# Working with vRealize Log Insight Agents

March 01, 2023
vRealize Log Insight 8.6

You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/

# Contents

# Working with vRealize Log Insight Agents

*Working with vRealize Log Insight Agents* describes how to install and configure vRealize™ Log Insight™ Windows and Linux agents. It also includes troubleshooting tips.

This information is intended for anyone who wants to install, configure, or troubleshoot Log Insight Agents. The information is written for experienced Windows or Linux system administrators who are familiar with virtual machine technology and data center operations. Administrators are users assigned the Super Admin role or a role that has all the permissions of the Super Admin role. Users assigned a role with edit access for agents can also configure Log Insight Agents.

For information about how to create configuration classes for agents with the vRealize Log Insight server, refer to *Administering vRealize Log Insight*.

# Overview of vRealize Log Insight Agents

A vRealize Log Insight agent collects logs from log files and forwards them to a vRealize Log Insight server or any third-party syslog destination.

Agents support syslog and the vRealize Log Insight ingestion API (cfapi protocol) and can be used with Linux or Windows platforms. You configure agents through the web interface, with the liagent.ini file on the server and client side, or as part of installation.

Agents include the following features:

- Single or group deployment

- Automatic upgrade

- Parsing that operates on log messages and extracts structured data. You can configure parsers for FileLog and WinLog collectors or both.

- Support for multi-line messages

- Native support for several log rotation schemes

- An extensive ingestion API that includes client-side compression, encryption, and the ability to add metadata to logs

The vRealize Log Insight server supports centralized configuration management and creation and management of groups of agents.

The following figure shows the elements of an agent deployment configuration.
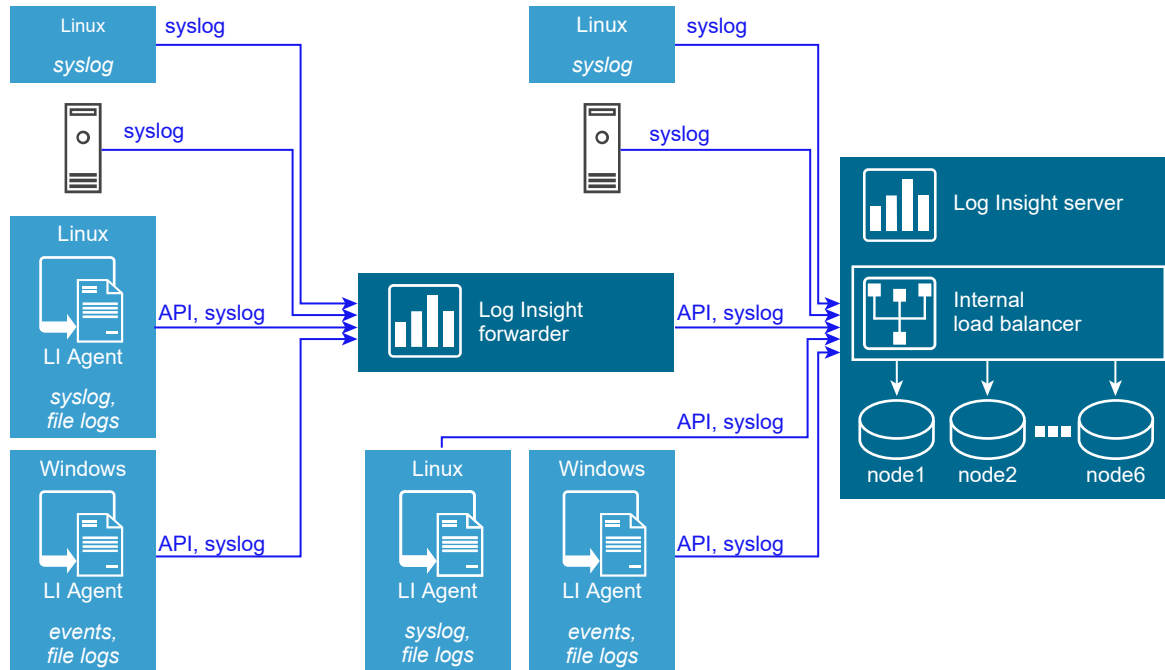
A vRealize Log Insight log forwarder is a dedicated instance of a vRealize Log Insight server whose primary job is to forward logs to a remote destination. Normally, a server instance used as a forwarder is not used for query. The forwarder uses an internal load balancer and is otherwise structured like a vRealize Log Insight server.

Agents write their own operation logs. For Windows, these logs are located in the `C:\ProgramData\VMware\Log Insight Agent\logs` directory. For Linux, the path for the operation log is `/var/log/loginsight-agent/liagent_*.log`. Log files are rotated when an agent is restarted or when the file reaches a size of 10 MB. A combined limit of 50 MB of files is kept in rotation. You cannot collect agent logs with the vRealize Log Insight agent itself.

Agents are used for real-time log collection. Use the vRealize Log Insight Importer to import historic log collections, including support bundles.

Separate installation downloads for Windows and Linux operating systems are provided.

On Windows systems, the agent runs as a Windows service and starts immediately after installation. The agent monitors application log files and Windows event channels, pools for collecting related Windows system logs. Collected logs are forwarded to vRealize Log Insight servers or third-party syslog destinations.

On Linux systems, the agent runs as a daemon and starts immediately after installation. The vRealize Log Insight Linux agent collects logs from log files on Linux machines and forwards them to vRealize Log Insight servers or syslog destinations. Debian, Red Hat, and Linux binary installation packages are available.

# Log Rotation Schemes Supported by vRealize Log Insight Agents

# 2

vRealize Log Insight agents support several log rotation schemes.

Log rotation ensures that log files do not grow infinitely. There are several log rotation schemes, each designed for a particular set of use cases. vRealize Log Insight includes native support for the following schemes.

Table 2-1. Log Rotation Schemes Supported by vRealize Log Insight Agents

| Log rotation scheme | Description |
| --- | --- |
| `create-new` | New log files are created when a size or time limit is reached. The logger process stops writing to the current log file and directs log output to a newly created file. No existing file is renamed or touched in any other way. |
| `rename-recreate` | An external utility such as `logrotate` renames the log file when a size or time limit is reached. The logger process then creates a log file with the previous name. |
| `copy-truncate` | An external utility such as `logrotate` copies the log file when a size or time limit is reached. The log process renames the copied file and truncates the original file so that its size becomes 0. The logger process can continue to write logs to the original file. |

# Installing or Upgrading vRealize Log Insight Agents

3

You can install or upgrade vRealize Log Insight agents on Windows machines or Linux machines, including machines with third-party log management systems.

Agents collect logs and forward them to the vRealize Log Insight server. During installation, you can specify parameters for the server, port, and protocol settings or keep the default settings. For the installation instructions, on the **Log Sources** tab, click **Agents** in the left pane and then click **LI Agent**.

You can upgrade agents using the same methods you use for installation, or you can use auto-upgrade. Auto-upgrade propagates upgrades to agents when you deploy a new version of vRealize Log Insight. See Automatic Update for vRealize Log Insight Agents for more information. Upgrade is not available for Linux binary packages.

## Hardware Support

To install and run a vRealize Log Insight agent, your hardware must support the minimum parameters required for hosts or machines that support x86 and x86_64 architecture and MMX, SSE, SSE2, and SSE3 instruction sets.

## Platform Support

| Operating System | Processor Architecture |
|---|---|
| Windows 7, Windows 8, Windows 8.1, and Windows 10 | x86_64, x86_32 |
| Windows Server 2008, Windows Server 2008 R2, | x86_64, x86_32 |
| Windows Server 2012, Windows Server 2012 R2, Windows Server 2016, and Windows Server 2019 | x86_64 |
| RHEL 5, RHEL 6, RHEL 7, and RHEL 8 | x86_64, x86_32 |
| SuSE Enterprise Linux (SLES) 11 SP3 and SLES 12 SP1 | x86_64 |
| Ubuntu 14.04 LTS, Ubuntu 16.04 LTS, and Ubuntu 18.04 | x86_64 |
| VMware Photon, version 1 revision 2, version 2, and version 3 | x86_64 |

# Linux Notes

If you implement a default installation of the Log Insight Linux Agent for a user without root privileges, the default configuration might create problems with the data collection. The agent does not log a warning that the subscription to the channel is unsuccessful, and files in the collection do not have read permissions. The message `Inaccessible log file ... will try later` is repeatedly added to the log. You can comment out the default configuration that is causing the problem or change the user permissions.

If you use an rpm or DEB package to install Linux agents, the init.d script named `liagentd` is installed as part of the package installation. The bin package adds the script, but does not register it. You can register the script manually.

You can verify that the installation was successful by running the (/sbin/) service `liagentd` status command.

This chapter includes the following topics:

- Download Agent Installation Files
- Installing Windows Agents
- Install or Update the vRealize Log Insight Linux Agent RPM package
- Install or Update the vRealize Log Insight Linux Agent DEB Package
- Customizing Your Agent Installation for Debian Linux
- Install the Log Insight Linux Agent Binary Package
- Command-line Options for vRealize Log Insight Agent Installation on Linux
- Automatic Update for vRealize Log Insight Agents

# Download Agent Installation Files

The first step to setting up a vRealize Log Insight agent is to download an agent installation package for your platform.

All packages downloaded from the vRealize Log Insight server agent page include the destination hostname appended to the package name. The server.hostname is applied during an initial installation for the MSI, RPM, and DEB agents. If a hostname is present in the configuration file, or if you are running the package by the hostname parameter, the embedded server hostname is ignored.

**Procedure**

1   Navigate to the **Administration** tab of the vRealize Log Insight web user interface.

2   In the Management section, click **Agents**.

3   Scroll to the bottom of the screen and click **Download Log Insight Agent**.

**4** Download an installation package by selecting it from the pop-up menu and clicking **Save**.

| Option | Description |
| --- | --- |
| Windows MSI | Installation package for a Windows platform (32-bit/64-bit) |
| Linux RPM | Installation package for a Linux Red Hat, openSuSE (32-bit/64-bit), or VMware Photon platform |
| Linux DEB | Installation package for a Linux Debian platform (32-bit/64-bit) |
| Linux BIN | Self-installing package for Linux (32-bit/64-bit). A package management system is not required. |

**What to do next**

Use the downloaded files to deploy the vRealize Log Insight agent.

# Installing Windows Agents

You can install an agent on a single machine through an installation wizard or through the command-line, or you can deploy multiple instances of an agent by using a script.

## Upgrading Windows Agents

You can upgrade a Windows agent by applying an upgrade file using any of the methods you can use to install. You can also choose to use the auto-upgrade feature to upgrade your agents in the background.

## Install or Update the vRealize Log Insight Windows Agent with the Installation Wizard

You can install or upgrade a Windows agent on a single machine with the installation wizard.

**Prerequisites**

- Verify that you have a copy of the vRealize Log Insight Windows agent `.msi` file. See Download Agent Installation Files.

- Verify that you have permissions to perform installations and start services on the Windows machine.

**Procedure**

**1** Log in to the Windows machine on which to install the vRealize Log Insight Windows agent.

**2** Change to the directory where you have the vRealize Log Insight Windows agent `.msi` file.

**3** Double-click the vRealize Log Insight Windows agent `.msi` file, accept the terms of the License Agreement, and click **Next**.

**4**    Enter the IP address or host name of the vRealize Log Insight server and click **Install**.

The wizard installs or updates the vRealize Log Insight Windows agent as an automatic Windows Service under the Local System service account.

**5**    Click **Finish**.

**What to do next**

Configure the vRealize Log Insight Windows agent by editing `liagent.ini` file. See Configure the Log Insight Windows Agent .

## Install or Update the vRealize Log Insight Windows Agent from the Command Line

You can install or update the Windows agent from the command line.

You can use the default or specify a service account, and use command-line parameters to specify server, port, and protocol information. For MSI command-line options, see the Microsoft Developer Network (MSDN) Library website and search for MSI command-line options.

**Prerequisites**

▪   Verify that you have a copy of the vRealize Log Insight Windows agent `.msi` file. See Download Agent Installation Files.

▪   Verify that you have permissions to perform installations and start services on the Windows machine.

▪   If you use the silent installation options `/quiet` or `/qn`, verify that you run the installation as an administrator. If you are not an administrator and run silent installation, the installation does not prompt for administrator privileges and fails. Use the logging option and parameters `/lxv*` *file_ name* for diagnostic purposes.

**Procedure**

**1**    Log in to the Windows machine on which to install or update the vRealize Log Insight Windows agent.

**2**    Open a **Command Prompt** window.

**3**    Change to the directory where you have the vRealize Log Insight Windows agent `.msi` file.

**4**    Install or update with default values with a command of the following form. Replace *version-build_ number* with your version and build number.

The `/quiet` option runs the command silently, and the `/lxv` option creates a log file in the current directory.

```
Drive:\path-to-msi_file>VMware-Log-Insight-Agent-version-build_number.msi
/quiet /lxv* li_install.log
```

**5** (Optional) Specify a user service account for the vRealize Log Insight Windows agent service to run under.

```
Drive:\path-to-msi_file>VMware-Log-Insight-Agent-*.msi SERVICEACCOUNT=domain\user
SERVICEPASSWORD=user_password
```

**Note** The account supplied in the `SERVICEACCOUNT` parameter is granted with the **Log On As a Service** right and full-write access to the `%ProgramData%\VMware\Log Insight Agent` directory. If the supplied account does not exist, it is created. The user name must not exceed 20 characters. If you do not specify a `SERVICEACCOUNT` parameter, the vRealize Log Insight Windows agent service is installed or updated under the LocalSystem service account.

**6** (Optional) You can specify values for the following command-line options as needed.

| Option | Description |
|---|---|
| `SERVERHOST=`*hostname* | IP address or host name of the vRealize Log Insight virtual appliance. |
| `SERVERPROTO=`*protocol* | Protocol that the agent uses to send log events to the vRealize Log Insight server. The possible values are `cfapi` and `syslog`. The default is cfapi. |
| `SERVERPORT=`*portnumber* | Communication port that the agent uses to send log events to the vRealize Log Insight or third party server. By default the agent uses the appropriate port based on the options that are set for SSL and the protocol. See default port values provided in the list below. You need to specify the port option only if it's different from these defaults. ■ cfapi with SSL activated: 9543 ■ cfapi with SSL deactivated: 9000 ■ syslog with SSL activated: 6514 ■ syslog with SSL deactivated: 514 |
| `SERVICEACCOUNT=`*account-name* | User service account under which the Log Insight Windows Agent service is run. **Note** The account supplied in the `SERVICEACCOUNT` parameter must have the **Log On As a Service** privilege and write access to `%ProgramData%\VMware\Log Insight Agent` directory so that the installer runs correctly. If you do not specify a `SERVICEACCOUNT` parameter, the vRealize Log Insight Windows agent service is installed under the LocalSystem service account. |
| `SERVICEPASSWORD=`*password* | Password for the user service account. |
| `AUTOUPDATE={yes\|no}` | Activate or deactivate auto-update for the agent. You must also activate auto-update from the vRealize Log Insight server to fully activate auto-update. The default is yes. |
| `LIAGENT_SSL={yes\|no}` | Activate secure connection. If SSL is activated, the agent uses TLS 1.2 protocol to communicate to the server. The default is yes. |

**Results**

The command installs or updates the vRealize Log Insight Windows agent as a Windows service. The vRealize Log Insight Windows agent service starts when you start the Windows machine.

**What to do next**

Verify that the command-line parameters you set are applied correctly in the `liagent.ini` file. See Configure the Log Insight Windows Agent .

# Deploy the Log Insight Windows Agent to Multiple Machines

You can perform a mass deployment of the Log Insight Windows Agent on target computers in a Windows domain.

**Procedure**

1  Create a Transform File to Deploy Multiple vRealize Log Insight Windows Agents

    As part of deploying multiple agents, you must create a transform file that specifies configuration parameters for deployment. The .mst transform file is applied to the .msi file when you install agents. Parameters include the destination server for the agents and the communication protocol, port, and user account for installing and starting the Log Insight agent service.

2  Deploy Multiple Instances of the vRealize Log Insight Windows Agent

    You can deploy multiple instances of the vRealize Log Insight Windows agent on target computers in a Windows domain.

## Create a Transform File to Deploy Multiple vRealize Log Insight Windows Agents

As part of deploying multiple agents, you must create a transform file that specifies configuration parameters for deployment. The .mst transform file is applied to the .msi file when you install agents. Parameters include the destination server for the agents and the communication protocol, port, and user account for installing and starting the Log Insight agent service.

Parameters include the destination server for the agents and the communication protocol, port, and user account for installing and starting the Log Insight agent service.

**Prerequisites**

▪  Verify that you have a copy of the vRealize Log Insight Windows `.msi` file. See Download Agent Installation Files.

▪  Download and install the Orca database editor. See http://support.microsoft.com/kb/255905.

**Procedure**

1  Open the vRealize Log Insight Windows agent `.msi` file in the Orca editor and select **Transform > New Transform**.

**2** Edit the Property table and add necessary parameters and values for a customized installation or upgrade.

Figure 3-1. Property Table
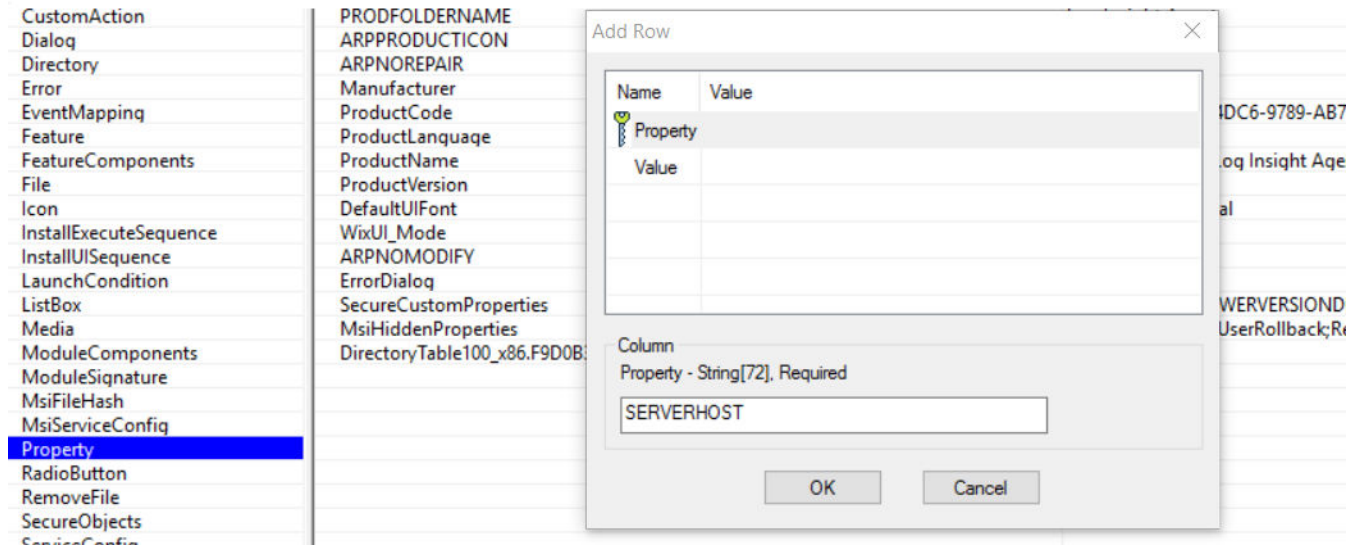


a Click **Property**.

b From the **Table** drop-down menu, select **Add Row**.

c Enter a property name and value in the Add Row dialog box.

Parameters are shown in the following table.

| Parameter | Description |
| --- | --- |
| **SERVERHOST** | IP address or host name of the vRealize Log Insight virtual appliance.<br>The default is **loginsight**. |
| **SERVERPROTO** | Protocol that the agent uses to send log events to the vRealize Log Insight server. The possible values are `cfapi` and `syslog`.<br>The default is cfapi. |
| **SERVERPORT** | Communication port that the agent uses to send log events to the vRealize Log Insight or third party server. By default the agent uses the appropriate port based on the options that are set for SSL and the protocol. See default port values provided in the list below. You need to specify the port option only if it's different from these defaults.<br>■ cfapi with SSL activated: 9543<br>■ cfapi with SSL deactivated: 9000<br>■ syslog with SSL activated: 6514<br>■ syslog with SSL deactivated: 514 |

| Parameter | Description |
| --- | --- |
| SERVICEACCOUNT | User service account under which the Log Insight Windows Agent service is run. |
| | **Note**  The account supplied in the SERVICEACCOUNT parameter must have the **Log On As a Service** privilege and write access to %ProgramData% \VMware\Log Insight Agent directory so that the installer runs correctly. If you do not specify a SERVICEACCOUNT parameter, the vRealize Log Insight Windows agent service is installed under the LocalSystem service account. |
| SERVICEPASSWORD | Password for the user service account. |
| AUTOUPDATE | Activate or deactivate auto-update for the agent. You must also activate auto-update from the vRealize Log Insight server to fully activate auto-update. The default is yes. |
| LIAGENT_SSL | Activate secure connection. If SSL is activated, the agent uses TLS 1.2 protocol to communicate to the server. The default is yes. |

**3** Select **Transform > Generate Transform** and save the .mst transform file.

**What to do next**

Use the .msi and .mst files to deploy the vRealize Log Insight Windows agent.

