



ArcSight SmartConnector

Software Version: CE 24.1

Configuration Guide for IBM AIX Audit Syslog SmartConnector

Document Release Date: January 2024

Software Release Date: January 2024

Legal Notices

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Copyright Notice

Copyright 2024 Open Text.

The only warranties for products and services of Open Text and its affiliates and licensors (“Open Text”) are as may be set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Open Text shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Trademark Notices

“OpenText” and other Open Text trademarks and service marks are the property of Open Text or its affiliates. All other trademarks or service marks are the property of their respective owners.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number
- Document Release Date, which changes each time the document is updated
- Software Release Date, which indicates the release date of this version of the software

To check for recent updates or to verify that you are using the most recent edition of a document, go to:

<https://www.microfocus.com/support-and-services/documentation>

Configuration Guide for SmartConnector for IBM AIX Audit Syslog

This guide provides information about installing the SmartConnector for IBM AIX Audit Syslog and configuring the device for event collection.

The purpose of the AIX auditing system is to record instances of access by subjects to objects and to allow detection of any (repeated) attempts to bypass the protection mechanism and any misuses of privileges.

Intended Audience

This guide provides information for IT administrators who are responsible for managing the ArcSight software and its environment.

Additional Documentation

The ArcSight SmartConnector documentation library includes the following resources:

- [Technical Requirements Guide for SmartConnector](#), which provides information about operating system, appliance, browser, and other support details for SmartConnector.
- [Installation and User Guide for SmartConnectors](#), which provides detailed information about installing SmartConnectors.
- [Configuration Guides for ArcSight SmartConnectors](#), which provides information about configuring SmartConnectors to collect events from different sources.
- [Configuration Guide for SmartConnector Load Balancer](#), which provides detailed information about installing Load Balancer.

For the most recent version of this guide and other ArcSight SmartConnector documentation resources, visit the [documentation site for ArcSight SmartConnectors 8.4](#).

Contact Information

We want to hear your comments and suggestions about this book and the other documentation included with this product. You can use the comment on this topic link at the bottom of each page of the online documentation, or send an email to MFI-Documentation-Feedback@opentext.com.

For specific product issues, [contact Open Text Support for Micro Focus products](#).

Configuration

Configuring AIX Audit

Collecting Events

Information collection encompasses logging the selected auditable events. The audit logger is responsible for constructing the complete audit record, consisting of the audit header, which contains information common to all events (such as the name of the event, the user responsible, the time and return status of the event) and the audit trail, which contains event-specific information. The audit logger appends each successive record to the kernel audit trail, which can be written in either (or both) BIN and STREAM modes.

Selecting Audit Events

Auditing lets you detect activities that might compromise the security of your system. When performed by an unauthorized user, the following activities violate system security and are candidates for an audit:

- Engaging in activities in the Trusted Computing Base
- Authenticating users
- Accessing the system
- Changing the configuration of the system
- Circumventing the auditing system
- Initializing the system
- Installing programs
- Modifying accounts
- Transferring information into or out of the system

The audit system does not have a default set of events to be audited. You must select events or event classes according to your needs.

To audit an activity, identify the command or process that initiates the audit event and ensure that the event is listed in the `/etc/security/audit/events` file for your system. Then add the event either to an appropriate class in the `/etc/security/audit/config` file, or to an object stanza in the `/etc/security/audit/objects` file.

See the `/etc/security/audit/events` file on your system for the list of audit events and trail formatting instructions. For a description of how audit event formats are written and used, see the `auditpr` command.

Grouping into Audit Classes

After you have selected the events to audit, combine similar events into audit classes. These audit classes are defined in the classes stanza of the `/etc/security/audit/config` file. Then assign audit classes to users. Some typical audit classes are:

- **General**
Events that alter the state of the system and change user authentication. Audit attempts to circumvent system access controls.
- **Objects**
Write access to security configuration files.
- **Kernel**
Events in the kernel class are generated by the process management functions of the kernel.

An example of a stanza in the `/etc/security/audit/config` file follows.

classes:

```
general = USER_SU,PASSWORD_Change,FILE_Unlink,FILE_Link,FILE_Rename
system = USER_Change,GROUP_Change,USER_Create,GROUP_Create
init = USER_Login,USER_Logout
```

Assigning Audit Events to an Object

Assign the audit events to an object (data or executable file) by adding a stanza for that file to the `/etc/security/audit/objects` file. To get all audit events, specify the ALL class; however, be aware that with this option, a huge amount of data will be generated.

