Working with vRealize Log Insight Agents

Update 1

Modified on 03 SEP 2017

vRealize Log Insight 4.0

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

About 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 datacenter operations.

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

Updated Information for Working with vRealize Log Insight Agents

Working with vRealize Log Insight Agents is updated with each release of the product or when necessary.

This table provides a summary of changes.

Revision	Description
002369 -1	Editorial changes.
002369-0	Initial release.

Overview of vRealize Log Insight Agents

1

vRealize Log Insight agents collect events from log files on Linux and Windows machines and forwards them to the vRealize Log Insight server.

This chapter includes the following topics:

Overview of the vRealize Log Insight Windows Agent

The vRealize Log Insight Windows agent collects events from Windows event channels and log files, and forwards them to the vRealize Log Insight server.

A Windows event channel is a pool for collecting related events in a Windows system.

In a Windows system, applications can store log data in flat text files on the file system. The vRealize Log Insight Windows agent can monitor directories and collect events from flat text log files.

The vRealize Log Insight Windows agent has a limit of 64 KB per request to the vRealize Log Insight server.

The vRealize Log Insight Windows agent runs as a Windows service and starts immediately after installation. During and after installation, you can configure the following options for the vRealize Log Insight Windows agent:

- Select the target vRealize Log Insight server to which the vRealize Log Insight Windows agent forwards events.
- Select the communication protocol and port that the vRealize Log Insight Windows agent uses.
- Add additional Windows event channels from which the vRealize Log Insight Windows agent collects events.
- Select Windows directories to monitor and add flat log files to the collection.

The vRealize Log Insight Windows agent requires Windows Vista or later, or Windows Server 2008 or later.

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

Overview of the Log Insight Linux Agent

The Log Insight Linux Agent collects events from log files on Linux machines and forwards them to the vRealize Log Insight server.

In a Linux system, applications can store log data in flat text files on the file system. The Log Insight Linux Agent can monitor directories and collect events from flat text log files.

The Log Insight Linux Agent runs as a daemon and starts immediately after installation. After installation, you can configure the following options:

- Select the target vRealize Log Insight server to which the Log Insight Linux Agent forwards events.
- Configure which directories the Log Insight Linux Agent monitors.

The Log Insight Linux Agent supports the following distributions and versions.

- RHEL 5, RHEL 6, and RHEL 7
- SLES 11 SP3 and SLES 12 SP1
- Ubuntu 12.04 LTS, 14.04 LTS, and 16.04 LTS

The Log Insight Linux Agent writes its own operation log files to `/var/log/loginsight-agent/liagent_*.log`. Log files are rotated when the Log Insight Linux Agent is restarted and when they reach a size of 10 MB. A combined limit of 50 MB are kept in rotation.

To download the Log Insight Linux Agent package, navigate to the Administration page of the vRealize Log Insight Web user interface, click **Agents** in the Management section, and click the appropriate package link.

If you implement a default installation of the Log Insight Linux Agent for a user without root privileges to use, 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.

Installing vRealize Log Insight Agents

2

The Log Insight Windows and Linux agents collect events from Windows and Linux machines and forwards them to the vRealize Log Insight server. You can install and configure parameters for the server, port, and protocol or chose to keep the default settings.

To install and run vRealize Log Insight agent, there are minimum hardware parameters required for hosts/machines to support x86 and x86_64 architecture and MMX, SSE, SSE2, and SSE3 instruction sets.

This chapter includes the following topics:

Download Windows or Linux Agent Files

Before you install and configure a vRealize Log Insight agent, download the agent file.

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

Procedure

1. Navigate to the **Administration** page of the vRealize Log Insight web user interface.
2. In the Management section, click **Agents**.

3 Click **Download Log Insight Agent** and choose the agent file to download.

Option	Description
Windows MSI	Windows MSI (32-bit/64-bit)
Linux RPM	Linux RPM (32-bit/64-bit)
Linux DEB	Linux DEB (32-bit/64-bit)
Linux BIN	Linux BIN (32-bit/64-bit)

What to do next

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

Install the vRealize Log Insight Windows Agent with a Default Configuration

You can install the vRealize Log Insight Windows agent without configuring command-line parameters.