## Deploy Multiple Instances of the vRealize Log Insight Windows Agent

You can deploy multiple instances of the vRealize Log Insight Windows agent on target computers in a Windows domain.

For more information about why you need to reboot the client machine twice, see http:// support.microsoft.com/kb/305293.

**Prerequisites**

▪ Verify that you have an administrator account or an account with administrative privileges on the domain controller.

▪ Verify that you have a copy of the vRealize Log Insight Windows agent .msi file. See Download Agent Installation Files.

▪ Familiarize yourself with the procedures described in http://support.microsoft.com/kb/887405 and http://support.microsoft.com/kb/816102.

**Procedure**

**1** Log in to the domain controller as an administrator or a user with administrative privileges.

**2** Create a distribution point and copy the vRealize Log Insight Windows agent .msi file to the distribution point.

**3** Open the Group Policy Management Console and create a Group Policy Object to deploy the vRealize Log Insight Windows agent .msi file.

**4** Edit the Group Policy Object for software deployment and assign a package.

5   (Optional) If you generated an `.mst` file before deployment, select the `.mst` configuration file on the **Modifications** tab of the **GPO Properties** window. and use the Advanced method to edit a Group Policy Object to deploy the `.msi` package.

6   (Optional) Upgrade the vRealize Log Insight Windows agent.

   a   Copy the upgrade `.msi` file to the distribution point.

   b   Click the **Upgrade** tab on the Group Policy Object **Properties** window.

   c   Add the initially installed version of the `.msi` file in the Packages that this package will upgrade section.

7   Deploy the vRealize Log Insight Windows agent to specific security groups that include the domain users.

8   Close all Group Policy Management Console and Group Policy Management Editor windows on the domain controller and restart the client machines.

   If Fast Login Optimization is enabled, reboot the client machines twice.

9   Verify that vRealize Log Insight Windows agent is installed on the client machines as a local service.

   If you configured `SERVICEACCOUNT` and `SERVICEPASSWORD` parameters for using an `.mst` file to deploy multiple instances of vRealize Log Insight Windows agent, verify that vRealize Log Insight Windows agent is installed on the client machines under the user account that you specified.

**What to do next**

If the multiple instances of vRealize Log Insight Windows agent is not successful, see Mass Deployment of the Log Insight Windows Agent is Not Successful.

# Install or Update the vRealize Log Insight Linux Agent RPM package

You can install or update the vRealize Log Insight Linux agent as a root or non-root user and you can set configuration parameters during installation. After installation, you can verify the installed version.

**Prerequisites**

- Read about installation defaults and how to change them at Command-line Options for vRealize Log Insight Agent Installation on Linux .

- Log in as **root** or use `sudo` to run console commands.

- The vRealize Log Insight Linux agent needs access to syslog and networking services to function. Install and run the vRealize Log Insight Linux agent on runlevels 3 and 5. If you want the vRealize Log Insight Linux agent to work under other runlevels, configure the system appropriately.

**Procedure**

**1** You can install or upgrade an agent from the console.

- To install the vRealize Log Insight Linux agent with default configuration settings, open a console and run the following command.

  ```
  rpm -i VMware-Log-Insight-Agent-<version-and-build-number>.rpm
  ```

- To upgrade the agent without changing current configuration settings, open a console and run the following command.

  ```
  rpm -Uhv VMware-Log-Insight-Agent-<version-and-build-number>.rpm
  ```

**2** (Optional) You can override the default configuration values for installation or the current configuration values during an update. You do this by specifying options as part of the install or upgrade command.

```
sudo <OPTION=value> rpm -i <version-and-build-number>.rpm
```

**3** (Optional) Verify the installed version by running the following command.

```
rpm -qa | grep Log-Insight-Agent
```

## Example: Linux Agent Install and Update Examples

- The following command installs the vRealize Log Insight agent for a Linux RPM distribution. It installs the agent on a separate server, assigns a non-default port number, and creates a vRealize Log Insight agent user.

  ```
  sudo SERVERHOST=myagentserver SERVERPORT=1234 LIAGENTUSER=liagent rpm -i VMware-Log-
  Insight-Agent-44.1234.rpm
  ```

- The following command updates the agent with the given rpm file. Current agent configuration is unchanged.

  ```
  rpm -Uhv VMware-Log-Insight-Agent-44.1234.rpm
  ```

# Install or Update the vRealize Log Insight Linux Agent DEB Package

You can install or update the vRealize Log Insight Linux agent DEB (Debian) package from the command line or through the debconf database. After installation, you can verify the installed version.

**Prerequisites**

- Read about installation defaults and how to change them at Command-line Options for vRealize Log Insight Agent Installation on Linux .

- Log in as **root** or use `sudo` to run console commands.

- Verify that the vRealize Log Insight Linux agent has access to syslog and networking services to function. By default, the vRealize Log Insight Linux agent runs on runlevels 2, 3, 4, and 5 and stops on runlevels 0, 1, and 6.

- For more information and examples, see Customizing Your Agent Installation for Debian Linux.

**Procedure**

1 To install or update the vRealize Log InsightLinux agent, open a console and run the `dpkg -i` *package_name* command.

The *package_name* consists of the prefix **vmware-log-insight-agent-** and the version build number of your download version.

The following command form installs the package with default values.

```
dpkg -i vmware-log-insight-agent-VERSION-BUILD_NUMBER_all.deb
```

2 (Optional) Verify the installed version by running the following command:

```
dpkg -l | grep -i vmware-log-insight-agent
```

**Example**

- Configure the connection protocol from the command line.

To override the default connection protocol, use the SERVERPROTO variable as shown in the following example:

```
sudo SERVERPROTO=syslog dpkg -i vmware-log-insight-agent-<version-and-build-number>_all.deb
```

# Customizing Your Agent Installation for Debian Linux

You can customize you installation by using command options to override the current configuration values for installation or by configuring the debconf database.

## Customization from the Command Line

To configure your installation from the command line, use a command of the following form:

```
sudo <OPTION=value> dpkg  -i vmware-log-insight-agent-<version-and-build-number>_all.deb
```

For a complete list of options, see Command-line Options for vRealize Log Insight Agent Installation on Linux .

The following examples show some show some typical configurations done from the command line.

■  Specify a target vRealize Log Insight server.

■  To set the target during installation, run the `sudo` command and replace hostname with the IP address or hostname of the vRealize Log Insight server as shown in the following example:

```
sudo SERVERHOST=hostname dpkg -iv mware-log-insight-agent-<version-and-build-
number>_all.deb
```

Unless you enabled the `--force-confold` flag during installation, whenever you update to a newer version, the system prompts you to keep or replace the liagent.ini configuration file. The following system message appears:

```
Configuration file `/var/lib/loginsight-agent/liagent.ini'
 ==> Modified (by you or by a script) since installation.
 ==> Package distributor has shipped an updated version.
   What would you like to do about it ?  Your options are:
    Y or I  : install the package maintainer's version
    N or O  : keep your currently-installed version
      D     : show the differences between the versions
      Z     : start a shell to examine the situation
 The default action is to keep your current version.
*** liagent.ini (Y/I/N/O/D/Z) [default=N] ?
```

To preserve the existing configuration, use [default=N]. The additional parameters passed from the command line are still applied.

■  Configure the connection protocol.

To override the default connection protocol, use the SERVERPROTO variable as shown in the following example:

```
sudo SERVERPROTO=syslog dpkg -i vmware-log-insight-agent-<version-and-build-number>_all.deb
```

■  Configure the connection port.

To override the default connection port, provide a value for the SERVERPORT variable to the installer as shown in the following example:

```
sudo SERVERPORT=1234 dpkg -i vmware-log-insight-agent-<version-and-build-number>_all.deb
```

■  Run the agent as a non-root user.

To run the vRealize Log Insight Linux agent as a **non-root** user, run the `sudo` command.

```
sudo LIAGENTUSER=liagent dpkg -i vmware-log-insight-agent-<version-build-number>_all.deb
```

If the specified user does not exist, the vRealize Log Insight Linux agent creates the user account during the installation. The created account is not deleted after uninstallation. If you install the Linux agent with the `LIAGENTUSER=`*non_root_user* parameter and try to upgrade with the `LIAGENTUSER=`*non_root_user2* parameter, a conflict occurs. Warnings appear because the *non_root_user2* user does not have the permissions of the *non_root_user* user.

## DEB Package Customization Options for the debconf Database

The agent DEB package can also be configured through the debconf database. The following table shows supported debconf options and corresponding vRealize Log Insight agent DEB installer options:

| Command-line Options | Debconf Options | Description |
| --- | --- | --- |
| `SERVERHOST=`*hostname* | vmware-log-insight-agent/serverhost | IP address or host name of the vRealize Log Insight virtual appliance.<br>The default is **loginsight**. |
| `SERVERPROTO={`**cfapi**\|**syslog** `}` | vmware-log-insight-agent/serverproto | Protocol that the agent uses to send log events to the vRealize Log Insight server. The possible values are `cfapi` and `syslog`.<br>The default is cfapi. |
| `SERVERPORT=`*portnumber* | vmware-log-insight-agent/serverport | Communication port that the agent uses to send log events to the vRealize Log Insight or third party server. By default the agent uses the appropriate port based on the options that are set for SSL and the protocol. See default port values provided in the list below. You need to specify the port option only if it's different from these defaults.<br><br>■ cfapi with SSL activated: 9543<br>■ cfapi with SSL deactivated: 9000<br>■ syslog with SSL activated: 6514<br>■ syslog with SSL deactivated: 514 |
| `LIAGENT_INITSYSTEM={`**init**\|**systemd**`}` | log-insight-agent/init_system | During install time, the agent automatically detects the type of init system for the machine you are installing the agent on. You can override this behavior by specifying the type of system value with this option. There are two types of supported init systems: `init` and `systemd`. |
| `LIAGENT_AUTOUPDATE={`**yes**\|**no**`}` | vmware-log-insight-agent/auto_update | Activate or deactivate auto-update for the agent. You must also activate auto-update from the vRealize Log Insight server to fully activate auto-update. The default is yes.<br>Auto-update is not supported for Linux BIN packages. |

| Command-line Options | Debconf Options | Description |
|---|---|---|
| `LI_AGENT_RUNSERVICES` | vmware-log-insight-agent/init_system | Immediately after the installation, the services `liagentd` (agent) and `liupdaterd` (updater) are started by default. You can prevent them from starting by setting the LIAGENT_RUNSERVICES debconf parameter to **no**. The default is yes. The only accepted values are **yes** and **no**; **1** or **0** are not supported values. |
| `LIAGENT_SSL` | vmware-log-insight-agent/ssl | C |
| `LIAGENTUSER=`*user-account-name* | vmware-log-insight-agent/liagentuser | Specifies an account under which the agent is run. If the user does not exist, the installer creates it as a regular user. If the specified user account does not exist, the vRealize Log Insight Linux agent creates the user account during the installation. The created account is not deleted after uninstallation.<br><br>By default the agent is installed to run as a root user.<br><br>If you install the agent with the `LIAGENTUSER=`*non_root_user* parameter and try to upgrade with `LIAGENTUSER=`*non_root_user2*, a conflict occurs. Warnings appear because *non_root_user2* user does not have the permissions of the user *non_root_user*.<br><br>The created user is not removed during uninstall. It can be removed manually. This parameter is intended for the agent service only. The updater service is always running as a root user. |

# Install the Log Insight Linux Agent Binary Package

Installing the binary package includes changing the .bin file to an executable file and then installing the agent.

Upgrading the .bin package is not officially supported. If you used the .bin package to install an existing Log Insight Linux Agent, make a backup copy of the `liagent.ini` file located in `/var/lib/loginsight-agent` directory to keep the local configuration. After you have a backup copy, manually uninstall the Log Insight Linux Agent. See Manually Uninstall the Log Insight Linux Agent bin package.

If you use the .bin package to install Linux agents, the `init.d` script named `liagentd` is installed as part of the package installation, but the package does not register the script. You can register the script manually.

You can verify that the installation is successful by running `(/sbin/)service liagentd status` command.

### Prerequisites

■  Download and copy the Log Insight Linux Agent `.bin` package to the target Linux machine.

- Verify that the Log Insight Linux Agent has access to syslog and networking services.

- Read about default configuration values and how to change them at installation. See
  Command-line Options for vRealize Log Insight Agent Installation on Linux .

**Procedure**

1 Open a console and run the `chmod` command to change the `.bin` file to an executable file.

   Replace *filename-version* with the appropriate version.

   ```
   chmod +x filename-version.bin
   ```

2 From a command prompt, run the `./filename-version.bin` command to install the agent.

   Replace *filename-version* with the appropriate version.

   ```
   ./filename-version.bin
   ```

3 (Optional) To set the target vRealize Log Insight server during installation, run the `sudo SERVERHOST=hostname ./filename-version.bin` command.

   Replace *hostname* with the IP address or hostname of the vRealize Log Insight server.

   ```
   sudo SERVERHOST=hostname ./filename-version.bin
   ```

4 (Optional) To override the default connection protocol use the SERVERPROTO variable as shown in the following example:

   ```
   sudo SERVERPROTO=syslog ./filename-version.htm
   ```

5 (Optional) To override the default connection port provide a value for the SERVERPORT variable to the installer as shown in the following example:

   ```
   sudo SERVERPORT=1234 ./filename-version.htm
   ```

6 (Optional) To run the Log Insight Linux Agent as a **non-root** user run the `sudo` command.

   ```
   sudo LIAGENTUSER=liagent ./filename-version.bin
   ```

   If the specified user does not exist, the Log Insight Linux Agent creates the user account during the installation. The created account is not deleted after uninstallation. If you install the Log Insight Linux Agent with the `LIAGENTUSER=non_root_user` parameter and try to upgrade with the `LIAGENTUSER=non_root_user2` parameter, a conflict occurs and warnings appear because the *non_root_user2* user does not have the permissions of the *non_root_user* user.

# Command-line Options for vRealize Log Insight Agent Installation on Linux

When you install vRealize Log Insight agents from the command line, you can include options to configure your deployment during installation. These options correspond to settings in the `liagent.ini` file.

The following options can be used during installation to configure vRealize Log Insight agents that run on Linux systems.

| Option | Description |
|---|---|
| SERVERHOST=*hostname* | IP address or host name of the vRealize Log Insight virtual appliance.<br>The default is **loginsight**. |
| SERVERPROTO={**cfapi**\|**syslog**} | Protocol that the agent uses to send log events to the vRealize Log Insight server. The possible values are `cfapi` and `syslog`.<br>The default is cfapi. |
| SERVERPORT=*portnumber* | Communication port that the agent uses to send log events to the vRealize Log Insight or third party server. By default the agent uses the appropriate port based on the options that are set for SSL and the protocol. See default port values provided in the list below. You need to specify the port option only if it's different from these defaults.<br>■ cfapi with SSL activated: 9543<br>■ cfapi with SSL deactivated: 9000<br>■ syslog with SSL activated: 6514<br>■ syslog with SSL deactivated: 514 |
| LIAGENT_INITSYSTEM={**init**\|**systemd**} | During install time, the agent automatically detects the type of init system for the machine you are installing the agent on. You can override this behavior by specifying the type of system value with this option. There are two types of supported init systems: `init` and `systemd`. |
| LIAGENT_AUTOUPDATE={**yes**\|**no**} | Activate or deactivate auto-update for the agent. You must also activate auto-update from the vRealize Log Insight server to fully activate auto-update. The default is yes.<br>Auto-update is not supported for Linux BIN packages. |

| Option | Description |
|--------|-------------|
| `LIAGENT_SSL={`**`yes`**`|`**`no`**`}` | Activate secure connection. If SSL is activated, the agent uses TLS 1.2 protocol to communicate to the server. The default is yes. |
| `LIAGENTUSER=`*user-account-name* | Specifies an account under which the agent is run. If the user does not exist, the installer creates it as a regular user. If the specified user account does not exist, the vRealize Log Insight Linux agent creates the user account during the installation. The created account is not deleted after uninstallation. |
| | By default the agent is installed to run as a root user. |
| | If you install with the `LIAGENTUSER=`*non_root_user* parameter and try to upgrade with `LIAGENTUSER=`*non_root_user2*, a conflict occurs. Warnings appear because *non_root_user2* user does not have the permissions of the user *non_root_user* |
| | The created user is not removed during uninstall. It can be removed manually. This parameter is intended for the agent service only. The updater service is always running as a root user. |

# Automatic Update for vRealize Log Insight Agents

The auto-update feature for vRealize Log Insight agents allows active agents to check, download, and automatically update based on the agent install packages from the vRealize Log Insight server.

You can enable auto-update from the server for all agents, or from clients for individual agent instances. Agents must have an active status and be version 4.3 or later.

Auto-update is not supported for Linux BIN packages.

## Deactivate or Activate Auto-Update for Individual Agents

You can enable or deactivate auto-update for individual agents by editing the client-side configuration file for that agent.

By default, auto-update is enabled from the client side for an agent.

### Prerequisites

Agents must be version 4.3 or later.

### Procedure

1  Open the local `liagent.ini` file in an editor.

**2** Locate the `[update]` section.

It looks similar to the following example.

```
[update]
; Do not change this parameter
package_type=msi
; Enable automatic update of the agent. If enabled:
; the agent will silently check for updates from the server and
; if available will automatically download and apply the update.
; auto_update=yes
```

**3** To deactivate auto-update, uncomment `auto_update=yes` and change it to `auto_update=no`.

**Note**  Auto-update for agents is enabled by default. So, the default value for `auto_update` is "yes", even when commented.

**4** Save and close the `liagent.ini` file.

# Configuring vRealize Log Insight Agents

<div style="text-align: right; font-size: 4em; color: #cccccc;">4</div>

After you have deployed an agent, you can configure it to send logs to the vRealize Log Insight server that you select, specify communication protocols, and set other parameters.

Use these instructions as required to configure your agents to your requirements.

■ Configure the Log Insight Windows Agent

You can configure the Log Insight Windows Agent after you install it. Edit the `liagent.ini` file to configure Log Insight Windows Agent to send logs to a vRealize Log Insight, set the communication protocol and port, add Windows event channels, and configure flat file log collection. The file is located in the `%ProgramData%\VMware\Log Insight Agent` directory.

■ Configure the Log Insight Linux Agent

You can configure the Log Insight Linux Agent after you install it.

■ Filtering Log Events from vRealize Log Insight Agents

You can provide the information that an agent sends to a destination with the filter option in the `[server|<dest_id>]` section of your local `liagent.ini` file.

■ Forwarding Logs from a vRealize Log Insight Agent

You can forward logs collected by an agent to up to three destinations. A destination can include vRealize Log Insight servers or forwarder, or third-party log management solutions.

■ Centralized Configuration of vRealize Log Insight Agents

You can configure multiple vRealize Log Insight agents.

■ Using Common Values for Agent Configuration

You can override the default values of the agent configuration file with common parameter values that apply for each agent configuration section for Windows or Linux agents.

■ Parsing Logs

Agent-side log parsers extract structured data from raw logs before delivering to the vRealize Log Insight server. Using log parsers, vRealize Log Insight can analyze logs, extract information from them, and show those results on the server. Log parsers can be configured for both Windows and Linux vRealize Log Insight Agents.

# Configure the Log Insight Windows Agent

You can configure the Log Insight Windows Agent after you install it. Edit the `liagent.ini` file to configure Log Insight Windows Agent to send logs to a vRealize Log Insight, set the communication protocol and port, add Windows event channels, and configure flat file log collection. The file is located in the `%ProgramData%\VMware\Log Insight Agent` directory.

## Default Configuration of the Log Insight Windows Agent

After installation, the `liagent.ini` file contains pre-configured default settings for the Log Insight Windows Agent.

### Log Insight Windows Agent liagent.ini Default Configuration

If you use non-ASCII names and values, save the configuration as UTF-8.

If you are using central configuration, the final configuration is this file joined with settings from the server to form the `liagent-effective.ini` file.

