browser`.	String		Chrome	Splunk data model, Splunk SPL
User	User name.	String		Domain\JohnDoe	Splunk data model
TabUriSchemeDisplayName	Uri scheme name of the entered URL. Possible values: `Other`, `http`, `https`, `ftp`, `file`.	String		https	Splunk data model, Splunk SPL
RequestUriSchemeDisplayName	Uri scheme name of the requested URL. Possible values: `Other`, `http`, `https`, `ftp`, `file`.	String		https	Splunk data model, Splunk SPL

Field	Description	Data type	Unit	Example	Where available
RequestTypeDisplayName	Type of the requested URL. Possible values: `Unknown`, `XmlHttpRequest`, `WebSocket`, `PageLoad`.	String		XmlHttpRequest	Splunk data model, Splunk SPL

Citrix NetScaler (ADC) Metrics

Citrix NetScaler (ADC) monitoring requires its metrics to be enabled on at least one machine. See the documentation for details.

Appliance Inventory

uberAgent collects metrics like MAC address, model, and version for the primary as well as the secondary appliance.

Details

- Source type: `uberAgent:CitrixADC:ApplianceInventory`
- Used in dashboards: *NetScaler Inventory, Single NetScaler Inventory*
- Enabled through configuration setting: `CitrixADCInventory`
- Related configuration settings: `[CitrixADC_Config]`
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
HostName	Appliance name.	String		CitrixADC1

Field	Description	Data type	Unit	Example
Version	Firmware version.	String		NS12.0: Build 59.8.nc Date: Sep 12 2018 06:16:17
License	Installed license.	String		Platinum
Model	Appliance model.	Number		1000
LicensingMode	Configured licensing mode. Possible values: `Local`, `Pooled`, `CICO`.	String		Local
HwDescription	Hardware and its ports detail.	String		NetScaler Virtual Appliance.
SysId	System id.	String		450088
ManufactureDay	Day the appliance was manufactured.	Number		17
ManufactureMonth	Month the appliance was manufactured.	Number		2
ManufactureYear	Year the appliance was manufactured.	Number		2009
CpuFrequncy	CPU frequency.	Number		3890
SerialNo	Serial number.	Number		ABCDEFG12
EncodedSerialno	Encoded serial number.	String		68310055cb267307ffg8
MACAddress	MAC address.	String		00155d3d890d
HostId	Host id.	String		123456789
LastConfigSaveTime	Time when the configuration was last saved.	String		2018-12-07 12:06:36
Timezone	Timezone.	String		GMT+01:00-CET-Europe/Berlin
WeakPassword	List of all weak passwords.	String		password

Field	Description	Data type	Unit	Example
SystemType	System type. Possible Values: `Standalone`, `HA`, `Cluster`.	String		HA
LastConfigChangedTime	Time when the configuration was last changed.	String		2018-12-07 12:06:21
LastStartupTime	Time when the appliance was last started.	String		2018-11-24 15:26:25
CurrentHAStatus	Indicates the high availability state of the node. Possible values: `Staysecondary`, `Primary`, `Secondary`, `Claiming`, `Force change`.	String		Primary
LastHAStatusChange	Time when the last master state transition occurred.	String		2018-12-04 10:34:51
SSLCards2	Number of available SSL cards.	Number		0
SSLCardsUp2	Number of working SSL cards.	Number		0

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
SSLCards	Number of available SSL cards.	Number		0	Splunk data model
SSLCardsUp	Number of working SSL cards.	Number		0	Splunk data model

Appliance Performance

uberAgent collects metrics like CPU usage, RAM usage as well as network throughput for the primary appliance.

Details

- Source type: `uberAgent:CitrixADC:AppliancePerformance`
- Used in dashboards: *NetScaler Appliance Performance*
- Enabled through configuration setting: `CitrixADCPerformance`
- Related configuration settings: `[CitrixADC_Config]`
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
HostName	Appliance name.	String		CitrixADC1
MemUsagePcnt	Memory usage.	Number	%	17.62
MgmtCpuUsagePcnt	Management CPU usage.	Number	%	14.6
PktCpuUsagePcnt	Packet CPU usage.	Number	%	3
Disk0PerUsage	/flash disk usage.	Number	%	41
Disk0Size	/flash disk size.	Number	MB	1585
Disk0Used	/flash disk allocated space.	Number	MB	598

Field	Description	Data type	Unit	Example
Disk0Avail	/flash disk free space.	Number	MB	860
Disk1PerUsage	/var disk usage.	Number	%	45
Disk1Size	/var disk size.	Number	MB	14179
Disk1Used	/var disk allocated space.	Number	MB	5988
Disk1Avail	/var disk free space.	Number	MB	7056
ApplianceBytesReceived	Amount of incoming network data.	Number	Bytes	608359
ApplianceBytesSent	Amount of outgoing network data.	Number	Bytes	616856
NumCpus2	Number of CPUs.	Number		3
MemSizeMB2	Memory size.	Number	MB	8096
MemUseInMB2	Memory usage.	Number	MB	196
CpuFan0Speed	CPU fan 0 speed. Acceptable range is 3000 through 6000 RPM.	Number		5000
CpuFan1Speed	CPU fan 1 speed. Acceptable range is 3000 through 6000 RPM.	Number		5000
SystemFanSpeed	System fan speed. Acceptable range is 3000 through 6000 RPM.	Number		5000
Cpu0Temp	CPU 0 temperature.	Number		80
Cpu1Temp	CPU 1 temperature.	Number		80

Field	Description	Data type	Unit	Example
InternalTemp	Internal temperature of health monitoring chip.	Number		80
PowerSupply1Status	Power supply 1 failure status. Can be `0`, `1`, `2`, `3`, or `4`. See also `PowerSupply1StatusDisplayName`.	Number		1
PowerSupply2Status	Power supply 2 failure status. Can be `0`, `1`, `2`, `3`, or `4`. See also `PowerSupply2StatusDisplayName`.	Number		1
PowerSupply3Status	Power supply 3 failure status. Can be `0`, `1`, `2`, `3`, or `4`. See also `PowerSupply3StatusDisplayName`.	Number		1
PowerSupply4Status	Power supply 4 failure status. Can be `0`, `1`, `2`, `3`, or `4`. See also `PowerSupply4StatusDisplayName`.	Number		1
VoltageV33Main	Main power supply +3.3V output. Acceptable range is 2.970 through 3.630 volts.	Number		3.630
ICAOnlySessions	Number of ICA sessions.	Number		45

Field	Description	Data type	Unit	Example
ICAOnlyConnections	Number of ICA connections.	Number		45
SmartAccessSessions	Number of VPN sessions.	Number		45
SmartAccessICAConnections	Number of VPN ICA connections.	Number		45
SSLSessions	Number of SSL sessions.	Number		400

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
ApplianceMbpsSent	Amount of outgoing network data.	Number	Mbps	602.39	Splunk data model
NumCpus	Number of CPUs.	Number		3	Splunk data model
MemSizeMB	Memory size.	Number	MB	8096	Splunk data model
MemUseInMB	Memory usage.	Number	MB	196	Splunk data model
PowerSupply1StatusDisplayName	Power supply 1 failure status. Can be `Unknown`, `Normal`, `Failed`, `Not present`, or `Not supported`. Based on the lookup file `citrixadc_powersupplystatus.csv`.	Number		Normal	Splunk data model, Splunk SPL

Field	Description	Data type	Unit	Example	Where available
PowerSupply2StatusDisplayName	Power supply 2 failure status. Can be `Unknown`, `Normal`, `Failed`, `Not present`, or `Not supported`. Based on the lookup file `citrixadc_powersupplystatus.csv`.	Number		Normal	Splunk data model, Splunk SPL
PowerSupply3StatusDisplayName	Power supply 3 failure status. Can be `Unknown`, `Normal`, `Failed`, `Not present`, or `Not supported`. Based on the lookup file `citrixadc_powersupplystatus.csv`.	Number		Normal	Splunk data model, Splunk SPL

Field	Description	Data type	Unit	Example	Where available
PowerSupply4StatusDisplayName	Power supply 4 failure status. Can be `Unknown`, `Normal`, `Failed`, `Not present`, or `Not supported`. Based on the lookup file `citrixadc_powersupplystatus.csv`.	Number		Normal	Splunk data model, Splunk SPL

Certificates

uberAgent collects certificate metrics like name, type, and expiration date for the primary as well as the secondary appliance.

Details

- Source type: `uberAgent:CitrixADC:Certificate`
- Used in dashboards: *Single NetScaler Inventory*
- Enabled through configuration setting: `CitrixADCInventory`
- Related configuration settings: `[CitrixADC_Config]`
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
HostName	Appliance name.	String		CitrixADC1
CertName	Certificate name.	String		nsmgmtcer
CertFile	Certificate file.	String		nsmgmt.cer

Field	Description	Data type	Unit	Example
CertType	Certificate type. Possible values: `Client`, `Server`, `Client;Server`, `Root`, `Intermediate`.	String		Client;Server
CertExpirationDate	Certificate expiration date.	String		2020-11-22 11:06:46

Gateways

uberAgent collects gateway metrics like network throughput and SSL as well as authentication policies for the primary appliance.

Details

- Source type: `uberAgent:CitrixADC:Gateway`
- Used in dashboards: *NetScaler Gateway Performance*,*Single NetScaler Inventory*
- Enabled through configuration setting: `CitrixADCGateway`
- Related configuration settings: `[CitrixADC_Config]`
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
HostName	Appliance name.	String		CitrixADC1
vServerName	Gateway name.	String		_XD_10.1.1.22_443
PrimaryIpAddress	IP address.	IPv4		10.1.1.22
PrimaryPort	Port.	Number		443
Type	vServer type. Possible values: `SSL`.	String		SSL

Field	Description	Data type	Unit	Example
State	vServer state.	String		Up
RequestBytesRate	Incoming network traffic.	Number	Bytes	1024
ResponseBytesRate	Outgoing network traffic.	Number	Bytes	1024
AuthLDAPPolicies	Bound LDAP authentication policies.	String		pol_ldap1;pol_ldap2
AuthRadiusPolicies	Bound Radius authentication policies.	String		pol_radius1;pol_radius2
AuthSAMLPolicies	Bound SAML authentication policies.	String		pol_saml1
SessionPolicies	Bound session policies.	String		sp_storefront
STAs	Secure ticket authorities.	String		http://DDC.vastlimits.com
ResponderPolicies	Bound responder policies.	String		rp_storefront
RewritePolicies	Bound rewrite policies.	String		rw_clientname
Certificates	Bound certificates.	String		wildcard.vastlimits.com
PortalTheme	Bound portal theme.	String		Default
SSL2*	Indicates if SSL2 is enabled.	String		0
SSL3*	Indicates if SSL3 is enabled.	String		0
TLS1*	Indicates if TLS1 is enabled.	String		0
TLS11*	Indicates if TLS1.1 is enabled.	String		0
TLS12*	Indicates if TLS1.2 is enabled.	String		1

Field	Description	Data type	Unit	Example
DiffeHellman*	Indicates if Diffie-Hellman is enabled.	String		1
SSLProfile	Name of the bound SSL profile.	String		ns_default_ssl_profile_fronter
Ciphers*	Bound ciphers.	String		DEFAULT
CipherSuites*	Bound cipher suites.	String		DEFAULT
TotalRequests2	Total number of requests received.	Number		456789
TotalResponses2	Total number of responses.	Number		456789
SessionTimeout2*	Session timeout.	Number		120
HSTS	State of HSTS protocol support for the SSL Virtual Server. Using HSTS, a server can enforce the use of an HTTPS connection for all communication with a client. Can be `Enabled` or `Disabled`.	String		Disabled
HSTSMaxAge	The maximum time, in seconds, in the strict transport security (STS) header during which the client must send only HTTPS requests to the server.	Number		0

Field	Description	Data type	Unit	Example
HSTSInclSubdom	Indicates if HSTS for subdomains is enabled or disabled. If set enabled, a client must send only HTTPS requests for subdomains. Can be `Yes` or `No`.	String		No
TLS13	Indicates if TLS1.3 is enabled.	String		Enabled
VpnUsers	Number of users on this gateway .	Number		10
SslVpnUsers	Number of AAA sessions on this gateway.	Number		56

** If an SSL profile is bound, then these settings are taken from it.*

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
RequestResponseBytesRate	`RequestBytesRate` + `ResponseBytesRate`.	Number	Bytes	2048	Splunk data model
RequestResponseRateMbps	`round(RequestResponseBytesRate/125000,1)`	Number	Mbps	2	Splunk data model
TotalRequests2	Total number of requests received.	Number		456789	Splunk data model
TotalResponses2	Total number of responses.	Number		456789	Splunk data model

Field	Description	Data type	Unit	Example	Where available
SessionTimeout2	Session timeout.	Number		120	Splunk data model

Ips

uberAgent collects Ip metrics like address, type, and if it is used for management access for the primary as well as the secondary appliance.

Details

- Source type: `uberAgent:CitrixADC:Ip`
- Used in dashboards: *Single NetScaler Inventory*
- Enabled through configuration setting: `CitrixADCInventory`
- Related configuration settings: [`CitrixADC_Config`]
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
HostName	Appliance name.	String		CitrixADC1
IpAddress	IP address.	IPv4		10.1.1.25
IpNetmask	IP netmask.	String		255.255.0.0
IpType	IP type. Possible values: `SNIP`, `VIP`, `NSIP`, `GSLBsiteIP`, `CLIP`, `LSN`.	String		VIP
IpvServer	Indicates if the IP is bound to a vServer.	String		Yes
IpMgmtAccess	Indicates if management access is allowed.	String		0

Services

uberAgent collects service metrics like protocol, state as well as bound monitors for the primary appliance.

Details

- Source type: `uberAgent:CitrixADC:Service`
- Used in dashboards: *Single Virtual Server Performance*
- Enabled through configuration setting: `CitrixADCvServer`
- Related configuration settings: `[CitrixADC_Config]`
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
HostName	Appliance name.	String		CitrixADC1
ServiceName	Service name.	String		webserver1

Field	Description	Data type	Unit	Example
Protocol	Service protocol. Possible values: HTTP, FTP, TCP, UDP, SSL, SSL_BRIDGE, SSL_TCP, DTLS, NNTP, DNS, DHCPRA, ANY, SIP_UDP, SIP_TCP, SIP_SSL, DNS_TCP, RTSP, PUSH, SSL_PUSH, RADIUS, RDP, MYSQL, MSSQL, DIAMETER, SSL_DIAMETER, TFTP, ORACLE, SMPP, SYSLOGTCP, SYSLOGUDP, FIX, SSL_FIX, USER_TCP, USER_SSL_TCP .	String		HTTP
PrimaryIpAddress	IP address.	IPv4		10.1.1.11
PrimaryPort	Port.	Number		80

Field	Description	Data type	Unit	Example
State	Service state. Possible values: `UP`, `DOWN`, `UNKNOWN`, `OFS (Out of Service)`, `TROFS (Transition Out of Service)`, `TROFS_DOWN (Down when going Out of Service)`.	String		Up
vServerName	Name of the vServer the service is bound to.	String		lbvs_CompanyWebserver
ServiceMonitors	Bound service monitors.	String		tcp-default

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
ServiceMonitorsSplit	`ServiceMonitors` split by semicolon.	Number	MB		Splunk data model

Service Groups

uberAgent collects service group metrics like protocol, state as well as bound monitors for the primary appliance.

Details

- Source type: uberAgent:CitrixADC:ServiceGroup
- Used in dashboards: *Single Virtual Server Performance*
- Enabled through configuration setting: CitrixADCvServer
- Related configuration settings: [CitrixADC_Config]
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
HostName	Appliance name.	String		CitrixADC1
ServiceGroupName	Service group name.	String		sgr1
Protocol	Service group protocol: ADNS, DNS, MYSQL, RTSP, SSL_DIAMETER, ADNS_TCP, DNS_TCP, NNTP, SIP_UDP, SSL_TCP, ANY, FTP, RADIUS, SNMP, TCP, DHCPRA, HTTP, RDP, SSL, TFTP, DIAMETER, MSSQL, RPCSVR, SSL_BRIDGE, UDP.	String		HTTP

Field	Description	Data type	Unit	Example
State	Service group state. Possible values: `UP`, `DOWN`, `UNKNOWN`, `OFS (Out of Service)`, `TROFS (Transition Out of Service)`, `TROFS_DOWN (Down when going Out of Service)`.	String		Up
EffectiveState	Indicates the effective servicegroup state based on the state of the bound service items. Possible values: `UP`, `DOWN`, `OUT OF SERVICE`, `PARTIAL-UP`, `PARTIAL-DOWN`.	String		Up
vServerName	Name of the vServer the service group is bound to.	String		lbvs_Storefront
ServiceGroupMembers	Services which are members of the service group.	String		Service1;Service2
ServiceGroupMonitors	Bound service group monitors.	String		ping;tcp

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
ServiceGroupMembersSplit	`ServiceGroupMembers` split by semicolon.	Number	MB	Service1 Service2	Splunk data model
ServiceGroupMonitorsSplit	`ServiceGroupMonitors` split by semicolon.	Number	MB	ping tcp	Splunk data model

Virtual Server

uberAgent collects virtual server (load-balancing virtual server only) metrics like network throughput and bound policies for the primary appliance.

Details

- Source type: `uberAgent:CitrixADC:vServer`
- Used in dashboards: *NetScaler Virtual Server Performance*, Single Virtual Server Performance,*Single NetScaler Inventory*
- Enabled through configuration setting: `CitrixADCGateway`
- Related configuration settings: [`CitrixADC_Config`]
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
HostName	Appliance name.	String		CitrixADC1
vServerName	Virtual server name.	String		vs_Storefront
PrimaryIpAddress	IP address.	IPv4		10.1.1.5
PrimaryPort	Port.	Number		443

Field	Description	Data type	Unit	Example
Type	vServer type. Possible values: HTTP, FTP, TCP, UDP, SSL, SSL_BRIDGE, SSL_TCP, DTLS, NNTP, DNS, DHCPRA, ANY, SIP_UDP, SIP_TCP, SIP_SSL, DNS_TCP, RTSP, PUSH, SSL_PUSH, RADIUS, RDP, MYSQL, MSSQL, DIAMETER, SSL_DIAMETER, TFTP, ORACLE, SMPP, SYSLOGTCP, SYSLOGUDP, FIX, SSL_FIX, USER_TCP, USER_SSL_TCP .	String		SSL
State	vServer state.	String		Up
RequestBytesRate	Incoming network traffic.	Number	Bytes	1024
ResponseBytesRate	Outgoing network traffic.	Number	Bytes	1024
RewritePolicies	Bound rewrite policies.	String		rw_clientname
ResponderPolicies	Bound responder policies.	String		rp_storefront
Certificates	Bound certificates.	String		Cert1;Cert2

Field	Description	Data type	Unit	Example
SSL2*	Indicates if SSL2 is enabled.	String		0
SSL3*	Indicates if SSL3 is enabled.	String		0
TLS1*	Indicates if TLS1 is enabled.	String		0
TLS11*	Indicates if TLS1.1 is enabled.	String		0
TLS12*	Indicates if TLS1.2 is enabled.	String		1
DiffeHellman*	Indicates if Diffie-Hellman is enabled.	String		1
SSLProfile	Name of the bound SSL profile.	String		vastlimits_ssl_profile
Ciphers*	Bound ciphers.	String		Secure_Cipher_Group
CipherSuites*	Bound cipher suites.	String		DEFAULT
ActSvcs2	Active services.	Number		2
VSLBHealth2	The health of the vserver. This gives the percentage of UP services bound to this vserver.	Number	%	100
SessionTimeout2*	Session timeout.	Number		120

Field	Description	Data type	Unit	Example
HSTS	State of HSTS protocol support for the SSL Virtual Server. Using HSTS, a server can enforce the use of an HTTPS connection for all communication with a client. Can be `Enabled` or `Disabled`.	String		Disabled
HSTSMaxAge	The maximum time, in seconds, in the strict transport security (STS) header during which the client must send only HTTPS requests to the server.	Number		0
HSTSInclSubdom	Indicates if HSTS for subdomains is enabled or disabled. If set enabled, a client must send only HTTPS requests for subdomains. Can be `Yes` or `No`.	String		No
TLS13	Indicates if TLS1.3 is enabled.	String		Enabled
HitsRate	Rate (/s) of vServer hits.	Number		5000
RequestsRate	Rate (/s) of vServer requests.	Number		5000

Field	Description	Data type	Unit	Example
ResponsesRate	Rate (/s) of vServer responses.	Number		25
CurrentClientConnections	Number of current client connections.	Number		80
CurrentServerConnections	Number of current connections to the actual servers behind the virtual server.	Number		60

* *If an SSL profile is bound, then these settings are taken from it.*

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
RequestMbpsRate	Incoming network traffic.	Number	Mbps	1	Splunk data model
ResponseMbpsRate	Outgoing network traffic.	Number	Mbps	1	Splunk data model
ActSvcs	Active services.	Number		2	Splunk data model
VSLBHealth	The health of the vserver. This gives the percentage of UP services bound to this vserver.	Number	%	100	Splunk data model
SessionTimeout	Session timeout.	Number		120	Splunk data model

Citrix Session Metrics

Virtual Channel Detail

uberAgent collects the input and output data volume for every virtual channel per Citrix session.

Details

- Source type: `uberAgent:CitrixSession:VirtualChannelDetail`
- Used in dashboards: *Citrix Session Protocol Insights*, *Single User Detail*, *Analyze data over time*
- Enabled through configuration setting: `CitrixSessionVirtualChannelDetail`
- Related configuration settings: *CitrixSessionConfig*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example	Platform
SessionGUID	Unique identifier that is generated by uberAgent when the session is created. Valid for this session only.	String		00000002-f295-9109-e7c7-c964011dd401	Windows
SessionUser	User name.	String		Domain\JohnDoe	Windows
VirtualChannelVendorName	A unique identifier for a virtual channel.	String		CTXTW	Windows

Field	Description	Data type	Unit	Example	Platform
VirtualChannelDataVolumeInputMB	Amount of data volume received by the session per virtual channel.	Number	MB	1	Windows
VirtualChannelDataVolumeOutputMB	Amount of data volume sent by the session per virtual channel.	Number	MB	10	Windows

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
VirtualChannelDataVolumeInputOutputMB	Total virtual channel data volume.	Number	MB	11	Splunk data model
VirtualChannelDisplayName	Descriptive virtual channel name. Possible values: too many to list here. Please check the lookup file `citrixsession_virtualchannels.csv`.	String		Thinwire Graphics	SPL

Field	Description	Data type	Unit	Example	Where available
VirtualChannel	Content of `VirtualChannelDisplayName`. If `VirtualChannelDisplayName` is not filled, `VirtualChannelVendorName` is used instead.	String		Thinwire Graphics	Splunk data model

Session Config

uberAgent collects detailed information about applied Citrix HDX policies and important active session settings.

Details

- Source type: `uberAgent:CitrixSession:SessionConfig`
- Used in dashboards: *Citrix Session Configuration Details*, *Single User Detail*
- Enabled through configuration setting: `CitrixSessionConfig`
- Related configuration settings: *CitrixSessionVirtualChannelDetail, TriggerCitrixSessionConfig*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example	Platform
SessionGUID	Unique identifier that is generated by uberAgent when the session is created. Valid for this session only.	String		00000002-f295-9109-e7c7-c964011dd401	Windows
SessionUser	User name.	String		Domain\JohnDoe	Windows

Field	Description	Data type	Unit	Example	Platform
AudioActualPriority	Applied virtual channel priority for audio.	String		RealTime	Windows
AudioPolicyAllowMicrophoneRedirection	Citrix policy to allow/deny microphone redirection.	String		1	Windows
AudioPolicyAllowRedirection	Citrix policy to allow/deny audio redirection.	String		1	Windows
AudioPolicyPriority	Citrix policy to set the virtual channel priority.	String		RealTime	Windows
AudioPolicySoundQuality	Citrix policy to configure the sound quality.	String		High	Windows
CdmActualPriority	Applied virtual channel priority for client drive mapping.	String		Medium	Windows
CdmVolumes	Volumes the client mapped into the session.	String		Local Disk (C: on MACHINE1)	Windows
CdmPolicyAllowDriveRedirection	Citrix policy to allow/deny client drive mapping.	String		1	Windows
CdmPolicyPriority	Citrix policy to set the virtual channel priority.	String		Medium	Windows

Field	Description	Data type	Unit	Example	Platform
CdmPolicyReadOnly	Citrix policy to allow/deny read-only client drive access.	String		1	Windows
DisplayMode	Active display mode. See `DisplayModeName` for details.	Number		4	Windows
ThinwireActualPriority	Applied virtual channel priority for this display mode.	String		High	Windows
ThinwireColorDepth	Applied session color depth.	Number	bit	24	Windows
ThinwireComponentEncoder	Applied video component encoder.	String		DeepCompressionV2Encoder	Windows
ThinwireHardwareEncodeInUse	See if hardware encode is actually in use.	String		1	Windows
ThinwireVideoCodecType	Applied video codec type.	String		H264	Windows
ThinwireColorspace	Applied session color space.	String		Yuv420	Windows
ThinwireVideoCodecUse	Applied video codec usage setting.	String		For actively changing regions	Windows
ThinwirePolicyFps	Citrix policy to configure the maximum session FPS.	Number		30	Windows

Field	Description	Data type	Unit	Example	Platform
ThinwirePolicyPriority	Citrix policy to set the virtual channel priority.	String		High	Windows
ThinwirePolicyUseHardwareEncoding	Citrix policy to enable/disable hardware encoding.	String		1	Windows
ThinwirePolicyUseVideoCodec	Citrix policy to configure the video codec.	String		UseVideoCodecIfPreferred	Windows
ThinwirePolicyVisualQuality	Citrix policy to configure the visual quality.	String		Medium	Windows
FramehawkActualPriority	Applied virtual channel priority for this display mode.	String		High	Windows
FramehawkPolicyPriority	Citrix policy to set the virtual channel priority.	String		High	Windows
D3DActualPriority	Applied virtual channel priority for this display mode.	String		High	Windows
D3DPolicyAeroRedirection	Citrix policy to enable/disable DCR aero redirection.	String		1	Windows
D3DPolicyGraphicsQuality	Citrix policy to configure the DCR graphics quality.	String		Medium	Windows

Field	Description	Data type	Unit	Example	Platform
D3DPolicyPriority	Citrix policy to set the virtual channel priority.	String		High	Windows
GraphicsActualPriority	Applied virtual channel priority for this display mode.	String		High	Windows
GraphicsPolicyDisplayDegradeNotifyUser	Citrix policy to notify the user about display quality degradation.	String		1	Windows
GraphicsPolicyDisplayDegradePolicy	Citrix policy to configure the display quality degradation.	String		DEGRADE_RESOLUTION_FIRST	Windows
GraphicsPolicyPriority	Citrix policy to set the virtual channel priority.	String		High	Windows
NetworkConnectedVia	IP address the session connected from. This is the same information as shown in Citrix Director.	String		10.1.1.150	Windows
NetworkEdtMtu	Applied Enlightened Data Transport MTU size.	Number		1400	Windows

Field	Description	Data type	Unit	Example	Platform
NetworkPolicyAcceptSessionReliabilityConnections	Citrix policy to enable/disable session reliability.	String		1	Windows
NetworkPolicyICAListenerPortNumber	Citrix policy to configure the ICA listener port.	Number		1494	Windows
NetworkPolicySessionReliabilityPort	Citrix policy to configure the session reliability port.	Number		2598	Windows
NetworkPolicySessionReliabilityTimeout	Citrix policy to configure the session reliability timeout.	Number	s	180	Windows
PrinterActualPriority	Applied virtual channel priority for printing.	String		Low	Windows
PrinterSessionPrinter	All mapped session printers.	String		Microsoft Print to PDF (from MACHINE1) in session 9	Windows
PrinterPolicyAllowRedirection	Citrix policy to allow/deny printer redirection.	String		1	Windows
PrinterPolicyAllowRedirection	Citrix policy to allow/deny printer redirection.	String		Allowed	Windows

Field	Description	Data type	Unit	Example	Platform
PrinterPolicyAutoCreateClientPrinters	Citrix policy to configure printer auto-creation. See `PrinterPolicyAutoCreateClientPrintersDisplayName` for details.	Number		3	Windows
PrinterPolicyPriority	Citrix policy to set the virtual channel priority.	String		Low	Windows
USBActualPriority	Applied virtual channel priority for USB devices.	String		Medium	Windows
USBPolicyAllowPNPRedirection	Citrix policy to allow/deny PNP redirection.	String		1	Windows
USBPolicyAllowUSBSupport	Citrix policy to allow/deny USB support.	String		1	Windows
USBPolicyPriority	Citrix policy to set the virtual channel priority.	String		Medium	Windows

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
DisplayModeName	Descriptive display mode name. Possible values: `Thinwire`, `Framehawk`, `DCR`, `Legacy Graphics`.	String		Thinwire	SPL

PrinterPolicyAutoCreateClientPrintersDisplayName	Descriptive name of how client printer creation is configured. Possible values: `Do not auto-create client printers`, `Auto-create the client's default printer only`, `Auto-create local (non-network) client printers only`, `Auto-create all client printers`.	String	Auto-create the client’s default printer only	SPL

Citrix Virtual Apps & Desktops Site Metrics

Citrix Virtual Apps and Desktops site metrics can be determined from on-premises machines or from the Citrix Cloud.
Citrix site monitoring requires its metrics enabled on at least one machine. See the documentation for details.

Applications

uberAgent collects metrics like application name, site name as well as several application properties (is enabled, creation date, modification date, type) for each published application.

Details

- Source type: `uberAgent:Citrix:Applications`
- Used in dashboards: *Citrix Virtual Apps and Desktops*
- Enabled through configuration setting: `CitrixDCApplication`
- Related configuration settings: *n/a*
- Supported platform: *Windows*, *Citrix Virtual Apps and Desktops on-premises*, *Citrix Cloud*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
Id	Application ID.	String		10
Name	Application name.	String		GoogleChrome
PublishedName	Application published name.	String		Google Chrome
SiteName	Related Citrix site name.	String		vl-Site
SiteGuid	Related Citrix site GUID.	String		0a0fda60-aebf-4ac0-920d-44bd509ca927
DesktopGroupId	Desktop group ID.	String		05bddbe7-cf12-4ee2-ab02-93d0428d3847
DesktopGroupName	Desktop group name.	String		Finance users
ApplicationType	Application type. Possible values: `0`, `1`.	Number		0
Enabled	Indicates if application is enabled. Possible values: `0`, `1`.	String		1

Field	Description	Data type	Unit	Example
AdminFolder	Application admin folder.	String		Finance
Tags	Configured tags.	String		Germany France
ApplicationGroupId	Application group ID.	String		f5411e59-bc4f-4dfa-a893-c54015aa4a66
ApplicationGroupName	Application group name.	String		Finance apps
CustomerId	Citrix Cloud CustomerId.	String		a230abcdej1e

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
ApplicationTypeDisplayName	Application type name. Possible values: `HostedOnDekstop`, `InstalledOnClient`.	String		HostedOnDekstop	Splunk data model, Splunk SPL
TagsSplit	Tags split by semicolon.	String		Germany;France	Splunk data model
SiteNameExtended	Site name including the Citrix Cloud CustomerId.	String		cloudxdsite (a230abcdej1e)	Splunk data model

Databases

uberAgent collects information like name and data store about databases configured in a site.

Details

- Source type: `uberAgent:Citrix:Databases`
- Used in dashboards: *Citrix Virtual Apps and Desktops Databases*
- Enabled through configuration setting: `CitrixDCGeneralInformation`
- Related configuration settings: *n/a*
- Supported platform: *Windows*, *Citrix Virtual Apps and Desktops on-premises*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
SiteName	Related Citrix site name.	String		vl-Site
SiteGuid	Related Citrix site GUID.	String		0a0fda60-aebf-4ac0-920d-44bd509ca927
DataStore	Datastore type. Possible values: 1, 2, 3 (1 = Site, 2 = Logging, 3 = Monitor).	Number		1
IntegratedSecurity	Database authentication method. Possible values: 1 (1 = **`true`**).	String		1
MirrorServerAddress	Mirror database server address.	String		DB2.domain.ad
Name	Database name.	String		CitrixSiteDB
ServerAddress	Database server address.	String		DB1.domain.ad

Desktops

uberAgent collects metrics like desktop name, site name as well as tags for each published desktop.

Details

- Source type: `uberAgent:Citrix:PublishedDesktops`
- Used in dashboards: *Citrix Virtual Apps and Desktops*
- Enabled through configuration setting: `CitrixDCPublishedDesktops`
- Related configuration settings: *n/a*
- Supported platform: *Windows*, *Citrix Virtual Apps and Desktops on-premises*, *Citrix Cloud*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
Id	Published desktop ID.	String		10
Name	Published desktop name (internal).	String		Finance Desktop
PublishedName	Published desktop name (published).	String		Finance Desktop
SiteName	Related Citrix site name.	String		vl-Site
SiteGuid	Related Citrix site GUID.	String		0a0fda60-aebf-4ac0-920d-44bd509ca927
DesktopGroupId	Related desktop group ID.	String		05bddbe7-cf12-4ee2-ab02-93d0428d3847
DesktopGroupName	Related desktop group name.	String		Finance users
BrowserName	Unique published desktop name in a site.	String		FinanceDesktop
ColorDepth	Color depth. Possible values: `0`, `1`, `2`, `3`.	Number		3
Description	Description.	String		Desktop for finance users

Field	Description	Data type	Unit	Example
Enabled	Indicates if the published desktop is enabled. Possible values: `0`, `1`.	String		1
ExcludedUserFilterEnabled	Indicates if the excluded user filter is enabled. Possible values: `0`, `1`.	String		1
ExcludedUsers	Excluded users.	String		AD\Marketing AD\HR
IncludedUserFilterEnabled	Indicated if the included user filter is enabled. Possible values: `0`, `1`.	String		1
IncludedUsers	Included users.	String		AD\Finance
LeasingBehavior	Leasing behavior. Possible values: `0`, `1`.	Number		1
RestrictToTag	Indicates if a published desktop is restricted to machines with specific tags.	String		Finance
SecureIcaRequired	Indicates if SecureICA is required. Possible values: `0`, `1`.	String		1

Field	Description	Data type	Unit	Example
SessionReconnection	Indicates the session reconnection behaviour. Possible values: `0`, `1`, `2`.	Number		2
Tags	Configured tags.	String		Germany France
CustomerId	Citrix Cloud CustomerId.	String		a230abcdej1e

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
ColorDepthBits	Color depth. Possible values: `4`, `8`, `16`, `24`.	Number		24	Splunk data model, Splunk SPL
ColorDepthDisplayName	Color depth. Possible values: `FourBit`, `EightBit`, `SixteenBit`, `TwentyFourBit`.	String		TwentyFourBit	Splunk data model, Splunk SPL
ExcludedUsersSplit	Excluded user split by semicolon.	String		AD\Marketing;AD\HR	Splunk data model
IncludedUsersSplit	Included users split by semicolon.	String		AD\Finance	Splunk data model

Field	Description	Data type	Unit	Example	Where available
LeasingBehaviorDisplayName	Leasing behavior. Possible values: `Allowed`, `Disallowed`.	String		Allowed	Splunk data model, Splunk SPL
SessionReconnectionDisplayName	Indicates the session reconnection behaviour. Possible values: `Always`, `DisconnectedOnly`, `SameEndpointOnly`.	String		SameEndpointOnly	Splunk data model, Splunk SPL
TagsSplit	Tags split by semicolon.	String		Germany;France	Splunk data model
SiteNameExtended	Site name including the Citrix Cloud CustomerId.	String		cloudxdsite (a230abcdej1e)	Splunk data model

Desktop Groups

uberAgent collects metrics like desktop group name and tags for each desktop group in a site.

Details

- Source type: `uberAgent:Citrix:DesktopGroups`
- Used in dashboards: *Citrix Virtual Apps and Desktops*, *Citrix Virtual Apps and Desktops Machines*
- Enabled through configuration setting: `CitrixDCDesktopGroup`
- Related configuration settings: *n/a*
- Supported platform: *Windows*, *Citrix Virtual Apps and Desktops on-premises*, *Citrix Cloud*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
Id	Desktop group ID.	String		10
Name	Desktop group name.	String		Finance
SiteName	Related Citrix site name.	String		vl-Site
SiteGuid	Related Citrix site GUID.	String		0a0fda60-aebf-4ac0-920d-44bd509ca927
IsRemotePC	Indicates if a desktop group is configured for Remote PC. Possible values: `0`, `1`.	String		0
DesktopKind	Kind of contained desktops. Possible values: `0`, `1`.	Number		1
LifecycleState	Lifecycle state. Possible values: `0`, `1`, `2`, `3`.	Number		1
SessionSupport	Indicates a desktop group's session support. Possible values: `0`, `1`, `2`.	Number		1
DeliveryType	Desktop group delivery type. Possible values: `0`, `1`, `2`.	Number		2
Tags	Configured tags.	String		Germany France
CreatedDate	Desktop group creation date.	String		2018-03-16 14:16:07.008 +0200

Field	Description	Data type	Unit	Example
ModifiedDate	Desktop group modification date.	String		2018-03-16 14:16:07.008 +0200
CustomerId	Citrix Cloud CustomerId.	String		a230abcdej1e

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
DesktopGroupId	Content of field `Id`.	String		05bddbe7-cf12-4ee2-ab02-93d0428d3847	Splunk data model, Splunk SPL
DesktopKindDisplayName	Kind of contained desktops. Possible values: `Private`, `Shared`.	String		Shared	Splunk data model, Splunk SPL
LifecycleStateDisplayName	Lifecycle state name. Possible values: `Active`, `Deleted`, `RequiresResolution`, `Stub`.	String		Active	Splunk data model, Splunk SPL

Field	Description	Data type	Unit	Example	Where available
SessionSupportDisplayName	Includes desktop group's session support. Possible values: `Unknown`, `SingleSession`, `MultiSession`.	String		SingleSession	Splunk data model, Splunk SPL
DeliveryTypeDisplayName	Desktop group delivery type. Possible values: `DesktopsOnly`, `AppsOnly`, `DesktopAndApps`.	String		DesktopAndApps	Splunk data model, Splunk SPL
TagsSplit	Tags split by semicolon.	String		Germany;France	Splunk data model
SiteNameExtended	Site name including the Citrix Cloud CustomerId.	String		cloudxdsite (a230abcdej1e)	Splunk data model

Hypervisors

DEPRECATED: this sourcetype is deprecated and will be removed in a future release.

uberAgent collects information like name and lifecycle state about hypervisors configured in a site.

Details

- Source type: `uberAgent:Citrix:Hypervisors`
- Used in dashboards: *n/a*

- Enabled through configuration setting: `CitrixDCHypervisor`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
Id	Hypervisor ID.	String		10
Name	Hypervisor name.	String		XenServer1
SiteName	Related Citrix site name.	String		vl-Site
SiteGuid	Related Citrix site GUID.	String		0a0fda60-aebf-4ac0-920d-44bd509ca927
LifecycleState	Lifecycle state. Possible values: `0`, `1`, `2`, `3`.	Number		2
CustomerId	Citrix Cloud CustomerId.	String		a230abcdej1e

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
LifecycleStateDisplayName	Lifecycle state name. Possible values: `Active`, `Deleted`, `RequiresResolution`, `Stub`.	String		Active	Splunk data model, Splunk SPL
SiteNameExtended	Site name including the Citrix Cloud CustomerId.	String		cloudxdsite (a230abcdej1e)	Splunk data model

Licenses

uberAgent collects metrics like site name and ID as well as license server for each active license in a site. It also collects Ctirix DaaS license usage.

Details

- Source type: `uberAgent:Citrix:Licenses`
- Used in dashboards: *Citrix Virtual Apps and Desktops Licensing*
- Enabled through configuration setting: `CitrixDCLicenseInformation`
- Related configuration settings: *n/a*
- Supported platform: *Windows*, *Citrix Virtual Apps and Desktops on-premises*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
SiteName	Related Citrix site name.	String		vl-Site
SiteGuid	Related Citrix site GUID.	String		0a0fda60-aebf-4ac0-920d-44bd509ca927
LicenseServer	License server. CVAD only.	String		licsrv1
LicenseProductName	License product name. Possible values CVAD: `XDS`, `XDT`, `MPS`. Possible values DaaS: arbitrary.	String		XDT
LicenseEdition2	License edition. Possible values CVAD: `PLT`, `ENT`, `APP`, `STD`, `ADV`. Possible values DaaS: arbitrary.	String		PLT
LicenseExpirationDate	License expiration date. CVAD & DaaS.	String		2019-04-16 02:00:00.000 +0200

Field	Description	Data type	Unit	Example
LicenseSubscriptionAdvantageDate	License subscription advantage date. CVAD only.	String		2019-03-16 02:00:00.000 +0200
LicenseType	License type. CVAD only.	String		NFR
LicenseTypeLocalized	License type localized. CVAD only.	String		Not For Resale
LicensesInUse	Indicates how many licenses are in use. CVAD only.	Number		5
LicensesAvailable	Indicates how many licenses are available. CVAD & DaaS.	Number		100
LicenseOverdraft	Indicates how many overdraft licenses were in use. CVAD only.	Number		0
LicenseModel	License model. Possible values: CCS, UD, CCU. CVAD only.	String		UD
CustomerId	CustomerId. DaaS only.	String		a230abcdej1e
LicenseUserUsage	User license usage. DaaS only.	Number		3
LicenseDeviceUsage	Device license usage. DaaS only.	Number		7

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
LicenseProductDisplayName	License product name. Possible values CVAD: `XenDesktop (Legacy)`, `XenDesktop`, `XenApp`. Possible values DaaS: same as `LicenseProductName`.	String		XenDesktop	Splunk data model, Splunk SPL
LicenseEditionDisplayName	License edition. Possible values CVAD: `Platinum Edition`, `Enterprise Edition`, `App Edition`, `VDI Edition`, `Advanced Edition`. Possible values DaaS: same as `LicenseEdition2`.	String		Platinum Edition	Splunk data model, Splunk SPL

Field	Description	Data type	Unit	Example	Where available
LicenseModelDisplayName	License model. Possible values: `Concurrent`, `User`/`Device`, `Concurrent`.	String		User/Device	Splunk data model, Splunk SPL

Machines

uberAgent collects metrics like machine name and ID as well as the load index for each machine in a site. The terms ”desktop group” and ”delivery group” are used interchangeably.

Details

- Source type: `uberAgent:Citrix:Machines`
- Used in dashboards: *Citrix Virtual Apps and Desktops Machines*, *Citrix Virtual Apps and Desktops*
- Enabled through configuration setting: `CitrixDCMachine`
- Related configuration settings: *n/a*
- Supported platform: *Windows*, *Citrix Virtual Apps and Desktops on-premises*, *Citrix Cloud*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
Id	Machine ID.	String		10
Sid	Machine SID.	String		S-1-5-21-3286265390-2319899195-874027928-2628

Field	Description	Data type	Unit	Example
Name	Machine name with Active Directory domain.	String		Domain\VM1
NameHost	Machine name without Active Directory domain.	String		VM1
SiteName	Related Citrix site name.	String		vl-Site
SiteGuid	Related Citrix site GUID.	String		0a0fda60-aebf-4ac0-920d-44bd509ca927
EffectiveLoadIndex	Effiective load index.	Number		10000
DnsName	Machine name with full qualified domain name.	String		VM1.domain.ad
LifecycleState	Lifecycle state. Possible values: `0`, `1`, `2`, `3`.	Number		1
IPAddress	IP address.	String		10.1.1.100
HostedMachineId	Hypervisor identifier for this Machine.	String		F8143B4F-7371-4efa-868A-54787EF9F64E
HostingServerName	DNS name of the hypervisor that is hosting the machine if managed.	String		XenServer1.domain.ad
HostedMachineName	The friendly name of a hosted machine as used by its hypervisor. This is not necessarily the DNS name of the machine.	String		XenServer1

Field	Description	Data type	Unit	Example
IsAssigned	Indicates if machine is assigned. Possible values: `0`, `1`.	String		1
IsInMaintenanceMode	Indicates if machine is in maintenance mode. Possible values: `0`, `1`.	String		0
IsPendingUpdate	Indicates if machine is in pending update state. Possible values: `0`, `1`.	String		0
AgentVersion	Citrix VDA version.	String		7.15.1000.150
AssociatedUserFullNames	Full names of the users that have been associated with the machine. Associated users are the current user(s) for shared machines and the assigned users for private machines.	String		Testuser 1
AssociatedUserNames	Usernames of the users that have been associated with the machine. Associated users are the current user(s) for shared machines and the assigned users for private machines.	String		AD\Testuser1

Field	Description	Data type	Unit	Example
AssociatedUserUPNs	The User Principal Names of the users associated with the machine. Associated users are the current user(s) for shared machines and the assigned users for private machines.	String		Testuser1@domain.ad
CurrentRegistrationState	Current registration state. Possible values: `0`, `1`, `2`.	Number		1
RegistrationStateChangeDate	When the registration state changed.	String		2018-07-29 03:03:43.953 +0200
LastDeregisteredCode	Machine's last deregistration code. Possible values: too many to list here. Please have a look at the `citrix_deregistrationreasoncode` lookup.	Number		207
LastDeregisteredDate	Machine's last deregistration date.	String		2018-07-29 02:33:34.037 +0200
CurrentPowerState	Current power state. Possible values: `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `10`.	Number		9
CurrentSessionCount	Current session count.	Number		20

Field	Description	Data type	Unit	Example
ControllerDnsName	Controller name.	String		DDC1.domain.ad
PoweredOnDate	Date when machine was powered on.	String		2018-03-16 15:18:27.407 +0200
PowerStateChangeDate	Date when power state changed.	String		2018-03-16 15:18:27.407 +0200
FunctionalLevel	Functional level. Possible values: `0`, `1`, `2`, `3`, `4`, `5`, `6`.	Number		6
FailureDate	Date when machine failed.	String		2018-03-16 15:18:27.407 +0200
WindowsConnectionSetting	Windows connection setting. Possible values: `0`, `1`, `2`, `3`, `4`.	Number		1
IsPreparing	Indicates if the machine is in preparing state. Possible values: `0`, `1`.	String		0
FaultState	Fault state. Possible values: `0`, `1`, `2`, `3`, `4`, `5`.	Number		1
CatalogId	Related catalog ID.	String		10
DesktopGroupId	Related desktop group ID.	String		05bddbe7-cf12-4ee2-ab02-93d0428d3847
HypervisorId	Related hypervisor ID.	String		10
Hash	Hash.	String		CC7380E01A5FD3403708B7D34
MachineRole	Machine role. Possible values: `0`, `1`, `2`.	Number		2

Field	Description	Data type	Unit	Example
HypervisorDisplayName	Related hypervisor name.	String		XenServer1
CatalogDisplayName	Related catalog name.	String		Finance machines
DesktopGroupDisplayName	Related desktop group name.	String		Finance
CreatedDate	Machine creation date.	String		2018-03-16 14:16:07.008 +0200
ModifiedDate	Machine modification date.	String		2018-03-16 14:16:07.008 +0200.
Tags	Configured tags.	String		Germany France
CustomerId	Citrix Cloud CustomerId.	String		a230abcdej1e

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
LifecycleStateDisplayName	Lifecycle state name. Possible values: `Active`, `Deleted`, `RequiresResolution`, `Stub`.	String		Active	Splunk data model, Splunk SPL
IsVDA	Indicates if machine is a VDA. Possible values: `0`, `1`. Is 1 if field `MachineRoleDisplayName` is either `Vda` or `Both`.	String		1	Splunk data model

Field	Description	Data type	Unit	Example	Where available
IsVDANumber	Indicates if machine is a VDA. Possible values: `0`, `1`. Is 1 if field `MachineRoleDisplayName` is either `Vda` or `Both`.	Number		1	Splunk data model
IsDDC	Indicates if machine is a DDC. Possible values: `0`, `1`. Is 1 if field `MachineRoleDisplayName` is either `Ddc` or `Both`.	String		1	Splunk data model
IsRegistered	Indicates if machine registered. Possible values: `0`, `1`. Is 1 if field `CurrentRegistrationState` is 1.	String		1	Splunk data model
IsRegisteredNumber	Indicates if machine registered. Possible values: `0`, `1`. Is 1 if field `CurrentRegistrationState` is 1.	Number		1	Splunk data model

Field	Description	Data type	Unit	Example	Where available
CurrentRegistrationStateDisplayName	Current registration state. Possible values: `Unknown`, `Registered`, `Unregistered`.	String		Registered	Splunk data model, Splunk SPL
LastDeregisteredCodeDisplayName	Machine's last deregistration code. Possible values: too many to list here. Please have a look at the `citrix_deregistrationreasoncode` lookup.	String		ContactLost	Splunk data model, Splunk SPL
CurrentPowerStateDisplayName	Current power state. Possible values: `Unknown`, `Unavailable`, `Off`, `On`, `Suspended`, `TurningOn`, `TurningOff`, `Suspending`, `Resuming`, `Unmanaged`, `NotSupported`.	String		Unmanaged	Splunk data model, Splunk SPL

Field	Description	Data type	Unit	Example	Where available
FunctionalLevelDisplayName	Functional level. Possible values: `Unknown`, `L5`, `L7`, `L7_6`, `L7_7`, `L7_8`, `L7_9 / MAX`.	String		L7_9 / MAX	Splunk data model, Splunk SPL
WindowsConnectionSettingDisplayName	Windows connection setting. Possible values: `Unknown`, `LogonEnabled`, `Draining`, `DrainnigUntilRestart`, `LogonDisabled`.	String		Unknown	Splunk data model, Splunk SPL
FaultStateDisplayName	Fault state. Possible values: `Unknown`, `None`, `FailedToStart`, `StuckOnBoot`, `Unregistered`, `MaxCapacity`.	String		None	Splunk data model, Splunk SPL

Field	Description	Data type	Unit	Example	Where available
MachineRoleDisplayName	Machine role. Possible values: `Vda`, `Ddc`, `Both`.	String		Both	Splunk data model, Splunk SPL
TagsSplit	Tags split by semicolon.	String		Germany;France	Splunk data model
SiteNameExtended	Site name including the Citrix Cloud CustomerId.	String		cloudxdsite (a230abcdej1e)	Splunk data model

Machine Catalogs

uberAgent collects metrics like catalog name and ID as well as site name for each machine catalog in a site.

Details

- Source type: `uberAgent:Citrix:Catalogs`
- Used in dashboards: *Citrix Virtual Apps and Desktops Machines*
- Enabled through configuration setting: `CitrixDCCatalog`
- Related configuration settings: *n/a*
- Supported platform: *Windows*, *Citrix Virtual Apps and Desktops on-premises*, *Citrix Cloud*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
Id	Machine catalog ID.	String		10
Name	Machine catalog name.	String		Finance machines
SiteName	Related Citrix site name.	String		vl-Site

Field	Description	Data type	Unit	Example
SiteGuid	Related Citrix site GUID.	String		0a0fda60-aebf-4ac0-920d-44bd509ca927
LifecycleState	Lifecycle state. Possible values: `0`, `1`, `2`, `3`.	Number		0
ProvisioningType	Provisioning type. Possible values: `0`, `1`, `2`, `3`.	Number		1
PersistentUserChanges	Indicates if and where user changes should be saved. Possible values: `0`, `1`, `2`, `3`.	Number		1
IsMachinePhysical	Indicates if machine catalog contains physical machines. Possible values: `0`, `1`.	String		0
AllocationType	Allocation type. Possible values: `0`, `1`, `2`, `3`.	Number		2
SessionSupport	Indicates a machine catalog's session support. Possible values: `0`, `1`, `2`.	Number		1
ProvisioningSchemeId	Associated provisioning scheme Id.	String		0a0fda60-aebf-4ac0-920d-44bd509ca927
CreatedDate	Machine catalog creation date.	String		2018-03-16 14:16:07.008 +0200
ModifiedDate	Machine catalog modification date.	String		2018-03-16 14:16:07.008 +0200

Field	Description	Data type	Unit	Example
CustomerId	Citrix Cloud CustomerId.	String		a230abcdej1e

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
CatalogId	Content of field `Id`.	String		10	Splunk data model
LifecycleStateDisplayName	Lifecycle state name. Possible values: `Active`, `Deleted`, `RequiresResolution`, `Stub`.	String		Active	Splunk data model, Splunk SPL
ProvisioningTypeDisplayName	Provisioning type. Possible values: `Unknown`, `MCS`, `PVS`, `Manual`.	String		MCS	Splunk data model, Splunk SPL
PersistentUserChangesDisplayName	Indicates if and where user changes should be saved. Possible values: `Unknown`, `Discard`, `OnLocal`, `OnPvd`.	String		Discard	Splunk data model, Splunk SPL

Field	Description	Data type	Unit	Example	Where available
AllocationTypeDisplayName	Allocation type. Possible values: `Unknown`, `Static`, `Random`, `Permanent`.	String		Random	Splunk data model, Splunk SPL
SessionSupportDisplayName	The machine catalog's session support. Possible values: `Unknown`, `SingleSession`, `MultiSession`.	String		SingleSession	Splunk data model, Splunk SPL
SiteNameExtended	Site name including the Citrix Cloud CustomerId.	String		cloudxdsite (a230abcdej1e)	Splunk data model

Machine Metrics

Machine Crashes & Hangs Metrics

Windows Blue Screens and Hangs

uberAgent collects information on every blue screen and hang, like the type of error (blue screen, hard power off or hang) and the stop error code.

Details

- Source type: `uberAgent:System:Bugcheck`
- Used in dashboards: *Stop Errors (Blue Screen & Power Loss)*
- Enabled through configuration setting: `ApplicationErrors`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
BugcheckCode	Stop error code ID. Possible values: too many to list here. Please check the lookup file `bugcheck_codes.csv` or Microsoft's bug check code reference.	Number		0x1
BugcheckParameter1	Stop error parameter 1. The meaning of this value depends on the bugcheck code and can be looked up in Microsoft's bug check code reference.	String		0x7ffd8e8c7864

Field	Description	Data type	Unit	Example
BugcheckParameter2	Stop error parameter 2. The meaning of this value depends on the bugcheck code and can be looked up in Microsoft's bug check code reference.	String		0x1
BugcheckParameter3	Stop error parameter 3. The meaning of this value depends on the bugcheck code and can be looked up in Microsoft's bug check code reference.	String		0x0
BugcheckParameter4	Stop error parameter 4. The meaning of this value depends on the bugcheck code and can be looked up in Microsoft's bug check code reference.	String		0xffffe181ead22b80
SleepInProgress	Indicates if the machine was in sleep mode when stop error occurred. Possible values: `0`, `1`.	Number		0

Field	Description	Data type	Unit	Example
PowerButtonTimestamp	Indicates if the power button on the computer was pushed and held for at least four seconds. Possible values: `0` or Windows `FILETIME` timestamp of when the power button was pressed.	Number		131768171003182508
PowerButtonTimestampEpoch	Indicates if the power button on the computer was pushed and held for at least four seconds. Possible values: `0` or Unix epoch timestamp of when the power button was pressed.	Number		1532343500318
BootAppStatus	n/a	String		0
Checkpoint	n/a	Number		0
ConnectedStandbyInProgress	Indicates if the machine was in connected standby mode when a stop error occurred. Possible values: `0`, `1`.	String		0

Field	Description	Data type	Unit	Example
SystemSleepTransitionsToOn	Indicates if the machine was in the transition from sleep to on mode when a stop error occurred. Possible values: `0`, `1`.	Number		0
CsEntryScenarioInstanceId	n/a	Number		0

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
BugcheckCodeDisplayName	Stop error code name. Possible values: too many to list here. Please check the lookup file `bugcheck_codes.csv` or Microsoft's bug check code reference.	String		Hard power off	Splunk data model

Interpreting the Data

The data collected by uberAgent helps to identify three different types of blue screens and hangs:

"Normal" Bugcheck Conditions:

- `BugcheckCode` > 0

Explanation: the bugcheck code can be determined and written to disk before the computer shuts down or restarts.

Hard Power Off Conditions:

- `PowerButtonTimestamp` > 0

Explanation: the machine was turned off by pressing and holding the power button for at least 4 seconds.

Random Restart Conditions:

- `BugcheckCode` = 0
- `PowerButtonTimestamp` = 0

Explanation: power loss or hard hang.

macOS Kernel Panics

uberAgent collects information on every macOS kernel panic and reports details like bug type, kernel version and more.

Details

- Source type: `uberAgent:System:MacOsErrors`
- Enabled through configuration setting: `ApplicationErrors`
- Related configuration settings: *n/a*
- Supported platform: *macOS*

List of Fields in the Raw Agent Data

Field	Description	Data type	Example
KernelBuild	The macOS build version that the system was running when the panic occurred.	String	macOS 14.1 (23B74)

Field	Description	Data type	Example
KernelProduct	The model identifier for the machine.	String	Mac13,1
KernelVersion	The version of the Darwin kernel that was running.	String	Darwin Kernel Version 23.1.0: Mon Oct 9 21:27:24 PDT 2023; root:xnu-10002.41.9~6/RELEASE_ARM64_T60
KernelIncident	A unique identifier for the specific panic incident.	String	0039B7C3-9D24-4DF4-B08F-432475FCA067
KernelCrashReporterKey	A unique identifier that helps Apple track the crash report without revealing personal information.	String	7E439903-F3C7-16EB-ED55-F393AE09615C
KernelPanicString	The actual panic message which contains the reason for the panic. CPU state and the call stack leading up to the panic are excluded.	String	panic(cpu 2 caller 0xfffffe001a373598): dtrace: panic action at probe dtrace:::BEGIN (ecb 0xfffffe1b2f31a140) @dtrace.c:6673
KernelPanicFlags	Flags that provide additional information about the panic, useful for debugging.	String	0x802
KernelBugType	An internal Apple classification code for the type of crash or panic. Available from macOS 14 (Sonoma) upwards.	Number	210
KernelErrorType	The type of system error that occurred. Value 1 indicates system error type `Crash`	Number	1

Machine Inventory Metrics

Machine Inventory

Contains inventory metrics for machines like operating system name and hardware manufacturer.

Details

- Source type: `uberAgent:System:MachineInventory`
- Used in dashboards: *Machine Inventory,Single Machine Inventory, Single Application Inventory, Update Inventory*
- Enabled through configuration setting: `MachineInventory`
- Related configuration settings: *n/a*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example	Platform
OsName	Operating system name	String		Windows 7 Enterprise	all
OsSpName	Operating system service pack name	String		Service Pack 1	Win
OsVersion	Operating system version	Number		6.1	all
OsBuild	Operating system build	Number		7601	Win
OsBuildString	Operating system build	String		19D76	macOS
OsUpdateBuildRevision	Operating system Update Build Revision number. More information.	Number		165	Win

Field	Description	Data type	Unit	Example	Platform
OsArchitecture	Operating system architecture	String		x64	all
OsSpVersion	Operating system service pack version	Number		1.0	Win
OsType	Operating system type. Possible values: `Client`, `Domain Controller`, `Server`, `Terminal Server`	String		Client	all
OsInstallDate	Operating system installation date.	String		2018-04-13	all
HwManufacturer	Hardware manufacturer	String		LENOVO	all
HwModel	Hardware model	String		ThinPad T470s	all
HwBiosVersion	Hardware bios version	String		N1WET46W (1.25)	Win
AdDomainDns	Active Directory DNS domain name	String		vastlimits.com	Win
AdDomainNetBios	Active Directory NetBios name	String		VASTLIMITS	Win
AdSite	Active Directory site	String		Default-First-Site-Name	Win

Field	Description	Data type	Unit	Example	Platform
AdOu	Active Directory organizational unit	String		vast limits/Computers/Clients	Win
ComputerNameDn	Computer name Dn formatted	String		CN=Client1,OU=Clients,OU=Computers,OU= limits,DC=vastlimits,DC=com	Win
ComputerNameCanonical	Computer name canonical formatted	String		vastlimits.com/Computers/Clients/Client1	Win
CtxFarmName	Citrix farm name	String		vast limits Farm	Win
CtxMachineCatalogName	Citrix machine catalog name	String		Machine Catalog 1	Win
CtxDeliveryGroupName	Citrix delivery group name	String		Delivery Group 1	Win
RAMSizeGB	RAM size	Number	GB	15.86	all
PowerSupportsConnectedStandby	Indicates if the operating system supports Connected Standby. Possible values: `0`, `1`	String		0	Win
PowerSupportsS1	Indicates if the operating system supports sleep state S1. Possible values: `0`, `1`	String		0	Win

Field	Description	Data type	Unit	Example	Platform
PowerSupportsS2	Indicates if the operating system supports sleep state S2. Possible values: `0`, `1`	String		0	Win
PowerSupportsS3	Indicates if the operating system supports sleep state S3. Possible values: `0`, `1`	String		0	Win
PowerSupportsS4	Indicates if the operating system supports sleep state S4. Possible values: `0`, `1`	String		1	Win
PowerSupportsS5	Indicates if the operating system supports sleep state S5. Possible values: `0`, `1`	String		1	Win
IsUpsPresent	Indicates if there is an uninterruptible power supply (UPS) present. Possible values: `0`, `1`	String		0	all

Field	Description	Data type	Unit	Example	Platform
IsBatteryPresent	Indicates if there is a battery present. Possible values: `0`, `1`	String		1	all
BatteryWearLevelPercent	Percent of battery capacity gone. The accuracy strongly depends on the built-in hardware.	Number	%	7.8	all
CPUName	CPU name	String		Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz	all
CPUSockets	CPU sockets	Number		1	Win
CPUCoresPhysical	CPU physical cores	Number		2	all
CPUCoresLogical	CPU logical cores	Number		4	all
CPUMaxMhz	CPU base frequency	Number	MHz	2904	all
HwHypervisorVendor	Detected hypervisor technology for this system.	String		Microsoft Hv	Win, Mac (Intel)
HwIsVirtualMachine	Indicates if machine is a virtual machine. Possible values: 0 ,1	String		0	all

Field	Description	Data type	Unit	Example	Platform
BIOSSerial	BIOS serial number	String		YM5S012345	Win
BaseboardSerial	Baseboard serial number	String		50123445	All
IsSipEnabled	Status of the System Integrity Protection	String		1	macOS
OsInstallDateOriginal	Date of the first clean operating system installation.	String		2018-04-13	All
CoverageEndDate	End date of device warranty coverage. Only available if a user signed in with an Apple-ID.	String		2025-01-01	macOS

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
OsBuildUbr	Operating system build and Update Build Revision combined. For operating systems lower than Windows 10 it is just the content of `OsBuild`.	Number		17134.165	Splunk data model
HwManufacturerModel	Hardware manufacturer and model combined	String		LENOVO ThinkPad T470s	Splunk data model

Disk Inventory

Contains inventory metrics for physical disks like disk name and capacity.

Details

- Source type: `uberAgent:System:DiskInventory`
- Used in dashboards: *Machine Storage*
- Enabled through configuration setting: `MachineInventory`
- Related configuration settings: *n/a*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example	Platform
Name	Disk name	String		SAMSUNG MZVLW1T0HMLH-000L7	all

Field	Description	Data type	Unit	Example	Platform
Enumerator	Disk enumerator (e.g., `IDE`, `SCSI`, `USBSTOR`)	String		SCSI	all
DiskNumber	Disk number	Number		0	all
CapacityMB	Disk capacity	Number	MB	976762	all
IsWritable	Indicates if the disk is writable. Possible values: `0`, `1`	String		1	all
IsRemovable	Indicates if the disk is removable. Possible values: `0`, `1`	String		0	all

Monitor Inventory

Contains inventory metrics for monitors like resolution and color depth.