Selecting an Audit Data Collection Method

The audit data collection method you choose depends upon how you intend to use the audit data. If you need long-term storage of a large amount of data, select BIN collection. If you want to process the data as it is collected, select STREAM collection. If you need both long-term storage and immediate processing, select both methods.



You can use `streammode=on` and `binmode=off` when using the IBM AIX Audit Syslog SmartConnector.

In the `/etc/security/audit/config` file, configure whether you want to use BIN collection, STREAM collection, or both methods. Use a separate file system for audit data to ensure that audit data does not compete with other data for file space.

To configure STREAM collection:

1. Enable the STREAM mode collection by setting `streammode = on` in the **Start** stanza.
2. Edit the **Streammode** string to specify the path to the file containing the streammode processing commands. The default file containing this information is `/etc/security/audit/streamcmds`.
3. Include the shell commands that process the stream records in an audit pipe in the `/etc/security/audit/streamcmds` file.

Enabling the Audit Subsystem

When you have finished making any necessary changes to the configuration files, you can use the `audit start` command to enable the audit subsystem. You can use the `audit shutdown` command to deactivate the audit subsystem.

The auditpr Command

The `auditpr` command reads audit records, in bin or stream format, from standard input and sends formatted records to standard output.

The output format is determined by flags that are selected. If you specify the `-m` flag, a message is displayed before each heading. Use the `-h` flag to change the default fields and the `-v` flag to append an audit trail. The `auditpr` command searches the local `/etc/passwd` file to convert user and group IDs to names.

Values that can be used with the `-h` flag to select fields are as follows:

Value	Description
e	The audit event.
l	The user's login name.
R	The audit status.
t	The time the record was written.
c	The command name.
r	The real user name.
p	The process ID.
i	The IDs or the names of roles of the audited process.

Value	Description
E	The effective privilege.
S	The effective sensitivity label (SL).
I	The effective integrity label (TL).
W	The workload partition name.
P	The ID of the parent process.
T	The kernel thread ID (local to the process; different process can contain threads with the same thread ID).
h	The name of the host that generated the audit record. If there is no CPU ID in the audit record, the value none is used. If there is no matching entry for the CPU ID in the audit record, the 16-character value for the CPU ID is used instead.

The e, I, R, t, and c flags are used by default. For more information on auditpr commands, see [auditpr Command in IBM Documentation](#).

Examples

Sample Event File

An example of the /etc/security/audit/event file:

```
[#]/etc/security/audit]> cat events
```

```
....
```

```
auditpr:
```

```
    ...other rows precede
```

```
*kernel proc events
```

```
*   fork()
    PROC_Create = printf "forked child process %d"
```

```
*   exit()
    PROC_Delete = printf "exited child process %d"
```

```
*   exec()
    PROC_Execute = printf "euid: %d egid: %d epriv: %x:%x name %s"
```

```
    ... other rows follow
```

For examples of audit trails, see the /etc/security/audit/events file where the audit trail formats are defined.

Example of auditpr Command

```
[#][/] /usr/sbin/audit pr -v < audit/trail
```

event	login	status	time	command
FS_Chdir	root	OK	Tue Oct 05 12:58:26 2004	ksh
FILE_Unlink	root	OK	Tue Oct 05 12:59:03 2004	vi
FILE_Unlink	root	OK	Tue Oct 05 12:59:12 2004	vi
FS_Chdir	root	OK	Tue Oct 05 12:59:34 2004	ksh
FS_Chdir	root	OK	Tue Oct 05 12:59:37 2004	ksh
FILE_Unlink	root	OK	Tue Oct 05 12:59:40 2004	vi
FILE_Unlink	root	OK	Tue Oct 05 12:59:59 2004	vi
CRON_Start	root	OK	Tue Oct 05 13:00:00 2004	cron
FS_Chdir	root	OK	Tue Oct 05 13:00:00 2004	cron
FILE_Unlink	root	OK	Tue Oct 05 13:00:02 2004	vi
FILE_Unlink	root	OK	Tue Oct 05 13:00:04 2004	vi
FILE_Unlink	root	OK	Tue Oct 05 13:02:38 2004	vi
FILE_Unlink	root	OK	Tue Oct 05 13:02:44 2004	vi
FILE_Unlink	root	OK	Tue Oct 05 13:02:44 2004	vi
TCPIP_connect	root	OK	Tue Oct 05 13:20:15 2004	telnetd
FILE_Write	root	OK	Tue Oct 05 13:20:15 2004	telnetd