You may find it more efficient to configure the settings from the server's agents page.

```
; Client-side configuration of VMware Log Insight Agent.
; See liagent-effective.ini for the actual configuration used by VMware Log Insight Agent.

[server]
; Log Insight server hostname or ip address
; If omitted the default value is LOGINSIGHT
;hostname=LOGINSIGHT

;Enables or deactivates centralized configuration from the vRealize Log Insight server.
;When enabled, agent configuration changes made to the liagent.ini file on the server
;are joined with the settings in this file. to this agent. Accepted values are yes or no and
0 or 1.
;The default is yes.
;
;central_config=yes
;

; Set protocol to use:
; cfapi - Log Insight REST API
; syslog - Syslog protocol
; If omitted the default value is cfapi
;
;proto=cfapi

; Log Insight server port to connect to. If omitted the default value is:
; for syslog: 514
; for cfapi without ssl: 9000
; for cfapi with ssl: 9543
;port=9000

;ssl - enable/deactivate SSL. Applies to cfapi protocol only.
; Possible values are yes or no. If omitted the default value is no.
```

```
;ssl=no

; Time in minutes to force reconnection to the server
; If omitted the default value is 30
;reconnect=30

; FIPS mode.
; Possible values are 1 or 0. If omitted the default value is 1.
; ssl_fips_mode=1

[storage]
;max_disk_buffer - max disk usage limit (data + logs) in MB:
; 100 - 2000 MB, default 200
;max_disk_buffer=2000

[logging]
;debug_level - the level of debug messages to enable:
;   0 - no debug messages
;   1 - trace essential debug messages
;   2 - verbose debug messages (will have negative impact on performance)
;debug_level=0
;
;The interval in minutes to print statistics
;stats_period=15

[update]
; Do not change this parameter
package_type=msi

; Enable automatic update of the agent. If enabled:
; the agent will silently check for updates from the server and
; if available will automatically download and apply the update.
;auto_update=yes

[winlog|Application]
channel=Application
raw_syslog=no

[winlog|Security]
channel=Security

[winlog|System]
channel=System
```

| Parameter | Default Value | Description |
|---|---|---|
| hostname | LOGINSIGHT | IP address or host name of the vRealize Log Insight virtual appliance.<br><br>The default is **loginsight**. |
| central_config | yes | Activate or deactivate centralized configuration for this agent.<br>When centralized configuration is deactivated, the agent ignores configuration provided by the vRealize Log Insight server. Accepted values are yes, no,1, or 0. The default value is yes. |
| proto | cfapi | Protocol that the agent uses to send log events to the vRealize Log Insight server. The possible values are cfapi and syslog.<br><br>The default is cfapi. |
| port | 9543, 9000, 6514, and 514 | Communication port that the agent uses to send log events to the vRealize Log Insight or third party server. By default the agent uses the appropriate port based on the options that are set for SSL and the protocol. See default port values provided in the list below. You need to specify the port option only if it's different from these defaults.<br>■ cfapi with SSL activated: 9543<br>■ cfapi with SSL deactivated: 9000<br>■ syslog with SSL activated: 6514<br>■ syslog with SSL deactivated: 514 |
| ssl | yes | Enables or deactivates SSL. The default value is yes.<br><br>When ssl is set to yes, if you do not set a value for the port, the port is automatically picked up as 9543. |
| max_disk_buffer | 200 | The maximum disk space in MB that the Log Insight Windows Agent uses to buffer events and its own logs.<br><br>When the specified max_disk_buffer is reached, the agent begins to drop new incoming events. |
| debug_level | 0 | Defines the log details level. See Define Log Details Level in the Log Insight Agents. |

| Parameter | Default Value | Description |
|-----------|---------------|-------------|
| channel | Application, Security, System | The Application, Security, and System Windows Event Log channels are commented by default; the Log Insight Windows Agent does not collect logs from these channels.<br><br>See Collect Logs from Windows Events Channels . |
| raw_syslog | no | For agents that use the syslog protocol, allows the agent to collect and send raw syslog events. The default is no, which means collected events are transformed with user-specified syslog attributes. Activate this option to collect log events without any syslog transformations.<br><br>Accepted values are yes or 1 and no or 0. |
| ssl_fips_mode | 1 | Enables or deactivates FIPS mode for the Log Insight Windows Agent through the liagent.ini file. Accepted values are 1 and 0. |

## Collect Logs from Windows Events Channels

You can add a Windows event channel to the Log Insight Windows Agent configuration. The Log Insight Windows Agent will collect the log events and send them to the vRealize Log Insight server.

Field names are restricted. The following names are reserved and cannot be used as field names.

- event_type

- hostname

- source

- text

### Prerequisites

Log in to the Windows machine on which you installed the vRealize Log Insight Windows agent and start the Services manager to verify that the vRealize Log Insight agent service is installed.

### Procedure

1  Navigate to the program data directory of the vRealize Log Insight Windows agent.

    %ProgramData%\VMware\Log Insight Agent

2  Open the liagent.ini file in any text editor.

**3** Add the following parameters and set the values for your environment.

| Parameter | Description |
|-----------|-------------|
| **[winlog\|**_section_name_**]** | A unique name for the configuration section. |
| **channel** | The full name of the event channel as shown in the Event Viewer built-in Windows application. To copy the correct channel name, right-click a channel in Event Viewer, select **Properties** and copy the contents of **Full Name** field. |
| **enabled** | An optional parameter to enable or deactivate the configuration section. The possible values are yes or no (case-insensitive). The default value is yes. |
| **tags** | Optional parameter to add custom tags to the fields of collected events. Define tags using JSON notation. Tag names can contain letters, numbers, and underscores. A tag name can only begin with a letter or an underscore and cannot exceed 64 characters. Tag names are not case-sensitive. For example, if you use tags={"tag_name1" : "tag value 1", "Tag_Name1" : "tag value 2" }, Tag_Name1 is ignored as a duplicate. You cannot use event_type and timestamp as tag names. Any duplicates within the same declaration are ignored. |
|  | If the destination is a syslog server, tags can override the APP-NAME field. For example, tags={"appname":"VROPS"}. |
| **whitelist, blacklist** | Optional parameters to explicitly include or exclude log events. |
|  | **Note**  The blacklist option only works for fields; it cannot be used to block text. |
| **exclude_fields** | (Optional) A parameter to exclude individual fields from collection. You can provide multiple values as a semicolon separated list. For example, `exclude_fields=EventId; ProviderName` |

```
[winlog|section_name]
channel=event_channel_name
enabled=yes_or_no
tags={"tag_name1" : "Tag value 1", "tag_name2" : "tag value 2" }
```

**4** Save and close the `liagent.ini` file.

## Example: Configurations

See the following `[winlog|` configuration examples.

```
[winlog|Events_Firewall ]
channel=Microsoft-Windows-Windows Firewall With Advanced Security/Firewall
enabled=no
```

```
[winlog|custom]
channel=Custom
tags={"ChannelDescription": "Events testing channel"}
```

## Set up Filtering for Windows Event Channels

You can set up filters for Windows Event channels to explicitly include or exclude log events.

You use the `whitelist` and `blacklist` parameters to evaluate a filter expression. The filter expression is a Boolean expression that consists of event fields and operators.

---

**Note**  The `blacklist` option only works for fields; it cannot be used to block text.

---

- The `whitelist` parameter collects only log events for which the filter expression evaluates to non-zero. If you omit this parameter, the value is an implied 1.

- The `blacklist` parameter excludes log events for which the filter expression evaluates to non-zero. The default value is 0.

For a complete list of Windows event fields and operators see Event Fields and Operators.

### Prerequisites

Log in to the Windows machine on which you installed the vRealize Log Insight Windows agent and start the Services manager to verify that the vRealize Log Insight agent service is installed.

### Procedure

1  Navigate to the program data directory of the vRealize Log Insight Windows agent.

    `%ProgramData%\VMware\Log Insight Agent`

2  Open the `liagent.ini` file in any text editor.

3  Add a `whitelist` or `blacklist` parameter in the `[winlog|]` section.

    For example

    ```
    [winlog|unique_section_name]
    channel = event_channel_name
    blacklist = filter_expression
    ```

4  Create a filter expression from Windows events fields and operators.

    For example

    ```
    whitelist = level > WINLOG_LEVEL_SUCCESS and level < WINLOG_LEVEL_INFO
    ```

5  Save and close the `liagent.ini` file.

### Example: Filter Configurations

You can configure the agent to collect only error events, for example

```
[winlog|Security-Error]
channel = Security
whitelist = Level == WINLOG_LEVEL_CRITICAL or Level == WINLOG_LEVEL_ERROR
```

You can configure the agent to collect only VMware Network events from Application channel, for example

```
[winlog|VMwareNetwork]
channel = Application
whitelist = ProviderName == "VMnetAdapter" or ProviderName == "VMnetBridge" or ProviderName
== "VMnetDHCP"
```

You can configure the agent to collect all events from Security channel except particular events, for example

```
[winlog|Security-Verbose]
channel = Security
blacklist = EventID == 4688 or EventID == 5447
```

## Event Fields and Operators

Use the Windows event fields and operators to build filter expressions.

### Filter Expression Operators

| Operator | Description |
| --- | --- |
| ==, != | equal and not equal. Use with both numeric and string fields. |
| >=, >, <, <= | greater or equal, greater than, less than, less than or equal. Use with numeric fields only. |
| &, \|, ^, ~ | Bitwise AND, OR, XOR and complement operators. Use with numeric fields only. |
| and, or | Logical AND and OR. Use to build complex expressions by combining simple expressions. |
| not | Unary logical NOT operator. Use to reverse the value of an expression. |
| () | Use parentheses in a logical expression to change the order of evaluation. |

### Windows Event Fields

You can use the following Windows event fields in a filter expression.

| Field name | Field type |
| --- | --- |
| Hostname | string |
| Text | string |
| ProviderName | string |
| EventSourceName | string |
| EventID | numeric |
| EventRecordID | numeric |
| Channel | string |
| UserID | string |

| Field name | Field type |
|---|---|
| Level | numeric |
| | You can use the following predefined constants |
| | ■ WINLOG_LEVEL_SUCCESS = 0 |
| | ■ WINLOG_LEVEL_CRITICAL = 1 |
| | ■ WINLOG_LEVEL_ERROR = 2 |
| | ■ WINLOG_LEVEL_WARNING = 3 |
| | ■ WINLOG_LEVEL_INFO = 4 |
| | ■ WINLOG_LEVEL_VERBOSE = 5 |
| Task | numeric |
| OpCode | numeric |
| Keywords | numeric |
| | You can use the following predefined bit masks |
| | ■ WINLOG_KEYWORD_RESPONSETIME = 0x0001000000000000; |
| | ■ WINLOG_KEYWORD_WDICONTEXT = 0x0002000000000000; |
| | ■ WINLOG_KEYWORD_WDIDIAGNOSTIC = 0x0004000000000000; |
| | ■ WINLOG_KEYWORD_SQM = 0x0008000000000000; |
| | ■ WINLOG_KEYWORD_AUDITFAILURE = 0x0010000000000000; |
| | ■ WINLOG_KEYWORD_AUDITSUCCESS = 0x0020000000000000; |
| | ■ WINLOG_KEYWORD_CORRELATIONHINT = 0x0040000000000000; |
| | ■ WINLOG_KEYWORD_CLASSIC = 0x0080000000000000; |

**Examples**

Collect all critical, error and warning events

```
[winlog|app]
channel = Application
whitelist = level > WINLOG_LEVEL_SUCCESS and level < WINLOG_LEVEL_INFO
```

Collect only Audit Failure events from Security channel

```
[winlog|security]
channel = Security
whitelist = Keywords & WINLOG_KEYWORD_AUDITFAILURE
```

## Collect Log Events from a Log File

You can configure the vRealize Log Insight Windows agent to collect log events from one or more log files.

Field names are restricted. The following names are reserved and cannot be used as field names.

■ event_type

■ hostname

■ source

■ text

You can have up to three destinations for agent information and filter the information before it is sent. See Forwarding Logs from a vRealize Log Insight Agent.

---

**Note**

- Monitoring a large number of files, such as a thousand or more, leads to higher resource utilization by the agent and impacts the overall performance of the host machine. To prevent this, configure the agent to monitor only the necessary files using patterns and globs, or archive the old log files. If monitoring a large number of files is a requirement, consider increasing the host parameters such as CPU and RAM.

- The agent can collect from encrypted directories, but only if it is run by the user who encrypted the directory.

- The agent supports only static directory structures. If the directories have been renamed or added, you must restart the agent to start monitoring these directories, provided the configuration covers the directories.

---

Prerequisites

Log in to the Windows machine on which you installed the vRealize Log Insight Windows agent and start the Services manager to verify that the vRealize Log Insight agent service is installed.

Procedure

**1** Navigate to the program data directory of the vRealize Log Insight Windows agent.

```
%ProgramData%\VMware\Log Insight Agent
```

**2** Open the `liagent.ini` file in any text editor.

**3** Locate the [server|<dest_id>] section of the file. Add configuration parameters and set the values for your environment.

```
[filelog|section_name]
directory=path_to_log_directory
include=glob_pattern
...
```

| Parameter | Description |
|---|---|
| **[filelog\|** *section_name* **]** | A unique name for the configuration section. |
| **directory=** *full-path-to-log-file* | The full path to the log file directory. Glob patterns are supported. Example configurations: <br><br>■ To collect from all sub-directories of `D:\Logs\new_test_logs`directory, use `directory=D:\Logs\new_test_logs\*` <br><br>■ If your sub-directories have their own sub-directories, use the following configuration to monitor all sub-directories `directory=D:\Logs\new_test_logs\*\*` <br><br>**Note** To limit the number of files and directories and avoid high resource consumption, you cannot define a directory glob for either the first or second level directories such as: `directory=c:/tmp/*` or `directory=c:\Logs\*`. The directory path must be at least two levels. <br><br>You can define a path to a non-existing directory, and the agent will collect the log files in that directory once the directory and files are created. <br><br>You can define the same directory under one or more different configuration sections, to collect logs from the same file multiple times. This process makes it possible to apply different tags and filters to the same source of events. <br><br>**Note** If you use identical configurations for these sections, duplicated events are observed on the server side. |

| Parameter | Description |
|---|---|
| **include=**_file_name; ..._ | (Optional) The name of a filename or a file mask (glob pattern) from which to collect data. You can provide values as a semicolon separated list. The default value is *, which means that all files are included. The parameter is case-sensitive.<br><br>A file mask (glob pattern) can be used to group files that follow the same naming convention, as well as within a single filename. For example, filenames that include spaces, such as `vRealize Ops Analytics.log` and `vRealize Ops Collector.log`, can be specified with `vRealize?Ops?Analytics*.log` or `vRealize*.log`. By using file masks, you can specify filenames that are acceptable for agent configuration under Linux and Windows hosts.<br><br>By default `.zip` and `.gz` files are excluded from collection.<br><br>**Important**  If you are collecting a rotated log file, use the `include` and `exclude` parameters to specify a glob pattern that matches both the primary and the rotated file. If the glob pattern matches only the primary log file, the vRealize Log Insight agents might miss events during rotation. The vRealize Log Insight agents automatically determine the correct order of rotated files and sends events to thevRealize Log Insight server in the right order. For example, if your primary log file is named `myapp.log` and rotated logs are `myapp.log.1` and `myapp.log.2` and so on, you can use the following `include` pattern:<br><br>`include= myapp.log;myapp.log.*` |
| **exclude=**_regular_expression_ | (Optional) A filename or file mask (glob pattern) to exclude from collection. You can provide values as a semicolon separated list. The default value is empty, which means that no file is excluded. |
| **event_marker=**_regular_expression_ | (Optional) A regular expression that denotes the start of an event in the log file. If omitted defaults to newline. The expressions you type must use the Perl regular expressions syntax.<br><br>**Note**  Symbols, for example quotation marks (" "), are not treated as wrappers for regular expressions. They are treated as part of the pattern.<br><br>Since the vRealize Log Insight agent is optimized for real-time collection, partial log messages written with an internal delay may be split into multiple events. If log file appending stops for more than 200 ms without a new observed `event_marker`, the partial event is treated as complete, parsed, and delivered. This timing logic is non-configurable and has priority over the `event_marker` setting. Log file appenders should flush full events. |
| **enabled=yes\|no** | (Optional) A parameter to activate or deactivate the configuration section. The possible values are `yes` or `no`. The default value is `yes`. |
| **charset=**_char-encoding-type_ | (Optional) The character encoding of the log files that the agent monitors. Possible values are:<br><br>■  `UTF-8`<br><br>■  `UTF-16LE`<br><br>■  `UTF-16BE`<br><br>The default value is `UTF-8`. |

| Parameter | Description |
|---|---|
| `tags={"tag-name" : "tag-value", ...}` | (Optional) A parameter to add custom tags to the fields of collected events. Define tags using JSON notation. Tag names can contain letters, numbers, and underscores. A tag name can only begin with a letter or an underscore and cannot exceed 64 characters. Tag names are not case-sensitive. For example, if you use tags={"tag_name1" : "tag value 1", "Tag_Name1" : "tag value 2" }, Tag_Name1 is ignored as a duplicate. You cannot use event_type and timestamp as tag names. Any duplicates within the same declaration are ignored. <br><br> If the destination is a syslog server, tags can override the APP-NAME field. For example, tags={"appname":"VROPS"}. |
| `exclude_fields` | (Optional) A parameter to exclude individual fields from collection. You can provide multiple values as a semicolon- or comma-separated list. For example, <br><br> ■ `exclude_fields=hostname; filepath` <br><br> ■ `exclude_fields=type; size` <br><br> ■ `exclude_fields=type, size` |
| `raw_syslog=Yes\|No` | For agents that use the syslog protocol, this option allows the agent to collect and send raw syslog events. The default is No, which means collected events are transformed with user-specified syslog attributes. Activate this option to collect events without any syslog transformations. |

## Example: Configurations

```
[filelog|vCenterMain]
directory=C:\ProgramData\VMware\VMware VirtualCenter\Logs
include=vpxd-*.log
exclude=vpxd-alert-*.log;vpxd-profiler-*.log
event_marker=^\d{4}-\d{2}-\d{2}[A-Z]\d{2}:\d{2}:\d{2}\.\d{3}
```

```
[filelog|ApacheAccessLogs]
enabled=yes
directory=C:\Program Files (x86)\Apache Software Foundation\Apache2.2\logs
include=*.log
exclude=*_old.log
tags={"Provider" : "Apache"}
```

```
[filelog|MSSQL]
directory=C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\Log
charset=UTF-16LE
event_marker=^[^\s]
```

## Set up Windows Log File Channel Filtering

You can set up filters for Windows log files to explicitly include or exclude log events.

You use the `whitelist` and `blacklist` parameters to evaluate a filter expression. The filter expression is a Boolean expression that consists of event fields and operators.

**Note** The `blacklist` option only works for fields; it cannot be used to block text.

■ The `whitelist` parameter collects only log events for which the filter expression evaluates to non-zero. If you omit this parameter, the value is an implied 1.

■ The `blacklist` parameter excludes log events for which the filter expression evaluates to non-zero. The default value is 0.

For a complete list of Windows event fields and operators see Event Fields and Operators.

### Prerequisites

Log in to the Windows machine on which you installed the vRealize Log Insight Windows agent and start the Services manager to verify that the vRealize Log Insight agent service is installed.

### Procedure

1 Navigate to the program data directory of the vRealize Log Insight Windows agent.

   `%ProgramData%\VMware\Log Insight Agent`

2 Open the `liagent.ini` file in any text editor.

3 Add a `whitelist` or `blacklist` parameter in the `[filelog|]` section.

   For example:

   ```
   [filelog|apache]
   directory = path_to_log_directory
   include = glob_pattern
   blacklist = filter_expression
   ```

4 Create a filter expression from Windows events fields and operators.

   For example

   ```
   whitelist = myServer
   ```

5 Save and close the `liagent.ini` file.

### Example: Filter Configurations

You can configure the agent to collect only Apache logs where the server_name is

```
[filelog|apache]
directory=C:\Program Files\Apache Software Foundation\Apache2.4\logs
include=error.log
parser=clf
whitelist = server_name == "sample.com"
blacklist = remote_host == "127.0.0.1"
```

# Forward Logs to the Log Insight Windows Agent

You can forward logs from Windows machines to a machine where the Log Insight Windows Agent is running.

You can use Windows Log Forwarding to forward logs from multiple Windows machines to a machine on which the Log Insight Windows Agent is installed. You can then configure the Log Insight Windows Agent to collect all forwarded logs and send them to a vRealize Log Insight server.

Get familiar with Windows Log Forwarding. See http://technet.microsoft.com/en-us/library/cc748890.aspx and http://msdn.microsoft.com/en-us/library/windows/desktop/bb870973(v=vs.85).aspx.

**Prerequisites**

See Collect Logs from Windows Events Channels .

**Procedure**

1   Add a new section to the Log Insight Windows Agent configuration to collect logs from the Windows event channel that receives forwarded logs.

    The default channel name is ForwardedEvents.

2   Set up Windows Log Forwarding.

**What to do next**

Go to the vRealize Log Insight Web user interface and verify that forwarded logs are arriving.

# Configure the Log Insight Linux Agent

You can configure the Log Insight Linux Agent after you install it.

You can use the centralized Agent Configuration to set up the agent to send log events to a vRealize Log Insight server, specify the communication protocol and port, and configure flat file log collection. For the location of the `liagent.ini` file, see Define Log Details Level in the Log Insight Agents.

## Default Configuration of the vRealize Log Insight Linux Agent

After installation, the `liagent.ini` file contains preconfigured default settings for the Log Insight Linux Agent.

### vRealize Log Insight Linux Agent liagent.ini Default Configuration

If you use non-ASCII names and values, save the configuration as UTF-8.

If you are using central configuration, the final configuration is this file joined with settings from the server to form the liagent-effective.ini file.