Prerequisites

- Verify that you have a copy of the vRealize Log Insight Windows agent `.msi` file. See Download Windows or Linux Agent 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 the vRealize Log Insight Windows agent as an automatic Windows Service under the LocalSystem 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 After Installation.

Install, Update, and Configure the vRealize Log Insight Windows Agent

You can install or update the vRealize Log Insight Windows agent, specify a service account, and configure command-line parameters for the server, port, and protocol.

For MSI command-line options, see the Microsoft Developer Network (MSDN) Library Web site 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 Windows or Linux Agent 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. Run the command to start the installation or update and replace *Version-Build_Number* with your version and build number.

   ```
   Drive:\path-to-msi_file>VMware-Log-Insight-Agent-Version-Build_Number.msi
   ```

   `Drive:\path-to-msi_file>VMware-Log-Insight-Agent-30.msi`.
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 username 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) Enter the vRealize Log Insight server, port, and protocol.

Parameter	Description
`SERVERHOST`	IP address or host name of the vRealize Log Insight virtual appliance.
`SERVERPROTO`	Protocol that the agent uses to send events to the vRealize Log Insight server. The possible values are `cfapi` and `syslog`. Use the default cfapi setting.
`SERVERPORT`	The port number depends on the value of `SERVERPROTO`. The default value for `SERVERPORT` is 9000, which corresponds to the default `SERVERPROTO=cfapi`. Use `SERVERPORT=514` for `SERVERPROTO=syslog`.

The command-line parameters correspond to `hostname`, `proto`, and `port` in the `[server]` section of the `liagent.ini` file.

7 Press Enter.

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 After Installation.

Deploy the Log Insight Windows Agent to Multiple Machines

You can deploy the Log Insight Windows Agent to multiple target machines in a Windows domain.

Deploy the vRealize Log Insight Windows Agent with a Transform .mst file

To specify installation parameters to be used during deployment, you create an `.mst` transform file. You can configure the vRealize Log Insight Windows agent to send events to a vRealize Log Insight server, and to set 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 Windows or Linux Agent 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` fie 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.

Parameter	Description
SERVERHOST	IP address or host name of the vRealize Log Insight virtual appliance.
SERVERPROTO	Protocol that the agent uses to send events to the vRealize Log Insight server. The possible values are `cfapi` and `syslog`. Use the default cfapi setting.
SERVERPORT	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 disabled, 6514 for syslog with SSL enabled and 514 for `syslog` with SSL disabled.
SERVICEACCOUNT	User service account under which the Log Insight Windows Agent service will 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 of the user service account.

3 Select **Transform > Generate Transform** and save the `.mst` 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 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 Windows or Linux Agent 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 the target server during installation. After installation, you can verify the installed version.

Prerequisites

- 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 run levels 3 and 5. If you want the vRealize Log Insight Linux agent to work under other runlevels, configure the system appropriately.

Procedure

1 Open a console and run the `rpm -i package_name` command to install the vRealize Log Insight Linux agent.

Replace *package_name* with the appropriate version.

```
rpm -i VMware-Log-Insight-Agent-VERSION-BUILD_NUMBER.rpm
```

Note

```
sudo SERVERHOST=hostname rpm -i VMware-Log-Insight-Agent-VERSION-BUILD_NUMBER.rpm
```

2 To set the target vRealize Log Insight server during installation run the `sudo` command and replace *hostname* with the IP address or hostname of the vRealize Log Insight server.

```
sudo SERVERHOST=hostname rpm -i VMware-Log-Insight-Agent-VERSION-BUILD_NUMBER.rpm
```

3 (Optional) To update the vRealize Log Insight Linux agent run the `rpm -Uhv` command.

```
rpm -Uhv VMware-Log-Insight-Agent-VERSION-BUILD_NUMBER.rpm
```

Note You can run other RPM commands such as `-h, --hash`, `--version`,`--allfiles` and so on during the installation, update, or uninstallation of the vRealize Log Insight Linux agent RPM package, but they are not supported.