Details

- Source type: `uberAgent:System:MonitorInventory`
- Used in dashboards: *Machine Inventory*,*Single Machine Inventory*
- Enabled through configuration setting: `MachineInventory`
- Related configuration settings: *n/a*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example	Platform
MonitorIndex	Monitor index	Number		1	all

Field	Description	Data type	Unit	Example	Platform
MonitorHRes	Monitor horizontal resolution	Number		3440	all
MonitorVRes	Monitor vertical resolution	Number		1440	all
MonitorColorDepth	Monitor color depth	Number		32	all
MonitorIsPrimary	Indicates if monitor is the primary one. Possible values: `0`, `1`	String		1	all
MonitorDisplayName	Monitor name	String		Generic PnP Monitor	all

Network Configuration Information

Contains inventory metrics for network connections like interface type and SSIDs.

Details

- Source type: `uberAgent:System:NetworkConfigInformation`
- Used in dashboards: *Machine Inventory,Single Machine Inventory, Machine Network Configuration*
- Enabled through configuration setting: `NetworkConfigInformation`
- Related configuration settings: *n/a*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example	Platform
NetworkConfigFriendlyName	Network friendly name	String		WiFi	all

Field	Description	Data type	Unit	Example	Platform
NetworkConfigDescription	Network description	String		Intel(R) Dual Band Wireless-AC 8265	all
NetworkConfigInterfaceType	Network interface type ID. Possible values: `0`, `6`, `23`, `63`, `71`, `243`	Number		71	all
NetworkConfigInterfaceTypeDisplayName	Network interface type name. Possible values: `Other`, `Ethernet`, `VPN`, `ISDN`, `WiFi`, `WWAN`	String		WiFi	all
NetworkConfigDnsSuffix	Network DNS suffix	String		fritz.box	all
NetworkConfigSsid	Network SSID	String		HomeWiFi	all
NetworkConfigIsConnectedToInternet	Indicates if network is connected to the Internet. Possible values: `0`, `1`	String		1	all
NetworkConfigIPv4	IPv4 address	IPv4		192.168.178.40	all
NetworkConfigIPv6	IPv6 address	String		2001:0db8:85a3:0000:0000:8a2e:0370:7334	all
NetworkConfigMACAddress	MAC address	String		AC-ED-5C-02-F0-30	all
NetworkConfigWiFiSignalQuality	WiFi signal quality	Number	%	100	all
NetworkConfigWiFiType	WiFi connection type	Number		7	all

Field	Description	Data type	Unit	Example	Platform
NetworkConfigWiFiAuthentication	WiFi authentication method	Number		7	all

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
Ipv4Address	Ipv4 address	IPv4		192.168.178.40	Splunk data model
NetworkConfigWiFiTypeDisplayName	WiFi type display name	String		IEEE 802.11n	Splunk data model
NetworkConfigWiFiAuthenticationDisplayName	WiFi authentication display name	String		WPA2	Splunk data model

Volume Inventory

Contains inventory metrics for volumes like capacity and filesystem.

Details

- Source type: `uberAgent:System:VolumeInventory`
- Used in dashboards: *Machine Storage*
- Enabled through configuration setting: `MachineInventory`
- Related configuration settings: *n/a*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example	Platform
Guid	Volume GUID	String		{d4eadd95-6dce-426e-b55b-521800346789}	all
DeviceName	Device name	String		DeviceHarddiskVolume3	all
Label	Volume label	String		Windows	all
FileSystem	Volume file system	String		NTFS	all
MountPoints	Volume mount points	String		C	all
DiskNumbers	Disk numbers	String		0	all
FreeMB	Volume free capacity	Number		695395	all
CapacityMB	Volume capacity	Number		975485	all
UsedSpacePercent	Volume used space	Number	%	29	all
PartitionStyle	Volume partition style	String		GPT	all
IsSystemVolume	Indicates if volume is a system volume. Possible values: `0`, `1`	String		1	Win
IsBootVolume	Indicates if volume is a boot volume. Possible values: `0`, `1`	String		1	all

Field	Description	Data type	Unit	Example	Platform
IsDirty	Indicates if volume has a dirty bit. If yes, the volume might be corrupted. Possible values: `0`, `1`	String		0	Win

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
IsBootVolumeNumber	Indicates if volume is a boot volume. Possible values: `0`, `1`	Number		1	Splunk data model
IsDirtyNumber	Indicates if volume has a dirty bit. If yes, the volume might be corrupted. Possible values: `0`, `1`	Number		0	

Machine Performance And Utilization Metrics

GPU Usage

uberAgent collects information about GPU compute and memory usage.

Details

- Source type: `uberAgent:System:GpuUsage`
- Used in dashboards: *Machine GPU*, *Analyze data over time*
- Enabled through configuration setting: `GpuUsage`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
DisplayAdapterName	GPU name.	String		Intel(R) HD Graphics 520
DisplayAdapterIndex	GPU index.	Number		1
MemorySharedMB	Shared memory usage.	Number	MB	512
MemoryDedicatedMB	Dedicated memory usage.	Number	MB	64
MemorySharedPercent	Shared memory usage.	Number	%	50
MemoryDedicatedPercent	Dedicated memory usage.	Number	%	50
MemorySharedSizeMB	Shared memory size.	Number	MB	1024
MemoryDedicatedSizeMB	Dedicated memory size.	Number	MB	128
ComputeUsagePercentAllEngines	Compute usage for all engines.	Number	%	10.00

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
DisplayAdapterIndexEval	GPU index - validated.	Number		1	Splunk data model

GPU Usage per Engine

uberAgent collects information about GPU compute usage per engine. This requires Windows 10 1709 or newer. On older versions of Windows `ComputeUsagePercentEngineN` is populated instead.

Details

- Source type: `uberAgent:System:GpuUsageEngine`
- Used in dashboards: *Machine GPU*
- Enabled through configuration setting: `GpuUsage`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
DisplayAdapterIndex	GPU index.	Number		1
GpuEngineIndex	GPU engine index.	Number		8
GpuEngineType	GPU engine type ID. Possible values: `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`.	Number		4
GpuEngineComputeUsagePercent	GPU engine compute usage.	Number	%	10.00

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
GpuEngineCombined	`"GPU"` + `DisplayAdapterIndex` + `"-E"` + `GpuEngineIndex` + `" - "` + `GpuEngineTypeDisplayName`.	String		GPU1-E8-Video processing	Splunk data model
GpuEngineTypeDisplayName	GPU engine type name. Possible values: `Other`, `3D`, `Video decode`, `Video encode`, `Video processing`, `Scene assembly`, `Copy`, `Legacy overlay`, `Crypto`.	String		Video processing	Splunk data model

SMB Client Performance

uberAgent collects metrics per network share the endpoint is connected to, such as share path and latency.

Details

- Source type: `uberAgent:System:SmbClient`
- Used in dashboards: *SMB Client Performance*, *Machine Storage*, *Machine Uptime*, *Analyze data over time*

- Enabled through configuration setting: `SMBClientSharePerformance`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
SharePath	Share path.	String		\dc1\home
IOPSRead	I/O operations read.	Number		100
IOPSWrite	I/O operations write.	Number		100
IOPSMetadata	I/O operations metadata.	Number		100
IOCountRead	Count of read I/O operations.	Number		20
IOCountWrite	Count of write I/O operations.	Number		20
IOCountMetadata	Count of metadata I/O operations.	Number		20
IOMBRead	Amount of read I/O operation data volume.	Number	MB	30
IOMBWrite	Amount of write I/O operation data volume.	Number	MB	30
IOLatencyMsRead	Latency of read I/O operations.	Number	ms	5
IOLatencyMsWrite	Latency of write I/O operations.	Number	ms	5

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
IOPSRW	`IOPSRead` + `IOPSWrite`.	Number		200	Splunk data model
IOCountRW	`IOCountRead` + `IOCountWrite`.	Number		40	Splunk data model
IODurationReadMs	`IOLatencyMsRead` x `IOCountRead`.	Number	ms	100	Splunk data model
IODurationWriteMs	`IOLatencyMsWrite` x `IOCountWrite`.	Number	ms	100	Splunk data model
IODurationRWMs	`IODurationReadMs` + `IODurationWriteMs`.	Number	ms	200	Splunk data model
IOMBRW	`IOMBRead` + `IOMBWrite`.	Number	MB	60	Splunk data model
IOMBPSRW	`IOMBRW` / `IOCountRW` x `IOPSRW`.	Number	MB	300	Splunk data model
IOLatencyMsRW	`IOLatencyMsRead` + `IOLatencyMsWrite`.	Number	ms	10	Splunk data model

System Performance Summary

Details

- Source type: `uberAgent:System:SystemPerformanceSummary2`
- Used in dashboards: *Machine Performance*, *SBC Sizing and Capacity Planning*, *Single Machine*

Detail, Analyze data over time
- Enabled through configuration setting: `SystemPerformanceSummary`
- Related configuration settings: *n/a*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Platform	Example
CPUUsagePercent	CPU usage.	Number	%	all	20.45
RAMUsagePercent	RAM usage.	Number	%	all	12
RAMUsageGB	RAM usage.	Number	GB	all	50
IOPSRead	I/O operations read.	Number		all	200
IOPSWrite	I/O operations write.	Number		all	200
IOCountRead	Count of read I/O operations.	Number		all	100
IOCountWrite	Count of writeI/O operations.	Number		all	100
IOMBRead	Amount of read I/O operation data volume.	Number	MB	Windows	150
IOMBWrite	Amount of write I/O operation data volume.	Number	MB	Windows	150
IOLatencyMsRead	Latency of read I/O operations.	Number	ms	all	300
IOLatencyMsWrite	Latency of write I/O operations.	Number	ms	all	300
IOPercentDiskTime	Disk utilization.	Number	%	Windows	20

Field	Description	Data type	Unit	Platform	Example
NetUtilizationPercent	Network utilization.	Number	%	Windows	15
KernelPagedMB	Kernel paged memory usage.	Number	MB	all	100
KernelNonPagedMB	Kernel non-paged memory usage.	Number	MB	all	300
HandleCount	On Windows: Handle count. On macOS: Number of open file descriptors.	Number		all	17
ThreadCount	Thread count.	Number		all	63
IdlenessPercent	Status of the idle timer expressed as a percentage. When this value reaches 100%, the system is considered completely idle, and the OS shows the screen saver or turns the display off (depending on the configuration).	Number	%	Windows	50

Field	Description	Data type	Unit	Platform	Example
CPURelativeFrequencyPercent	Average actual CPU frequency as a Percentage of the base frequency. More information.	Number	%	Windows	21
PagefileTotalSizeGB	Pagefile total size.	Number	GB	all	8
PagefileUsagePercent	Pagefile usage.	Number	%	all	60
PagefileIOPSRead	Pagefile I/O operations read.	Number		Windows	300
PagefileIOPSWrite	Pagefile I/O operations write.	Number		Windows	300

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
TotalRamGb	Total RAM size.	Number	ms	24	Splunk data model
IOPS	`IOPSRead` + `IOPSWrite`.	Number		400	Splunk data model
IOCount	`IOCountRead` + `IOCountWrite`.	Number		200	Splunk data model
IOMB	`IOMBRead` + `IOMBWrite`.	Number	MB	300	Splunk data model

Field	Description	Data type	Unit	Example	Where available
IOMBPS	IOMB / IOCount x IOPS.	Number		600	Splunk data model
IOLatencyMs	IOLatencyMsRead + IOLatencyMsWrite.	Number		600	Splunk data model
IODurationReadMs	IOLatencyMsRead x IOCountRead.	Number	ms	30000	Splunk data model
IODurationWriteMs	IOLatencyMsWrite x IOCountWrite.	Number	ms	30000	Splunk data model
IODurationMs	IODurationReadMs + IODurationWriteMs.	Number	ms	60000	Splunk data model
User	User name. Always empty.	Number			Splunk data model

Computer Startup, Shutdown & Hibernation Metrics

Boot Processes Metrics

Boot Processes

uberAgent collects detailed information about each process running during the boot process like the command line and latency.

Details

- Source type: `uberAgent:OnOffTransition:BootProcesses`
- Used in dashboards: *Single Boot Duration*
- Enabled through configuration setting: `BootDetail`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
ProcName	Process name.	String		svchost.exe
ProcID	Process ID.	Number		456
ProcParentID	Parent process ID.	Number		789
ProcStartTimeRelativeMs	Relative process start time.	Number	ms	100
ProcLifetimeMs	Process lifetime.	Number	ms	50
ProcCmdline	Process commandline.	String		C:\WINDOWS\system32\svchos -k DcomLaunch -p
ProcIOReadCount	Count of process I/O read operations.	Number		91
ProcIOWriteCount	Count of process I/O write operations.	Number		97
ProcIOReadMB	Amount of process I/O read operations data volume.	Number	MB	50
ProcIOWriteMB	Amount of process I/O write operations data volume.	Number	MB	50
ProcIOLatencyReadMs	Process I/O read operations latency.	Number	ms	30

Field	Description	Data type	Unit	Example
ProcIOLatencyWriteMs	Process I/O write operations latency.	Number	ms	30
SessionID	Unique identifier that is generated by the machine when the session is created. Will be reassigned to other sessions after logoff.	Number		3
TotalBootDurationMs	Total boot duration.	Number	ms	500000
SortOrder2	Sort order number to sort the table *Boot process performance* on the *Single Boot* dashboard correctly.	Number		29
BootUID	Unique UID for a boot generated by uberAgent.	String		00000408-0049-004b-0bb9-8ed17e25d401

List of Calculated Fields

Field	Description	Data type	Unit	Where available	Example
SortOrder	Sort order number to sort the table *Boot process performance* on the *Single Boot* dashboard correctly.	Number		Splunk data model	1

Boot Process Detail

uberAgent collects detailed information about each process running during the boot process like the disk IO and latency.

Details

- Source type: `uberAgent:OnOffTransition:BootProcessDetail`
- Used in dashboards: *Single Boot Duration*
- Enabled through configuration setting: `BootDetail`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
ProcessName	Process name.	String		svchost.exe
ProcessIOReadCount	Count of process I/O read operations.	Number		91
ProcessIOWriteCount	Count of process I/O write operations.	Number		97

Field	Description	Data type	Unit	Example
ProcessIOReadMB	Amount of process I/O read operations data volume.	Number	MB	50
ProcessIOWriteMB	Amount of process I/O write operations data volume.	Number	MB	50
ProcessIOLatencyMs	Process I/O operations latency.	Number	ms	60
BootUID	Unique UID for a boot generated by uberAgent.	String		00000408-0049-004b-0bb9-8ed17e25d401

List of Calculated Fields

Field	Description	Data type	Unit	Where available	Example
ProcessIOCount	ProcessIOReadCount+ProcessIOWriteCount.	Number		Splunk data model	188
ProcName	Process name.	String		Splunk data model	chrome.exe

Computer Startup (Machine Boot) Metrics

Boot Detail

uberAgent collects metrics about computer startup duration including Smss initialization as well as autostart services. The metrics in this section are based on the Microsoft documents Windows On/Off Transitions Solutions Guide and Post On/Off Duration.

Details

- Source type: `uberAgent:OnOffTransition:BootDetail2`

- Used in dashboards: *Boot Duration*, *Single Boot Duration*
- Enabled through configuration setting: `BootDetail`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
KernelInitTimeMs	Time from the beginning to the start of the first smss.exe process.	Number	ms	1683
SmssInitTimeMs	Time from the start of the first smss.exe process to the start of the second smss.exe process (in session 0).	Number	ms	4007
AutoCheckTimeMs	Auto check time. More information.	Number	ms	20
Session0InitDurationMs	Session 0 initialization duration: from the start of the second smss.exe process to the start of wininit.exe.	Number	ms	9572

Field	Description	Data type	Unit	Example
Session1InitDurationMs	Session 1 initialization duration: from the start of the third smss.exe process (in session 1) to the start of winlogon.exe.	Number	ms	71
WininitInitDurationMs	Wininit initialization duration: from the start of wininit.exe to the start of services.exe.	Number	ms	61
WinlogonInitDurationMs	Winlogon initialization duration: from the start of winlogon.exe to the start of logonui.exe.	Number	ms	369
AutostartServicesMs	Autostart services: from the start of services.exe to the start of the first taskhost.exe.	Number	ms	13269
ComputerStartupMs	Computer startup: from the beginning to the start of LogonUI.exe. The user is now able to log in to the computer.	Number	ms	15700

Field	Description	Data type	Unit	Example
MainPathBootTimeMs*	Main path boot time: from the beginning to the start of the shell (typically explorer.exe). The user has successfully logged in, and the shell is starting.	Number	ms	28361

Field	Description	Data type	Unit	Example
PostBootTimeMs*	Post boot time: from shell start to the system reaching an 80% idle state. This time is accumulated by checking the CPU and storage utilization in 500ms time windows. If the cumulative time of both CPU and storage utilization is below 20%, the idle time of this window (500ms – max [CPU time, Disk time] in the window) is added to the total idle time until 5 seconds is reached. The metric reports this duration minus the 5 seconds of collected idle time. The desktop is now ready to use.	Number	ms	6882

Field	Description	Data type	Unit	Example
TotalBootTimeMs	The time from Windows start to the desktop being shown and the system reaching an 80% idle state. This is the sum of `MainPathBootTimeMs` and `PostBootTimeMs`. If `TotalBootTimeMs` is empty, it means that Windows stopped collecting data too early, or because the user waited longer before logging in.	Number	ms	130765
BootUID	Unique UID for a boot created by uberAgent.	String		00000408-0049-004b-0bb9-8ed17e25d401
UserLogonWaitDurationMs	The measurement starts with the welcome display screen, thus as soon as the user can perform a login. It stops with the successful login.	Number	ms	345696

> **Note**
>
> - This field is only available on client operating systems.

Computer Shutdown Metrics

Computer Shutdown

uberAgent collects the duration of the major phases as well as the total duration for each shutdown.

> **Note**
>
> This metric is only available on client operating systems.

Details

- Source type: `uberAgent:OnOffTransition:ShutdownDetail`
- Used in dashboards: *Shutdown Duration*
- Enabled through configuration setting: `ShutdownDetail`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
TotalShutdownTimeMs	Total shutdown time.	ms		700
UserSessionTimeMs	User session time.	ms		100
UserPolicyTimeMs	User policy time.	ms		100
UserProfilesTimeMs	User profiles time.	ms		100
SystemSessionsTimeMs	System sessions time.	ms		100
PreShutdownNotificationsTimeMs	Pre-shutdown notifications time.	ms		100

Field	Description	Data type	Unit	Example
ServicesTimeMs	Services time.	ms		100
KernelTimeMs	Kernel time.	ms		100

On/Off Transition Delay Metrics

Slow App Startup

uberAgent records which applications are responsible for delays during system boot. For each delay, uberAgent collects metrics like application name and version.

> **Note**
>
> This metric is only available on client operating systems.

Details

- Source type: `uberAgent:OnOffTransition:SlowAppStartup`
- Used in dashboards:
- Enabled through configuration setting: `BootDetail`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
Name	Application name.	String		explorer.exe
FriendlyName	Friendly application name.	String		Windows Explorer
Version	Application version.	String		6.2.9200.16384 (win8_rtm.120725-1247)
TotalTimeMs	Total time.	Number	ms	100

Field	Description	Data type	Unit	Example
DegradationTimeMs	Degradation (how much longer it took than normal).	Number	ms	20

Slow App Shutdown

uberAgent records which applications are responsible for delays during system shutdown. For each delay, uberAgent collects metrics like application name and version.

> **Note**
>
> This metric is only available on client operating systems.

Details

- Source type: `uberAgent:OnOffTransition:SlowAppShutdown`
- Used in dashboards:
- Enabled through configuration setting: `ShutdownDetail`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
Name	Application name.	String		explorer.exe
FriendlyName	Friendly application name.	String		Windows Explorer
Version	Application version.	String		6.2.9200.16384 (win8_rtm.120725-1247)
TotalTimeMs	Total time.	Number	ms	100

Field	Description	Data type	Unit	Example
DegradationTimeMs	Degradation (how much longer it took than normal).	Number	ms	20

Slow App Standby

uberAgent records which applications are responsible for delays during standby. For each delay, uberAgent collects metrics like application name and version.

> **Note**
>
> This metric is only available on client operating systems.

Details

- Source type: `uberAgent:OnOffTransition:SlowAppStandby`
- Used in dashboards:
- Enabled through configuration setting: `StandbyDetail`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
Name	Application name.	String		stisvc
FriendlyName	Friendly application name.	String		Still Image Devices Service
Version	Application version.	String		6.2.9200.16384 (win8_rtm.120725-1247)
TotalTimeMs	Total time.	Number	ms	100

Field	Description	Data type	Unit	Example
DegradationTimeMs	Degradation (how much longer it took than normal).	Number	ms	20

Slow Service Startup

uberAgent records which services are responsible for delays during system boot. For each delay, uberAgent collects metrics like service name and version.

> **Note**
>
> This metric is only available on client operating systems.

Details

- Source type: `uberAgent:OnOffTransition:SlowServiceStartup`
- Used in dashboards:
- Enabled through configuration setting: `BootDetail`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
Name	Service name.	String		stisvc
FriendlyName	Friendly service name.	String		Still Image Devices Service
Version	Service version.	String		6.2.9200.16384 (win8_rtm.120725-1247)
TotalTimeMs	Total time.	Number	ms	100
DegradationTimeMs	Degradation (how much longer it took than normal).	Number	ms	20

Slow Service Shutdown

uberAgent records which services are responsible for delays during system shutdown. For each delay, uberAgent collects metrics like service name and version.

> **Note**
>
> This metric is only available on client operating systems.

Details

- Source type: `uberAgent:OnOffTransition:SlowServiceShutdown`
- Used in dashboards:
- Enabled through configuration setting: `ShutdownDetail`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
Name	Service name.	String		stisvc
FriendlyName	Friendly service name.	String		Still Image Devices Service
Version	Service version.	String		6.2.9200.16384 (win8_rtm.120725-1247)
TotalTimeMs	Total time.	Number	ms	100
DegradationTimeMs	Degradation (how much longer it took than normal).	Number	ms	20

Slow Service Hybrid Standby

uberAgent records which services are responsible for delays during hybrid standby. For each delay, uberAgent collects metrics like service name and version.

> **Note**
>
> This metric is only available on client operating systems.

Details

- Source type: `uberAgent:OnOffTransition:SlowServiceHybridStandby`
- Used in dashboards:
- Enabled through configuration setting: `StandbyDetail`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
Name	Service name.	String		stisvc
FriendlyName	Friendly service name.	String		Still Image Devices Service
Version	Service version.	String		6.2.9200.16384 (win8_rtm.120725-1247)
TotalTimeMs	Total time.	Number	ms	100
DegradationTimeMs	Degradation (how much longer it took than normal).	Number	ms	20

Slow Driver Startup

uberAgent records which drivers are responsible for delays during system boot. For each delay, uberAgent collects metrics like driver name and version.

> **Note**
>
> This metric is only available on client operating systems.

Details

- Source type: `uberAgent:OnOffTransition:SlowDriverStartup`
- Used in dashboards:
- Enabled through configuration setting: `BootDetail`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
Name	Driver name.	String		\Driver\pci
FriendlyName	Friendly driver name.	String		NT Plug and Play PCI Enumerator
Version	Driver version.	String		6.2.9200.16384 (win8_rtm.120725-1247)
TotalTimeMs	Total time.	Number	ms	100
DegradationTimeMs	Degradation (how much longer it took than normal).	Number	ms	20

Slow Driver Shutdown

uberAgent records which drivers are responsible for delays during system shutdown. For each delay, uberAgent collects metrics like driver name and version.

> **Note**
>
> This metric is only available on client operating systems.

Details

- Source type: `uberAgent:OnOffTransition:SlowDriverShutdown`
- Used in dashboards:
- Enabled through configuration setting: *ShutdownDetail*
- Related configuration settings: *n/a*

- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
Name	Driver name.	String		\Driver\pci
FriendlyName	Friendly driver name.	String		NT Plug and Play PCI Enumerator
Version	Driver version.	String		6.2.9200.16384 (win8_rtm.120725-1247)
TotalTimeMs	Total time.	Number	ms	100
DegradationTimeMs	Degradation (how much longer it took than normal).	Number	ms	20

Slow Driver Standby

uberAgent records which drivers are responsible for delays during standby. For each delay, uberAgent collects metrics like driver name and version.

> **Note**
>
> This metric is only available on client operating systems.

Details

- Source type: `uberAgent:OnOffTransition:SlowDriverStandby`
- Used in dashboards:
- Enabled through configuration setting: `StandbyDetail`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
Name	Driver name.	String		\Driver\pci
FriendlyName	Friendly driver name.	String		NT Plug and Play PCI Enumerator
Version	Driver version.	String		6.2.9200.16384 (win8_rtm.120725-1247)
TotalTimeMs	Total time.	Number	ms	100
DegradationTimeMs	Degradation (how much longer it took than normal).	Number	ms	20
DeviceFriendlyName	Device friendly name.	String		Ricoh PCIe SDXC/MMC Host Controller

Slow Driver Resume

uberAgent records which drivers are responsible for delays during resume. For each delay, uberAgent collects metrics like driver name and version.

> **Note**
>
> This metric is only available on client operating systems.

Details

- Source type: `uberAgent:OnOffTransition:SlowDriverResume`
- Used in dashboards:
- Enabled through configuration setting: `StandbyDetail`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
Name	Driver name.	String		\Driver\pci
FriendlyName	Driver friendly name.	String		NT Plug and Play PCI Enumerator
Version	Driver version.	String		6.2.9200.16384 (win8_rtm.120725-1247)
TotalTimeMs	Total time.	Number	ms	100
DegradationTimeMs	Degradation (how much longer it took than normal).	Number	ms	20
DeviceFriendlyName	Device friendly name.	String		Ricoh PCIe SDXC/MMC Host Controller

Slow User Policy

DEPRECATED: this sourcetype is deprecated and will be removed in a future release.

uberAgent records which user policies are responsible for delays during system boot. For each delay, uberAgent collects metrics like name and total time.

Details

- Source type: `uberAgent:OnOffTransition:SlowUserPolicy`
- Used in dashboards:
- Enabled through configuration setting: `BootDetail`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
Name	User policy name.	String		PreShellInit
TotalTimeMs	Total time.	Number	ms	100

Field	Description	Data type	Unit	Example
DegradationTimeMs	Degradation (how much longer it took than normal).	Number	ms	20

Slow SMSS Initialization

DEPRECATED: this sourcetype is deprecated and will be removed in a future release.

uberAgent records SMSS initialization is responsible for delays during system boot. For each delay, uberAgent collects metrics like name and total time.

Details

- Source type: `uberAgent:OnOffTransition:SlowSMSSInit`
- Used in dashboards:
- Enabled through configuration setting: `BootDetail`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
Name	SMSSInit name.	String		SMSSInit
TotalTimeMs	Total time.	Number	ms	100
DegradationTimeMs	Degradation (how much longer it took than normal).	Number	ms	20

Standby/Resume Duration Metrics

Standby Detail

uberAgent collects information about each standby/resume like duration, the reason for waking up as well as pages written to the hibernation file.

> **Note**
>
> This metric is only available on client operating systems.

Details

- Source type: `uberAgent:OnOffTransition:StandbyDetail2`
- Used in dashboards: *Standby/Resume Duration*
- Enabled through configuration setting: `StandbyDetail`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
SleepTime	Time the sleep started.	String		2018-07-31 19:02:37.582 +0200
WakeTime	Time the wake completed.	String		2018-07-32 07:31:58.434 +0200
EnterStandbyMs	Duration of the standby process.	Number	ms	5737
ResumeFromStandbyMs	Duration of the resume process.	Number	ms	1269
DriverInitDuration	Duration of driver initialization at resume.	Number	ms	897
BiosInitDuration	Duration of BIOS initialization at resume.	Number	ms	0

Field	Description	Data type	Unit	Example
HiberWriteDuration	Duration of write to hibernation file.	Number	ms	8034
HiberReadDuration	Duration of read from hibernation file.	Number	ms	4419
HiberPagesWritten	Memory pages written to the hibernation file.	Number		738245
Attributes	n/a	Number		33571073
TargetState	Target sleep state. Possible values: 0-7. For a string representation see the calculated field `TargetStateDisplayName` .	Number		5
EffectiveState	Actual sleep state. Possible values: 0-7. For a string representation see the calculated field `EffectiveStateDisplayName` .	Number		5
WakeSourceType	Source of a wake up (what woke the machine up?). Possible values: 0-6. For a string representation see the calculated field `WakeSourceTypeDisplayName` .	Number		1

Field	Description	Data type	Unit	Example
WakeSourceTextLength	Number of characters in the field `WakeSourceText`.	Number		125
WakeSourceText	Description of the source of a wake up	String		Windows will execute ‘NT TASK\Microsoft\Windows\Upd scheduled task that requested waking the computer.
WakeTimerOwnerLength	Number of characters in the field `WakeTimerOwner`.	Number		52
WakeTimerContextLength	Number of characters in the field `WakeTimerContext`.	Number		18
NoMultiStageResumeReason	n/a	Number		0
WakeTimerOwner	If a wake up was caused by a timer: owner of the timer.	String		\Device\HarddiskVolume4\Win
WakeTimerContext	If a wake up was caused by a timer: additional information.	String		SystemEventsBroker

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
TargetStateDisplayName	String representation of the field `TargetState`. Possible values: `undefined`, `S0`/`working`, `S1`, `S2`, `S3`/`standby`, `Hibernate`, `Shutdown`, `Maximum enumeration value`.	String		Hibernate	Splunk data model, Splunk SPL
EffectiveStateDisplayName	String representation of the field `EffectiveState`. Possible values: `undefined`, `S0`/`working`, `S1`, `S2`, `S3`/`standby`, `Hibernate`, `Shutdown`, `Maximum enumeration value`.	String		Hibernate	Splunk data model, Splunk SPL

Field	Description	Data type	Unit	Example	Where available
WakeSourceTypeDisplayName	String representation of the field WakeSourceType. Possible values: unknown, Power Button, S4 Doze to Hibernate, Device, Device or Timer, Timer.	String		Timer	Splunk data model, Splunk SPL

Network Metrics

Network Communication

uberAgent collects metrics like source process (process sending/receiving data on the machine uberAgent is running on) as well as IP address and port per network target and sending/receiving process. A network target is a communication endpoint from the point of view of the machine uberAgent is running on. uberAgent distinguishes between different services on the target machine and can show latencies for, say, SMB and SQL Server independently. Of course, uberAgent supports both IPv4 and IPv6! uberAgent optionally only shows new ports and targets never seen before.

Notes:

- Field: `AppVersion` - uberAgent has an internal filter to minimize data volume by suppressing version information for system processes and system services. As a result, the `AppVersion` field is typically empty for most system processes and services.

Details

- Source type: `uberAgent:Process:NetworkTargetPerformance`

- Used in dashboards: *Application Network Issues*, *Citrix XA/XD Databases*, *Process Network Communication*, *Application Network Communication*, *Machine Network Communication*, *Single Application Performance*, *Single Machine Detail*, *Single User Detail*
- Enabled through configuration setting: `NetworkTargetPerformanceProcess`
- Related configuration settings: [`NetworkTargetPerformanceProcess_Config`]
- Supported platform: *all*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
ProcName	Process name.	String		chrome.exe
ProcUser	Process user.	String		Domain\JohnDoe
NetTargetRemoteAddress	Network target remote address.	String		138.201.31.60
NetTargetRemoteName	Network target remote name.	String		vastlimits.com
NetTargetRemotePort	Network target remote port.	Number		443
NetTargetSendCount	Count of packets to network target containing payload.	Number		15
NetTargetReceiveCount	Count of packets from network target containing payload.	Number		14
NetTargetConnectCount	Count of connects to network target.	Number		1
NetTargetSendKBPS	Kilobytes per second sent to network target.	Number	KB/s	1024
NetTargetReceiveKBPS	Kilobytes per second received from network target.	Number	KB/s	1024

Field	Description	Data type	Unit	Example
NetTargetSendMB	Amount of data volume send to network target. Payload only, no protocol overhead.	Number	MB	1
NetTargetReceiveMB	Amount of data volume received from network target. Payload only, no protocol overhead.	Number	MB	1
NetTargetSendLatencyMs	Latency to network target.	Number	ms	100
NetTargetProtocols	Protocols used.	String		TCP
NetTargetSendLatencyCount	Count of events with latency.	Number		2
AppId	Associated application ID. Used by uberAgent to lookup application names and populate field AppName.	String		GglChrm
NetTargetReconnectCount	Count of reconnects to network target.	Number		3
NetTargetRetransmitCount**	Count of retransmits to network target.	Number		3
AppVersion	Application version.	String		67.0.3396.99
NetTargetSendJitterMs	Jitter to network target.	Number	ms	5

Field	Description	Data type	Unit	Example
NetTargetSendJitterCount*	Count of events with jitter.	Number		1
NetTargetSourceAddress	Network source IP.	String		127.0.0.1
NetTargetSendLatencyInitialMs*	Initial latency to network target (TCP handshake).	Number	ms	100
NetTargetSendLatencyInitialCount*	Count of events with initial latency.	Number		1
NetTargetSourcePort*	Network source port.	Number		43021

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
NetTargetRemoteNameAddress	Contains `NetTargetRemoteName`. If `NetTargetRemoteName` is not filled, `NetTargetRemoteAddress` is used instead.	String		vastlimits.com	Splunk data model
NetTargetRemoteNameAddressPort	Concatenation of `NetTargetRemoteNameAddress` and `NetTargetRemotePort`.	String		vastlimits.com:443	Splunk data model
NetTargetSendReceiveMB	`NetTargetSendMB` + `NetTargetReceiveMB`.	Number	MB	2	Splunk data model

Field	Description	Data type	Unit	Example	Where available
NetTargetSendReceiveCount	Count of sends and receive events combined.	Number		29	Splunk data model
NetTargetSendReceiveKBPS	*NetTargetSendKBPS* + *NetTargetReceiveKBPS*.	Number	KB/s	2048	Splunk data model
NetTargetSendDurationMs	`NetTargetSendLatencyMs` x `NetTargetSendLatencyCount`.	Number	ms	200	Splunk data model
AppName	Associated application name.	String		Google Chrome	Splunk data model, Splunk SPL
ProcUser	Process user.	String		Domain\JohnDoe	Splunk data model, Splunk SPL
User	User name.	String		Domain\JohnDoe	Splunk data model
ProcessName	Process name.	String		chrome.exe	Splunk data model

Network Connection Failures

uberAgent collects metrics like source application name as well as protocols used whenever a network connection attempt fails.

Details

- Source type: `uberAgent:Application:NetworkConnectFailure`
- Used in dashboards: *Application Network Issues*, *Machine Network Issues*, *Process Network Issues*
- Enabled through configuration setting: `NetworkTargetPerformanceProcess`
- Related configuration settings: `[NetworkTargetPerformanceProcess_Config]`
- Supported platform: *all*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
AppId	Associated application ID. Used by uberAgent to lookup application names and populate field AppName.	String		GglChrm
AppVersion	Application version.	String		67.0.3396.99
ProcessName	Process name.	String		chrome.exe
ProcessId	Process ID.	Number		456
SessionGUID	Unique identifier that is generated by uberAgent when the session is created. Valid for this session only.	String		00000002-f295-9109-e7c7-c964011dd401
NetTargetRemoteAddress	Network target remote address.	String		138.201.31.60
NetTargetRemoteName	Network target remote name.	String		vastlimits.com
NetTargetRemotePort	Network target remote port.	Number		443
NetTargetProtocols	Protocols used. The only protocol supported is TCP.	String		TCP
NetTargetSourceAddress	Network source IP.	String		127.0.0.1
NetTargetSourcePort	Network source port.	Number		43021

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
AppName	Associated application name.	String		Google Chrome	Splunk data model, Splunk SPL
ProcName	Process name.	String		chrome.exe	Splunk data model
User	User name.	String		Domain\JohnDoe	Splunk data model, Splunk SPL
NetTargetRemoteNameAddress	Content of NetTargetRemoteName. If NetTargetRemoteName is not filled, NetTargetRemoteAddress is used instead.	String		vastlimits.com	Splunk data model

* Fields only available when NetworkDriverEnabled = **true**.
** In cases of high network activity combined with a large number of retransmissions, uberAgent may not detect all retransmissions. This design choice ensures minimal overhead and avoids any negative impact on system performance.

Microsoft Office Metrics

Outlook Plugin Load

uberAgent collects Outlook plugin load duration and related information like the ProgID and GUID.

Details

- Source type: uberAgent:Application:OutlookPluginLoad
- Used in dashboards: *Outlook Plugin Load Performance*
- Enabled through configuration setting: OutlookPerformanceEvents
- Related configuration settings: *n/a*

- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Measurement type	Example
Name	Plugin name (may change with version and language).	String		Snapshot	Egnyte Outlook Add-In
ProgID	Unique plugin ID (language-independent).	String		Snapshot	EgnyteAddin.Connect
GUID	Plugin GUID (language-independent).	String		Snapshot	CEBEF5C6-2F24-4EC1-A005-A04E55EF0C52
LoadBehavior	Plugin load behavior ID. Possible values: `0`, `1`, `2`, `3`, `8`, `9`, `16`.	Number		Snapshot	3
HKLM	Indicates if plugin is configured for all users. Possible values: `0`, `1`.	Number		Snapshot	0
BootTimeMs	Plugin load duration.	Number	ms	Sum	6000

List of Calculated Fields

Field	Description	Data type	Unit	Measurement type	Where available	Example
LoadBehaviorDisplayName	Plugin load behavior name. Possible values: `Do not load automatically` (0), `Do not load automatically` (1), `Load at startup` (2), `Load at startup` (3), `Load on demand` (8), `Load on demand` (9), `Load first time then load on demand` (16).	String		Snapshot	Splunk data model	Load at startup (3)
BootTimeS	Plugin load duration.	Number	s	Sum	Splunk data model	6.00

Session Metrics

Session Detail

uberAgent collects metrics like the session ID, connection state, protocol latency, and foreground application per user session.

Details

- Source type: `uberAgent:Session:SessionDetail`
- Used in dashboards: *Application Usage*, *Machine Uptime*, *SBC Sizing and Capacity Planning*, *Session 0*, *Session Info: Citrix*, *Session Info: VMware*, *User Session Overview*, *User Sessions*, *Single Application Performance*, *Single Machine Performance*, *Single User Performance*, *Analyze data over time*
- Enabled through configuration setting: `SessionDetail`
- Related configuration settings: *n/a*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example	Platform
SessionID	Session ID generated by the OS. Session IDs are reused and cannot be used to uniquely identify a session. Use `SessionGUID` for that purpose instead.	Number		3	all
SessionLogonTime	Time when the user logged on.	String		2018-07-23 10:06:02.123 +0200	all

Field	Description	Data type	Unit	Example	Platform
SessionProtocol	Remoting protocol used to connect to the session. Possible values on Windows: `Console`, `ICA`, `RDP`, `PCoIP`, `Blast`, `Frame`, `Protocol [ID]`. Possible values on macOS: `Console`, `RFB`, `SSH`, `Protocol [ID]`.	String		ICA	all

Field	Description	Data type	Unit	Example	Platform
SessionConnectionState	Session connection state. Possible values: `active`, `connected`, `connecting`, `shadowing`, `disconnected`, `idle`, `listening`, `reset`, `down`, `initializing`, `unknown`, `Unsupported state: [ID]`.	String		active	Win
SessionProcessCount	Number of processes that are running in the session.	Number		34	all
SessionCPUTimeMs	Session CPU usage (absolute usage in milliseconds).	Number	ms	3000	all
SessionCPUUsagePercent	Process CPU usage (relative usage in percent).	Number	%	29	all
SessionIOPS	I/O operations per second of all processes in the session combined.	Number		500	Win

Field	Description	Data type	Unit	Example	Platform
SessionIOCount	I/O operation count of all processes in the session combined.	Number		300	Win
SessionIOMB	I/O data volume of all processes in the session combined.	Number	MB	20	all
SessionIOLatencyMs	Latency of I/O operations.	Number	ms	300	Win
SessionWorkingSetMB	RAM usage (working set) of all processes in the session combined.	Number	MB	450.2	all
SessionNetKBPS	Process network traffic data volume per second of all processes in the session combined.	Number	KB	21	Win
SessionUser	User name.	String		Domain\JohnDoe	all
SessionGUID	Unique identifier for a session that is generated by uberAgent.	String		00000002-f295-9109-e7c7-c964011dd401	all

Field	Description	Data type	Unit	Example	Platform
SessionRpLatencyMs2	Remoting protocol latency. This field is filled with the latency in ms at collection time. If the protocol is ICA, uberAgent uses the perfmon counter `Latency - Last Recorded` as the source. This metric is different from Citrix Round-Trip-Time (more information).	Number	ms	30	Win
SessionClientMac	Remote client MAC address. Only collected for VMware sessions.	String		AC-ED-5C-02-F0-30	Win
SessionClientIp	Remote client IP address.	String		192.168.178.40	all
SessionClientName	Remote client name.	String		Client1	Win
SessionClientDomain	Remote client Active Directory domain.	String		vastlimits.com	Win

Field	Description	Data type	Unit	Example	Platform
SessionClientUser	Remote client user name.	String		JohnDoe	Win
SessionClientUserDomain	Remote client user Active Directory domain.	String		Domain	Win
SessionHRes	Horizontal resolution.	String		1920	Win
SessionVRes	Vertical resolution.	String		1080	Win
SessionColorDepth	Color depth.	String		32	Win
SessionClientPlatform	Remote client platform.	String		Windows	Win
SessionClientVersion	Remote client version.	String		13.0.0.256735	Win
SessionClientOsLanguage	Remote client operating system language.	String		en-us	Win
SessionPublishedName	Published resource name. This resource was used to start or connect to the session.	String		Google Chrome	Win
SessionPublishedAppsCtx	List of used Citrix published apps in the session.	String		Word Excel	Win

Field	Description	Data type	Unit	Example	Platform
SessionAppStateCtx	Citrix application state. Possible values: `n/a`, `active`, `pre-launched`, `lingering`, `app not running`, `unsupported`.	String		lingering	Win
SessionEncryptionCtx	Citrix remoting protocol encryption. Possible values: `unknown`, `basic`, `logon`, `40 bit`, `56 bit`, `128 bit`, `SecureICA`, `unsupported`.	String		SecureICA	Win
SessionClientTypeCtx	Citrix client type. Possible values: `WI`, `ICA Client`.	String		WI	Win
SessionBrokerDnsVmw	VMware broker DNS name.	String		HorizonCS.vastlimits.com	Win
SessionBrokerUrlVmw	VMware broker URL.	String		https://192.168.8.6:443	Win

Field	Description	Data type	Unit	Example	Platform
SessionBrokerTunneledVmw	VMware indicates if session is tunneled. Possible values: **`true`**, **`false`**.	String		true	Win
SessionBrokerTunnelUrlVmw	VMware URL of the View Connection Server tunnel connection.	String		https://HorizonCS.vastlimits.com:443	Win
SessionBrokerRemoteIpVmw	VMware broker remote IP address.	String		192.168.178.40	Win
SessionBrokerUserVmw	VMware broker user name.	String		John	Win
SessionBrokerDomainVmw	VMware broker user domain name.	String		Doe	Win
SessionClientTimezoneVmw	VMware client timezone.	String		Europe/Italy	Win
SessionClientIdVmw	VMware client ID.	String		c47c60cf41ec44829ff91e1822b24dd8	Win
SessionTypeVmw	VMware session type. Possible values: `application`, `desktop`.	String		application	Win

Field	Description	Data type	Unit	Example	Platform
SessionBrokerType	Remoting broker type. Possible values: `Citrix`, `Microsoft`, `VMware`, `Nutanix Frame`, `none`, `unknown`, `Unsupported broker: [number]`.	String		Citrix	Win
SessionFgAppId	Foreground application ID. Used by uberAgent to look up application names and populate the field `SessionFgAppName`.	String		GglChrm	all
SessionFgAppVersion	Foreground application version.	String		67.0.3396.99	all
SessionFgProcessName	Foreground process name.	String		chrome.exe	all
SessionFgProcessId	Foreground process ID.	Number		456	all

Field	Description	Data type	Unit	Example	Platform
SessionFgBrowserType	If the foreground window is a browser: browser type. Possible values: 1, 2, 3, 4, 5. Also see the field `SessionFgBrowserDisplayName`.	Number		1	all
SessionFgBrowserActiveTabHost	If the foreground window is a browser: the URL of active tab. See this KB article for troubleshooting empty values.	String		uberagent.com	all
SessionFgWindowTitle	The title of the active foreground window.	String		Search | Splunk 7.2.1 - Google Chrome	all
SessionClientHwIdCitrix2	Citrix client hardware ID.	Number		409D39C2	Win
SessionRoundTripTimeMs	Citrix session round trip time.	Number	ms	50	Win

Field	Description	Data type	Unit	Example	Platform
SessionFps	Remoting protocol frames per second at collection time. Only available for Citrix and Nutanix Frame sessions.	Number		20	Win
SessionTransportProtocols	The protocol chain used for this connection. Only available for Citrix sessions.	String		UDP-CGP-ICA	Win

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
AppName	Application name. Source is the field `SessionFgAppId` .	String		Google Chrome	Splunk data model, Splunk SPL
SessionCPUTime	Session CPU usage (absolute usage in seconds).	Number	s	3	Splunk data model

Field	Description	Data type	Unit	Example	Where available
SessionIODurationMs	SessionIOLatencyMs x SessionIOCount.	Number	ms	90000	Splunk data model
SessionRAMUsagePercent	Percentage of the machine's RAM used by the session.	Number	%	34	Splunk data model
User	Alias for SessionUser.	String		Domain\JohnDoe	Splunk data model
SessionUserLower	User name converted to lower case.	String		domain\johndoe	Splunk data model
SessionDisplaySpecs	SessionHRes + "x" + SessionVRes + "x" + SessionColorDepth.	String		1920x1080x32	Splunk data model
SessionPublishedAppsCtxSplit	SessionPublishedAppsCtx split by semicolon.	String		Word;Excel	Splunk data model
SessionPublishedAppsCtxSplitLower	SessionPublishedAppsCtxSplit converted to lower case.	String		word;excel	Splunk data model
SessionFgAppName	Foreground application name.	String		Google Chrome	Splunk data model, Splunk SPL

Field	Description	Data type	Unit	Example	Where available
SessionFgBrowserDisplayName	Foreground browser name. Possible values: `Chrome`, `Internet Explorer`, `Firefox`, `Edge`, `Citrix Enterprise Browser`.	String		Chrome	Splunk data model, Splunk SPL
SessionClientHwIdCtx	`SessionClientHwIdCtx2` or `SessionClientHwIdCtx`, whichever is present.	Number		409D39C2	Splunk data model

Performance Counter Metrics

uberAgent can be configured to collect the values of Windows performance counters. Please see this document for more information.

Details

- Source type: `uberAgent:System:PerformanceCounter`
- Used in dashboards: *Windows Performance Counters*
- Enabled through configuration setting: `Perf counter`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
PerformanceCounterObject	Performance counter object.	String		Process
PerformanceCounterInstance	Performance counter instance.	String		conhost
PerformanceCounterName	Performance counter name.	String		Handle Count
PerformanceCounterValue	Performance counter value.	Number		100

uberAgent Licensing Metrics

uberAgent Licensing

Information on uberAgent licenses.

Details

- Source type: `uberAgent:License:LicenseInfo`
- Used in dashboards: *Licensing Status*, *uberAgent Versions*
- Enabled through configuration setting: *n/a*
- Related configuration settings: *n/a*
- Supported platform: *all*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
LicensingState	Licensing state. Possible values: `licensed`, `unlicensed`.	String		licensed
LicenseId	License ID.	String		407d288f-02f7-4375-ae44-ea36654772fa

Field	Description	Data type	Unit	Example
LicenseCountTotal	Total number of licenses.	Number		2000
LicensingModel	Licensing model. Possible values: `device`, `user`, `site`.	String		device
LicensingModelDetail	Contains more detailed information about the licensing model like if server or client licenses are in use.	String		server
LicensingType	Licensing type. Possible values: `eval`, `NFR`, `full`.	String		full
MaintenanceEnd	Maintenance end. Only for perpetual licenses.	String		2018-07-07 01:59:59.000 +0200
Expiration	Expiration date. Only for one-year term, eval, and community licenses.	String		2018-07-07 01:59:59.000 +0200
LicensedComponents	Licensed components.	String		unrestricted
ProductVersion	Product version.	String		5.0.1.1609

User & Host Metrics

User & Host Tags

uberAgent can collect custom tags for users and machines (more information).

Details

- Source type: `uberAgent:Tags:UserHost`
- Used in dashboards: *All except Citrix XA/XD and ADC dashboards*
- Related configuration settings: `[UserHostTagging]`
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
User	The name of the user (in the case of a user tag).	String		domain\User1
TagType	The type of tag (User or Host).	String		User
TagName	The user-defined name of the tag.	String		Department
TagSource	The source of the tag (`AD`, `Environment` or `Registry`).	String		Registry
TagValue	Defines the complete path of the tag. This can be an AD attribute, an environment variable or the full path to a registry value. The latter must include the registry hive, key, and value.	String		HKCU/Software/vast limits/Department
TagData	The determined value for the current tag.	String		Development

User Logoff Metrics

Logoff Detail

uberAgent collects various details about logoffs like profile unload time, Group Policy logoff script time as well as process performance.

Notes:

- Field: `AppVersion` - uberAgent has an internal filter to minimize data volume by suppressing version information for system processes and system services. As a result, the `AppVersion` field is typically empty for most system processes and services.

Details

- Source type: `uberAgent:Logoff:LogoffDetail`
- Used in dashboards: *User Logoff Duration*, *Single Logoff*
- Enabled through configuration setting: `LogonDetail`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Measurement type	Example
SessionGUID	Unique identifier that is generated by uberAgent when the session is created. Valid for this session only.	String		Snapshot	00000002-f295-9109-e7c7-c964011dd401

Field	Description	Data type	Unit	Measurement type	Example
SessionID	Unique identifier that is generated by the machine when the session is created. Will be reassigned to other sessions after logoff.	Number		Snapshot	3
User	User name.	String		Snapshot	Domain\JohnDoe
SessionLogoffTime	Time when the user logged off.	String		Snapshot	2018-07-23 10:06:02
SessionEndTime	Time when the session ended.	String		Snapshot	2018-07-23 10:08:02
SessionDurationMs	Session duration.	Number	ms	Sum	25000
ProfileUnloadTimeMs2	User profile unloading time.	Number	ms	Sum	300
GroupPolicyLogoffScriptTimeMs	Group Policy logoff script processing time.	Number	ms	Sum	358
TotalLogoffTimeMs	Logoff duration combined for all phases.	Number	ms	Sum	40000
ProcessStartCount	Number of processes started.	Number		Count	8

Field	Description	Data type	Unit	Measurement type	Example
IOCountRead	Count of read I/O operations.	Number		Count	100
IOCountWrite	Count of write I/O operations.	Number		Count	100
IOMBRead	Amount of read data volume.	Number	MB	Sum	50
IOMBWrite	Amount of write data volume.	Number	MB	Sum	50
IOLatencyReadMs	I/O read operation duration divided by count of read I/O operations.	Number	ms	Average	358
IOLatencyWriteMs	I/O write operation duration divided by count of write I/O operations.	Number	ms	Average	358

Logoff processes

Detailed performance data about all processes active during user logoff like process start time and lifetime duration, commandline, executable path, and CPU footprint.

Details

- Source type: `uberAgent:Process:LogoffProcesses`
- Used in dashboards: *Single Logoff*

- Enabled through configuration setting: `LogonProcesses`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Measurement type	Example
ProcName	Process name.	String		Snapshot	chrome.exe
ProcID	Process ID.	Number		Snapshot	456
ProcParentName	Parent process name.	String		Snapshot	PowerShell.exe
ProcParentID	Parent process ID.	Number		Snapshot	789
ProcUser	User who ran the process.	String		Snapshot	Domain\JohnDoe
AppId	Associated application ID. Used by uberAgent to lookup application names and populate field `AppName`.	String		Snapshot	GglChrm
AppVersion	Associated application version.	String		Snapshot	67.0.3396.99

Field	Description	Data type	Unit	Measurement type	Example
LogoffProcType	uberAgent groups processes running during logon into types. Possible values: `Other`, `GP logoff script`, `Session teardown`.	String		Snapshot	Other
ProcStartTimeRelativeMs	Process relative start time.	Number	ms	Snapshot	16764
ProcLifetimeMs	Process lifetime.	Number	ms	Sum	73615
ProcCmdline	Process command line.	String		Snapshot	C:\Program Files (x86)\Google\Chrome\Appl –url http://vastlimits.com
ProcPath	Process path.	String		Snapshot	C:\Program Files (x86)\Google\Chrome\Appl
ProcIOReadCount	Process I/O operation read count.	Number		Count	2000
ProcIOWriteCount	Process I/O operation write count.	Number		Count	990
ProcIOReadMB	Process I/O operation read data volume.	Number	MB	Sum	100.05

Field	Description	Data type	Unit	Measurement type	Example
ProcIOWriteMB	Process I/O operation write data volume.	Number	MB	Sum	16.06
ProcIOLatencyReadMs2	Process I/O operation read latency.	Number	ms	Average	300
ProcIOLatencyWriteMs2	Process I/O operation write latency.	Number	ms	Average	300
ProcNetKBPS	Process generated network traffic.	Number	KB	Sum	19.18
SessionGUID	Unique identifier that is generated by uberAgent when the session is created. Valid for this session only.	String		Snapshot	00000002-f295-9109-e7c7-c964011dd401
SessionID	Unique identifier that is generated by the machine when the session is created. Will be reassigned to other sessions after logoff.	Number		Snapshot	3
TotalLogoffDurationMs	Total logoff duration.	Number	ms	Sum	40000

Field	Description	Data type	Unit	Measurement type	Example
SortOrder2	Sort order number to sort the table *Boot process performance* on the *Single Boot* dashboard correctly.	Number		Snapshot	29

List of Calculated Fields

Field	Description	Data type	Unit	Measurement type	Where available	Example
AppName	Associated application name.	String		Snapshot	Splunk data model, Splunk SPL	Google Chrome
SortOrder	Sort order number to sort the table *Boot process performance* on the *Single Boot* dashboard correctly.	Number		Snapshot	Splunk data model	29

User Logon Metrics

Logon Detail

uberAgent collects various details about logons like profile load time, Group Policy processing time as well as process performance.

Notes:

- Field: `AppVersion` - uberAgent has an internal filter to minimize data volume by suppressing version information for system processes and system services. As a result, the `AppVersion` field is typically empty for most system processes and services.

Details

- Source type: `uberAgent:Logon:LogonDetail`
- Used in dashboards: *Session Info: Citrix*, *Session Info: VMware*, *User Logon Duration*, *User Logon Duration - Group Policy*, *User Session Overview*, *User Sessions*, *Single Machine Detail*, *Single Logon*, *Single User Detail*
- Enabled through configuration setting: `LogonDetail`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Measurement type	Example
SessionGUID	Unique identifier that is generated by uberAgent when the session is created. Valid for this session only.	String		Snapshot	00000002-f295-9109-e7c7-c964011dd401

Field	Description	Data type	Unit	Measurement type	Example
SessionID	Unique identifier that is generated by the machine when the session is created. Will be reassigned to other sessions after logoff.	Number		Snapshot	3
User	User name.	String		Snapshot	Domain\JohnDoe
SessionLogonTime	Time when the user logon started.	String		Snapshot	2018-07-23 08:50:14
SiteName	Active Directory site name.	String		Snapshot	Default-First-Site-Name
LogonServer	Authenticating Active Directory domain controller.	String		Snapshot	DC1
ProfileLoadTimeMs	User profile loading time - Microsoft user profile service.	Number	ms	Sum	40000
CitrixPMLoadTimeMs	User profile loading time - Citrix Profile Management.	Number	ms	Sum	40000
GroupPolicyTotalProcessingTimeMs	Total Group Policy processing time.	Number	ms	Sum	250

Field	Description	Data type	Unit	Measurement type	Example
DcDiscoveryTimeMs	Domain controller discovery time	Number	ms	Sum	10
LoopbackMode	Group Policy loopback mode. Possible values: `replace`, `merge`, `no loopback`.	String		Snapshot	replace
ADLogonScriptTimeMs	Active Directory logon script processing time.	Number	ms	Sum	358
GroupPolicyLogonScriptTimeMs	Group Policy logon script processing time.	Number	ms	Sum	358
ResWmProcessingTimeMs	RES ONE Workspace shell startup time.	Number	ms	Sum	358
ShellStartupTimeMs	Shell startup time. Typically Windows Explorer.	Number	ms	Sum	358

Field	Description	Data type	Unit	Measurement type	Example
TotalLogonTimeMs	Total logon duration is defined as the time from the actual logon until the shell is fully initialized.	Number	ms	Sum	40000
ProcessStartCount	Number of processes started.	Number		Count	8
IOCountRead	Count of read I/O operations.	Number		Count	100
IOCountWrite	Count of write I/O operations.	Number		Count	100
IOMBRead	Amount of read I/O operation data volume.	Number	MB	Sum	50
IOMBWrite	Amount of write I/O operation data volume.	Number	MB	Sum	50
IOLatencyReadMs	I/O read operation duration divided by count of read I/O operations.	Number	ms	Average	358

Group Policy CSE Detail

uberAgent collects detailed information about Client-Side-Extensions (CSEs) like name, duration, and return code.

Details

- Source type: `uberAgent:Logon:GroupPolicyCSEDetail2`
- Used in dashboards: *Session Info: Citrix*, *Session Info: VMware*, *User Logon Duration*, *User Logon Duration - Group Policy*, *User Session Overview*, *User Sessions*, *Single Machine Detail*, *Single Logon*, *Single User Detail*
- Enabled through configuration setting: `LogonDetail`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Measurement type	Example
SessionGUID	Unique identifier that is generated by uberAgent when the session is created. Valid for this session only.	String		Snapshot	00000002-f295-9109-e7c7-c964011dd401

Field	Description	Data type	Unit	Measurement type	Example
SessionID	Unique identifier that is generated by the machine when the session is created. Will be reassigned to other sessions after logoff.	Number		Snapshot	3
User	User name.	String		Snapshot	Domain\JohnDoe
CseName	Client-side extension name.	String		Snapshot	Citrix Group Policy
CseDurationS	Client-side extension processing time.	Number	s	Sum	5.40
CseGPONames	Group Policy where client-side extension is configured.	String		Snapshot	Default Domain Policy
CseReturnCode	Client-side extension processing return code. Everything except 0 is bad.	Number		Snapshot	0

Logon Processes

Detailed performance data about all processes active during user logon like process start time and lifetime duration, commandline, executable path, and CPU footprint.

Details

- Source type: `uberAgent:Process:LogonProcesses`
- Used in dashboards: *Single Logon*
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Measurement type	Example
ProcName	Process name.	String		Snapshot	chrome.exe
ProcID	Process ID.	Number		Snapshot	456
ProcParentName	Parent process name.	String		Snapshot	PowerShell.exe
ProcParentID	Parent process ID.	Number		Snapshot	789
ProcUser	User who ran the process.	String		Snapshot	Domain\JohnDoe
AppId	Associated application ID. Used by uberAgent to lookup application names and populate field `AppName`.	String		Snapshot	GglChrm
AppVersion	Associated application version.	String		Snapshot	67.0.3396.99

Field	Description	Data type	Unit	Measurement type	Example
LogonProcType	uberAgent groups processes running during logon into types. Possible values: `Other`,`Userinit`,`AppSetup`,`Active Setup`,`AD logon script`,`GP logon script`,`Shell`,`RES Workspace Manager shell`,`RES Workspace Manager shell child`,`GP software installation`,`Run once`,`Initial program,` `User profile`,`Group Policy`,`Session setup`,`First logon animation`.	String		Snapshot	GP logon script

Field	Description	Data type	Unit	Measurement type	Example
ProcStartTimeRelativeMs	Process relative start time.	Number	ms	Snapshot	16764
ProcLifetimeMs	Process lifetime.	Number	ms	Sum	73615
ProcCmdline	Process command line.	String		Snapshot	C:\Program Files (x86)\Google\Chrome\Appl –url http://vastlimits.com
ProcPath	Process path.	String		Snapshot	C:\Program Files (x86)\Google\Chrome\Appl
ProcCPUTimeMs	Process consumed CPU time.	Number	ms	Sum	11859
ProcIOReadCount	Process I/O operation read count.	Number		Count	2000
ProcIOWriteCount	Process I/O operation write count.	Number		Count	990
ProcIOReadMB	Process I/O operation read data volume.	Number	MB	Sum	100.05
ProcIOWriteMB	Process I/O operation write data volume.	Number	MB	Sum	16.06
ProcIOLatencyReadMs2	Process I/O operation read latency.	Number	ms	Average	300
ProcIOLatencyWriteMs2	Process I/O operation write latency.	Number	ms	Average	300

Field	Description	Data type	Unit	Measurement type	Example
ProcWorkingSetMB	Process consumed RAM.	Number	MB	Snapshot	500.06
ProcNetKBPS	Process generated network traffic.	Number	KB	Sum	19.18
SessionGUID	Unique identifier that is generated by uberAgent when the session is created. Valid for this session only.	String		Snapshot	00000002-f295-9109-e7c7-c964011dd401
SessionID	Unique identifier that is generated by the machine when the session is created. Will be reassigned to other sessions after logoff.	Number		Snapshot	3
TotalLogonDurationMs	Total logon duration.	Number	ms	Sum	40000

Field	Description	Data type	Unit	Measurement type	Example
SortOrder2	Sort order number to sort the table *Logon process performance_on the _Single Logon* dashboard correctly.	Number		Snapshot	29

List of Calculated Fields

Field	Description	Data type	Unit	Measurement type	Where available	Example
AppName	Associated application name.	String		Snapshot	Splunk data model, Splunk SPL	Google Chrome
SortOrder	Sort order number to sort the table *Boot process performance* on the *Single Boot* dashboard correctly.	Number		Snapshot	Splunk data model	1

uberAgent Configuration Metrics

uberAgent Configuration Metrics

Information about the configuration uberAgent is using, i.e., where uberAgent is getting its configuration from and which configuration version it is using.