You may find it more efficient to configure the settings from the server's agents page.

```
[server]
; Hostname or IP address of your Log Insight server / cluster load balancer. Default:
;hostname=LOGINSIGHT

;Enables or deactivates centralized configuration from the vRealize Log Insight server.
;When enabled, agent configuration changes made to the liagent.ini file on the server
;are joined with the settings in this file. to this agent. Accepted values are yes or no and
0 or 1.
;The default is yes.
;
;central_config=yes
;
;
; Protocol can be cfapi (Log Insight REST API), syslog. Default:
;proto=cfapi

; Log Insight server port to connect to. Default ports for protocols (all TCP):
; syslog: 514; syslog with ssl: 6514; cfapi: 9000; cfapi with ssl: 9543. Default:
;port=9543

; SSL usage. Default:
;ssl=yes
; Example of configuration with trusted CA:
;ssl=yes
;ssl_ca_path=/etc/pki/tls/certs/ca.pem

; Time in minutes to force reconnection to the server.
; This option mitigates imbalances caused by long-lived TCP connections. Default:
;reconnect=30

; FIPS mode.
; Possible values are 1 or 0. If omitted the default value is 1.
; ssl_fips_mode=1

[logging]
; Logging verbosity: 0 (no debug messages), 1 (essentials), 2 (verbose with more impact on
performance).
; This option should always be 0 under normal operating conditions. Default:
;debug_level=0

[storage]
; Max local storage usage limit (data + logs) in MBs. Valid range: 100-2000 MB.
;max_disk_buffer=2000

; Uncomment the appropriate section to collect system logs
; The recommended way is to enable the Linux content pack from LI server
;[filelog|syslog]
;directory=/var/log
;include=messages;messages.?;syslog;syslog.?
```

| Parameter | Default Value | Description |
|-----------|---------------|-------------|
| hostname | LOGINSIGHT | IP address or host name of the vRealize Log Insight virtual appliance. The default is **loginsight**. |
| central_config | yes | Activate or deactivate centralized configuration for this agent. When centralized configuration is deactivated, the agent ignores configuration provided by the vRealize Log Insight server. Accepted values are yes, no,1, or 0. The default value is yes. |
| proto | cfapi | Protocol that the agent uses to send log events to the vRealize Log Insight server. The possible values are cfapi and syslog. The default is cfapi. |
| port | 9543, 9000, 6514, and 514 | Communication port that the agent uses to send events to the vRealize Log Insight server. The default values are 9543 for cfapi with SSL enabled, 9000 for cfapi with SSL deactivated, 6514 for syslog with SSL enabled and 514 for syslog with SSL deactivated. |
| ssl | yes | Enables or deactivates SSL. The default value is yes. When ssl is set to yes, if you do not set a value for the port, the port is automatically picked up as 9543. |
| max_disk_buffer | 200 | The maximum disk space in MB that the Log Insight Linux Agent uses to buffer events and its own logs. When the specified max_disk_buffer is reached, the agent begins to drop new incoming events. |
| debug_level | 0 | Defines the log details level. See Define Log Details Level in the Log Insight Agents. |
| ssl_fips_mode | 1 | Enables or deactivates FIPS mode for the Log Insight Linux Agent through the liagent.ini file. Accepted values are 1 and 0. |

## Collect Log Events from a Log File

You can configure the vRealize Log Insight Linux agent to collect log events from one or more log files.

By default the vRealize Log Insight Linux agent collects hidden files created by applications or editors. The hidden filenames start with a period. You can prevent the vRealize Log Insight Linux agent from collecting hidden files by adding an exclude parameter, **exclude=.\***.

Field names are restricted. The following names are reserved and cannot be used as field names.

- `event_type`

- `hostname`

- `source`

- `text`

You can specify up to three destinations for agent information and filter the information before it is sent. See Forwarding Logs from a vRealize Log Insight Agent

---

**Note**   Monitoring a large number of files, such as a thousand or more, leads to a higher resource utilization by vRealize Log Insight Agent and impacts the overall performance of the host machine. To prevent this, configure the agent to monitor only the necessary files using patterns and globs, or archive the old log files. If monitoring a large number of files is a requirement, consider increasing the host parameters such as CPU and RAM.

---

Prerequisites

- Log in as **root** or use `sudo` to run console commands.

- Verify that the vRealize Log Insight Linux agent is installed and running. Log in to the Linux machine on which you installed the vRealize Log Insight Linux agent, open a console, and run `pgrep liagent`.

Procedure

**1**   Open the `/var/lib/loginsight-agent/liagent.ini` file in any text editor.

**2** Locate the [server|<dest_id>] section of the file. Add configuration parameters and set the values for your environment.

```
[filelog|section_name]
directory=path_to_log_directory
include=glob_pattern
...
```

| Parameter | Description |
|---|---|
| **[filelog|***section_name***]** | A unique name for the configuration section. |
| **directory=***full-path-to-log-file* | The full path to the log file directory. Glob patterns are supported. Example configurations:<br><br>■ To collect from all sub-directories of `D:\Logs\new_test_logs`directory, use `directory=D:\Logs\new_test_logs\*`<br>■ If your sub-directories have their own sub-directories, use the following configuration to monitor all sub-directories `directory=D:\Logs\new_test_logs\*\*`<br><br>**Note** To limit the number of files and directories and avoid high resource consumption, you cannot define a directory glob for either the first or second level directories such as: `directory=c:/tmp/*` or `directory=c:\Logs\*`. The directory path must be at least two levels.<br><br>You can define a path to a non-existing directory, and the agent will collect the log files in that directory once the directory and files are created.<br><br>You can define the same directory under one or more different configuration sections, to collect logs from the same file multiple times. This process makes it possible to apply different tags and filters to the same source of events.<br><br>**Note** If you use identical configurations for these sections, duplicated events are observed on the server side. |

| Parameter | Description |
|---|---|
| **include=**_file_name; ..._ | (Optional) The name of a filename or a file mask (glob pattern) from which to collect data. You can provide values as a semicolon separated list. The default value is `*`, which means that all files are included. The parameter is case-sensitive. |
| | A file mask (glob pattern) can be used to group files that follow the same naming convention, as well as within a single filename. For example, filenames that include spaces, such as `vRealize Ops Analytics.log` and `vRealize Ops Collector.log`, can be specified with `vRealize?Ops?Analytics*.log` or `vRealize*.log`. By using file masks, you can specify filenames that are acceptable for agent configuration under Linux and Windows hosts. |
| | By default `.zip` and `.gz` files are excluded from collection. |
| | **Important** If you are collecting a rotated log file, use the `include` and `exclude` parameters to specify a glob pattern that matches both the primary and the rotated file. If the glob pattern matches only the primary log file, the vRealize Log Insight agents might miss events during rotation. The vRealize Log Insight agents automatically determine the correct order of rotated files and sends events to thevRealize Log Insight server in the right order. For example, if your primary log file is named `myapp.log` and rotated logs are `myapp.log.1` and `myapp.log.2` and so on, you can use the following `include` pattern: |
| | `include= myapp.log;myapp.log.*` |
| **exclude=**_regular_expression_ | (Optional) A filename or file mask (glob pattern) to exclude from collection. You can provide values as a semicolon separated list. The default value is empty, which means that no file is excluded. |
| **event_marker=**_regular_expression_ | (Optional) A regular expression that denotes the start of an event in the log file. If omitted defaults to newline. The expressions you type must use the Perl regular expressions syntax. |
| | **Note** Symbols, for example quotation marks (`"  "`), are not treated as wrappers for regular expressions. They are treated as part of the pattern. |
| | Since the vRealize Log Insight agent is optimized for real-time collection, partial log messages written with an internal delay may be split into multiple events. If log file appending stops for more than 200 ms without a new observed `event_marker`, the partial event is treated as complete, parsed, and delivered. This timing logic is non-configurable and has priority over the `event_marker` setting. Log file appenders should flush full events. |
| **enabled=yes\|no** | (Optional) A parameter to activate or deactivate the configuration section. The possible values are `yes` or `no`. The default value is `yes`. |
| **charset=**_char-encoding-type_ | (Optional) The character encoding of the log files that the agent monitors. Possible values are: |
| | ■  `UTF-8` |
| | ■  `UTF-16LE` |
| | ■  `UTF-16BE` |
| | The default value is `UTF-8`. |

| Parameter | Description |
|---|---|
| **tags=***{"tag-name" : "tag-value", ...}* | (Optional) A parameter to add custom tags to the fields of collected events. Define tags using JSON notation. Tag names can contain letters, numbers, and underscores. A tag name can only begin with a letter or an underscore and cannot exceed 64 characters. Tag names are not case-sensitive. For example, if you use tags={"tag_name1" : "tag value 1", "Tag_Name1" : "tag value 2" }, Tag_Name1 is ignored as a duplicate. You cannot use event_type and timestamp as tag names. Any duplicates within the same declaration are ignored.<br><br>If the destination is a syslog server, tags can override the APP-NAME field. For example, tags={"appname":"VROPS"}. |
| **exclude_fields** | (Optional) A parameter to exclude individual fields from collection. You can provide multiple values as a semicolon- or comma-separated list. For example,<br><br>■ `exclude_fields=hostname; filepath`<br><br>■ `exclude_fields=type; size`<br><br>■ `exclude_fields=type, size` |
| **raw_syslog=Yes|No** | For agents that use the syslog protocol, this option allows the agent to collect and send raw syslog events. The default is No, which means collected events are transformed with user-specified syslog attributes. Activate this option to collect events without any syslog transformations. |

**3** Save and close the `liagent.ini` file.

## Example: Configurations

```
[filelog|messages]
directory=/var/log
include=messages;messages.?

[filelog|syslog]
directory=/var/log
include=syslog;syslog.?

[filelog|Apache]
directory=/var/log/apache2
include=*
```

## Filter Log Events

You can filter all collected log events on the vRealize Log Insight Linux agent based on their field values to specify which log events to pick or drop. You can use the `whitelist` and `blacklist` collector options to define filters.

**Tip** By default thevRealize Log Insight Linux agent collects hidden files created by programs or editors. The hidden file names start with a period. You can prevent the vRealize Log Insight Linux agent from collecting hidden files, by adding an exclude **exclude=.*** parameter.

For each log event, the collector evaluates the `whitelist` and `blacklist` filter expression. If the `whitelist` expression evaluates to true and the `blacklist` expression evaluates to false or cannot be evaluated, the event moves to the queue for further processing. In any other case, the collector drops the event. The default value of the `whitelist` expression is true and the default value of the `blacklist` expression is false.

**Tip**  The `Filelog` collector provides fewer fields for filtering. To obtain fields for filtering, you can parse the logs. For more information, see Parsing Logs.

A `whitelist` or `blacklist` filter is a set of variables, literals, and operators that evaluates to a single logical or integer value. You use the log event fields as variables and double quoted strings and numbers as literals. For information about the operators that you can use within a filter expression, see Event Fields and Operators.

**Note**

- If you compare a number with a string or if the comparison involves numerical strings, each string is converted to a number and the comparison is performed numerically. For example:

  - The expression `whitelist = 123.0 == "000123"` evaluates to true.

  - The expression `whitelist = "00987" == "987.00"` evaluates to true.

  - In the expression `whitelist = response_size >= "12.12"`, if the `response_size` field has a numeric value, the expression is evaluated numerically. If the response size is greater than 12.12, the expression is true, else it is false.

  - In the expression `whitelist = "09123" < "234"`, both the string literals are converted to numeric values and the expression evaluates to false.

- If one of the string operands cannot be converted to numeric values, both the operands are converted to string. A simple case-sensitive lexicographical comparison is performed. For example:

  - The expression `whitelist = "1234a" == "1234A"` is a string comparison that evaluates to false.

  - The expression `whitelist = 4 < "four"` converts 4 to "4" and evaluates to true.

  - In the expression `whitelist = response_size > "thousand"`, the value of the `response_size` field is converted to a string value, which evaluates the expression to false.

- If a filter expression evaluates to an integer value, it is treated as false if it is 0 and true otherwise.

  For example, the expression `whitelist = some_integer & 1` evaluates to true if the `some_integer` field has a least significant bit set and false otherwise.

For a complete list of log event fields and operators see Collect Log Events from a Log File.

In this example, you collect Apache access logs from the file `/var/log/httpd/access`. Some sample logs from the file are:

- `127.0.0.1 - frank [10/Oct/2016:13:55:36 +0400] "GET /apache_pb.gif HTTP/1.0" 200 2326`

- `198.51.100.56 - john [10/Oct/2016:14:15:31 +0400] "GET /some.gif HTTP/1.0" 200 8270`

- `198.51.100.12 - smith [10/Oct/2016:14:15:31 +0400] "GET /another.gif HTTP/1.0" 303 348`

- `198.51.100.32 - test [10/Oct/2016:15:22:55 +0400] "GET /experimental_page.gif HTTP/1.0" 400 46374`

- `127.0.0.1 - test [10/Oct/2016:15:22:57 +0400] "GET /experimental_page2.gif HTTP/1.0" 301 100`

**Prerequisites**

- Log in as **root** or use `sudo` to run console commands.

- Log in to the Linux machine on which you installed the vRealize Log Insight Linux agent, open a console and run `pgrep liagent` to verify that the vRealize Log Insight Linux agent is installed and running.

**Procedure**

1 Define a parser for the logs, as shown in the following snippet:

```
[parser|apache-access]
base_parser=clf
format=%h %l %u %t \"%r\" %s %b
```

The parser that you have defined extracts the `remote_host`, `remote_log_name`, `remote_auth_user`, `timestamp`, `request`, `status_code`, and `response_size` fields for every log event collected from the file `/var/log/httpd/access`. You can use these fields to filter events.

2 Open the `/var/lib/loginsight-agent/liagent.ini` file in any text editor.

3 Define a `Filelog` section in the file to collect and parse logs, as shown in the following snippet:

```
[filelog|apache-access]
directory = /var/log/httpd/
include = access
parser = apache-access
```

**4** Filter log events according to your requirement.

- To collect logs where the HTTP status is 200, you can define a `whitelist` in the `Filelog` section as shown in the following snippet:

```
[filelog|apache-access]
directory = /var/log/httpd/
include = access
parser = apache-access
whitelist = status_code == 200
```

The `whitelist` expression evaluates to true only for the first and second log events from the sample logs and the collector picks these events.

If the `status_code` field does not exist in the log event because it is not present in the log or is not parsed, the `whitelist` expression cannot be evaluated, which means it evaluates to false and collector drops the event.

- To drop a log event that you are not interested in, you can use the `blacklist` option. For example, if you are not interested in local traffic, you can block the local IP as shown in the following snippet:

```
[filelog|apache-access]
directory = /var/log/httpd/
include = access
parser = apache-access
blacklist = remote_host == "127.0.0.1"
```

The collector picks the second, third, and fourth log events from the sample logs.

- To filter log events based on more than one predicate, you can use `or` and `and` operators. For example, you can drop events generated from a local IP or events generated by test users from any host that you do not require, as shown in the following snippet:

```
[filelog|apache-access]
directory = /var/log/httpd/
include = access
parser = apache-access
blacklist = remote_host == "127.0.0.1" or remote_auth_user == "test"
```

Using the `or` operator evaluates the `blacklist` expression to true to skip an unwanted log event. The expression instructs the collector to drop the event if the `remote_host` field value is "127.0.0.1" or the `remote_auth_user` field value is "test".

The collector picks the second and the third log events from the sample logs.

- To drop log events generated from a local IP by test users, you can use `and` in the `blacklist` expression, as shown in the following snippet:

```
[filelog|apache-access]
directory = /var/log/httpd/
include = access
parser = apache-access
blacklist = remote_host == "127.0.0.1" and remote_auth_user == "test"
```

The collector drops the fifth log event from the sample logs.

- You can use `whitelist` and `blacklist` filters together. For example, if you require log events where the response size is greater than 1024 bytes but you do not require events that originated from a local host, you can use the following snippet:

```
[filelog|apache-access]
directory = /var/log/httpd/
include = access
parser = apache-access
whitelist = response_size > 1024
blacklist = remote_host == "127.0.0.1" or remote_host == "localhost"
```

The collector picks the second log event from the sample logs.

## Collecting Events from `journald`

Beginning with vRealize Log Insight 4.6, agents can read logs from the `journald` system service for log data in Linux distributions running `systemd`. `journald` is now the default standard for logging in `systemd`-based Linux platforms. The `journald` configuration section supports the following options:

**`journal_files`**

The journal files to monitor. The following values are supported:

| Value | Description |
| --- | --- |
| all | Open and monitor all the available journal files. |
| local | Monitor and read only the journal files generated on the local machine. |
| runtime | Monitor and read only the volatile journal files, excluding the files in the persistent storage. |
| system | Monitor and read only the system services and kernel journal files. |
| user | Monitor and read only journal files of the current user. |

**`fetch_fields`**

The fields to fetch with the message from the journal log entries. The value for this option is a case-insensitive list of field names separated by comma. The following values are supported:

| Value | Description |
|---|---|
| pri_severity,pri_facility,syslog_identifier | Default value for this option. |
| * | Fetch all the fields. |
| all | Do not fetch any fields. |

# Filtering Log Events from vRealize Log Insight Agents

You can provide the information that an agent sends to a destination with the filter option in the `[server|<dest_id>]` section of your local `liagent.ini` file.

The option is of the following form:

```
filter = {collector_type; collector_filter; event_filter}
```

| Filter type | Description |
|---|---|
| collector_ type | A comma-separated list that defines the collector types. Supported values are filelog or winlog. If no value is provided, all collector types are used. |
| collector_ filter | Specifies the name of a collector section in a regex format. For example, *vcops_.* refers to all collector sections that begin with "vcops_". |
| event_filt er | Filters for log event fields use the same syntax as an acceptlist or a denylist in collector sections. An agent sends only log events that evaluate the expression to True or a non-zero value. An empty event_filter always evaluates to True. To use event_filter on log events, you must have a parser defined in appropriate collector sections for field extraction. If an expression cannot be evaluated due to absence of fields in the collected log event, then the event is dropped. |

More than one filter expression can be specified by separating them with a comma as shown in the following example:

```
filter=
{winlog;Micr.*;},{filelog;apache-access;level=="error"}
```

If a message meets more than one set of filter criteria for a destination target, it is sent only once.

Table 4-1. Syntax Examples

| Filter | Meaning |
|---|---|
| filter= {winlog;Microsoft.*;} | Sends log events from winlog collectors only if the event name begins with "Microsoft". |
| filter= {winlog;Microsoft.*; eventid == 1023} | Sends log events from winlog collectors only if the event name begins with "Microsoft" and Event ID equal to 1023. |
| filter= {;.*;} | Default filter value. Sends all log events from all sources. |
| filter= {winlog;.*;} | Sends all log events from winlog sections. |

Table 4-1. Syntax Examples (continued)

| Filter | Meaning |
|---|---|
| filter= {filelog;syslog;facility<5} | Sends log events from [filelog\|syslog] section if facility less than 5. [filelog\|syslog] sections must have a parser that extracts the facility field. Otherwise, all events are skipped. |
| filter= {;;} | Matches no log events. Use this syntax to deactivate log forwarding. |

The following example adds a filter to the configuration of the second destination of the previous example.

```
; The second destination receives just syslog events through the plain syslog protocol.
[server|syslog-audit]
hostname=third_party_audit_management.eng.vmware.com
proto=syslog
ssl=no
filter= {filelog; syslog; }
```

The next example uses a more complex filter expression.

```
; This destination receives vRealize Operations events if they have the level field equal
;to "error" or "warning" and they are collected by sections whose name begins with "vrops-"

[server|licf-prod1]
hostname=vrops-errors.licf.vmware.com
filter= {; vrops-.*; level == "error" || level == "warning"}
```

More than one filter expression can be specified by separating them with a comma as shown in the following example.

```
filter= e.
{winlog;Micr.*;},{filelog;apache-access;level=="error"}
```

# Forwarding Logs from a vRealize Log Insight Agent

You can forward logs collected by an agent to up to three destinations. A destination can include vRealize Log Insight servers or forwarder, or third-party log management solutions.

For example, you might want to send audit or system logs to a server for your security team, application logs to a dev ops team server, and metrics logs to an IT management system. You use filters to specify which information goes to a destination. You can forward logs from a single vRealize Log Insight agent to up to three destinations.

Agent configuration is done through the `[server|<dest_id>]` section of your local `liagent.ini` file. Use the cfapi protocol with vRealize Log Insight servers or forwarders and syslog with other targets or destinations.

When you specify more than one destination for an agent, the first destination uses the default `loginsight` location. You must specify location information for other destinations.

The next example shows a portion of an liagent.ini file that specifies two destinations. The default server name `loginsight` is implicit for the first destination by default and is not specified. The second `[server|<dest_id>]` section specifies a destination.

```
; The first (default) destination receives all collected events.
[server]
ssl=yes

; The second destination receives just syslog events through the plain syslog protocol.
[server|syslog-audit]
hostname=third_party_audit_management.eng.vmware.com
proto=syslog
ssl=no
```

For information about creating filters for agents, see Filtering Log Events from vRealize Log Insight Agents.

## Set Target vRealize Log Insight Server

You can set or change the target vRealize Log Insight server for a vRealize Log Insight agent running on Windows. You can send log events to up to three destinations and filter output per destination.

The default destination can be configured through the `[server]` section of the `liagent.ini` file. The default destination is always present and by default the hostname is set to `loginsight`. To add more target destinations, create a `[server|<dest_id>]` section for each target. You must specify a unique hostname as the destination ID for each additional connection. You can use the same options for additional destinations as for the default `[server]` section. Do not configure additional destinations for auto-upgrade or use them for agent configuration. You can specify two additional destinations.

By default, the agent sends all collected logs to all destinations. You can filter logs to send different logs to different destinations with the `file` option. For more information, see Filtering Log Events from vRealize Log Insight Agents.