4 (Optional) To install the vRealize Log Insight Linux agent as a **non root** user, run the `sudo` command.

```
sudo LIAGENTUSER=liagent rpm -i VMware-Log-Insight-Agent-VERSION-BUILD_NUMBER.rpm
```

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 vRealize Log Insight Linux agent with the `LIAGENTUSER=`*`non_root_user`* parameter and try to upgrade with `LIAGENTUSER=`*`non_root_user2`*, a conflict occurs and warnings appear because *non_root_user2* user does not have the permissions of the user *non_root_user*.

5 (Optional) Double click the appropriate version of the RPM package to install or update the vRealize Log Insight Linux agent.

6 (Optional) Verify the installed version by running the `rpm -qa | grep Log-Insight-Agent` command.

Install or Update the vRealize Log Insight Linux Agent DEB package

When you install or update the vRealize Log Insight Linux agent DEB package, you can set the target server during installation and keep or replace the liagent.ini configuration file. After installation, you can verify the installed version.

Prerequisites

- 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.

Procedure

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

Replace *package_name* with the appropriate version.

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

2 To set the target vRealize Log Insight server during installation run the `sudo` command and replace *hostname* with the IP address or hostname of the vRealize Log Insight server.

```
sudo SERVERHOST=hostname dpkg -i vmware-log-insight-agent-VERSION-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] ?
```

3 (Optional) To preserve the existing configuration, use `[default=N]`. The additional parameters passed from the comand line are still applied.

4 (Optional) 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 vRealize 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.

5 (Optional) Verify the installed version by running the `dpkg -l | grep -i vmware-log-insight-agent` command.

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 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.
- Log in as **root** or use `sudo` to run console commands.
- Verify that the Log Insight Linux Agent has access to syslog and networking services.

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 Run the `./filename-version.bin` command to install the agent.

   Replace *filename-version* with the appropriate version.

   Note

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

3 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.

4 (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 oocurs and warnings appear because the *non_root_user2* user does not have the permissions of the *non_root_user* user.

Configuring a vRealize Log Insight Agent 3

After you have deployed an agent, you can configure it to send events to the vRealize Log Insight server that you select, specify communication protocols, and so on.

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

- Configure the Log Insight Windows Agent After Installation

  You can configure the Log Insight Windows Agent after the installation. You must edit the `liagent.ini` file to configure Log Insight Windows Agent to send events to a vRealize Log Insight server of your choice, set communication protocol and port, add Windows event channels, and configure flat file log collection.

- Configure the Log Insight Linux Agent

  You can configure the Log Insight Linux Agent after you install it.. The `liagent.ini` file is located in `/var/lib/loginsight-agent/`. Edit the file to configure the Log Insight Linux Agent to send events to a vRealize Log Insight server of your choice, set communication protocol and port, and configure flat file log collection.

- Centralized Configuration of vRealize Log Insight Agents

  You can configure multiple Windows or Linux vRealize Log Insight 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 After Installation

You can configure the Log Insight Windows Agent after the installation. You must edit the `liagent.ini` file to configure Log Insight Windows Agent to send events to a vRealize Log Insight server of your choice, set communication protocol and port, add Windows event channels, and configure flat file log collection.

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.

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=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
; 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

[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=200

; Uncomment the following sections to collect these channels.
; The recommended way is to enable Windows content pack from LI server.
;[winlog|Application]
;channel=Application

;[winlog|Security]
;channel=Security
```

```
;[winlog|System]
;channel=System
```

Parameter	Value	Description
`proto`	`cfapi`	Protocol that the agent uses to send events to the vRealize Log Insight server. The possible values are `cfapi` and `syslog`. Use the default cfapi setting.
`hostname`	`LOGINSIGHT`	IP address or host name of the vRealize Log Insight virtual appliance.
`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 disabled, 6514 for syslog with SSL enabled and 514 for `syslog` with SSL disabled.
`ssl`	yes	Enables or disables 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 Windows 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.
`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. See Collect Events from Windows Events Channels.

Set Target vRealize Log Insight Server

You can set or change the target vRealize Log Insight server that the vRealize Log Insight Windows agent sends event to, if you have not set the values during the installation process.

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 folder 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 events to the vRealize Log Insight server. The possible values are `cfapi` and `syslog`. Use the default cfapi setting.

hostname	IP address or host name of the vRealize Log Insight virtual appliance.
	You can specify an IPv4 or IPv6 address. An IPv6 address can be specified with or without square brackets. For example:
	`hostname = 2001:cdba::3257:9652`
	`or`
	`hostname = [2001:cdba::3257:9652]`
	If the host supports both IPv4 and IPv6 stacks and a domain name is specified as the hostname, then the agent will use the IP stack depending on the IP address that is returned by the name resolver. If the resolver returns both IPv4 and IPv6 addresses, than the agent will try to connect sequentially to both addresses in the given order.

port	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 disabled, 6514 for syslog with SSL enabled and 514 for `syslog` with SSL disabled.
ssl	Enables or disables 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.
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
```