Example of Config File for BIN Mode

```
[#][etc/security/audit]> head -20 config
```

```
start:
```

```
    binmode = on
    streammode = on
```

```
bin:
```

```
    trail = /audit/trail
    bin1 = /audit/bin1
    bin2 = /audit/bin2
    binsize = 10240
    cmds = /etc/security/audit/bincmds
    freespace = 65536
```

```
...
```

```
[#][etc/security/audit]> cat /etc/securitiy/audit/bincmds /usr/sbin/auditcat -p -o $trail $bin
[p630n02][etc/security/audit]>
```


Example of Config File for STREAM Mode

```
[#][etc/security/audit]> cat config
```

start:

```
    binmode = on
    streammode = on
```

stream:

```
    cmds = /etc/security/audit/streamcmds
```

...

```
[#][etc/security/audit]> cat /etc/security/audit/streamcmds
/usr/sbin/auditstream | auditpr -v > /audit/stream.out &
[#][ETC/SECURITY/AUDIT]>
```

Example of a Generic Audit Log Scenario

This example was derived from the AIX Security Guide in the Setting Up Auditing chapter in the section called Generating a Generic Audit Log. See

https://www.ibm.com/support/knowledgecenter/ssw_aix_71/com.ibm.aix.security/generic_audit_log.htm for details.

In this example, assume that a SYSADMIN wants to use the audit subsystem to monitor a large multi-user server system. No direct integration into an IDS is performed, all audit records will be inspected manually for irregularities. Only a few essential audit events are recorded, to keep the amount of generated data to a manageable size.

The audit events that are considered for event detection are:

Audit Events	Description
FILE_WRITE	We want to know about file writes to configuration files, so this event will be used with all files in the /etc tree.
PROC_SetUserIDs	All changes of user ids.
AUD_Bin_Def	Audit bin configuration.
USER_SU	The su command.
PASSWORD_Change	The passwd command.
AUD_Lost_Rec	Notification in case there where lost records.
CRON_JobAdd	New cron jobs.

AT_JobAdd	New at jobs.
USER_Login	All logins.
PORT_Locked	All locks on terminals because of too many invalid attempts.

The following is an example of how to generate a generic audit log:

1. Set up a list of critical files to be monitored for changes, such as all files in /etc, and configure them for FILE_Write events in the objects file as follows:

```
find /etc -type f | awk '{printf("%s:\n\tw = FILE_Write\n\n",$1)}' >>
/etc/security/audit/objects
```
2. se the auditcat command to set up BIN mode auditing. The /etc/security/audit/bincmds file is similar to the following:

```
/usr/sbin/auditcat -p -o $trail $bin
```
3. Edit the /etc/security/audit/config file and add a class for the events in which we are interested. List all existing users and specify the custom class for them.

start:

```
binmode = on
streammode = off
```

bin:

```
cmds = /etc/security/audit/bincmds
trail = /audit/trail
bin1 = /audit/bin1
bin2 = /audit/bin2
binsize = 100000
freespace = 100000
```

classes:

```
custom = FILE_Write,PROC_SetUser,AUD_Bin_Def,AUD_Lost_Rec,
        USER_SU,PASSWORD_Change,CRON_JobAdd,AT_JobAdd,USER_Login,
        PORT_Locked
```

users:

```
root = custom
afx = custom
...
```

4. Add the custom audit class to the /usr/lib/security/mkuser.default file, so that new IDs will automatically have the correct audit call associated:

user:

```
auditclasses = custom
pgrp = staff
```

```
groups = staff
shell = /usr/bin/ksh
home = /home/$USER
```

5. Create a new file system named /audit by using SMIT or the crfs command. The file system should be large enough to hold the two bins and a large audit trail.
6. Run the audit start command option and examine the /audit file. You should see the two bin files and an empty trail file initially. After you have used the system for a while, you should have audit records in the trail file that can be read with:

```
auditpr -hhhelpPRrTc -v | more
```

This example uses only a few events. To see all events, you could specify the classname ALL for all users. This action will generate large amounts of data. You might want to add all events related to user changes and privilege changes to your custom class.