Details

- Source type: `uberAgent:Config:ConfigInfo`
- Used in dashboards: *uberAgent Versions*
- Enabled through configuration setting: *n/a*
- Related configuration settings: *n/a*
- Supported platform: *all*

List of Fields in the Raw Agent Data

Field	Description	Data type	Example
ConfigOption	The configuration variant uberAgent is using. Valid values on Windows: `CCFM`, `Local (installation directory)`, `Group Policy`, `Local (ProgramData)`. Valid values on macOS: `CCFM`, `Local (installation directory)`	String	CCFM
ConfigSource	Source of the configuration values. Can be a file, the registry (Windows), or a configuration archive.	String	Example 1: `\\Server1\Share\Software\uberAgent\Configuration\7.1.0.5327\` Example 2: `/Library/Application Support/uberAgent/uberAgent.conf`

Field	Description	Data type	Example
ConfigTimeStamp	The version of the configuration, i.e., the timestamp (Unix epoch) of the last modification.	Number	1678092747

ESA Metrics

uberAgent ESA collects many different metrics, logically grouped into sourcetypes. All metrics are explained in detail in this chapter, including:

- Name and description
- Supported platform
- Unit and example
- Related source type and where to find it in the dashboards
- Calculated fields and and their availability in Splunk's data model and/or "Search Processing Language" (SPL)
- The uberAgent template configuration files can be found at our Github uberAgent-config repository.

Threat Detection Metrics

Process Tagging

uberAgent processes a rule set and applies tags accordingly.

Notes:

- Field: `AppVersion` - uberAgent has an internal filter to minimize data volume by suppressing version information for system processes and system services. As a result, the `AppVersion` field is typically empty for most system processes and services.

Details

- Source type: `uberAgentESA:ActivityMonitoring:ProcessTagging`
- Used in dashboards: *Threat Detection Events*

- Enabled through configuration setting: `ActivityMonitoring`
- Related configuration settings: [`ActivityMonitoringRule`]
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
EventType	Event type. Can be `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `10`, `11`, `12`, `13`, `14`, `15`, `16`, `17`, `18`, `19`, `20`, `21`, `22`, `23`, `24`, `25` or `26`. See also `EventTypeName` .	Number		4
ProcName	Process name.	String		svchost.exe
ProcParentName	Parent process name.	String		services.exe
ProcUser	Process user.	String		domain\JohnDoe
ProcLifetimeMs	Process lifetime.	Number	ms	500
ProcId	Process ID.	Number		12345
ProcParentId	Parent process ID.	Number		67890
ProcGUID	Process GUID.	String		4b3e3686-7854-4d98-0023-1e0e617bf2e4
ProcParentGUID	Parent process GUID.	String		d72ceb7e-7851-02ec-005d-139741c4afd6
ProcPath	Process path	String		C:\WINDOWS\System32\svcho
ProcCmdline	Process commandline.	String		C:\WINDOWS\System32\svcho -k LocalSystemNetworkRestricted

Field	Description	Data type	Unit	Example
ProcTag1	Rule tag: the tag assigned to events originating from the matching rule.	String		net-connect-suspicious-sources
ProcRiskScore1	Rule risk score: the risk score assigned to events originating from the matching rule.	Number		75
IsElevated	Indicates if the process was started elevated (admin rights).	String		1
SessionId	Session ID.	Number		2
SessionGUID	Session GUID.	String		00000000-b242-d759-7a63-d686b0ffd501
AppId	Application ID.	String		Svc:WdiSystemHost
AppVersion	Application version.	String		1.0
IsProtected	Indicates if the process was started protected.	String		1
EventCount	The number of identical events that occurred during the interval period.	Number		42
RuleAnnotation	JSON of rule annotations like security frameworks.	String		{“mitre_attack”: [“T1086”, “T1059.001”]}

Additionally, one can enhance the information sent to the back-end by defining a number of generic properties that will be sent along with the fields above. Any field listed under Common Event Properties, Network Event Properties, Image Load Event Properties, or Registry Event Properties can be used as a generic property.

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
Field	Description	Data type	Unit	Example	Where available
EventTypeName	Names for event types based on the lookup `lookup_process_tagging_eventtype`. Can be `Process.Start`, `Process.Stop`, `Image.Load`, `Net.Connect`, `Net.Receive`, `Net.Reconnect`, `Net.Retransmit`, `Net.Send`, `Reg.Key.Create`, `Reg.Value.Write`, `Reg.Delete`, `Reg.Key.Delete`, `Reg.Value.Delete`, `Reg.Key.SecurityChange`, `Reg.Key.Rename`,`Reg.Key.SetInformation`, `Reg.Key.Load`,	String		Process.Start	Splunk data model, Splunk SPL

Field	Description	Data type	Unit	Example	Where available
ProcUser	coalesce (ProcUserExpanded , ProcUser).	String		Domain\JohnDoe	Splunk data model
User	ProcUser.	String		Domain\JohnDoe	Splunk data model
TimestampMs	_time * 1000.	Number	ms	1585913547467	Splunk data model
RuleAnnotation.mitre_attack.id	One or more MITRE ATT&CK® IDs for the event. The source is the field *RuleAnnotation*.	String		T1086 T1059.001	Splunk data model, Splunk SPL

Generic Properties

Sometimes, when a Threat Detection rule matches an event, one would like to have more information than what the fields of the source type uberAgentESA:ActivityMonitoring:ProcessTagging provide. In such a case, one can define up to 10 generic properties per rule that can access the event information the query has access to. Any event property listed under Common Event Properties, Network Event Properties, Image Load Event Properties, or Registry Event Properties can be used as a generic property. Note that certain properties are only defined for specific event types. For instance, Net.Target.Port and Reg.Key.Path are only available for *network* and *registry* event types, respectively. Please refer to Event Types for a list of available event types, as well as the individual event properties documentation pages mentioned above.

Generic properties can be defined using one of the two syntaxes, long form:

```
GenericProperty1Name = ProcHash
GenericProperty1Data = Process.Hash.MD5
```

or short form:

```
GenericProperty1 = Process.Hash.MD5
```

in which case, the fields GenericProperty1Name and GenericProperty1Data, containing Process.Hash.MD5 and the process's MD5 hash respectively, will be sent to uberAgentESA:ActivityMonitoring:ProcessTagging.

DNS Query Monitoring Metrics

DNS Query Monitoring

uberAgent collects detailed information about DNS queries: the request, all responses, and the process from which the query originated.

Details

- Source type: uberAgentESA:Process:DnsQuery
- Used in dashboards: *Process DNS*
- Enabled through configuration setting: DnsMonitoring
- Related configuration settings: *n/a*
- Supported platform: *all*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
ProcName	Process name.	String		svchost.exe
ProcGUID	Process GUID.	String		4b3e3686-7854-4d98-0023-1e0e617bf2e4
DnsRequest	DNS query name.	String		www.example.com
DnsResponse	DNS query response.	String		10.1.3.12
DnsResponseType	DNS query response type (e.g.: A, AAAA, CNAME).	String		A
DnsEventCount	Number of requests in the last interval.	Number		1

Field	Description	Data type	Unit	Example
DnsRisk52Chars	Tests whether the DNS host name contains more than 52 chars.	Number		1
DnsRisk27UniqueChars	Tests whether the DNS host name contains more than 27 unique chars.	Number		1
DnsRiskEmptyResponse	Tests whether the response is either not available or empty (e.g., SOA).	Number		1
DnsRiskTXTRecord	Tests for the uncommon response type *TXT*.	Number		1
DnsRiskHighEntropy	Tests the DNS request for high entropy based on Shannon entropy.	Number		1
DnsResponseStatus	Dns response status. Empty if the query was successful. Any other value indicates an error.	Number		9501

The fields `DnsRequest`, `DnsResponse`, and `DnsResponseType` may contain multiple values, separated by a semicolon `;`.

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
TimestampMs	`_time` * 1000.	Number	ms	1585913547467	Splunk data model

Event Log Forwarding Metrics

Event Log Forwarding

uberAgent can collect Windows system logs from the Eventlog: Application, Security, etc, with advanced filtering capabilities aimed to reduce the data volume processed by the agent.

Details

- Source type: `uberAgentESA:System:WinEvtLogForwarding`
- Used in dashboards: *Windows Eventlogs*
- Enabled through configuration setting: `EventLog`
- Related configuration settings: *Timer*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
ProviderName	Eventlog provider name	String		Microsoft-Windows-Security-Auditing
EventID	Event ID	Number		1001
Level	Event level	Number		3
ProcessID	Process ID	Number		53646
Channel	Eventlog channel name	String		Security
User	User, under which the process is executing	String		NT AUTHORITY\SYSTEM

Field	Description	Data type	Unit	Example
EventData	The content of the -tag in JSON format	String		{“SubjectUserSid” :”S-1-5-18”, “SubjectUserName”:” SYSTEM”, “SubjectDomainName”:”NT AUTHORITY”, “SubjectLogonId” :”0x3e7”, “PrivilegeList”:” SeAssignPrimaryTokenPrivilege SeTcbPrivilege SeSecurityPrivilege”}

Process Stop Metrics

Process Stop

uberAgent collects detailed process stop information like the process name, the process lifetime as well as the parent process.

Notes:

- Field: `AppVersion` - uberAgent has an internal filter to minimize data volume by suppressing version information for system processes and system services. As a result, the `AppVersion` field is typically empty for most system processes and services.

Details

- Source type: `uberAgentESA:Process:ProcessStop`
- Used in dashboards: *Process Tree*
- Enabled through configuration setting: `ProcessStop`
- Related configuration settings: *n/a*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Platform	Example
ProcName	Process name.	String		all	svchost.exe
ProcUser	Process user.	String		all	domain\JohnDoe
ProcLifetimeMs	Process lifetime.	Number	Ms	all	500
AppId	Application ID.	String		all	Svc:WdiSystemHost
ProcId	Process ID.	Number		all	12345
ProcParentId	Parent process ID.	Number		all	67890
SessionId	Session ID.	Number		all	2
ProcGUID	Process GUID.	String		all	4b3e3686-7854-4d98-0023-1e0e617bf2e4
SessionGUID	Session GUID.	String		all	00000000-b242-d759-7a63-d686b0ffd501
ProcParentName	Parent process name.	String		all	services.exe
ProcPath	Process path.	String		all	C:\WINDOWS\System32\sv
ProcCmdline	Process commandline.	String		all	C:\WINDOWS\System32\sv -k LocalSystem-NetworkRestricted
IsElevated	Indicates if the process was started elevated (admin rights).	String		all	1
AppVersion	Application version.	String		all	1.0

Field	Description	Data type	Unit	Platform	Example
ProcParentGUID	Parent process GUID.	String		all	d72ceb7e-7851-02ec-005d-139741c4afd6
IsProtected	Indicates if the process was started protected.	String		Win	1
HashMD5	Process hash value in MD5. Configurable via settings `EnableCalculateHash` and `HashAlgorithm`.	String		Win	7FFE122B109F1B586DEA2E
HashSHA1	Process hash value in SHA1. Configurable via settings `EnableCalculateHash` and `HashAlgorithm`.	String		Win	26DBC241A37881072689CD
HashSHA256	Process hash value in SHA256. Configurable via settings `EnableCalculateHash` and `HashAlgorithm`.	String		Win	95F0FBBAEF289992385985

Field	Description	Data type	Unit	Platform	Example
HashIMP	Import-table hash. Configurable via settings `EnableCalculateHash` and `HashAlgorithm` .	String		Win	188392D5FBCC485811BB54
CdHash	Hash of the code directory of a signed executable. Configurable via setting `EnableCdHash` .	String		macOS	24e4b80198b220e4a0ea87

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
ProcUser	`coalesce ( ProcUserExpanded , ProcUser ).`	String		Domain\JohnDoe	Splunk data model
User	`ProcUser`.	String		Domain\JohnDoe	Splunk data model
TimestampMs	`_time` * 1000.	Number	Ms	1585913547467	Splunk data model

Scheduled Task Metrics

Scheduled Tasks

uberAgent collects detailed scheduled task information like the task name, the author as well as if it has actions.

Details

- Source type: `uberAgentESA:System:ScheduledTasks`
- Used in dashboards: *Scheduled Tasks*
- Enabled through configuration setting: `ScheduledTaskMonitoring`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
TaskEventType	Scheduled task event type. Possible values: `0`, `1`, `2`. See also `TaskEventDisplayName` .	Number		2
TaskFolder	Folder where the scheduled task is stored.	String		\Microsoft\Windows\Flighting\
TaskName	Scheduled task name.	String		RefreshCache

Field	Description	Data type	Unit	Example
TaskUserName	Account that created, changed, or deleted the task. Possible values: `sys`, `lvc`, `nvc` or any other user. The first three will be expanded in the field `TaskPrincipalExpanded` through the lookup [`systemusers`] from the uberAgent UXM Splunk app.	String		AD\JohnDoe
TaskPrincipal	Account that is used when running the task. Possible values: `sys`, `lvc`, `nvc` or any other user. The first three will be expanded in the field `UserNameExpanded` through the lookup [`systemusers`] from the uberAgent UXM Splunk app.	String		sys

Field	Description	Data type	Unit	Example
LogonType	The logon type for the account configured in the field TaskPrincipal. Possible values: 0, 1, 2, 3, 4, 5, 6. See also LogonTypeDisplayName.	Number		5
Elevated	Indicates whether the task is running elevated or not. Possible values: 0, 1.	Number		0
TaskAuthor	Author that created the task. Can be any string and will often be empty.	String		Microsoft Corporation
TaskHidden	Indicates if the task is hidden in the UI or not. Possible values: 0, 1.	String		0
WakeToRun	Indicates if the Task Scheduler will wake the computer when it is time to run the task. Possible values: 0, 1.	String		0

Field	Description	Data type	Unit	Example
HasActions	Indicates if the task has actions. Actions are send separately in the source type `uberAgentESA:System:ScheduledTaskActions`. Possible values: `0`, `1`.	String		1

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
TaskPath	`TaskFolder` + `TaskName`.	String		\Microsoft\Windows\Flighting\OneSettings\	Splunk data model
time	`_time`.	Number		2020-04-06T14:48:01.394+02:00	Splunk data model
UserNameExpanded	coalesce(UserNameExpanded,TaskUserName).	String		SYSTEM	Splunk data model

Field	Description	Data type	Unit	Example	Where available
LogonTypeDisplayName	Expansion for the field `LogonType` based on the lookup `scheduledtasks_logontypes`. Possible values: `Unspecified`: used for non-NT credentials.`UsePassword`: use a password for logging on the user.`ServiceForUser`: the service will log the user on using Service For User (S4U), and the task will run in a non-interactive desktop.`LogonInteractive`: user must already be logged on. The task will be run only in an existing interactive session.`LogonGroup`: group activation. The groupId field specifies the group.`ServiceAccount`	String		ServiceAccount	Splunk data model, Splunk SPL

Field	Description	Data type	Unit	Example	Where available
TaskEventDisplayName	Expansion for the field `TaskEventType` based on the lookup `scheduledtasks_eventtypes`. Possible values: `Created`, `Updated`, `Deleted`.	String		Updated	Splunk data model, Splunk SPL

Scheduled Task Actions

uberAgent collects details about configured actions of scheduled tasks like the action type, the path to the exe as well as mail settings.

Details

- Source type: `uberAgentESA:System:ScheduledTaskActions`
- Used in dashboards: *Scheduled Tasks*
- Enabled through configuration setting: `ScheduledTaskMonitoring`
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
TaskPath	Task path.	String		\Microsoft\Windows\Windows Start
IsDeprecated	Indicates if the task is deprecated or not. Possible values: `0`, `1`.	String		0

Field	Description	Data type	Unit	Example
ActionType	The configured action. Possible values: `0`, `1`, `2`, `3`. See also `ActionTypeDisplayName`.	Number		2
ActionListIndex	Represents the position in the list of configured actions. 1 means that the action is at the top of the list, 2 represents the second position, and so on.	Number		1
ExePath	Path to the executable which is run. Only filled if `ActionTypeDisplayName` is `ExecutableAction`.	String		C:\WINDOWS\system32\sc.exe
ExeArguments	Arguments of the executable which is run. Only filled if `ActionTypeDisplayName` is `ExecutableAction`.	String		start wuauserv

Field	Description	Data type	Unit	Example
ExeWorkingDir	Working dir of the executable which is run. Only filled if `ActionTypeDisplayName` is `ExecutableAction`.	String		C:\WINDOWS\system32
ComClsid	COM action ID. Only filled if `ActionTypeDisplayName` is `ComAction`.	String		b1aebb5d-ead9-4476-b375-9c3ed9f32afc
ComData	COM action data. Only filled if `ActionTypeDisplayName` is `ComAction`.	String		timer
ComBinary	COM action binary. Only filled if `ActionTypeDisplayName` is `ComAction`.	String		%SystemRoot%\System32\spp
ComHandlerDescription	COM action handler description. Only filled if `ActionTypeDisplayName` is `ComAction`.	String		SppSvcRestartTaskHandler Class>
ComRemoteComputer	COM action remote computer. Only filled if `ActionTypeDisplayName` is `ComAction`.	String		

Field	Description	Data type	Unit	Example
ComServiceName	COM action service name. Only filled if `ActionTypeDisplayName` is `ComAction`.	String		wuauserv
AutoElevated	Indicates if the COM action runs auto-elevated. Only filled if `ActionTypeDisplayName` is `ComAction`. Possible values: `0`, `1`.	String		0
EmailBcc	Email Bcc value. Only filled if `ActionTypeDisplayName` is `EmailAction`.	String		johndoe@company.com
EmailCc	Email Cc value. Only filled if `ActionTypeDisplayName` is `EmailAction`.	String		johndoe@company.com
EmailFrom	Email sender. Only filled if `ActionTypeDisplayName` is `EmailAction`.	String		Alerting@company.com
EmailServer	Email server. Only filled if `ActionTypeDisplayName` is `EmailAction`.	String		mail.company.com
EmailSubject	Email subject. Only filled if `ActionTypeDisplayName` is `EmailAction`.	String		Very urgent altert

Field	Description	Data type	Unit	Example
EmailTo	Email recipient. Only filled if `ActionTypeDisplayName` is `EmailAction`.	String		alerts@company.com
MsgTitle	Message title. Only filled if `ActionTypeDisplayName` is `MessageAction`.	String		Some title
MsgContent	Message content. Only filled if `ActionTypeDisplayName` is `MessageAction`.	String		Some content

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
ActionTypeDisplayName	Expansion of the field ActionType based on the lookup scheduledtasks_actiontypes. Possible values: ExecutableAction, ComAction, EmailAction, MessageAction.	String		ExecutableAction	Splunk data model, Splunk SPL
time	_time.	Number		2020-04-06T14:48:01.394+02:00	Splunk data model

Scheduled Task Triggers

uberAgent collects details about configured triggers of scheduled tasks like the trigger type as well as the repetition.

Details

- Source type: uberAgentESA:System:ScheduledTaskTriggers
- Used in dashboards: *Scheduled Tasks*
- Enabled through configuration setting: ScheduledTaskMonitoring
- Related configuration settings: *n/a*
- Supported platform: *Windows*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
TaskPath	Task path.	String		\Microsoft\Windows\Device Information\Device
TriggerId	Identifier for the trigger. Is often empty.	String		NightlyTrigger
TriggerType	Trigger type. Possible values: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12. See also TriggerTypeDisplayName.	Number		1
EventTriggerSubscription	A query string that identifies the event that fires the trigger.	String		`<query id='1'><select path='System'>*[System/Level=2]</select></query>`
EventTriggerNumValues	The number of queries specified on the matching event.	Number		2
TriggerUserId	The ID of the user that fires the trigger (only in State-change trigger and logon trigger).	String		AD\JohnDoe

Field	Description	Data type	Unit	Example
LogonTriggerPossiblyGroup	The ID displayed in the field `TriggerUserId` is possibly the ID of user-group, instead of an individual user. Possible values: `0`, `1`.	String		1
TriggerEnabled	Indicates if the trigger is enabled or not. Possible values: `0`, `1`.	String		1
TriggerStartBoundary	The start date when the trigger is active.	String		2020-04-09 15:41:27.000 +0200
TriggerEndBoundary	The end date after which the trigger is not active anymore.	String		2020-04-11 15:41:27.000 +0200
TriggerRepetitionDuration	For how long the repetition pattern (repetition interval) is repeated, see ISO8601 Durations.	String		PT23H59M
TriggerRepetitionInterval	The repetition pattern (e.g. daily,monthly, etc.), see ISO8601 Durations.	String		PT2H

Field	Description	Data type	Unit	Example
TriggerRepetitionStopAtDurationEnd	Indicates whether a running task is stopped when the repetition pattern duration expires. Possible values: `0`, `1`.	String		0
TriggerListIndex	Represents the position in the list of configured triggers. 1 means that the trigger is at the top of the list, 2 represents the second position, and so on.	Number		1
DayDisplayName	Indicates on which days the trigger runs.	String		Sunday
WeekDisplayName	Indicates on which weeks the trigger runs.	String		First;Second;Third;Fourth
MonthDisplayName	Indicates in which months the trigger runs.	String		Jan;Feb;Mar;Apr;May;Jun;Jul;A
DayOfMonthDisplayName	Indicates on which days of a month the trigger runs.	String		1;15;30
DailyTriggerDaysInterval	The number of days between the subsequent firing of the daily trigger.	Number		2

Field	Description	Data type	Unit	Example
WeeklyTriggerWeeksInterval	The number of weeks between the subsequent firing of the weekly trigger.	Number		3
MonthlyTriggerRunOnLastDayOfMonth	Indicates if the monthly trigger is fired on the last day of the month. Possible values: `0`, `1`.	String		1
MonthlyDowTriggerRunOnLastWeekOfMonth	Indicates if the monthly day-of-week trigger is fired on the last week of the month. Possible values: `0`, `1`.	String		1
StateChangeId	User session state change ID. Only filled if `TriggerTypeDisplayName` is `SessionStateChangeTrigger`. Possible values: `0`, `1`, `2`, `3`, `4`, `7`, `8`. See also `StateChangeDisplayName`.	String		1
WnfTriggerStateName	Windows Notification Facility (WNF) state name. Also see `WnfIdDisplayName`.	String		1192063AA3BC0875

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
StateChangeDisplayName	Expansion of the field `StateChange` based on the lookup `scheduledtasks_sessionstatechanges`. Possible values: `UndefinedStateChange0`, `ConsoleConnect`, `ConsoleDisconnect`, `RemoteConnect`, `RemoteDisconnect`, `UndefinedStateChange1`, `UndefinedStateChange2`, `SessionLock`, `SessionUnlock`.	String		ConsoleConnect	Splunk data model, Splunk SPL

Field	Description	Data type	Unit	Example	Where available
TriggerTypeDisplayName	Expansion of the field TriggerType based on the lookup scheduledtasks_triggertypes. Possible values: EventTrigger, TimeTrigger, DailyTrigger, WeeklyTrigger, MonthlyTrigger, MonthlyDowTrigger, IdleTrigger, RegistrationTrigger, BootTrigger, LogonTrigger, UndefinedTrigger, SessionStateChangeTrigger, CustomTrigger01. Further explanations on these triggers are available in the lookup scheduledtasks_triggertypes.	String		SessionStateChangeTrigger	Splunk data model, Splunk SPL

Field	Description	Data type	Unit	Example	Where available
time	`_time`.	Number		2020-04-06T14:48:01.394+02:00	Splunk data model
WnfIdDisplayName	Expansion of the field `WnfTriggerStateName` based on the lookup `wnf_ids`. Further explanations on the collected WNF IDs are available in the lookup `wnf_ids`.	String		WNF_RTDS_NAMED_PIPE_TRIGGER_CHANG	Splunk data model, Splunk SPL

Security & Compliance Inventory Metrics

Security Inventory

uberAgent periodically runs security inventory tests that check the configuration of operating systems and applications.

Details

- Source type: `uberAgentESA:System:SecurityInventory`
- Used in dashboard: *Security Score*
- Enabled through configuration setting: `SecurityInventory`
- Related configuration settings: `SecurityInventoryTest`
- Supported platform: *Windows, macOS*

List of Fields in the Raw Agent Data

Field	Description	Data type	Unit	Example
SecurityInventoryName	The name of the test.	String		Daily antivirus check
SecurityInventoryCategory	The name of the test category.	String		Antivirus
SecurityInventoryScore	The resulting test score on a scale from `0` (very bad) to `10` (excellent).	Number		7
SecurityInventoryResultData	Configuration information determined by the test.	String		“AntivirusEnabled”=true “AntivirusName” =”Windows Defender”“AntivirusUpToDate” =true
SecurityInventoryRiskScore	The severity of the test (how risky is the tested thing) on a scale from `0` (low risk) to `100` (high risk).	Number		50
SecurityInventoryErrorCode	An error code returned by the test. `0` is interpreted as success.	Number		0
SecurityInventoryErrorMessage	Optional error message returned by the test.	String		PowerShell commandlet not found.
SecurityInventoryScope	The scope of the script. Possible values: `1`, `2`. Also see the field `SecurityInventoryScopeDisplayName` .	Number		1

Field	Description	Data type	Unit	Example
SecurityInventoryScopeUser	The User name if the test was run in the user scope.	String		Domain\JohnDoe

List of Calculated Fields

Field	Description	Data type	Unit	Example	Where available
SecurityInventoryDisplayName	The display name of the test to improve readability.	String		Protected root certificates	Splunk data model, Splunk SPL
SecurityInventoryNameDescription	The description of a test.	String		Checks if root certificates can be installed by users.	Splunk data model, Splunk SPL
SecurityInventoryScopeDisplayName	Scope display name. Possible values: `Machine`, `User`.	String		Machine	Splunk data model, Splunk SPL

Advanced Topics

This section contains information about advanced topics like multi-tenancy or data volume optimization.

Multi-Tenancy With Splunk

If you are a service provider, it is not necessary to set up a dedicated Splunk server for every customer. Splunk fully supports multi-tenancy and uberAgent does, too. This article describes how to configure it.

Concepts

Splunk stores data in indexes. Regarding multi-tenancy, the important thing is that you can set permissions per index.

Setting up a multi-tenant Splunk & uberAgent installation is simple. We are going to follow the concept outlined in the Splunk blog.

Implementation

Actually, all you have to do is described very well in the Splunk blog article mentioned earlier. Here is a summary:

- Create a unique index per customer. Please note that all uberAgent indexes should start with a common prefix (e.g. `uberagent`) so that they can be searched with a single wildcard statement (e.g. `uberagent*`). Valid names would be `uberagent-customer1` and `uberagent-customer2`.
- To set up custom index names follow the instructions in this article.
- Configure users according to the Splunk blogs article.

Result

If you have configured this, you have a setup where Splunk administrators can search everything while individual customer admins can only search their own data.

Automatic Application of Configuration Changes

uberAgent monitors its configuration source for changes and applies updated configurations automatically. This is called Automatic Application of Configuration Changes (AACC).

Configuring AACC

Enabling or Disabling AACC

In the default configuration, Automatic Application of Configuration Changes (AACC) is **enabled**. To **disable** AACC, set the value of both configuration settings **Configuration_Settings** > **ConfigCachePollInterval** and **Configuration_Settings** > **ConfigChangeRestartTimeoutSec** to `0`.

Refresh Interval

Applies to configuration sources: central config file management (CCFM).

uberAgent regularly checks the CCFM file share for updated configurations. The refresh interval is configurable via the setting **Configuration_Settings** > **ConfigCachePollInterval**. To evenly distribute the load on the CCFM file share, each agent adds a randomized offset to the refresh interval (between 0 and 50% of `ConfigCachePollInterval`).

Setting **Configuration_Settings** > **ConfigCachePollInterval** to `0` disables monitoring of remote configuration sources for changes.

Agent Restart Wait Time

Applies to configuration sources: local config files, Group Policy.

uberAgent waits for a specified time before applying an updated configuration. This wait time is configurable via the setting **Configuration_Settings** > **ConfigChangeRestartTimeoutSec**.

Setting **Configuration_Settings** > **ConfigChangeRestartTimeoutSec** to `0` disables monitoring of local configuration sources for changes.

How AACC Works

Configuration Sources

This section lists the locations uberAgent monitors for configuration updates. Please note that new configurations are applied **only** if a meaningful change is detected (see below).

Central Config File Management (CCFM) If uberAgent uses a configuration from a file share via central config file management, it monitors the entire CCFM directory structure for new or modified configurations. uberAgent updates its configuration in the following two cases:

1. **Current location is modified:** uberAgent finds a configuration archive with a changed timestamp in the currently used subdirectory of the CCFM file share. Please note that the timestamp doesn't have to be newer, just different. This makes it possible to roll back a configuration update simply by replacing the newer configuration archive with the previous version.
2. **Better-matching location is available:** uberAgent finds a better-matching configuration in another subdirectory of the CCFM base path. In this case, the archive's timestamp is ignored. See this document for details on what constitutes a better match.

Local Config Files If uberAgent uses local config files, it monitors the local configuration files for changes.

Group Policy If uberAgent is configured via Group Policy, it registers for Group Policy refresh notifications.

Switching the Configuration Source

The configuration source can be switched dynamically between all supported configuration options.

Meaningful Change Detection

When uberAgent detects an updated configuration (see above), it doesn't immediately apply the updated configuration. Instead, it performs two types of checks:

1. It loads the updated configuration in test mode and checks it for **validity**.
2. It then compares the updated with the current configuration for **meaningful changes**.

A meaningful change is an actual change to a configuration setting. Modifying a comment, for example, does not constitute a meaningful change.

Forcing change detection uberAgent monitors for changes only its configuration sources (e.g. configuration files, Group Policy), but not the scripts that are mentioned therein. Should such scripts have changed, one can force a configuration change detection by setting the value of **Configuration_Settings** > **Version** to a random string or changing it from its previous value.

AACC Application Matrix

The following matrix helps to understand in which cases Automatic Application of Configuration Changes applies a configuration update.

Config source	Change event?	Better-matching subdir found?	Meaningful change detected?	uberAgent behavior
Local	yes	n/a	yes	Configuration is applied, the uberAgent service/daemon restarts automatically.
Local	yes	n/a	no	Unchanged.
CCFM	yes	no	yes	Configuration is applied, the uberAgent service/daemon restarts automatically.
CCFM	yes	no	no	Unchanged.
CCFM	no	yes	yes	Configuration is applied, the uberAgent service/daemon restarts automatically.
CCFM	no	yes	no	Unchanged.
Group Policy	yes	n/a	yes	Configuration is applied, the uberAgent service/daemon restarts automatically.
Group Policy	yes	n/a	no	Unchanged.

Depending on the config source, **Change event** means different things:

- **Local:** a file change notification was received from the OS.
- **CCFM:** the timestamp of the configuration archive in the current configuration's directory changed (to a newer or older value).
- **Group Policy:** a Group Policy refresh notification was received.

Reducing the Data Volume

Since Splunk is licensed by daily indexed data volume, it is in every customer's interest to keep the data volume generated by uberAgent as small as possible. uberAgent comes prepared for that by offering two default configurations and many ways for fine-tuning.

Choose Between Detail and Data Volume

Start by choosing either the default configuration, which provides **full detail** and high resolution or the configuration optimized for data volume, which differs from the default in the following ways:

- Process & application performance: information is collected only on the 10-15 **most active processes** in terms of CPU, RAM, disk, and network utilization. The processes included in the data collection are determined dynamically for every collection interval. One could say uberAgent "follows" the active processes.
- **Collection interval** of 120 s instead of 30 s.

See this document for instructions on how to switch between the two configurations.

Take Stock

Before modifying the configuration, find out how much data is generated per endpoint by the default settings. The easiest way to do that is to have uberAgent tell you in the Data Volume dashboard.

Reduce the Data Volume per Endpoint

Once you know the currently generated data volume, you should have an idea by how much it needs to be reduced. Start with the endpoint configuration.

Through uberAgent's configuration you can do three things to reduce the data volume:

Reduce the Frequency

By default, uberAgent collects performance data every 30 seconds. You can cut the volume nearly in half by changing the frequency to one minute (any other value is possible, too, of course).

You can fine-tune the data collection by adding additional timers. The data collection frequency can be set per timer. Move each metric to the timer with the desired frequency to optimally balance accuracy and data volume. While optimizing, focus on those metrics that generate the highest data volume (the *Data Volume* dashboard shows you which those are).

Remove Metrics

By default, all metrics are enabled. If you do not need the information collected by some of them, turn them off by removing them from the configuration.

Special Treatment for ProcessDetail

As you can see in the *Data Volume* dashboard, the ProcessDetail metric generates by far the highest data volume. Consider replacing `ProcessDetailFull` with `ProcessDetailTop5`. Once you do that, uberAgent only collects performance data for processes with the highest activity. This may lead to a dramatic reduction in data volume.

ProcessDetailTop5 By configuring `ProcessDetailTop5`, only the top 5 ProcessDetail metrics are collected based on each of the following criteria:

- Process CPU usage
- Count of process I/O read/write operations
- Amount of process I/O read/write operations data volume
- Process consumed RAM
- Process generated network traffic

Event Data Filtering Event Data Filtering is a powerful feature that replaces the previous allowlist and denylist options. This feature allows defining rules with conditions that are evaluated for every event before it is sent to the backend. With each matching rule, a pre-defined action is executed that controls whether the event is sent to the backend or not. Additionally, certain fields can be cleared before sending the event.

For detailed guidance, refer to the Event Data Filtering documentation.

Reduce the Number of Endpoints

If the data volume is still too high after optimizing the configuration as recommended above you need to reduce the number of endpoints that send data to Splunk. You can simply do that by stopping and disabling the `uberAgent` system service on select endpoints.

Demoing uberAgent With the Event Generator for Splunk

Demonstrating uberAgent can be a bit difficult if you do not have a few dozen machines with live users available. To simplify demos, we offer an event generator that simulates an active environment with various hosts and users.

Architecture

Starting with uberAgent version 6, the Splunk event generator dependency was removed and uberAgent event generator is a single Splunk app. When Splunk is started, a .NET program generates sample data. By default sample data for two hours is generated. If you want to generate additional sample data, you can either restart the Splunk service after 2 hours, or modify the `uAEventGen.conf.json` file (see section "Advanced configuration").

The Splunk app can be used on Windows, Linux, and on macOS-based Splunk installations. Single server setups and distributed deployments are fully supported. The standard installation sends the data to a local Splunk instance using the TCP port 19500.

Installation

.NET 7

As of uberAgent version 7.1, .NET 7.0 is a prerequisite that must be installed on the same server where Splunk is installed. In the case of a distributed environment, .NET 7 must be installed on the same Splunk indexers where you want to install the uberAgent event generator Splunk app.

You can download .NET 7.0 here.

uberAgent Event Generator

Install the uberAgent event generator on one of the indexers. If you have a single Splunk server, install the event generator on that server.

- Download the uberAgent event generator (find out what's new in the changelog)
- On the Splunk server navigate to **Manage apps**
- Click **Install app from file**
- Select the archive you downloaded earlier and click **Upload**
- Restart Splunk

That's it. The event generator starts generating events right after Splunk has been restarted. It will continue to do so for approx. 2 hours and then stop on its own. Just what you need for a demo. To re-enable restart Splunk again.

Configuration

Enabling or Disabling the Event Generator

To enable or disable the uberAgent event generator:

1. On the Splunk server where the uberAgent event generator app is installed navigate to **Manage apps**
2. Locate the *uberAgent event generator* app and click on **enable** or **disable**
3. Restart Splunk

Advanced Configuration

The default configuration should work for a single instance Splunk environment. If you have a distributed Splunk environment or you want to generate different generated sample data, you can modify the configuration file `uAEventGen.conf.json` which is located `%Splunkhome%/etc/apps/uberAgent_eventgenerator/bin/uAEventGenBinaries/your platform`. On a Linux system, for example, this would be: /opt/splunk/etc/apps/uberAgent_eventgenerator/bin/uAEventGenBinaries/Linux
The file contains full documentation of all possible configuration options.

Running Event Generator on macOS ARM

The Eventgen binaries are currently not signed with any certificate. MacOS on an ARM CPU requires a valid certificate otherwise the executable is terminated/killed directly after process startup.
In order to start the event generator on a macOS run the following command:

```
codesign -s "-"/opt/splunk/etc/apps/uberAgent_eventgenerator/bin/
uAEventGenBinaries/macOS/uAEventGen
```

The command adds an ad-hoc certificate to the executed binary.

Recommendations for Custom Dashboards (Splunk)

Why Create Custom Dashboards

uberAgent ships with more than 60 dashboards that visualize every metric uberAgent collects. So why create custom dashboards?

uberAgent's dashboards need to work for all customers. They must be generic and cannot have dependencies on other apps.

The needs of our customers are as diverse as their networks. Custom dashboards meet the most specific requirements. They provide KPIs and views tailored to a customer's needs. And custom dashboards can bring data from multiple sources together: uberAgent + X + Y. That is what Splunk calls operational intelligence.

Where to Create Custom Dashboards

Dashboards need to "live" somewhere. In Splunk, a dashboard's natural habitat is an app. Apps are containers for dashboards and files. Apps allow you to:

- Keep related items in one place
- Add a corporate logo
- Assign permissions
- Publish to Splunkbase

Dashboard Authoring Recommendations

Create Your Own App

At first, people are often tempted to store their custom dashboard files in the uberAgent app directory. That seems to be the easiest way, but it breaks easily when the uberAgent Splunk apps are updated. **We recommend hosting custom dashboards in your own app**, independent of uberAgent.

Do Not Hard-Code Things

Networks are dynamic, as is uberAgent. New versions may bring new sourcetypes, data models, and fields.

uberAgent provides macros to simplify future upgrades. uberAgent's macros can be used globally (e.g., in other apps, too). The use of the following macros is highly recommended:

- `uberAgent_index`: stores the name of the uberAgent Splunk index
- `uA_DM_dataset`: stores the name of the data model that contains a given dataset

All macros are defined in the file `macros.conf` of the uberAgent Splunk dashboard app.

Copy From Us

The best way to get to know the data collected and visualized by uberAgent is to look at the source code of the dashboards that ship with the product. We understand that practical examples can be very helpful and we explicitly recommend to reuse parts of the product's default dashboards in your custom dashboards.

Professional Services

Help and support services for custom dashboard creation are available through our network of solution partners and consultants. Our partners are competent both with uberAgent and with Splunk. They are able to assist you with PoCs, architecture, customizations as well as quotes and pricing.

Custom Scripts

uberAgent can collect data for arbitrary custom metrics through a generic script execution engine. It runs any type of script at any desired interval, either per machine or per user session.

How to Configure Custom Script Execution

The execution of custom scripts is handled by uberAgent's endpoint agent. Scripts are configured as part of `Timer` stanzas in uberAgent's configuration. The following lists the relevant configuration options:

```
#   Setting name: Name
#   Description: Arbitrary name for the timer.
#   Valid values: any string
#   Default: empty
#   Required: yes
#
#   Setting name: Interval
#   Description: How long to wait before collecting data again. Unit: milliseconds.
#   Valid values: any number
#   Default: [none]
#   Required: yes
#
#   Setting name: Script
#   Description: Run a script once or periodically, depending on the configured Interval (0 = run only once). The script's output to stdout is sent to the backend, each line as a new event. Can be specified more than once per timer.
#   Valid values: Any valid command line, optionally including command line parameters.
#   Default: empty
#   Required: no
#
#   Setting name: ScriptContext
#   Description: The user context to run a script in.
#   Valid values:
#       Windows:    Session0AsSystem | UserSessionAsSystem | UserSessionAsUser
#       macOS:      Root | User
#   Default: Windows: Session0AsSystem, macOS: Root
#   Required: no
#
#   Setting name: ScriptTimeout
#   Description: Time in ms until a running script is stopped.
#   Valid values: any number
#   Default: 0 (no timeout)
#   Required: no
```

Please note that uberAgent on the endpoint is running in the context of LocalSystem (Windows)/root (macOS), so the referenced script must be accessible by these accounts. This is particularly relevant when running scripts stored on a network file share.

Script

You can use any script written in your preferred scripting language, e.g., PowerShell, Python, VBScript... On macOS the value of `Script` is passed as argument to `/bin/zsh -c`.

uberAgent captures all script output sent to standard output (`stdout`), i.e., printed to the console. If your script writes output to `stderr`, then the script execution is considered failed and the script's output is discarded. Every line of output is sent to the backend as one event. Script output must be formatted as **key-value** pairs (e.g., `key1=value1 key2=value2`).

The key name may contain characters that are invalid for field names on some backends. Such characters are automatically replaced with underscores (`_`). If field names do not start with a letter, they are prepended with `uA_`. This is done for every type of receiver to ensure that field names are identical on every backend.

Please keep in mind that any data collected in addition to our default dataset has an impact on the generated data volume. Running custom scripts generates additional load on the endpoint the amount of which depends on the executed process (e.g., `powershell.exe` or `cscript.exe`) and the underlying data source. Especially Windows Management Instrumentation (WMI) can cause a significant load.

Additionally, please choose an appropriate timer interval for your script. Data that does not change often, like inventory information, probably only need to be collected once a day whereas volatile metrics like network throughput might have to be collected once per minute.

Deployment uberAgent does not manage the deployment process of custom scripts to the endpoints. Please feel free to use either your existing software distribution system or Splunk's Deployment Server.

An alternative for PowerShell scripts is to encode the script in Base64 and reference the result as parameter.

```
# Place the script content in the variable MyScript
$MyScript = @'
script content goes here
'@

# Convert to base64 and output to a file to avoid word wrapping on the commandline.
[Convert]::ToBase64String([Text.Encoding]::Unicode.GetBytes($MyScript)) > C:\base64_output.txt
```

Add the result as parameter to PowerShell in a timer:

```
[Timer]
Name                  = Some name
Interval              = 30000
Script                = powershell.exe -executionpolicy bypass -EncodedCommand "JABFAHIAcgB ... vAGkAbgAgACcAIAAnACkA"
ScriptContext         = UserSessionAsUser
```

macOS command escaping

Example timer

```
[Timer]
Name                  = Some name
Interval              = 30000
Script                = zsh /Library/Application\ Support/uberAgent/script/myscript.sh
```

Note that on macOS the script command is internally executed via `/bin/zsh -c` and special care must be taken when paths contain spaces. The following examples are correct ways to refer to scripts:

```
# correct strings for the script fields
/Library/Application\ Support/uberAgent/script/myscript.sh
zsh /Library/Application\ Support/uberAgent/script/myscript.sh
zsh "/Library/Application Support/uberAgent/script/myscript.sh"
```

While these commands fail:

```
# incorrect strings for the script fields
"/Library/Application\ Support/uberAgent/script/myscript.sh"
"zsh /Library/Application\ Support/uberAgent/script/myscript.sh"
```

When you test your command on your local machine in a shell, keep in mind that that outer shell interprets your quotes, which is not the case for the command/script path you put in the config.

Script Context

Custom scripts can be executed in different contexts, depending on the operating system.

Windows

- **Session0AsSystem:** the script runs in session 0 as LocalSystem.
- **UserSessionAsSystem:** the script runs in every interactive user session as LocalSystem.
- **UserSessionAsUser:** the script runs in every interactive user session as the user logged on to the session.

macOS

- **User:** specifies the user as which the script is executed. The script is executed once as every logged-in user at the time of the timer interval. This does include users logged in via SSH.
- **Root:** either explicitly set or if no script context is specified, the script is executed once as the root user.

The script context is configured per timer. Of course, you can configure multiple timers for independent execution of different scripts.

Session GUIDs on Windows

To uniquely identify RDS sessions, uberAgent creates its own session GUIDs. These GUIDs are part of all sourcetypes containing data related to user sessions (e.g., logon and logoff).

If the data from a custom script needs to be associated with a specific session, the script should make use of the session's GUID.

Scripts can read session GUIDs from the registry key `HKLM\SOFTWARE\vast limits\uberAgent\SessionGuids`. Each value *name* is the ID of a Windows session. The associated value *data* contains uberAgent's session GUID.

Sourcetype

The **sourcetype** used for the script's output is a concatenation of `uberAgent:Script:` and the timer name specified in uberAgent's configuration.

Examples

For examples that make use of the custom script functionality please see the practice guide section. Another example can be found here.

Data Distribution and Separation (Routing to Multiple Backends)

In some scenarios, it is necessary to distribute the data collected by uberAgent to different backend systems. Do you need to build an even more complex backend infrastructure for that? No, this can be achieved in an easy and straightforward way just by modifying the agent's configuration.

Concept

uberAgent supports multiple data receivers per endpoint by default. Every timer and metric can be bound to one or multiple receivers. That architectural option gives you amazing flexibility out of the box.

The index name can be configured per receiver stanza as well. This makes it possible to configure the data retention per uberAgent metric based on the resulting index in the backend infrastructure.

Implementation

You can make use of this configuration as follows:

- Create a second (, third, ...) `[Receiver]` stanza.
- Bind all `[Timer]` stanzas to the correct receiver based on your requirements.
- Restart the uberAgent service.

Configuration Example

In this example, we are separating uberAgent's inventory data by routing it to an additional Splunk environment.

Configure two `[Receiver]` stanzas like this:

```
[Receiver]
Name     = uberAgent
Type     = Splunk
Protocol = TCP
Index    = uberAgent
Servers  = Server_A:19500

[Receiver]
Name     = uberAgent_Inventory
Type     = Splunk
Protocol = TCP
Index    = uberAgent_Inventory
Servers  = Server_B:19500
```

Bind all `[Timer]` stanzas and the `[OnDemand]` stanza to the default receiver **uberAgent** like this:

```
############################################
# On-demand metrics
############################################
[OnDemand]
Receivers      = uberAgent
UA metric      = LogonDetail
UA metric      = LogonProcesses
UA metric      = BootDetail
UA metric      = ShutdownDetail
UA metric      = StandbyDetail
UA metric      = ProcessStartup
UA metric      = OutlookPerformanceEvents
UA metric      = ApplicationErrors
UA metric      = ApplicationUIDelay
############################################
# Timer 1
############################################
[Timer]
Receivers      = uberAgent
Name           = Default timer
Comment        = Metrics are placed here unless there is a reason to have them run at different frequencies or to isolate them
Interval       = 30000
UA metric      = ProcessDetailFull
UA metric      = ApplicationUsage
UA metric      = SessionCount
UA metric      = SessionDetail
UA metric      = SystemPerformanceSummary
UA metric      = SMBClientSharePerformance
```

Configure only the inventory `[Timer]` stanza (default: Timer 6) to use the `uberAgent_Inventory` receiver:

```
############################################
# Timer 6
############################################
[Timer]
Receivers         = uberAgent_Inventory
Name              = Inventory
Comment           = Perform an inventory at a very low frequency
Interval          = 86400000
Start delay       = 6000
Persist interval  = true
Thread priority   = background
UA metric         = ApplicationInventory
UA metric         = SoftwareUpdateInventory
UA metric         = MachineInventory
UA metric         = CitrixADCInventory
```

Result

As described in this example, you have the ability to granularly route uberAgent's data on a per-metric level to different backend systems. That can be very handy to separate and distribute a subset of data to a specific group or to implement a per-metric data retention policy.

User & Host Tags

uberAgent collects an extensive list of metrics. Pre-built Splunk dashboards allow you to filter the data by computer name, hardware manufacturer, IP address, or user, to name just a few of the available filtering options.

Businesses often store data about a machine's build in the device's registry. User information, on the other hand, is typically managed in Active Directory (e.g., the user's department).

uberAgent's user & host tags feature allows you to integrate custom identifiers in such a way that dashboards can be filtered by their values.

Here is an **example:** your organization stores the version of a machine's OS image in a registry value on each machine. If you configure uberAgent to pick up that image build version through a host tag, you can filter the uberAgent dashboards by the version of your OS images. This allows you to determine if a new issue you are analyzing was introduced with a recent OS image, for example.

Tag Sources

Supported sources for tags are:

- Active Directory attributes
- Environment variables
- Registry values

These sources can be queried either per user or per computer.

Requirements

- uberAgent version 5.3 or later
- Host tags are determined with the *HostTags* metric.
- User tags are determined with the *UserTags* metric.

Configuration

To configure the tags feature you can either use the configuration file or group policy. In this article, we are using the configuration file.

The stanza of interest in the configuration file is `[UserHostTagging]`. You can have multiple of these, one for each tag you want to set. The configurable settings are:

Tag name

- A user-defined unique name of the tag. Spaces are permitted.
- This is the tag you can search for later in the dashboards.
- Example: `Department`

Tag type

- Defines if the tag is a host or a user tag
- Valid values: `Host`, `User`
- Example: `User`

Tag source

- Defines the source where the tag data is read from. Registry values of type `REG_EXPAND_SZ` are automatically expanded.
- Valid values: `Registry`, `Environment`, `Ad`
- Example: `Registry`

Tag value

- The source element/path to read the tag's data from.
- Registry format: any registry path, e.g., `HKCU\Software\vast limits\Department` (supported hives: `HKLM` and `HKCU`)
- Environment variable format: the variable name enclosed in percent signs, e.g., `%DEPARTMENT%`
- AD format: just the attribute name, e.g., `Department`

Additionally, the metric *UserTags* and/or *HostTags* need to be configured in at least one timer.

With the examples from above, you would get the following stanza:

```
[Timer]
Name         = User tags
Comment      = Collect configured user tags
Interval     = 60000
UA metric    = UserTags

[UserHostTagging]
Tag name     = Department
Tag type   = User
Tag source = Registry
Tag value  = HKCU\Software\vast limits\Department
```

Additionally, an event trigger can be defined to collect the user tag for a single user after logon.

```
[Timer]
Name         = User tags
Comment      = Collect configured user tags
Interval     = 60000
UA metric    = UserTags
EventTrigger = TriggerUserTags

[EventTrigger platform=Windows]
Name = TriggerUserTags
Type = UserLogon

[UserHostTagging]
Tag name   = Department
Tag type   = User
Tag source = Registry
Tag value  = HKCU\Software\vast limits\Department
```

Another example: group machines by roles like notebook, graphics workstation, desktop PC, virtual machine, or executive machine with an environment variable.

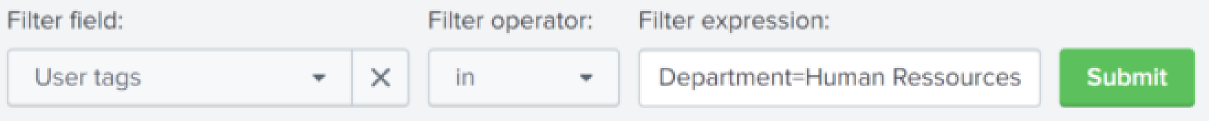
```
[Timer]
Name       = Host tags
Comment    = Collect configured host tags
Interval   = 600000
UA metric  = HostTags

[UserHostTagging]
Tag name   = Machine role
Tag type   = Host
Tag source = Environment
Tag value  = %MACHINEROLE%
```

You can find more examples in our practice guide on the subject.

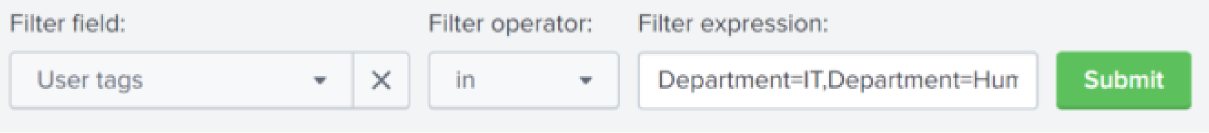
Usage in Splunk

The capability to filter for tags is built into the dashboards since version 5.3. To filter for host tags choose the `Host tags` filter field and to filter for user tags choose the `User tags` filter field.

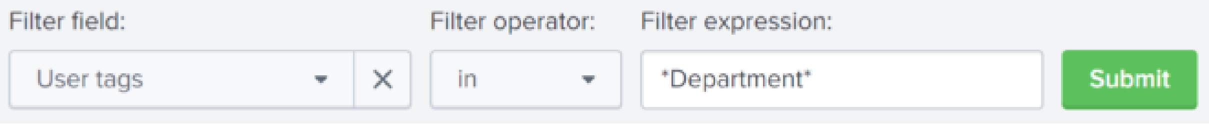
When using our user tag `Department` from above a search for all users from human resources would look like this:


Filter field: User tags
Filter operator: in
Filter expression: Department=Human Ressources

You can also search for users from two or more departments by listing them comma-separated.


Filter field: User tags
Filter operator: in
Filter expression: Department=IT,Department=Hum

Interested in all users where `Department` has a value (is not **`null`**)?


Filter field: User tags
Filter operator: in
Filter expression: *Department*

Note the asterisks around the tag's name. This is needed by design when searching for the tag's name only.

uAQL - uberAgent Query Language

Overview

The **uberAgent Query Language (uAQL)** is the language for uberAgent's powerful Threat Detection and Event Data Filtering queries. Take a look at this blog post for an quick **introduction**.

uAQL query strings evaluate to either **`true`** or **`false`**. A simple example: the query `Process.Name == "explorer.exe"` tests if the current process name is equal to `explorer.exe`.

1. If yes, the query yields **`true`**
2. If no, the query yields **`false`**

Data Types

uAQL is a type-safe language supporting the following data types:

- Integer (a number)
- Boolean (**`true`** or **`false`**)
- String (a UTF-8 string literal, in other words: text)

Integer

A uAQL integer is a 64-bit signed number. Valid values are in the range between `-9223372036854775808` and `9223372036854775807`.

Boolean

A uAQL boolean is either **`true`** or **`false`**. Strings are implicitly converted to **`false`**. Integers with the value 1 are implicitly converted to **`true`**. All other integers are implicitly converted to **`false`**.

String

A uAQL string represents a sequence of characters that are wrapped in double quotes (`"`) or single quotes (`'`), e.g., `"string 1"` or `'string 2'`. Mixing different types of quotes (e.g., `"uberAgent '`) is not allowed and results in a syntax error.

Character Escaping Special characters can be specified by escaping them with a prepended backslash (`\`). This makes it possible to use newlines (`\n`), tabs (`\t`), and many other special characters in queries.

As a result, backslashes in paths, etc. need to be escaped, too. A Windows path such as `C:\Windows` is written in uAQL as `"C:\\Windows"`.

Important: Regular expressions require a second round of escaping. To use the path `C:\Windows` in a regex, specify it as `"C:\\\\Windows"`.

Raw Strings To avoid having to write so many backslashes, consider using raw string literals, in other words, strings with character escaping disabled. Raw string literals start with a leading `r` before the opening quote. The path `C:\Windows` would be specified as a raw string like this: `r"C:\Windows"`.

Important: regular expressions and the operators `like` as well as `glob` still require escaping, even with raw strings. To use the path `C:\Windows` in a regex or with `like` or `glob`, specify it as a raw string literal as `r"C:\\Windows"`.

Character Escaping Examples The following uAQL queries demonstrate different ways of matching files in uberAgent's `Config cache` directory.

Using the `icontains` operator and a regular string:

```
Query = icontains(File.Path, "\\vast limits\\uberAgent\\Config cache\\"
    )
```

Using the `like` operator and a raw string:

```
Query = File.Path like r"%\\vast limits\\uberAgent\\Config cache\\%"
```

Using the `like` operator and a regular string:

```
Query = File.Path like "%\\\\vast limits\\\\uberAgent\\\\Config cache\\\\%"
```

Array

Arrays are special datatypes in uAQL that can hold one or multiple integers, booleans, and strings. An array is defined through square brackets: `[Element 1, Element 2, Element n]`.

Arrays can be used in queries with the `in` operator. The following example tests if the current process name is equal to one of the elements in the array:

```
Process.Name in ["uberAgent.exe", "explorer.exe", "cmd.exe"]
```

Keywords

uAQL comes with keywords for all sorts of logical and binary comparisons.

Operator	Description
`and`, `AND`	Logical AND
`or`, `OR`	Logical OR
`not`, `NOT`	Logical NOT
`==`	Equality. Case-insensitive for strings.
`===`	Equality. Case-sensitive for strings.
`!=`	Inequality. Case-insensitive for strings.
`!==`	Inequality. Case-sensitive for strings.
`<`	Less than.
`<=`	Less than or equal.
`>`	Greater than.
`>=`	Greater than or equal.
`in`, `IN`	Tests if the value on the left-hand side is equal to any element of the array on the right-hand side. Case-sensitive for strings.

Operator	Description
`like`, `LIKE`	Pattern matching comparison as in SQL. The following wildcards can be used: The percent sign (%) matches zero, one, or many characters, including spaces. The underscore (_) matches a single character.
`glob`, `GLOB`	Uses the Unix file globbing syntax for wildcards. Case-sensitive.
`iregex`	Matches the event property against a regular expression. Case-insensitive.
`regex`	Matches the event property against a regular expression. Case-sensitive.
`iregex_envvars`	Unlike the default `regex` this version evaluates paths first. See PATH_REGEX for more details. Case-insensitive. Windows only.
`regex_envvars`	Unlike the default `regex` this version evaluates paths first. See PATH_REGEX for more details. Case-sensitive. Windows only.

Functions

The following functions can be used in uAQL queries:

Function	Description
`strlen(string)`	Returns the length of a string in characters as an integer.
`concat(string, string)`	Concatenates two strings and returns the result.
`lower(string)`	Transforms a string to lowercase and returns the result.
`upper(string)`	Transforms a string to uppercase and returns the result.
`startswith(string, string)`	Returns **`true`** if the first string starts with the second string, otherwise **`false`**. Case-sensitive.
`istartswith(string, string)`	Returns **`true`** if the first string starts with the second string, otherwise **`false`**. Case-insensitive.

Function	Description
`endswith(string, string)`	Returns **`true`** if the first string ends with the second string, otherwise **`false`**. Case-sensitive.
`iendswith(string, string)`	Returns **`true`** if the first string ends with the second string, otherwise **`false`**. Case-insensitive.
`contains(string, string)`	Returns **`true`** if the first string contains the second string, otherwise **`false`**. Case-sensitive.
`icontains(string, string)`	Returns **`true`** if the first string contains the second string, otherwise **`false`**. Case-insensitive.
`join(string array, seperator)`	Returns a concatenated list of all elements from the string array. The separator can contain 0 to n characters. This function is useful when querying results returned by the registry_read_stringmulti function.
`registry_read_binary(hive, key, value)`	Reads a binary value from the registry and returns the data as a hexadecimal string (e.g. 0102FCFF). The hive, key and value parameters are strings.
`registry_read_number(hive, key, value)`	Reads a number from the registry. The hive, key and value parameters are strings.
`registry_read_string(hive, key, value)`	Reads a string from the registry. The hive, key and value parameters are strings.
`registry_read_stringmulti(hive, key, value)`	Reads a multi line string from the registry. The hive, key and value parameters are strings.
`get_env(environment variable name )`	Reads and returns the string value of a given environment variable name.
`set(variable name, value)`	Changes the string value of a given variable. Returns true on success, otherwise false.
`seti(variable name, value)`	Changes the integer value of a given variable. Returns true on success, otherwise false.
`setb(variable name, value)`	Changes the boolean value of a given variable. Returns true on success, otherwise false.
`isnull(variable name)`	Returns **`true`** if the given variable name has a null (undefined) value.

Function	Description
`isnull_or_empty(variable name)`	Returns **true** if the given variable name has a null (undefined) value or if it is an empty string.
`r"string"`	Evaluates a string as raw string, i.e., with character escaping turned off (see above).
`jsonp(variable name, path)`	Accesses values directly from JSON data using a JSON pointer path. Returns the value at the specified path. If the path doesn't exist, returns null.

jsonp Function

The `jsonp` function is a powerful addition to uAQL that allows you to access values directly from JSON data. It uses a JSON pointer path to navigate through the JSON structure and retrieve specific values. This function is particularly useful when working with complex JSON data structures in your queries.

Syntax

```
jsonp(variable name, path)
```

- `variable name`: A variable that contains the JSON data.
- `string path`: The JSON pointer path to the desired value.

Examples Let's consider the following JSON structure:

```
{

 "user": {

   "name": "John Doe",
   "age": 30,
   "active": true
  }
 ,
 "preferences": {

   "theme": "dark",
   "notifications": ["email", "sms"]
  }

 }
```

Here are some examples of how to use the `jsonp` function with this JSON structure:

1. Access a simple property:

   ```
   jsonp(CEB.EventPayload, "/user/name") == "John Doe"
   ```

2. Check a boolean value:

   ```
   jsonp(CEB.EventPayload, "/user/active") == true
   ```

3. Compare a numeric value:

   ```
   jsonp(CEB.EventPayload, "/user/age") > 25
   ```

4. Check if a value exists in an array:

   ```
   "email" in jsonp(CEB.EventPayload, "/preferences/notifications")
   ```

5. Combine with other uAQL functions:

   ```
   lower(jsonp(CEB.EventPayload, "/preferences/theme")) == "dark"
   ```

These examples demonstrate how the `jsonp` function can be used to navigate and extract data from JSON structures in your uAQL queries. Remember that if the specified path doesn't exist in the JSON data, the function will return null.

Limitations While the `jsonp` function is powerful, it does have some limitations:

1. **Nested Objects in Arrays**: The function cannot directly query nested objects within arrays. For example, if you have an array of user objects, you can't directly query for a specific user's property.
2. **Complex Queries**: The function is designed for straightforward JSON navigation. It may not be suitable for complex queries that require filtering or searching within arrays.

Reserved Keywords

All event property names are reserved. Also, operator names including the following keywords are reserved:

Keyword	Description
`true`, `TRUE`, `True`	Boolean **`true`** constant.
`false`, `FALSE`, `False`	Boolean **`false`** constant.

Keyword	Description
null, NULL, Null	Null (undefined) value. E.g. returned by any registry_read_ function which reads data that does not match the expected registry value type.

Multiline Queries

To improve the readability of complex and lengthy queries, consider breaking them down into multiple lines. Use QueryStart to indicate the start of the query and QueryEnd to indicate its end. Inline comments are valid within the multi-line block of the query, but you should keep them to a single line for clarity and ease of maintenance.

Illustrative example:

```
[ThreatDetectionRule]
RuleId = 881834a4-6659-4773-821e-1c151789d873
RuleName = Browser-Starts
EventType = Process.Start
Tag = Browser-Starts-By-Users
QueryStart
   # Comment about the first line
   Process.Name in ["chrome.exe", "firefox.exe"] and
   # Comment about the second line
   Process.IsSigned == true and
   Process.User == "John Doe"
QueryEnd
GenericProperty1 = Process.Name
GenericProperty2 = Process.CommandLine
GenericProperty3 = Process.Path
GenericProperty4 = Process.User
```

Note:
To format an array correctly, ensure that the opening bracket is not the first character of a line. It can be placed at any subsequent position within the same line. For instance:

```
[ThreatDetectionRule]
RuleId = 881834a4-6659-4773-821e-1c151789d873
RuleName = Browser-Starts
EventType = Process.Start
Tag = Browser-Starts-By-Users
QueryStart
   Process.Name in [
   "chrome.exe", "firefox.exe"
   ] and
   Process.IsSigned == true and
   Process.User == "John Doe"
QueryEnd
```

```
GenericProperty1 = Process.Name
```

The following is an example of invalid array formatting:

```
[ThreatDetectionRule]
RuleId = 881834a4-6659-4773-821e-1c151789d873
RuleName = Browser-Starts
EventType = Process.Start
Tag = Browser-Starts-By-Users
QueryStart
   Process.Name in
   ["chrome.exe", "firefox.exe"] and
   Process.IsSigned == true and
   Process.User == "John Doe"
QueryEnd
GenericProperty1 = Process.Name
```

Persistent Output Queue With Intelligent Disk Buffering

Persistent Output Queue with Intelligent Disk Buffering is uberAgent's sophisticated technology that ensures no outgoing events are lost while maintaining a minimal footprint and generating no disk IO at all during normal (online) operations.

How Does the Persistent Output Queue Work

Case A: Backend is Reachable

uberAgent constantly monitors network connectivity to its backend. As long as the backend is reachable and processes new events correctly, the endpoint agent does not store events on disk but sends them to the backend directly, thus avoiding any disk IO on the endpoint.