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

Example: Configuration

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
```

What to do next

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

Collect Events 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 events and send them to the vRealize Log Insight server.

Field names are restricted. The following field 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 folder 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 disable 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 will be ignored as a duplicate. You cannot use event_type and timestamp as tag names. Any duplicates within the same declaration are ignored. Tags can override the APP-NAME field, if the destination is a syslog server. 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 `blacklist` 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 `blacklist` text.

- `whitelist` collects only log events for which the filter expression evaluates to non-zero. If you omit `whitelist`, the value is an implied 1.
- `blacklist` 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 folder 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 Events from a Log File

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

Collecting from Encrypted Folders

An agent is able to collect from encrypted folders. The Agent will collect from an encrypted folder only if it is run by the user who encrypted the folder.

Field names are restricted. The following field 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 folder of the vRealize Log Insight Windows agent.

   `%ProgramData%\VMware\Log Insight Agent`

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

3 Add configuration parameters and set the values for your environment.

Parameter	Description
`[filelog|`*`section_name`*`]`	A unique name for the configuration section.
`directory`	The full path to the log file directory. 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. **Note** If you use exactly the same configurations for these sections, duplicated events are observed on the server side.
`include`	(Optional) The name of a file name 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. **Note** 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 the vRealize 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`	(Optional) A file name 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.

Parameter	Description
event_marker	(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 200ms 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	(Optional) A parameter to enable or disable the configuration section. The possible values are `yes` or `no`. The default value is `yes`.
charset	(Optional) The character encoding of the log files that the agent monitors. The possible values are `UTF-8`, `UTF-16LE`, and `UTF-16BE`. The default value is `UTF-8`.
tags	(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 will be ignored as a duplicate. You cannot use event_type and timestamp as tag names. Any duplicates within the same declaration are ignored. Tags can override the APP-NAME field, if the destination is a syslog server. 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, ■ `exclude_fields=hostname; filepath` ■ `exclude_fields=type; size` ■ `exclude_fields=type, size`

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

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 `blacklist` text.

- `whitelist` collects only log events for which the filter expression evaluates to non-zero. If you omit `whitelist`, the value is an implied 1.
- `blacklist` 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 folder 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 Events to the Log Insight Windows Agent

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

You can use Windows Event Forwarding to forward events 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 events and send them to a vRealize Log Insight server.

Get familiar with Windows Event 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 Events from Windows Events Channels.

Procedure

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

The default channel name is ForwardedEvents.

2 Set up Windows Event Forwarding.

What to do next

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

Configure the Log Insight Linux Agent

You can configure the Log Insight Linux Agent after you install it.. The `liagent.ini` file is located in `/var/lib/loginsight-agent/`. Edit the file to configure the Log Insight Linux Agent to send events to a vRealize Log Insight server of your choice, set communication protocol and port, and configure flat file log collection.

Default Configuration of the vRealize Log Insight Linux Agent

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

vRealize Log Insight Linux Agent liagent.ini Default Configuration

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

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

; 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

[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=200

; 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	Value	Description
`proto`	`cfapi`	Protocol that the agent uses to send events to the vRealize Log Insight server. The possible values are `cfapi` and `syslog`. Use the default cfapi setting.
`hostname`	`LOGINSIGHT`	IP address or host name of the vRealize Log Insight virtual appliance.
`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 disabled, 6514 for syslog with SSL enabled and 514 for syslog with SSL disabled.
`ssl`	yes	Enables or disables 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 Windows 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.

Using Common Values for Linux Agent Configuration

You can override the default values of the agent configuration file with common parameter values that apply each agent configuration section.

Common Options

Options specified in the `[common|global]` section of the 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:

```
[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 follows 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.