Example of Real-Time File Modification Monitoring

The following example can be used to monitor file access to critical files in real time:

1. Set up a list of critical files to be monitored for changes; for example, all files in /etc, and configure them for FILE_Write events in the objects file.

```
find /etc -type f | awk '{printf("%s:\n\tw = FILE_Write\n\n",$1)}' >>
/etc/security/audit/objects
```
2. Set up stream auditing to list all file writes. (This example lists all file writes to the console, but in using the ArcSight SmartConnector in a production environment, you would want to have a backend that sends the events into an Intrusion Detection System.) The /etc/security/audit/streamcmds file is similar to the following:

```
/usr/sbin/auditstream | /usr/sbin/auditselect -e "event == FILE_Write" | auditpr -
hhhelpPRrTc -v > /dev/console &
```
3. Set up STREAM mode auditing in /etc/security/audit/config; add a class for the file write events and configure all users that should be audited with that class:

start:

```
binmode = off
streammode = on
```

stream:

```
cmds = /etc/security/audit/streamcmds
```

classes:

```
filemon = FILE_Write
```

users:

```
root = filemon
```

afx = filemon

...

- Now run audit start. All FILE_Write events are displayed on the console.



When the audit start or audit shutdown command is executed, the configuration information is reset and the audit logs are flushed to the streams. When this happens, the SmartConnector must be restarted.

AIX Configuration Files

AIX configuration files you might need to access include:

File	Description
/usr/sbin/auditselect	Specifies the path of the auditselect command.
/etc/rc	Contains the system initialization commands.
/etc/security/audit/config	Contains audit system configuration information.
/etc/security/audit/events	Contains the audit events of the system.
/etc/security/audit/objects	Contains audit events for audit objects (files).
/etc/security/audit/bincmds	Contains auditbin backend commands.
/etc/security/audit/streamcmds	Contains auditstream commands.

For information about configuring AIX auditing for your AIX version, see [AIX 7.1 Security](#) and [AIX 7.2 Security](#).

Selecting a Method to Send AIX Audit Messages Using Syslog

AIX Audit messages are multiline and are not supported by syslog. The following instructions allow the multiline messages to be reassembled to be forwarded and understood by the IBM AIX Audit Syslog SmartConnector in one of three ways:

- Solution 1: Send the messages to the local syslog daemon using a UNIX socket and configure the daemon to forward the messages to the SmartConnector.
- Solution 2: Send the messages directly to the SmartConnector using syslog UDP.
- Solution 3: Send the messages using syslog UDP to the local syslog daemon and configure the daemon to forward the messages to the SmartConnector.

To implement your preferred solution:

- Create the three perl scripts as shown in the "Create Three Perl Scripts" section.
- Follow the instructions in the "Configure AIX to Issue Audit Messages in Stream" section.

Important about Implementing Solutions

- The Solution 3 setup cannot be used with stock AIX syslogd. Instructions vary depending on syslog daemon installed. In this case, you have three choices: On the UDP receiving port, increase the UDP buffers available in the system to avoid message loss; use TCP to forward the log to the SmartConnector; or, configure a cache size on the TCP connection that provides guaranteed delivery.
- Only solutions 1 and 3 support keeping a local copy of the log on the issuing server. Solution 3 is not supported if you are using stock AIX syslogd.
- Refer to the Manage Prefixes section under Additional Configuration (after installing the core connector software) to learn how to manage the parsing of custom AIX-specific forwarding prefixes and remove the "forwarding message" phrase.
- It is important to note the difference between using UDP and Raw TCP. To ensure data integrity, Raw TCP should be used. For better performance, use UDP.
- The example scripts provided work, but cannot be guaranteed to function by HPE ArcSight on every implementation of an AIX Audit system.

Creating Three Perl Scripts

Create the following three perl scripts to implement the solutions that allow multiline messages to be changed and used by the IBM AIX Audit Syslog SmartConnector. For examples of these scripts, see ["Appendix - IBM AIX Audit Syslog Script Examples" on page 24](#).

1. Create a perl script named /etc/security/audit/streamcmds (or streamcmds.orig if you don't want to overwrite the default streamcmds file) as shown in the ["Script Example - streamcmds" on page 31](#).
2. Create a perl script named /etc/security/audit/perljoiner-syslogd as shown in the ["Script Example - perljoiner-syslogd" on page 24](#).
3. Create a perl script named /etc/security/audit/perljoiner-syslogr as shown in the ["Script Example - perljoiner-syslogr" on page 28](#).

Configuring AIX to Issue Audit Messages in Stream

1. Edit the `/etc/security/audit/config` file.

start:

```
binmode = off
streammode = on
```

stream:

```
streamcompact = off
cmds = /etc/security/audit/streamcmds
```

2. Install the SmartConnector.
3. Implement one of the three solutions mentioned earlier, as shown in the sections below.
4. (Optional) Remove the phrase message forwarded as shown in the "Additional Configuration" section.