Case B: Backend is Unreachable

If the backend becomes unreachable, either because network connectivity is lost or because the backend has issues accepting data, the agent writes new events to the Persistent Output Queue's local database. uberAgent periodically checks if connectivity is back up. Once that is the case, it sends all events that were buffered to disk and reverts to the efficient direct sending (see case A).

Persistent Output Queue and Splunk Universal Forwarder

On mobile devices, uberAgent was traditionally coupled with Splunk's Universal Forwarder (UF) due to the UF's persistent queue functionality. With uberAgent's built-in POQ, it's not necessary anymore to deploy the UF just for its disk buffering feature.

Disk Space Requirements

The maximum disk space to be used by the POQ can be limited in uberAgent's configuration (see below). The following table is intended as a rough indicator of disk space requirements (PC with average usage, 10 hours uptime, default uberAgent configuration):

Backend	Disk space per day
Elastic	200 MB
Kafka	410 MB
Microsoft OMS	220 MB
Splunk via HTTP	120 MB
Splunk via TCP	70 MB

Configuration Settings

Persistent Output Queue is configured per receiver. Please note that receivers with a POQ must have a unique name.

Enabling or Disabling the POQ

PersistentOutputQueue Enables or disables the Persistent Output Queue for the current receiver.

If the Persistent Output Queue is disabled, uberAgent reverts to simple memory buffering (see the configuration setting `MaxQueueSizeRamMb`).

- Setting name: `PersistentOutputQueue`
- Valid values: `Off`, `SQLite`
- Default: `SQLite`
- Required: no

Limiting POQ Disk Space Usage

PersistentOutputQueueMaxAgeDays Configures the maximum retention time for POQ entries. Entries older than the specified number of days are removed.

- Setting name: `PersistentOutputQueueMaxAgeDays`
- Valid values: positive numbers (0 = unlimited)
- Default: 120
- Required: no

PersistentOutputQueueMaxSizeMb Configures the maximum disk space in MB for the Persistent Output Queue. The file size is checked after events are stored. It is therefore possible that the configured limit is slightly exceeded.

- Setting name: `PersistentOutputQueueMaxSizeMb`
- Valid values: positive numbers (0 = unlimited)
- Default: 500
- Required: no

POQ events are stored in the format expected by the receiver's backend. Some backends expect the data in a JSON format, while others accept raw data. Especially the JSON format bloats events considerably, which means that the required disk space can be much larger than the raw event data as indexed by Splunk.

POQ Location on Disk

PersistentOutputQueuePathWindows Specifies the POQ storage location on Windows machines. Environment variables are supported.

- Setting name: `PersistentOutputQueuePathWindows`
- Valid values: any valid path
- Default: `%PROGRAMDATA%\vast limits\uberAgent\Output Queue`
- Required: no

When the default path is used, uberAgent sets the permissions on the directory so that only members of the local group `Administrators` and the local system have read/write access. When a custom path is specified, uberAgent does not set permissions on the POQ directory. We highly recommend securing the directory.

PersistentOutputQueuePathMacOS The setting `PersistentOutputQueuePathMacOS` specifies the POQ storage location on Windows machines. Environment variables are *not* supported.

- Setting name: `PersistentOutputQueuePathMacOS`
- Valid values: any valid path
- Default: `/Library/Application Support/uberAgent/Output Queue`
- Required: no

When the default path is used, uberAgent sets the permissions on the directory so that only the root user has read/write access. When a custom path is specified, uberAgent does not set permissions on the POQ directory. We highly recommend securing the directory.

Miscellaneous Settings

PersistentOutputQueueErrorHandling Configures how errors during POQ event processing are handled.

- Setting name: `PersistentOutputQueueErrorHandling`
- Valid values: `IgnoreFile`, `StopReceiver`, `Shutdown`
- Default: `IgnoreFile`
- Required: no

In case of an error (e.g., out of disk space), uberAgent may either stop the current receiver (`StopReceiver`), shut down the agent entirely (`Shutdown`), or ignore the current POQ file (`IgnoreFile`). In case all receivers have stopped, uberAgent shuts itself down.

Advanced Settings

PersistentOutputQueueNextSendAttemptSeconds Configures when to perform a next send attempt in case the backend is not reachable.

- Setting name: `PersistentOutputQueueNextSendAttemptSeconds`
- Valid values: positive numbers (0 = no wait delay)
- Default: 25
- Required: no

PersistentOutputQueueMaxInMemoryBulkOperations Configures how many bulk send operations are kept in memory before storing them in the Persistent Output Queue. A single bulk send operation consists of all events within a run that are generated for an uberAgent metric (e.g., process detail).
A value of 0 (zero) disables the in-memory cache. In this case, all events are stored in the Persistent Output Queue before sending them to the backend. This provides maximum preservation of data at the expense of I/O.

- Setting name: `PersistentOutputQueueMaxInMemoryBulkOperations`
- Valid values: positive numbers (0 = disables in-memory cache)
- Default: 10
- Required: no

Advanced Topics

Data Collection Without Network Connectivity

Receivers with a POQ can be used without a configured backed. In other words: the `Servers` setting can be left empty. This is useful to collect data on a machine without network connectivity, such as in a DMZ. The collected data can then be transferred to a machine with network connectivity simply by copying the POQ files.

Recommendations DMZ machine:

- Configure the receiver's settings `Name`, `Type`, and `Protocol` to match the settings on the machine you later plan to transfer the POQ files to.
- Once sufficient data has been collected, stop uberAgent and copy the POQ files off the machine.

Machine with network connectivity:

- Stop uberAgent.
- Copy the POQ files from the DMZ machine to the local POQ path.
- Start uberAgent. The agent sends the DMZ machine's data off to its backend.

POQ File Data Format

The internal format of the Persistent Output Queue is an SQLite database that is not protected and can be read by SQLite browsers like DB Browser for SQLite.

Event Triggers for Timers

uberAgent timers are configuration items that specify when and how often metrics are collected. Classic uberAgent timers operate on a fixed schedule, collecting data at static time intervals. Event triggers extend classic timers by enabling uberAgent to collect data when a specified event occurs on the endpoint, independent of the time.

Classic timers and event triggers are compatible and can be used together. You can, for example, configure a timer to run a script once when a user has logged on to a new session and continue to run the script periodically for every session on the endpoint. This ensures that information about new sessions is collected quickly after the logon and that information about older sessions is refreshed regularly. This is but one use case for event triggers. Please see below for full-featured examples.

Configuration

Event Trigger Stanza

The stanza [EventTrigger] starts a new event trigger configuration block. Please note that every event trigger must be linked to a [Timer] stanza (see the examples below).

Event triggers are configured using the following configuration settings:

Setting	Description	Valid Values	Required
Name	Specifies the name of an event trigger. This name must be linked in the [Timer] configuration.	Any	Yes
Type	The type of event that triggers the timer. This configuration option may be used more than once per [EventTrigger] stanza if the timer should be triggered by multiple different events.	Any event type from the list below. Example: UserLogon.	Yes
Query	The query rule to limit trigger evaluation using uAQL.	Please refer to the uAQL documentation.	No

Event Types

The following table lists the event types that can be used with the Type field of the [EventTrigger] stanza:

Event Type	Description	Platform
UserLogon	This event type is triggered shortly after a user logs on to the endpoint (on average, after half the duration of the `SessionDetail` timer interval).	Windows
SessionConnectionStateChange	This event type is triggered each time a user's session state changes.	Windows
SessionRoaming	This event type is triggered each time a session is roamed to another endpoint.	Windows

Supported Metrics

Event Triggers for timers support the following metrics:

- UserTags
- CitrixSessionConfig
- Script

Examples

Example 1: Configuration With Metric UserTags

In this example, the metric `UserTags` is collected through a standard timer run at a specific interval. Using the standard timer configuration, uberAgent collects the metric `UserTags` for all users logged on to the endpoint at the execution time. Since this metric supports the usage of event triggers, uberAgent can collect this metric upon the occurrence of a specified event for a single user instead of all logged-in users. This example shows that the metric `UserTags` is collected every 10000 ms and when a `UserLogon` event occurs.

```
[Timer platform=Windows]
Name                  = Determine user tags
Comment               = Determine user tags
Interval              = 10000
UA metric             = UserTags
EventTrigger          = TriggerUserTags

```

```
[EventTrigger platform=Windows]
Name                  = TriggerUserTags
Type                  = UserLogon
```

Please find more examples of configuring User & Host Tags here.

Example 2: Configuration With Metric CitrixSessionConfig

This example shows a standard timer to collect the metric `CitrixSessionConfig`. To collect data for a single user that logs on, disconnects, or continues the session on another endpoint, all event types need to be configured within the `[EventTrigger]` stanza. The event trigger in this example is limited to Citrix sessions only by using the uAQL query `BrokerType == "Citrix"`. Using this limitation ensures that the data collection is only started for Citrix remoting sessions.

```
[Timer platform=Windows]
Name                  = Citrix session configuration details
Comment               = Collect Citrix HDX metrics
Interval              = 900000
UA metric             = CitrixSessionConfig
EventTrigger          = TriggerCitrixSessionConfig

[EventTrigger platform=Windows]
Name                 = TriggerCitrixSessionConfig
Type                 = UserLogon
Type                 = SessionConnectionStateChange
Type                 = SessionRoaming
Query                = BrokerType == "Citrix"
```

Example 3: Configuration With a Custom Script

This example describes a standard timer that executes a custom script at a defined interval. This timer can also be enriched with an event trigger so that the script is started on the occurrence of a particular event. In this configuration example, the script is executed in the script context `UserSessionAsUser`. Therefore, uberAgent executes this script additionally for a user that logs on and collects data only for this user.

```
[Timer platform=Windows]
Name                = CustomScriptTrigger
Comment             = Executes a custom script
Interval            = 10000
Script              = powershell.exe -executionpolicy bypass -file "C:\Program Files\vast limits\uberAgent\scripts\test-script.ps1"
ScriptContext       = UserSessionAsUser
EventTrigger        = TriggerCustomScript

[EventTrigger platform=Windows]
Name                = TriggerCustomScript
```

```
Type            = UserLogon
```

uAQL Properties for Event Triggers

Event Triggers can be limited in evaluation using uAQL. The following table shows the options that are available for a query.

UserLogon, SessionConnectionStateChange and SessionRoaming

The following properties are available if one of the listed events is triggered.

Property name	uAQL Data Type	Description
`SessionId`	Integer	The session id of the associated user.
`BrokerType`	String	The broker type. (`Citrix` or `Microsoft` or `VMware` or `NutanixFrame` or `Unknown` or `None`)
`Current.ConnectionStateId`	Integer	The system's connection state id. This value is from WTS_CONNECTSTATE_CLASS.
`Current.ClientName`	String	Computername of the remoting client.
`Current.ClientIp`	String	IP address of the remoting client.
`Previous.ConnectionStateId`	Integer	Previous system's connection state id. This value is from WTS_CONNECTSTATE_CLASS.
`Previous.ClientName`	String	Previous computer name of the remoting client.
`Previous.ClientIp`	String	Previous IP address of the remoting client.

Troubleshooting

The information in this section helps you troubleshoot uberAgent deployments.

Creating an uberAgent Support Bundle

It may happen that the uberAgent support team asks you to create an uberAgent support bundle. A support bundle is a collection of log files and registry items which helps the support team troubleshoot your case. For Windows there is a PowerShell module available which does the work for you. On macOS we provide a convenient shell script. Learn below how to use it.

Requirements

Windows

- At least Windows 7 or Windows Server 2008 R2
- PowerShell version 5 or above
- Administrative permissions
  - Needed to collect process owners of all running uberAgent processes
- The script execution policy must be set to either `RemoteSigned` or `Unrestricted`. Check the script execution policy setting by executing `Get-ExecutionPolicy`. If the policy is not set to one of the two required values, run PowerShell as an administrator and execute `Set-ExecutionPolicy RemoteSigned -Scope CurrentUser -Confirm`

macOS

- At least macOS Catalina
- Z shell (Zsh)
- Administrative permissions

Download & Installation

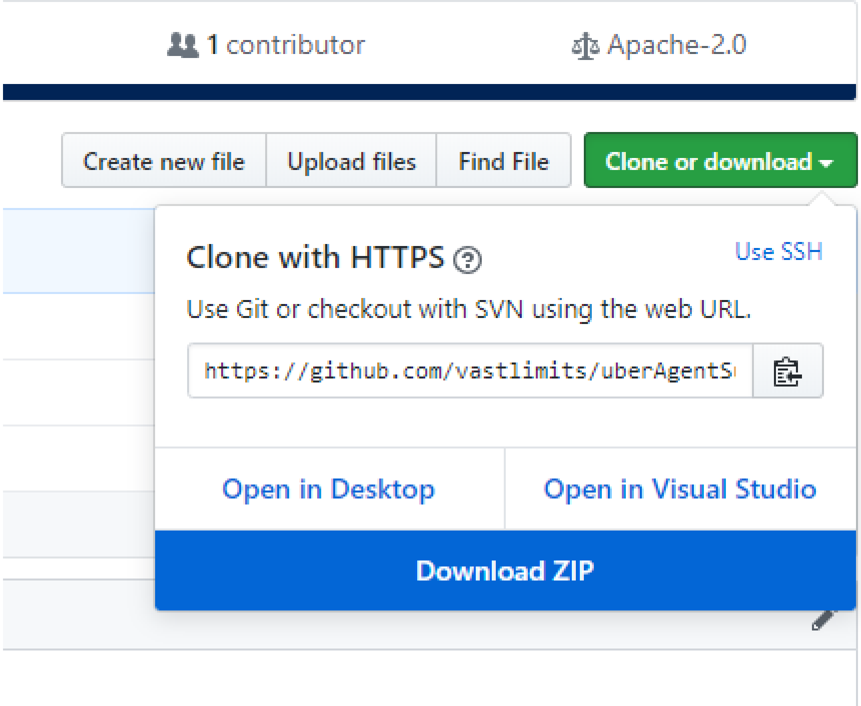
Windows

Via PowerShellGet The uberAgent support module is available on the PowerShell Gallery and can be installed using the PowerShellGet module.

- Open PowerShell and run `Install-Module uberAgentSupport`
- If you see an error like "*The term 'Install-Module' is not recognized as the name...*"install PowerShellGet or update to at least PowerShell version 5.0.
- To update to the latest module version use `Update-Module uberAgentSupport`

Manually

- Download the module from GitHub as a zip file and unzip it somewhere



- Run `Import-Module path-to-module\uberAgentSupport.psd1`
- To update to the latest module version use `Import-Module path-to-module\uberAgentSupport.psd1 -Force` when importing

macOS

- Download the bundle from GitHub as a zip file and unzip it somewhere
- Grant execute permissions for the script.
  - `chmod +x path-to-folder/uberAgentSupport-macOS/uberAgentSupport.zsh`

Usage

Windows

Open an elevated PowerShell and run `New-uASupportBundle`. No parameters required. A zip file will be placed on your desktop.

macOS

Open a Terminal and run `sudo path-to-folder/uberAgentSupport-macOS/uberAgentSupport.zsh`. No parameters required. A zip file will be placed on your desktop.

Deinstallation

Windows

To completely remove the module from your system run

```
$Module = Get-Module uberAgentSupport
Remove-Module $Module.Name
Remove-Item $Module.ModuleBase -Recurse -Force
```

macOS

Just delete the downloaded zip file, extracted folders, and files created on your desktop.

Log Files

Things do not always work the way they should. When that happens, uberAgent does not keep you in the dark. Its log files show you exactly what is going on.

Agent Log

Explanation

This is the log file of uberAgent's main component, the system service/daemon.

Location

Windows The agent log file `uberAgent.log` is stored by default in the SYSTEM account's `Temp` directory, which typically resolves to `C:\Windows\Temp`.

Note: Starting with uberAgent version 7.3 it is possible to change the path via the Windows Registry or Group Policy.

macOS The default location for the `uberAgent.log` file is `/Library/Logs/uberAgent`.

Note: Starting with uberAgent version 6.2 this directory will be owned by root with permissions 700. As a consequence Console.app won't be able to access the log files if it runs as a normal user. If you need to use Console.app to view uberAgent logs in this directory you can either change the permissions or start the app as root from terminal with this command: `sudo /System/Applications/Utilities/Console.app/Contents/MacOS/Console`.

Agent Configuration Log

Explanation

This is the log file of the system service/daemon‘s configuration.

Location

Windows The configuration log file `uberAgentConfiguration.log` is stored by default in the SYSTEM account's `Temp` directory, which typically resolves to `C:\Windows\Temp`.

macOS The default location for the `uberAgentConfiguration.log` file is `/Library/Logs/uberAgent`.

In-Session Helper Log

Explanation

This is the log file of uberAgent's in-session helper component which is used for collecting information from within user sessions.

Location

Windows The in-session helper log file `uAInSessionHelper.log` is stored by default in the SYSTEM account's `Temp` directory, which typically resolves to `C:\Windows\Temp`.

macOS The default location for the `uAInSessionHelper.log` file is `~/Library/Logs/uberAgent`.

Chrome/Edge/Firefox Browser Extension In-Session Helper Log

Explanation

This is the log file of uberAgent's in-session helper instances that are acting as communication gateways between the agent and the Chrome and Firefox browser extensions.

Location

Windows The Chrome/Firefox extension in-session helper log file `uAInSessionHelper.log` is stored by default in the user account's `Temp` directory, which typically resolves to `C:\Users\USERNAME\AppData\Local\Temp`.

macOS The default location for the `uAInSessionHelper.log` file is `~/Library/Logs/uberAgent`.

IE Browser Add-on Log

Explanation

This is the log file of uberAgent's Internet Explorer add-on.

Location

The IE add-on's log file `uberAgentIEExtension.log` is stored by default in the user account' s low-integrity `Temp` directory, which typically resolves to `C:\Users\USERNAME\AppData\Local\Temp\Low`.

If Enhanced Protection Mode is enabled and OS is Windows 8 (or newer), the IE add-on's log file is stored in `C:\Users\USERNAME\AppData\Local\Packages\windows_ie_ac_001\AC\Temp`. For Windows 7 the log files'location is the same as described in the previous paragraph.

Sandbox Log

Explanation

This is the log file of uberAgent's XPC Service that wraps potentially unsafe API calls.

Location

macOS The default location for the `uberAgentSandbox.log` file is `/Library/Logs/uberAgent`.

uAGuardian Log

Explanation

This is the log file for uberAgent's helper process, which is started when the agent service is restarted due to a configuration change to apply the new configuration.

Location

Windows The helper's process log file `uAGuardian.log` is stored by default in the SYSTEM account's `Temp` directory, which typically resolves to `C:\Windows\Temp`.

More Information

Configuring a custom path for log file storage

Windows On Windows systems, you can configure a custom directory for storing uberAgent log files either through Group Policy (GPO) or by modifying the Windows Registry. To achieve this, you need to set a value for the `LogPath` key within the registry path `HKEY_LOCAL_MACHINE\SOFTWARE\vast limits\uberAgent\LogConfig`. The LogPath key should be of type `REG_SZ` and contain the desired directory path.

Example:

```
HKEY_LOCAL_MACHINE\SOFTWARE\vast limits\uberAgent\LogConfig
LogPath = C:\Logs\uberAgent\%COMPUTERNAME%
```

If a custom path is specified, all log files will be stored in the designated location.

Priority Order The priority for determining the log file storage location is as follows:

1. **Group Policy (GPO)**: If configured, this takes the highest priority.
2. **Registry Software Path**: The path specified in HKEY_LOCAL_MACHINE\SOFTWARE\vast limits\uberAgent\LogConfig.
3. **Default Path**: If no GPO or registry path is configured or accessible, the `Temp` directory of the SYSTEM account is used.

Handling Not Accessible Paths If e.g. a network path is configured for log file storage and it becomes not accessible, uberAgent will process the configured paths in order of priority. The logging will then rotate to the next accessible path.

When rotation occurs, a message will be written at the beginning of the log file indicating that the original log file was not accessible and that rotation has taken place. Note that this message may have a later timestamp than subsequent log entries, as the log queue might not have been fully processed at the time of the rotation.

This ensures that logging continues seamlessly even if the initially configured path is not accessible, maintaining the integrity of the log data.

macOS On macOS, you can designate a custom directory for log file storage by modifying the `uberAgent-meta-config.conf` file.
To achieve this, assign a value to the `LogFilePath` key located within the `[Meta:Logging]` section.

Example:

```
[Meta:Logging]
LogFilePath = /tmp/uberAgentLogFiles
```

If a custom path is specified, all aforementioned log files will be stored in the designated location. The files `uberAgent.log`, `uberAgentConfiguration.log`, `uberAgentSandbox.log` and `uberAgentScriptHelper.log` will have the hostname appended to their names. Additionally, the remaining log files will include both the hostname and a session ID in their filenames.

Example:

```
uberAgent_WORKSTATION1.log
uberAgentConfiguration_WORKSTATION1.log
uberAgentSandbox_WORKSTATION1.log
uberAgentScriptHelper_WORKSTATION1.log
uberAgentBrowserHelper_WORKSTATION1_123456.log
uberAgentSessionHelper_WORKSTATION1_123456.log
```

Enabling Debug Mode

Unless debug mode is enabled uberAgent logs only important events like errors. To enable debug mode make sure the following settings are present in the configuration:

```
[Miscellaneous]
debugMode = true
```

Activating Trace Logging

Trace logging is a very detailed log level that can be enabled to facilitate troubleshooting of specific agent components. We recommend only enabling trace logging temporarily.

To enable trace logging for an agent component, add the component's name to the `TraceLogFilterExpressi` regex of the `ConfigFlags` setting, e.g.:

```
ConfigFlags = TraceLogFilterExpression:REGEX
```

The following table lists examples for `REGEX`:

TraceLogFilterExpression regex	Description
.Dns.	Logs additional information for DNS queries.
.POQ create new file.	Logs additional information if a new persistent output queue file was created.
.Event POQ/queue send.	Logs additional information if data was sent to the backend.
.Event POQ increase error count.	Logs additional information if illformed data was sent to the backend and the error count was increased.
.Event POQ remove.	Logs additional information if events were removed from the persistent output queue.
.Event POQ store.	Logs additional information if events were stored in the persistent output queue.
.Event queue store.	Logs additional information if events were stored in an in-memory queue.
.Event POQ read.	Logs additional information if events were read from the persistent output queue.
.Performance counter.	Logs additional information for mapping performance counter names (english to localized and vice versa) and determination times.
.Locking.	Logs additional information if internal locking mechanism for lists took too long.
.StartProcess.	Logs the stdout/stderr content of started scripts.
.SendEventMulti.	Logs additional information if a single send operation was split into multiple.
.Citrix.	Logs additional information for Citrix DC/ADC queries.

TraceLogFilterExpression regex	Description
.*Time-change*.	Logs additional information if system time change was detected.
.*SessionTrace*.	Logs additional information if a user profile event cannot be mapped to an active session.
.*Find*.	Logs additional information if a process cannot be found in uberAgent's internal process list.

File Size and Log Rotation

When the size of the log file grows to 10 MB uberAgent archives it. This is done by appending the current timestamp to the filename and starting a new empty log file. uberAgent keeps the four newest archive files. When four archive files are present and a fifth file is archived the oldest archive file is deleted. This log rotation mechanism guarantees that the total log file size never exceeds 50 MB.

The number of log files to keep around can be changed via the configuration parameter `LogFileCount`.

Log Format

Log file entries always have the same structure, explained in the following table:

Timestamp	Severity	Domain	Thread Owner	Thread ID	Source	Message
Timestamp in the machine's time zone	Possible entries: DEBUG, INFO, WARN, ERROR	The computer's Active Directory domain	**Windows:** the *name* of the computer account **macOS:** the user *root*	The ID of the thread that logged the message	Message source. For example LicenseCheck or ReceiverStatistics	Actual message to be logged

Here is an example:

```
2018-10-04 11:19:51.076 +0100,INFO ,VASTLIMITS,PC1$,4432,ReceiverStatistics,Splunk; localhost:19500 - Events in queue: 11961, queue size: 3073.1 KB, sent: 0, added to queue: 361, rejected from queue: 0

```

```
Timestamp = 2018-10-04 11:19:51.076 +0100
Severity  = INFO
Domain    = VASTLIMITS
Machine   = PC1
Thread ID = 4432
Source    = ReceiverStatistics
Message   = Splunk; localhost:19500 - Events in queue: 11961, queue size: 3073.1 KB, sent: 0, added to queue: 361, rejected from queue: 0
```

Notepad++ Syntax Highlighter

Even though we take great care to optimize the log for readability it is sometimes hard to find the needle in the haystack. That is why we created an uberAgent log syntax highlighter for Notepad++. It highlights the key information, making it easier to find what you are searching for.

Splunk It

As text-based log files, uberAgent's logs are ideal candidates for processing by Splunk. We have built the uberAgent Log Collector specifically for that purpose.

uberAgent Log Collector Splunk App

The endpoint agent's log files are the primary source for agent health and status information. The log files are kept locally on the endpoint, but they can easily be forwarded to the backend with the uberAgent Log Collector Splunk App.

Please refer to the log collector's documentation for details.

uberAgent Browser Extensions

uberAgent's browser web app performance functionality requires a browser extension to be installed. This article lists possible troubleshooting steps.

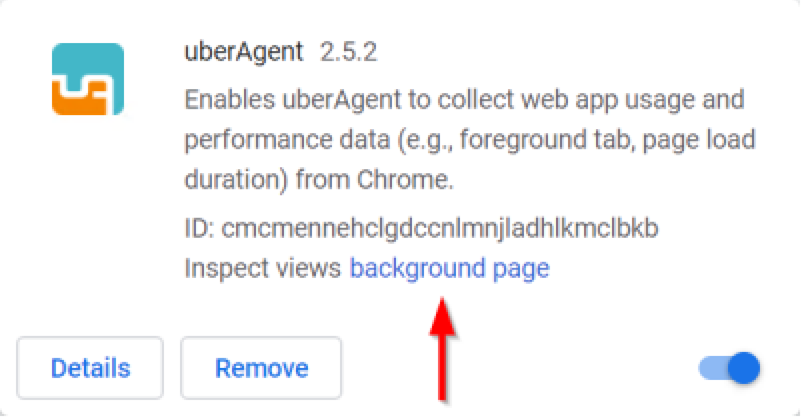
Dashboards That Do Not Require Browser Extensions

uberAgent monitors Chrome and Internet Explorer performance on a process level. The collected data is available in the dashboards *Browser Performance: Chrome* and *Browser Performance: Internet Explorer*, respectively. You do not need any extensions for this to work.

However, if you want data about websites, like page load time, extensions are needed. Website data is visualized in the dashboards *Browser Web App Performance* and *Browser Web App Usage* for all supported browsers. Grouping and filtering by browser type or browser version are possible, of course.

Troubleshooting the Chrome Extension

1. Reread the Chrome extension installation guide carefully. Take notice of the relevant configuration settings section.
2. Check the uberAgent service log as well as the Chrome extension logs for any errors.
3. uberAgent is using native messaging for communication with the extension. Make sure that the extension is not on the native messaging denylist.
   a) If the denylist is enforced in the company, adding `com.vastlimits.uainsessionhelper` to the allowlist overrides the denylist.
4. Check if the native messaging host is configured correctly:
   a) On an x86 machine running Windows, the following registry item is created by the uberAgent service and must be present.
   Key: `HKEY_LOCAL_MACHINE\SOFTWARE\Google\Chrome\NativeMessagingHosts\com.vastlimits.uainsessionhelper`
   Value: `(Default)`
   Data: `C:\Program Files\vast limits\uberAgent\uAInSessionHelperChrome.json`
   b) On an x64 machine running Windows, the following registry item is created by the uberAgent service and must be present.
   Key: `HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Google\Chrome\NativeMessagingHosts\com.vastlimits.uainsessionhelper`
   Value: `(Default)`
   Data: `C:\Program Files\vast limits\uberAgent\uAInSessionHelperChrome.json`
   c) On macOS, a file called `com.vastlimits.uainsessionhelper.json` is created and must be present in the directory `/Library/Google/Chrome/NativeMessagingHosts/`.
5. Enable console logging
   a) Navigate to: `chrome://extensions/`
   b) Enable developer mode
   c) Click the extension's background page link



uberAgent 2.5.2

Enables uberAgent to collect web app usage and performance data (e.g., foreground tab, page load duration) from Chrome.

ID: cmcmennehclgdccnlmnjladhlkmclbkb

Inspect views background page

a) The developer tools appear

b) In the dev tools console enter:

```
logSeverity = LogSeverities.Info
logContext = LogContexts.All
```

6. The extension now logs events to the dev tools console

Troubleshooting the Firefox Extension

1. Reread the Firefox extension installation guide carefully.
2. Check the uberAgent service log as well as the Firefox extension logs for any errors.
3. Check if the native messaging host is configured correctly:

   a) On an x86 machine running Windows, the following registry item is created by the uberAgent service and must be present.
   Key: `HKEY_LOCAL_MACHINE\SOFTWARE\Mozilla\NativeMessagingHosts\com.vastlimits.uainsessionhelper`
   Value: `(Default)`
   Data: `C:\Program Files\vast limits\uberAgent\uAInSessionHelperFirefox.json`

   b) On an x64 machine running Windows, the following registry items are created by the uberAgent service and must be present.

   c) Key: `HKEY_LOCAL_MACHINE\SOFTWARE\Mozilla\NativeMessagingHosts\com.vastlimits.uainsessionhelper`
   Value: `(Default)`
   Data: `C:\Program Files\vast limits\uberAgent\uAInSessionHelperFirefox.json`

   d) Key: `HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Mozilla\NativeMessagingHost` `\com.vastlimits.uainsessionhelper`
      Value: `(Default)`
      Data: `C:\Program Files\vast limits\uberAgent\uAInSessionHelperFirefox.json`

   e) On macOS, a file called `com.vastlimits.uainsessionhelper.json` is created and must be present in the directory `/Library/Application Support/Mozilla/NativeMessagingHosts/`.

4. Enable console logging

   a) Navigate to: `about:debugging#/runtime/this-firefox`
   b) Identify the uberAgent extension and click **Inspect**
   c) The developer tools appear
   d) In the dev tools console enter:

```
logSeverity = LogSeverities.Info
logContext = LogContexts.All
```

5. The extension now logs events to the dev tools console

Troubleshooting the Internet Explorer Extension

1. Reread the Internet Explorer installation guide carefully. Take notice of the relevant configuration settings section.

2. Check the uberAgent service log as well as the IE add-on logs for any errors.

3. Check if the IE add-on is registered correctly

   a) On an x86 machine, the following registry items must be present.

   b) Key: `HKEY_CLASSES_ROOT\CLSID\\{ 82004312-5B53-46F1-B179-4FCE28048E6F } \InProcServer32`
      Value: `(Default)`
      Data: `C:\Program Files\vast limits\uberAgent\uaIEExtension32.dll`

   c) Key:`HKEY_CLASSES_ROOT\CLSID\\{ 82004312-5B53-46F1-B179-4FCE28048E6F } \InProcServer32`
      Value: `ThreadingModel`
      Data: `Apartment`

   d) On an x64 machine, the following registry items must be present.

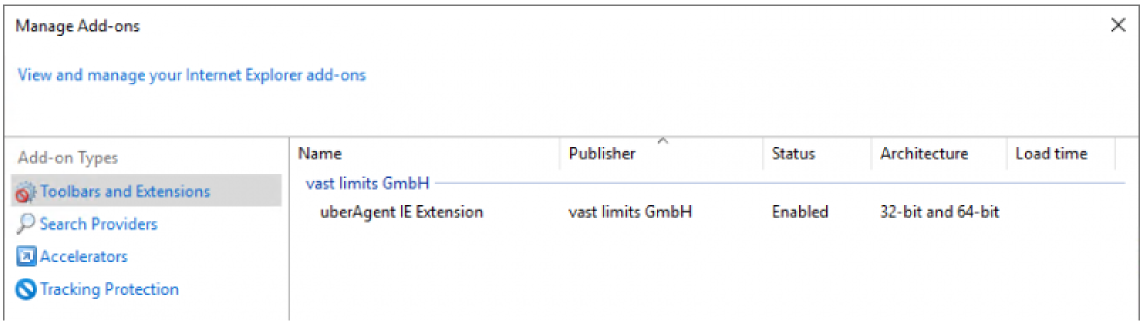
e) Key: `HKEY_CLASSES_ROOT\CLSID\\{ 82004312-5B53-46F1-B179-4FCE28048E6F } \InProcServer32`
Value: `(Default)`
Data: `C:\Program Files\vast limits\uberAgent\uaIEExtension64.dll`

f) Key: `HKEY_CLASSES_ROOT\CLSID\\{ 82004312-5B53-46F1-B179-4FCE28048E6F } \InProcServer32`
Value: `ThreadingModel`
Data: `Apartment`

g) Key: `HKEY_CLASSES_ROOT\WOW6432Node\CLSID\\{ 82004312-5B53-46F1-B179-4FCE28048E6F } \InProcServer32`
Value: `(Default)`
Data: `C:\Program Files\vast limits\uberAgent\uaIEExtension32.dll`

h) Key: `HKEY_CLASSES_ROOT\WOW6432Node\CLSID\\{ 82004312-5B53-46F1-B179-4FCE28048E6F } \InProcServer32`
Value: `ThreadingModel`
Data: `Apartment`

4. Check if the IE add-on is enabled in *Settings* > *Manage add-ons*



Name	Publisher	Status	Architecture	Load time
vast limits GmbH				
uberAgent IE Extension	vast limits GmbH	Enabled	32-bit and 64-bit	

5. uberAgent identifies IE processes by the full path to the EXE file. For this, the path must correspond to one of the following paths entered in the registry.

a) Key: `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\IEXPLORE.EXE`
Value: `(Default)`
Data (example): `C:\Program Files\Internet Explorer\IEXPLORE.EXE`

b) Key: `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\Main`
Value: `x86AppPath`
Data (example for a x86 machine): `C:\Program Files\Internet Explorer\`

`IEXPLORE.EXE`
Data (example for a x64 machine): `C:\Program Files (x86)\Internet Explorer\IEXPLORE.EXE`

Troubleshooting Specific Fields

Please see the following articles for additional information on specific fields:

- Reasons For Empty SessionFgBrowserActiveTabHost Field

Creating an uberAgent Performance Recording

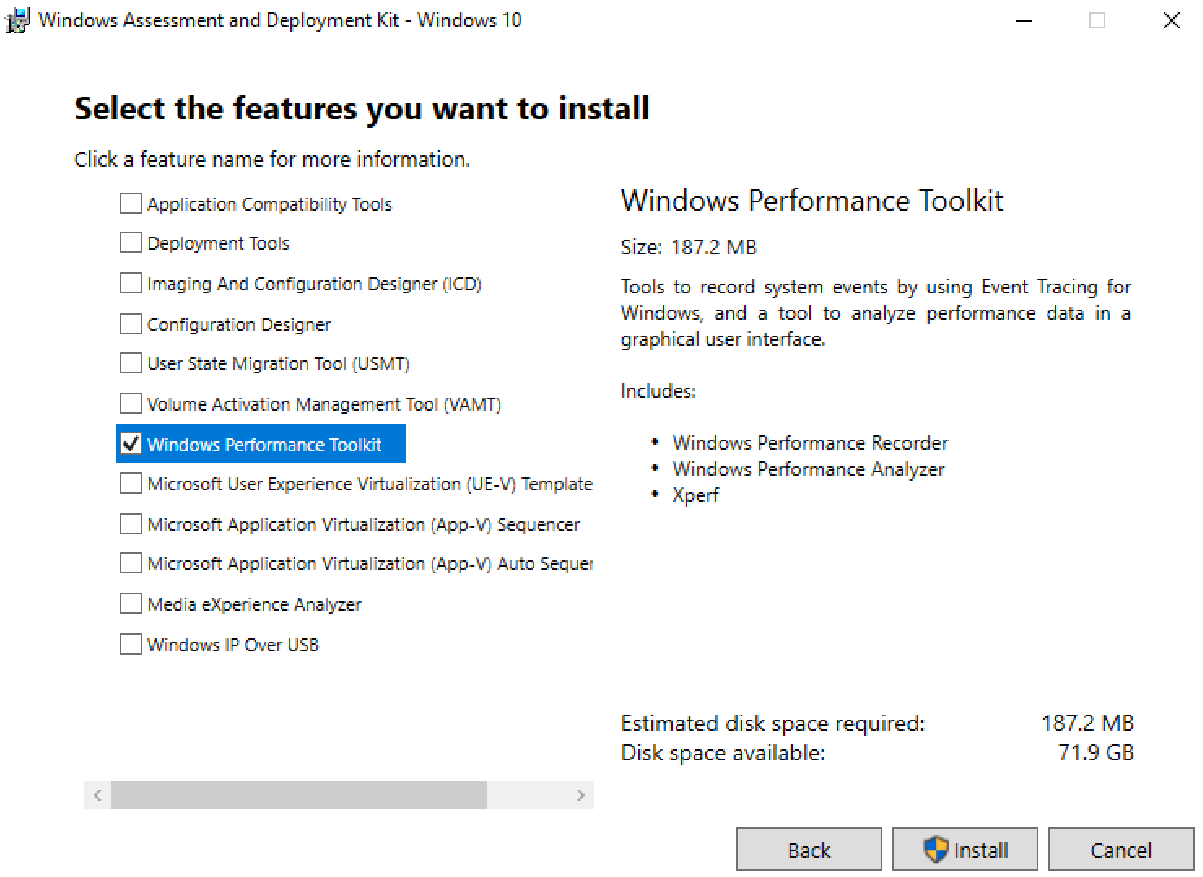
It may happen that the uberAgent support team asks you to create an uberAgent performance recording. A performance recording is a detailed analysis of the processes running on a machine. The recording is created with the Windows Performance Recorder.

Downloading and Installing the Windows Performance Recorder

The Windows Performance Recorder is part of the Windows Assessment and Deployment Kit (Windows ADK).

Download the appropriate version of the ADK for your Windows operating system from Microsoft.

Run the setup. When asked for features to install, select **Windows Performance Toolkit**. The other features may be deselected.



Windows Assessment and Deployment Kit - Windows 10

Select the features you want to install

Click a feature name for more information.

- □ Application Compatibility Tools
- □ Deployment Tools
- □ Imaging And Configuration Designer (ICD)
- □ Configuration Designer
- □ User State Migration Tool (USMT)
- □ Volume Activation Management Tool (VAMT)
- ☑ Windows Performance Toolkit
- □ Microsoft User Experience Virtualization (UE-V) Template
- □ Microsoft Application Virtualization (App-V) Sequencer
- □ Microsoft Application Virtualization (App-V) Auto Sequer
- □ Media eXperience Analyzer
- □ Windows IP Over USB

Windows Performance Toolkit

Size: 187.2 MB

Tools to record system events by using Event Tracing for Windows, and a tool to analyze performance data in a graphical user interface.

Includes:

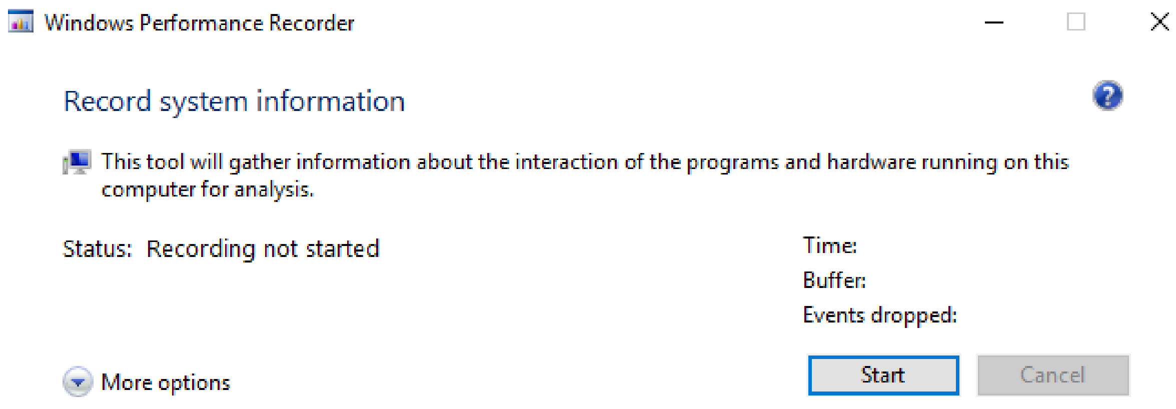
- Windows Performance Recorder
- Windows Performance Analyzer
- Xperf

Estimated disk space required: 187.2 MB
Disk space available: 71.9 GB

Creating The Recording

After the installation has finished, start the **Windows Performance Recorder** from the start menu.

Click on **Start** to start the recording and let the recording run for five minutes, if not told otherwise by uberAgent support.



Windows Performance Recorder

Record system information

This tool will gather information about the interaction of the programs and hardware running on this computer for analysis.

Status: Recording not started

Time:
Buffer:
Events dropped:

When the recording has finished, save the recording on disk and send the generated ETL file to the uberAgent support.

uberAgent's Driver Safety Net Feature

To collect high-quality network and process metrics, uberAgent relies on kernel-mode drivers that are started when the computer boots. While we invested a lot of time and energy to make our drivers as reliable as humanly possible, a driver is just a piece of software, and even software developed to the highest standards cannot be tested in all of the millions of possible combinations of hardware and software. One of the worst possible situations is a driver that crashes during the machine startup phase, more specifically during *every* machine startup. In such a bluescreen loop situation it can be difficult to restore the operating system to working order.

uberAgent's drivers come with **Driver Safety Net**, a unique feature that prevents bluescreen loops: in the unlikely event of a crash in one of uberAgent's drivers, the driver disables itself.

How Driver Safety Net Detects Crashes

In a nutshell, Driver Safety Net works as follows: uberAgent's drivers write their current states to a location in the registry. The information from these registry values is used during the driver initialization phase. When a driver determines that its last start was unsuccessful it increments a failure count value. Once that count reaches a threshold of 3 (three), the driver disables itself.

False Positives

There are events that "look and feel" like a blue screen but have very different root causes, for example, power outages or hardware failures. If multiple such events occur in a short timeframe, uberAgent's Driver Safety Net may be triggered, even though no driver crashed.

How to Re-Enable uberAgent's Drivers After Driver Safety Net Triggered

Once Driver Safety Net triggers, uberAgent's drivers remain disabled until re-enabled with the simple procedure outlined below. This is to ensure maximum system stability and reliability.

To re-enable uberAgent's drivers remove the following registry values and restart the `uberAgent` service:

- `DrvFailureCount`
- `DrvStartupFinished`
- `DrvShutdownFinished`
- `DrvLastStart`

Depending on the driver that triggered Driver Safety Net, the above registry values are located in one of the following two registry paths:

- **Process** driver: `HKLM\SYSTEM\CurrentControlSet\Services\UberAgentDrv`
- **Network** driver: `HKLM\SYSTEM\CurrentControlSet\Services\uberAgentNetMon`

Knowledge Base

Welcome to the uberAgent knowledge base. Here you'll find best practices, explanations as well as tips & tricks.

Architecture

This section contains knowledge base articles about the architecture.

Description of the uAInSessionHelper/uberAgentHelper Process

Some information is only available to processes running in a user's session, not to services running in session 0 respectively in the context of the system. In order to be able to collect such data, too, uberAgent makes use of the helper process `uAInSessionHelper.exe` on Windows and `uberAgentHelper` on macOS.

The helper is a lightweight process that collects specific data and sends it to the main service.

The Helper Process

Windows

There is one `uAInSessionHelper.exe` process running permanently for every user session in the local SYSTEM context, e.g. to collect GPU usage information. When collecting web app metrics from Chrome, Edge, or Firefox, a dedicated `uAInSessionHelper.exe` instance is started (with user privileges) by each browser that has uberAgent's extension installed.

macOS

There is one `uberAgentHelper` process running permanently for every user session with user privileges. When collecting web app metrics from Chrome, Edge, or Firefox, a dedicated `uberAgentHelper` instance is started (with user privileges) by each browser that has uberAgent's extension installed.

Helper Metrics

The helper process is utilized to collect the following metrics:

Windows

- Information about a **UWP app**
- **Internet Explorer performance** data
  Metric: `BrowserPerformanceIE`
- UX metrics for SaaS and web apps in **Internet Explorer**
  Metric: `BrowserPerformanceIE`
- UX metrics for SaaS and web apps in **Chrome**
  Metric: `BrowserPerformanceChrome`
- UX metrics for SaaS and web apps in **Firefox**
  Metric: `BrowserPerformanceFirefox`
- UX metrics for SaaS and web apps in **Edge**
  Metric: `BrowserPerformanceEdge`
- **GPU performance** data
  Metric: `GpuUsage`

macOS

- **Session Detail** data
  Metric: `SessionDetail`
- UX metrics for SaaS and web apps in **Chrome**
  Metric: `BrowserPerformanceChrome`
- UX metrics for SaaS and web apps in **Firefox**
  Metric: `BrowserPerformanceFirefox`
- UX metrics for SaaS and web apps in **Edge**
  Metric: `BrowserPerformanceEdge`

Troubleshooting the Helper

Helper log locations are documented here.

Please review the contents of these log files when you suspect things are not working correctly.

Common Errors

Class Not Registered (Windows)

You may see messages like the following in uAInSessionHelper's log file:

```
2015-09-04 22:16:48.140 +0200, ERROR, HK, XA1$, 1856, IE Performance,
    Action 2 failed with: -2147221164
2015-09-04 22:16:48.142 +0200, ERROR, HK, XA1$, 1856,
    GetBrowserPerformanceIE, Failed (2) with: Class not registered
```

The root cause for these errors is that uberAgent can only determine IE performance per website if IE is run in desktop mode, i.e. not as a Citrix XenApp published application. If IE is run as a published application certain classes required by uAInSessionHelper are not available, thus the messages.

Name or Version may be Inconsistent Between App Inventory & Usage

uberAgent collects information about applications on multiple levels:

- Which applications are installed (app **inventory**)
- Which applications are running (app **usage**)
- Which applications are being worked with (**foreground** app)

Not all of that information is available in one data source. In fact, uberAgent has to query quite disparate sources to collect its extensive application information. That can lead to inconsistencies, specifically between the app inventory and all the other application source types.

Root Cause

The root of the problem is that Windows has no generic concept of applications. Windows knows and cares about processes, threads and a good number of other things, but APIs for applications? Nonexistent.

As a consequence, there are no application naming and versioning standards for software vendors. This may lead to the situation that vendors use one name/version in one place and a different name/version in another - for the same application.

uberAgent

uberAgent has sophisticated algorithms to deal with the situation and determine the best possible application names and versions. However, there may be inconsistencies between the source types `ApplicationInventory` and other application source types, e.g., `ApplicationUsage` or `ProcessDetail`.

The reason for these inconsistencies lies in the fact that the data for the `ApplicationInventory` sourcetype is based on the `Uninstall` registry key so that it matches what Windows displays in *Apps & features*.

All other source types'application info, on the other hand, is based on an algorithm that determines application properties depending on the application type (Win32, UWP, App-V, Java, …). Naturally, this algorithm cannot use the `Uninstall` registry key.

For that reason, inconsistencies may exist and mapping between the `ApplicationInventory` and other application source types is not recommended.

Using uberAgent With a Proxy

uberAgent uses libcurl to send data when using HTTP or HTTPS as protocol. libcurl ignores common per-system or per-user proxy settings. Customers who want to use uberAgent with a proxy, must configure the following two system environment variables:

- `http_proxy`
- `https_proxy`

Customers who have a proxy configured but don't want to route the traffic from uberAgent to Splunk through it, must set the environment variable `no_proxy` to the Splunk server name.

More information is available in the libcurl documentation.

uberAgent vs. Splunk Template for XenDesktop 7

Both uberAgent and the Splunk Template for XenDesktop 7 can be used to monitor a Citrix XenDesktop/XenApp environment. This article highlights the key differences between both products.

Scope

The Splunk Template for XenDesktop 7 monitors part of the XenDesktop infrastructure (e.g. license server) and collects performance data per session and per machine on the XenApp session hosts.

uberAgent monitors user experience and application performance of Windows machines.

Support and Maintenance

Splunk does not provide support for its App for Citrix XenDesktop. New versions with bugfixes and additional features are rare.

uberAgent is fully supported. New versions are being released every 2-3 months. Fixes for critical bugs are made available immediately.

Footprint

The Splunk Template for XenDesktop 7 collects data via PowerShell scripts and WMI.

uberAgent, on the other hand, is written in C++ and accesses native APIs (no WMI). Therefore uberAgent has a much smaller footprint and consumes fewer resources on the monitored system.

Data volume

The Splunk Template for XenDesktop 7 generates a higher data volume per XenApp server than uberAgent.

Data collection

The Splunk Template for XenDesktop 7 collects the following data:

- High-level overviews supporting multiple farms
- Alerting
- ICA latency reporting
- User experience investigation
- User logon time details
- Performance visualization and monitoring
- Application usage
- Critical service monitoring

uberAgent collects the data about the following:

- Logon duration
- Computer startup duration
- Machine performance
- Session performance
- Process performance
- Application performance
- Application usage

- Application versions
- Process startup duration
- GPU usage
- Browser performance per website

Here is the full list of metrics collected by uberAgent.

Configuration

This section contains knowledge base articles about the configuration.

Metric Dependencies

Use one process detail metric only

You can either use `ProcessDetailFull` OR `ProcessDetailTop5`, not both. If you have accidentally configured both, uberAgent uses `ProcessDetailTop5`.

There is no entry in the log for this case.

Use a process detail metric in one timer only

`ProcessDetailTop5` or `ProcessDetailFull` can be configured in one timer only.

If you have it configured in two or more timers, you get the following warning in the `uberAgentConfiguration.log`:

```
CheckForUnconfiguredMetrics,Metric(s) ProcessDetailFull can only be configured once.
```

Use session and process detail metrics in the same timer only

`SessionDetail` and `ProcessDetailTop5` OR `ProcessDetailFull` must be configured in the same timer. If you have accidentally configured these metrics in different timers, you get the following warning in the `uberAgentConfiguration.log`:

```
CheckSessionAndProcessInSameTimer,Metric(s) SessionDetail and (ProcessDetailFull or ProcessDetailTop5) should only be configured in the same timer.
```

Session details require logon details

If you enable `SessionDetail`, you must also enable the metric `LogonDetail`.

Custom Queries

This section contains knowledge base articles about custom queries.

Walkthrough: Building a Splunk Search

The following walkthroughs illustrate how to build a Splunk search from scratch.

Scenario: we want to identify users who launch a specific executable more often than `n` times in a given time range.

Splunk SPL Search

Let's build the Splunk SPL search step by step.

Step 1

We start with all events from uberAgent's index.

Note: to facilitate changing the index name used by uberAgent, all our dashboards make use of the `uberAgent_index` macro which contains the actual index name. The macro is defined in `macros.conf` of the uberAgent searchhead app.

```
index=`uberAgent_index`
```

Step 2

We filter for the process startup sourcetype which contains one event per started process.

Note: The documentation of uberAgent's sourcetypes and fields can be found here.

```
index=`uberAgent_index` sourcetype=uberAgent:Process:ProcessStartup
```

Step 3

We ignore processes started by SYSTEM, LOCAL SERVICE and NETWORK SERVICE.

Note: The pseudo-users `sys`, `lvc` and `nvc` are defined in the lookup table `systemusers.csv` of the uberAgent searchhead app. They are auto-expanded to the proper user names SYSTEM, LOCAL SERVICE and NETWORK SERVICE in uberAgent's data model.

```
index=`uberAgent_index` sourcetype=uberAgent:Process:ProcessStartup
    ProcUser!=sys ProcUser!=lvc ProcUser!=nvc
```

Step 4

We add a filter for the name of the process we are interested in, `Winword.exe` in this example.

```
index=`uberAgent_index` sourcetype=uberAgent:Process:ProcessStartup
    ProcUser!=sys ProcUser!=lvc ProcUser!=nvc ProcName=Winword.exe
```

Step 5

We count the number of (start) events per user.

Note: The only purpose of adding the field `ProcName` to the `stats` command is to make it part of the results table, too.

```
index=`uberAgent_index` sourcetype=uberAgent:Process:ProcessStartup
    ProcUser!=sys ProcUser!=lvc ProcUser!=nvc ProcName=Winword.exe
| stats count as Starts by ProcName ProcUser
```

Step 6

We only keep users with more than five starts in the results list.

```
index=`uberAgent_index` sourcetype=uberAgent:Process:ProcessStartup
    ProcUser!=sys ProcUser!=lvc ProcUser!=nvc ProcName=Winword.exe
| stats count as Starts by ProcName ProcUser
| where Starts > 5
```

Step 7

We rename fields to make them look nicer.

```
index=`uberAgent_index` sourcetype=uberAgent:Process:ProcessStartup
    ProcUser!=sys ProcUser!=lvc ProcUser!=nvc ProcName=Winword.exe
```

```
| stats count as Starts by ProcName ProcUser
| where Starts > 5
| rename ProcUser as User ProcName as Process
```

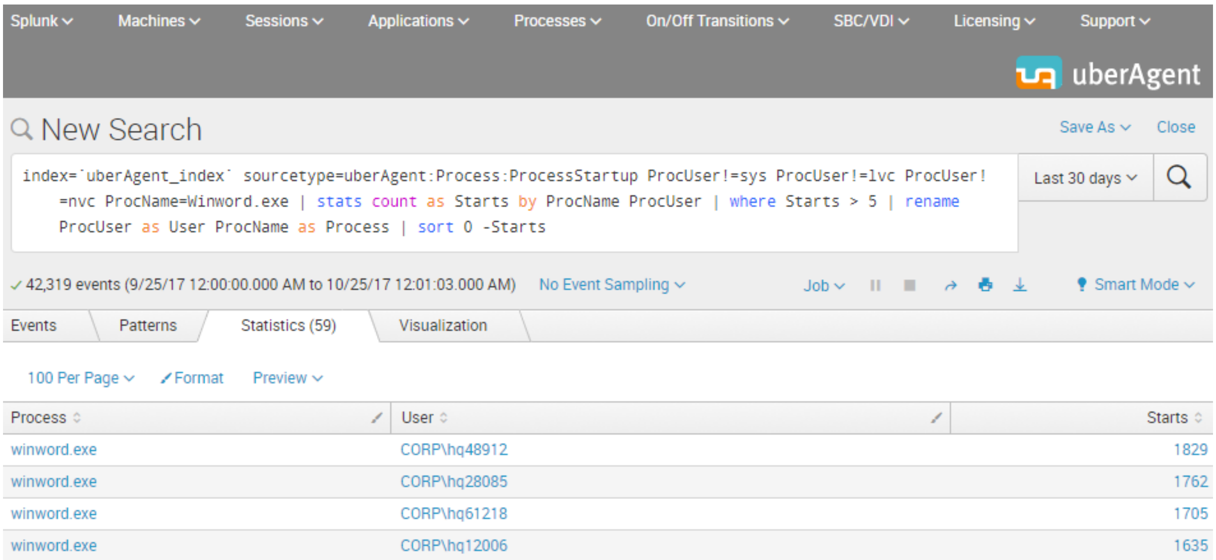
Step 8

We sort the results so that the user with the highest number of starts is listed first. The `0` in the `sort` command ensures that the output is not truncated after the 10,000th result.

```
index=`uberAgent_index` sourcetype=uberAgent:Process:ProcessStartup
    ProcUser!=sys ProcUser!=lvc ProcUser!=nvc ProcName=Winword.exe
| stats count as Starts by ProcName ProcUser
| where Starts > 5
| rename ProcUser as User ProcName as Process
| sort 0 -Starts
```

The Result

This is what the final Splunk SPL search for users with more than five starts of `Winword.exe` looks like. The screenshot below shows the search being run over the past 30 days. In practice, you would adjust the time range to any relevant time interval.



```
index=`uberAgent_index` sourcetype=uberAgent:Process:ProcessStartup ProcUser!=sys ProcUser!=lvc ProcUser!
    =nvc ProcName=Winword.exe | stats count as Starts by ProcName ProcUser | where Starts > 5 | rename
    ProcUser as User ProcName as Process | sort 0 -Starts
```

✓ 42,319 events (9/25/17 12:00:00.000 AM to 10/25/17 12:01:03.000 AM)

Process	User	Starts
winword.exe	CORP\hq48912	1829
winword.exe	CORP\hq28085	1762
winword.exe	CORP\hq61218	1705
winword.exe	CORP\hq12006	1635

Accelerated Data Model Search

uberAgent comes with an accelerated data model. Searching an accelerated data model is a lot faster than searching the underlying index (by ”a lot” we mean at least 50x), but requires a different search syntax based on the `pivot` or `tstats` commands. We are using `pivot` because of the easier syntax compared to `tstats`.

In this second example, we demonstrate how to search for starts of a ”modern” UWP app, specifically the weather app that is part of Windows. Most UWP apps cannot be identified by process name - which is simply backgroundTaskHost.exe. Luckily uberAgent determines the real app name automatically.

Step 1

We start with a count of all process starts.

Note: The documentation for the Splunk pivot command can be found here.

Note: the macro uA_DM_Process_ProcessStartup resolves to the name of the data model containing the Process_ProcessStartup dataset. We use this technique to facilitate moving datasets between data models.

```
| pivot `uA_DM_Process_ProcessStartup` Process_ProcessStartup count(
   Process_ProcessStartup) as Starts
```

Step 2

We filter for the weather app.

Note: An easy way to identify the name of the weather app is to dig around with a search like the following: index=`uberAgent_index` sourcetype=uberAgent:Process:ProcessStartup AppName=*Weather*

```
| pivot `uA_DM_Process_ProcessStartup` Process_ProcessStartup count(
   Process_ProcessStartup) as Starts
  filter AppName is "Microsoft.BingWeather"
```

Step 3

We split by user so that we get a count of process starts per user (renaming the ProcUser field to User in the process).

```
| pivot `uA_DM_Process_ProcessStartup` Process_ProcessStartup count(
   Process_ProcessStartup) as Starts
  filter AppName is "Microsoft.BingWeather"
  splitrow ProcUser as User
```

Step 4

We only keep users with more than five starts in the results list. We also sort the results so that the user with the highest number of starts is listed first. The 0 in the sort command ensures that the output is not truncated after the 10,000th result.

```
| pivot `uA_DM_Process_ProcessStartup` Process_ProcessStartup count(Process_ProcessStartup) as Starts
  filter AppName is "Microsoft.BingWeather"
  splitrow ProcUser as User
| where Starts > 5
| sort 0 -Starts
```

Step 5

We add the application name as a row to the results table.

```
| pivot `uA_DM_Process_ProcessStartup` Process_ProcessStartup count(Process_ProcessStartup) as Starts latest(AppName) as Application
  filter AppName is "Microsoft.BingWeather"
  splitrow ProcUser as User
| where Starts > 5
| sort 0 -Starts
| table Application User Starts
```

The Result

The resulting output is very similar to the first example above: a table with the application, the users and the number of starts that can easily be exported to CSV or otherwise be processed further.

How to Implement Drilldowns on Custom Dashboards

When you create a custom dashboard, Splunk automatically adds a drilldown to the search page. In many cases, that is not what you want. This article describes an easy way to implement custom drilldowns.

How It Works

1. Add a JavaScript file to your dashboard
2. Implement click event handlers in JavaScript, in which you set the new URL and optionally pass values to the drilldown page
3. Use the passed values in the drilldown page

Let’s explain these steps in more detail.

Add a JavaScript File

To add one or more JavaScript files to your Simple XML dashboard specify the names of the `.js` files in the `form` tag like this:

JavaScript files reside in the `appserver\static` directory of your app.

Click Event Handlers

In your JavaScript file add click event handlers to those dashboard elements (charts, tables, …) you want to implement a custom drilldown behavior for.

Please note that the dashboard elements need to have unique IDs in Simple XML for this to work, e.g.:

```
<table id="Table_Panel41"></table>
```

The JavaScript could look like this:

```
// RequireJS dependency handling
require (["splunkjs/mvc",
         "splunkjs/mvc/simplexml/ready!"], function (mvc)
{

   // Set click event handlers
   var chart21 = mvc.Components.getInstance ("Chart_Panel21");
   chart21.on ("click", drilldown);
   var table41 = mvc.Components.getInstance ("Table_Panel41");
   table41.on ("click:row", drilldown);

   // Define a drilldown function that can be used by multiple click event handlers
   function drilldown (event)
   {

      // Don't do any further event processing
      event.preventDefault ();

      // Get token values
      var earliest = GetToken (mvc, "earliest");
      var latest = GetToken (mvc, "latest");

      // Build the new URL
      var drilldownUrl = "single_machine_detail";
      drilldownUrl += "?earliest=" + encodeURIComponent (earliest);
      drilldownUrl += "&latest=" + encodeURIComponent (latest);
      drilldownUrl += "&FilterHostName=" + encodeURIComponent (event.data["click.value"]);

      // Go to the new URL
```

```
         window.location = drilldownUrl;
      }

   }
   );
```

Use the Passed Values on the Drilldown Page

In the example above we are passing the search time range (earliest and latest) to the drilldown page, where it is used automatically.

In addition to time, we pass the variable `FilterHostName`. This variable can be used in the Simple XML drilldown page searches by enclosing it in dollar signs, e.g. `$FilterHostName$`.

If you want to pass a value to an input field instead, prepend the field token's name with "form.". Example: your input field on the drilldown page is defined like this:

Specify the field's value as follows in JavaScript:

```
drilldownUrl += "&form.FilterField="+ encodeURIComponent ("Some field value");
```

Examples

The technique described here is used extensively on uberAgent's dashboards. Poke around our search head app and you will find many real-world examples.

How to Report on CPU Seconds & RAM GB Hours per User

uberAgent determines all the required data for charging based on usage. If you want to bill your customers depending on how much CPU or RAM their users'sessions consume you basically need the metrics *CPU seconds per user* and *RAM GB hours per user*. The latter can be explained as the number of gigabytes of RAM multiplied by the number of hours.

Example: RAM GB Hours per User

User A runs one session with an average RAM footprint of 2 GB for 3 hours. User B runs 2 sessions for 5 hours, one with an average RAM footprint of 1 GB and the other with an average RAM footprint of 2 GB. *GB hours* would be:

- **User A:** 2 GB * 3 hours = 6 GB hours
- **User B:** 1 GB * 5 hours + 2 GB * 5 hours = 15 GB hours

Splunk Search

The following Splunk search will return the total CPU seconds and RAM GB hours per Active Directory user account. It can be run over any time range (= billing period). Please note that due to the nature of the calculation every started hour counts in full, i.e. the resolution is one hour.

```
| pivot uberAgent Session_SessionDetail_Users
   first(SessionUserLower) as User
   avg(SessionWorkingSetMB) as AvgHourSessionWorkingSetMB
   sum(SessionCPUTimeS) as SumHourSessionCPUTimeS
   splitrow
      _time
      period hour
   splitrow
      SessionGUID
| eval AvgHourSessionWorkingSetGB=AvgHourSessionWorkingSetMB/1024
| stats
   count as SessionHours
   sum(AvgHourSessionWorkingSetGB) as GBHours
   sum(SumHourSessionCPUTimeS) as CPUSeconds
   avg(AvgHourSessionWorkingSetGB) as AvgSessionGB
   by
      User
| eval GBHoursRounded=round(GBHours, 2)
| eval CPUSecondsRounded=round(CPUSeconds, 2)
| eval AvgSessionGBRounded=round(AvgSessionGB, 2)
| fields
   User
   SessionHours
   GBHoursRounded
   CPUSecondsRounded
   AvgSessionGBRounded
```

Above search returns these fields per user:

- RAM GB hours
- CPU seconds
- Session hours (sessions * hours run per session)
- Average memory usage per session

How it Works

We are searching uberAgent's accelerated data model. That is why we use the `pivot` command.