Set Target vRealize Log Insight Server

You can set or change the target vRealize Log Insight server that the vRealize Log Insight Linux agent sends events to.

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 enabled Integrated Load Balancer, see Enable 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 events to the vRealize Log Insight server. The possible values are `cfapi` and `syslog`. Use the default cfapi setting.

hostname	IP address or host name of the vRealize Log Insight virtual appliance.
	You can specify an IPv4 or IPv6 address. An IPv6 address can be specified with or without square brackets. For example:
	`hostname = 2001:cdba::3257:9652`
	`or`
	`hostname = [2001:cdba::3257:9652]`
	If the host supports both IPv4 and IPv6 stacks and a domain name is specified as the hostname, then the agent will use the IP stack depending on the IP address that is returned by the name resolver. If the resolver returns both IPv4 and IPv6 addresses, than the agent will try to connect sequentially to both addresses in the given order.

port	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 disabled, 6514 for syslog with SSL enabled and 514 for `syslog` with SSL disabled.
ssl	Enables or disables 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.
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: Configuration

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
```

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.

Collect Events from a Log File

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

Note By default the vRealize 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.

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

- `event_type`
- `hostname`
- `source`
- `text`

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

2 Add configuration parameters and set the values for your environment.

Parameter	Description
`[filelog|`*section_name*`]`	A unique name for the configuration section.
`directory`	The full path to the log file directory. 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. **Note** If you use exactly the same configurations for these sections, duplicated events are observed on the server side.
`include`	(Optional) The name of a file name 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. **Note** 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 the vRealize 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`	(Optional) A file name 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`	(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 200ms 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`	(Optional) A parameter to enable or disable the configuration section. The possible values are `yes` or `no`. The default value is `yes`.
`charset`	(Optional) The character encoding of the log files that the agent monitors. The possible values are `UTF-8`, `UTF-16LE`, and `UTF-16BE`. The default value is `UTF-8`.

Parameter	Description
tags	(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 will be ignored as a duplicate. You cannot use event_type and timestamp as tag names. Any duplicates within the same declaration are ignored. Tags can override the APP-NAME field, if the destination is a syslog server. 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, ■ `exclude_fields=hostname; filepath` ■ `exclude_fields=type; size` ■ `exclude_fields=type, size`

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

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=*
```

Set up Linux Log File Channel Filtering

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

Note By default the vRealize 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.

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 `blacklist` text.

- `whitelist` collects only log events for which the filter expression evaluates to non-zero. If you omit `whitelist`, the value is an implied 1.
- `blacklist` excludes log events for which the filter expression evaluates to non-zero. The default value is 0.

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

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

3. Create a filter expression from Linux events fields and operators.

   For example

   ```
   whitelist = server_name
   ```

4. 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 `sample.com` and the `remote_host` is not equal to 127.0.0.1, for example

```
[filelog|apache]
directory=/var/log/httpd
include=access_log
parser=clf
whitelist = server_name == "sample.com"
blacklist = remote_host == "127.0.0.1"
```

Centralized Configuration of vRealize Log Insight Agents

You can configure multiple Windows or Linux 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 machine where the vRealize Log Insight agent is installed. The server-side configuration is accessible and editable, for example, in Windows from **Administration > Agents** in the Web user interface. The configuration of each vRealize Log Insight agent is composed of sections and keys. Keys have configurable values.

The 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.

An Admin vRealize Log Insight user can apply centralized configuration to all vRealize Log Insight agents. For example, in Windows, 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 connected agents during the next poll cycle.

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

See Configure the Log Insight Windows Agent After Installation.