Implement Solution 1

This solution allows you to send the messages to the local syslog daemon using UNIX socket and configure the daemon to forward the messages to the SmartConnector.

1. Go to the `/etc/security/audit/streamcmds` file.
2. Include the line that uses `perljoiner-syslogd` and comment out the line that uses `perljoiner-syslogr`.

```
#Use this to send to the local syslog daemon
/usr/sbin/auditstream | /usr/sbin/auditpr -v -t0 -h e,l,R,t,c,p,P | perl
/etc/security/audit/perljoiner-syslogd &
#Use this to send to a remote destination such as smartconnector
#/usr/sbin/auditstream | /usr/sbin/auditpr -v -t0 -h e,l,R,t,c,p,P | perl
/etc/security/audit/perljoiner-syslogr &
```

3. Modify the Syslog NG Daemon SmartConnector to redirect the audit log to the SmartConnector.



By default, logs will get to syslog daemon with facility-priority set as `local0.info` and with `pgmname` set as `"auditpr"`. `syslogd` only supports UDP implicit port 514, `rsyslog` and `syslog-ng` support TCP and UDP and a specific port can be supplied.

```
#To forward using udp with implicit port 514
local0.info @ipSmartconnector
#To forward using udp with specific port
local0.info @ipSmartconnector:port
```

#To forward using tcp with specific port
local0.info @@ipSmartconnector:port



If you use stock AIX syslogd, verify that the -n switch is not configured when the daemon is started. This switch instructs syslogd to add the hostname inside the syslog message. The hostname presence in the syslog message is required by the SmartConnector.

4. Edit /etc/security/audit/config/perljoiner-syslogd to adjust the following two lines to values for your environment:

```
my $maxretry=20;
my $delaybetweenretry=30;
```

Implement Solution 2

This solution allows you to send the messages directly to the SmartConnector using syslog UDP.

1. Go to the /etc/security/audit/streamcmds file.
2. Include the line that uses perljoiner-syslogr and comment out the line that uses perljoiner-syslogd

```
#Use this to send to the local syslog daemon
#/usr/sbin/auditstream | /usr/sbin/auditpr -v -t0 -h e,l,R,t,c,p,P | perl
/etc/security/audit/perljoiner-syslogd &
#Use this to send to a remote destination such as smartconnector
/usr/sbin/auditstream | /usr/sbin/auditpr -v -t0 -h e,l,R,t,c,p,P | perl
/etc/security/audit/perljoiner-syslogr &
```

3. Edit /etc/security/audit/perljoiner-syslogr to change the destination address to the following address and port:

```
SyslogHost => 'ipSmartconnector'
SyslogPort => 'port'
```

Implement Solution 3

This solution allows you to send the messages using syslog UDP to the local syslog daemon and configure the daemon to forward the messages to the SmartConnector.

1. Go to the /etc/security/audit/streamcmds file.
2. Include the line that uses perljoiner-syslogr and comment out the line that uses perljoiner-syslogd.

```
#Use this to send to the local syslog deamon
#/usr/sbin/auditstream | /usr/sbin/auditpr -v -t0 -h e,l,R,t,c,p,P | perl
```

```
/etc/security/audit/perljoiner-syslogd &
#Use this to send to a remote destination such as smartconnector
/usr/sbin/auditstream | /usr/sbin/auditpr -v -t0 -h e,l,R,t,c,p,P | perl
/etc/security/audit/perljoiner-syslogr &
```

3. Edit /etc/security/audit/perljoiner-syslogr to change the destination address to the following address and port:

```
SyslogHost => '127.0.0.1'
```

```
SyslogPort => '11514'
```

4. Modify syslog daemon to listen on UDP 11514 and redirect audit log to the SmartConnector.

Modifying the Daemon to Listen on UDP and Redirecting the Audit Log

Redirection instructions vary depending on your installed syslog daemon. The following are sample configurations for rsyslog and syslog-ng.

To avoid message loss on the AIX syslog daemon:

- Use TCP to forward the log to the SmartConnector instead of UDP.
- Increase the UDP buffers available on the UDP receiving port.
- Configure guaranteed delivery on the TCP connection to the SmartConnector.