The search consists of two main parts. First, we determine the average RAM usage per session and hour. Then we summarize the session hours per user.

CPU usage is even easier to calculate because uberAgent already reports it as CPU seconds so that we already have the length of time. We only need to summarize; we do that in two steps because the search needs to be constructed that way to determine RAM usage.

Data Collection

This section contains knowledge base articles about the data collection.

Citrix Applications Are Still Displayed with Old Name After Renaming

After renaming published applications in Citrix Studio, they still show up with the old name in uberAgent's dashboards.

uberAgent reads information for user's published applications from the registry from the path `HKEY_LOCAL_MACHINE\SOFTWARE\Citrix\Ica\Session\`.

Registry value	uberAgent field
PublishedName	SessionPublishedName
AppNames	SessionPublishedAppsCtx

When you look at a published application with `Get-BrokerApplication` via PowerShell you can identify multiple application name fields.

- ApplicationName
- BrowserName
- Name
- PublishedName

Citrix writes the application's BrowserName to the registry. But, this property does not get altered when changing the application's name through Studio. You can change the BrowserName property with PowerShell: `Get-BrokerApplication -Name SomeName | Set-BrokerApplication -BrowserName NewBrowserName`

After setting the BrowserName correctly, published applications will show up with the new name in uberAgent's dashboards.

Dashboards Have No Data

If you do not see any data on a dashboard please check all of the following.

uberAgent indexer app

Check the following in *Apps* > *Manage Apps*:

- Do you have a Splunk app with the name *uberAgent indexer* installed?

Index uberagent

Check the following in *Settings* > *Indexes*:

- Do you have an index with the name *uberagent*?
- What is the index'event count?
- How long ago was the latest event recorded?

Data flow

There are multiple options for sending the data uberAgent collects on the endpoints to the Splunk backend. Make sure one of the following is configured correctly:

- **uberAgent** on endpoint > **TCP port** 19500 on Splunk **indexer**
- **uberAgent** on endpoint > Splunk **HTTP Event Collector** on Splunk **indexer**
- **uberAgent** on endpoint > TCP port 19500 on **Universal Forwarder** on endpoint > receiver port 9997 on Splunk **indexer**

The port numbers above are default values that can be changed.

Index contents

Run the following Splunk search: `index=uberagent*`

- Do you see results from all hosts with uberAgent installed?
- Do you see results from many different source types?

Endpoints

If data from a specific endpoint is missing check the following on the endpoint:

- Make sure the service `uberAgent` is running.
- Check uberAgent's log file for issues.

Index permissions

The account you are accessing the uberAgent app with needs read permissions on the uberAgent index(es) or else no data will be returned by the searches. For details please see the article about multi-tenancy.

Time range

The dashboards display data from the selected time range only. Please make sure the time range selector is set to a period from which you expect events.

Differences between SessionPublishedAppsCtx and SessionPublishedName

The source type `uberAgent:Session:SessionDetail` contains two fields that appear to have the same purpose: `SessionPublishedAppsCtx` and `SessionPublishedName`.

One might assume, that both contain information about Citrix published applications run by a user. But, there are differences.

- `SessionPublishedAppsCtx`: all published applications run by a user. Applies to Citrix sessions.
- `SessionPublishedName`: name of the published application which was used to create the session. Applies to Citrix and VMware sessions.

How Application Startup Duration is Measured

uberAgent measures the startup duration of every process. This happens fully automatically whenever an application is launched and does not require configuration per application. A high-level overview of this feature can be found here.

How Not to Do it

uberAgent does not monitor application startup duration by waiting for specific windows to show up on the screen. Such an approach would be difficult to implement reliably because applications often display splash screens or other helper windows while starting up in order to reduce perceived launch times. A reliable mechanism would have to ignore those and wait for the main application window.

Additionally, even when the main window is shown, it may not be fully populated, the content might be missing.

Summing up, this would be hard to get right and would most likely require configuration per application, something we are trying very hard to avoid.

How Instead

Windows processes need to load DLLs in order to access operating system APIs or call functions from their own libraries. When a new process is created, it loads many DLLs in quick succession during its initialization phase. Once the process is ready, the DLL loading stops.

uberAgent monitors the DLL loading phase of new processes closely. The end of that phase is determined by looking for the first 30 second time window in which no DLLs are loaded.

Additionally, if there are IO operations during the DLL loading phase, uberAgent calculates the average IOPS during that phase and waits until IOPS drop to less than 20% for at least 10 seconds after the end of the DLL loading phase.

When those conditions are met uberAgent considers the process to be fully started up. Both of these timeout values can be adjusted globally. The DLL loading timeout is also available on a per-process level. Take look at the section ProcessStartupDurationWaitIntervalOverride of the configuration file.

DLL Loading and Disk IO

DLLs are only loaded from disk the first time around. When a process is started a second time most or all DLLs will already be in the operating system's file system cache. This optimization often effectively reduces the number of disk IOs incurred by a process start to zero.

uberAgent's startup duration algorithm works correctly regardless of whether DLLs are loaded from disk or from RAM (the file system cache).

Because uberAgent reports the number of disk IOs per process startup the data it collects can be used to easily determine how efficient the caching mechanism is.

”Visual” Startup Duration

uberAgent's algorithm for determining process and application startup durations is reliable and produces consistent results. However, it may not always correlate 100% with the duration as perceived by the end-user. The reason for this potential discrepancy is that uberAgent does not monitor what happens on the screen.

How to Configure the Data Collection Frequency

By default, uberAgent collects performance data every 30 seconds. That can be changed through the configuration file. uberAgent can run an unlimited number of independent *timers*. Each timer wakes up at regular intervals and collects the metrics assigned to it. As stated before, this happens every 30 seconds for all metrics by default.

To change this to 10 seconds edit the file `uberAgent.conf` so that the lines where the intervals are configured read like this:

```
# interval unit: milliseconds
interval = 10000
```

If, on the other hand, you want only specific metrics at a higher resolution (say 5 seconds), create a new timer section like this:

```
[Timer]
name = High-frequency app usage
comment = This field is optional
# interval unit: milliseconds
interval = 5000
UA metric = ApplicationUsage
```

This makes uberAgent collect application usage data every 5 seconds. Be sure to remove the *ApplicationUsage* metric from any other timers or it will be collected there, too.

Note: In addition to timers that collect data at fixed intervals uberAgent also has on-demand metrics that gather data when certain events occur, for example, user logons.

Incomplete Boot Duration Metrics

uberAgent collects detailed information about each machine boot process. This is governed by the on-demand metric *BootDetail* which is enabled by default and can be adjusted through uberAgent's configuration.

Symptoms

Sometimes the collected *BootDetail* metrics are incomplete and not all data is visualized in the *Boot Duration* dashboard. After reviewing uberAgent's logfile you may find entries like this:

```
OnOffTransition,Data from boot trace file missing! The following analysis may be incomplete. Lost events: 1000000, lost buffers: 0. Typical cause: insufficient disk bandwidth
```

Resolution

uberAgent does not add any load to the boot processes. It uses logs that are created by Windows anyway. Depending on the complexity of the boot process (e.g. installed drivers, applications) the pre-defined number of buffers, as well as the buffer size, are insufficient. You can adjust the number of buffers and the buffer size within the endpoint's registry:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Diagnostics\Performance\BootCKCLSettings
"BufferSize"
"MaximumBuffers"
"MinimumBuffers"
```

Increase all three values until the *lost events* message disappears in uberAgent's logfile. Doubling the default values is a good starting point.

No Data in Splunk Even Though uberAgent Sends Successfully

Scenario 1: uberAgent is sending data directly to Splunk

Symptoms

According to uberAgent's log file, there are no events in its in-memory send queue, indicating a successful data transfer to the Splunk server(s):

```
ReceiverStatistics,Splunk; SPLUNKSERVER:19500 - Events in queue: 0, queue size: 0.0 KB, sent: 1597, added to queue: 1597, rejected from queue: 0
```

When searching in Splunk's internal logs for errors using the following search:`index=_internal error` you see messages like the following:

```
ERROR TcpInputProc - Message rejected. Received unexpected 707406419 byte message! from src=IPADDRESS:PORT. Maximum message allowed: 67108864
```

Cause

The port uberAgent sends data to (default: 19500) is configured to receive data from Universal Forwarders.

Splunk Universal Forwarders do not send raw event data, they use a specific protocol. uberAgent, on the other hand, sends raw event data. If the formats sent by the source and expected on the target do not match the above error message may be logged and the incoming data is ignored on the Splunk server.

Resolution

If uberAgent is sending data directly to Splunk do **not** open port 19500 via *Forwarding and receiving* on the Splunk server. Install the *uberAgent_indexer* app instead which opens port 19500 as a raw TCP port.

Scenario 2: uberAgent is sending data to a locally installed Splunk Universal Forwarder

Symptoms

According to uberAgent's log file, there are no events in its in-memory send queue, indicating a successful data transfer to the locally installed Universal Forwarder: `ReceiverStatistics,Splunk; localhost:19500 - Events in queue: 0, queue size: 0.0 KB, sent: 1597, added to queue: 1597, rejected from queue: 0`
When searching for incoming data on port 19500 on the Splunk server using the following search: `index=* source="tcp:19500"` you see messages like the following (shortened):

```
--splunk-cooked-mode-v3--\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
    x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
    x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
    x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00...
```

Cause

The port Universal Forwarder sends data to (default: 9997) is configured to receive data from uberAgent.

Splunk Universal Forwarders do not send raw event data, they use a specific protocol. uberAgent, on the other hand, sends raw event data. If the formats sent by the source and expected on the target do not match above messages may be seen and the dashboards will be empty.

Resolution

If uberAgent is sending data to Splunk via a locally installed Universal Forwarder open TCP port 19500 on the Universal Forwarder (the same machine uberAgent runs on) and configure uberAgent to send to localhost:19500 as described here.

To enable Splunk to receive the Universal Forwarder's data open a receiving Port (default: 9997) on Splunk via *Forwarding and receiving* as described here. Do **not** open port 9997 as a raw TCP port.

Reasons For Empty SessionFgBrowserActiveTabHost Field

When the foreground window is a browser, uberAgent determines the active tab's URL and stores it in the `SessionFgBrowserActiveTabHost` field of the `uberAgent:Session:SessionDetail` sourcetype.

The `SessionFgBrowserActiveTabHost` field may be empty even though a browser window is in the foreground if any of the following is true.

Extension Not Installed or Enabled

The `SessionFgBrowserActiveTabHost` field is determined with the help of uberAgent's browser extension. If the extension is not installed or enabled in the browser window that is in the foreground, the field cannot be populated.

Private Mode

The uberAgent extension is not active in private mode unless the browser has specifically been configured to run the extension in private windows, too.

Denylist/Allowlist

If a tab's site URL is excluded by the `BrowserWebAppURL_Filter` configuration setting uberAgent leaves the `SessionFgBrowserActiveTabHost` field empty.

Save As or Similar Dialog

While a browser dialog such as *Save As* or *Authention Required* is being displayed, the foreground window is not a tab, but that modal dialog. Therefore, uberAgent leaves the foreground tab field `SessionFgBrowserActiveTabHost` empty. When that happens, the `SessionFgWindowTitle` field contains the title of the modal dialog, not the title of the previously active tab.

Developer Tools

While a browser's developer tools window is in the foreground, uberAgent leaves the foreground tab field `SessionFgBrowserActiveTabHost` empty. When that happens, the `SessionFgWindowTitle` field contains the title of the developer tools window.

Reported GPU Memory Usage per Process is Too High

uberAgent determines and reports the GPU memory usage. This happens if the *GpuUsage* configuration option is enabled (default: yes). GPU memory and compute usage are determined for every process that is using the GPU as part of the ProcessDetail source type.

Symptoms

On rare occasions, we have observed that the reported GPU memory usage is far too high. In those cases, the *ProcGpuMemMB* field would reach values as high as 17592178311168.

Cause

In the observed cases, the cause of the high values was a broken Citrix XenDesktop graphics driver that would cause display issues, too. The issue could be fixed by uninstalling and reinstalling the Citrix XenDesktop VDA (including a reboot after uninstall and reinstall).

Event Data Filtering Examples

Article Purpose

This article is a pool of Event Data Filtering rules and gets extended continually. Sources could be support tickets or community posts.

The rules serve as a suggestion of what is possible with Event Data Filtering and must be tested before use in your own company.

First, read the official documentation on Event Data Filtering. A few examples are listed there. Then, browse through the rules below.

List of Rules

Logons

```
# Do not send data about logons of the service user "LogonTest" in domain "AD" and of local as well as domain admins
# Last rule update: 2021-12-22

[EventDataFilter]
Action = deny
Sourcetype = Logon:LogonDetail
Sourcetype = Logon:GroupPolicyCSEDetail2
Query = User == "AD\\LogonTest" OR User LIKE r"%\\administrator"

[EventDataFilter]
Action = deny
Sourcetype = Process:LogonProcesses
Query = ProcUser == "AD\\LogonTest" OR ProcUser LIKE r"%\\administrator"
```

Miscellaneous

```
# Do not send any data about the user "JohnDoe" in domain "AD"
# You need to list every source type with one of the following fields: User, ProcUser, or SessionUser
# A list of source types is available in the metrics documentation on /en-us/uberagent/
# Last rule update: 2021-12-22

[EventDataFilter]
Action = deny
Sourcetype = Application:BrowserPerformanceChrome
Sourcetype = Application:BrowserWebRequests2
Sourcetype = Application:Errors
Sourcetype = Process:LogoffProcesses
Sourcetype = Process:LogonProcesses
Sourcetype = Process:NetworkTargetPerformance
Sourcetype = Process:ProcessDetail
Sourcetype = Process:ProcessStartup
Sourcetype = Process:ProcessStop
Sourcetype = ActivityMonitoring:ProcessTagging
Query = ProcUser == "AD\\JohnDoe"

[EventDataFilter]
Action = deny
Sourcetype = Application:NetworkConnectFailure
Sourcetype = Application:UIDelay
Sourcetype = Logoff:LogoffDetail
Sourcetype = Logon:GroupPolicyCSEDetail2
Sourcetype = Logon:LogonDetail
Sourcetype = Tags:UserHost
Query = User == "AD\\JohnDoe"

[EventDataFilter]
Action = deny
Sourcetype = Session:SessionDetail
Query = SessionUser == "AD\\JohnDoe"
```

PowerShell Constrained Language mode

This document explains how to use uberAgent with PowerShell's Constrained Language mode enabled.

Understanding Constrained Language Mode

PowerShell Constrained Language mode is a security feature that restricts access to sensitive language elements that can be used to invoke arbitrary Windows APIs. These features are often required to perform sophisticated cyber attacks. For a detailed description, see this Microsoft blog post.

Impact on uberAgent

uberAgent relies on PowerShell for collecting various metrics, such as details related to Citrix or Custom Scripts. The required data is accumulated via multiple APIs, most of which need full access to PowerShell's capabilities.

Identifying Potential Issues

If you encounter problems in this context, you will notice the following keywords near `powershell.exe` in uberAgent's log files:

- `PermissionDenied`
- `PSNotSupportedException`

The above keywords indicate issues that may have arisen due to the limitations imposed by Constrained Language mode in PowerShell.

How to use Constrained Language Mode With uberAgent

Constrained Language mode is often implemented by system-wide application control tools, such as AppLocker or Windows Defender Application Control. These tools can also remove the restrictions for files and folders you trust, allowing full command functionality for those particular files.

AppLocker

If AppLocker is used for application control, you can allow-list uberAgent's PowerShell scripts with the following steps:

1. Open the Group Policy editor.

2. Navigate to **Computer Configuration** > **Windows Settings** > **Security Settings** > **Application Control Policies** > **AppLocker** > **Script Rules**.

3. Create a new rule

   a) Action: **Allow**
   b) You can choose between Publisher (4) and Path (5)

4. Did you choose Publisher?

   a) Select an uberAgent script, e.g., `C:\ProgramData\vast limits\uberAgent\Configuration\Security inventory\Windows\Antivirus\Antivirus.ps1`
   b) Set the slider to **Publisher**

5. Did you choose Path?

   a) Select **Browse Folders**
   b) Select a folder e.g., `%OSDRIVE%\ProgramData\vast limits\uberAgent\Configuration\Security inventory\*`

6. If you want to exclude files from the allowlist, you can do that on the Exceptions page.

7. Finally, enter a name for the rule and a description.

8. Click **Create** to add the new rule.

Once the policies are synchronized at the endpoint, uberAgent's scripts should run in `FullLanguage` mode.

Ensure that allow-listed folders and scripts are read-only for regular users. This prevents privilege escalation and ensures PowerShell can execute scripts without modifications.

Saving uberAgent Data to Disk on the Endpoint

Occasionally, you may find yourself in a situation where you need to look at uberAgent's collected data before it is sent to the configured backend(s). This knowledgebase article shows you how to do that by guiding you through the following steps:

- Make the required **adjustments** to your **uberAgent configuration**.
- Set up a **local TCP listener** on the endpoint, which ultimately saves the data to a text file on the local disk.
- **Compare** local with backend data.

uberAgent Configuration

Add an additional `[Receiver]` stanza to your active uberAgent configuration (`uberAgent.conf`) as follows:

```
[Receiver]
Name = Local TCP Receiver
Type = Splunk
Protocol = TCP
Servers = localhost:19501
```

This instructs uberAgent to send the data to a local TCP listener on port 19501 - in addition to any other backends you already had configured.

TCP Listener

Download the latest version of the TCP listener script from the uberAgent Support Scripts GitHub repository and save it to a location of your choice on the endpoint. We use `C:\Temp\uA-StartTcpReceiver.ps1` here. Start the TCP listener script by running one of the following examples in an elevated PowerShell console.

Example 1

```
C:\Temp\uA-StartTcpReceiver.ps1 -Port 19501 -TimeoutMinutes 10 -
    EnableLogging
```

This starts the TCP listener on port **19501** and saves the received data to `C:\Temp\uA-Data.txt` . The script automatically **stops after 10 minutes**. Logging is enabled, and the log file is saved to `C:\Temp\uA-StartTcpReceiver.ps1.log`.

Example 2

```
C:\Temp\uA-StartTcpReceiver.ps1 -Port 19501 -TimeoutMinutes 0 -
    SearchString "sourcetype=uberAgent:Application:Errors" -
    EnableLogging
```

This starts the TCP listener on port **19501** and saves the received data to `C:\Temp\uA-Data.txt`. The script does not stop after a fixed timeout. It **stops when the string** `sourcetype=uberAgent:Application:Errors` **is found** in the received data, meaning an application crash occurred. Logging is enabled, and the log file is saved to `C:\Temp\uA-StartTcpReceiver.ps1.log`.

Comparing Local With Backend Data

After the TCP listener stops, you can now open the file `C:\Temp\uA-Data.txt` with your text editor of choice and compare the raw data before it leaves the endpoint with the data that is available in your backend(s). For the sake of this article, we compare the formatting and values of a particular event from the `uberAgent:System:SystemPerformanceSummary2` sourcetype.

Local Data

The raw data, as saved locally by the script, looks as follows:

```
***SPLUNK*** host=VM-WINDOWS11ARM index=uberagent source=uberAgent
    sourcetype=uberAgent:System:SystemPerformanceSummary2
1694008385925,0.4,32,2.57,0,1,0,29,0,0.3,,0.6,0.1,0,266,124.5,51369,1213,99,100,0
```

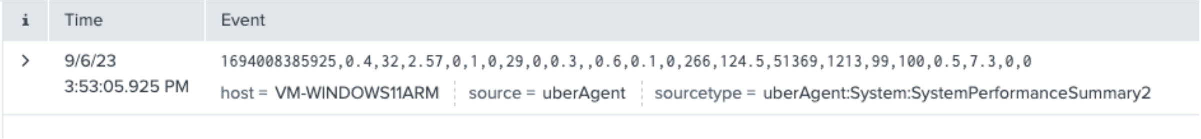
Please note that this data format uses two lines per event: one line of metadata (which does **not** count towards the Splunk data volume) and one line of actual event data.

Backend Data

We use the below Splunk search to retrieve the same data as saved locally:

```
index=uberagent host=VM-WINDOWS11ARM sourcetype=uberAgent:System:
    SystemPerformanceSummary2 Timestamp=1694008385925
```

The search gives us an event that is identical to the locally saved data (the metadata has, of course, been processed by Splunk).



i	Time	Event
>	9/6/23 3:53:05.925 PM	1694008385925,0.4,32,2.57,0,1,0,29,0,0.3,,0.6,0.1,0,266,124.5,51369,1213,99,100,0.5,7.3,0,0 host = VM-WINDOWS11ARM ⁞ source = uberAgent ⁞ sourcetype = uberAgent:System:SystemPerformanceSummary2

So, in this case, everything works as it should.

Data Storage

This section contains knowledge base articles about data storage.

How to Configure Data Retention

Splunk is very flexible with regards to data retention. You can configure when data is old enough to be deleted (after optionally being archived elsewhere). The place to do this is the file `indexes.conf` in uberAgent's app directory.

uberAgent stores its data in its own index called `uberagent`. The default configuration for the uberagent index comes from the file `[uberagent app directory]\default\indexes.conf`:

```
[uberagent]
homePath = $SPLUNK_DB/uberagent/db
coldPath = $SPLUNK_DB/uberagent/colddb
thawedPath = $SPLUNK_DB/uberagent/thaweddb
maxMemMB = 20
maxConcurrentOptimizes = 6
maxHotIdleSecs = 86400
maxHotBuckets = 10
maxDataSize = auto_high_volume
```

Important: Do not change this file! If you do, your changes will be lost when the application is updated. Instead, create a new indexes.conf in the app's `local` subdirectory. Settings from `local/indexes.conf` overwrite settings from `default/indexes.conf`.

The Splunk documentation page Configure index storage lists the relevant settings from `indexes.conf`. The most important settings for controlling index storage and data retention are:

- `frozenTimePeriodInSecs`: Absolute time in seconds after which data is deleted (default) or archived (if configured). The default is approximately 6 years.
- `maxTotalDataSizeMB`: Maximum total size of the index in MB. The default is 500,000 MB. When the index reaches this size, the oldest buckets (data directories) are ”rolled to frozen”, a process that triggers archival (if configured) and subsequent deletion.

By the way, Splunk will never completely fill your disks. By default, it stops accepting new data when the free disk space reaches 2,000 MB.

Data Volume

This section contains knowledge base articles about the data volume.

The Data Volume Dashboard Does Not Display Values For All Metrics

Symptom

The data volume dashboard only displays ”N/A” for the volume of some or all metrics.

Possible Cause: Splunk Configuration

The data volume dashboard queries Splunk's `_internal` index which in turn reads that data from Splunk's log file `license_usage.log`. By default, every Splunk indexer periodically reports to the license manager stats of the data indexed: broken down by source, source type, host, and index. If the number of distinct (source, source type, host, index) tuples grows over the *squash_threshold*, Splunk squashes the {host, source} values and only reports a breakdown by {sourcetype, index}. This is to prevent high memory usage and an unwieldy number of `license_usage.log` lines.

Resolution

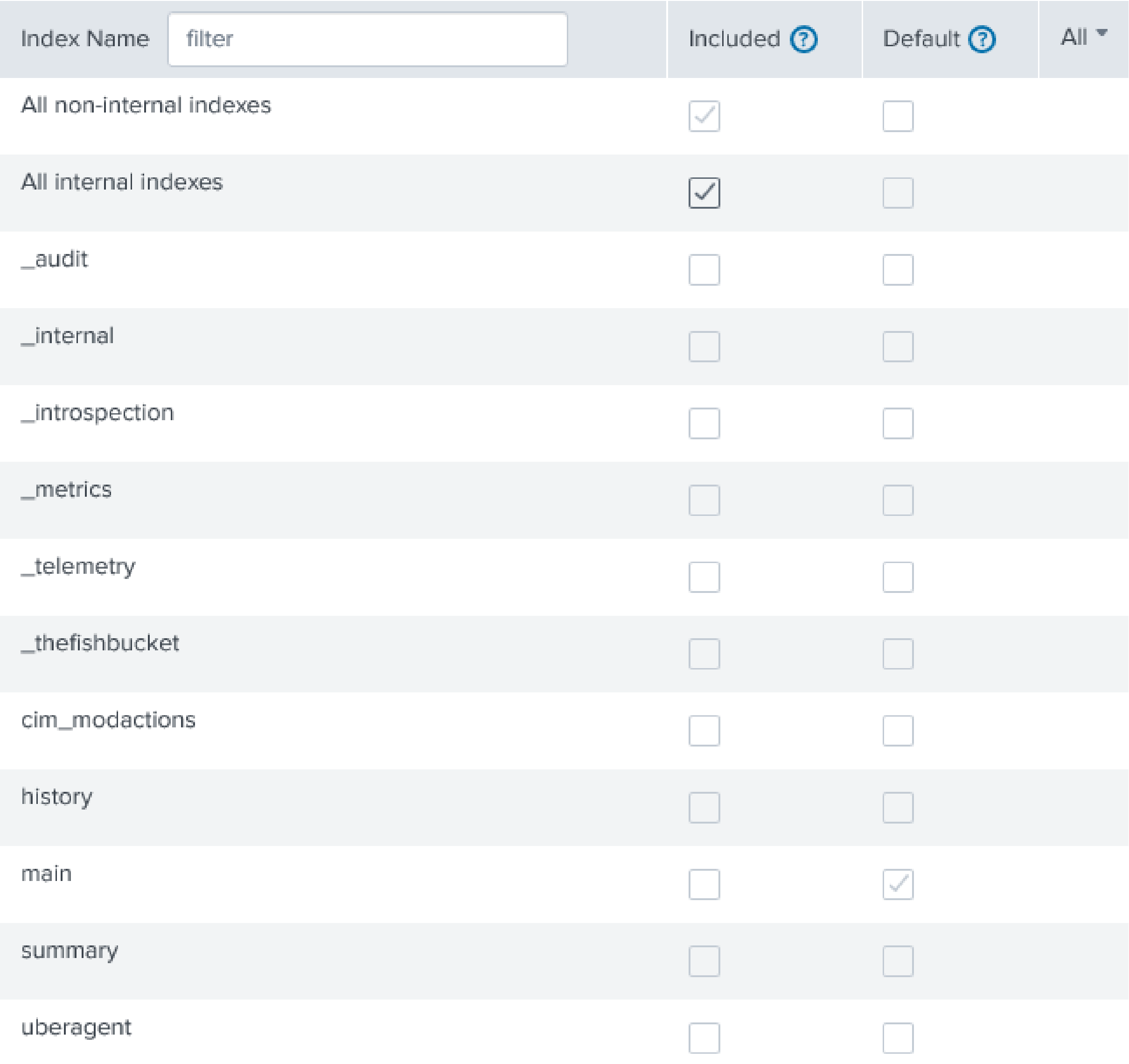
Increase the squash_threshold by adding the following to `$SPLUNK_HOME/etc/system/local/server.conf`:

```
[license]
squash_threshold = 10000
```

You will find more information here.

Possible Cause: Permissions

Your user account might not have permissions for the index `_internal`. To check first go to *Settings* -> *Access controls* -> *Users* -> *Your user name* and note the roles assigned to you. Then go to *Settings* -> *Access controls* -> *Roles* and check if any of your roles has either `_internal` or *All internal indexes* in the box *Selected search indexes*, i.e. one of these:



Index Name filter	Included	Default	All
All non-internal indexes	☑	□	
All internal indexes	☑	□	
_audit	□	□	
_internal	□	□	
_introspection	□	□	
_metrics	□	□	
_telemetry	□	□	
_thefishbucket	□	□	
cim_modactions	□	□	
history	□	□	
main	□	☑	
summary	□	□	
uberagent	□	□	

Resolution

Assign either `_internal` or *All internal indexes* to *Selected search indexes* of one of your roles.

Footprint & Performance

This section contains knowledge base articles about footprint and performance.

Changing the Accelerated Time Range

uberAgent's dashboards make use of a Splunk technology called *accelerated data model*. In a nutshell, an accelerated data model relies on an additional index that speeds up searches by 50x to 100x. As a caveat, an accelerated data model creates a slightly higher CPU load during index time and takes up more disk space on the indexers.

When Dashboards Might Load Slowly

You may notice that dashboards load slowly if the selected time range goes further back than seven days. This is due to the fact that uberAgent's default accelerated time range is exactly seven days. Older events will be included in searches and dashboards, too, of course, given an appropriate time range was selected, but will be a lot slower to come up.

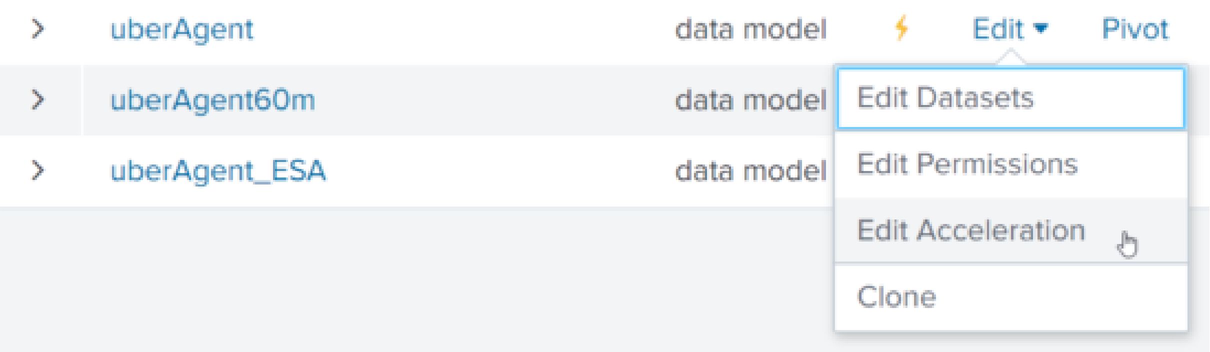
What To Do About It

To speed up searches that go further back than seven days you can increase the summary range, i.e. the number of days the accelerated data goes back in time. This can be done via the UI or via configuration files. In both cases, the actually changed setting is stored in `[uberagent app directory]\local\datamodels.conf` on the search heads.

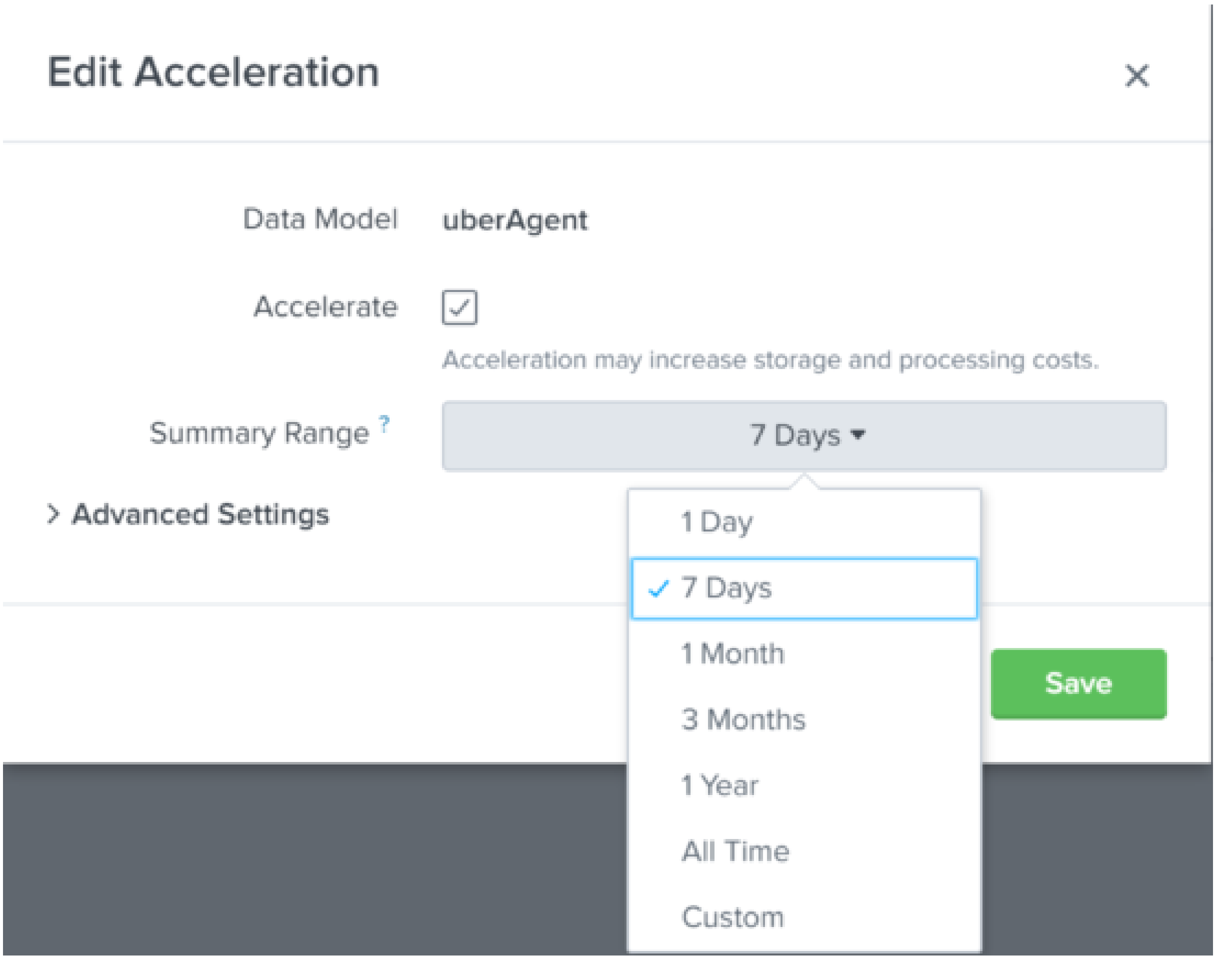
Important: After updating the uberAgent search head (dashboard) app, make sure your changed file(s) in the `local` subdirectory are still there.

User Interface

To change the accelerated summary range via the UI navigate to *Settings* > *Data models* and click *Edit* > *Edit Acceleration* for any of the uberAgent data models:



Select the desired summary range in the dialog that comes up:



Edit Acceleration

Data Model: uberAgent

Accelerate: ☑

Acceleration may increase storage and processing costs.

Summary Range: 7 Days

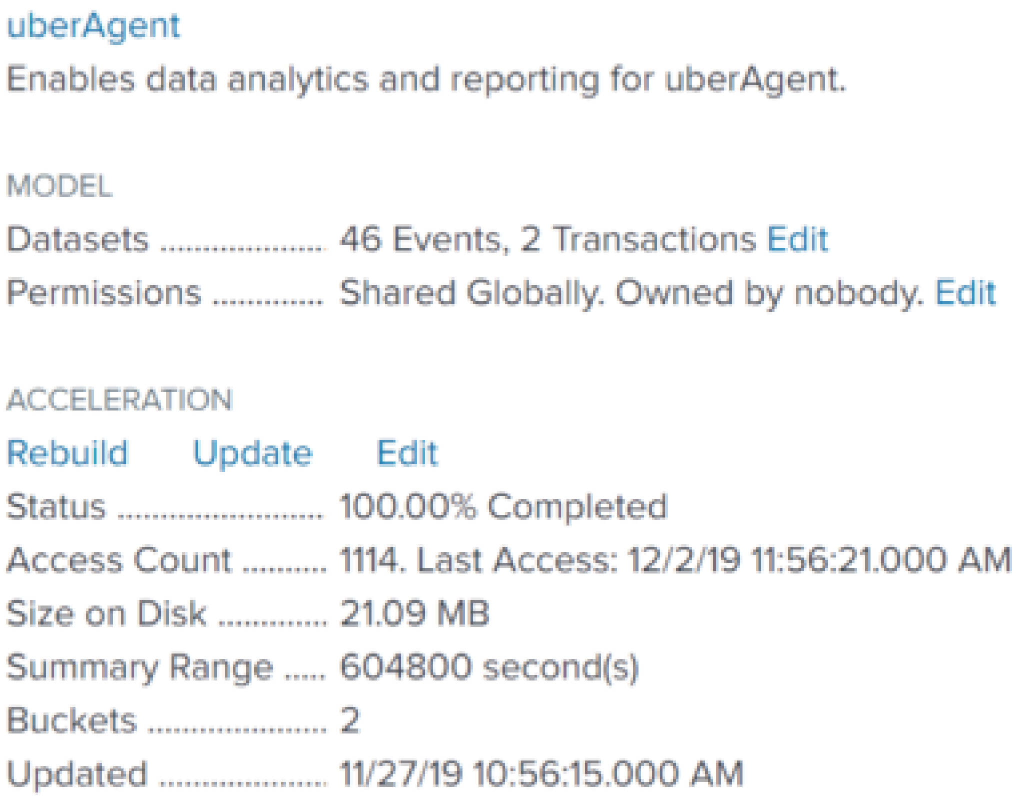
Configuration File

To change the accelerated summary range via the configuration files:

- Create a new directory `$SPLUNK_HOME\etc\apps\uberAgent\local` (if it does not already exist)
- Copy `$SPLUNK_HOME\etc\apps\uberAgent\`**`default`**`\datamodels.conf` to `$SPLUNK_HOME\etc\apps\uberAgent\local\datamodels.conf`
- Edit the setting *acceleration.earliest_time* in `$SPLUNK_HOME\etc\apps\uberAgent\local\datamodels.conf`
- Restart Splunk

Estimating Disk Space Requirements

Before you change the accelerated summary range you should make sure that sufficient disk space is available. To determine the disk space currently occupied by the high-performance analytics store (the special index used by data model acceleration) navigate to *Settings* > *Data models* in the Splunk UI and click the arrow next to uberAgent. You should see information similar to the following:



uberAgent
Enables data analytics and reporting for uberAgent.

MODEL
Datasets .... 46 Events, 2 Transactions Edit
Permissions .... Shared Globally. Owned by nobody. Edit

ACCELERATION
Rebuild Update Edit
Status .... 100.00% Completed
Access Count .... 1114. Last Access: 12/2/19 11:56:21.000 AM
Size on Disk .... 21.09 MB
Summary Range ..... 604800 second(s)
Buckets 2
Updated .... 11/27/19 10:56:15.000 AM

This shows the summary range (the default is 604800 seconds which is equivalent to 7 days) and the *Size on Disk*. If you plan to extend the summary range to a month expect an approximate increase by 4x.

Reuse of Open HTTP Connections

This article explains how the uberAgent endpoint agent handles open HTTP connections.

Where HTTP is Used

uberAgent makes use of HTTP(S) to send the collected data from the endpoint to some types of backends, for example, Splunk HEC, Elasticsearch, or Apache Kafka (via Confluent REST Proxy).

HTTP Connection Reuse

All HTTP(S) communication initiated by uberAgent is performed through libcurl (a variant of curl), which is probably the highest-quality networking library available today.

Libcurl automatically caches and reuses HTTP(S) connections. As the documentation states:

> Reusing a connection instead of creating a new one offers significant benefits in speed and required resources.

Requirements for HTTP Connection Reuse

For an HTTP connection to be reused, both server and client must support and enable **HTTP/1.1**.

Splunk HTTP Event Collector (HEC)

In its default configuration, Splunk's HTTP Event Collector (HEC) only enables HTTP/1.1 if the client sends a user agent string. uberAgent does that starting with version 6.0 beta 2.

We recommend configuring Splunk to always enable HTTP/1.1 for all versions of uberAgent by setting the following in the HEC's inputs.conf file, located in `$SPLUNK_HOME/etc/apps/splunk_httpinput/local`:

```
[http]
forceHttp10 = never
```

Server Failover and Load-Balancing

uberAgent can be configured with multiple servers per receiver. When there is more than one server per receiver, uberAgent randomly switches between servers every 100 seconds. If a server does not respond, uberAgent fails over to the next server in the list. This algorithm ensures that the load is distributed evenly between backend servers.

Installation & Upgrade

This section contains knowledge base articles about the installation and upgrades.

Directories and Registry Key Created by uberAgent's Installer

uberAgent is not only lightweight and unobtrusive, but it also creates only a minimal set of files and registry keys during installation that can be removed easily.

Uninstalling uberAgent

Windows

If you want to remove uberAgent from a Windows machine please use the installer's uninstall functionality from *Programs and Features*. The following list is intended as a reference.

macOS

Use the `uninstall.command` file to remove uberAgent. It can only be executed with elevated privileges. Only use this script if you want to completely remove uberAgent and its configuration from the computer. Its default location is `/Library/Application Support/uberAgent/uninstall.command`.

File System Changes

Windows

All files required by uberAgent go into the installation directory selected during setup (typically `c:\Program Files\vast limits\uberAgent`). During operation, the following additional files/-folders are created:

- uberAgent's log file
- `%ProgramData%\vast limits\uberAgent`
- `%SystemRoot%\System32\drivers\uberAgentDrv.sys`
- `%SystemRoot%\System32\drivers\uberAgentFilter.sys`
- `%SystemRoot%\System32\drivers\uberAgentNetMon.sys`

macOS

- uberAgent's log file
- uberAgent's configuration file
- Application bundle `/Library/uberAgent/uberAgent.app`
- Daemon configuration file `/Library/LaunchDaemons/com.vastlimits.uberAgent.plist`

Registry Changes

uberAgent stores installation and runtime state information to the key `HKEY_LOCAL_MACHINE\SOFTWARE\vast limits`.

uberAgent macOS Installation Fails When Executed From a Network Drive

Problem

uberAgent installation on macOS might fail when executed from a network drive, with a similar error.

```
PackageKit: Install Failed: Error Domain=PKInstallErrorDomain Code=106 "The package "uberAgent.pkg"is missing or invalid."
```

Workaround

If you are facing a problem like the one described above, copy the `uberAgent.pkg` file to a local directory on the endpoint first and repeat the installation from there.

uberAgent With Splunk Cloud: Differences to On-Premises Splunk Enterprise

If you are running **uberAgent** in combination with **Splunk Cloud**, there are several things to be aware of. Due to the certification policies for Splunk Cloud apps, the uberAgent indexer app and uberAgent dashboard apps (UXM & ESA) slightly differ from the ones available for download on our website. Hence you need to perform the following tasks by hand after the uberAgent apps have been installed in your Splunk Cloud environment.

uberAgent Indexer App

Index Creation

Manually create the uberAgent index. This can be achieved by using Splunk Web and the steps provided here. Use the following settings.

uberAgent Events

- **Index Name**: uberagent
- **Index Data Type**: Events
- **App**: uberAgent_indexer

uberAgent Scores

- **Index Name**: score_uberagent_uxm
- **Index Data Type**: Events
- **App**: uberAgent_indexer
- **Index Name**: score_uberagent_esa
- **Index Data Type**: Events
- **App**: uberAgent_indexer

Further index settings depend on your individual requirements. Feel free to download uberAgent from our website and have a closer look at the settings we ship with the `indexes.conf` for non-cloud deployments.

Configure Data Input

Splunk Cloud only accepts data via encrypted protocols. This means that you have the following options for sending uberAgent data to Splunk Cloud:

- via uberAgent's native functionality to send to Splunk's HTTP Event Collector (HEC). A guide on how to create an HEC token in Splunk Cloud is available here
- via a Universal Forwarder installed on the endpoint and configured to communicate with Splunk Cloud

Read here if you run into problems with self-signed certificates and how that affects your uberAgent deployment.

Alternatively, heavy forwarders can be used. More information on how to configure forwarders to send data to Splunk Cloud can be found here and here.

uberAgent Dashboard App

Data Model Acceleration

Enable acceleration for all uberAgent data models. Please follow the Splunk documentation for guidance.

Data Model Configuration

To avoid skipped searches, we recommend allowing the Splunk search scheduler to randomly distribute scheduled searches more evenly over their periods. To enable this setting, you need `acceleration.allow_skew = 100%` to be set within **every uberAgent's app** `datamodels.conf` file for **each uberAgent data model** (documentation).

The customization may be applied in one of these two places:

- Directly modify the `datamodels.conf` file shipped with the uberAgent apps in `$SPLUNK_HOME/etc/apps/[uberAgent|uberAgent_ESA]/`**`default`**`/`
- Create a new `datamodels.conf` file containing the `acceleration.allow_skew` setting in `$SPLUNK_HOME/etc/apps/[uberAgent|uberAgent_ESA]/local/`

For reference, please download uberAgent from our website to inspect the settings we ship with the `datamodels.conf` for non-cloud deployments. As direct modifications to configuration files are not possible when using Splunk Cloud, you need the assistance of Splunk support to change this setting.

uberAgent HTTP Event Collector (HEC) Receiver Configuration

Carefully read Splunk's documentation on how to send data to a HEC on Splunk Cloud Platform.

Important: Do not add the `<endpoint>` option, e.g. `/services/collector/event`, to your receiver configuration, as uberAgent does this automatically.

Using uberAgent With Self-Signed Certificates

uberAgent natively supports secure data transport for multiple backends, like Splunk, Elasticsearch, or Apache Kafka (via Confluent REST Proxy). For such communication via HTTPS uberAgent uses libcurl (a variant of curl), which is probably the highest-quality networking library available today.

Why Using Self-Signed Certificates

One might come to a point, during a PoC or evaluation phase, when using a certificate issued by an external or internal CA is not possible or a very complex process. Backends like Splunk or Elasticsearch offer the creation of self-signed certificates, which, by default, are not trusted by libcurl when presented to the client during the communication process. Therefore communication between uberAgent and the desired backend will fail. However, there are ways to change the configuration to make such a test scenario work.

Working With Self-Signed Certificates on macOS

uberAgent on macOS uses the operating systems implementation of libcurl. And libcurl itself utilizes *LibreSSL* as its library.

This can be easily verified by typing `curl --version` in the Terminal. The following output is an excerpt from a system running macOS 11.1 (Big Sur).

```
curl 7.64.1 (x86_64-apple-darwin20.0) libcurl/7.64.1 (SecureTransport)
    LibreSSL/2.8.3 zlib/1.2.11 nghttp2/1.41.0
```

In order to establish a successful connection, the following tasks need to be accomplished:

- Import the self-signed certificate into the macOS system keychain (aka *Keychain.app*), e.g. by double-clicking it
- Set the trust level for the just imported certificate to *Always Trust*, by using the *Keychain.app*

Working With Self-Signed Certificates on Windows

uberAgent supports a variety of different versions of Microsoft Windows. Since not every operating system release comes with its own implementation of libcurl, uberAgent takes care of that.

Libcurl for Windows relies on Schannel as its library. Since Schannel acts differently, compared to LibreSSL on macOS when dealing with self-signed certificates, the following steps are required:

- Import the self-signed CA certificate into the Windows certificate store (Trusted Root Certification Authorities)
- Depending on your requirement, add either *TLSRevocationChecksDisabled* or *TLSRevocationChecksBestEffort* or *TLSVerifyPeerDisabled* or *TLSVerifyHostDisabled* as an additional `ConfigFlag` to your uberAgent configuration

An example configuration stanza looks like this:

```
[Miscellaneous]
DebugMode = true
ConfigFlags = TLSRevocationChecksBestEffort
```

Licensing

This section contains knowledge base articles about the licensing.

How to Enable uberAgent on a Subset of Machines Only

With technologies like Citrix Provisioning Services (PVS) a single operating system image can be used for many machines. Typically all relevant applications are installed in that image.

Some customers may want to install uberAgent in such an image but not use it on all machines that run the image. To only enable uberAgent on a subset of machines do the following:

- Install uberAgent in the image
- Change the start type of the system service `uberAgent` from `automatic` to `manual`
- Use a computer startup script to determine whether uberAgent should be enabled on a machine. If so, start the service `uberAgent`

Supported License File Names & Multiple License Files

When you receive a license file for uberAgent it will most likely be named `uberAgent.lic`. That name can be changed, but when doing so please follow these guidelines:

- License files names must follow the pattern: `uberAgent*.lic`
- Multiple license files are supported. uberAgent will use the first license file that is valid.
- On Windows the license file(s) need to reside in the same directory as the main executable `uberAgent.exe`. On macOS the license file needs to be copied to `/Library/Application Support/uberAgent`.
- Alternatively, license file(s) can be hosted on a central file share (details).

These are examples for valid license file names:

- `uberAgent.lic`
- `uberAgent-clients.lic`
- `uberAgent license 2019-05-08.lic`

Storing uberAgent's License in Azure Files

Prerequisites

> **Note**
>
> This article describes how to use an Azure storage account key to access a file share hosted on Azure Files. Storage account keys are administrative keys for a storage account that grant full read and write permissions for all files, folders, and file shares within the storage account. The permissions are also applied to other storage resources, such as blobs, queues, and tables, contained within the storage account.

Please review Microsoft's documentation and make sure you understand the potential security concerns and whether this concept of storing uberAgent data in Azure Files meets your requirements.

If you want to store your uberAgent license file(s) in Azure Files, you need to meet the following prerequisites:

- TCP port 445 is not blocked (firewall/ISP)
- For mounting an SMB Azure file share on a Windows machine, SMB 3.1.1 is used. This means that Windows 10, version 1507, or Windows Server 2016 or above is needed. Microsoft provides a detailed list of supported OS versions.
- For mounting an SMB Azure file share on a macOS machine, SMB 3 is used. This means that macOS High Sierra 10.13 or above is needed. Further details on the requirements for macOS can be found here.

This KB article assumes that a file share within the Azure Storage account already exists. The uberAgent license file(s) can be stored in the file share directly.

uberAgent Configuration

To configure a central license directory, the `LicenseFilePath` option must be configured in the uberAgent configuration; the value of that option must contain the UNC path to the Azure files directory.

Using Azure Files With Your On-Premises Client

By default, Azure Files SMB shares can be accessed with a storage key that must be provided when mounting the directory, e.g.:

```
net use
```

When mapping the drive with the command shown above, it is mapped in the *USER* context. Therefore, uberAgent cannot interact with the file share because the agent runs in the *SYSTEM* context.

uberAgent accesses its central license file path via integrated authentication. The login data must be stored for the *SYSTEM* account by adding the credentials to the credential store. To store credentials as *SYSTEM*, one could use `PsExec` by running the following command:

```
.\psexec.exe -s -i cmd.exe cmdkey /add:"
```

After adding the credentials to the credential store for the `SYSTEM` account, uberAgent should be able to access the license file(s) stored in Azure Files.

Setting Up an Azure File Share With On-Premises AD-Authentication

There is a possibility to integrate a storage account into the local Active Directory and set permissions for your on-premises groups or users. Based on our research, it is not possible to set permissions for computer objects, though. Hence, this can’t be used for uberAgent’s license validation.

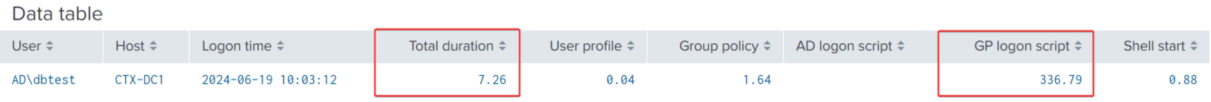
Logon

This section contains knowledge base articles about the logon.

”GP logon script” is longer than ”Total duration”

Symptoms

Looking at uberAgent’s **User Logon Duration** dashboard, you notice that for some logons, the value for **GP logon script** is larger than the value for **Total duration**.



Data table

User ⇕	Host ⇕	Logon time ⇕	Total duration ⇕	User profile ⇕	Group policy ⇕	AD logon script ⇕	GP logon script ⇕	Shell start ⇕
AD\dbtest	CTX-DC1	2024-06-19 10:03:12	7.26	0.04	1.64		336.79	0.88

Cause

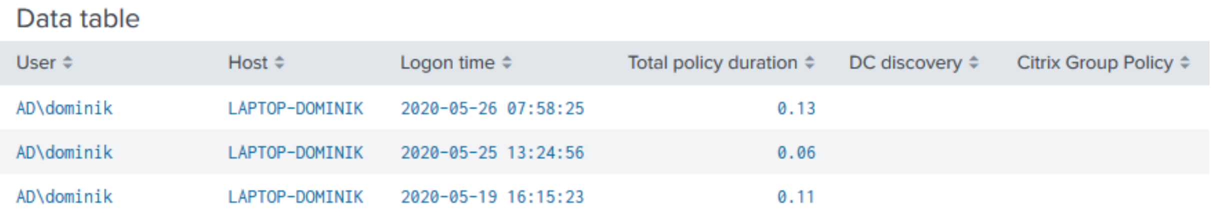
Most likely, your group policy logon scripts are not configured to run synchronously. By default, Explorer is started before logon scripts have finished. Thus, the desktop appears earlier, and the logon seems to be faster. Many administrators do not want this —they want to make sure all customizations are in place when the desktop is displayed. That is why synchronous script execution is often enabled.

In any case, with uberAgent, the **total logon duration is defined as the time from the actual logon until the shell is fully initialized**. If logon scripts are still running (potentially invisibly) when the second point is reached, then the script execution duration can be longer than the total logon duration.

See also Logon Data Arrives Late or Never in the Backend for more information how uberAgent calculates the **GP logon script** phase.

Not all CSEs Used are Listed on the Dashboard ”User Logon Duration - Group Policy”

The dashboard **Session** > **User Logon Duration - Group Policy** lists all Group Policy Client-Side Extensions (CSEs) that were applied during each of the logons listed. However, sometimes CSEs that are used in Group Policy are not shown on the dashboard. The display may look like this:



Data table

User	Host	Logon time	Total policy duration	DC discovery	Citrix Group Policy
AD\dominik	LAPTOP-DOMINIK	2020-05-26 07:58:25	0.13		
AD\dominik	LAPTOP-DOMINIK	2020-05-25 13:24:56	0.06		
AD\dominik	LAPTOP-DOMINIK	2020-05-19 16:15:23	0.11		

The dashboard only lists CSEs whose settings are *applied* during a logon. When a user has an existing user profile and a Group Policy Object (GPO) has not changed since the last logon the GPO normally is not applied again - this improves logon performance. Since the CSE's settings are not applied, it does not slow down the logon and consequently, it does not show up on the dashboard.

What is the Definition of the Metric "Pre Logon Init"

The *pre login init* metric, displayed on the dashboard *Single Logon*, measures the time from session creation until user logon. High values may or may not indicate a problem, depending on the situation.

Example 1: Citrix Virtual Apps and Desktops (CVAD)

During a logon to a Citrix XenApp terminal server typically a new session is created. In this case, the *pre logon init* metric is important: it shows how long it takes to prepare the new session and connect to it from the client.

Example 2: Physical PC

On a PC, on the other hand, the situation is often different. When a machine is booted, it presents its logon screen, waiting for the user to enter credentials. That, however, may take a very long time. The user might not even be near the machine when it is started. Therefore the *pre logon init* value can be very high without indicating a problem.

Logon Data Arrives Late or Never in the Backend

Symptoms

- Logon data for a user session is not available in Splunk immediately. Instead, it arrives after the user logged off.
- Logon data for a user session is not available in Splunk at all.

Cause

The cause is similar to ”GP logon script” is longer than ”Total duration” and how uberAgent determines the **GP logon script** phase.

For uberAgent, the phase starts with the process start `gpscript.exe /logon`. The phase ends when all recursive children of that process are stopped. So, if you have a main logon script that starts other scripts or processes and these continue to run, uberAgent waits for these to stop.

Two situations make uberAgent stop monitoring the logon:

1. **Logoff**: the user logs off, which effectively stops all processes in the session, including the ones from your logon script. uberAgent stops the logon monitoring and sends the data to the backend. This typically results in high values for the phase **GP logon script**.
2. **Logon timeout expires**: after 30 minutes, uberAgent stops monitoring the logon. No data about the logon is sent.

Splunk

This section contains knowledge base articles about Splunk.

Configuring Alerts

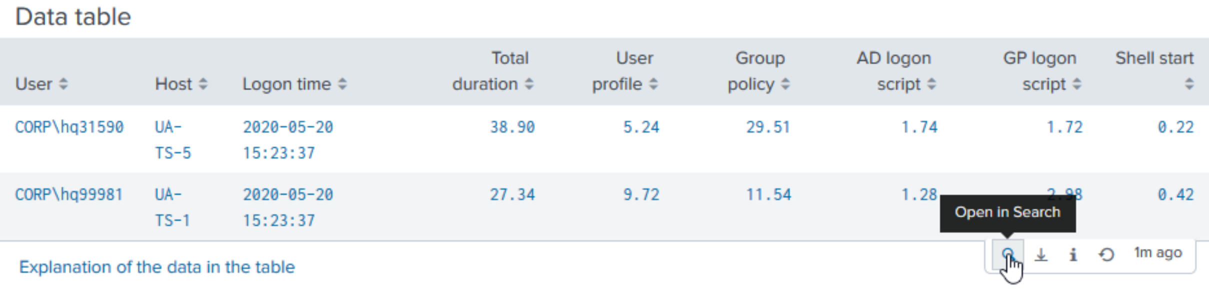
Splunk has a powerful alerting system that can be used to send emails or run arbitrary commands whenever a specific condition is met, as defined in a Splunk search.

In this walkthrough, we demonstrate how to alert when the duration of user logons exceeds a certain threshold.

Defining the Condition

An easy way to find the right Splunk search for defining the alert condition is to use an element already present on one of uberAgent's dashboards. In this example, we are interested in logon duration which is displayed on the **User Logon Duration** dashboard.

Navigate to the table near the bottom and click the button with the little lens symbol on it:

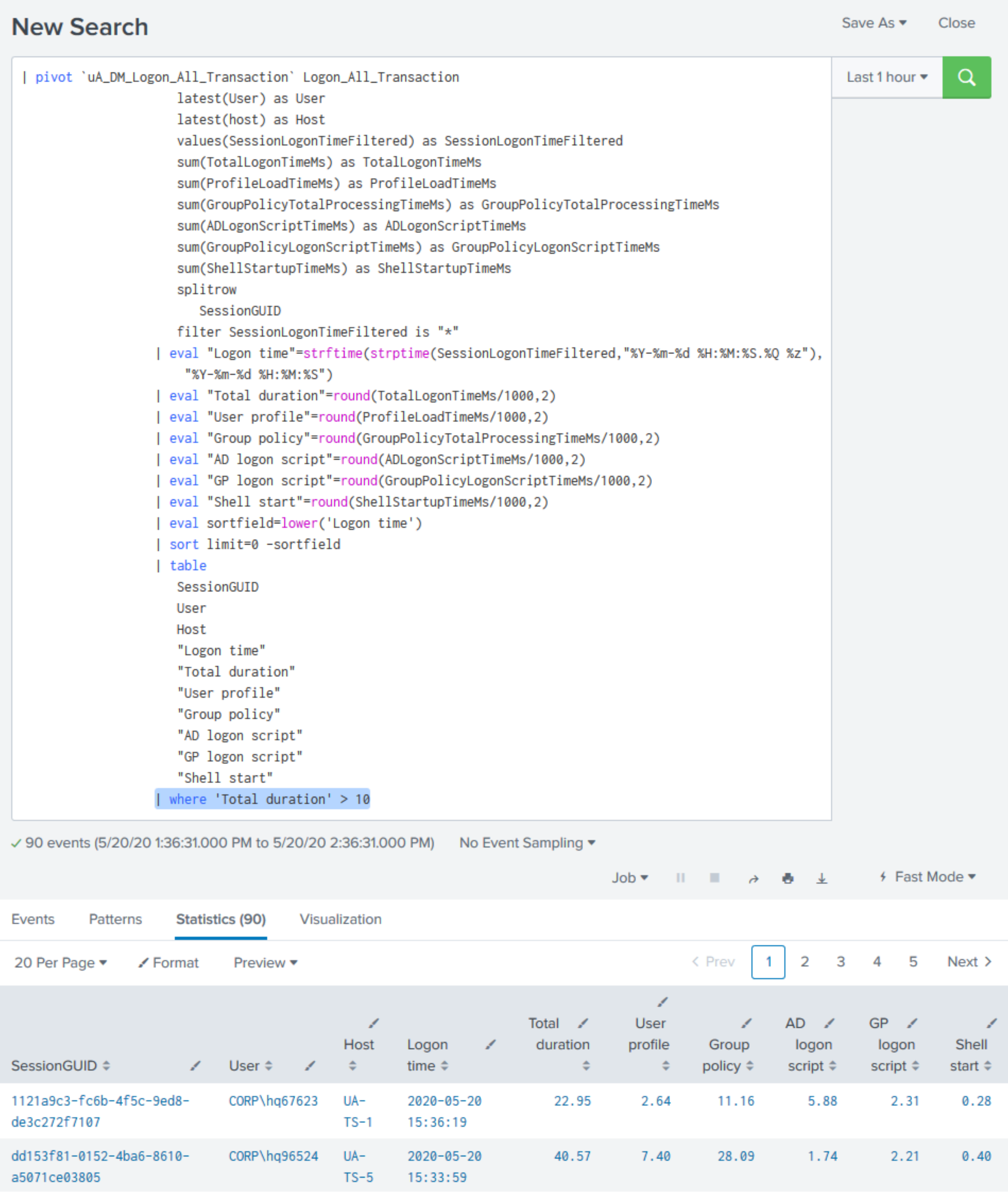


Data table

User	Host	Logon time	Total duration	User profile	Group policy	AD logon script	GP logon script	Shell start
CORP\hq31590	UA-TS-5	2020-05-20 15:23:37	38.90	5.24	29.51	1.74	1.72	0.22
CORP\hq99981	UA-TS-1	2020-05-20 15:23:37	27.34	9.72	11.54	1.28	2.98	0.42

Explanation of the data in the table

This opens the search in another window. At this stage you can add a condition (highlighted below) to filter for specific events:



```
| pivot `uA_DM_Logon_All_Transaction` Logon_All_Transaction
                     latest(User) as User
                     latest(host) as Host
                     values(SessionLogonTimeFiltered) as SessionLogonTimeFiltered
                     sum(TotalLogonTimeMs) as TotalLogonTimeMs
                     sum(ProfileLoadTimeMs) as ProfileLoadTimeMs
                     sum(GroupPolicyTotalProcessingTimeMs) as GroupPolicyTotalProcessingTimeMs
                     sum(ADLogonScriptTimeMs) as ADLogonScriptTimeMs
                     sum(GroupPolicyLogonScriptTimeMs) as GroupPolicyLogonScriptTimeMs
                     sum(ShellStartupTimeMs) as ShellStartupTimeMs
                     splitrow
                        SessionGUID
                     filter SessionLogonTimeFiltered is "*"
                   | eval "Logon time"=strftime(strptime(SessionLogonTimeFiltered,"%Y-%m-%d %H:%M:%S.%Q %z"),
                      "%Y-%m-%d %H:%M:%S")
                   | eval "Total duration"=round(TotalLogonTimeMs/1000,2)
                   | eval "User profile"=round(ProfileLoadTimeMs/1000,2)
                   | eval "Group policy"=round(GroupPolicyTotalProcessingTimeMs/1000,2)
                   | eval "AD logon script"=round(ADLogonScriptTimeMs/1000,2)
                   | eval "GP logon script"=round(GroupPolicyLogonScriptTimeMs/1000,2)
                   | eval "Shell start"=round(ShellStartupTimeMs/1000,2)
                   | eval sortfield=lower('Logon time')
                   | sort limit=0 -sortfield
                   | table
                     SessionGUID
                     User
                     Host
                     "Logon time"
                     "Total duration"
                     "User profile"
                     "Group policy"
                     "AD logon script"
                     "GP logon script"
                     "Shell start"
                   | where 'Total duration' > 10
```

✓ 90 events (5/20/20 1:36:31.000 PM to 5/20/20 2:36:31.000 PM) No Event Sampling ▾

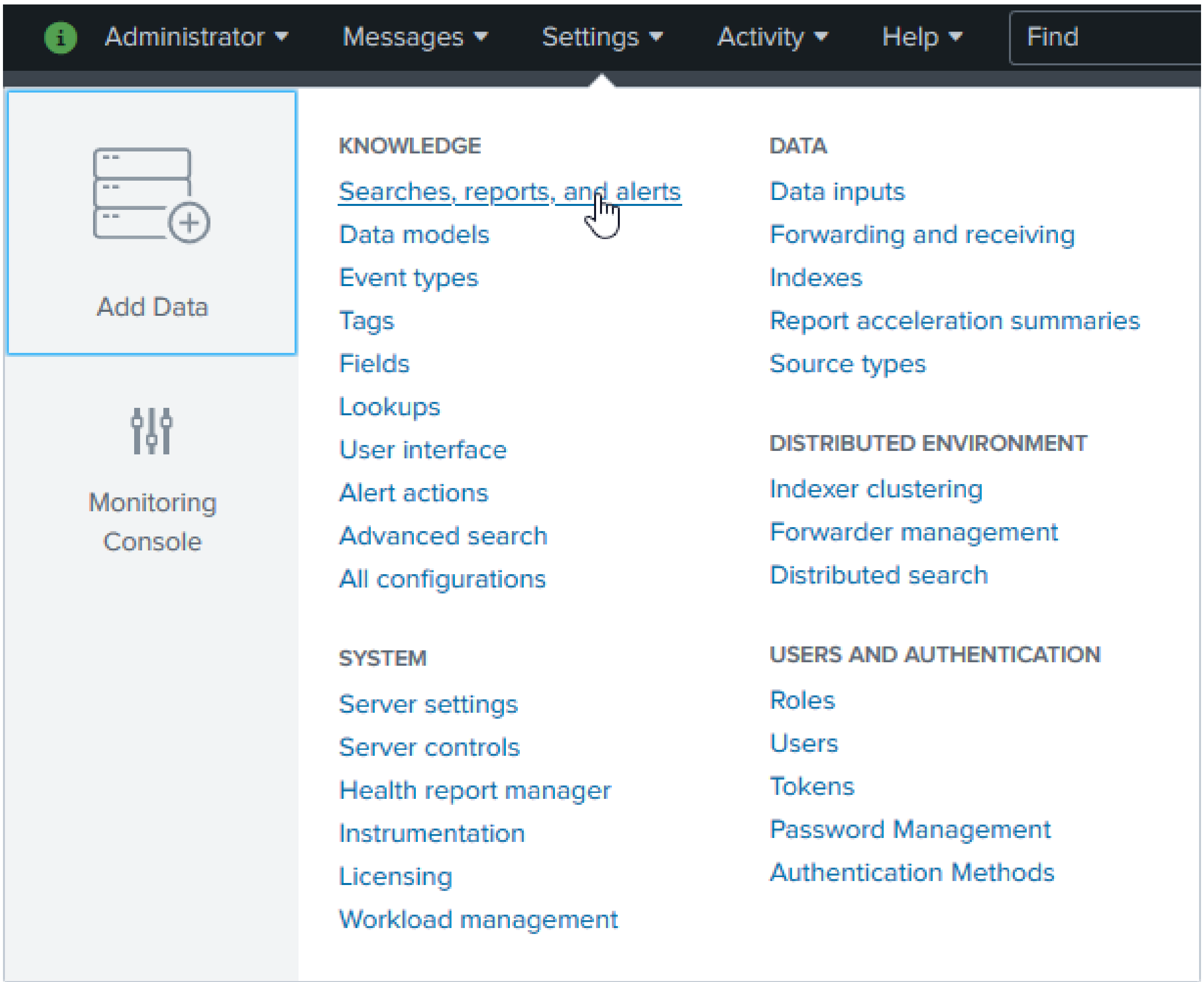
SessionGUID	User	Host	Logon time	Total duration	User profile	Group policy	AD logon script	GP logon script	Shell start
1121a9c3-fc6b-4f5c-9ed8-de3c272f7107	CORP\hq67623	UA-TS-1	2020-05-20 15:36:19	22.95	2.64	11.16	5.88	2.31	0.28
dd153f81-0152-4ba6-8610-a5071ce03805	CORP\hq96524	UA-TS-5	2020-05-20 15:33:59	40.57	7.40	28.09	1.74	2.21	0.40

Setting up the Alert

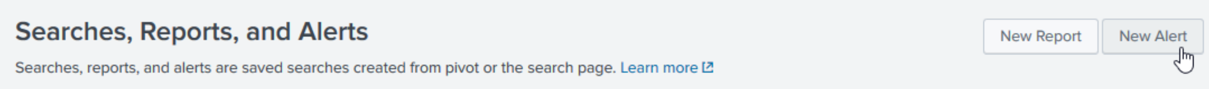
Once you are happy with the results returned from your conditional search, copy the search string.

Click **Settings** > **Searches, reports and alerts**:

]

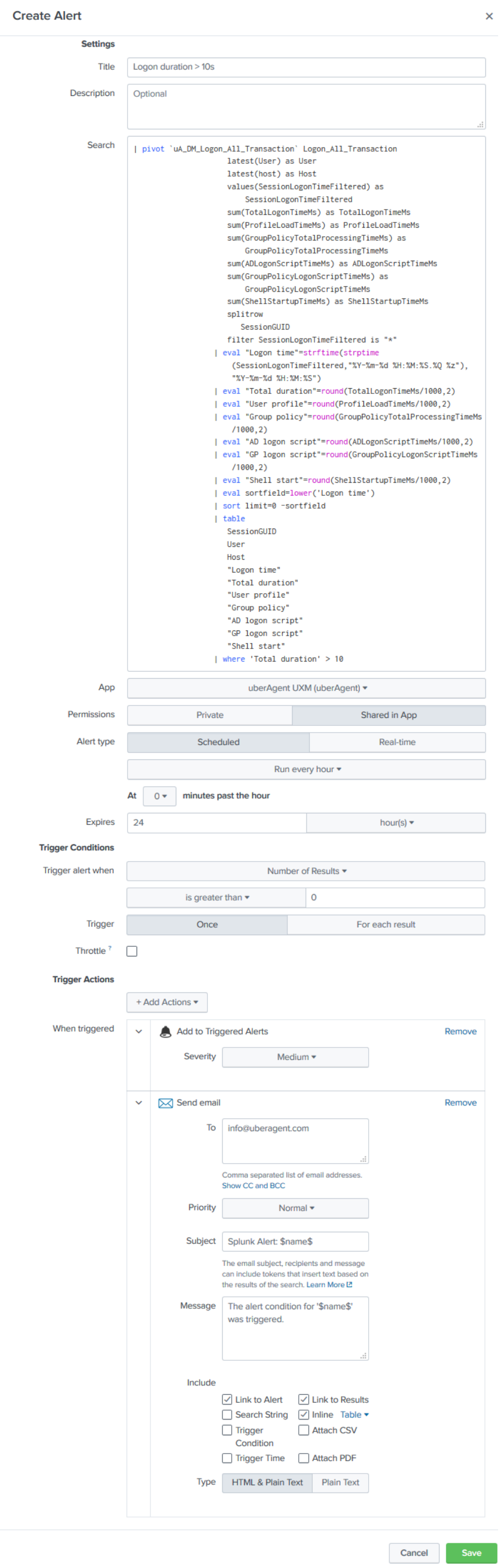


Click **New Alert**:



Searches, reports, and alerts are saved searches created from pivot or the search page. Learn more

Configure the alert as shown in the following very large screenshot:



Create Alert

Settings

Title: Logon duration > 10s

Description: Optional

Search:

```
| pivot `uA_DM_Logon_All_Transaction` Logon_All_Transaction
                    latest(User) as User
                    latest(host) as Host
                    values(SessionLogonTimeFiltered) as
                        SessionLogonTimeFiltered
                    sum(TotalLogonTimeMs) as TotalLogonTimeMs
                    sum(ProfileLoadTimeMs) as ProfileLoadTimeMs
                    sum(GroupPolicyTotalProcessingTimeMs) as
                        GroupPolicyTotalProcessingTimeMs
                    sum(ADLogonScriptTimeMs) as ADLogonScriptTimeMs
                    sum(GroupPolicyLogonScriptTimeMs) as
                        GroupPolicyLogonScriptTimeMs
                    sum(ShellStartupTimeMs) as ShellStartupTimeMs
                    splitrow
                       SessionGUID
                    filter SessionLogonTimeFiltered is "*"
                 | eval "Logon time"=strftime(strptime
                     (SessionLogonTimeFiltered,"%Y-%m-%d %H:%M:%S.%Q %z"),
                     "%Y-%m-%d %H:%M:%S")
                 | eval "Total duration"=round(TotalLogonTimeMs/1000,2)
                 | eval "User profile"=round(ProfileLoadTimeMs/1000,2)
                 | eval "Group policy"=round(GroupPolicyTotalProcessingTimeMs
                     /1000,2)
                 | eval "AD logon script"=round(ADLogonScriptTimeMs/1000,2)
                 | eval "GP logon script"=round(GroupPolicyLogonScriptTimeMs
                     /1000,2)
                 | eval "Shell start"=round(ShellStartupTimeMs/1000,2)
                 | eval sortfield=lower('Logon time')
                 | sort limit=0 -sortfield
                 | table
                    SessionGUID
                    User
                    Host
                    "Logon time"
                    "Total duration"
                    "User profile"
                    "Group policy"
                    "AD logon script"
                    "GP logon script"
                    "Shell start"
                 | where 'Total duration' > 10
```

App: uberAgent UXM (uberAgent)

Permissions: Private [✓ Shared in App]

Alert type: [✓ Scheduled] Real-time

Run every hour

At 0 minutes past the hour

Expires: 24 hour(s)

Trigger Conditions

Trigger alert when: Number of Results

is greater than 0

Trigger: [✓ Once] For each result

Throttle ? □

Trigger Actions

When triggered

Add to Triggered Alerts — Remove

Severity: Medium

Send email — Remove

To: info@uberagent.com

Comma separated list of email addresses.
Show CC and BCC

Priority: Normal

Subject: Splunk Alert: $name$

The email subject, recipients and message can include tokens that insert text based on the results of the search. Learn More

Message: The alert condition for '$name$' was triggered.

Include

☑ Link to Alert ☑ Link to Results
□ Search String ☑ Inline Table
□ Trigger Condition □ Attach CSV
□ Trigger Time □ Attach PDF

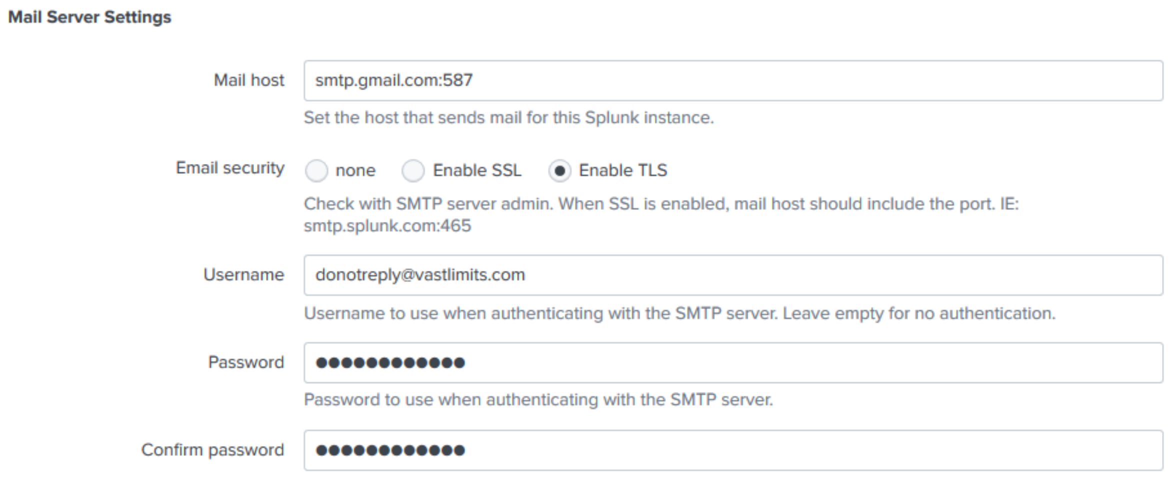
Type: [✓ HTML & Plain Text] Plain Text

Specify email addresses for the email trigger action. It might also be a good idea to check **Inline** so that you get the actual data causing the alert right in the email.

Configure Email Alerting

To configure alerting via Email go to **Settings** > **Server settings** > **Email settings**.

The following mail server settings apply to Google Apps:



Mail Server Settings

Mail host: smtp.gmail.com:587

Set the host that sends mail for this Splunk instance.

Email security: ○ none ○ Enable SSL ● Enable TLS

Check with SMTP server admin. When SSL is enabled, mail host should include the port. IE: smtp.splunk.com:465

Username: donotreply@vastlimits.com

Username to use when authenticating with the SMTP server. Leave empty for no authentication.

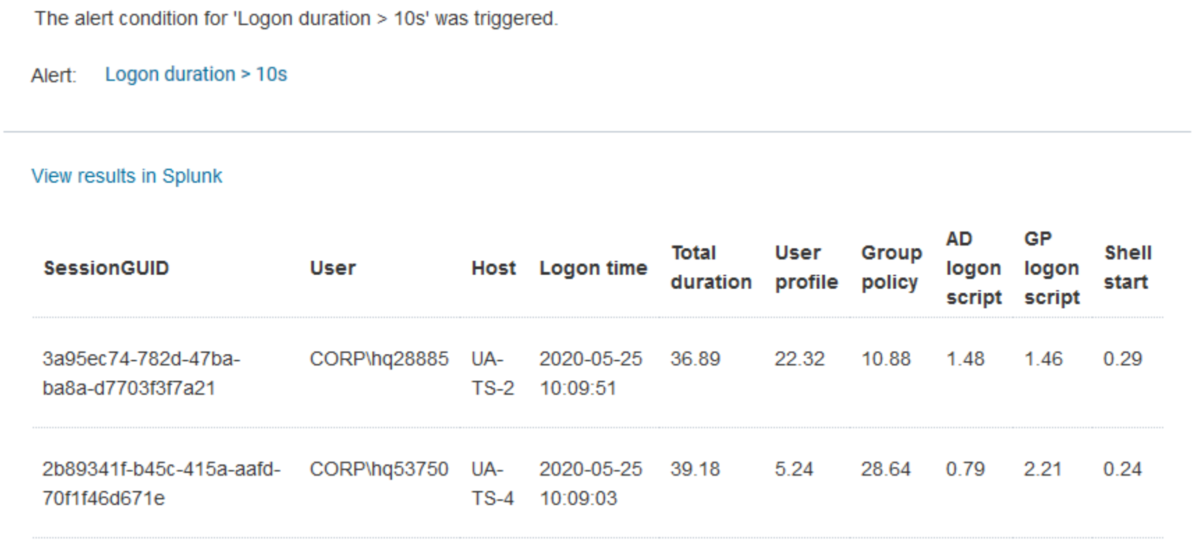
Password: ●●●●●●●●●●●●

Password to use when authenticating with the SMTP server.

Confirm password: ●●●●●●●●●●●●

The Resulting Alerts

With that, you are all done. With the sample configuration you will receive an email like the following whenever a user logon takes more than 10 seconds:



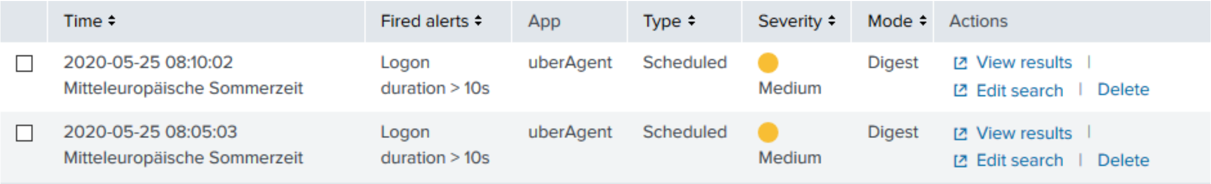
The alert condition for 'Logon duration > 10s' was triggered.

Alert: Logon duration > 10s

View results in Splunk

SessionGUID	User	Host	Logon time	Total duration	User profile	Group policy	AD logon script	GP logon script	Shell start
3a95ec74-782d-47ba-ba8a-d7703f3f7a21	CORP\hq28885	UA-TS-2	2020-05-25 10:09:51	36.89	22.32	10.88	1.48	1.46	0.29
2b89341f-b45c-415a-aafd-70f1f46d671e	CORP\hq53750	UA-TS-4	2020-05-25 10:09:03	39.18	5.24	28.64	0.79	2.21	0.24

You can also see all alerts in **Activity** > **Triggered Alerts**:



	Time ⬍	Fired alerts ⬍	App	Type ⬍	Severity ⬍	Mode ⬍	Actions
□	2020-05-25 08:10:02 Mitteleuropäische Sommerzeit	Logon duration > 10s	uberAgent	Scheduled	● Medium	Digest	View results | Edit search | Delete
□	2020-05-25 08:05:03 Mitteleuropäische Sommerzeit	Logon duration > 10s	uberAgent	Scheduled	● Medium	Digest	View results | Edit search | Delete

How to Change uberAgent’s Splunk Index Name

By default, uberAgent sends the data it collects to the Splunk index *uberagent*.

How is the index created

The index *uberagent* is created in the indexer app *uberAgent_indexer*. The file `default\indexes.conf` contains the relevant definitions.

How to change the index name

If you want to change the index name, you need to do so in the following places:

- `indexes.conf` (see above)
- uberAgent configuration (see below)
- `macros.conf` (see below)
- `eventtypes.conf` (see below)

uberAgent configuration

The index uberAgent sends the collected data to can be configured in uberAgent’s configuration with the setting *Index*.

Macros.conf

The index searched in the dashboards can be configured through the *uberAgent_index* stanza in the file `macros.conf` of the dashboard (search head) app. The default is as follows:

```
[uberAgent_index]
definition = uberagent*
```

Eventtypes.conf

The index searched for the event type *uberAgent_index_query* can be configured through the *uberAgent_index_query* stanza in the file `eventtypes.conf` of the dashboard (search head) apps. The default is as follows:

```
[uberAgent_index_query]
search = index=uberagent*
```

Splunk Product Editions (SKUs) Supported by uberAgent

Splunk comes in multiple product editions:

- Splunk Enterprise
- Splunk Cloud
- Splunk Free
- Splunk Light (discontinued as of 2020-05-01)

Splunk Enterprise, Splunk Free, and Splunk Light are hosted on-premises while Splunk Cloud is hosted by Splunk.

Out of these SKUs, **Splunk Enterprise and Splunk Cloud are fully supported by uberAgent**.

Splunk Enterprise

After installation, Splunk operates in Enterprise mode for 60 days after the installation. After that, it reverts to Free mode if no license is added.

During the 60-day trial period, Splunk is restricted to a daily data volume of 500 MB per day.

Splunk Cloud

uberAgent apps can be installed using the self-service app installation method. More information is available in Splunk's documentation.

The following apps are available in Splunk Cloud:

- uberAgent indexer app
- uberAgent UXM dashboard app
- uberAgent ESA dashboard app

After the uberAgent apps have been installed in your Splunk Cloud environment, some manual actions need to be taken. This knowledgebase article contains all the necessary details.

Splunk Free

uberAgent generally works well with Splunk Free except for one thing: Splunk bug SPL-40332 breaks not the initial creation but the update of CSV lookup tables. To workaround that we had to replace the `outputlookup`command with `action.populate_lookup` in a saved search. That, however, is a feature not enabled with Splunk Free.

As a result, uberAgent does not work correctly on Splunk Free until SPL-40332 has been fixed in a future version of Splunk.

However, using this workaround the lookup table can be generated manually.

Please see this document for more information on Splunk Free. There is also a feature comparison table between Splunk Free, Splunk Enterprise and Splunk Cloud available.

Splunk Light (Discontinued)

Splunk Light has a very limited feature set that does not even include the installation of apps (see Splunk Light vs. Splunk Enterprise). Due to these limitations, uberAgent cannot work with Splunk Light.

What to Do When You Get Splunk License Errors

What Are Splunk License Warnings and Violations

The Splunk documentation explains license warnings and violations as follows:

Warnings and violations occur when you exceed the maximum indexing volume allowed for your license.

If you exceed your licensed daily volume on any one calendar day, you will get a violation warning. The message persists for 14 days. If you have 5 or more warnings on an Enterprise license or 3 warnings on a Free license in a rolling 30-day period, you are in violation of your license, and search will be disabled for the offending pool (but indexing continues). Search capabilities return when you have fewer than 5 (Enterprise) or 3 (Free) warnings in the previous 30 days, or when you apply a temporary reset license (available for Enterprise only).

Symptoms of Splunk License Violations

When a license violation occurs you typically get this message:

```
Error in 'litsearch'command: Your Splunk license expired or you have
```

```
exceeded your license limit too many times. Renew your Splunk license
 by visiting www.splunk.com/store or calling 866.GET.SPLUNK.
```

What You Can Do

You can do either one of the following things to get back search functionality:

- Send fewer data to Splunk for indexing and then wait until there are no more than three (Splunk Free) / five (Splunk Enterprise) violations in the past 30 days
- Uninstall and then reinstall Splunk
- If you are on Splunk Enterprise contact Splunk to get a temporary reset license
- If you are trying the product out contact Splunk to get an evaluation license *and* a temporary reset license

Workaround for Lookup Errors with Splunk Free

On Splunk Free, uberAgent cannot automatically generate needed lookup tables. Because of the missing lookup tables, the following or similar error messages are displayed on many dashboards:

- No results found
- The lookup table `lookup_appnameidmapping` does not exist. It is referenced by configuration `uberAgent:Process:NetworkTargetPerformance`.
- The lookup table `lookup_appnameidmapping` does not exist. It is referenced by configuration `uberAgent:Process:ProcessDetail`.
- The lookup table `lookup_appnameidmapping` does not exist. It is referenced by configuration `uberAgent:Process:ProcessStartup`.

See this article for details and for information on officially supported versions.

Workaround

As a workaround, the lookup tables can be generated manually. To do that run slightly modified searches from uberAgent's `savedsearches.conf`.

One of the original searches:

```
index=`uberAgent_index` sourcetype=uberAgent:Application:AppNameIdMapping
 AppId=* AppName=* | stats latest(_time)as _time mode(AppName)as
AppName by AppId | inputlookup append=t lookup_appnameidmapping |
stats latest(_time)as _time latest(AppName)as AppName by AppId | eval
```

```
 TimeDelta=now()-_time | search TimeDelta<31536000 | fields AppName
AppId _time
```

We simply append the `outputlookup` command to have Splunk create the lookup tables. The full searches look like this:

```
index=`uberAgent_index` sourcetype=uberAgent:Application:AppNameIdMapping
 AppId=* AppName=* | stats latest(_time)as _time mode(AppName)as
AppName by AppId | inputlookup append=t lookup_appnameidmapping |
stats latest(_time)as _time latest(AppName)as AppName by AppId | eval
 TimeDelta=now()-_time | search TimeDelta<31536000 | fields AppName
AppId _time | outputlookup lookup_appnameidmapping
```

```
index=`uberAgent_index` (sourcetype=uberAgent:System:MachineInventory
 OR sourcetype=uberAgent:System:NetworkConfigInformation)OsVersion=*
 | stats latest(_time)as _time latest(OsVersion)as OsVersion latest
(OsBuild)as OsBuild latest(OsType)as OsType latest(AdDomainDns)as
 AdDomainDns latest(AdSite)as AdSite latest(AdOu)as AdOu latest(
CtxFarmName)as CtxFarmName latest(CtxMachineCatalogName)as CtxMachineCatalogNa
 latest(CtxDeliveryGroupName)as CtxDeliveryGroupName latest(HwManufacturer
)as HwManufacturer latest(HwModel)as HwModel values(NetworkConfigIPv4
)as NetworkConfigIPv4 by host | inputlookup append=t lookup_hostinfo
| fields - Ipv4Address | stats latest(_time)as _time latest(OsVersion
)as OsVersion latest(OsBuild)as OsBuild latest(OsType)as OsType
latest(AdDomainDns)as AdDomainDns latest(AdSite)as AdSite latest(AdOu
)as AdOu latest(CtxFarmName)as CtxFarmName latest(CtxMachineCatalogName
)as CtxMachineCatalogName latest(CtxDeliveryGroupName)as CtxDeliveryGroupName
 latest(HwManufacturer)as HwManufacturer latest(HwModel)as HwModel
values(NetworkConfigIPv4)as Ipv4Address by host | eval TimeDelta=now
()-_time | search TimeDelta<31536000 | fields host OsVersion OsBuild
 OsType AdDomainDns AdSite AdOu CtxFarmName CtxMachineCatalogName
 CtxDeliveryGroupName HwManufacturer HwModel Ipv4Address _time |
outputlookup lookup_hostinfo
```

```
index=`uberAgent_index` sourcetype=uberAgent:System:MachineInventory
 RAMSizeGB=* | stats latest(_time)as _time latest(RAMSizeGB)as
RAMSizeGB latest(IsBatteryPresent)as IsBatteryPresent latest(CPUName
)as CPUName latest(CPUSockets)as CPUSockets latest(CPUCoresPhysical)
as CPUCoresPhysical latest(CPUCoresLogical)as CPUCoresLogical latest(
CPUMaxMhz)as CPUMaxMhz latest(HwIsVirtualMachine)as HwIsVirtualMachine
 latest(OsUpdateBuildRevision)as OsUpdateBuildRevision by host |
 inputlookup append=t lookup_hostinfo2 | stats latest(_time)as
```

```
_time latest(RAMSizeGB)as RAMSizeGB latest(IsBatteryPresent)as
IsBatteryPresent latest(CPUName)as CPUName latest(CPUSockets)as
 CPUSockets latest(CPUCoresPhysical)as CPUCoresPhysical latest(
CPUCoresLogical)as CPUCoresLogical latest(CPUMaxMhz)as CPUMaxMhz
latest(HwIsVirtualMachine)as HwIsVirtualMachine latest(OsUpdateBuildRevision
)as OsUpdateBuildRevision by host | eval TimeDelta=now()-_time |
search TimeDelta<31536000 | fields host RAMSizeGB IsBatteryPresent
 CPUName CPUSockets CPUCoresPhysical CPUCoresLogical CPUMaxMhz
HwIsVirtualMachine OsUpdateBuildRevision _time | outputlookup lookup_hostinfo2
```

```
| pivot uberAgent Process_ProcessStartup latest(_time)as LastSeen
splitrow ProcName | eval ProcName = lower (ProcName)| inputlookup
append=t lookup_processstartup_processlist | stats first(LastSeen)
as LastSeen by ProcName | eval LastSeen = round (strptime (LastSeen
, "%Y-%m-%dT%H:%M:%S.%Q%z"), 0)| eval TimeDelta=now()-LastSeen |
search TimeDelta<31536000 | fields ProcName LastSeen | outputlookup
lookup_processstartup_processlist
```

```
| pivot uberAgent Process_NetworkTargetPerformance latest(_time)as
LastSeen splitrow NetTargetRemoteNameAddress | eval ProcName = lower
(NetTargetRemoteNameAddress)| inputlookup append=t lookup_networktargetperform
 | stats first(LastSeen)as LastSeen by NetTargetRemoteNameAddress |
eval LastSeen = round (strptime (LastSeen, "%Y-%m-%dT%H:%M:%S.%Q%z"
), 0)| eval TimeDelta=now()-LastSeen | search TimeDelta<31536000 |
fields NetTargetRemoteNameAddress LastSeen | outputlookup lookup_networktarget
```

Run these searches over a longer time range (e.g. last seven days) to capture among other things mappings for applications that are run only infrequently, too.

After running one of the full searches Splunk might display error messages like `"Could not write to file 'lookup_appnameidmapping': Failed to move file to final destination."` However, we found that it still creates all output files successfully.

User Interface

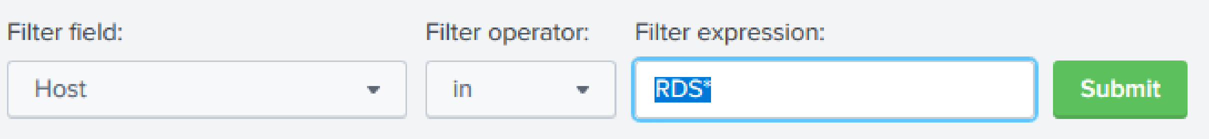
This section contains knowledge base articles about the user interface.

How to Separate Data from Different Types of Machines

If you are monitoring different types of machines with uberAgent, say XenDesktop/VDI clients and XenApp/RDS servers, you may want to separate the data in the dashboards so that you only see data from one group of machines or the other.

Temporary Separation

To temporarily view only data from a group of machines enter that part of the computer names that uniquely define machines in the desired group in the *Filter expression* input box on the dashboard. E.g. to filter for machines whose names start with the string *RDS*:



Filter field: Host

Filter operator: in

Filter expression: RDS*

Permanent Separation

To permanently separate data from a group of machines utilize uberAgent's multi-tenancy capabilities.

Splunk stores data in containers called indexes. By default uberAgent stores all its data in the index *uberagent*. But you can easily configure one group of machines to store data in an index called *uberagent-clients* and another to store in *uberagent-servers*. The only requirement is that all index names should start with a common prefix, otherwise the dashboards will not work.

By default uberAgent's dashboards search all indexes whose names start with *uberagent*. But you can configure individual Splunk user accounts so that they only have permissions to search in the index of one machine group, not the other. Depending on the user account you log on to Splunk with you see data from different machine groups.

For details on how to implement this please see this article.

Alternative for Permanent Separation

As an alternative to separating data into multiple indexes, you can always create custom dashboards that use some other means of differentiating between different types of machines. You could, for example, use a lookup table that stores a mapping between machine groups and computer name regexes.

Remoting Protocol is Console Instead of ICA or RDP

It may happen the dashboard *User Sessions* displays the remoting protocol as *Console* even though you know that the users were connecting over ICA or RDP.

Explanation

The Windows API used to query for the connection type does not always return ICA/RDP for an ICA/RDP session. During the logon and logoff phases, the connection type returned by the API is *Console*.

uberAgent determines the connection type every 30 seconds by default. Since the connection type may change during the lifetime of a session uberAgent needs to choose somehow which of the potentially multiple values to display. It does so by displaying the last reported value for that session.

For that reason sessions that take long to log on and/or off and/or are short-lived may be displayed with a connection type of *Console* instead of the expected *ICA* or *RDP*.

The Dashboards Do Not Work Correctly in Internet Explorer

Unfortunately part of the dashboard functionality does not work correctly in Internet Explorer 8. This also affects newer versions of IE in compatibility mode. If you have IE9 or newer, disable compatibility mode.

As of Splunk 6.1 support for Internet Explorer 7 and 8 has officially been removed by Splunk. The release notes state:

Internet Explorer versions 7 and 8: Browser support has been removed because they have either been marked as end-of-life by Microsoft or are no longer widely used. Support for these browsers might be removed entirely in a future release. For Internet Explorer support, use versions 9, 10, or 11.

Practice Guides

This section contains practical examples of how to use uberAgent for specific tasks.

Detecting Network Connectivity Problems

Network connectivity issues severely impact application functionality and performance, yet are often difficult to diagnose. This practice guide demonstrates how uberAgent helps.

Network issues come in two flavors: permanent and intermittent. The following sections explain both in detail.

Permanent Network Connectivity Issues

Permanent issues are often caused by configuration errors. Examples:

- A server service was migrated to a new machine but the application configuration is still pointing to the old machine.
- Firewall rules are missing so application connectivity is blocked.

Intermittent Network Connectivity Issues

Intermittent issues only occur from time to time and may cause seemingly random application slowness or occasional application errors. Examples:

- Bad WiFi signal reception
- Packet loss
- Saturated internet connection

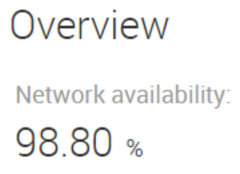
Diagnosing Network Connectivity Issues

uberAgent monitors all network traffic and enriches it with metadata like the originating application, the user account, the machine type, Active Directory OU, Citrix site, and much more. Network traffic is categorized as successful or failed connection, regular transmission or retransmit, and so on.

When suspecting network problems the dashboard to consult is *Application Network Issues*, located in the *Applications* menu.

Network Availability Indicator

The first thing to check is the *Network availability* metric:



Overview

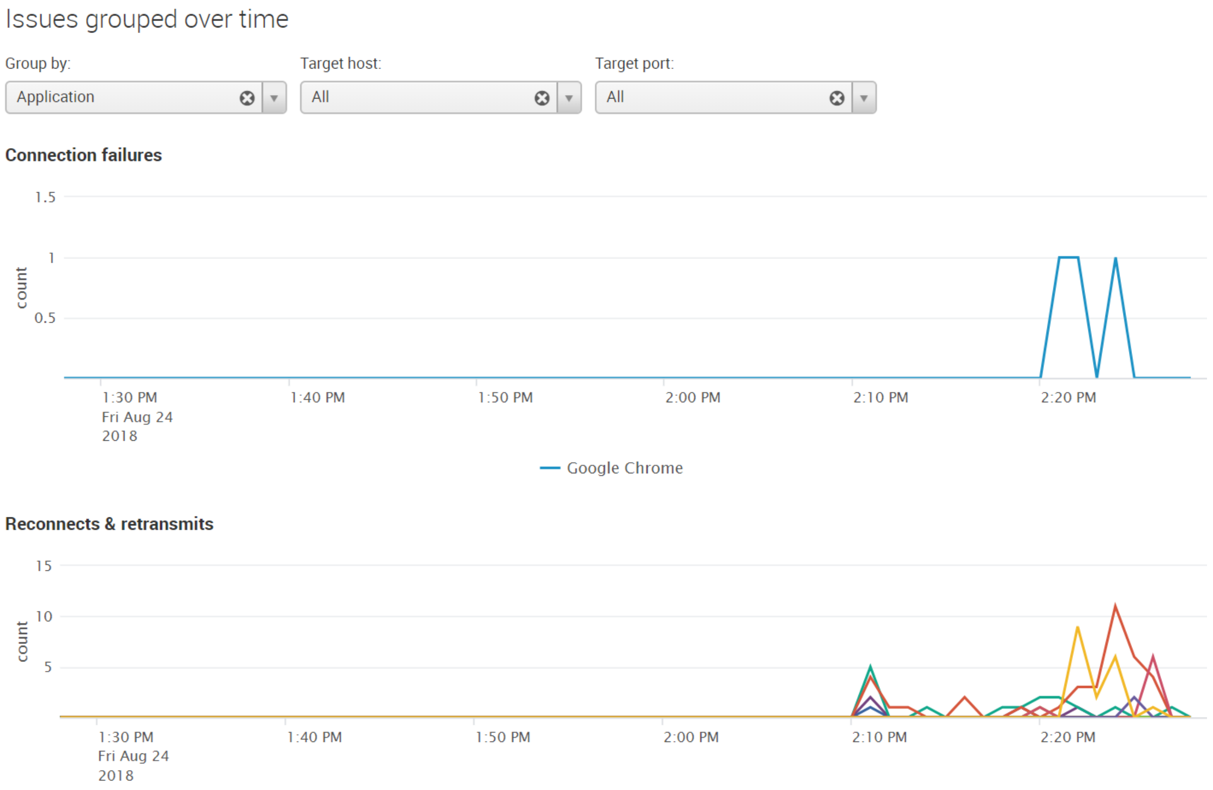
Network availability:

98.80 %

It is the ratio of successful connections and transmissions to the total number of connection and transmission attempts and should be at 100%.

Network Issues Over Time

The next thing to check are the timecharts in the section *Issues grouped over time*:



Issues grouped over time

Group by: Application

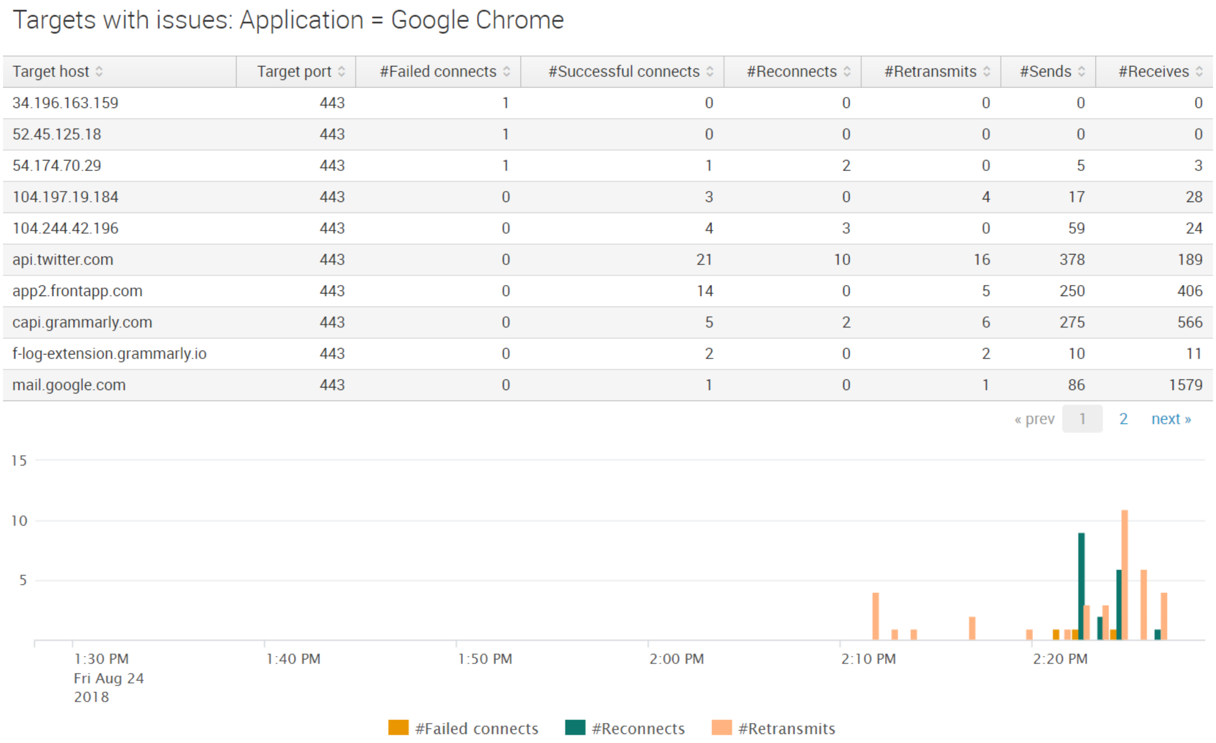
Target host: All

Target port: All

The two graphs show connection failures and reconnects & retransmits over the selected time range. By default both graphs group errors per application, but you can switch to errors per process, per source or target host, per target port or even per user.

Network Issues per Application

To further analyze the errors for a specific application (or host, user, etc.) click on the respective row of the table *Issues grouped (top 100)*. This performs an in-page drilldown, bringing up an additional table and chart with details on the selected item:



Targets with issues: Application = Google Chrome

Target host	Target port	#Failed connects	#Successful connects	#Reconnects	#Retransmits	#Sends	#Receives
34.196.163.159	443	1	0	0	0	0	0
52.45.125.18	443	1	0	0	0	0	0
54.174.70.29	443	1	1	2	0	5	3
104.197.19.184	443	0	3	0	4	17	28
104.244.42.196	443	0	4	3	0	59	24
api.twitter.com	443	0	21	10	16	378	189
app2.frontapp.com	443	0	14	0	5	250	406
capi.grammarly.com	443	0	5	2	6	275	566
f-log-extension.grammarly.io	443	0	2	0	2	10	11
mail.google.com	443	0	1	0	1	86	1579

In the screenshot above we can see that reconnection and retransmission errors started at around 2:10 PM for the selected application (Google Chrome). The errors are not limited to specific target hosts. This indicates intermittent problems, most likely related to unstable internet connectivity.

Citrix ADC Monitoring & Alerting

uberAgent 5.2 introduced Citrix ADC monitoring (formerly NetScaler ADC). This functionality lets you check the appliance health, gateway data volume, and service availability of your ADCs. While it is great to have all this information in dashboards, it sometimes makes sense to get notified when things are not working the way they should. This article demonstrates how to use Splunk's alerting capabilities to configure just that.

Splunk Alerting

If you are not familiar with Splunk alerting here is the explanation in a nutshell: you create a search; every time it produces a result you will get notified. Several notification methods are available (email, webhooks and so on). More on this topic is available in Splunk's documentation.

Example Searches for Citrix ADC Alerts

We created some example searches showing what is possible with uberAgent's data and Splunk's alerting capabilities. Feel free to customize them to your needs or use them as a starting point for your own creations.

Max Bandwidth per Appliance

The purchased license determines the maximum amount of data the ADC will process - it is a software limitation. Hence it is important to keep an eye on the bandwidth usage to react in a timely manner and buy a bigger license to not slow down the users. Here is an example where you will be notified when the bandwidth usage exceeded 900 megabytes.

```
| pivot `uA_DM_CitrixADC_AppliancePerformance` CitrixADC_AppliancePerformance
   max(ApplianceMBReceived) as "max(ApplianceMBReceived)"
   max(ApplianceMBSent) as "max(ApplianceMBSent)"
   splitrow
      HostName
| eval "Max. network data volume (MB)" = round('max(ApplianceMBReceived)' + 'max(ApplianceMBSent)',1)
| where 'Max. network data volume (MB)' > 900
```

Too Many HA Status Changes

A Citrix ADC high-availability (HA) pair consists of a primary and a secondary node. If the primary fails the secondary takes over. Too many status changes indicate that there is something wrong. Maybe your hypervisor does live migration of VMs? It should not for Citrix ADC VPX machines!

```
| pivot `uA_DM_CitrixADC_ApplianceInventory` CitrixADC_ApplianceInventory
   dc(LastHAStatusChange) as "#Status changes"
   splitrow
      HostName
| where '#Status changes' > 5
```

The current status is collected once per day by default consequently let that search run for at least a few days.

Get Notified When SSL Cards Are Down

In a hardware Citrix ADC appliance, SSL cards handle the encryption and decryption of SSL traffic. If an SSL card is down the encryption/decryption rate drops which possibly affects user experience.

```
| pivot `uA_DM_CitrixADC_ApplianceInventory` CitrixADC_ApplianceInventory
  latest(SSLCards) as "#SSL cards"
  latest(SSLCardsUp) as "#SSL cards up"
  splitrow
     HostName
| eval "#SSL cards down" = '#SSL cards' - '#SSL cards up'
| where '#SSL cards down' != 0
```

Get Notified When the Configuration Was Not Saved Within 24 Hours After a Change

Citrix ADC configuration changes are active immediately, even if they have not been specifically saved. However, unsaved changes are lost when the appliance is rebooted. With this search, you will be notified after 24 hours if someone forgot to save the configuration.

```
| pivot `uA_DM_CitrixADC_ApplianceInventory` CitrixADC_ApplianceInventory
  latest(LastConfigSaveTime) as LastConfigSaveTime
  latest(LastConfigChangedTime) as LastConfigChangedTime
  splitrow
     HostName
| eval LastConfigSaveTime_epoch = strptime(LastConfigSaveTime, "%Y-%m-%d %H:%S:%M")
| eval LastConfigChangedTime_epoch = strptime(LastConfigChangedTime, "%Y-%m-%d %H:%S:%M")
| eval Offset =  LastConfigChangedTime_epoch - LastConfigSaveTime_epoch
| where Offset > 86400
```

Get Notified When a Certificate Expires Within Four Weeks

The all-time favorite. A certificate is installed but the team managing it forgot to update it. Never let that happen again with this search!

```
| pivot `uA_DM_CitrixADC_Certificate` CitrixADC_Certificate
   latest(CertExpirationDate) as CertExpirationDate
   splitrow
      CertName as "Certificate name"
| eval CertExpirationDate_epoch = strptime(CertExpirationDate, "%Y-%m-%d %H:%S:%M")
| eval CurrentTime_epoch = now()
| convert ctime(now)
| eval thirty_days_from_now = CurrentTime_epoch + 2592000
| eval Expires_within_next_30_days = if(CertExpirationDate_epoch >= CurrentTime_epoch AND CertExpirationDate_epoch <= thirty_days_from_now, "yes", "no")
| where Expires_within_next_30_days = "yes"
| table
  "Certificate name"
```

```
   CertExpirationDate
   Expires_within_next_30_days
```

List All IPs Where Management Access Is Enabled but It Is Not an NSIP

It is a common best practice to enable management only on the NetScaler IP (NSIP). Anyway, managing it through the Subnet IP (SNIP) is much more easy, right? Just don't! Keep safe with this search:

```
| pivot `uA_DM_CitrixADC_Ip` CitrixADC_Ip
   latest(IpNetmask) as Netmask
   latest(IpType) as Type
   latest(IpvServer) as "Bound to vServer"
   latest(IpMgmtAccess) as "Mgmt. access"
   splitrow
      IpAddress as "IP address"
   filter HostName is "ns1"
| eval "Mgmt. access" = if('Mgmt. access'=1,"Yes","No")
| where Type != "NSIP" and 'Mgmt. access' = "Yes"
| table
   "IP address"
   Netmask
   Type
   "Bound to vServer"
   "Mgmt. access"
```

Generating Driver Version Inventory Reports

When users report hardware problems, it is often not the hardware itself that is malfunctioning, but the installed driver. In these situations, it can be invaluable to have an overview of all drivers on your endpoints. Here is how to create driver reports that list names and versions of some or all installed drivers.

The scripts listed in this guide are managed in vast limits'public GitHub repository.

Listing Drivers With uberAgent's Built-In Functionality

uberAgent collects version numbers for all installed applications, including driver packages. The following search, for example, gives you an overview of all installed Nvidia and Intel graphics software:

```
| pivot `uberAgent_index` Application_ApplicationInventory
  dc(host) as "#Installations"
  splitrow
      DisplayName as Name
  splitrow
      DisplayVersion as Version
```

```
    filter DisplayName in ("Nvidia Graphics Driver*","Intel(R) Processor Graphics")
| table
  Name
  Version
  "#Installations"
```

Listing Even More Drivers With Custom Scripts

Anyway, not all drivers are installed via software packages. Some are e.g. installed via Microsoft updates. uberAgent's custom script functionality to the rescue! In a nutshell, it executes any script you like and sends the output to Splunk.

Use this PowerShell script to list all installed drivers:

```
$Output = @{
  }

$DriverPackages = $null

$DriverPackages = Get-WmiObject Win32_PnPSignedDriver | select devicename, driverversion, driverprovidername | where-object {
 $PSItem.driverprovidername -notlike "" -and $PSItem.driverprovidername -notlike "*Microsoft*" }


Foreach ($DriverPackage in $DriverPackages)
{

    # Do some formatting for Intel drivers as the vendor name is not consistent
    If ($DriverPackage.driverprovidername -like "*Intel*")
    {

        $DriverPackage.driverprovidername = "Intel"
     }

    $Output = @{

       'DeviceName' = "`"$($DriverPackage.devicename)`""
       'DriverVersion' = $DriverPackage.driverversion
       'DriverVendor' = "`"$($DriverPackage.driverprovidername)`""
     }

    Write-Output $($Output.Keys.ForEach({
 "$_=$($Output.$_)" }
 ) -join ' ')
 }
```

I denylisted everything Microsoft related reducing the list to third-party drivers only. Another way would be to just include specific vendors:

```
$Output = @{
  }

$DriverPackages = $null

$DriverPackages = Get-WmiObject Win32_PnPSignedDriver | select devicename, driverversion, driverprovidername | where-object {
 $PSItem.driverprovidername -like "*Intel*" -or $PSItem.driverprovidername -like "*Lenovo*" }


Foreach ($DriverPackage in $DriverPackages)
{

    # Do some formatting for Intel drivers as the vendor name is not consistent
    If ($DriverPackage.driverprovidername -like "*Intel*")
    {

        $DriverPackage.driverprovidername = "Intel"
     }

    $Output = @{

       'DeviceName' = "`"$($DriverPackage.devicename)`""
       'DriverVersion' = $DriverPackage.driverversion
       'DriverVendor' = "`"$($DriverPackage.driverprovidername)`""
     }

    Write-Output $($Output.Keys.ForEach({
 "$_=$($Output.$_)" }
 ) -join ' ')
 }
```

In any case, filtering is recommended to sort out unneeded drivers and keep the indexed data volume as small as possible.

Let us stick to the first example script and have a look at the output:

```
DriverVendor="Lenovo" DeviceName="System Interface Foundation V2 Device " DriverVersion=1.1.17.2
DriverVendor="Synaptics" DeviceName="Synaptics HID-Compliant Touch pad Device" DriverVersion=19.3.4.219
DriverVendor="Intel" DeviceName="Intel(R) Software Guard Extensions Platform Software Component" DriverVersion=2.1.100.46245
DriverVendor="Intel" DeviceName="Intel(R) Software Guard Extensions Device" DriverVersion=1.9.103.38781
DriverVendor="Intel" DeviceName="Intel(R) Ethernet Connection (4) I219-V" DriverVersion=12.15.24.1
DriverVendor="Synaptics" DeviceName="Synaptics SMBus Driver" DriverVersion=19.3.4.219
DriverVendor="Intel" DeviceName="Intel(R) Display-Audio" DriverVersion=10.22.1.97
```

```
DriverVendor="Realtek Semiconductor Corp." DeviceName="Realtek High Definition Audio" DriverVersion=6.0.1.8551
DriverVendor="Intel" DeviceName="Mobile 6th/7th Generation Intel(R) Processor Family I/O PMC - 9D21" DriverVersion=10.1.1.38
DriverVendor="Synaptics" DeviceName="Synaptics Pointing Device" DriverVersion=19.3.4.219
DriverVendor="Lenovo" DeviceName="Lenovo Power Manager" DriverVersion=10.0.56.0
DriverVendor="Lenovo" DeviceName="Lenovo PM Device" DriverVersion=1.67.12.23
DriverVendor="Intel" DeviceName="Mobile 7th Generation Intel(R) Processor Family I/O LPC Controller (U Premium) - 9D58" DriverVersion=10.1.1.38
DriverVendor="Intel" DeviceName="Mobile 6th/7th Generation Intel(R) Processor Family I/O PCI Express Root Port #9 - 9D18" DriverVersion=10.1.1.38
DriverVendor="Intel" DeviceName="Intel(R) Dual Band Wireless-AC 8265" DriverVersion=20.70.3.3
DriverVendor="Intel" DeviceName="Mobile 6th/7th Generation Intel(R) Processor Family I/O PCI Express Root Port #3 - 9D12" DriverVersion=10.1.1.38
DriverVendor="Intel" DeviceName="Mobile 6th/7th Generation Intel(R) Processor Family I/O PCI Express Root Port #1 - 9D10" DriverVersion=10.1.1.38
DriverVendor="Intel" DeviceName="Intel(R) Management Engine Interface " DriverVersion=11.7.0.1040
DriverVendor="Intel" DeviceName="Mobile 6th/7th Generation Intel(R) Processor Family I/O Thermal subsystem - 9D31" DriverVersion=10.1.1.38
DriverVendor="SunplusIT" DeviceName="Integrated Camera" DriverVersion=3.5.7.4802
DriverVendor="Intel" DeviceName="Intel(R) Wireless Bluetooth(R)" DriverVersion=19.30.1648.920
DriverVendor="Sierra Wireless Inc." DeviceName="Sierra Wireless EM7455 Qualcomm Snapdragon swmbbnode device 01" DriverVersion=17.3.2.9
DriverVendor="AlcorMicro" DeviceName="Alcor Micro USB Smart Card Reader " DriverVersion=1.7.45.15
DriverVendor="Logitech" DeviceName="Logitech USB Input Device" DriverVersion=1.10.78.0
DriverVendor="Realtek Semiconductor Corp." DeviceName="Realtek USB 3.0 Card Reader" DriverVersion=10.0.14393.31228
DriverVendor="Lenovo" DeviceName="Wide viewing angle & High density FlexView Display 2560x1440" DriverVersion=6.3.0.0
DriverVendor="Intel" DeviceName="Intel(R) HD Graphics 620" DriverVersion=21.20.16.4590
DriverVendor="Intel" DeviceName="Intel(R) Xeon(R) E3 - 1200 v6/7th Gen Intel(R) Core(TM) Host Bridge/DRAM Registers - 5904" DriverVersion=10.1.1.38
DriverVendor="Sierra Wireless Incorporated" DeviceName="Sierra Wireless Location Sensor" DriverVersion=17.8.1030.33
```

Start Collecting Data

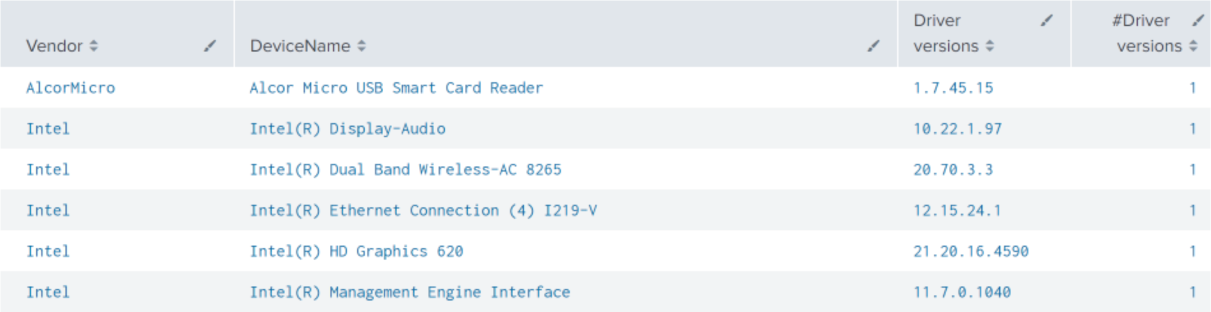
Save the script to disk and create a new timer in uberAgent's configuration. After a service restart, uberAgent starts collecting data.

```
[Timer]
Name                 = PowerShell Driver Versions
Interval             = 86400000
Start delay          = 600000
Persist interval     = true
Thread priority      = background
Script               = powershell.exe -executionpolicy bypass -file "C:\
    Program Files\vast limits\uberAgent\Scripts\Get-DriverVersions.ps1"
ScriptContext        = Session0AsSystem
```

Splunk it

Once the data is in Splunk you can use it to help you troubleshooting these nasty driver issues. Run the following search to list all driver versions per device:

```
index = `uberAgent_index` sourcetype = "\"uberAgent:Script:PowerShell
    Driver Versions\""
| stats
   values(DriverVersion) as "Driver versions"
   dc(DriverVersion) as "#Driver versions"
   latest(DriverVendor) as Vendor
      by DeviceName
| sort limit=0 Vendor
| table
   Vendor
   DeviceName
   "Driver versions"
   "#Driver versions"
```



Vendor	DeviceName	Driver versions	#Driver versions
AlcorMicro	Alcor Micro USB Smart Card Reader	1.7.45.15	1
Intel	Intel(R) Display-Audio	10.22.1.97	1
Intel	Intel(R) Dual Band Wireless-AC 8265	20.70.3.3	1
Intel	Intel(R) Ethernet Connection (4) I219-V	12.15.24.1	1
Intel	Intel(R) HD Graphics 620	21.20.16.4590	1
Intel	Intel(R) Management Engine Interface	11.7.0.1040	1

Note that the number of driver versions is always one as I ran this only on my laptop. The number will likely change when running the script in a corporate environment on multiple machines.

Querying Windows Performance Counters

In addition to its built-in metrics, uberAgent can collect data from Windows performance counters.

Configuration

As with all other uberAgent metrics, performance counters need to be associated with a timer that defines how often data is collected. Performance counter metrics can be added to an existing timer or placed in a new timer section. See the configuration docs and the performance counter metrics documentation for details.

Example

In the following example, we configured a new timer that collects data every 60 seconds. We added two performance counters to the timer that collect .NET information for the entire machine and for a specific process (PowerShell). We also added a performance counter that collects the system's uptime. Finally, we added a performance counter that collects the handle count for every running process. The result looks like this:

```
[Timer]
Name             = Performance counter timer
Interval         = 60000
Perf counter     = \.NET CLR Memory(_Global_)\# Gen 0 Collections
Perf counter     = \.NET CLR Memory(powershell)\# Gen 0 Collections
Perf counter     = \System\System Up Time
Perf counter     = \Process(*)\Handle Count
```

Counter Paths

Most performance counter paths have one of the following two formats:

```
\object(instance)\counter
\object\counter
```

As you can see, some counter paths have instance names (`powershell` in one of the examples above) while others do not.

See Microsoft Docs for more information.

Wildcards

uberAgent can use the wildcard characters `*` and `?` in the instance section of performance counter paths.

Finding Counter Paths

To find the paths for your counters you can use the following method.

Add the desired counters to Performance Monitor. Then right-click the graph section of Perfmon's window, select *Save settings as…*, save as web page (`*.html`) and locate the counter path in the HTML code. Perfmon saves the counter path in language-neutral format, which is what is uberAgent needs.

Language-Neutral or Localized

Some tools require localized performance counter paths. This becomes a problem if you configure paths for machines with different language versions of Windows.

With uberAgent, you can configure localized or language-neutral performance counter paths. The performance counter path is always sent to the backend in English.

Documenting Applied Computer GPOs

Managing thousands of machines is a tough job for which many organizations are using Group Policy. If you are familiar with uberAgent you may know that it lists GPOs that are applied during user logon. Computer policies, on the other hand, are not part of uberAgent's regular feature set. However, that can easily be remedied with the help of uberAgent's custom script functionality. This article shows how.

The scripts listed in this guide are managed in vast limits'public GitHub repository.

Listing Applied Machine Policies

In a nutshell, the uberAgent custom scripts feature executes any script you like and sends the output to Splunk. This works with any scripting engine, of course. In this case, we used PowerShell.

The following script lists all applied machine policies.

```
# We need a temporary XML file to store the data
$OutputFile = "C:\Windows\Temp\TempGPOExport.xml"

# If the file exists delete it
If (Test-Path $OutputFile)
{

   Remove-Item $OutputFile -Force
 }

```

```

# Run gpresult and store the result in the temporary xml file
gpresult.exe /Scope Computer /X $OutputFile /F

# Read the xml file
[xml]$XML = Get-Content $OutputFile

# Get the names of all GPOs
$GPOs = ($XML.Rsop.ComputerResults.GPO).Name | Sort-Object

# Remove the temporary file
Remove-Item $OutputFile -Force

# Join all GPOs to a long string
$GPOs = $GPOs -join ';'

# Write the output. uberAgent will pick this up.
Write-Output "MachineGPOs=`"$GPOs`""
```

Save the script somewhere. For this example, we save the script as `C:\Program Files\vast limits\uberAgent\Scripts\Get-MachineGPOs.ps1`.

Configure uberAgent to Run the Script

Create a new timer in uberAgent's configuration. With the settings shown below, the script will be executed once per day.

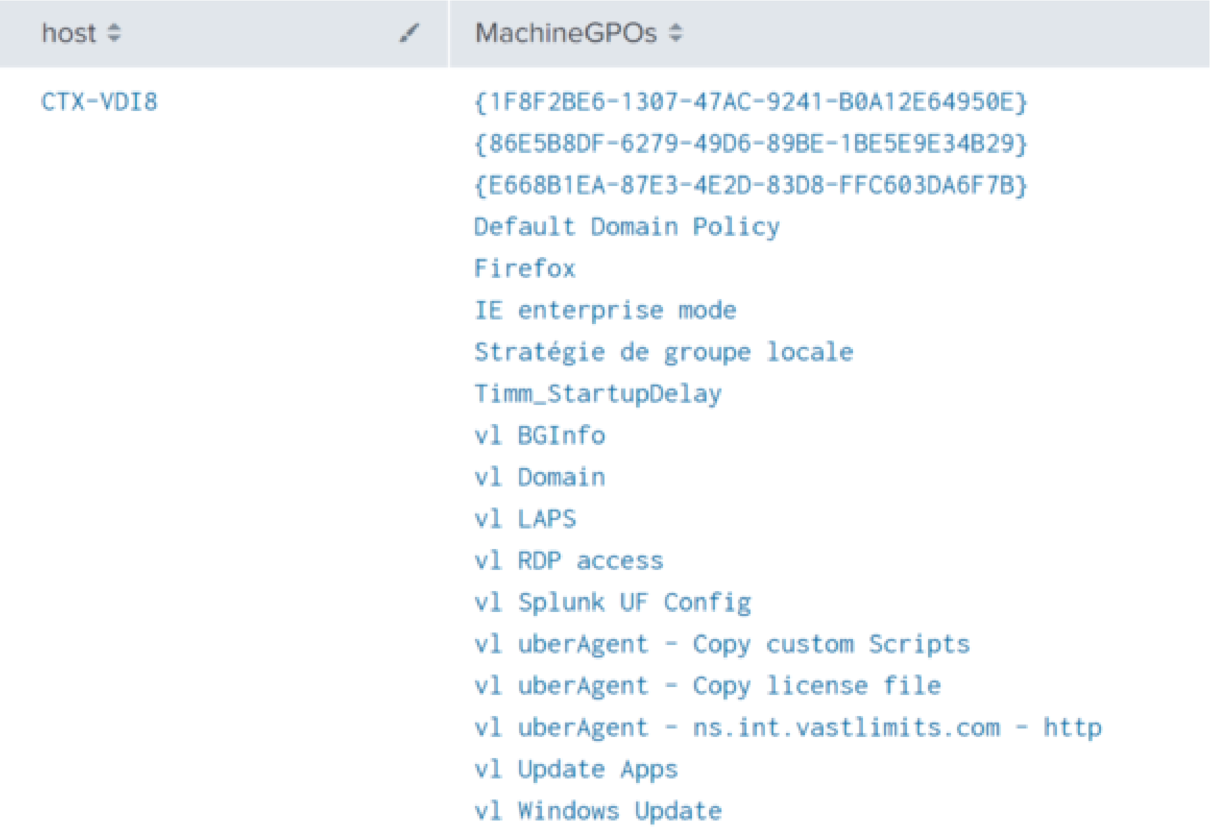
```
[Timer]
Name = Get-MachineGPOs
Interval = 86400000
Start delay = 600000
Persist interval = true
Thread priority = background
Script = powershell.exe -executionpolicy bypass -file "C:\Program Files\vast limits\uberAgent\Scripts\Get-MachineGPOs.ps1"
ScriptContext = Session0AsSystem
```

Splunk it

Once the data is in Splunk it can be used to document the applied computer GPOs. Run the following search to list all group policy objects per machine.