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} - -
```

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.

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

- `event_type`
- `hostname`
- `source`
- `text`

Common 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 string where keys are the names of the field to apply 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.
`field_rename`	Renames extracted fields. This is a JSON string where keys are the original names of the fields and values are the new desired names of the fields. Note that `field_decoder` 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. This is applied 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 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 that is defined by the particular section. The default value for debug is `debug=no` for parsers.

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 = timepstamp,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"
`'%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" 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" 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"

Fields	Value
`'%q':`	"query_string"
`'%r':`	"request"
`'%R':`	"response_handler"
`'%s':`	"status_code". which generates the final status of the request, is also supported. This appears on the server as "status_code".

`'%t':`	"timestamp" will work as event timestamp on ingestion, engages 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" depends on the name of variable specified in the format
`'%x':`	depends on the name of variable specified in the format
`'%I':`	"received_bytes"

Fields	Value
'%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 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.
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 which are enclosed in the double quotes will be used as a delimiter. For the kvp parser, any character can be considered as delimiter. You an 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 an 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 = 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

Key	Definition
`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 required.

```
[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
```

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, zero-padded (01-31), for example: 03
'%e':    Day of the month, space-padded ( 1-31), for example:  3
'%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
'%I':    Hour in 12h format (01–12), for example: 02
'%m':    Month as a decimal number (01–12), for example: 08
'%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.

Syslog Parser

The syslog parser by default extracts the `timestamp` and `app name` fields only.

All common options and the `message_decoder` option is available for the syslog parser.

```
[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=*
```

For example, if the syslog format log is:

```
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"
```

a syslog parser for which the `message_decoder` option is applied to run a KVP parser, will return the following:

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

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.

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 should 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, the parser will not be configured.

Regex Parser Options

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

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

Configuration

To create a `regex` parser, `regex` should be used as a `base_parser` and the `format` option must be provided.

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 would 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 would 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+)
```

Uninstalling vRealize Log Insight Agents 4

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

You can uninstall the Log Insight Windows Agent.

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**.

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
```

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
```

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.

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. 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 5

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

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`

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
```

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.

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 Set Target vRealize Log Insight Server.

vRealize Log Insight Agents Do Not Send Events

An incorrect configuration can prevent the vRealize Log Insight agents from forwarding events 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 events 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 events 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 3 Configuring a vRealize Log Insight Agent.
- 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 After Installation 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.

Installation of RPM Package Update Fails

Attempts to install an RPM package update fail when you use the Linux GUI.

Problem

Installing or updating the Log Insight Linux Agent RPM package fails when you use the GUI in RHEL and SUSE Linux distributions. You might see the error message `PK_TMP_DIR| dir:///var/tmp/TmpDir.MtqOPs] Repository already exists.`

Cause

The cache and repository list might not clean after you install applications.

Solution

1 Log in to the Linux system where the Log Insight Linux Agent RPM is installed and open a system console.

2 Run the following commands as a **root** user.

```
sudo zypper rr 2
sudo zypper rr 1
sudo zypper clean -a
sudo zypper ref
```

3 Double-click the Log Insight Linux Agent RPM package to install the update.

Log Insight Agents Reject Self-Signed Certificate

The Log Insight Agents reject self-signed certificate.

Problem

The Log Insight Agents reject self-signed certificate and cannot establish connection with the server.

Note If you experience connection problems with the Agent, you can check the detailed logs by changing the debug level for the vRealize Log Insight Agent to 1. See Define Log Details Level in the Log Insight Agents.

Cause

The messages you see in the vRealize Log Insight Agent log have specific reasons.

Message	Cause
`Rejecting peer self-signed certificate. Public key doesn't match previously stored certificate's key.`	■ This might happen when the Log Insight Server certificate is replaced. ■ 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 at 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.

Problem

When you attempt to use `cfapi` to send nonencrypted traffic, the vRealize Log Insight Server rejects your connection. The following error message appears in the Log Insight Agent log.

```
403 Forbidden.
```

Cause

vRealize Log Insight is configured to accept only SSL connections, but the Log Insight Agents are configured to use non-SSL connection.

Solution

You can configure vRealize Log Insight Server to accept non-SSL connections or configure the Log Insight Agents to send data through SSL `cfapi` protocol connection.

Procedure

1 Configure vRealize Log Insight Server to accept non-SSL connection.

   a Click the configuration drop-down menu icon ☰ and select **Administration**.

   b Under Configuration, click **SSL**.

   c Under the API Server SSL header, deselect **Require SSL Connection**.

   d Click **Save**.

2 Configure the Log Insight Agents to send data through 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 `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.