Sample rsyslog configuration

```
# Provides UDP syslog reception on port 11514
$ModLoad imudp
$UDPServerRun 11514
#To forward using TCP
local0.info @@ipSmartconnector:port
```

Sample syslog-ng configuration

```
source s_udplocal { tcp(ip(127.0.0.1) port(11514) so_rcvbuf(2097152));
};
destination remote_smart { tcp("ipSmartconnector" port(port));
log { source(s_udplocal); destination(remote_smart); };
```

Configuring the Syslog SmartConnectors

Syslog Daemon SmartConnector

The Syslog Deamon SmartConnector is a syslogd-compatible daemon designed to work in operating systems that have no syslog daemon in their default configuration, such as Microsoft Windows. The SmartConnector for Syslog Daemon implements a UDP receiver on port 514 by

default, or can be configured on another port to receive syslog events. You can also configure to use the TCP protocol.

To use the SmartConnector for Syslog Daemon, add the following statement in the *rsyslog.conf* file:

```
*.* @@(remote/local-host-IP):514
```

Example: local1.warning @@10.0.0.1:514

- To read all Syslog events, use *.*
- To filter specific events, replace regex with the specific event name.
- For example: *.* @@(remote/local-host-IP):514 and local1.warning @@10.0.0.1:514.
- To send events over a TCP connection, use @@ and to send events over an UDP connection, use @.

If you are running SmartConnector for Syslog Daemon on the same machine as the server, you must provide the IP address of the local host. If you want to forward events to other machines, you must provide the IP address of the same.

Messages longer than 1024 bytes might be split into multiple messages on syslog daemon. No such restriction exists on syslog file or pipe.

Syslog Pipe and File SmartConnectors

When a syslog daemon is already in place and configured to receive syslog messages, an extra line in the syslog configuration file *rsyslog.conf* can be added to write the events to either a file or a system pipe and the ArcSight SmartConnector can be configured to read the events from it. In this scenario, the ArcSight SmartConnector runs on the same machine as the syslog daemon. The additional configurations for the ArcSight syslog file or syslog pipe SmartConnectors in the system where all Syslog Daemon SmartConnector configurations are done.

The Syslog Pipe SmartConnector is designed to work with an existing syslog daemon. This SmartConnector is especially useful when storage is a factor. In this case, syslogd is configured to write to a named pipe, and the Syslog Pipe SmartConnector reads from it to receive events.

The Syslog File SmartConnector is similar to the Pipe SmartConnector. However, this SmartConnector monitors events written to a syslog file such as *messages.log* rather than to a system pipe.

Using the SmartConnector for Syslog Pipe or File

This section provides information to set up your existing syslog infrastructure to send events to the ArcSight Syslog Pipe or File SmartConnector.

The standard UNIX implementation of a syslog daemon reads the configuration parameters from the `/etc/rsyslog.conf` file, which contains specific details about which events to write to files, write to pipes, or send to another host.

For Syslog Pipe:

1. Execute the following command to create a pipe:

```
mkfifo /var/tmp/syspipe
```

2. Add one of the following lines depending on your OS to the `/etc/rsyslog.conf` file:

```
*.debug /var/tmp/syspipe
```

or

```
*.debug | /var/tmp/syspipe
```

3. Restart the syslog daemon in one of the following methods:

Enter the following commands:

```
/etc/init.d/syslogd stop
/etc/init.d/syslogd start
```

or

Execute the following command to send a configuration restart signal:

On RedHat Linux:

```
service syslog restart
```

On Solaris:

```
kill -HUP `cat /var/run/syslog.pid`
```

For Syslog File:

1. Create a file or use the default file into which log messages must be written.
2. Modify the `/etc/rsyslog.conf` file

The syslog daemon is forced to reload the configuration and start writing to the pipe.

3. Restart the syslog daemon in one of the following methods:

- a. Restart the syslog daemon in one of the following methods:

Enter the following commands:

```
/etc/init.d/syslogd stop
/etc/init.d/syslogd start
```

or

Execute the following command to send a configuration restart signal:

On RedHat Linux:

```
service syslog restart
```

On Solaris:

```
kill -HUP `cat /var/run/syslog.pid`
```

Installing the SmartConnector

The following sections provide instructions for installing and configuring your selected SmartConnector.

The syslog SmartConnectors use a sub-connector architecture that lets them receive and process syslog events from multiple devices. There is a unique regular expression that identifies the device. For example, the same SmartConnector can process events from a Cisco Router and a NetScreen Firewall simultaneously. The SmartConnector inspects all incoming messages and automatically detects the type of device that originated the message.

You can install the syslog SmartConnector as a syslog daemon, pipe, or file connector. You can use the Syslog Deamon, Syslog Deamon NG, or Syslog File connector types depending on your requirement. The Syslog File type SmartConnectors also support Syslog Pipe.