Prerequisites

- Log in to the Windows machine on which you installed the vRealize Log Insight Windows agent and start the Services manager to verify that the vRealize Log Insightagent service is installed.

- If you have a vRealize Log Insight cluster with an enabled Integrated Load Balancer, see Enable Integrated Load Balancer for custom SSL certificate-specific requirements.

Procedure

1  Navigate to the program data directory of the vRealize Log Insight Windows agent.

   `%ProgramData%\VMware\Log Insight Agent`

2  Open the `liagent.ini` file in any text editor.

3  Modify the following parameters and set the values for your environment.

| Parameter | Description |
| --- | --- |
| **proto** | Protocol that the agent uses to send log events to the vRealize Log Insight server. The possible values are `cfapi` and `syslog`.<br><br>The default is cfapi. |
| **hostname** | IP address or host name of the vRealize Log Insight virtual appliance.<br><br>You can specify an IPv4 or IPv6 address. An IPv6 address can be specified with or without square brackets. For example:<br><br>```<br>hostname = 2001:cdba::3257:9652<br>or<br>hostname = [2001:cdba::3257:9652]<br>```<br><br>If the host supports both IPv4 and IPv6 stacks and a domain name is specified as the hostname, then the agent chooses the IP stack based on the IP address that the name resolver returns. If the resolver returns both IPv4 and IPv6 addresses, then the agent tries to connect sequentially to both addresses in the given order. |
| **max_disk_buffer** | The maximum disk space in MB that the Log Insight Windows Agent can use to buffer log events collected for this particular server. The option overrides the `[storage].max_disk_buffer` value for this server.<br><br>The default value is 150 MB and you can set the buffer size to between 50 through 8000 MB. |
| **port** | Communication port that the agent uses to send log events to the vRealize Log Insight or third party server. By default the agent uses the appropriate port based on the options that are set for SSL and the protocol. See default port values provided in the list below. You need to specify the port option only if it's different from these defaults.<br><br>■ cfapi with SSL activated: 9543<br>■ cfapi with SSL deactivated: 9000<br>■ syslog with SSL activated: 6514<br>■ syslog with SSL deactivated: 514 |
| **ssl** | Enables or disables SSL. The default value is yes.<br><br>When `ssl` is set to yes, the port is set as 9543, unless you specify otherwise. |

| Parameter | Description |
|-----------|-------------|
| `reconnect` | The time in minutes to force re-connection to the server. The default value is 30. |
| `filter` | Specifies the information an agent sends to a destination. This option takes three arguments:<br><br>`{collector_type; collector_filter; event_filter}` |

```
[server]
hostname=LOGINSIGHT
; Hostname or IP address of your Log Insight server / cluster load balancer. Default:
;hostname=LOGINSIGHT

; Protocol can be cfapi (Log Insight REST API), syslog. Default:
;proto=cfapi

; Log Insight server port to connect to. Default ports for protocols (all TCP):
; syslog: 514; syslog with ssl: 6514; cfapi: 9000; cfapi with ssl: 9543. Default:
;port=9543

; SSL usage. Default:
;ssl=yes
```

**4**  Save and close the `liagent.ini` file.

**Example**

The following configuration example sets a target vRealize Log Insight server that uses a trusted certificate authority.

```
[server]
proto=cfapi
hostname=LOGINSIGHT
port=9543
ssl=yes;
ssl_ca_path=/etc/pki/tls/certs/ca.pem
```

The following example shows a multi-destination configuration that includes filtering messages per destination.

```
; The first (default) destination receives all collected events.
[server]
hostname=prod1.licf.vmware.com

; The second destination receives just syslog events through the plain syslog protocol.
[server|syslog-audit]
hostname=third_party_audit_management.eng.vmware.com
proto=syslog
ssl=no
filter={filelog; syslog; }
```

```
; The third destination receives vRealize Operations events if they have the level field
equal to "error" or "warning"
; and they are collected by sections whose name begins with "vrops-"

[server|licf-prod1]
hostname=vrops-errors.licf.vmware.com
filter={; vrops-.*; level == "error" || level == "warning"}

; Collecting syslog messages.
[filelog|syslog]
directory=/var/log
include=messages

; various vROPs logs. Note that all section names begin with a "vrops-" prefix, which is used
in third destination filter.
[filelog|vrops-ANALYTICS-analytics]
directory=/data/vcops/log
include=analytics*.log*
exclude=analytics*-gc.log*
parser=auto

[filelog|vrops-COLLECTOR-collector]
directory=/data/vcops/log
include=collector.log*
event_marker=^\d{4}-\d{2}-\d{2}[\s]\d{2}:\d{2}:\d{2}\,\d{3}
parser=auto

[filelog|vrops-COLLECTOR-collector_wrapper]
directory=/data/vcops/log
include=collector-wrapper.log*
event_marker=^\d{4}-\d{2}-\d{2} [\s]\d{2}:\d{2}:\d{2}\.\d{3}
parser=auto
```

**What to do next**

You can configure additional SSL options for the vRealize Log Insight agent. See Configure SSL Connection Between the Server and the Log Insight Agents.

## Specify an Agent's Target

You can specify up to three destinations for the vRealize Log Insight Linux agent to send log events to.

Multiple destination connections are defined through the `[server|<dest_id>]` section of the `li-agent.ini` file where *<dest_id>* is a unique per-configuration connection id. You can use the same options for additional destinations as for the default `[server]` section. However, do not configure additional destinations for auto-upgrade or use them for agent configuration. You can specify two additional destinations.

The first target you define can use the default server value `loginsight`. When you define additional targets, you must specify a hostname in the `[server]` sections for subsequent targets. Without filtering, the agent sends all collected logs to all destinations. This is the default. However, you can filter logs to send different logs to different destinations.

**Prerequisites**

- Log in as **root** or use `sudo` to run console commands.

- Log in to the Linux machine on which you installed the vRealize Log Insight Linux agent, open a console and run `pgrep liagent` to verify that the vRealize Log Insight Linux agent is installed and running.

- If you have a vRealize Log Insight cluster with an activated Integrated Load Balancer, see Activate Integrated Load Balancer for custom SSL certificate-specific requirements.

**Procedure**

1    Open the `/var/lib/loginsight-agent/liagent.ini` file in any text editor.

2    Modify the following parameters and set the values for your environment.

| Parameter | Description |
| --- | --- |
| `proto` | Protocol that the agent uses to send log events to the vRealize Log Insight server. The possible values are `cfapi` and `syslog`.<br>The default is cfapi. |
| `hostname` | IP address or host name of the vRealize Log Insight virtual appliance.<br>You can specify an IPv4 or IPv6 address. An IPv6 address can be specified with or without square brackets. For example:<br><br>```<br>hostname = 2001:cdba::3257:9652<br>or<br>hostname = [2001:cdba::3257:9652]<br>```<br><br>If the host supports both IPv4 and IPv6 stacks and a domain name is specified as the hostname, then the agent uses the IP stack depending on the IP address that is returned by the name resolver. If the resolver returns both IPv4 and IPv6 addresses, then the agent tries to connect sequentially to both addresses in the given order. |
| `max_disk_buffer` | The maximum disk space in MB that the Log Insight Linux Agent can use to buffer log events collected for this particular server. The option overrides the `[storage].max_disk_buffer` value for this server.<br>The default value is 150 MB and you can set the buffer size to between 50 through 8000 MB. |

| Parameter | Description |
|---|---|
| **port** | Communication port that the agent uses to send log events to the vRealize Log Insight or third party server. By default the agent uses the appropriate port based on the options that are set for SSL and the protocol. See default port values provided in the list below. You need to specify the port option only if it's different from these defaults.<br><br>■ cfapi with SSL activated: 9543<br>■ cfapi with SSL deactivated: 9000<br>■ syslog with SSL activated: 6514<br>■ syslog with SSL deactivated: 514 |
| **ssl** | Activates or deactivates SSL. The default value is yes.<br><br>When `ssl` is set to yes, if you do not set a value for the port, the port is automatically picked up as 9543. |
| **reconnect** | The time in minutes to force reconnection to the server. The default value is 30. |

```
[server]
hostname=LOGINSIGHT
; Hostname or IP address of your Log Insight server / cluster load balancer. Default:
;hostname=LOGINSIGHT

; Protocol can be cfapi (Log Insight REST API), syslog. Default:
;proto=cfapi

; Log Insight server port to connect to. Default ports for protocols (all TCP):
; syslog: 514; syslog with ssl: 6514; cfapi: 9000; cfapi with ssl: 9543. Default:
;port=9543

; SSL usage. Default:
;ssl=yes
```

3  Save and close the `liagent.ini` file.

### Example

The following configuration example sets a target vRealize Log Insight server that uses a trusted certificate authority.

```
[server]
proto=cfapi
hostname=LOGINSIGHT
port=9543
ssl=yes;
ssl_ca_path=/etc/pki/tls/certs/ca.pem
```

The following example shows a multi-destination configuration.

■ The first (default) destination receives all collected log events.

```
[server]
hostname=prod1.licf.vmware.com
```

- The second destination receives just syslog events through the plain syslog protocol.

```
[server|syslog-audit]
hostname=third_party_audit_management.eng.vmware.com
proto=syslog
ssl=no
filter= {filelog; syslog; }
```

- The third destination receives vRealize Operations log events if they have the level field equal to "error" or "warning" and they are collected by sections whose name begins with "vrops-"

```
[server|licf-prod1]
hostname=vrops-errors.licf.vmware.com
filter= {; vrops-.*; level == "error" || level == "warning"}

;Collecting syslog messages.
[filelog|syslog]
directory=/var/log
include=messages

;various vRops logs. Note that all section names begin with "vrops-" prefix, which is used in
third destination filter.
[filelog|vrops-ANALYTICS-analytics]
directory=/data/vcops/log
include=analytics*.log*
exclude=analytics*-gc.log*
parser=auto
[filelog|vrops-COLLECTOR-collector]
directory=/data/vcops/log
include=collector.log*
event_marker=^\d
{4}-\d{2}-\d{2}[\s]\d{2}:\d{2}:\d{2}\,\d{3}
parser=auto

[filelog|vrops-COLLECTOR-collector_wrapper]
directory=/data/vcops/log
include=collector-wrapper.log*
event_marker=^\d{4}
-\d
{2}-\d{2}
[\s]\d
{2}:\d{2}
:\d
{2}
\.\d
{3}
parser=auto
```

## What to do next

You can configure additional SSL options for the vRealize Log Insight Linux agent. See Configure SSL Connection Between the Server and the Log Insight Agents.

# Centralized Configuration of vRealize Log Insight Agents

You can configure multiple vRealize Log Insight agents.

Each vRealize Log Insight agent has a local configuration and a server-side configuration. The local configuration is stored in the `liagent.ini` file on the virtual or physical machine where the vRealize Log Insight agent is installed. The server-side configuration is accessible and editable, for example, from **Administration > Agents** in the web user interface. The configuration of each vRealize Log Insight agent is made up of sections and keys. Keys have configurable values.

vRealize Log Insight agents periodically poll the vRealize Log Insight server and receive the server-side configuration. The server-side configuration and the local configuration are merged and the result is the effective configuration. Each vRealize Log Insight agent uses the effective configuration as its operating configuration. Configurations merge section by section and key by key. The values in the server-side configuration override the values in the local configuration. The merging rules are the following:

- If a section is present only in the local configuration or only in the server-side configuration, this section and all its content become a part of the effective configuration.

- If a section is present in both the local and server-side configuration, the keys in the section are merged according to the following rules:

  - If a key is present only in the local configuration or only in the server-side configuration, the key and its value become a part of this section in the effective configuration.

  - If a key is present in both the local configuration and the server-side configuration, the key becomes a part of this section in the effective configuration, and the value in the server-side configuration is used.

As a vRealize Log Insight administrator or a user with edit access for agents, you can apply a centralized configuration to all vRealize Log Insight agents. For example, you can navigate to the **Administration** page, and in the **Management** section, click **Agents**. Enter the configuration settings in the **Agent Configuration** box and click **Save Configuration for All Agents**. The configuration is applied to all the configurable active agents during the next poll cycle.

As an administrator, you can also use specific filters in agent groups, such as by OS, agent version, hostname, or IP ranges, and apply the configuration to specific vRealize Log Insight agents. For information about agent groups, see *Working with Agent Groups*.

---

**Note**

- You can apply a centralized configuration only to vRealize Log Insight agents that use the cfapi protocol.

- A vRealize Log Insight agent is not configurable in either of the following scenarios:

  - The current vRealize Log Insight server is not a primary destination. For information about configuring multiple destinations, see Specify an Agent's Target .

  - The parameter `central_config = no` is used in the agent configuration. For information about the default agent configuration for Windows, see Default Configuration of the Log Insight Windows Agent.

---

## An Example of Configuration Merging

An example of merging local and server-side configuration of the Log Insight Windows Agent.

### Local Configuration

You can have the following local configuration of the Log Insight Windows Agent.

```
[server]
proto=cfapi
hostname=HOST
port=9000

[winlog|Application]
channel=Application

[winlog|Security]
channel=Security

[winlog|System]
channel=System

[filelog|ApacheAccessLogs]
enabled=yes
directory=C:\Program Files (x86)\Apache Software Foundation\Apache2.2\logs
include=*.log
exclude=*_old.log
event_marker=^(\d{1,3}\.){3}\d{1,3} - -
```

## Server-Side Configuration

You can use the **Administration > Agents** page of the Web user interface to apply centralized configuration to all agents. For example, you can exclude and add collection channels, and change the default reconnect setting.

```
[server]
reconnect=20

[winlog|Security]
channel=Security
enabled=no

[winlog|Microsoft-Windows-DeviceSetupManagerOperational]
channel=Microsoft-Windows-DeviceSetupManager/Operational
```

## Effective Configuration

The effective configuration is a result of the merging of the local and the server-side configurations. The Log Insight Windows Agent is configured to :

- reconnect to the vRealize Log Insight server every 20 minutes

- continue to collect Application and System event channels

- stop collecting Security event channel

- start to collect Microsoft-Windows-DeviceSetupManager/Operational event channel

- continue to collect ApacheAccessLogs

```
[server]
proto=cfapi
hostname=HOST
port=9000
reconnect=20

[winlog|Application]
channel=Application

[winlog|Security]
channel=Security
enabled=no

[winlog|System]
channel=System

[winlog|Microsoft-Windows-DeviceSetupManagerOperational]
channel=Microsoft-Windows-DeviceSetupManager/Operational

[filelog|ApacheAccessLogs]
enabled=yes
directory=C:\Program Files (x86)\Apache Software Foundation\Apache2.2\logs
include=*.log
```

```
exclude=*_old.log
event_marker=^(\d{1,3}\.){3}\d{1,3} - -
```

# Using Common Values for Agent Configuration

You can override the default values of the agent configuration file with common parameter values that apply for each agent configuration section for Windows or Linux agents.

## Common Options

Options specified in the `[common|global]` section of the `liagent.ini` configuration file are propagated to all sections, options specified in the `[common|filelog]` section are propagated to all and only filelog sections, and `[common|winlog]` options are propagated to all and only winlog sections.

You can define the following parameters in common sections: `tags`, `include`, `exclude`, `event_marker`, `charset`, `exclude_fields`, and `parser` as shown in the following example. The example is for a Windows agent:

```
[common|global]

tags = {"log_source_vm":"win-2008r2-64"}
exclude_fields = test_tag;some_other_tag
parser = auto

[common|filelog]
tags = {"collector_type":"filelog"}
exclude = *.trc

[filelog|channel_1]
directory = C:\app\log
include = *.log

...
```

This example specifies the following behavior:

- All logs from filelog sections have both `log_source_vm` and `collector_type` tags with their corresponding values.

- `test_tag` and `some_other_tag` tags are excluded from all logs sent.

- `auto` parser is applied to all collected logs.

- By default, all filelog collectors exclude *.trc files from monitoring.

Options in `[common|global]` are also applied to all winlog sections.

## Merge and Override Criteria

If options are defined in more than one section, their values are merged or overridden and the section with a smaller scope has a higher priority when merging/overriding. That is, a value from `[common|global]` is merged with or overridden by a value from `[common|filelog]`] which in turn is combined with or overridden by a value from `[filelog|sample_section]`.

Merge and override behavior conform to the following rules:

- Options whose values represent a list of values (tags, include, exclude and exclude_fields ) are merged with values of that option from a section with a higher priority. And in case of tags, values of tags from sections with a higher priority override the value of that same tag from a section with a lower priority, as described previously.

- The value of options that can have single value (event_marker, charset and parser) are overridden by values of that option from sections with higher priority.

  This means that the value of charset=UTF-8 from [filelog|sample_section] overrides the global value of charset= UTF-16LE from [common|global].

So, for example, if you have `tags={"app":"global-test"}` in `[common|filelog]` and `tags={"app":"local-test","section":"flg_test_section"}` in `[filelog|flg_test_section]`, the value of the "app" tag from the `[filelog|flg_test_section]` section overrides the value from `[common|filelog]`. All logs collected through this filelog section will have an additional "app" tag with "local-test" value and "section" tag with "flg_test_section" value. For winlog sections, the chain of priority is the same, with any `[winlog|...]` section having the highest priority and `[common|global]` having the lowest priority.

When invalid values are specified in common sections, generally they are skipped and not merged with values from prior and corresponding filelog/winlog sections. In the case of invalid values in tags or `exclude_fields` options, the agent extracts as much valid data as possible and skips the rest of the file once invalid data is encountered. All anomalies are reported in the agent log file. Consult the log file if unexpected behavior is encountered and fix all errors reported by the agent.

If the agent detects an invalid value for an option in a filelog or winlog section, then it does not merge option values from that section with option values from common sections and does not enable that section. All errors are reported in an agent log file. Consult the log file if unexpected behavior is encountered and fix all reported errors by agent.

## Parsing Logs

Agent-side log parsers extract structured data from raw logs before delivering to the vRealize Log Insight server. Using log parsers, vRealize Log Insight can analyze logs, extract information from them, and show those results on the server. Log parsers can be configured for both Windows and Linux vRealize Log Insight Agents.

If the syslog protocol is used, fields extracted by parsers are part of STRUCTURED-DATA according to RFC5424.

# Configure Log Parsers

You can configure parsers for both `FileLog` and `WinLog` collectors.

**Prerequisites**

For the vRealize Log Insight Linux Agent:

■ Log in as root or use `sudo` to run console commands.

■ Log in to the Linux machine on which you installed the Log Insight Linux Agent, open a console and run `pgrep liagent` to verify that the Log Insight Linux Agent is installed and running.

For the vRealize Log Insight Windows Agent:

■ Log in to the Windows machine on which you installed the Log Insight Windows Agent and start the Services manager to verify that the vRealize Log Insightt service is installed.

**Procedure**

1 Navigate to the folder containing the `liagent.ini` file.

| Operating System | Path |
| --- | --- |
| Linux | `/var/lib/loginsight-agent/` |
| Windows | `%ProgramData%\VMware\Log Insight Agent` |

2 Open the `liagent.ini` file in any text editor.

3 To configure a specific parser, define a parser section. `[parser|myparser]`

Where `myparser` is an arbitrary name of the parser which can be referred from log sources. Parser section should refer to any built in (or any other defined) parser and configure that parser's mandatory options and non-required options if needed.

For example, `base_parser=csv` shows that `myparser` parser is derived from built-in parser `csv`. It expects that input logs consist of two fields which are separated with a semicolon.

```
[parser|myparser]

base_parser=csv

fields=field_name1,field_name2

delimiter=";"
```

4 After defining `myparser`, refer to it from log sources `winlog` or `filelog`.

```
[filelog|some_csv_logs]

directory=D:\Logs
```

```
include=*.txt;*.txt.*

parser=myparser
```

The logs collected from `some_csv_logs` sources, for example from the `D:\Logs` directory, are parsed by `myparser` and extracted events appear on the server as `field_name1` and `field_name2` respectively.

**Note**  The static logs in the `D:\Logs` directory are not get pulled into vRealize Log Insight by the agent. However, new files that are created in the `D:\Logs` directory are available in vRealize Log Insight.

5   Save and close the `liagent.ini` file.

## Common Options for Parsers

You can configure common options for all parsers that produce named fields.

### Reserved Words for Field Names

Field names are restricted. The following names are reserved and cannot be used as field names.

- `event_type`

- `hostname`

- `source`

- `text`

### Common Parser Options

Options in the following table can be used with all supported parsers.

| Option | Description |
|---|---|
| base_parser | The name of the base parser that this custom parser extends. It can be a built-in parser name or another customer parser name. This configuration key is mandatory. |
| field_decoder | Nested parsers specified as a JSON strings. Keys are the names of the field to apply the nested parser to and the value is the name of the parser to use for that field. Each nested parser is applied to the appropriate field decoded by the base parser. Field decoders are useful when the value of a field is a complex value, for example, a timestamp. The **field_decoder** option also supports more comlpex JSON objects as arguments that allow you to use conditions for specific field values that are checked before the nested parser is applied. |
| | **Note** For more information on usage and conditional configurations, see Conditional Configurations for the field_decoder option section below. |
| field_rename | Renames extracted fields. Use a JSON string where keys are the original names of the fields and values are the new desired names of the fields. The field_decoder option is always applied before field_rename. The order of these options in the INI file is not important. For clarity, specify field_decoder first. |
| next_parser | Name of the next parser to run. Allows multiple parsers to run sequentially on the same input. |
| | **Note** Parsers process all consequent parsers defined by the next_parser keyword and may replace a field value already extracted by a previous parser. |
| exclude_fields | A list of semicolon separated field names to remove from the event before it is delivered to the server. Field names are removed before event filtering is performed so that the field that you excluding during parsing cannot be used in the filter condition. |
| debug | Yes or No option that enables debugging of a particular parser. With debugging enabled, the parser performs detailed logging of input it receives, the operation it performed and the result it produced. The option applies per-section, that is, only to the parser defined by the particular section.<br>The default value for debug is debug=no for parsers. |

## Conditional configurations for the **field_decoder** option

For logs with the same common format but signficant differences related to specific field values, logs with **info** and **error**severities for example, you can use the conditional nested parser to reduce the application of unnecessary parsers to the corresponding fields of already parsed logs.