```
index = `uberAgent_index` sourcetype = "\"uberAgent:Script:Get-MachineGPOs\""
| stats
   latest(MachineGPOs) as MachineGPOs
      by host
| eval MachineGPOs = split(MachineGPOs,";")
| table
   host
   MachineGPOs
```

The results from the above search look like this:



host	MachineGPOs
CTX-VDI8	{1F8F2BE6-1307-47AC-9241-B0A12E64950E} {86E5B8DF-6279-49D6-89BE-1BE5E9E34B29} {E668B1EA-87E3-4E2D-83D8-FFC603DA6F7B} Default Domain Policy Firefox IE enterprise mode Stratégie de groupe locale Timm_StartupDelay vl BGInfo vl Domain vl LAPS vl RDP access vl Splunk UF Config vl uberAgent - Copy custom Scripts vl uberAgent - Copy license file vl uberAgent - ns.int.vastlimits.com - http vl Update Apps vl Windows Update

Identifying Applications That Use 100% of a CPU Core

Some applications fully use one or multiple CPU cores, either because they are badly written or because they simply make such complex calculations. Back in the days of single-core CPUs, this was easy to detect: the machine's CPU usage would spike to 100%. Today, even lightweight laptops have at least four physical cores, workstations many more. Looking at a machine's or a process'CPU usage percentage is not enough. You need to factor in the number of CPU cores in order to determine whether one or multiple cores are fully utilized. This article explains how to do that.

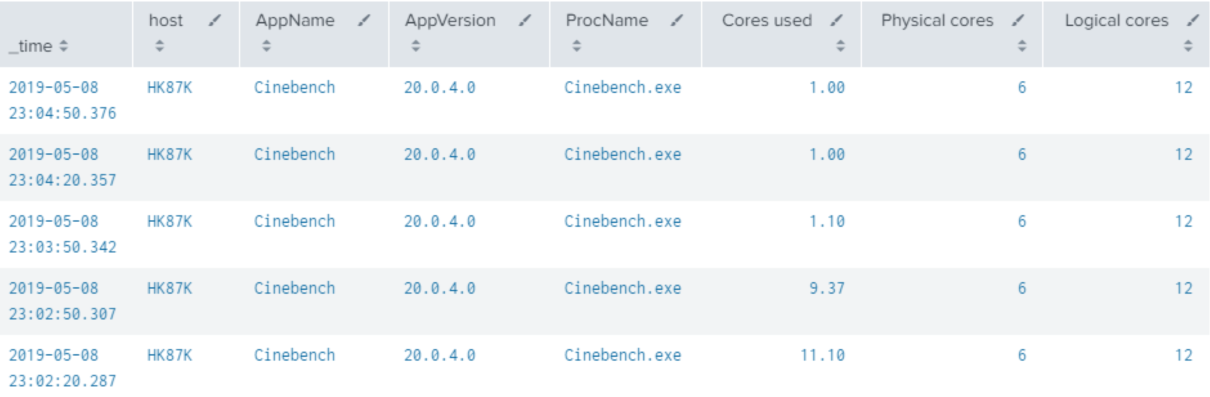
uberAgent determines the number of physical and logical CPU cores as part of its machine inventory metric. The resulting data is then added to all sourcetypes as additional fields during search time via automatic lookups. This means that you can use (nearly) all machine inventory fields in (nearly) all searches! The main exceptions are sourcetypes where the data does not apply to a specific endpoint, e.g., Citrix site or ADC metrics.

With that knowledge, we can build a simple search that lists processes that utilized at least a given number of CPU cores for the duration of a data collection interval (30 seconds by default):

```
index=`uberAgent_index` sourcetype=uberAgent:Process:ProcessDetail ProcCPUPercent>0
| eval CoresUsed = ProcCPUPercent * CPUCoresLogical / 100
| where CoresUsed > 0.8
| table _time, host, AppName, AppVersion, ProcName, CoresUsed, CPUCoresPhysical, CPUCoresLogical
```

In the example above, I filtered for events where one process used at least 80% of a CPU core (`where CoresUsed > 0.8`). The actual calculation of the number of cores used is simple: `CoresUsed = ProcCPUPercent * CPUCoresLogical / 100`. The last command, `table`, selects the fields to be displayed: application name, application version, process name, number of cores used, total number of physical and logical cores.

The result looks like this:



_time	host	AppName	AppVersion	ProcName	Cores used	Physical cores	Logical cores
2019-05-08 23:04:50.376	HK87K	Cinebench	20.0.4.0	Cinebench.exe	1.00	6	12
2019-05-08 23:04:20.357	HK87K	Cinebench	20.0.4.0	Cinebench.exe	1.00	6	12
2019-05-08 23:03:50.342	HK87K	Cinebench	20.0.4.0	Cinebench.exe	1.10	6	12
2019-05-08 23:02:50.307	HK87K	Cinebench	20.0.4.0	Cinebench.exe	9.37	6	12
2019-05-08 23:02:20.287	HK87K	Cinebench	20.0.4.0	Cinebench.exe	11.10	6	12

You may notice that I ran a CPU benchmark, Cinebench, to generate a high enough load. First I ran the multi-core, then the single-core benchmark. Please note that I renamed some fields to optimize formatting.

Collecting the Processor Temperature With uberAgent

The other day a customer asked if uberAget collects the CPU temperature. By default, it doesn't, but you can easily add the CPU temperature as a metric via uberAgent's custom script functionality. Here is how.

In a nutshell, uberAgent's custom script functionality allows you to execute arbitrary scripts whose output is sent to the backend (e.g., Splunk). Go here to get more details on the configuration.

The scripts listed in this guide are managed in vast limits' public GitHub repository.

Choose Your Preferred Data Source

The processor temperature is stored in one of these WMI objects, or both:

1. `MSAcpi_ThermalZoneTemperature`
2. `Win32_PerfFormattedData_Counters_ThermalZoneInformation`

On some test machines, the values differed between the two objects. We compared them with the values from other tools like HWInfo and found that `Win32_PerfFormattedData_Counters_ThermalZoneIn` is the better option for our machines. That could be different for yours, though.

Either way, your machines have to support at least one of these WMI classes. If your PCs are equipped with Intel processors, they will likely do. If not, you will get the error `Get-WMIObject : Not supported` when querying WMI.

The PowerShell script gives you the temperature in Celsius, Fahrenheit, and Kelvin. Choose your preferred unit by commenting out the other unnecessary rows.

```powershell
<#
  .SYNOPSIS
  uberAgent script to determine the current CPU temperature.

  .DESCRIPTION
  Reads the current CPU temperature from WMI and converts the resulting output to the KV format required by uberAgent custom scripts.

  If the default WMI class does not yield satisfactory resulty, try switching to the alternative data source by commenting out the relevant lines.

  Most machines have multiple thermal zones. Test with your PCs and enter the name of the appropriate thermal zone in the variable $thermalZone.
  To retrieve a list of all thermal zones run either of the following:
  - Get-CimInstance -ClassName Win32_PerfFormattedData_Counters_ThermalZoneInformation
  - Get-CimInstance -ClassName MSAcpi_ThermalZoneTemperature -Namespace "root/wmi"
#>

# Thermal zone (wildcard string)
$thermalZone = "*tz00*"

# Default: get the CPU temperature from the WMI class Win32_PerfFormattedData_Counters_ThermalZoneInformation
$temp = Get-CimInstance -ClassName Win32_PerfFormattedData_Counters_ThermalZoneInformation | Where-Object -Property Name -like $thermalZone
$TempKelvin     = $temp.Temperature

# Alternative: get the CPU temperature from the WMI class MSAcpi_ThermalZoneTemperature
# Note: requires elevation (admin rights)
# $temp = Get-CimInstance -ClassName MSAcpi_ThermalZoneTemperature -Namespace "root/wmi" | Where-Object -Property InstanceName -like $thermalZone
# $TempKelvin     = $temp.Temperature / 10

$TempKelvin     = $temp.Temperature
$TempCelsius    = $TempKelvin - 273.15
$TempFahrenheit = (9/5) * $TempCelsius + 32

$Output = @{

   # delete rows which are not needed
   'TempCelsius' = [math]::Round($TempCelsius)
   'TempFahrenheit' = [math]::Round($TempFahrenheit)
   'TempKelvin' = [math]::Round($TempKelvin)
 }

Write-Output $($Output.Keys.ForEach({
 "$_=$($Output.$_)" }
 ) -join ' ')
```

The output of the script looks like this: `TempCelsius=65 TempFahrenheit=149 TempKelvin=338`

Configure uberAgent to Start Querying

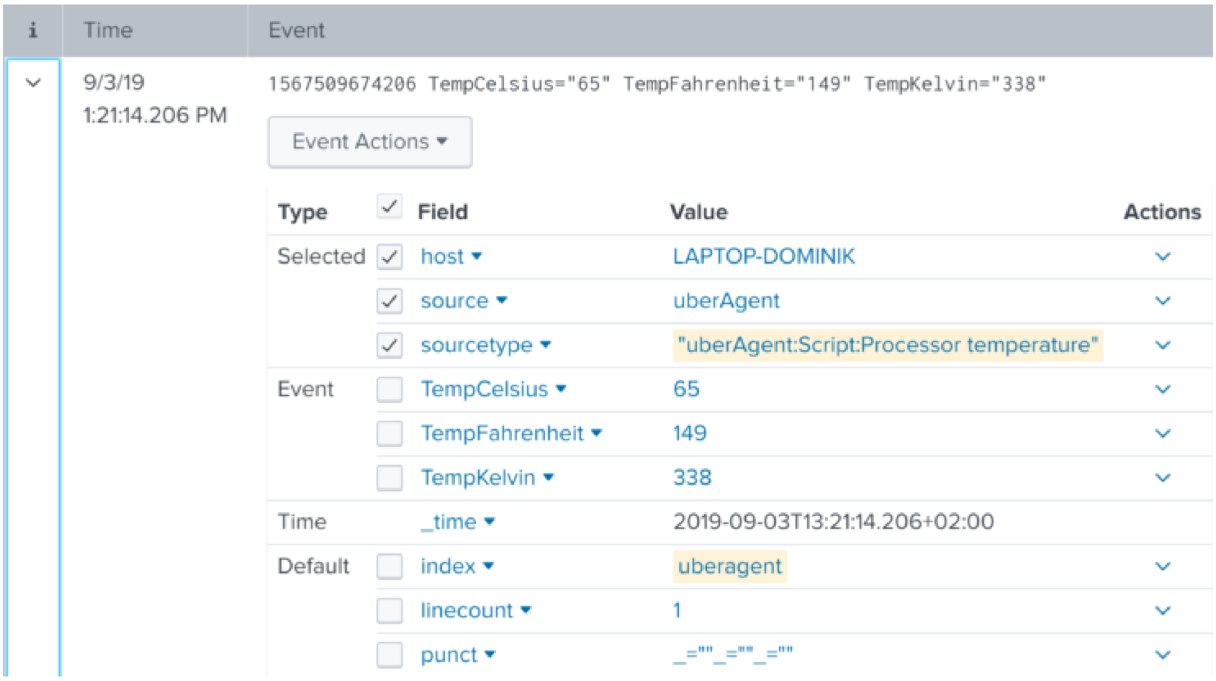
Store your preferred script somewhere on the target machines. We went with `C:\Program Files\vast limits\uberAgent\Scripts\Get-ProcessorTemperature.ps1`.

Next step is to create a new timer in uberAgent's configuration, like this:

```
############################################
# Timer 12
############################################
[Timer]
Name = Processor temperature
Comment = Collect processor temperature with a custom script
# Run every 5 minutes
Interval = 300000
# Run the first time 10 minutes after boot to give the machine time to get into a normal working state
Start delay = 600000
Script = powershell.exe -executionpolicy bypass -file "C:\Program Files\vast limits\uberAgent\Scripts\Get-ProcessorTemperature.ps1"
# Run this script with system permissions
ScriptContext = Session0AsSystem
```

Splunk it

Now you have the processor's temperature in Splunk and can relate it to the machine's performance.



i	Time	Event
	9/3/19 1:21:14.206 PM	1567509674206 TempCelsius="65" TempFahrenheit="149" TempKelvin="338"

Type	☑	Field	Value	Actions
Selected	☑	host ▾	LAPTOP-DOMINIK	⌄
	☑	source ▾	uberAgent	⌄
	☑	sourcetype ▾	"uberAgent:Script:Processor temperature"	⌄
Event	□	TempCelsius ▾	65	⌄
	□	TempFahrenheit ▾	149	⌄
	□	TempKelvin ▾	338	⌄
Time		_time ▾	2019-09-03T13:21:14.206+02:00	
Default	□	index ▾	uberagent	⌄
	□	linecount ▾	1	⌄
	□	punct ▾	_=""_=""_=""	⌄

Please note that there are more advanced technologies to get the processor temperature, like Intel DTS. But using these with PowerShell would go beyond the scope of a practice guide.

User & Host Tagging Examples

With uberAgent 5.3 comes a very powerful and flexible tagging feature which lets you enrich uberAgent's dataset with your own custom data. While there is a feature description in the advanced topics section, we wanted to share a comprehensive list of tags you may find useful which you can copy and paste without further ado. Please read the feature description first to get the most out of the following list.

Operating System Language

```
# The language the operating system was installed with
[UserHostTagging]
Tag name = Host install language
Tag type = Host
Tag source = Registry
Tag value = HKLM\SYSTEM\CurrentControlSet\Control\Nls\Language\
    InstallLanguage
```

```

# The default language for the operating system
[UserHostTagging]
Tag name = Host default language
Tag type = Host
Tag source = Registry
Tag value = HKLM\SYSTEM\CurrentControlSet\Control\Nls\Language\Default

# The user's preferred language
[UserHostTagging]
Tag name = User preferred language
Tag type = User
Tag source = Registry
Tag value = HKCU\Software\Microsoft\CTF\SortOrder\Language\00000000
```

Windows 10 Version

```
# Windows 10 version like 1709 or 1909. More information: https://docs.microsoft.com/en-us/windows/release-information/
[UserHostTagging]
Tag name = WindowsVersion
Tag type = Host
Tag source = Registry
Tag value = HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ReleaseId
```

User Name

```
# uberAgent collects the user name in the format domain\username. This tag gives you the user name without the domain.
[UserHostTagging]
Tag name = Username
Tag type = User
Tag source = Environment
Tag value = USERNAME
```

User Domain

```
# uberAgent collects the user name in the format domain\username. This tag gives you the user's domain without the user name.
[UserHostTagging]
Tag name = Userdomain
Tag type = User
Tag source = Environment
Tag value = USERDOMAIN
```

User Principal Name

```
# The user's principal name (UPN). Example: johndoe@company.com
[UserHostTagging]
Tag name = UPN
Tag type = User
Tag source = Ad
Tag value = userPrincipalName
```

User Company

```
# The user's company. The AD attribute company has to be set for this
    to work.
[UserHostTagging]
Tag name = User company
Tag type = User
Tag source = Ad
Tag value = company
```

User Department

```
# The user's department. The AD attribute department has to be set for
    this to work.
[UserHostTagging]
Tag name = User department
Tag type = User
Tag source = Ad
Tag value = department
```

User AD Organizational Unit

```
[UserHostTagging]
Tag name = User OU
Tag type = User
Tag source = Ad
Tag value = distinguishedName
```

User DNS Domain

```
# This gives you the user's full qualified DNS domain name
[UserHostTagging]
Tag name = User DNS domain
Tag type = User
Tag source = Environment
Tag value = USERDNSDOMAIN
```

User Account Control

```
# Account options. 512 = activated, 514 = deactivated, 66048 = password never expires. More information: https://support.microsoft.com/en-us/help/305144/how-to-use-useraccountcontrol-to-manipulate-user-account-properties
[UserHostTagging]
Tag name = User account control
Tag type = User
Tag source = Ad
Tag value = userAccountControl
```

Querying Windows Event Log

A question we get asked quite often is whether uberAgent is able to query Windows Event Log for specific events IDs. This functionality is not in the product, as Splunk, uberAgent's primary backend, has this built-in. Hence we generally recommend making use of Splunk's capabilities.

However, not all of our customers have Universal Forwarder, Splunk's client software, deployed to their endpoints, mostly for performance reasons. If that is the case for you, too, then this practice guide should be of interest. If you do have Splunk's Universal Forwarder deployed, please use its functionality instead. UF is more efficient than the script-based solution presented in this article.

The scripts listed in this guide are managed in vast limits' public GitHub repository.

Collect Event IDs With a Custom Script

As Windows Event Log data is not part of uberAgent's default data set, we are using the agent's custom script functionality.

In a nutshell, the uberAgent custom scripts feature executes any script you like and sends the output to Splunk. This works with any scripting engine, of course. In this case, we are using PowerShell.

The following script queries Event Log for a given event ID in a specific event log (Application, System, etc.). It also needs an interval in which it should search. Due to the parameters, you can use that script to query arbitrary events and don't need a new script for each. The required parameters are passed to the script through an uberAgent timer.

```
PARAM
(
    [Parameter(Mandatory = $true)]
    [System.Int32]$EventID,

    [Parameter(Mandatory = $true)]
    [System.String]$EventLog,

    [Parameter(Mandatory = $true)]
    [System.Int32]$IntervalMS
)

$IntervalMS = -$IntervalMS

$StartTime=$(get-date).AddMilliseconds($IntervalMS)

$Filter= @{

    LogName="$EventLog"
    StartTime=$StartTime
    Id=$EventID
 }


$Events = Get-WinEvent -FilterHashtable $Filter -ErrorAction SilentlyContinue

if (@($Events).Count -gt 0)
{

    foreach ($Event in $Events)
    {

        [Hashtable]$Output = @{

            'ProviderName' = "`"$($Event.ProviderName)`""
            'Id' = $Event.id
            'LevelDisplayName' = $Event.LevelDisplayName
            'LogName' = $Event.LogName
            'Message' = "`"$($Event.Message)`""
            'TimeCreated' = "`"$($Event.TimeCreated)`""
            'TaskDisplayName' = "`"$($Event.TaskDisplayName)`""
         }

        Write-Output $($Output.keys.foreach({
 "$_=$($Output.$_)" }
 ) -join ' ')

     }

 }
```

Save the script somewhere. For this example, we save the script as `C:\Program Files\vast limits\uberAgent\Scripts\Get-WinEvent.ps1`.

Configure uberAgent to Run the Script

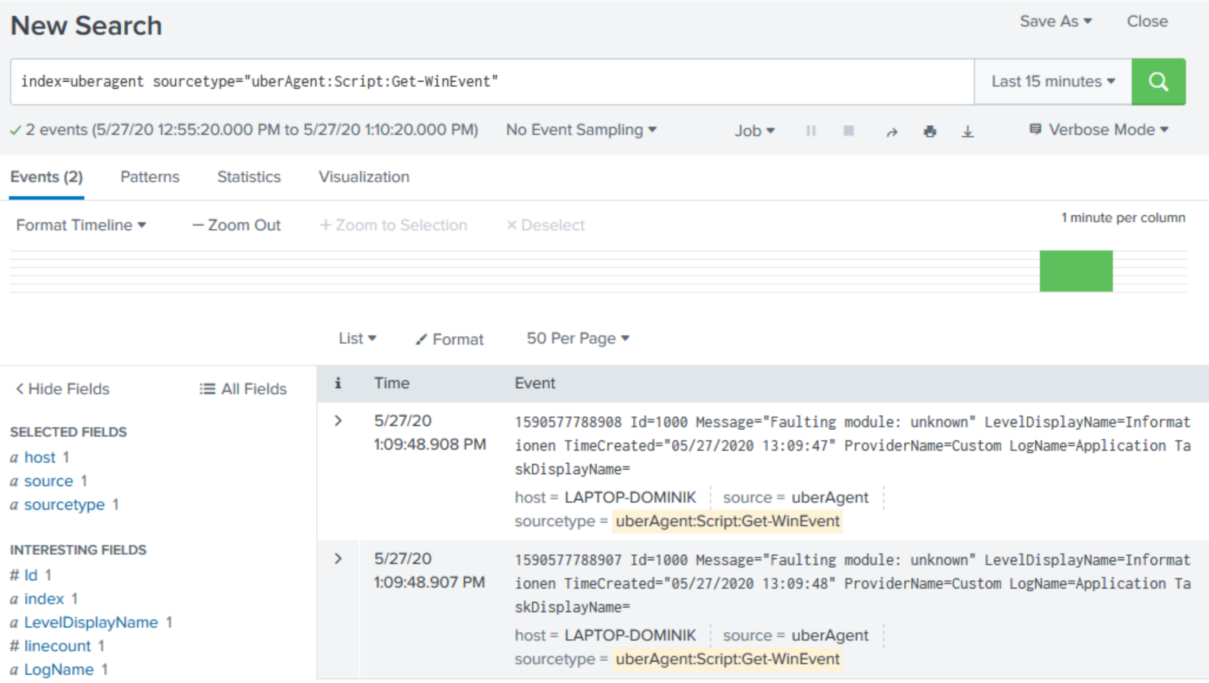
Create a new timer in uberAgent's configuration. With the settings shown below, the script will be executed every five minutes.

```
[Timer]
Name                 = Get-WinEvent
Interval             = 300000
Script               = powershell.exe -executionpolicy bypass -file "C:\Program Files\vast limits\uberAgent\Scripts\Get-WinEvent.ps1" -EventID 1000 -EventLog "Application" -IntervalMS 300000
ScriptContext        = Session0AsSystem
```

Note that you define the parameters in the `Script` setting. The parameter `IntervalMS` should be the same number of milliseconds as the setting `Interval`.

Splunk it

Once the data is in Splunk it can be used to search for Windows Events.



```
index=uberagent sourcetype="uberAgent:Script:Get-WinEvent"
```

✓ 2 events (5/27/20 12:55:20.000 PM to 5/27/20 1:10:20.000 PM)

SELECTED FIELDS
- a host 1
- a source 1
- a sourcetype 1

INTERESTING FIELDS
- # Id 1
- a index 1
- a LevelDisplayName 1
- # linecount 1
- a LogName 1

i	Time	Event

>	5/27/20
	1:09:48.908 PM
		1590577788908 Id=1000 Message="Faulting module: unknown" LevelDisplayName=Informationen TimeCreated="05/27/2020 13:09:47" ProviderName=Custom LogName=Application TaskDisplayName=
		host = LAPTOP-DOMINIK source = uberAgent
		sourcetype = uberAgent:Script:Get-WinEvent

>	5/27/20
	1:09:48.907 PM
		1590577788907 Id=1000 Message="Faulting module: unknown" LevelDisplayName=Informationen TimeCreated="05/27/2020 13:09:48" ProviderName=Custom LogName=Application TaskDisplayName=
		host = LAPTOP-DOMINIK source = uberAgent
		sourcetype = uberAgent:Script:Get-WinEvent

Collecting More WiFi Details From WFH Employees

Normally, laptop workers enjoy good quality WiFi or even a LAN network, connected through a docking station, in their company offices. Well, that changed during the Corona pandemic. The majority of employees are now working from home - connected through their personal WiFi and internet connection. That can be challenging for administrators when they have to troubleshoot a problem, as they have no insight into the quality of the employee's network.

uberAgent supports administrators already to identify if a bad network is the cause of an issue. Head over to this blog post to learn how!

One cannot have enough information when it comes to solving problems, though. Custom scripts to the rescue! This practice guide helps you with collecting even more information about WiFi connections.

The scripts listed in this guide are managed in vast limits'public GitHub repository.

Collecting WiFi Information With PowerShell

Data Source: Netsh Command-Line Tool

Of course, using PowerShell is the standard these days when it comes to scripting. Sadly, there is no native PowerShell cmdlet for the information we're interested in. We have to use the good old `netsh` command: `netsh wlan show interface`.

The output looks similar to the following:

```
There is 1 interface on the system:

    Name                   : WLAN
    Description            : Intel(R) Dual Band Wireless-AC 8265
    GUID                   : 1796b603-2174-45f7-8001-c05c4a011617
    Physical address       : ac:ed:5c:02:f0:30
    State                  : connected
    SSID                   : G12
    BSSID                  : cc:ce:1e:2d:b0:86
    Network type           : Infrastructure
    Radio type             : 802.11n
    Authentication         : WPA2-Personal
    Cipher                 : CCMP
    Connection mode        : Profile
    Channel                : 1
    Receive rate (Mbps)    : 144.4
    Transmit rate (Mbps)   : 144.4
    Signal                 : 86%
    Profile                : Dominiks-WiFi

    Hosted network status  : Not available
```

The most interesting bits are the **signal strength**, the **receive/transmit rates**, the **radio type**, and the **authentication type**. With a little work, we can refactor that old-style text output to some useful PowerShell objects we can actually work with.

Converting Netsh Output to PowerShell Objects (EN)

```
$interface = @(netsh wlan show interface)
$hash = $null

# We only support machines with one interface for now. The following line also makes sure that the scripts only proceed if a WiFi interface is found at all.
If ($interface -match 'There is 1 interface on the system')
{

    # The following builds a hash table from the netsh output
    foreach($item in $interface)
    {

        if($item.Contains(':'))
        {

            $hash += @{

                $item.Replace(" ","").split('{
 : }
 ')[0] = $item.Replace(" ","").split('{
 : }
 ')[1]
             }

         }

     }


    # Only connected interfaces are interesting
    If ($hash.State -eq 'connected')
    {

        # Get the WiFi band by looking at the used channel
        If ([int]$hash.Channel -gt 33)
        {

            $Band = '5'
         }

        Else
        {

            $Band = '2.4'
         }


        # Build the output hash
        $Output = @{

           'Signal' = $hash.Signal -replace '%',''
           'Type' = $hash.Radiotype
           'Receiverate' = $hash.'Receiverate(Mbps)'
           'Transmitrate' = $hash.'Transmitrate(Mbps)'
           'Band' = $Band
           'SSID' = "`"$($hash.SSID)`""
           'Authentication' = $hash.Authentication
           'Cipher' = $hash.Cipher
         }


        # Finally, write the hash to stdout. The output will be picked up by uberAgent.
        Write-Output $($Output.Keys.ForEach({
 "$_=$($Output.$_)" }
 ) -join ' ')

     }

    Else
    {

        Throw "Interface $($hash.Name) not connected. Exiting..."
     }

 }

Else
{

    Throw 'Zero or more than one interface on machine. Exiting...'
 }
```

Running that script outputs the following:

```
Authentication=WPA2-Personal Transmitrate=144.4 Signal=86 Type=802.11n SSID="Dominiks-WiFi"Cipher=CCMP Receiverate=144.4 Band=2.4
```

Converting Netsh Output to PowerShell Objects (DE)

Unfortunately, the `netsh` output is language-specific. Hence the above script only works on **English** Windows operating systems. Below is a version for a **German** Windows, too. If you have other languages, please check the script and replace localized words according to your requirements.

```
$interface = @(netsh wlan show interface)
$hash = $null

# We only support machines with one interface for now. The following line also makes sure that the scripts only proceed if a WiFi interface is found at all.
```

```
If ($interface -match 'Es ist 1 Schnittstelle auf dem System vorhanden')
{

    # The following builds a hash table from the netsh output
    foreach($item in $interface)
    {

        if($item.Contains(':'))
        {

            $hash += @{

                $item.Replace(" ","").split('{
 : }
')[0] = $item.Replace(" ","").split('{
 : }
')[1]
             }

         }

     }


    # Only connected interfaces are interesting
    If ($hash.Status -eq 'Verbunden')
    {

        # Get the WiFi band by looking at the used channel
        If ([int]$hash.Kanal -gt 33)
        {

            $Band = '5'
         }

        Else
        {

            $Band = '2.4'
         }


        # Build the output hash
        $Output = @{

           'Signal' = $hash.Signal -replace '%',''
           'Type' = $hash.Funktyp
           'Receiverate' = $hash.'Empfangsrate(MBit/s)'
           'Transmitrate' = $hash.'Übertragungsrate(MBit/s)'
           'Band' = $Band
           'SSID' = "`"$($hash.SSID)`""
           'Authentication' = $hash.Authentifizierung
```

```
                'Cipher' = $hash.Verschlüsselung
              }


            # Finally, write the hash to stdout. The output will be picked
                up by uberAgent.
            Write-Output $($Output.Keys.ForEach({
    "$_=$($Output.$_)" }
    ) -join ' ')

        }

      Else
      {

          Throw "Interface $($hash.Name) not connected. Exiting..."
        }

    }

  Else
  {

      Throw 'Zero or more than one interface on machine. Exiting...'
    }
```

Start Collecting Data

Save the script to disk and create a new timer in uberAgent's configuration. After a service restart, uberAgent starts collecting data.

```
[Timer]
Name                  = Get-WifiInfo
Interval              = 60000
Script                = powershell.exe -executionpolicy bypass -file "C:\
    Program Files\vast limits\uberAgent\Scripts\Get-WifiInfo.ps1"
ScriptContext         = Session0AsSystem
```

The interval at which the script is run in the timer configuration above is 60 seconds; adjust it to your needs as required. Just remember that the value is specified in milliseconds.

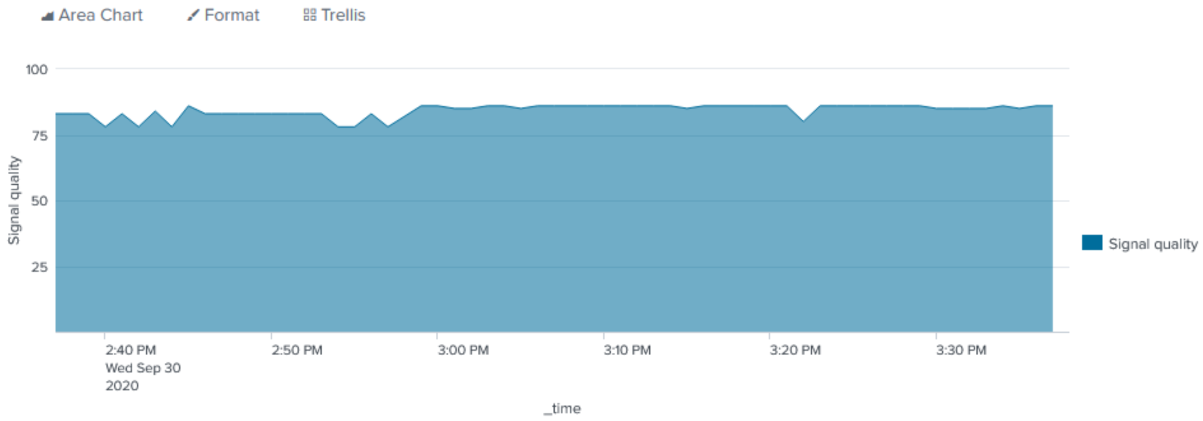
Splunk it

Once the data is in Splunk you can use it to help you troubleshoot issues in WFH scenarios even easier.

Let's assume you get a call by a WFH user who complains about a slow application. With the following search, you can quickly check if the user's WiFi signal strength is low, causing the slowness of the application.

```
index=uberagent sourcetype="uberAgent:Script:Get-WifiInfo"host="
LAPTOP-DOMINIK"| timechart latest(Signal)as "Signal quality (%)"
```

In the author's case, the signal quality looks okay, but could be better:



Signal quality is only one of the metrics we collected. Go and try one of the others for yourself!

Monitoring Windows Update Performance

Use the Splunk search below to list the duration of Windows Update activities that art part of the machine boot process. In addition to the Windows Update processing time, the search returns the disk IO incurred by the update process.

Why Measuring Windows Update Performance Is Crucial

A fundamental component of good endpoint management is Windows Update management. All devices must be kept up to date in order to provide as few opportunities as possible for malware to attack.

However, applying Windows updates also means stopping work for users, because typically the computer has to be restarted for this purpose. This is because Windows updates are installed during the computer startup.

Administrators are therefore keen to ensure that everything runs smoothly during the installation of updates in order to minimize the waiting time for users. So they need a way to monitor the performance of the update installation.

How uberAgent Can Help

uberAgent provides a deep insight into the computer startup performance of Windows endpoints in the dashboard *Boot Duration*. Every computer startup is divided into phases based on the Microsoft document Windows On/Off Transitions Solutions Guide. The installation of Windows updates is not a separate phase, but only a sub-phase.

Nevertheless, the installation of Windows Updates can be monitored with uberAgent, because uberAgent determines the runtime, command line and performance for each process that ran during boot.

Windows updates are installed by the *Windows Update* service. This is an instance of the `svchost.exe` process and can be identified by the following command line: `svchost.exe -k netsvcs -p -s wuauserv`.

The performance of processes at boot is best measured by the number of disk accesses (IOs). Many IOs means that the application or service has a negative impact on boot performance.

Splunk Search

The search below returns the following results for each *Windows Update* service startup:

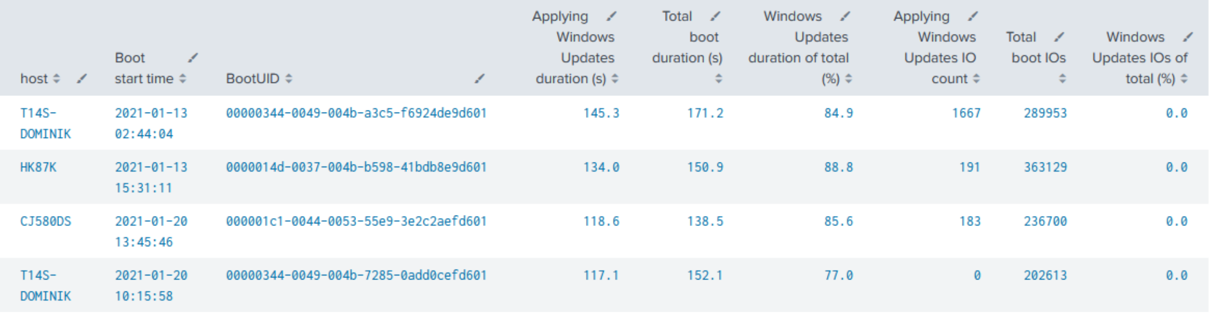
- The computer name
- The unique ID generated by uberAgent to analyze the complete computer startup
- The start time of the computer startup
- The duration of the *Windows Update* phase
- The percentage of the total boot time that is taken up by the *Windows Update* phase
- The total duration of the computer startup
- The number of IOs of the *Windows Update* phase
- The percentage of the *Windows Update* phase in the total IOs of the computer startup
- The total number of IOs of the computer startup

```
| pivot `uA_DM_OnOffTransition_BootProcesses` OnOffTransition_BootProcesses
  latest(ProcLifetimeMs) as ProcLifetimeMs
  latest(TotalBootDurationMs) as TotalBootDurationMs
  sum(ProcIOReadCount) as SumProcIOReadCount
  sum(ProcIOWriteCount) as SumProcIOWriteCount
  splitrow BootUID
  splitrow host
  splitrow ProcCmdline
  filter ProcCmdline is "*svchost.exe -k netsvcs -p -s wuauserv"
| join type=left BootUID
[
   | pivot `uA_DM_OnOffTransition_BootProcesses` OnOffTransition_BootProcesses
```

```
      sum(ProcIOReadCount) as SumTotalBootProcIOReadCount
      sum(ProcIOWriteCount) as SumTotalBootProcIOWriteCount
      splitrow BootUID
   | fields + BootUID SumTotalBootProcIOReadCount SumTotalBootProcIOWriteCount
]
| join type=left BootUID
[
   | pivot `uA_DM_OnOffTransition_BootDetail` OnOffTransition_BootDetail
      latest(_time) as BootStartTime
      splitrow BootUID
   | eval "Boot start time"=strftime(strptime(BootStartTime, "%Y-%m-%dT%H:%M:%S.%Q%z"), "%Y-%m-%d %H:%M:%S")
   | fields + BootUID "Boot start time"
]
| eval "Applying Windows Updates duration (s)" = round(ProcLifetimeMs / 1000,1)
| eval "Total boot duration (s)" = round(TotalBootDurationMs / 1000,1)
| eval "Windows Updates duration of total (%)" = round(ProcLifetimeMs / TotalBootDurationMs * 100,1)

| eval "Applying Windows Updates IO count" = SumProcIOReadCount + SumProcIOWriteCount
| eval "Total boot IOs" = SumTotalBootProcIOReadCount + SumTotalBootProcIOWriteCount
| eval "Windows Updates IOs of total (%)" = round('Applying Windows Updates IO count' / 'Total boot IOs',1)

| sort - "Applying Windows Updates duration (s)"
| table
   host
   "Boot start time"
   BootUID
   "Applying Windows Updates duration (s)"
   "Total boot duration (s)"
   "Windows Updates duration of total (%)"
   "Applying Windows Updates IO count"
   "Total boot IOs"
   "Windows Updates IOs of total (%)"
```

Splunk Search Result

This is what the result looks like in Splunk’s UI:



host	Boot start time	BootUID	Applying Windows Updates duration (s)	Total boot duration (s)	Windows Updates duration of total (%)	Applying Windows Updates IO count	Total boot IOs	Windows Updates IOs of total (%)
T14S-DOMINIK	2021-01-13 02:44:04	00000344-0049-004b-a3c5-f6924de9d601	145.3	171.2	84.9	1667	289953	0.0
HK87K	2021-01-13 15:31:11	0000014d-0037-004b-b598-41bdb8e9d601	134.0	150.9	88.8	191	363129	0.0
CJ580DS	2021-01-20 13:45:46	000001c1-0044-0053-55e9-3e2c2aefd601	118.6	138.5	85.6	183	236700	0.0
T14S-DOMINIK	2021-01-20 10:15:58	00000344-0049-004b-7285-0add0cefd601	117.1	152.1	77.0	0	202613	0.0

Example result for above search

Customize Using Filters

Above's search gives you probably a long list back. By extending the search with the following line, the output can be filtered to computer startups where the Windows Update phase exceeds a certain value in seconds.

```
| where 'Applying Windows Updates duration (s)'> 60
```

Creating a TPM Status Inventory Report

This article shows how to collect detailed status and inventory information about each endpoint's trusted platform module (TPM). The output includes the TPM's state and version and can easily be used in Splunk reports and dashboards.

There are multiple ways to get TPM information. We chose the command-line tool `tpmtool.exe` as our data source because its output is more complete than PowerShell's `Get-Tpm` cmdlet (which lacks the TPM version).

The scripts listed in this guide are managed in vast limits'public GitHub repository.

Collecting TPM Information With PowerShell From Tpmtool

Data Source: Tpmtool.exe

When run with the parameter `getdeviceinformation`, Tpmtool's output looks as follows

```
-TPM Present: True
-TPM Version: 2.0
-TPM Manufacturer ID: INTC
-TPM Manufacturer Full Name: Intel
-TPM Manufacturer Version: 302.12.0.0
-PPI Version: 1.3
-Is Initialized: True
```

```
-Ready For Storage: True
-Ready For Attestation: True
-Is Capable For Attestation: True
-Clear Needed To Recover: False
-Clear Possible: True
-TPM Has Vulnerable Firmware: False
-PCR7 Binding State: 0
-Maintenance Task Complete: True
-TPM Spec Version: 1.16
-TPM Errata Date: Wednesday, September 21, 2016
-PC Client Version: 1.00
-Is Locked Out: False
```

Converting Tpmtool's Output to a Key Value String

We'll use the following PowerShell script to execute `tpmtool` and convert its output to the KV string format required by uberAgent custom scripts:

```
<#
  .SYNOPSIS
  uberAgent script to collect TPM status information.

  .DESCRIPTION
  Runs the command "tpmtool.exe getdeviceinformation" and converts the output to the KV format required by uberAgent custom scripts.
#>

# Run tpmtool
$tpmtoolOutput = @(tpmtool getdeviceinformation)

# We'll store the output fields here
$output = @{
  }


foreach ($line in $tpmtoolOutput)
{

   # Ignore empty lines
   if (!$line) {
  continue  }


   # Remove the leading dash
   $line = $line -replace "^-", ""

   # Split at the colon
   $kv = $line.Split(":")

   if ($kv.Count -ne 2) {
  continue  }
```

```


   # Remove the leading space in the value
   $kv[1] = $kv[1] -replace "^\s+", ""

   # Ignore certain fields
   if ($kv[0] -like "*spec version*" -or $kv[0] -like "*errata date*") {
  continue  }


   # Remove spaces in the key name
   $kv[0] = $kv[0] -replace "\s+", ""

   # Add the field to the output hashtable
   $output.Add($kv[0], "`"$($kv[1])`"")
 }


# Write the KV output to the console so uberAgent can pick it up
Write-Output $($output.Keys.ForEach({
 "$_=$($output.$_)" }
 ) -join ' ')
```

The script's output looks like this:

```
ReadyForAttestation="True" ClearNeededToRecover="False" TPMVersion="2.0" TPMManufacturerVersion="302.12.0.0" IsInitialized="True" ReadyForStorage="True" PPIVersion="1.3" PCR7BindingState="0" TPMManufacturerFullName="Intel" TPMManufacturerID="INTC" IsLockedOut="False" PCClientVersion="1.00" ClearPossible="True" TPMPresent="True" MaintenanceTaskComplete="True" IsCapableForAttestation="True" TPMHasVulnerableFirmware="False"
```

Configuring uberAgent to Run the Script

Deploy the script to any directory on your endpoints. In this example, we're storing it in `C:\Program Files\vast limits\uberAgent\scripts` as `Get-TpmtoolAsKv.ps1`.

Create a new timer in uberAgent's configuration. With the settings shown below, the script is executed five minutes after uberAgent is started and then once every 24 hours.

```
[Timer]
Name                 = TPMStatusInventory
Interval             = 86400000
Start delay          = 300000
Persist interval     = true
Script               = powershell.exe -executionpolicy bypass -file "C:\Program Files\vast limits\uberAgent\Scripts\Get-TpmtoolAsKv.ps1"
ScriptContext        = Session0AsSystem
```

Restart the agent to start collecting data.

Splunk It

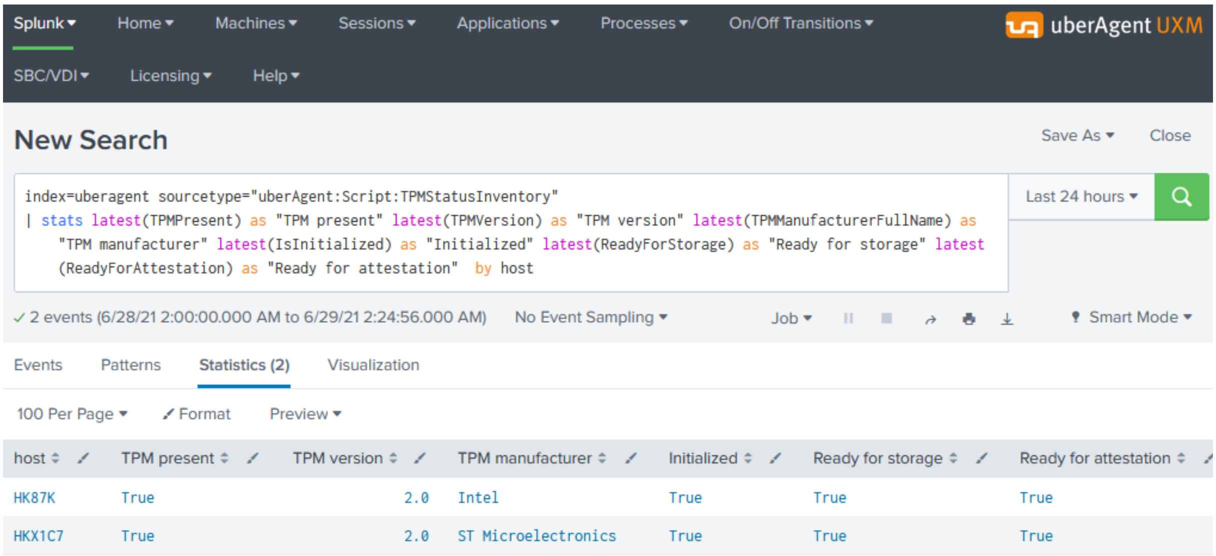
Once the data is in Splunk, you can list the incoming event data as follows

```
index=uberagent sourcetype="uberAgent:Script:TPMStatusInventory"
```

To generate a report, use a Splunk search like the following:

```
index=uberagent sourcetype="uberAgent:Script:TPMStatusInventory"
| stats latest(TPMPresent) as "TPM present" latest(TPMVersion) as "TPM
  version" latest(TPMManufacturerFullName) as "TPM manufacturer"
  latest(IsInitialized) as "Initialized" latest(ReadyForStorage) as "
  Ready for storage" latest(ReadyForAttestation) as "Ready for
  attestation"  by host
```

The above Splunk search creates a table with the latest TPM status per endpoint (host in Splunk terminology):



```
index=uberagent sourcetype="uberAgent:Script:TPMStatusInventory"
| stats latest(TPMPresent) as "TPM present" latest(TPMVersion) as "TPM version" latest(TPMManufacturerFullName) as
    "TPM manufacturer" latest(IsInitialized) as "Initialized" latest(ReadyForStorage) as "Ready for storage" latest
    (ReadyForAttestation) as "Ready for attestation"  by host
```

✓ 2 events (6/28/21 2:00:00.000 AM to 6/29/21 2:24:56.000 AM)

host	TPM present	TPM version	TPM manufacturer	Initialized	Ready for storage	Ready for attestation
HK87K	True	2.0	Intel	True	True	True
HKX1C7	True	2.0	ST Microelectronics	True	True	True

Building a Browser Extension Inventory Report (Chrome/Edge/Firefox)

This article shows how to collect detailed inventory information about all installed browser extensions. The solution presented includes a Splunk dashboard that visualizes the collected data.

The scripts listed in this guide are managed in vast limits’public GitHub repository.

Solution Brief: Browser Extension Inventory

1. Extensions and metadata are stored in **browser profiles**.
2. A **PowerShell script** extracts extension info from browser profiles. The script inspects all browser profiles in the current user's Windows user profile.
3. The script is executed by **uberAgent** for each user logged on to the endpoint. The script's output is captured by uberAgent and forwarded to the (Splunk) backend.
4. A **Splunk dashboard** visualizes the collected data.

Background Information

Listing Chrome Profiles

Getting the Location of the User Data Directory

- Default location on Windows: `%LocalAppData%\Google\Chrome\User Data`
- Documentation

Parsing the JSON file 'Local State'to Find Profile Directories

1. Profile objects are children of the following: `profile` > `info_cache`
2. Profile properties related to the user:
    - `gaia_given_name`
    - `gaia_id`
    - `gaia_name`
    - `name`
    - `shortcut_name`
    - `user_name`
3. Profile names
    - Default profile: `Default`
    - Additional profiles: `Profile`
4. Locating profile directories
    - Profile directories are subdirectories of the user data directory
    - Names of profile directories are idential to profile names

Listing Edge Profiles

Getting the Location of the User Data Directory

- Default location on Windows: `%LocalAppData%\Microsoft\Edge\User Data`
- Everything else is identical to Chrome

Collecting Chrome Extension Inventory Data

'Preferences'vs. 'Secure Preferences' Extensions are listed in one of two possible JSON files located in the profile directory: `Preferences` or `Secure Preferences`. The script checks both.

Note: `Secure Preferences` is typically used instead of the `Preferences` file.

Extension Settings Extension settings are children of the following: `extensions` > `settings` > `<extension ID>`.

Relevant settings for inventory purposes include:

- `from_webstore`: was the extension installed from the Chrome Web Store?
- `install_time`: timestamp of the last update (format: FILETIME / 10 [= microseconds since Jan 1st 1601])
- `path`: extension path, either absolute or relative to the profile subdirectory `Extensions`
- `location`: 1 = user data subdirectory `Extensions`, 5 = Chrome installation directory
- `state`: 1 = enabled
- `was_installed_by_default`: [part of Chrome, not removable?]
- `version`: extension version

Collecting Firefox Extension Inventory Data

Getting the Location of the Profile Directory

- Default location on Windows: `%AppData%\Mozilla\Firefox\Profiles`

Extension Settings The configuration of Firefox extensions is stored in each profile's `extensions.json` file. Within that file, extension settings are children of: `addons`.

Relevant settings for inventory purposes include:

- `sourceURI`: installation source (Firefox Addons or a different site?)
- `updateDate`: timestamp of the last update (format: Unix epoch in ms)
- `location`: `app-profile` = browser profile
- `active`: **`true`** = enabled

- `version`: extension version

Configuring uberAgent to Run the Inventory Script

Note: you can find the latest version of the script in vast limits' public GitHub repository.

1. Store the script file in `C:\Program Files\vast limits\uberAgent\scripts` as `Get-BrowserExtensionInfo.ps1`.
2. Create a new timer in uberAgent's configuration. With the settings shown below, the script is executed in each user session five minutes after uberAgent is started. The script execution is repeated every 24 hours.

```
[Timer]
Name                 = BrowserExtensionInventory
Interval             = 86400000
Start delay          = 300000
Persist interval     = true
Script               = powershell.exe -executionpolicy bypass -file "C:\Program Files\vast limits\uberAgent\Scripts\Get-BrowserExtensionInfo.ps1"
ScriptContext        = UserSessionAsUser
```

Restart the agent to start collecting data.

Script Output

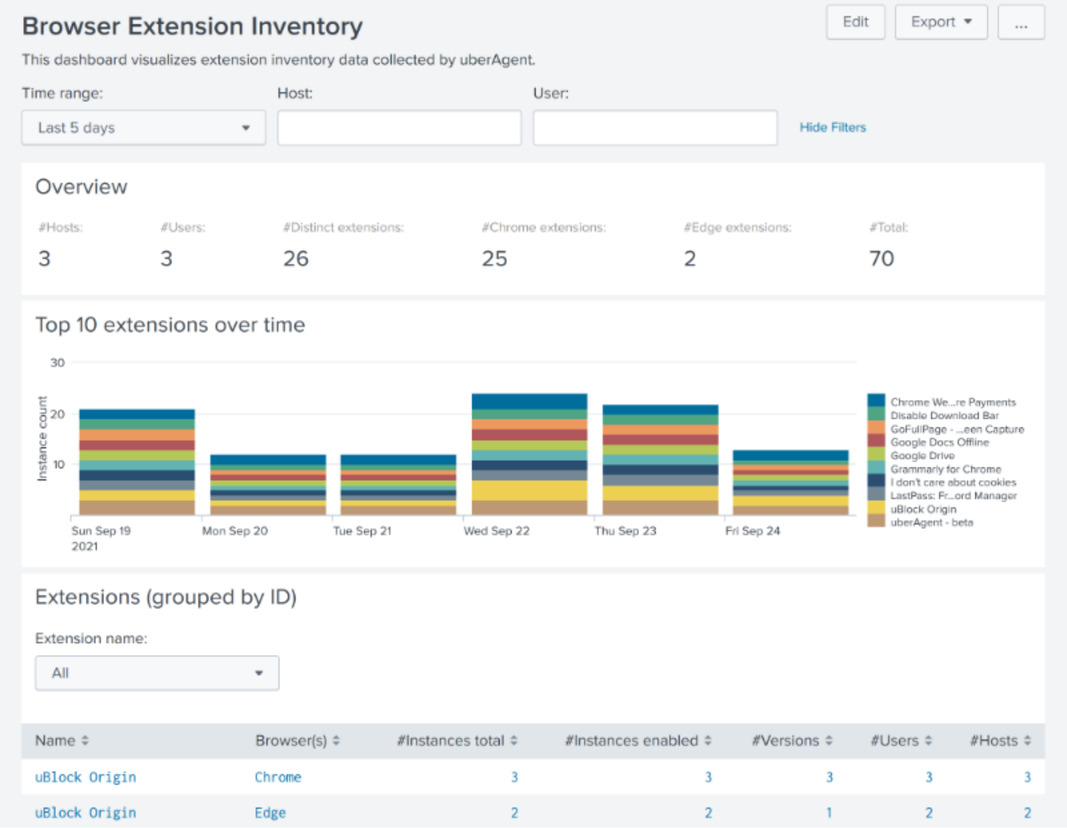
The output from the inventory script looks like this:

```
OsUser="helge" Browser="Chrome" ProfileDir="Default" ProfileName="Person 1" ProfileGaiaName="Helge Klein" ProfileUserName="email@domain.com" ExtensionId="cmcmennehclgdccnlmnjladhlkmclbkb" ExtensionName="uberAgent" ExtensionVersion="3.0.6" ExtensionFromWebstore="True" ExtensionState="1" ExtensionInstallTime="1606912508508" ExtensionInstalledByDefault="False"
```

Splunk Dashboard

Once inventory data is sent from endpoints with uberAgent to Splunk, install the Splunk app that is part of this practice guide. You can find it in this guide's GitHub repository or in Splunkbase.

The dashboard provides a timeline of the top 10 extensions along with a data table listing all browser extensions. Selecting an extension's row in the table brings up additional charts and tables that list all extension instances in full detail.



Browser Extension Inventory

This dashboard visualizes extension inventory data collected by uberAgent.

Time range: Last 5 days

Host:

User:

Overview

#Hosts:	#Users:	#Distinct extensions:	#Chrome extensions:	#Edge extensions:	#Total:
3	3	26	25	2	70

Top 10 extensions over time

Extensions (grouped by ID)

Extension name: All

Name	Browser(s)	#Instances total	#Instances enabled	#Versions	#Users	#Hosts
uBlock Origin	Chrome	3	3	3	3	3
uBlock Origin	Edge	2	2	1	2	2

Internet Explorer: Distinguish Standalone and Edge IE Mode Starts

Internet Explorer has been with us for many years as the ever-present browser. Most people will agree that the relationship is quite ambivalent. Especially in the corporate world, Internet Explorer was and still is deeply rooted. Many companies have applications in use that were not only developed on the basis of Internet Explorer but would not work at all without it.

Background Information

End-of-Life for Internet Explorer Announced

As you may have noticed, Microsoft has announced that the Internet Explorer 11 desktop application will be retired and go out of support on June 15, 2022. The challenge is to identify the dependencies of existing web applications on Internet Explorer and, perhaps more importantly, which of these applications are actually still being used.

Capture Internet Explorer Starts: Standalone & Edge IE Mode

Typically, two scenarios are of interest. Either the user launches Internet Explorer, `iexplore.exe`, directly, or a website is launched within Microsoft Edge in *Internet Explorer mode*. We would like to be able to capture both and distinguish between them.

To do so, the command line of the process contains the necessary information. To get a better understanding, let us compare a direct Internet Explorer startup vs. one coming from Edge IE mode.

Standalone Internet Explorer Start When Internet Explorer is launched as a standalone application, the command line looks as follows:

```
"C:\Program Files (x86)\Internet Explorer\IEXPLORE.EXE"SCODEF:6456 CREDAT:9730 /prefetch:2
```

Internet Explorer Started in Edge IE Mode In Edge IE mode, the command line of the embedded Internet Explorer executable looks as follows:

```
"C:\Program Files (x86)\Internet Explorer\IEXPLORE.EXE"SCODEF:11244 CREDAT:75266 APPID:MSEdge /prefetch:2
```

As you can see, the additional parameter `APPID:MSEdge` can be used to distinguish between standalone IE and Edge IE mode.

Configuration Instructions

Activate uberAgent Feature: EnableExtendedInfo

uberAgent does not send the command line to the backend by default. The reason behind this is that we want to keep the generated data volume as low as possible (more information about managing the data volume can be found here). However, for this scenario, the command line is required. Activate `EnableExtendedInfo` via `uberAgent.conf` as shown below.

```
#   Setting name: EnableExtendedInfo
#   Description: Send detailed information about each started process to the backend, e.g. path, command line, process ID, parent ID. This also enables the population of the ProcGUID field in other sourcetypes, which can be used for detailed process instance tracking.
#   Valid values: true | false
#   Default: false
#   Required: no

[ProcessStartupSettings]
EnableExtendedInfo = true
```

Configure uberAgent Feature: Event Data Filtering

uberAgent provides a feature called Event Data Filtering. It is a powerful filtering mechanism for the collected data that operates right on the endpoint. Please learn more about it in our documentation.

For this use scenario, Event Data Filtering allows us to reduce the additional data volume generated by activating by `EnableExtendedInfo` to an absolute minimum. Specifically, we only need the command line for the process `iexplore.exe` in the sourcetypes uberAgent:Process:ProcessDetail and uberAgent:Process:ProcessStartup.

Since the only extra fields of interest are `ProcID`, `ProcGUID` and `ProcCmdline`, everything else that is not needed can be removed. Please find the example configuration below. Save the content as `uberAgent-eventdata-filter-vastlimits-Internet-Explorer.conf` and place that file in the same directory as `uberagent.conf`.

```
[EventDataFilter]
# Unnecessary for this use case.
Action = clear
Sourcetype = Process:ProcessDetail
Field = ProcGUID
Query = true

[EventDataFilter]
# Unnecessary for any other process than iexplore.exe.
Action = clear
Sourcetype = Process:ProcessStartup
Field = ProcID
Field = ProcParentID
Field = SessionID
Field = ProcGUID
Field = SessionGUID
Field = ProcParentName
Field = ProcPath
Field = ProcCmdline
Field = ProcParentGUID
Query = ProcName != "iexplore.exe"

[EventDataFilter]
# Remove unnecessary fields for iexplore.exe itself.
Action = clear
Sourcetype = Process:ProcessStartup
Field = ProcParentID
Field = SessionID
Field = SessionGUID
Field = ProcParentName
Field = ProcPath
Field = ProcParentGUID
Query = ProcName == "iexplore.exe"
```

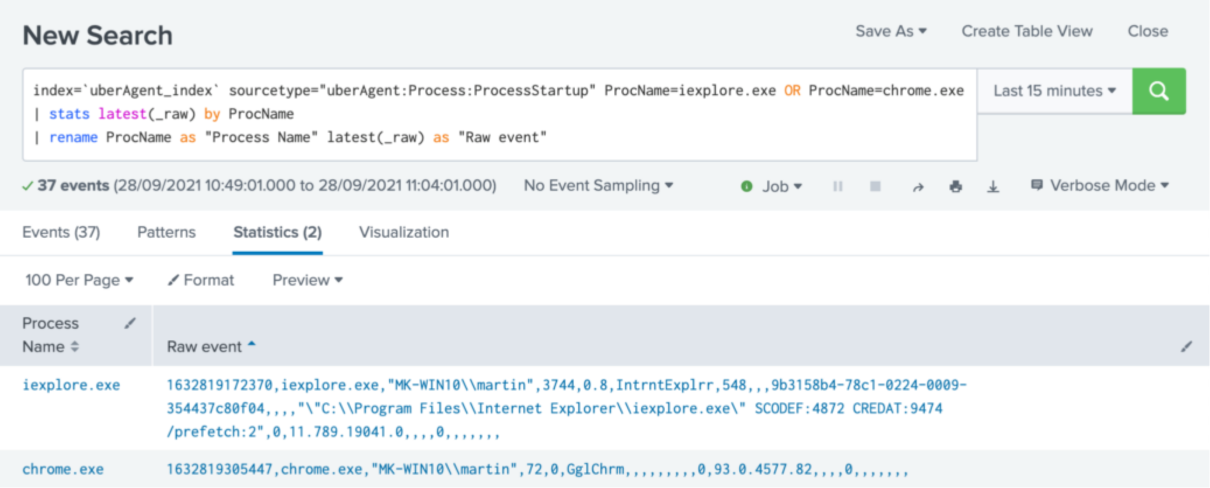
Include the new configuration file by using the `@ConfigInclude` directive below. A good place for the include directive is in `uberAgent-eventdata-filter.conf` which also resides in the same directory as `uberAgent.conf`. Find out more about configuration file locations here.

```
# Additional fields for Internet Explorer analysis (requires EnableExtendedInfo to be enabled)
@ConfigInclude uberAgent-eventdata-filter-vastlimits-Internet-Explorer.conf platform=Windows
```

Finally, restart uberAgent with the new configuration in place and head over to Splunk.

Verify Configuration

To prove our newly configured event data filtering works as expected, run the search below which highlights two raw events in Splunk. One for `iexplore.exe` and another one for `chrome.exe`. You can clearly see the difference in populated fields.



```
index=`uberAgent_index` sourcetype="uberAgent:Process:ProcessStartup" ProcName=iexplore.exe OR ProcName=chrome.exe
| stats latest(_raw) by ProcName
| rename ProcName as "Process Name" latest(_raw) as "Raw event"
```

✓ 37 events (28/09/2021 10:49:01.000 to 28/09/2021 11:04:01.000)

Process Name	Raw event
iexplore.exe	1632819172370,iexplore.exe,"MK-WIN10\\martin",3744,0.8,IntrntExplrr,548,,,9b3158b4-78c1-0224-0009-354437c80f04,,,,"\"C:\\Program Files\\Internet Explorer\\iexplore.exe\" SCODEF:4872 CREDAT:9474 /prefetch:2",0,11.789.19041.0,,,,0,,,,,,,
chrome.exe	1632819305447,chrome.exe,"MK-WIN10\\martin",72,0,GglChrm,,,,,,,,,0,93.0.4577.82,,,,0,,,,,,,

Splunk it

Once the data is available in Splunk, it can be used to search for every direct or indirect Internet Explorer start, for every user on every host. In order to get the information about the actual web applications being used, the uberAgent browser add-on for Internet Explorer is required (documentation). This information is only available for direct Internet Explorer starts though, because of Microsoft Edge/Chromium API limitations.

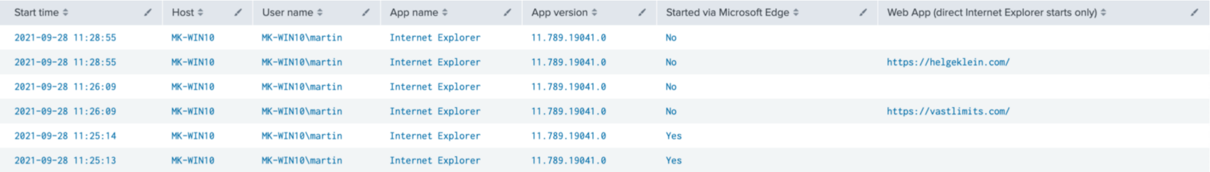
The Search

```
| pivot `uA_DM_Process_ProcessStartup` Process_ProcessStartup
   latest(AppName) as "App name"
   latest(AppVersion) as "App version"
```

```
    latest(_time) as StartTime
    latest(ProcCmdline) as "Process Commandline"
    latest(ProcID) as ProcID
    filter host in ("*")
    filter ProcName is "iexplore.exe"
    splitrow ProcGUID
    splitrow host as Host
    splitrow ProcUser as "User name"
| join type=left Host, ProcID
    [
    | pivot `uA_DM_Application_BrowserPerformanceIE` Application_BrowserPerformanceIE
        values(URL)
        splitrow host as Host
        splitrow ProcID
    | fields + *
    ]
| eval "Start time" = strftime (strptime (StartTime, "%Y-%m-%dT%H:%M:%S.%Q%z"), "%Y-%m-%d %H:%M:%S")
| eval "Started via Microsoft Edge"=if(like('Process Commandline', "%APPID:MSEdge%"), "Yes", "No")
| eval "Web App (direct Internet Explorer starts only)"=nullif('values(URL)',"none")
| sort - "Start time"
| table "Start time", Host, "User name", "App name", "App version", "Started via Microsoft Edge", "Web App (direct Internet Explorer starts only)"
```

The Results

The resulting table shows for each instance of `iexplore.exe` whether it was launched standalone or via Edge IE mode (column ”Started via Microsoft Edge”). For standalone IE processes that are used to rendering a web page, the URL is displayed, too.



Start time	Host	User name	App name	App version	Started via Microsoft Edge	Web App (direct Internet Explorer starts only)
2021-09-28 11:28:55	MK-WIN10	MK-WIN10\martin	Internet Explorer	11.789.19041.0	No	
2021-09-28 11:28:55	MK-WIN10	MK-WIN10\martin	Internet Explorer	11.789.19041.0	No	https://helgeklein.com/
2021-09-28 11:26:09	MK-WIN10	MK-WIN10\martin	Internet Explorer	11.789.19041.0	No	
2021-09-28 11:26:09	MK-WIN10	MK-WIN10\martin	Internet Explorer	11.789.19041.0	No	https://vastlimits.com/
2021-09-28 11:25:14	MK-WIN10	MK-WIN10\martin	Internet Explorer	11.789.19041.0	Yes	
2021-09-28 11:25:13	MK-WIN10	MK-WIN10\martin	Internet Explorer	11.789.19041.0	Yes	

Testing the Azure AD (Entra ID) Join Status

This article shows how to test each endpoint’s Azure AD (aka Entra ID) join status. The output can easily be used in Splunk reports and dashboards.

Machines do not always register smoothly with Azure AD. This is especially true for customers with non-persistent virtual desktops, as these types of machines must re-register with Azure AD each time they start. The script presented in this guide helps with troubleshooting by making the registration state available in Splunk.

Note: The script(s) listed in this guide are managed in vast limits' public GitHub repository.

Collecting AAD Join Status With PowerShell From Dsregcmd

Data Source: Dsregcmd.exe

When run with the parameter `Status`, `Dsregcmd`'s outputs a lot of information. Below is the output cropped to the data we are interested in:

```
+----------------------------------------------------------------------+

| Device State
   |
+----------------------------------------------------------------------+


             AzureAdJoined : YES
          EnterpriseJoined : NO
              DomainJoined : NO
           Virtual Desktop : NOT SET
               Device Name : MACHINE1
...
```

Converting Dsregcmd's Output to a Key-Value String

We'll use the following PowerShell script to execute `Dregcmd` and convert its output to the KV string format required by uberAgent custom scripts:

```
function Test-AzureAdJoined {

   $output = & dsregcmd /status

   # Check if the output contains "AzureAdJoined : YES"
   return $output -match "AzureAdJoined\s+:\s+YES"
 }


$AzureAdJoined = 0

$joinedToAzureAd = Test-AzureAdJoined

if ($joinedToAzureAd) {

   $AzureAdJoined = 1
 }
```

```


# Build the output hash
$Output = @{

   'AzureAdJoined' = $AzureAdJoined
 }


# Finally, write the hash to stdout. The output will be picked up by uberAgent.
Write-Output $($Output.Keys.ForEach({
 "$_=$($Output.$_)" }
 ) -join ' ')
```

The script’s output looks like this:

```
AzureAdJoined=1
```

Configuring uberAgent to Run the Script

Deploy the script to a directory on your endpoints. In this example, we’re storing it in `C:\Program Files\vast limits\uberAgent\scripts` as `Get-AzureAdJoinStatus.ps1`.

Create a new timer in uberAgent’s configuration. With the settings shown below, the script is executed one hour after uberAgent is started to give the registration process enough time, and then once every 24 hours.