Preparing to Install the Connector

Before you install any SmartConnectors, make sure that the OpenText ArcSight products with which the connectors will communicate have already been installed correctly (such as ArcSight ESM or ArcSight Logger).

For complete product information, refer to the *Administrator's Guide to ArcSight Platform*, available on [ArcSight Documentation](#).

If you are adding a connector to the ArcSight Management Center, see the *ArcSight Management Center Administrator's Guide* available on [ArcSight Documentation](#) for instructions.

Before installing the SmartConnector, ensure that you have the following:

- Local access to the machine where the SmartConnector is to be installed
- Administrator passwords

Installing and Configuring the SmartConnector

1. Start the installation wizard.
2. Follow the instructions in the wizard to install the core software.
3. Specify the relevant [Global Parameters](#), when prompted.
4. Do one of the following depending on your requirement:

- Select **Syslog Daemon** from the **Type** drop-down:
 - a. Click **Next** and specify the following parameters:

Parameter	Description
Network port	The SmartConnector for Syslog Daemon listens for syslog events from this port.
IP Address	The SmartConnector for Syslog Daemon listens for syslog events only from this IP address, apart from the default (ALL) to bind to all available IP addresses.
Protocol	Specify whether to read files in batch mode or real-time mode. In batch mode, all files are read from the beginning.
Forwarder	This option applies to Batch Mode only. Specify None , Rename , or Delete as the action to be performed to the file when the connector finishes reading and reaches end of file . For the real-time mode, retain the default value None .

- b. Click **Next**.
- Select **Syslog File** from the **Type** drop-down:

a. Click **Next**, and specify the following parameters:

Parameter	Description
Pipe Absolute Path Name	Specify an absolute path to the pipe, or accept the default value: <code>/var/tmp/syspipe</code> .
File Absolute Path Name	<p>Specify the full path name for the file from which this connector will read events. The following are default values:</p> <ul style="list-style-type: none"> • Solaris: <code>\var\adm\messages</code> • Linux: <code>\var\log\messages</code> <p>You can use a wildcard pattern in the file name.</p> <p>In the real-time mode, rotation can occur only if the file is over-written or removed from the folder. The real-time processing mode assumes the following external rotation:</p> <ul style="list-style-type: none"> • Date format log rotation: The device creates a new log at a specified time in the with the naming convention <code>filename.timestamp.log</code>. The connector detects the new log and terminates the reader thread to the previous log after the processing is complete. The connector then creates a new reader thread to the new <code>filename.timestamp.log</code> and begins processing that file. To enable this log rotation, specify timestamp in <code>yyyy-MM-dd</code> date format. For example, <code>filename.yyyy-MM-dd.log</code> • Index log rotation: The device writes to indexed files in the following format: <code>filename.log.001</code>, <code>filename.log.002</code>, <code>filename.log.003</code>, and so on. At startup, the connector processes the log with highest index. When the device creates a log with a greater index, the connector terminates the reader thread to the previous log after processing completes, creates a thread to the new log, and begins processing that log. To enable this log rotation, use an index format, as shown in the following example: <code>filename '%d,1,99,true'.log</code>; Specifying <code>true</code> indicates that the index can be skipped. For example, if 5 appears before 4, processing proceeds with 5 and will not read 4. Use of <code>true</code> is optional.
Reading Events Real Time or Batch	Specify whether to read files in batch mode or real-time mode. In batch mode, all files are read from the beginning.
Action Upon Reaching EOF	This option applies to Batch Mode only. Specify None , Rename , or Delete as the action to be performed to the file when the connector finishes reading and reaches end of file . For the real-time mode, retain the default value None .
File Extension If Rename Action	This option applies to Batch Mode only. Specify the extension to be added to the file name if the action on reaching the end of file is specified as Rename . The default value is Processed , which adds a <code>.processed</code> extension.

b. Click **Next**.

5. Select a [destination and configure parameters](#).
6. Specify a name for the connector.
7. (Conditional) If you have selected **ArcSight Manager** as the destination, the certificate import window for the ArcSight Manager is displayed. Select **Import the certificate to the connector from destination**, and then click **Next**. The certificate is imported and the **Add connector Summary** window is displayed.



Note: If you select Do not import the certificate to connector from destination, the connector installation will end.

8. Select whether you want to install the connector as a service or in the standalone mode.
9. Complete the installation.
10. [Run the SmartConnector](#).

For instructions about upgrading the connector or modifying parameters, see [Installation and User Guide for SmartConnector](#).