For example, using these logs:

```
2019-03-29T11:00:54.858Z
        host-FQDN Hostd: error hostd[2099230]
        [Originator@6876 sub=Default opID=1983bdbe-c1-800f user=admin.user] AdapterServer
caught
        exception: SSLExceptionE(SSL Exception: error:140000DB:SSL routines:SSL
routines:short read:
        The connection was closed by the remote end during handshake.)
```

```
2019-03-29T11:00:55.477Z
        host-FQDN Hostd: info hostd[6D620B70]
      ['commonhost' opID=5759adcc-cf] [transportConnector] -- FINISH task-internal-5726666
--  -- Completed connection restart --
```

You can use the following configuration to parse them:

```
[parser|clf_parser]
base_parser=clf
format=%t %{generator_host}i %i: %{log_severity}i %i[%{thread_id}i]%M
field_decoder={"log_message" : {"log_severity" : {"error" : "error_parser", "info" :
"info_parser"}}}
exclude_fields=log_message

[parser|info_parser]
base_parser=clf
format=[%{common_info}i] [%{process}i] %M
field_rename={"log_message" : "info_log_content"}

[parser|error_parser]
base_parser=clf
format=[%{common_info}i] %{exception_handler}i %i:%{exception_type}i:%i:%
{error_id}i:%i:%i: %M
field_rename={"log_message" : "exception_content"}
```

This configuration produces the following results:

```
timestamp=2019-03-29T11:00:54.858000
generator_host="host-FQDN"
log_severity="error"
thread_id="2099230"
common_info=Originator@6876 sub=Default opID=1983bdbe-c1-800f user=admin.user
exception_handler="AdapterServer"
exception_type="SSLExceptionE(SSL Exception"
error_id="140000DB"
exception_content="The connection was closed by the remote end during handshake.)"
```

Additioanlly the following fields are parsed for the **info** log:

```
timestamp=2019-03-29T11:00:55.477000
generator_host="host-FQDN"
log_severity="info"
thread_id="6D620B70"
```

```
log_message="['commonhost' opID=5759adcc-cf] [transportConnector] -- FINISH task-
internal-5726666 --  -- Completed connection restart --"
common_info="'commonhost' opID=5759adcc-cf"
process="transportConnector"
info_log_content="-- FINISH task-internal-5726666 --  -- Completed connection restart --"
```

## Comma-Separated Value Log Parsers

You can configure Comma-Separated Value (CSV) parsers for both `FileLog` and `WinLog` collectors.

The available options for the `csv` parser are `fields` and `delimiter`.

### Comma-Separated Value Parser Options

Note the following information about the structure of the `csv` parser.

| Option | Description |
| --- | --- |
| fields | The `fields` option specifies the names of the fields that exist in the log. The total number of the listed field names must be equal to the total number of comma-separated fields in the logs. |
| | The `fields` option is mandatory for the CSV parser. If it is not specified, nothing is parsed. Double quotes surrounding the field value are optional, depending on the field content. |
| | Field names must be separated by commas, for example |
| | `fields = field_name1, field_name2, field_name3, field_name4` |
| | This definition assumes that the names `field_name1`, `field_name2`, `field_name3` and `field_name4` are assigned sequentially to the extracted fields. |
| | If some fields must be omitted by the CSV parser, their names can be omitted from the list. For example, |
| | `fields = field_name1, , field_name3, field_name4` |
| | In this case, the parser extracts only the first, third and fourth fields from the event and subsequently assigns the names `field_name1`, `field_name3` and `field_name4` to them. |
| | If the fields option does not specify a complete list of the fields in your logs, the parser returns an empty list. For example, if the log file contains `field1`, `field2`, `field3`, `field4`, and `field5`, but only `fields= field1,field2,field3` is specified, the parser returns an empty fields list. |
| | You cannot use `fields=*` for a CSV parser, because the parser returns an empty fields list. A complete list of fields must be specified, unless you need certain fields omitted as already described. |
| delimiter | The `delimiter` option specifies the delimiter for the parser to use. By default, the `csv` parser uses a comma as a delimiter; however, you can change the delimiter to a semicolon, a space, or other special character. The defined delimiter must be enclosed in double quotes. |
| | For example, `delimiter=","` and `delimiter=";"`. |
| | The `csv` parser supports any set of characters as delimiters that are enclosed in quotes, for example "||" or "asd". The field values' separators in the logs should exactly match the pattern defined by the delimiter parameter, otherwise the parser will fail. |
| | Special characters such as a space or a tab can be defined for as a delimiter for the `csv` parser as long as the escape character precedes the special character for (\", \s, \t). For example, `delimiter="\s"` or `delimiter=" "`. |
| | The `delimiter` option is optional. |

### CSV Log Parser Configuration

To parse logs collected from either `winlog` or `filelog` sources, use the following configuration.

```
[filelog|some_csv_logs]
directory=D:\Logs
include=*.txt;*.txt.*
parser=myparser

[parser|myparser]
base_parser = csv
fields = timestamp,field_name1, field_name2, field_name3
delimiter = ";"
field_decoder={"timestamp": "tsp_parser"}
[parser|tsp_parser]
; timestamp is a built-in parser
base_parser=timestamp
; "format" is an option of timestamp parser
format=%Y-%m-%d %H:%M:%S
```

With this configuration, logs collected from `some_csv_logs` source (for example, from the `directory=D:\Logs` directory) are parsed by `myparser`. If the collected logs contain three values that are separated by a semicolon, the parsed events sequentially receive the `field_name1`, `field_name2` and `field_name3` names.

To parse the following CSV log:

```
"United States","USA","North America","High income: OECD","Fiscal year end: September 30;
reporting period for national accounts data: CY."
```

Define the CSV parser configuration:

```
[parser|csv_log_parser]
base_parser=csv
fields=country_name, country_code, region, income_group, special_notes
```

The CSV parser returns the following fields:

```
country_name=United States
country_code=USA
region=North America
income_group=High income: OECD
special_notes=Fiscal year end: September 30; reporting period for national accounts data: CY.
```

## Common Log Format (Apache) Log Parser

You can configure the Common Log Format (CLF) Apache parser for both `FileLog` and `WinLog` collectors.

## Common Log Format (Apache) Parser

The default CLF parser defines the following order and names of fields.

```
host ident authuser datetime request statuscode bytes
```

Parser name: `clf`

The CLF parser-specific option is `format`.

## format Option

The `format` option specifies the format with which Apache logs are generated. The option is not mandatory.

If no format is specified, the following default common log format is used.

```
%h %l %u %t \"%r\" %s %b
```

The CLF parser format string does not accept regex expressions. For example, specify a space instead of the expression \s+.

To parse other log formats, specify that format in the agent's configuration. Parsed fields appear on the server side with the following names.

**Note**  In the cases in which a variable is required, if `{VARNAME}` is not provided in the configuration, the fields are ignored.

| Fields | Value |
| --- | --- |
| `'%a':` | "remote_ip" |
| `'%A':` | "local_ip" |
| `'%B', '%b':` | "response_size" |
| `'%C':` | Depends on the name of variable specified in the format |
| `'%c':` | Depends on the name of variable specified in the format |
| `'%D':` | "request_time_mcs" |
| `'%E':` | "error_status" |
| `'%e':` | Depends on the name of variable specified in the format |
| `'%F', '%f':` | "file_name" |
| `'%h':` | "remote_host" |
| `'%H':` | "request_protocol" |
| `'%i':` | Depends on the name of variable specified in the format |
| `'%k':` | "keepalive_request_count" |

| Fields | Value |
|---|---|
| `'%l':` | "remote_log_name" |
| `'%L'` | "request_log_id" |
| `'%M':` | "log_message" (parser stops parsing the input log after reaching this specifier) |
| `'%m':` | "request_method" |
| `'%n':` | depends on the name of variable specified in the format |
| `'%o':` | depends on the name of variable specified in the format |
| `'%p':` | "server_port"<br><br>Additional formats can be used with this specifier: %{format}p. Supported formats are "canonical", "local", or "remote". When the "canonical" format is used, the field name remains as "server_port". When the "local" format is used, the field name will be "local_server_port", and when the "remote" format is used, the field name will be "remote_server_port". |
| `'%P':` | "process_id"<br><br>Additional formats can be used with this specifier: %{format}P. Supported formats are "pid", "tid", and "hextid". If "pid" is used as a format, the field name will be "process_id". While "tid" and "hextid" formats generate fields with the name "thread_id" |
| `'%q':` | "query_string" |
| `'%r':` | "request" |
| `'%R':` | "response_handler" |
| `'%s':` | "status_code", which generates the final status of the request. |

| Fields | Value |
| --- | --- |
| `'%t':` | "timestamp", which works as event timestamp on ingestion, and engages the timestamp parser. To override timestamp auto detection, date and time format can be specified in curly braces: %{%Y-%m-%d %H:%M:%S}t, see Timestamp Parser for more details. |
| | The timestamp format for the CLF parser can start with "begin:" or "end:" prefixes. If the format starts with `begin:` (default), the time is taken at the beginning of the request processing. If it starts with `end:`, it is the time when the log entry gets written, close to the end of the request processing. For example, such formats such as the following are supported by CLF parser: `%h %l %u [%{begin:%d/%b/%Y %T}t.%{msec_frac}t] \"%r\" %>s %b` |
| | The following format tokens are also supported for the CLF parer's timestamp format specifier: |
| | **sec** |
| | number of seconds since the Epoch. This is equivalent to Timestamp parser's `%s` specifier. |
| | **msec** |
| | number of milliseconds since the Epoch |
| | **usec** |
| | number of microseconds since the Epoch |
| | **msec_frac** |
| | millisecond fraction (is equivalent to Timestamp parser's %f specifier) |
| | **musec** |
| | microsecond fraction (is equivalent to Timestamp parser's %f specifier) |
| | To parse logs where timestamp is represented with format tokens, the following formats can be used in the configuration: |
| | ```
format=%h %l %u %{sec}t \"%r\" %s %b
format=%h %l %u %{msec}t \"%r\" %s %b
format=%h %l %u %{usec}t \"%r\" %s %b
``` |
| | These tokens cannot be combined with each other or Timestamp parser formatting in the same format string. You can use multiple %{format}t tokens instead. For example, to use Timestamp which includes milliseconds, except of using Timestamp parser's `%f` specifier, the following combined timestamp can be used: `%{%d/%b/%Y %T}t.%{msec_frac}t` . |
| `'%T':` | "request_time_sec" |
| `'%u':` | "remote_auth_user" |
| `'%U':` | "requested_url" |
| `'%v':` | "server_name" |
| `'%V':` | "self_referential_server_name" |
| `'%X':` | "connection_status", which depends on the name of variable specified in the format |
| `'%x':` | Depends on the name of variable specified in the format |

| Fields | Value |
|---|---|
| `'%I':` | "received_bytes" |
| `'%O':` | "sent_bytes" |
| `'%S':` | "transferred_size" |

For example, to parse logs collected from either `winlog` or `filelog` sources with the CLF parser, specify the following configuration:

```
[filelog|clflogs]
directory=D:\Logs
include=*.txt
parser=myclf

[parser|myclf]
debug=yes ;Note: use this option only while debugging and set it to 'no' when used in
production.
base_parser=clf
format=%h %l %u %b %t \"%r\" %s
```

Using this configuration, logs that are collected from the `clflogs` source, for example from the `directory=D:\Logs` directory, are parsed by `myclf`. The `myclf` parser only parses those logs that were generated with the format described in the configuration.

The default value for debug is `debug=no` for parsers.

### Parsing Logs that were Generated Using CLF

To parse logs that were generated using CLF, you must define the corresponding format in the configuration. For example,

```
format=%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User_Agent}i\"
```

Fields that are not empty that use the specifiers `%{Referer}i` and `%{User_Agent}i` appear on the vRealize Log Insight server with the names `referer` and `user_agent` respectively.

### Integrating the Timestamp Parser with the CLF Parser

You can parse Apache logs with a custom time format.

Access logs that have a custom time format as follows.

```
format = %h %l %u %{%a, %d %b %Y %H:%M:%S}t \"%r\" %>s %b
```

If a custom time is not specified, the CLF parser attempts to deduce the time format automatically by running the automatic timestamp parser, otherwise the custom time format is used.

The supported custom time formats that are supported for error logs are:

| Custom Time Format | Description | Configuration Format |
|---|---|---|
| `%{u}t` | Current time including micro-seconds | `format=[%{u}t] [%l] [pid %P] [client %a] %M` |
| `%{cu}t` | Current time in compact ISO 8601 format, including micro-seconds | `format=[%{cu}t] [%l] [pid %P] [client %a] %M` |

For a full list of supported timestamp specifiers, see Timestamp Parser.

**Example: Apache Default Access Logs Configuration for Windows**

**Example: Apache Default Error Logs Configuration for Windows**

This example shows how you can format Apache v2.4 access log configurations for Windows.

```
;ACCESS LOG
;127.0.0.1 - - [13/May/2015:14:44:05 +0400] "GET /xampp/navi.php HTTP/1.1" 200 4023 "http://
localhost/xampp/" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:37.0) Gecko/20100101 Firefox/37.0"
;format=%h %l %u %{%d/%b/%Y:%H:%M:%S %z}t \"%r\" %>s %b \"%{Referer}i\" \"%{User_agent}i\"

; Section to collect Apache ACCESS logs
[filelog|clflogs-access]
    directory=C:\xampp\apache\logs
    include=acc*
    parser=clfparser_apache_access
    enabled=yes

;Parser to parse Apache ACCESS logs
[parser|clfparser_apache_access]
    debug=yes
    base_parser=clf
    format=%h %l %u %{%d/%b/%Y:%H:%M:%S %z}t \"%r\" %>s %b \"%{Referer}i\" \"%{User_agent}i\"
```

Define the access log format:

1   Configure Apache for the access log format (httpd.conf):

```
 LogFormat "%h %l %u %{%d-%b-%Y:%H:%M:%S}t \"%r\" %a %A %e %k %l %L %m %n %T %v %V %>s %b
\"%{Referer}i\" \"%{User_Agent}i\"" combined
```

2   Define the CLF parser configuration:

```
;ACCESS LOG
;127.0.0.1 unknown - 21-May-2015:13:59:35 "GET /xampp/navi.php HTTP/1.1" 127.0.0.1 127.0.0.1
- 0 unknown - GET - 1 localhost localhost 200 4023 "http://localhost/xampp/" "-"
[filelog|clflogs-access]
    directory=C:\xampp\apache\logs
    include=acc*;_myAcc*
    parser=clfparser_apache_access
    enabled=yes
; Parser to parse Apache ACCESS logs
[parser|clfparser_apache_access]
   debug=yes
   base_parser=clf
```

```
   format=%h %l %u %{%d-%b-%Y:%H:%M:%S}t \"%r\" %a %A %e %k %l %L %m %n %T %v %V %>s %b \"%
{Referer}i\" \"%{User_Agent}i\"
```

The CLF parser returns the following:

```
remote_host=127.0.0.1
timestamp=2015-05-13T10:44:05
request=GET /xampp/navi.php HTTP/1.1
status_code=200
response_size=4023
referer=http://localhost/xampp/
user_agent=Mozilla/5.0 (Windows NT 6.1; WOW64; rv:37.0) Gecko/20100101 Firefox/37.0
```

This example shows how you can format Apache v2.4 error log configurations for Windows.

```
;ERROR LOG
;[Wed May 13 14:37:17.042371 2015] [mpm_winnt:notice] [pid 4488:tid 272] AH00354: Child:
Starting 150 worker threads.
;[Wed May 13 14:37:27.042371 2015] [mpm_winnt:notice] [pid 5288] AH00418: Parent: Created
child process 3480
;format=[%{%a %b %d %H:%M:%S%f %Y}t] [%m:%{severity}i] [pid %P:tid %{thread_id}i] %E: %M
;format=[%{%a %b %d %H:%M:%S%f %Y}t] [%m:%{severity}i] [pid %P] %E: %M

; Section to collect Apache ERROR logs
[filelog|clflogs-error]
    directory=C:\xampp\apache\logs
    include=err*
    parser=clfparser_apache_error
    enabled=yes

;Parser to parse Apache ERROR logs
[parser|clfparser_apache_error]
    debug=yes
    base_parser=clf
    format=[%{%a %b %d %H:%M:%S%f %Y}t] [%m:%{severity}i] [pid %P:tid %{thread_id}i] %E: %M
    next_parser=clfparser_apache_error2

;Parser to parse Apache ERROR logs
[parser|clfparser_apache_error2]
    debug=yes
```

```
base_parser=clf
format=[%{%a %b %d %H:%M:%S%f %Y}t] [%m:%{severity}i] [pid %P] %E: %M
```

**Note**  The provided names correspond to the combined log format. Apache error logs are also described using the above formatting keys, not the Apache error log format.

Define the error log format:

1   Configure Apache for the error log format (httpd.conf):

```
  LogFormat "%h %l %u %{%d-%b-%Y:%H:%M:%S}t \"%r\" %a %A %e %k %l %L %m %n %T %v %V %>s %b
  \"%{Referer}i\" \"%{User_Agent}i\"" combined
```

2   Define the CLF parser configuration:

```
;Parser to parse Apache ERROR logs
[parser|clfparser_apache_error]
    debug=yes
    base_parser=clf
    format=[%{%a %b %d %H:%M:%S%f %Y}t] [%m:%{severity}i] [pid %P] %E: %M
    next_parser=clfparser_apache_error2

;Parser to parse Apache ERROR logs
[parser|clfparser_apache_error2]
    debug=yes
    base_parser=clf
    format=[%{%a %b %d %H:%M:%S%f %Y}t] [%m:%{severity}i] [pid %P:tid %{thread_id}i] %E: %M
```

Log entry:

```
[Wed May 13 14:37:17.042371 2015] [mpm_winnt:notice] [pid 4488:tid 272] AH00354: Child:
Starting 150 worker threads.
```

The CLF parser returns the following fields for the log entry (If using a parser in a +0400 timezone):

```
timestamp=2015-05-13T10:37:17.042371
request_method=mpm_winnt
severity=notice
process_id=4488
thread_id=272
error_status=AH00354
log_message=Child: Starting 150 worker threads.
```

Log entry:

```
[Wed May 13 14:37:27.042371 2015] [mpm_winnt:notice] [pid 5288] AH00418: Parent: Created
child process 3480
```

The CLF parser returns the following fields for the log entry (If using a parser in a +0400 timezone):

```
timestamp=2015-05-13T10:37:27.042371
request_method=mpm_winnt
severity=notice
process_id=5288
error_status=AH00418
log_message=Parent: Created child process 3480
```

## Key/Value Pair Parser

You can configure the Key/Value Pair (KVP) parser for both `FileLog` and `WinLog` collectors.

### Key/Value Pair (KVP) Parser

The `kvp` parser finds and extracts all `key=value` matches from an arbitrary log message text. The following example shows the `kvp` parser format.