```
[Timer]
Name                 = AzureAdJoinStatus
Interval             = 86400000
Start delay          = 3600000
Persist interval     = true
Script               = powershell.exe -executionpolicy bypass -file "C:\Program Files\vast limits\uberAgent\Scripts\Get-AzureAdJoinStatus.ps1"
ScriptContext        = Session0AsSystem
```

Restart the agent to start collecting data.

Splunk It

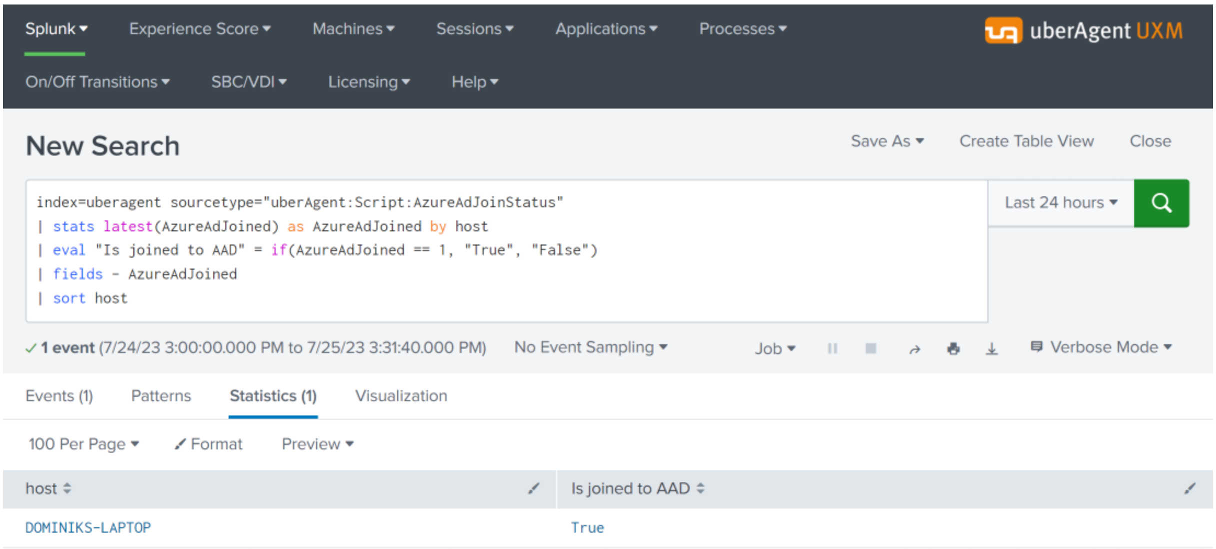
Once the data is in Splunk, you can list the incoming event data as follows:

```
index=uberagent sourcetype="uberAgent:Script:AzureAdJoinStatus"
```

To generate a report, use a Splunk search like the following:

```
index=uberagent sourcetype="uberAgent:Script:AzureAdJoinStatus"
| stats latest(AzureAdJoined) as AzureAdJoined by host
| eval "Is joined to AAD" = if(AzureAdJoined == 1, "True", "False")
| fields - AzureAdJoined
| sort host
```

The above Splunk search creates a table with the latest AAD join status per endpoint (host in Splunk terminology):



```
index=uberagent sourcetype="uberAgent:Script:AzureAdJoinStatus"
| stats latest(AzureAdJoined) as AzureAdJoined by host
| eval "Is joined to AAD" = if(AzureAdJoined == 1, "True", "False")
| fields - AzureAdJoined
| sort host
```

✓ 1 event (7/24/23 3:00:00.000 PM to 7/25/23 3:31:40.000 PM)

host	Is joined to AAD
DOMINIKS-LAPTOP	True

Addons and Tools

uberAgent Helpdesk Splunk App

uberAgent Helpdesk is a Splunk app for IT professionals who support virtual or physical desktops and who need to resolve issues quickly. This app provides a different view of the existing uberAgent dataset that is streamlined for helpdesk use.

uberAgent Event Generator

The event generator is the perfect tool for uberAgent product demonstrations. It simulates a large number of endpoints and generates data for all dashboards.

uberAgent Log Collector Splunk App

uberAgent Log Collector helps you monitor the health of your uberAgent deployment by collecting the data logged by uberAgent and sending it to Splunk for indexing. Dashboards provide easy access.

uberAgent Logfile Syntax Highlighter for Notepad++

This syntax highlighter makes uberAgent log files easier to read in Notepad++.

uberAgent Event Generator for Splunk

The uberAgent Event Generator is a powerful tool designed for product demonstrations of uberAgent. It simulates a large number of endpoints and generates data for all dashboards, allowing you to showcase the full capabilities of uberAgent without the need for a complex setup.

Features

- Simulates multiple endpoints
- Generates data for all uberAgent dashboards
- Perfect for product demonstrations and testing

Quick Links

- Changelog and Release Notes
- Quickstart Guide

Getting Started

To get started with the uberAgent Event Generator:

1. Download the Event Generator from the uberAgent website
2. Install the Event Generator on your Splunk instance
3. Access your uberAgent Splunk dashboards to see the simulated data

For more detailed instructions, please refer to our Quickstart Guide.

Related Tools

- uberAgent Helpdesk Splunk App
- uberAgent Log Collector Splunk App

For a complete list of uberAgent addons and tools, visit our Addons and Tools page.

Changelog and Release Notes

Version 7.1

Release notes

- Increased .NET version to 7.
- Changed configuration file format to JSON.

Improvements

- Log messages are written to stdout and a log file located in the `%TEMP%` (Windows) or `/tmp` (Linux) directory.
- The fields `SessionFgBrowserType`, `SessionFgBrowserType` and `SessionFgBrowserActiveTa` of sourcetype `uberAgent:Session:SessionDetail` are now available on macOS, too.

New Sourcetypes

- New sourcetype `uberAgentESA:System:SecurityInventory` with fields: `SecurityInventoryC` , `SecurityInventoryName`, `SecurityInventoryScore`, `SecurityInventoryRiskScore` , `SecurityInventoryResultData`, `SecurityInventoryErrorCode`, `SecurityInventoryEr` , `SecurityInventoryScope`, `SecurityInventoryScopeEntity`.

Updated Sourcetypes

- **Sourcetype [B287]:** `uberAgent:Process:ProcessStatistics` has new field(s): `ProcInputDelayMaxMs`, `ProcInputDelaySumMs` and `ProcInputDelayCount`.
- **Sourcetype [B287]:** `uberAgent:Session:SessionDetail` has new field(s): `SessionInputDelay` , `SessionInputDelaySumMs` and `SessionInputDelayCount`.
- **Sourcetype [B751]:** `uberAgent:OnOffTransition:BootDetail2` has new field(s): `UserLogonWaitDurationMs`.
- **Sourcetype [B766]:** `uberAgentESA:Process:DnsQuery` has new field(s): `DnsRisk52Chars` , `DnsRisk27UniqueChars`, `DnsRiskEmptyResponse`, `DnsRiskTXTRecord`, `DnsRiskHighEntropy`, `DnsResponseStatus`.

Version 7.0

Release notes

- macOS client machines are now supported.

Improvements

- Machines get their IP addresses once and they do not change.
- Machines get their disk volumes only once and they do not change during their lifetime.
- A random in-session process needs a long time to start.
- Application UI delay events can now only be generated by processes started during a session.
- WiFi data is now sent only for WiFi adapters.
- BSODs are now sent less frequently.

New Sourcetypes

- **Sourcetype:** new sourcetype `uberAgent:CitrixSession:VirtualChannelDetail` with fields: `SessionGUID`, `SessionUser`, `VirtualChannelVendorName`, `VirtualChannelData` , `VirtualChannelDataVolumeOutputMB`.
- **Sourcetype:** new sourcetype `uberAgent:CitrixSession:SessionConfig` with fields: `SessionGUID`, `SessionUser`, `AudioActualPriority`, `AudioPolicyAllowMicrophoneRedir` , `AudioPolicyAllowRedirection`, `AudioPolicyPriority`, `AudioPolicySoundQuality` , `CdmActualPriority`, `CdmVolumes`, `CdmPolicyAllowDriveRedirection`, `CdmPolicyPriority`, `CdmPolicyReadOnly`, `DisplayMode`, `ThinwireActualPriority` , `ThinwireColorDepth`, `ThinwireComponentEncoder`, `ThinwireHardwareEncodeInUse` , `ThinwireVideoCodecType`, `ThinwireColorspace`, `ThinwireVideoCodecUse`, `ThinwirePolicyFps`, `ThinwirePolicyPriority`, `ThinwirePolicyUseHardwareEncoding` , `ThinwirePolicyUseVideoCodec`, `ThinwirePolicyVisualQuality`, `FramehawkActualPri` , `FramehawkPolicyPriority`, `D3DActualPriority`, `D3DPolicyAeroRedirection` , `D3DPolicyGraphicsQuality`, `D3DPolicyPriority`, `GraphicsActualPriority` , `GraphicsPolicyDisplayDegradeNotifyUser`, `GraphicsPolicyDisplayDegradePolicy` , `GraphicsPolicyPriority`, `NetworkConnectedVia`, `NetworkEdtMtu`, `NetworkPolicyAcce` , `NetworkPolicyICAListenerPortNumber`, `NetworkPolicySessionReliabilityPort` , `NetworkPolicySessionReliabilityTimeout`, `PrinterActualPriority`, `PrinterSessionPrinter`, `PrinterPolicyAllowRedirection`, `PrinterPolicyAutoCreate` , `PrinterPolicyPriority`, `USBActualPriority`, `USBPolicyAllowPNPRedirection` , `USBPolicyAllowUSBSupport`, `USBPolicyPriority`.
- **Sourcetype:** new sourcetype `uberAgent:Process:ProcessStatistics` with fields: `ProcHandleCount`, `ProcThreadCount`, `ProcPriority`, `ProcPrivateMB`, `ProcVirtualSizeMB`, `ProcPageFaultsPS`, `ProcPageFileMB`, `ProcName`, `ProcID`, `ProcGUID`, `ProcUser` and `AppId`.
- **Sourcetype:** new sourcetype `uberAgent:System:PerformanceCounter` with fields: `PerformanceCounterObject`, `PerformanceCounterInstance`, `PerformanceCounterName` , `PerformanceCounterValue`.

Updated Sourcetypes

- **Sourcetype:** `uberAgent:Application:NetworkConnectFailure` has new field(s): `NetTargetSourcePort`.
- **Sourcetype:** `uberAgent:System:MachineInventory` has new field(s): `HwHypervisorVendor`.
- **Sourcetype:** `uberAgent:Session:SessionDetail` has new field(s): `SessionRoundTripTimeMs`, `SessionFps`, `SessionTransportProtocols`.
- **Sourcetype:** `uberAgent:Citrix:Applications` has new field(s): `CustomerId`.
- **Sourcetype:** `uberAgent:Citrix:Catalogs` has new field(s): `CustomerId`.
- **Sourcetype:** `uberAgent:Citrix:DesktopGroups` has new field(s): `CustomerId`.
- **Sourcetype:** `uberAgent:Citrix:Machines` has new field(s): `CustomerId`.
- **Sourcetype:** `uberAgent:Citrix:PublishedDesktops` has new field(s): `CustomerId`.
- **Sourcetype:** replaced KV sourcetype `uberAgent:PerformanceCounter:<TimerName>` with CSV sourcetype `uberAgent:System:PerformanceCounter`.
- **Sourcetype:** `uberAgent:Application:ApplicationUsage` has been removed (it was marked as deprecated as of version 6.1.1).

Version 6.2.0

Updated Sourcetypes

- **Sourcetype:** `uberAgent:Application:NetworkConnectFailure` has new field(s): `NetTargetSourcePort`.
- **Sourcetype:** `uberAgent:Process:NetworkTargetPerformance` has new field(s): `NetTargetSourcePort`.

Version 6.1.1.4416

Improvements

- Improved number of sent events during session lifetime.
- Improved values in *Application Performance* dashboard.

Bugfixes

- Added chmod call to “StartEventgen.cmd”.

Version 6.1.1

Improvements

- Added some more hardware models.
- Added `ModuleName` and `ExceptionCode` to the sourcetype `uberAgent:Application:Errors` (applications crashes).
- Generated outliers for Process DNS.

Updated Sourcetypes

- **Sourcetype:** `uberAgent:Process:ProcessStartup` has new field(s): `HashMD5`, `HashSHA1`, `HashSHA256`, `HashIMP`, `SignatureStatus`, `IsSignedByOSVendor`, `SignerName`.
- **Sourcetype:** `uberAgent:Process:ProcessStartup`: fields `ProcHash` and `HashType` have been removed.
- **Sourcetype:** `uberAgent:Process:ProcessStop` has new field(s): `HashMD5`, `HashSHA1`, `HashSHA256`, `HashIMP`.
- **Sourcetype:** `uberAgent:Process:ProcessStop`: fields `ProcHash` and `HashType` have been removed.
- **Sourcetype:** `uberAgent:Process:ProcessDetail` has new field(s): `SessionID`.
- **Sourcetype:** `uberAgent:CitrixADC:AppliancePerformance` has new field(s): `CpuFan0Speed`, `CpuFan1Speed`, `SystemFanSpeed`, `Cpu0Temp`, `Cpu2Temp`, `InternalTemp`, `PowerSupply1Status`, `PowerSupply2Status`, `PowerSupply3Status`, `PowerSupply4Statu` , `VoltageV33Main`, `ICAOnlySessions`, `ICAOnlyConnections`, `SmartAccessSessions`, `SmartAccessICAConnections`, `SSLSessions`.
- **Sourcetype:** `uberAgent:CitrixADC:Gateway` has new field(s): `HSTS`, `HSTSMaxAge`, `HSTSInclSubdom`, `TLS13`.
- **Sourcetype:** `uberAgent:CitrixADC:vServer` has new field(s): `HSTS`, `HSTSMaxAge`, `HSTSInclSubdom`, `TLS13`.
- **Sourcetype:** `uberAgent:System:NetworkConfigInformation` has new field(s): `NetworkConfigWiFiSignalQuality`, `NetworkConfigWiFiType`, `NetworkConfigWiFiAuthe` .

New Sourcetypes

- **Sourcetype**: new sourcetype `uberAgentESA:Process:DnsQuery` with fields: `ProcName`, `ProcGUID`, `DnsRequest`, `DnsResponse`, `DnsResponseType` and `DnsEventCount`.

Bugfixes

- SMB paths had only one backslash.

Demoing uberAgent With the Event Generator for Splunk

Demonstrating uberAgent can be a bit difficult if you do not have a few dozen machines with live users available. To simplify demos, we offer an event generator that simulates an active environment with various hosts and users.

Architecture

Starting with uberAgent version 6, the Splunk event generator dependency was removed and uberAgent event generator is a single Splunk app. When Splunk is started, a .NET program generates sample data. By default sample data for two hours is generated. If you want to generate additional sample data, you can either restart the Splunk service after 2 hours, or modify the `uAEventGen.conf.json` file (see section "Advanced configuration").

The Splunk app can be used on Windows, Linux, and on macOS-based Splunk installations. Single server setups and distributed deployments are fully supported. The standard installation sends the data to a local Splunk instance using the TCP port 19500.

Installation

.NET 7

As of uberAgent version 7.1, .NET 7.0 is a prerequisite that must be installed on the same server where Splunk is installed. In the case of a distributed environment, .NET 7 must be installed on the same Splunk indexers where you want to install the uberAgent event generator Splunk app.

You can download .NET 7.0 here.

uberAgent Event Generator

Install the uberAgent event generator on one of the indexers. If you have a single Splunk server, install the event generator on that server.

- Download the uberAgent event generator (find out what's new in the changelog)
- On the Splunk server navigate to **Manage apps**
- Click **Install app from file**

- Select the archive you downloaded earlier and click **Upload**
- Restart Splunk

That's it. The event generator starts generating events right after Splunk has been restarted. It will continue to do so for approx. 2 hours and then stop on its own. Just what you need for a demo. To re-enable restart Splunk again.

Configuration

Enabling or Disabling the Event Generator

To enable or disable the uberAgent event generator:

1. On the Splunk server where the uberAgent event generator app is installed navigate to **Manage apps**
2. Locate the *uberAgent event generator* app and click on **enable** or **disable**
3. Restart Splunk

Advanced Configuration

The default configuration should work for a single instance Splunk environment. If you have a distributed Splunk environment or you want to generate different generated sample data, you can modify the configuration file `uAEventGen.conf.json` which is located `%Splunkhome%/etc/apps/uberAgent_eventgenerator/bin/uAEventGenBinaries/your platform`. On a Linux system, for example, this would be: /opt/splunk/etc/apps/uberAgent_eventgenerator/bin/uAEventGenBinaries/Linux
The file contains full documentation of all possible configuration options.

Running Event Generator on macOS ARM

The Eventgen binaries are currently not signed with any certificate. MacOS on an ARM CPU requires a valid certificate otherwise the executable is terminated/killed directly after process startup.
In order to start the event generator on a macOS run the following command:

```
codesign -s "-"/opt/splunk/etc/apps/uberAgent_eventgenerator/bin/
uAEventGenBinaries/macOS/uAEventGen
```

The command adds an ad-hoc certificate to the executed binary.

uberAgent Helpdesk Splunk App

The uberAgent Helpdesk Splunk App is a specialized tool designed for IT professionals who support virtual or physical desktops and need to resolve issues quickly. This app provides a streamlined view of the existing uberAgent dataset, optimized for helpdesk use.

Features

- Simplified view of uberAgent data tailored for helpdesk professionals
- Quick access to key metrics and information for troubleshooting
- Intuitive interface for faster issue resolution
- Customized dashboards for common helpdesk scenarios

Quick Links

- Changelog and Release Notes
- Quickstart Guide

Getting Started

To get started with the uberAgent Helpdesk Splunk App:

1. Ensure you have uberAgent and Splunk installed and configured
2. Download the Helpdesk Splunk App from the Citrix downloads page
3. Install the app on your Splunk instance
4. Start using the specialized dashboards for efficient issue resolution

For more detailed instructions, please refer to our Quickstart Guide.

Related Tools

- uberAgent Event Generator for Splunk
- uberAgent Log Collector Splunk App

For a complete list of uberAgent addons and tools, visit our Addons and Tools page.

Changelog and Release Notes

Version 1.4.0

Improvements

- **Dashboards:** added user input delay metrics that were introduced in uberAgent 7.1.

Release notes

- **Dashboards:** replaced the chart *Disk latency (ms)* with *Input delay (ms)*.

Version 1.3.0

Improvements

- **Dashboards:** added Citrix metrics that were introduced in uberAgent 7.0.
- **Dashboards:** the WiFi signal quality chart is inserted into the *Network details* section dynamically. In older versions, it was always there, even for non-WiFi connections.
- **Dashboards:** all charts are now column charts for better visibility.

Bug fixes

- **Dashboards:** fixed wrong timeframe evaluation in the *Machine details* chart. It now loads much faster.

Release notes

- **Dashboards:** removed the chart *Session activity* in favor of session connection change annotations.

Version 1.2.3

Improvements

- **Splunk:** added `sc_admin` permissions to meet Splunk Cloud requirements.

Version 1.2.2

Improvements

- **Dashboards:** upgraded dashboards to version 1.1. This is a requirement for Splunk Cloud.requirements.

Version 1.2.1

Bug fixes

- **Dashboards:** rounded Wifi signal quality
- **Dashboards:** fixed timeframe issues for the machines and network details charts

Version 1.2.0

Improvements

- **Dashboards:** added WiFi metrics
- **Dashboards:** added network latency metric selector

Version 1.1

- Support for uberAgent 6.0.

Version 1.0

- Initial release

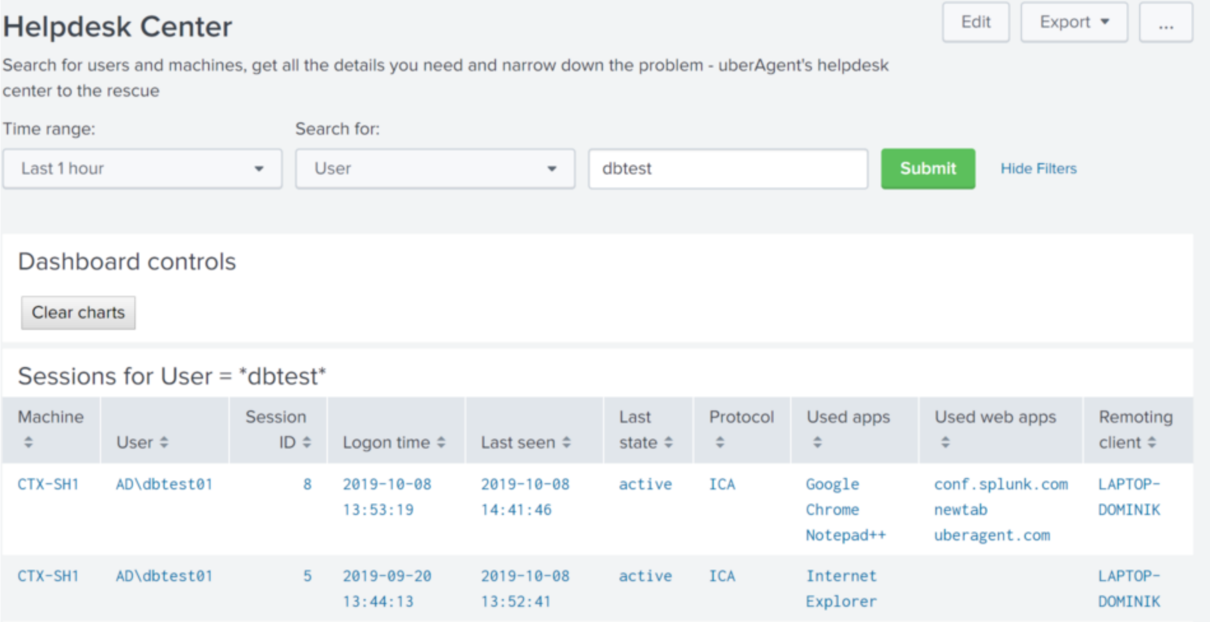
uberAgent Helpdesk Splunk App

The uberAgent Helpdesk Splunk App is designed for helpdesk heroes who support virtual or physical desktops and who need quick answers to typical questions like the following:

- Why is my login so slow? It was fast yesterday.
- Why is my app constantly crashing?
- Citrix is slow!
- The website is not loading fast enough!

What Does the App Look Like

The app has just one dashboard, nice and easy. You start by searching either for a user or a machine. After selecting a session of interest detailed information will be provided.



Helpdesk Center

Search for users and machines, get all the details you need and narrow down the problem - uberAgent's helpdesk center to the rescue

Time range: Last 1 hour

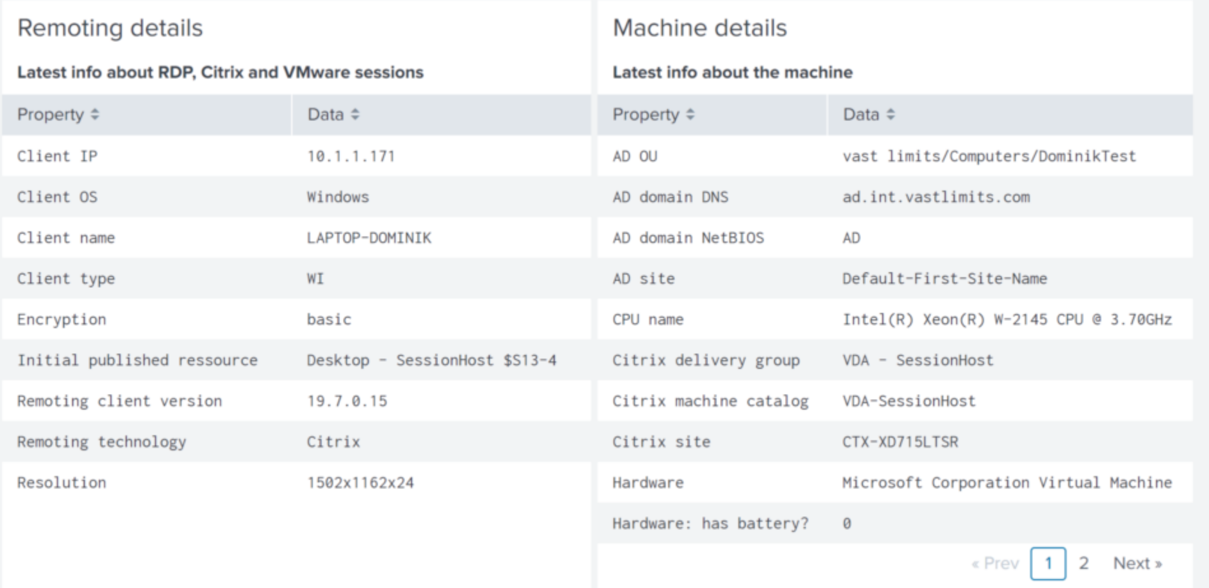
Search for: User — dbtest

Dashboard controls

Sessions for User = *dbtest*

Machine	User	Session ID	Logon time	Last seen	Last state	Protocol	Used apps	Used web apps	Remoting client
CTX-SH1	AD\dbtest01	8	2019-10-08 13:53:19	2019-10-08 14:41:46	active	ICA	Google Chrome Notepad++	conf.splunk.com newtab uberagent.com	LAPTOP-DOMINIK
CTX-SH1	AD\dbtest01	5	2019-09-20 13:44:13	2019-10-08 13:52:41	active	ICA	Internet Explorer		LAPTOP-DOMINIK

The dashboard gives you an overview of the user's session with helpful content like user and machine information, connection state over time as well as logon time compared to the previous week.



Remoting details

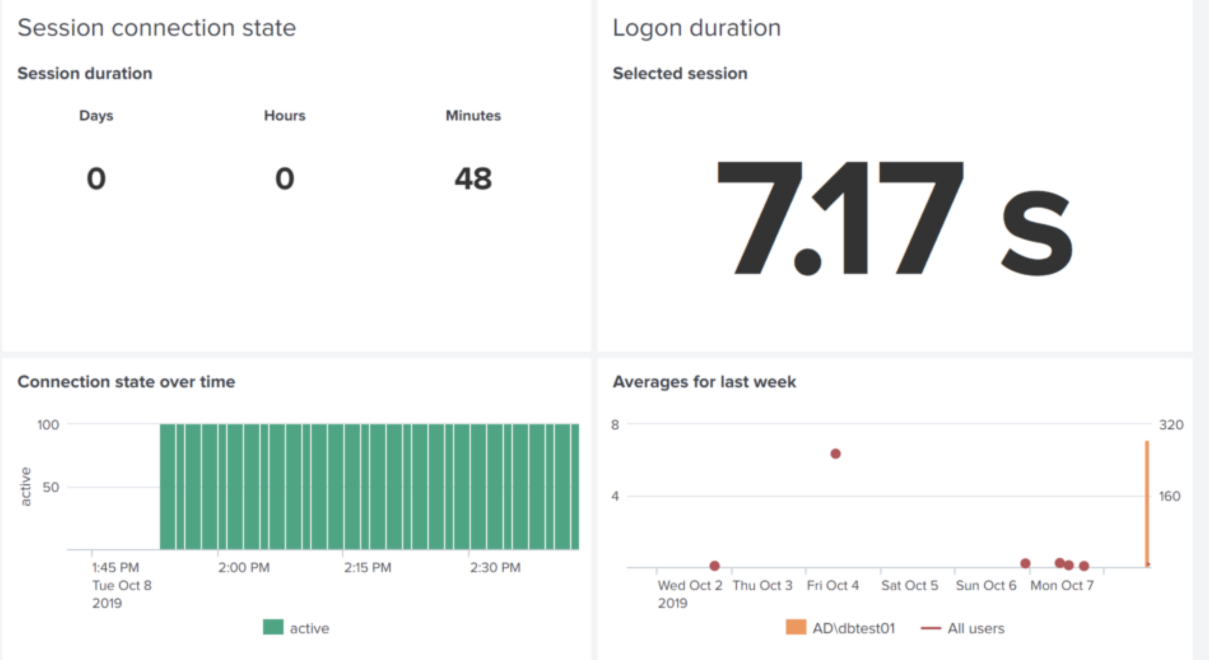
Latest info about RDP, Citrix and VMware sessions

Property	Data
Client IP	10.1.1.171
Client OS	Windows
Client name	LAPTOP-DOMINIK
Client type	WI
Encryption	basic
Initial published ressource	Desktop - SessionHost $S13-4
Remoting client version	19.7.0.15
Remoting technology	Citrix
Resolution	1502x1162x24

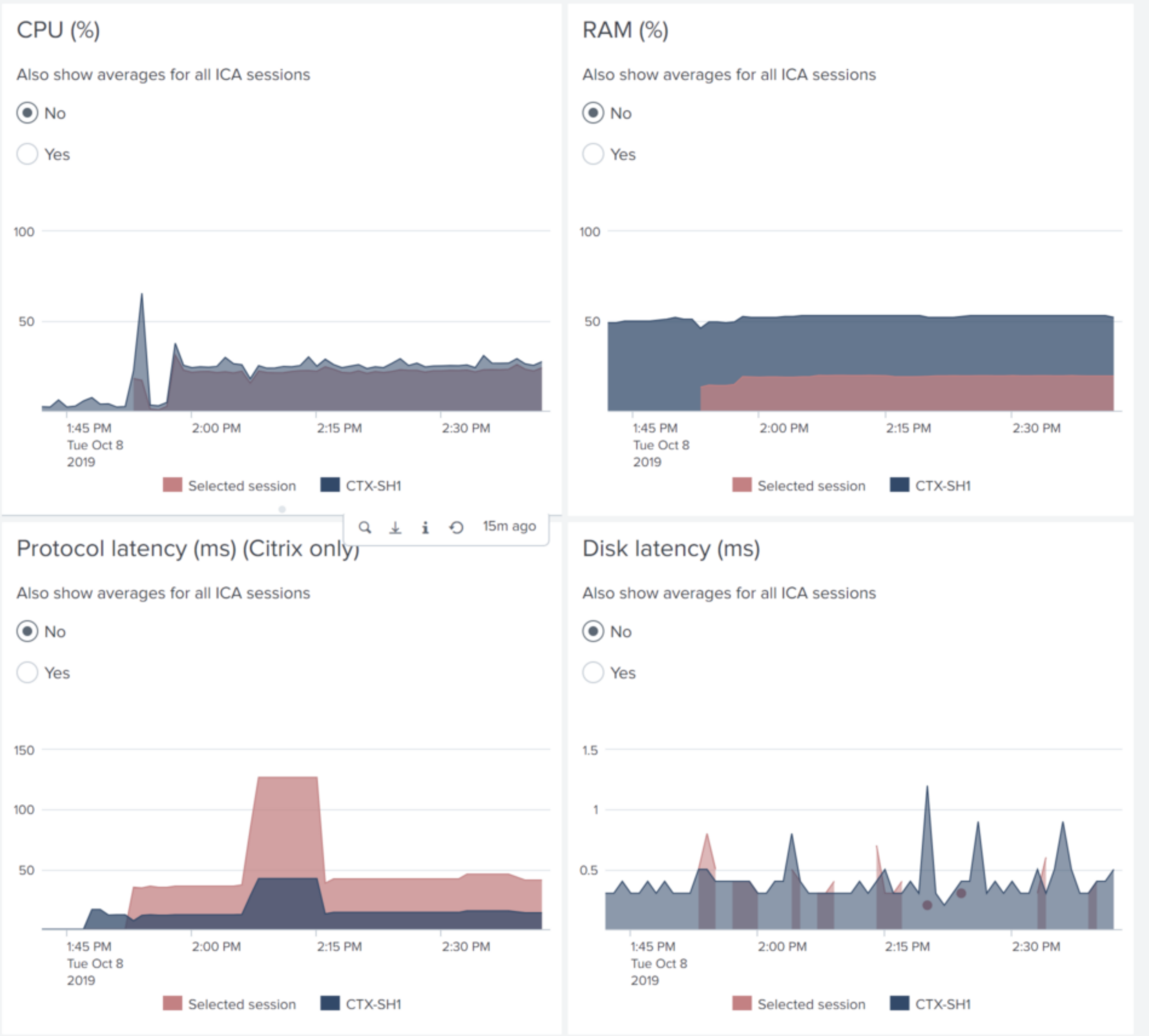
Machine details

Latest info about the machine

Property	Data
AD OU	vast limits/Computers/DominikTest
AD domain DNS	ad.int.vastlimits.com
AD domain NetBIOS	AD
AD site	Default-First-Site-Name
CPU name	Intel(R) Xeon(R) W-2145 CPU @ 3.70GHz
Citrix delivery group	VDA - SessionHost
Citrix machine catalog	VDA-SessionHost
Citrix site	CTX-XD715LTSR
Hardware	Microsoft Corporation Virtual Machine
Hardware: has battery?	0



Now dive into the session's performance and compare it to the entire machine or even to other sessions in the organization.



CPU (%)

Also show averages for all ICA sessions

● No

○ Yes

RAM (%)

Also show averages for all ICA sessions

● No

○ Yes

Protocol latency (ms) (Citrix only)

Also show averages for all ICA sessions

● No

○ Yes

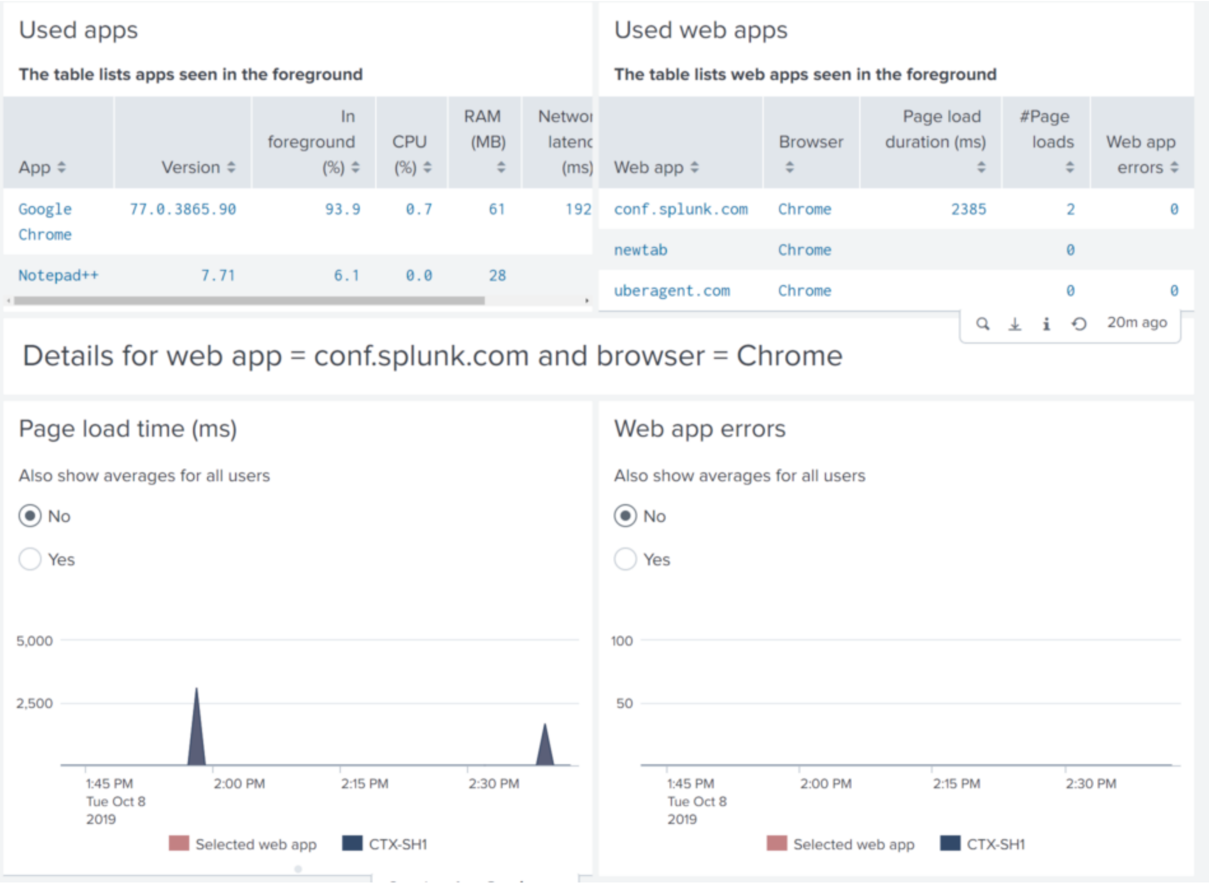
Disk latency (ms)

Also show averages for all ICA sessions

● No

○ Yes

The user is reporting issues with a specific native application or SaaS app? Click on an item in the list and you will get details to solve the user's problem immediately!



Used apps

The table lists apps seen in the foreground

App	Version	In foreground (%)	CPU (%)	RAM (MB)	Networ latenc (ms)
Google Chrome	77.0.3865.90	93.9	0.7	61	192
Notepad++	7.71	6.1	0.0	28	

Used web apps

The table lists web apps seen in the foreground

Web app	Browser	Page load duration (ms)	#Page loads	Web app errors
conf.splunk.com	Chrome	2385	2	0
newtab	Chrome		0	
uberagent.com	Chrome		0	0

Details for web app = conf.splunk.com and browser = Chrome

Page load time (ms)

Also show averages for all users

◉ No

○ Yes

Web app errors

Also show averages for all users

◉ No

○ Yes

Download

The app is available **for free** and can be downloaded from Splunkbase.

Requirements

The helpdesk app provides a different view of the regular dataset collected by uberAgent. As such it requires a working uberAgent infrastructure on Splunk. General uberAgent requirements apply.

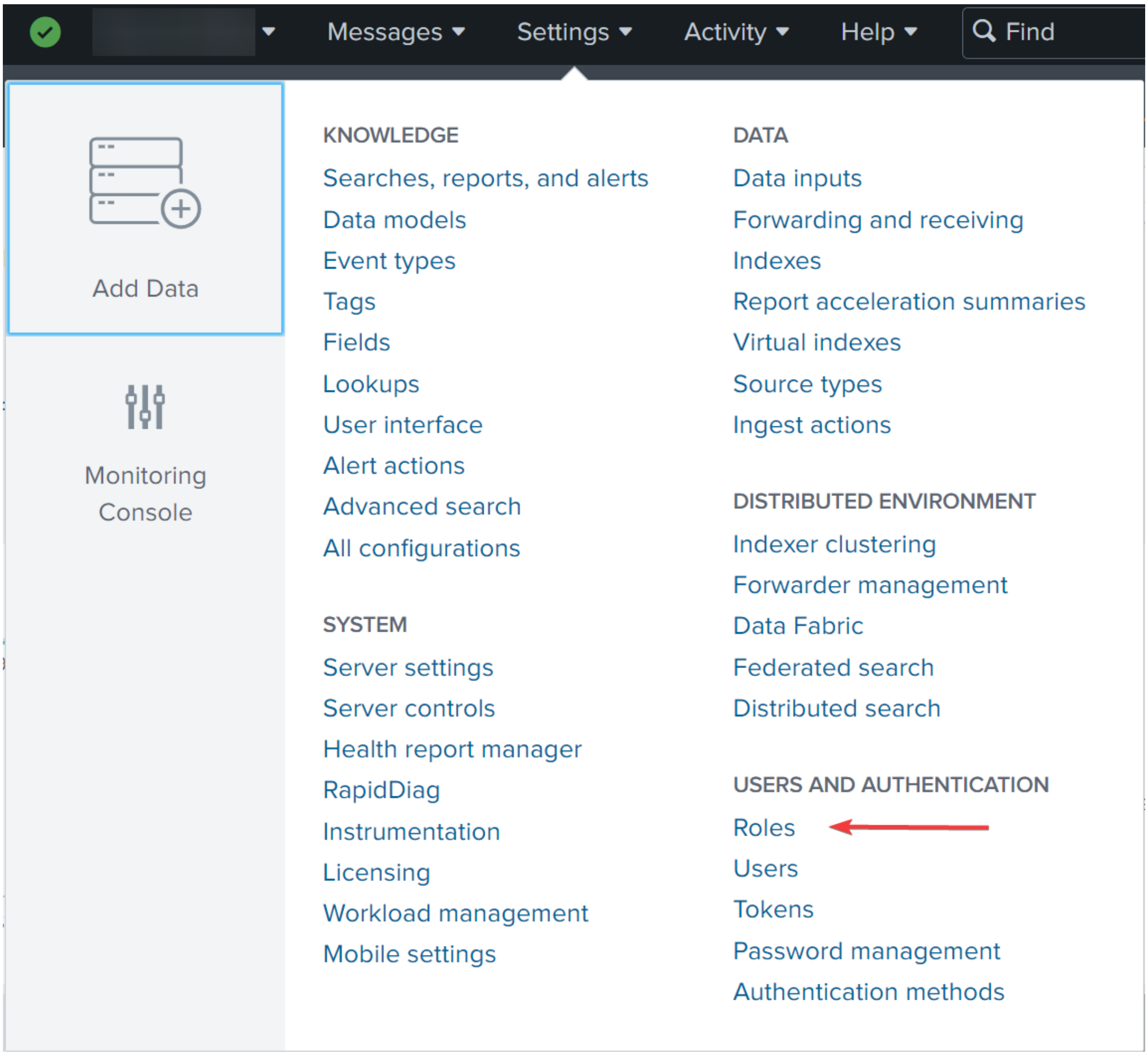
uberAgent Version

- uberAgent UXM 6.0 or greater

Limit Helpdesk Employees to the Helpdesk App

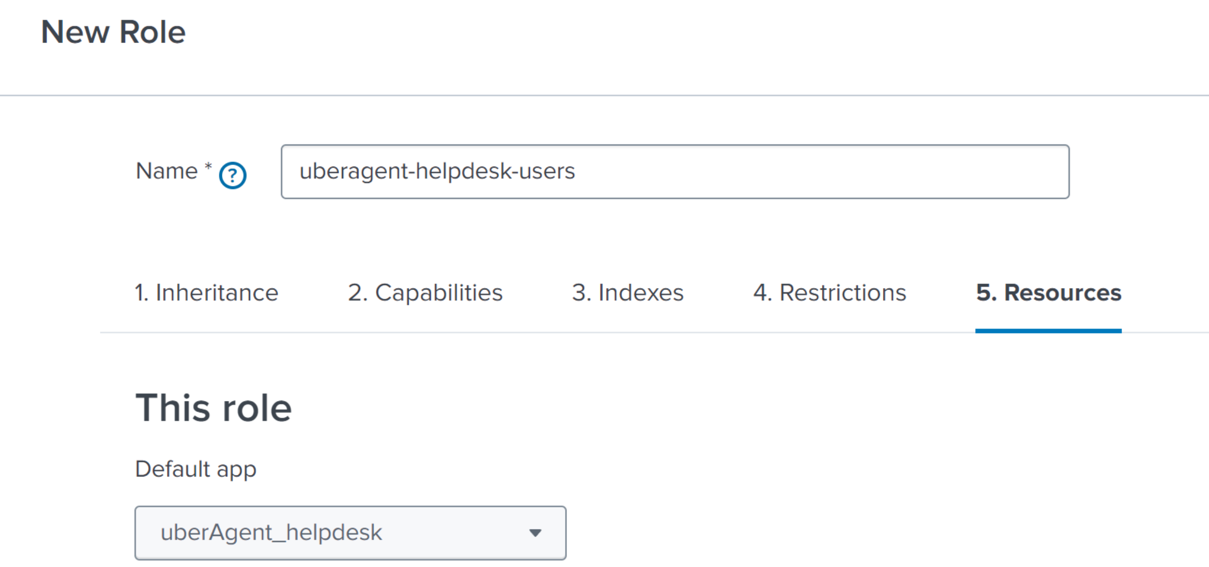
With role based access in Splunk, you can configure that helpdesk employees only see the helpdesk app and are automatically forwarded to the app after logging on to Splunk.

Navigate to **Settings -> Users and Authentication -> Roles**.



Create a new **user** role with a meaningful name like **uberagent-helpdesk-users**. More information on roles are available in Splunk's documentation.

In the last step, select the **uberAgent_helpdesk** app as default.



New Role

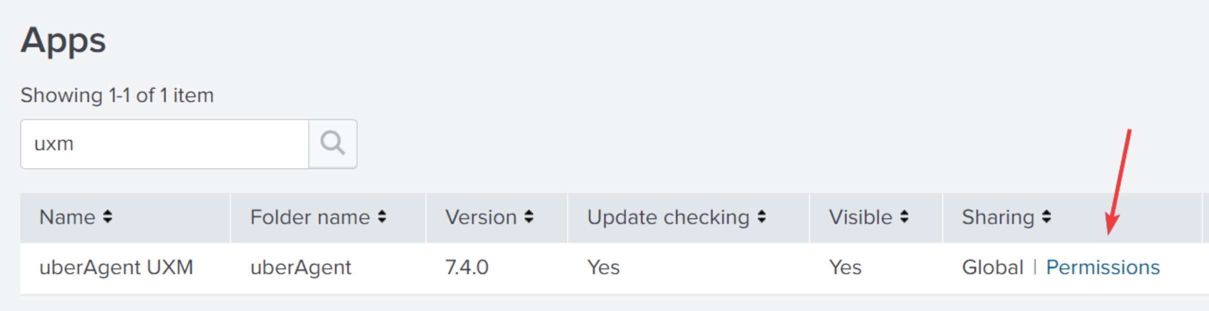
Name * uberagent-helpdesk-users

This role

Default app: uberAgent_helpdesk

Put your helpdesk users in the new **uberagent-helpdesk-users** role.

Go to **Apps -> Manage Apps**, find the uberAgent UXM app, click on **Permissions**, and remove **Read** access for **Everyone**. Give **Read** access to admins only.



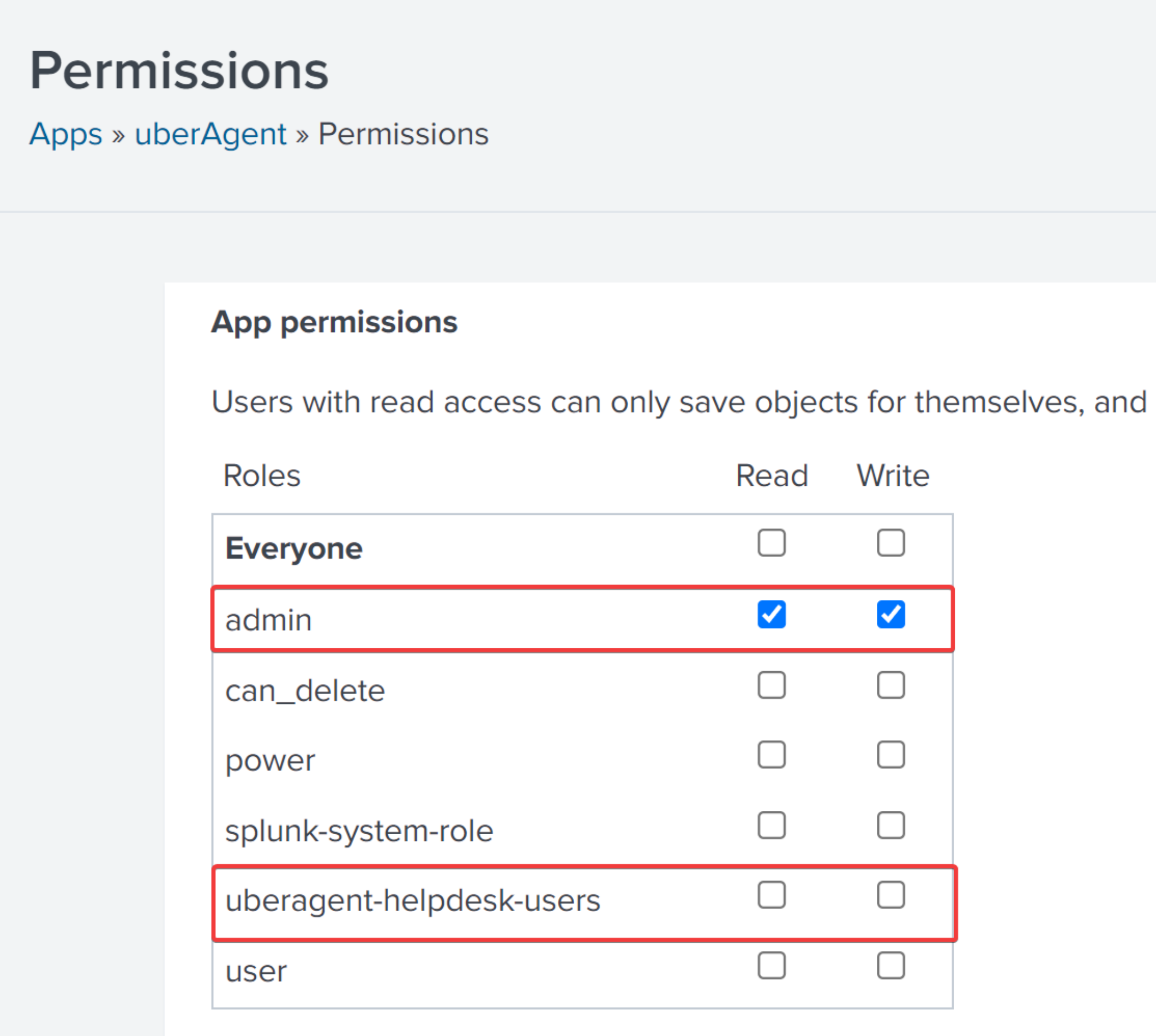
Apps

Showing 1-1 of 1 item

uxm

Name	Folder name	Version	Update checking	Visible	Sharing
uberAgent UXM	uberAgent	7.4.0	Yes	Yes	Global | Permissions

Note that the group **uberagent-helpdesk-users** does not have read access anymore.



Permissions

Apps » uberAgent » Permissions

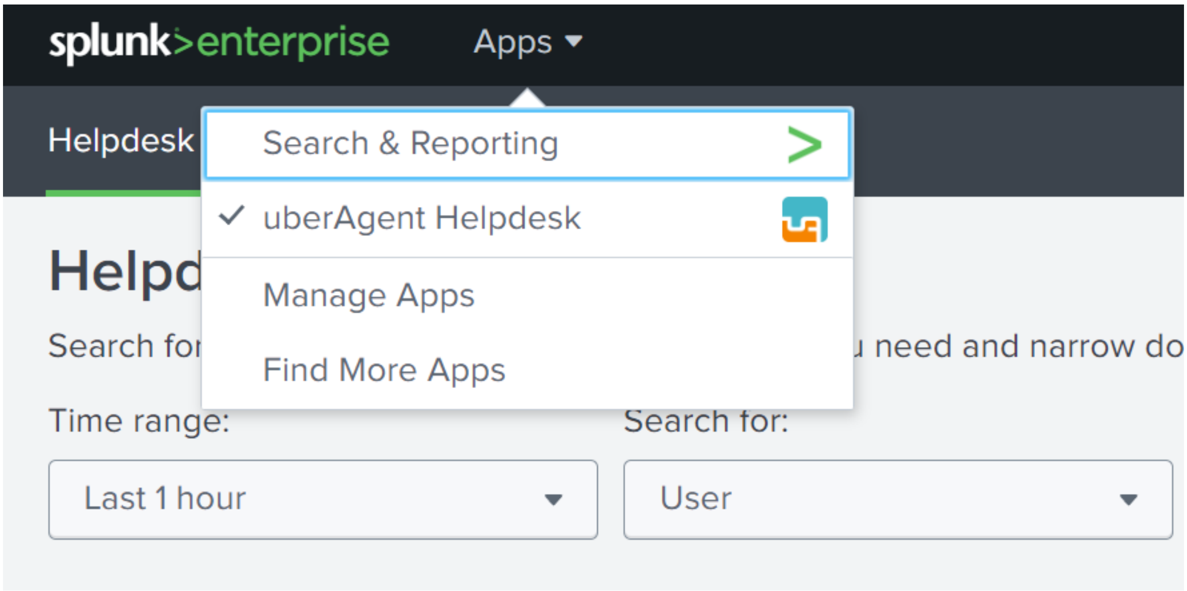
App permissions

Users with read access can only save objects for themselves, and

Roles	Read	Write
Everyone	□	□
admin	☑	☑
can_delete	□	□
power	□	□
splunk-system-role	□	□
uberagent-helpdesk-users	□	□
user	□	□

Repeat the process for the uberAgent ESA app.

Now helpdesk employees only see the helpdesk app when logging on to Splunk.



This chapter was contributed by Antoin Carroll.

uberAgent Log Collector Splunk App

The uberAgent Log Collector Splunk App is a powerful tool designed to help you monitor the health of your uberAgent deployment. It collects the data logged by uberAgent and sends it to Splunk, providing easy access through pre-build dashboards.

Features

- Automated collection of uberAgent log data
- Centralized logging and monitoring of uberAgent deployments
- Custom dashboards for quick health checks and troubleshooting
- Easy integration with existing Splunk environments

Quick Links

- Changelog and Release Notes
- Quickstart Guide

Getting Started

To get started with the uberAgent Log Collector App, please refer to our Quickstart Guide.

Related Tools

- uberAgent Event Generator for Splunk
- uberAgent Helpdesk Splunk App

For a complete list of uberAgent addons and tools, visit our Addons and Tools page.

Changelog and Release Notes

Version 1.2.3

- Compatibility with the latest Splunk app packaging rules
- No new features were added in this release

Version 1.2.2

- Upgraded dashboards to version 1.1
- Added sc_admin permissions to meet Splunk Cloud requirements.
- Disabled index creation to meet Splunk Cloud requirements.
- Disabled data model acceleration to meet Splunk Cloud requirements.

Version 1.2.1

- Disabled the truncation of lines greater than 10,000 bytes.

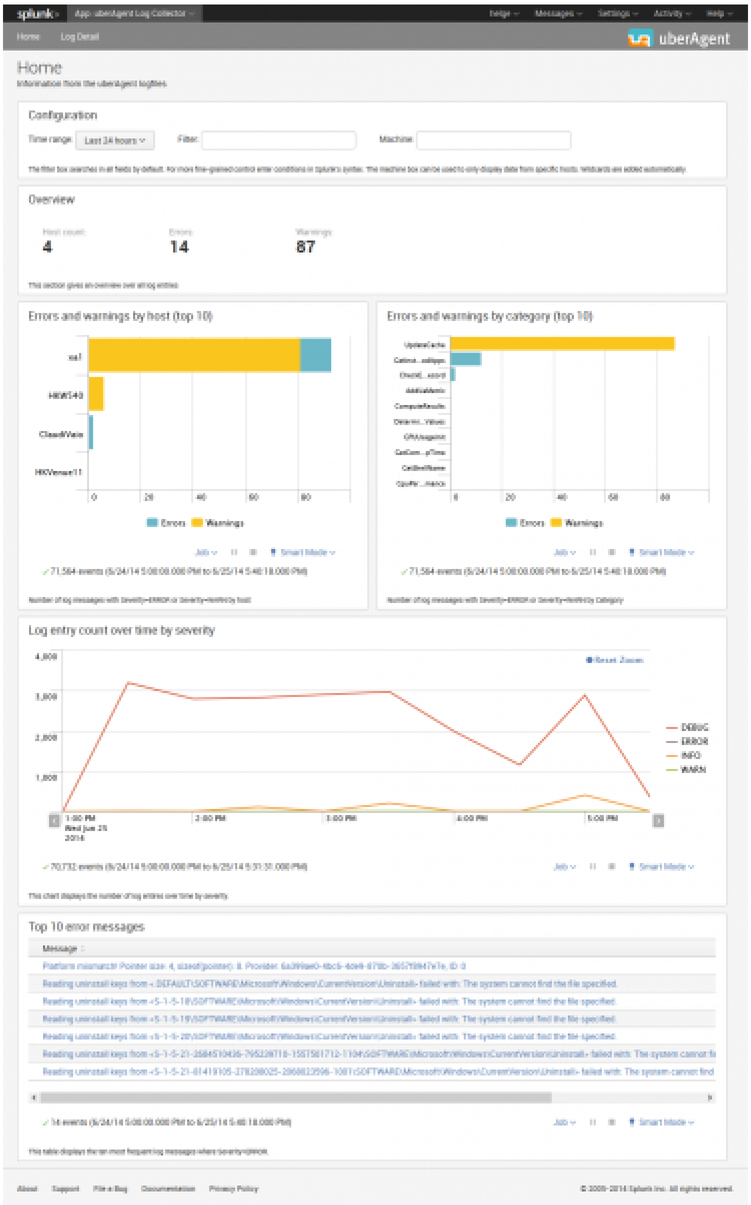
Version 1.2

- Complete dashboard redesign including various filtering capabilities
- Changed all dashboard searches from raw to pivot
- Added support for macOS log files
- Added Splunk data model `uberAgentMeta`
- Added additional logfile `uAInSessionHelper.log`
- Added new sourcetype `uberAgent:Meta:uAInSessionHelperLog`
- Added additional logfile `uberAgentIEExtension.log`
- Added new sourcetype `uberAgent:Meta:uAIEExtensionLog`
- Added additional logfile `uberAgentSandbox.log`
- Added new sourcetype `uberAgent:Meta:uASandboxLog`
- Changed the default index name from `uberagent_log` to `ua_meta_log`

- Added `macros.conf` to centrally rename the index
- Changed the sourcetype name from `uberAgent:log` to `uberAgent:Meta:uberAgentLog`

uberAgent Log Collector Splunk App

uberAgent maintains a very detailed and informative log file that can tell you a lot not only about uberAgent's health but also about the machine uberAgent is running on. Naturally, the log file is stored locally on the computer uberAgent is running on which makes analysis and troubleshooting a bit difficult in large environments. But luckily it is very easy to solve that problem with Splunk!



✓ 71,584 events (6/24/14 5:00:00.000 PM to 6/25/14 5:40:18.000 PM)

✓ 70,732 events (6/24/14 5:00:00.000 PM to 6/25/14 5:31:31.000 PM)

This chart displays the number of log entries over time by severity

Message
Platform mismatch! Pointer size: 4, sizeof(pointer): 8. Provider: 6a3f6ae0-4bc6-4de8-878b-365789847e1e, ID: 0
Reading uninstall keys from <.DEFAULT\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall> failed with: The system cannot find the file specified.
Reading uninstall keys from <S-1-5-18\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall> failed with: The system cannot find the file specified.
Reading uninstall keys from <S-1-5-19\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall> failed with: The system cannot find the file specified.
Reading uninstall keys from <S-1-5-20\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall> failed with: The system cannot find the file specified.
Reading uninstall keys from <S-1-5-21-2684510436-795228718-1557581712-1104\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall> failed with: The system cannot fi
Reading uninstall keys from <S-1-5-21-81419105-278208025-2068823596-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall> failed with: The system cannot find

✓ 14 events (6/24/14 5:00:00.000 PM to 6/25/14 5:40:18.000 PM)

This table displays the ten most frequent log messages where Severity=ERROR

What is it

uberAgent Log Collector is a set of associated Splunk apps that collect the data logged by uberAgent, send it to Splunk for indexing and provide dashboards for easy access.

Installation

uberAgent Log Collector consists of the Splunk app containing the dashboards and a technology add-on (TA) for collecting the data with Splunk's Universal Forwarder. These two components need to be installed on the following systems:

- **App:** search head(s)
- **TA:** endpoints where uberAgent and the Splunk Universal Forwarder are deployed

Splunk Index

You need to create the Splunk index `ua_meta_log` that stores the logs.

To add the new index `ua_meta_log` with the CLI run `splunk add index ua_meta_log`. The full documentation on creating Splunk indexes is available in the Splunk docs.

Configuration

Configurable Log Path

Since uberAgent 7.3, the log path is configurable. If you set a custom log path, you have to modify the **TA** app: copy the **`default`**`/inputs.conf` to `local/inputs.conf` and adjust the paths accordingly.

System Requirements

The TA requires Splunk's Universal Forwarder to be installed on the same machine.

Download

The *uberAgent Log Collector* apps are available in the Splunk App Directory:

- Download uberAgent Log Collector **app**
- Download uberAgent Log Collector **TA**