```
[parser|kvp_parser]
base_parser=kvp
fields=*
```

For example, the key-value log can be in the following format: `scope=local; abstract=false; lazyInit=false; autowireMode=0; dependencyCheck=0;`

With the `kvp` parser, you must specify the fields from which the values are to be extracted. For example, if the definition `fields=name,lastname,country` exists in the configuration, only the values with the specified keys are parsed and sent to the server.

Both the key and the value can be optionally surrounded by double quotes " " to define white space or other special characters.

When double quotes are used for the key or value, the backslash character " \ " can be used as the escape character. Any character following the backslash character is defined literally, including a double quote character or a backslash character. For example: " \\ "

Note the following considerations.

- If the key in a key/value pair is not followed by an equals sign and a `VALUE` is not provided, the option is skipped, as with free text.

- The key cannot be empty, the value can be empty.

- An equals sign that is not followed by a value is treated as free text and is skipped.

- A value can be a string of characters that are surrounded by double quote characters, or it can be empty. Use a backslash for escaping special characters that are part of the value.

### KVP Parser Options

Note the following information about the structure of the `kvp` parser.

| Option | Description |
|---|---|
| `fields` | The information that you want to extract described as units of data. For example, `fields=name,lastname,country`. |
| | If specific field names are defined by the `fields` option, then each invalid char in a field name extracted from a log is replaced with an underscore. For example, if the `fields` option looks for fields "x-A" and "a*(X+Y)", then the parser extracts these fields from logs and renames them to "x_a" and "a__x_y" fields respectively. This makes it possible to extract fields with any chars in the name. |
| | If the `fields` option is specified as "*", which means the `kvp` parser recognizes field/value pairs automatically, then the parser looks for fields that have only "alphanumeric+underscore" chars (supported by LI server). All other invalid chars are dropped instead of being converted to underscores. This prevents the parser from extracting unnecessary information into static fields. |
| `delimiter` | Optional. |
| | Default delimiters are the space character, tab, newline characters, comma, and semicolon characters. |
| | If delimiters are not specified in the configuration, the `kvp` parser uses default delimiters for parsing. |
| | To change the default delimiters to specific delimiters, you must define them between double quotes. For example: `delimiter = "#^|"`. This definition means that each of the characters that are enclosed in the double quotes is used as a delimiter. For the `kvp` parser, any character can be considered as delimiter. You can include the default delimiters with other delimiters in the definition. |
| | For example, the `delimiter = "#^|\t\r\n\s"` statement includes the tab, newline characters, and the space as delimiters. If these characters are used, they must be preceded by the escape character. For example, to define the space character as a delimiter, enter the escape character " \" before the space character when defining it as a delimiter, for example, `delimiter="\s"`. |
| `field_decoder` | Nested parsers are specified as a JSON string in which the keys are the names of the field to apply to the nested parser, and the value is the name of the parser to use for that field. |
| | Each nested parser is applied to the appropriate field, as decoded by the base parser. |
| | Field decoders are useful when the value of a key-value pair is a complex value such as a timestamp or a comma-separated list. |
| `debug =` | Optional. The `debug =` value can be `yes` or `no`. The default value for debug is `debug=no` for parsers. |
| | When the option is set to `yes`, you can view detailed logs of the parser ingestion in `liagent_<date>.log`. |

## Additional Keys Value Options

| Key | Definition |
|---|---|
| `KVP_MESSAGE = *(MESSAGE_ENTRY [WSPR])` | A list of message entries separated by optional white space |
| `MESSAGE_ENTRY = KVP / FREE_TEXT` | An entry is a key/value pair or just a free text |
| `KVP = KEY ["=" VALUE]` | Key/value pair. If KEY is not followed by an equal sign and VALUE, it is skipped like free text. |

| Key | Definition |
| --- | --- |
| `KEY = BARE_KEY / QUOTED_KEY` | |
| `FREE_TEXT = "="` | A free standing equal sign is considered a free text and is skipped. |
| `BARE_KEY = *1BARE_KEY_CHAR` | At least one character |
| `BARE_KEY_CHAR = %0x00-08 / %0x10-19 / %0x21-3C / %3E-%FF` | Any character excluding equal sign, space, or TAB |
| `QUOTED_KEY = 0x22 *1(QUOTED_STRING_CHAR / "\" CHAR) 0x22` | At least one character surrounded by double quote characters. The backslash is used as an escape character. |
| `QUOTED_STRING_CHAR = %0x00-21 / %0x23-FF` | Any character excluding double quote |
| `VALUE = BARE_VALUE / QUOTED_VALUE` | |
| `BARE_VALUE = *BARE_VALUE_CHAR` | Zero or more characters |
| `BARE_VALUE_CHAR = %0x00-08 / %0x10-19 / %0x21-FF` | Any character excluding space or TAB |
| `QUOTED_VALUE = 0x22 *(QUOTED_STRING_CHAR / "\" CHAR) 0x22` | A string of characters surrounded by double quote characters. This can be empty. The backslash is used as an escape character. |

## KVP Parser Configuration Examples

You can use `fields=*` to parse all fields, if necessary.

```
[parser|simple_kvp]
base_parser =kvp
fields=*
```

This example shows how to specify the field decoder.

```
[parser|mykvp]
debug=no
base_parser=kvp
delimiter="#^|"
fields=*
;OR fields=scope,abstract,lazyInit,autowireMode,dependencyCheck
field_decoder={"field1":"field1_parser1"}

[parser|field1_parser1]
base_parser=clf
format=[%{value1}i]]
field_decoder={"value1":"field_parser2"}
```

To parse the following KVP log:

```
Configuring transport... proto = cfapi server_hostname = LOCALHOST ssl = no port = 9000
reconnect = 30
```

Define the KVP parser configuration:

```
[parser|kvp_log_parser]
base_parser=kvp
fields=*
```

The KVP parser returns the following fields:

```
proto=cfapi
server_hostname=LOCALHOST
ssl=no
port=9000
reconnect=30
```

**Note**  In input logs, when an assignment operator (=) is preceded by a randomly generated key, it is a best practice not to use the KVP parser with the * option, because it generates many random fields, which might lead to UI and query performance issues. Instead, you can use the parser by specifying the fields that need to be parsed.

**Example: Simple and Complex KVP Parser Examples**

Simple KVP Parser Example

```
[filelog|MyLog]
directory=C:\<folder_name>\Parser_logs
include=*.log
parser=my_KVP_parser

[parser|my_KVP_parser]
base_parser=kvp
fields=*
```

Complex KVP Parser Example

```
[filelog|MyLog]
directory=C:\<folder_name>\Parser_logs
include=*.log
parser=my_KVP_parser

[parser|my_KVP_parser]
base_parser=kvp
fields=*
field_decoder={"field1":" field1_parser1"}

[parser| field1_parser1]
base_parser=clf
format=[%{value1}i]]
field_decoder={"value1":" field1_parser2"}
```

## Timestamp Parser

The `timestamp` parser does not produce fields but instead transforms its input from a string to an internal timestamp format displayed in milliseconds from the UNIX epoch start, January 1, 1970 (midnight UTC/GMT).

The only supported configuration option is `format`. For example, `format=%Y-%m-%d %H:%M:%S`.

Unlike the CLF parser, the `timestamp` parser can parse time when there are no delimiters between time specifiers, for example `%A%B%d%H%M%S%Y%z`.

Format specifiers that are used by the `timestamp` parser are:

```
'%a':    Abbreviated weekday name, for example: Thu
'%A':    Full weekday name, for example: Thursday
'%b':    Abbreviated month name, for example: Aug
'%B':    Full month name, for example: August
'%d':    Day of the month, for example: 23. strftime() expects zero-padded (01-31) digits
         for this specifier but Log Insight agents can parse space-padded and non-padded
         day numbers, too.
'%e':    Day of the month, for example: 13. strftime() expects space-padded ( 1-31) digits
         for this specifier but Log Insight agents can parse zero-padded and non-padded
         day numbers too.
'%f':    Fractional seconds of time, for example: .036 'f' specifier assumes that '.' or ','
         character should exist before fractional seconds and there is no need to mention
         that character in the format. If none of these characters precedes fractional
seconds,
         timestamp wouldn't be parsed.
'%H':    Hour in 24h format (00-23), for example: 14. Zero-padded, space-padded, and non-
padded hours
         are supported.
'%I':    Hour in 12h format (01-12), for example: 02. Zero-padded, space-padded and non-
padded hours
         are supported.
'%m':    Month as a decimal number (01-12), for example: 08. Zero-padded, space-padded
         and non-padded month numbers are supported.
'%M':    Minute (00-59), for example: 55
'%p':    AM or PM designation, for example: PM
'%S':    Second (00-61), for example: 02
'%s':    Total number of seconds from the UNIX epoch start, for example 1457940799
         (represents '2016-03-14T07:33:19' timestamp)
'%Y':    Year, for example: 2001
'%z':    ISO 8601 offset from UTC in timezone (1 minute=1, 1 hour=100)., for example: +100
```

Additional specifiers are accepted by the timestamp parser, but their values are ignored and do not affect the parsed time.

```
'%C':    Year divided by 100 and truncated to integer (00-99), for example: 20
'%g':    Week-based year, last two digits (00-99), for example, 01
'%G':    Week-based year, for example, 2001
'%j':    Day of the year (001-366), for example: 235
'%u':    ISO 8601 weekday as number with Monday as 1 (1-7), for example: 4
'%U':    Week number with the first Sunday as the first day of week one (00-53), for example:
33
```

```
'%V':    ISO 8601 week number (00-53), for example: 34
'%w':    Weekday as a decimal number with Sunday as 0 (0-6), for example: 4
'%W':    Week number with the first Monday as the first day of week one (00-53), for example:
34
'%y':    Year, last two digits (00-99), for example: 01
```

If a `format` parameter is not defined, the `Timestamp` parser parses the timestamps using the default formats.

### Automatic Timestamp Parser

The automatic timestamp parser is called when no format is defined for the timestamp parser or the parser can be called directly without a timestamp parser definition by using `timestamp` in the `field_decoder`. For example:

```
[parser|mycsv]
base_parser=csv
debug=yes
fields=timestamp,action,source_id,dest
field_decoder={"timestamp": "timestamp"}
```

### Example: A Timestamp Parser with the Default Configuration

This example shows a `timestamp` parser with a default configuration.

```
[parser|tsp_parser]
base_parser=timestamp
debug=no
format=%Y-%m-%d %H:%M:%S%f
```

To integrate a `timestamp` parser with other parsers, for example the CSV parser, specify the following configuration.

```
[parser|mycsv]
base_parser=csv
fields=timestamp,action,source_id,dest
field_decoder={"timestamp": "tsp_parser"}
```

When this configuration is defined, `mycsv` parser extracts the fields with the names that are specified in the configuration, and runs `tsp_parser` on the content of the `timestamp` field. If `tsp_parser` retrieves a valid timestamp, the server uses that timestamp for the log message.

## Automatic Log Parser

The automatic parser automatically detects the timestamp within the first 200 characters of a line. The format of auto-detected time stamps are the same as for the `timestamp` parser.

The automatic parser does not have any options. In addition to the automatic detection of the timestamp, the Key/Value parser runs on the log entry and automatically detects any existing key/value pairs in the logs and extracts the fields accordingly. For example,

```
[filelog|some_logs]
directory=/var/log
include=*
parser=auto
```

As with other parsers, you can define a separate action for the automatic parser.

```
[filelog|kvplogs]
directory=C:\temp_logs\csv-itbm
include=*.txt
parser=myauto
[parser|myauto]

base_parser=auto
debug=yes
```

If you have `debug` enabled for the automatic parser, additional information about parsing is printed. For example, information about on which log the automatic parser was run, and which fields ere extracted from the log.

The default value for debug is `debug=no` for parsers.

---

**Note**  In input logs, when an assignment operator (=) is preceded by a randomly generated key, it is a best practice not to use the automatic parser, because it generates many random fields, which might lead to UI and query performance issues. Instead, you can use the KVP parser by specifying the fields that need to be parsed.

---

## Syslog Parser

The syslog parser supports the message_decoder and extract_sd options and automatically detects two formats: RFC-6587, RFC-5424, and RFC-3164.

### Configuring the message_decoder Option

All common options and the `message_decoder` option are available for the syslog parser. By default, only the `timestamp` and `appname` fields are extracted. Enable the message_decoder option by setting configuration values in your liagent.ini file to be similar to the following example:

```
[filelog|data_logs]
directory=D:\Logs
include=*.txt
parser=mysyslog

[parser|mysyslog]
base_parser=syslog
message_decoder=syslog_message_decoder
debug=yes
```

```
[parser|syslog_message_decoder]
base_parser=kvp
fields=*
```

## Example: Parsing with the message_decoder Option

The following example shows a sample event and the fields that are added to the event by a syslog parser configured to use the message_decoder option:

- Sample event:

```
2015-09-09 13:38:31.619407 +0400 smith01 john: Fri Dec 5 08:58:26 2014 [pid 26123]
[jsmith.net] status_code=FAIL oper_
ation=LOGIN: Client "176.31.17.46"
```

- Returned by a syslog parser to which the message_decoder option is applied to run a KVP parser:

```
timestamp=2015-09-09T09:38:31.619407 appname=john status_code=FAIL operation=LOGIN:
```

## Configuring the extract_sd Option for Parsing Structured Data

To parse structured data, enable the extract_sd option by setting configuration values in your liagent.ini file to be similar to the following example:

```
[filelog|simple_logs]
directory=/var/log
include=*.txt
parser=syslog_parser

[parser|syslog_parser]
base_parser=syslog
extract_sd=yes
```

## Example: Parsing with the extract_sd Option

The following example shows a sample event and the fields that are added to the event by a syslog parser configured to use the extract_sd option:

- The sample event: `<165>1 2017-01-24T09:17:15.719Z localhost evntslog`
  `- ID47 [exampleSDID@32473 iut="3" eventSource="Application" eventID="1011"]`
  `[examplePriority@32473 class="high"] Found entity IPSet, display name dummy ip set`
  `1411`

- The following fields are added to the event by the syslog parser:

```
timestamp=2017-01-24T09:17:15.719000
pri_facility=20
pri_severity=5
procid="-"
msgid="ID47"
iut="3"
```

```
eventsource="Application"
eventid="1011"
class="high"
appname="evntslog"
```

## Fields Extracted By the Parser

The parser automatically extracts the following fields from an event:

| RFC Classification | pri_facility | pri_severity | timestamp | appname | procid | msgid |
|---|---|---|---|---|---|---|
| Non-RFC | | | X | X | | |
| RFC-3164 | X | X | X | X | | |
| RFC-5424 | X | X | X | X | X | X |

## Syslog Parser Options

The following table describes available syslog options.

| Option | Description |
|---|---|
| message_decoder | Defines an additional parser, which is used to parse the message body of an event. It can be a built-in parser, such as 'auto' or any custom-defined parser. |
| extract_sd | Parses structured data. <br> Only yes or no values are supported for the extract_sd option. The option is deactivated by default. When the extract_sd option is enabled, it simply extracts all key-value pairs from the structured data. |

## Example: Parsing for the RFC-5424 Standard

The following examples show two events parsed by a syslog instance configured shows the configuration used for the collector, a sample event, and the fields that the syslog parser adds to the event.

▪ Configuration:

```
[filelog|simple_logs]
directory=/var/log
include=*.txt
parser=syslog
```

▪ An event generated in the monitored file:

```
<165>1 2017-01-24T09:17:15.719Z router1 mgd 3046 UI_DBASE_LOGOUT_EVENT
[junos@2636.1.1.1.2.18 username=\"regress\"] User 'regress' exiting configuration
mode - Juniper format
```

■ Fields that are added to the event by the syslog parser:

```
The following fields will be added to the event by Syslog parser:
timestamp=2017-01-24T09:17:15.719000
pri_facility = 20
pri_severity = 5
procid = 3046
msgid = UI_DBASE_LOGOUT_EVENT
appname = mgd
```

### Example: Parsing for the RFC-3164 Standard

The following example shows the configuration used for the collector, a sample RFC-3164 event, and the fields that syslog adds to the event.

■ Configuration:

```
[filelog|simple_logs]
directory=/var/log
include=*.txt
parser=syslog
```

■ An RFC-3164 event generated in the monitored file:

```
<13>2017-01-24T09:17:15.719Z router1 mgd 3046 UI_DBASE_LOGOUT_EVENT User 'regress' exiting
configuration mode - Juniper format
```

■ Fields that are added to the event by the syslog parser:

```
timestamp=2017-01-24T09:17:15.719000
pri_facility=1
pri_severity=5
appname="mgd"
```

## Labeled Tab-separated Values Parser

The Labeled Tab-separated Values (LTSV) format is a variant of Tab-separated Values (TSV).

Each record in a LTSV file is represented as a single line. Each field is separated by `<TAB>` and has a label and a value. The label and the value are separated by `:`. With the LTSV format, you can parse each line by splitting the line with `<TAB>` (the same as the TSV format) and extend any fields with unique labels in no particular order. For more information about the LTSV definition and format, see `http://ltsv.org/`.

## Example: LTSV Parser Configuration

## Example: Sample LTSV Log

The LTSV parser does not require specific configuration options. To use the LTSV parser, specify the built-in `ltsv` parser name in the configuration.

```
[parser|myltsv]
base_parser=ltsv
```

An LTSV file must be a byte sequence that matches the LTSV production in the ABNF format.

```
ltsv = *(record NL) [record]
record = [field *(TAB field)]
field = label ":" field-value
label = 1*lbyte
field-value = *fbyte

TAB = %x09
NL = [%x0D] %x0A
lbyte = %x30-39 / %x41-5A / %x61-7A / "_" / "." / "-" ;; [0-9A-Za-z_.-]
fbyte = %x01-08 / %x0B / %x0C / %x0E-FF
```

```
host:127.0.0.1<TAB>ident:-<TAB>user:frank<TAB>time:[10/Oct/2000:13:55:36 -0700]<TAB>req:GET /
apache_pb.gif HTTP/1.0<TAB>status:200<TAB>size:2326<TAB>referer:http://www.example.com/
start.html<TAB>ua:Mozilla/4.08 [en] (Win98; I ;Nav)
```

With the sample LTSV configuration, the log's parsing should return the following fields:

```
host=127.0.0.1
ident=-
user=frank
time=[10/Oct/2000:13:55:36 -0700]
req=GET /apache_pb.gif HTTP/1.0
status=200
size=2326
referer=http://www.example.com/start.html
ua=Mozilla/4.08 [en] (Win98; I ;Nav)
```

## Debug Configuration

Additional debugging is also available for the LTSV parser. By default, LTSV debugging is disabled. To turn on LTSV debugging, enter `debug=yes`.

```
[parser|myltsv]
base_parser=ltsv
debug=yes
```

When debugging is turned on, the LTSV parser extracts values of all valid labels from the log. The LTSV parser requires that label names consist only of alpha-numeric characters, the underscore ('_"), dot ('.') and dash ('-') characters. If at least one invalid label name exists in the log, its parsing will fail. Even if the label name is valid, the agent will check the field name. If invalid names exist, the label name should be corrected to a valid field name.

### Configuring the LTSV Parser from the `filelog` Section

You can also configure the LTSV parser from the `filelog` section directly.

```
[filelog|simple_logs]
directory=/var/log
include=*
parser=ltsv
```

## Regex Parser

The `regex` parser enables the use of some regular expressions for collected data.

vRealize Log Insight agents use the C++ Boost library regex, which is in Perl syntax. The `regex` parser can be defined by specifying a regular expression pattern that contains named capture groups. For example: `(?<field_1>\d{4})[-](?<field_2>\d{4})[-](?<field_3>\d{4})[-](?<field_4>\d{4})`

The names specified in the groups (for example: `field_1`, `field_2`, `field_3`, and `field_4`) become names of the corresponding extracted fields. Names have the following requirements:

- Names specified in the regular expression pattern must be valid field names for vRealize Log Insight.

- The names can contain only alphanumeric characters and the underscore " _ " character.

- The name cannot start with a digital character.

If invalid names are provided, configuration fails.

### Regex Parser Options

The only required option for the `regex` parser is the `format` option.

The `debug` option can be used when additional debugging information is needed.

### Configuration

To create a `regex` parser, use `regex` as a `base_parser` and provide the `format` option.

### Example: Regex Configuration Examples

### Example: Parsing Apache Logs Example

The following example can be used to analyze `1234-5678-9123-4567`:

```
[parser|regex_parser]
base_parser=regex
format=(?<tag1>\d{4})[-](?<tag2>\d{4})[-](?<tag3>\d{4})[-](?<tag4>\d{4})
```

```
[filelog|some_info]
directory=D:\Logs
include=*.txt
parser=regex_parser
```

The results show:

```
tag1=1234
tag2=5678
tag3=9123
tag4=4567
```

To parse Apache logs with the `regex` parser, provide the specific `regex` format for Apache logs:

```
[parser|regex_parser]
base_parser=regex
format=(?<remote_host>.*) (?<remote_log_name>.*) (?<remote_auth_user>.*) \[(?
<log_timestamp>.*)\] "(?<request>.*)" (?<status_code>.*) (?<response_size>.*)
```

The results show:

```
127.0.0.1 - admin [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326
remote_host=127.0.0.1
remote_log_name=-
remote_auth_user=admin
log_timestamp=10/Oct/2000:13:55:36 -0700
request=GET /apache_pb.gif HTTP/1.0
status_code=200
response_size=2326
```

The following code shows another example of parsing Apache logs.

```
[parser|regex_parser]
base_parser=regex
format=(?<remote_host>.* (?<remote_log_name>.*)) (?<remote_auth_user>.*) \[(?
<log_timestamp>.*)\] "(?<request>.* (?<resource>.*) (?<protocol>.*))" (?<status_code>.*) (?
<response_size>.*)
127.0.0.1 unknown - [17/Nov/2015:15:17:54 +0400] \"GET /index.php HTTP/1.1\" 200 4868
remote_host=127.0.0.1 unknown
remote_log_name=unknown
remote_auth_user=-
log_timestamp=17/Nov/2015:15:17:54 +0400
request=GET /index.php HTTP/1.1
resource=/index.php
protocol=HTTP/1.1
status_code=200
response_size=4868
```

### Performance Considerations

The `regex` parser consumes more resources than other parsers, such as the `CLF` parser. If you can parse logs with other parsers, consider using those parsers instead of the `regex` parser to achieve better performance.

If a parser is not provided and you use the `regex` parser, define formats as clear as possible. The following example shows a configuration that provides better performance results. This example specifies fields that have digital values.

```
(?<remote_host>\d+.\d+.\d+.\d+) (?<remote_log_name>.*) (?<remote_auth_user>.*) \[(?
<log_timestamp>.*)\] "(?<request>.*)" (?<status_code>\d+) (?<response_size>\d+)
```

## JSON Parser

You can customize the JSON parser configuration to selectively parse the JSON log.

You can configure comma-separate value (CSV) parsers for both **FileLog** and **WinLog** collectors. Only valid JSON logs are parsed with the Log Insight agent JSON parser. Invalid JSON log parsers return empty results.

The default JSON parser configuration extracts all fields from the JSON log by the Log Insight agent. When the JSON log represents itself as a complex JSON object, which can contain JSON objects as well, the parser uses an underscore (_) characcter to concatenate names of nested and higher tiered JSON objects. This produces an informative field name for the corresponding elements. If the JSON log also contains an array, the member element names contain the array name followed by the element's index in the array.

The JSON parser also provides a specific option, known as **fields**.

### JSON Parser 'fields' Option

You can use the **fields** option to specify which fields are parsed in the configuration. The purpose of this option is to enable selective parsing of the JSON log.

**Note**  For selective parsing, you must specify the path to the desired JSON element. JSON objects from different tiers must be separated with a dot (.) character.

The following list provides example configurations that enable you to selective parse the JSON log as desired.

- To parse more than one element from the JSON log, the desired elements must be listed as parameters for the **fields** option and separated by commas. See example below:

```
{"operation" : {"timestamp" :
        "2018-11-22T15:28:58.094000", "thread_id" : "0x05673", "initiator" : "connector",
        "log_severity" : "info", "log_message" : "Requested connection to the server."},
        "operation_result" : "success"}
```

- To parse only the most inner JSON objects, such as **timestamp**, **log_severity**, and **log_message** see the example below. This example configuration produces the following field results: operation_timestamp ="2018-11-22T15:28:58.094000" and operation_log_severity ="info"

```
[parser|json_parser]
base_parser=json
fields=operation.timestamp,operation.log_severity, operation.log_message
```

- To parse the entire JSON object, include the path to the object followed by an asterisk (*) character.

```
{"product_name" : "LI Agent",
        "operation" : {"timestamp" : "2018-11-22T15:28:58.094000", "thread_id" :
        "0x05673", "initiator" : "connector", "log_severity" : "info", "log_message" :
        "Requested connection to the server."}, "operation_result" :
    "success"}
```

- To parse only the **operation** object, use the following configuration:

```
[parser|json_parser]
base_parser=json
fields=operation.*
```

- If the JSON log contains an array and you want to parse only specific elements of the array, use the array's element index in the configuration, as seen in this example configuration:

```
{
    "Records": [{
            "object":{
                "key": "/events/mykey",
                "size": 502,
                "eTag": "091820398091823",
                "sequencer": "1123123"
            }
        },
        {
            "object":{
                "key": "/events/user_key",
                "size": 128,
                "eTag": "09182039000001",
```

```
                    "sequencer": "1123231"
               }
          },
          {
             "object":{
                    "key": "/events/admin_key",
                    "size": 1024,
                    "eTag": "09182039547241",
                    "sequencer": "1123213"
               }
          }
     ]
}
```

- To only parse the **key** and **size** elements from the same log, use the following configuration to produce the following fields:

  records0_object_key="/events/mykey"

  records0_object_size=502

  records2_object_key="/events/admin_key"

  records2_object_size=1024

  ```
  [parser|json_parser]
  base_parser=json
  fields = Records0.object.key Records0.object.size, Records2.object.key,
  Records2.object.size
  ```

- To parse the **key** field for all array elements, use the following configuration:

  ```
  [parser|json_parser]
  base_parser=json
  fields=Records.#.object.key
  ```

- To parse all fields, use the fields option with an asterisk (*) character. This configuraiton is equivalent to the default JSON parser configuration.

  ```
  [parser|json_parser]
  base_parser=json
  fields=*
  ```

# Uninstalling vRealize Log Insight Agents

# 5

Should you need to uninstall a vRealize Log Insight agent, follow the instructions that are appropriate to the agent package that you installed.

This chapter includes the following topics:

- Uninstall the Log Insight Windows Agent
- Uninstall the Log Insight Linux Agent RPM package
- Uninstall the Log Insight Linux Agent DEB package
- Uninstall the Log Insight Linux Agent bin Package
- Manually Uninstall the Log Insight Linux Agent bin package

## Uninstall the Log Insight Windows Agent

You can uninstall the Log Insight Windows Agent from the Programs and Feature screen of the Windows Control Panel.

### Prerequisites

Log in to the Windows machine on which you installed the vRealize Log Insight Windows agent and start the Services manager to verify that the vRealize Log Insight agent service is installed.

### Procedure

1  Go to **Control Panel > Programs and Features**.

2  Select the VMware vRealize Log Insight Windows Agent and click **Uninstall**.

### Results

The uninstaller stops the VMware vRealize Log Insight Windows Agent service and removes its files from the system.

## Uninstall the Log Insight Linux Agent RPM package

You can uninstall the Log Insight Linux Agent RPM package.

**Prerequisites**

- Log in as **root** or use `sudo` to run console commands.

- Log in to the Linux machine on which you installed the Log Insight Linux Agent, open a terminal console and run `pgrep liagent` to verify that the VMware Log Insight Linux Agent is installed and running.

**Procedure**

- Run the following command replacing *VERSION* and *BUILD_NUMBER* with the version and build number of the installed agent.

  ```
  rpm -e VMware-Log-Insight-Agent-VERSION-BUILD_NUMBER
  ```

**Results**

The uninstaller stops the VMware Log Insight Linux Agent daemon and removes all its files except its own logs from the system.

# Uninstall the Log Insight Linux Agent DEB package

You can uninstall the Log Insight Linux Agent DEB package.

**Prerequisites**

- Log in as **root** or use `sudo` to run console commands.

- Log in to the Linux machine on which you installed the Log Insight Linux Agent, open a terminal console and run `pgrep liagent` to verify that the VMware Log Insight Linux Agent is installed and running.

**Procedure**

- Run the following command

  ```
  dpkg -P vmware-log-insight-agent
  ```

**Results**

The uninstaller stops the VMware Log Insight Linux Agent daemon and removes all its files except its own logs from the system.

# Uninstall the Log Insight Linux Agent bin Package

You can uninstall the Log Insight Linux Agent .bin package with a vRealize Log Insight script.

**Prerequisites**

- Log in as **root** or use `sudo` to run console commands.

- Log in to the Linux machine on which you installed the Log Insight Linux Agent, open a terminal console and run `pgrep liagent` to verify that the VMware vRealize Log Insight Linux Agent is installed and running.

**Procedure**

1   At the shell prompt, enter the following command to start the script.

```
loginsight-agent-uninstall
```

2   You can verify that the uninstall completed successfully by checking that the returned error code from the following command is 0.

echo $?

# Manually Uninstall the Log Insight Linux Agent bin package

You can manually uninstall the Log Insight Linux Agent .bin package if you choose not to use the uninstall script.

**Prerequisites**

Manually Uninstall the Log Insight Linux Agent bin package

- Log in as **root** or use `sudo` to run console commands.

- Log in to the Linux machine on which you installed the Log Insight Linux Agent, open a terminal console and run `pgrep liagent` to verify that the VMware vRealize Log Insight Linux Agent is installed and running.

**Procedure**

1   Stop the Log Insight Linux Agent daemon by running the following command

`sudo service liagentd stop` or `sudo /sbin/service liagentd stop` for older Linux distributions.

2   Manually remove the Log Insight Linux Agent files

- `/usr/lib/loginsight-agent` - Daemon binary and license files directory.

- `/usr/bin/loginsight-agent-support` - Used to generate the support bundle for the Log Insight Linux Agent.

- `/var/lib/loginsight-agent` - Configuration files and database storage directory.

- `/var/log/loginsight-agent` - Log directory for the Log Insight Linux Agent.

- `/var/run/liagent/liagent.pid` - Log Insight Linux Agent PID file. If it is not deleted automatically, remove the file manually.

- `/etc/init.d/liagentd` - Script directory for the Log Insight Linux Agent daemon.

- `/usr/lib/systemd/system/liagentd.service`

# Troubleshooting vRealize Log Insight Agents

# 6

Known troubleshooting information can help you diagnose and correct problems related to the operation of the vRealize Log Insight agents.

This chapter includes the following topics:

- Create a Support Bundle for the Log Insight Windows Agent
- Create a Support Bundle for the Log Insight Linux Agent
- Define Log Details Level in the Log Insight Agents
- Administration UI Does Not Show Log Insight Agents
- vRealize Log Insight Agents Do Not Send Logs
- Add an Outbound Exception Rule for the Log Insight Windows Agent
- Allow Outbound Connections from the Log Insight Windows Agent in a Windows Firewall
- Mass Deployment of the Log Insight Windows Agent is Not Successful
- Log Insight Agents Reject Self-Signed Certificates
- vRealize Log Insight Server Rejects the Connection for Non-Encrypted Traffic

## Create a Support Bundle for the Log Insight Windows Agent

If the Log Insight Windows Agent does not operate as expected because of a problem, you can send a copy of the log and configuration files to VMware Support Services.

### Procedure

1 Log in to the target machine where you installed the Log Insight Windows Agent.

2 Click the Windows **Start** button and then click **VMware > Log Insight Agent - Collect support Bundle**.

3 (Optional) If the shortcut is not available, navigate to the installation directory of the Log Insight Windows Agent and double-click `loginsight-agent-support.exe`.

   **Note** The default installation directory is `C:\Program Files (x86)\VMware\Log Insight Agent`

Results

The bundle is generated and saved as a `.zip` file in `My Documents`.

**What to do next**

Forward the support bundle to VMware Support Services as requested.

# Create a Support Bundle for the Log Insight Linux Agent

If the Log Insight Linux Agent does not operate as expected because of a problem, you can send a copy of the log and configuration files to VMware Support Services.

**Procedure**

1  Log in to the target machine where you installed the Log Insight Linux Agent.

2  Run the following command.

```
/usr/lib/loginsight-agent/bin/loginsight-agent-support
```

**Results**

The bundle is generated and saved as a `.zip` file in the current directory.

**What to do next**

Forward the support bundle to VMware Support Services as requested.

# Define Log Details Level in the Log Insight Agents

You can edit the configuration file of the vRealize Log Insight Agent to change the logging level.

**Prerequisites**

For the Log Insight Linux Agent:

- Log in as **root** or use `sudo` to run console commands.

- Log in to the Linux machine on which you installed the Log Insight Linux Agent, open a console and run `pgrep liagent` to verify that the VMware vRealize Log Insight Linux Agent is installed and running.

For the Log Insight Windows Agent:

- Log in to the Windows machine on which you installed the vRealize Log Insight Windows agent and start the Services manager to verify that the vRealize Log Insight agent service is installed.

**Procedure**

1   Navigate to the folder containing the `liagent.ini` file.

| Operating system | Path |
|---|---|
| **Linux** | `/var/lib/loginsight-agent/` |
| **Windows** | `%ProgramData%\VMware\Log Insight Agent` |

2   Open the `liagent.ini` file in any text editor.

3   Change the log debug level in the `[logging]` section of the `liagent.ini` file.

**Note**  The higher the debug level, the higher the impact it has on the vRealize Log Insight Agent. The default and recommended value is 0. Debug level 1 provides more information and is recommended for troubleshooting of most issues. Debug level 2 provides detailed information. Use levels 1 and 2 only when requested by VMware Support.

```
[logging]
; The level of debug messages to enable: 0..2
debug_level=1
```

4   Save and close the `liagent.ini` file.

**Results**

The log debug level is changed.

# Administration UI Does Not Show Log Insight Agents

Information about the Log Insight Agents instances does not appear on the Agents page of the Administration UI.

**Problem**

After you install the Log Insight Agents you do not see the Log Insight Agents in the Agents page of the Administration UI.

**Cause**

The most common causes are network connectivity problems or incorrect configuration of the Log Insight Agents in the `liagent.ini` file.

**Solution**

◆   Verify that the Windows or Linux system that the Log Insight Agents are installed on has connectivity to the vRealize Log Insight server.

◆   Verify that the Log Insight Agents use the cfapi protocol.

When using the syslog protocol the UI does not show Log Insight Windows Agents.

◆ View the contents of the Log Insight Agents log files located in the following directories .

- Windows - `%ProgramData%\VMware\Log Insight Agent\log`

- Linux - `/var/log/loginsight-agent/`

Look for log messages that contain the phrases `Config transport error: Couldn't resolve host name` and `Resolver failed. No such host is known.`

◆ Verify that the `liagent.ini` contains the correct configuration for the target vRealize Log Insight server. See Set Target vRealize Log Insight Server and Specify an Agent's Target .

# vRealize Log Insight Agents Do Not Send Logs

An incorrect configuration can prevent the vRealize Log Insight agents from forwarding logs to the vRealize Log Insight server. If a flat file collection channel is not configured correctly, you may see messages such as "Invalid settings were obtained for channel 'CHANNEL_NAME'. Channel 'CHANNEL_NAME' will stay dormant until properly configured."

### Problem

The vRealize Log Insight agents instances appears on the **Administration > Agent** page but no logs appear in Interactive Analytics page from the vRealize Log Insight agents host names. The flat file collection channel is not configured correctly.

### Cause

Incorrect configuration can prevent the vRealize Log Insight agents from forwarding logs to the vRealize Log Insight server.

### Solution

◆ Define a valid collection channel. Verify whether or not the flat file collection channel is configured correctly. See Chapter 4 Configuring vRealize Log Insight Agents.

◆ For the vRealize Log Insight Windows agent, try the following.

- If Windows channels are enabled, view the contents of the vRealize Log Insight Windows agent log files located at `%ProgramData%\VMware\Log Insight Agent\log`. Look for log messages related to channel configuration that contain the phrases `Subscribed to channel CHANNEL_NAME`. Typically-used channels are `Application`, `System`, and `Security`.

- If a channel is not configured correctly, you might see log messages similar to `Could not subscribe to channel CHANNEL_NAME events. Error Code: 15007. The specified channel could not be found. Check channel configuration.` You might see an error code number other than 15007.

- If a flat file collection channel is not configured correctly, you might see messages like `Invalid settings were obtained for channel 'CHANNEL_NAME'. Channel 'CHANNEL_NAME' will stay dormant until properly configured`

◆ For both vRealize Log Insight Windows agent and vRealize Log Insight Linux agent, try the following.

◆ If no flat file collection channel is configured, you might see messages similar to
```
Cannot find section 'filelog' in the configuration. The flat file log
collector will stay dormant until properly configured
```

The contents of the vRealize Log Insight agents log files are located in the following directories.

- Windows - `%ProgramData%\VMware\Log Insight Agent\log`

- Linux - `/var/log/loginsight-agent/`

**What to do next**

For more information about configuring the vRealize Log Insight agents see Configure the Log Insight Windows Agent and Configure the Log Insight Linux Agent.

# Add an Outbound Exception Rule for the Log Insight Windows Agent

Define an exception rule for unblocking the Log Insight Windows Agent in the Windows firewall.

The procedure applies to Windows Server 2008 R2 and later, and to Windows 7 and later.

**Prerequisites**

- Verify that you have an administrator account or an account with administrative privileges.

**Procedure**

1 Select **Start > Run**.

2 Type `wf.msc` and click **OK**.

3 Right-click **Outbound rules** in the left pane and click **New Rule**.

4 Select **Custom** and follow the wizard to set the following options.

| Option | Description |
| --- | --- |
| **Program** | `liwinsvc.exe` |
| **Service** | LogInsightAgentService |
| **Protocol and Ports** | TCP 9000 for cfapi and 514 for syslog |

5 On the Specify the profiles for which this rule applies page, select the appropriate network type.

- Domain

- Public

- ■ Private

**Note** You can select all network types to make sure that the exception rule is active regardless of the network type.

**What to do next**

Go to the Log Insight Windows Agent log directory `%ProgramData%\VMware\Log Insight Agent\log` and open the latest log file. If recent events contain the messages `Config transport error: Couldn't resolve host name` and `Resolver failed. No such host is known`, restart the Log Insight Windows Agent service and the Windows machine.

**Note** The Log Insight Windows Agent service can take up to 5 minutes to reconnect to the server.

# Allow Outbound Connections from the Log Insight Windows Agent in a Windows Firewall

Configure Windows firewall settings to allow outbound connections of the Log Insight Windows Agent to the vRealize Log Insight server.

After you install and start the Log Insight Windows Agent service, the Windows domain or local firewall may restrict the connectivity to the target vRealize Log Insight server.

The procedure applies to Windows Server 2008 R2 and later, and to Windows 7 and later.

**Prerequisites**

- ■ Verify that you have an administrator account or an account with administrative privileges.

**Procedure**

1 Select **Start > Run**.

2 Type `wf.msc` and click **OK**.

3 In the Actions pane click **Properties**.

4 On the **Domain Profile** tab, select **Allow(default)** from the **Outbound connections** drop-down menu.

 If the computer is not connected to a domain, you can select **Private Profile** or **Public Profile**, depending on the network type the computer is connected to.

5 Click **OK**.

**What to do next**

Define an unblocking exception rule for the Log Insight Windows Agent in the Windows firewall. See Add an Outbound Exception Rule for the Log Insight Windows Agent.

# Mass Deployment of the Log Insight Windows Agent is Not Successful

The mass deployment of the Log Insight Windows Agent is not successful on target machines.

### Problem

After performing a mass deployment on Windows domain machines by using Group Policy Objects, the Log Insight Windows Agent fails to install as a local service.

### Cause

Group policy settings might prevent the Log Insight Windows Agent from being installed correctly.

### Solution

1   Edit the Group Policy Object (GPO) settings and redeploy the Log Insight Windows Agent agent.

   a   Right-click the GPO, click **Edit** and navigate to **Computer Configuration > Policies > Administrative Templates > System > Logon.**

   b   Enable the **Always wait for the network at computer startup and logon** policy.

   c   Navigate to **Computer Configuration > Policies > Administrative Templates > System > Group Policy**.

   d   Enable the **Startup policy processing wait time**, and set **Amount of time to wait (in seconds)** to 120.

2   Run the `gpupdate /force /boot` command on target machines.

# Log Insight Agents Reject Self-Signed Certificates

The Log Insight Agents reject self-signed certificate.

### Problem

A vRealize Log Insight agent rejects self-signed certificate and cannot establish a connection with the server.

**Note**   If you experience connection problems with the agent, you can generate detailed logs to check by changing the debug level for the agent to 1. For more information, see Define Log Details Level in the Log Insight Agents.

### Cause

The messages you see in the agent log have specific causes.

| Message | Cause |
|---|---|
| `Rejecting peer self-signed certificate. Public key doesn't match previously stored certificate's key.` | ■ This might happen when thevRealize Log Insight certificate is replaced.<br>■ This might happen if the HA-enabled in-cluster environment is configured with different self-signed certificates on vRealize Log Insight nodes. |
| `Rejecting peer self-signed certificate. Have a previously received certificate which is signed by trusted CA.` | There is a CA-signed certificate stored on the agent side. |

**Solution**

◆ Verify whether your target host name is a trusted vRealize Log Insight instance, and then manually delete the previous certificate from vRealize Log Insight Agent `cert` directory.

 ■ For Log Insight Windows Agent, go to `C:\ProgramData\VMware\Log Insight Agent\cert`.

 ■ For Log Insight Linux Agent, go to `/var/lib/loginsight-agent/cert`.

**Note** Some platforms might use nonstandard paths for storing trusted certificates. The Log Insight Agents have an option to configure the path to trusted certificates store by setting the **`ssl_ca_path=<fullpath>`** configuration parameter. Replace `<fullpath>` with the path to the trusted root certificates bundle file. See Configure the Log Insight Agents SSL Parameters.

# vRealize Log Insight Server Rejects the Connection for Non-Encrypted Traffic

The vRealize Log Insight Server rejects the connection with the Log Insight Agents when you try to send non-encrypted traffic.

You can configure a vRealize Log Insight Server to accept non-SSL connections or configure Log Insight Agents to send data through an SSL `cfapi` protocol connection.

**Problem**

When you attempt to use `cfapi` to send non-encrypted traffic, the vRealize Log Insight Server rejects your connection. One of the following error messages appears in the agent log: `403 Forbidden` or `403 Only SSL connections are allowed`.

**Cause**

vRealize Log Insight is configured to accept only SSL connections, but the Log Insight Agents are configured to use a non-SSL connection.

Solution

1  Configure your vRealize Log Insight Server to accept a non-SSL connection.

   a  Navigate to the **Administration** tab.

   b  Under Configuration, click **SSL**.

   c  Under the API Server SSL header, deselect **Require SSL Connection**.

   d  Click **Save**.

2  Configure the vRealize Log Insight agent to send data through an SSL `Cfapi` protocol connection.

   a  Navigate to the folder containing the `liagent.ini` file.

   | Operating system | Path |
   | --- | --- |
   | Linux | `/var/lib/loginsight-agent/` |
   | Windows | `%ProgramData%\VMware\Log Insight Agent` |

   b  Open the `liagent.ini` file in any text editor.

   c  Change the value of the `ssl` key in the [server] section of the `liagent.ini` file to yes and the protocol to `cfapi`.

   ```
   proto=cfapi
   ssl=yes
   ```

   d  Save and close the `liagent.ini` file.