Additional Configuration

Manage Prefixes

You can manage parsing of custom AIX-specific forwarding prefixes by adding properties to the agent.properties file (located at \$ARCSIGHT_HOME\current\user\agent). These properties can be found in the agent.default.properties file (located at: \$ARCSIGHT_HOME\current\config\agent).

The following property controls whether custom AIX-specific forwarding prefixes and facility.priority portions of the headers are removed. This property is disabled (set to 'false') by default. To remove the "forwarding message" phrase, change the value to 'true', as shown below. Note that setting this value to 'true' may cause some performance degradation.

```
syslog.aix.enabled=true
```

The following property is used to strip out the prefix that AIX adds when it forwards a syslog message to another host:

```
syslog.aix.forwarded.prefixes=Message forwarded from,Forwarded from
syslog.aix.forwarded.prefixes.delimiter=,
```

Device Event Mapping to ArcSight Fields

The following section lists the mappings of ArcSight data fields to the device's specific event definitions. See the ArcSight Console User's Guide for more information about the ArcSight data fields.

IBM AIX Audit Event Mappings to ArcSight Fields

ArcSight ESM Field	Device-Specific Field
Agent (Connector) Severity	Medium = FAIL, Low = OK
Device Action	status
Device Custom Number 1	File Descriptor
Device Custom Number 2	Parent PID
Device Custom Number 3	Physical Volume Index
Device Custom String 1	ACL
Device Custom String 2	Group
Device Custom String 3	Owner
Device Custom String 4	Reason or Error Code
Device Custom String 5	PCL
Device Custom String 6	Volume Group ID
Device Event Class ID	event
Device Facility	facility
Device Process Name	processname
Device Product	'AIX Audit'
Device Receipt Time	time
Device Severity	oneOf(severity,status)
Device Vendor	'IBM'
Event Outcome	status
External ID	externalid
Message	message
Name	event

Source Process ID	process
Source Service Name	command
Source User Name	login

Appendix - IBM AIX Audit Syslog Script Examples

Script Example - perljoiner-syslogd

```
#!/usr/bin/perl -w
```

```
use strict;
```

```
#####
```

```
# POD DOCUMENTATION HEADER #
```

```
#####
```

```
=pod
```

```
=head1 NAME
```

```
perljoiner-syslogd - Perl joiner for AIX stream audit that redirect to local syslog daemon
```

```
=head1 DESCRIPTION
```

This script will join multiple aix auditpr lines into a single line as required to produce a valid syslog messages.

This version of the script will send the reassembled message to the local installed syslog daemon.

In order to avoid problems and crashes when the local syslog daemon is restarted,

as described in this IBM post: <http://www-01.ibm.com/support/docview.wss?uid=isg3T1011847>,

this script will

- Catch the error with eval when the syslog() system call is no longer available
- Retry a limited number of time to give the message to the local syslog daemon
- Wait for a delay between each attempts, leaving time for the local syslog daemon to be restarted

Messages will be dropped silently if syslog daemon is unavailable past the configured timeout.

To avoid filling the auditstream pipes, once a message is dropped by this script after all the retries,

every subsequent messages will only be attempted to be sent once, without retries.

=head1 PLATFORM

AIX

=head1 VERSION

1.2

=head1 USAGE

perljoiner-syslogd

=head1 AUTHOR

Adapted from original script found on Protect724 post #39629

=head1 REVISIONS

20160420 FMerchant Packaged to be presented at Protect 2016 from client contribution

=cut

#####

END OF POD DOCUMENTATION HEADER

#####

use POSIX qw(strftime);

#Set this to 1 to enable a debug log to /tmp/perljoiner.out

my \$debug=0;

#Set theses values for the max retry and the wait time

my \$maxretry=20;

my \$delaybetweenretry=30;

#Debug function to print an message with a timestamp prefix

sub mylogger

{

return unless (\$debug);

my (\$sec, \$min, \$hr, \$day, \$mon, \$year) = localtime;

printf TRACE "%02d/%02d/%04d %02d:%02d:%02d @_\\n",

```

$day, $mon + 1, 1900 + $year, $hr, $min, $sec;
}

#Initialize syslog output as unix sockets
sub myinitlog
{
setlogsock('unix');
mylogger("setlogsock RC=$?");
openlog('auditpr','', 'local0');
mylogger("openlog RC=$?");
}

#####

# MAIN

#####

use Sys::Syslog qw( :DEFAULT setlogsock);

#If debug is enabled, open trace file and redirect stdout to it
if ($debug) {
open(TRACE, ">/tmp/perljoiner.out") or die "Cannot open perljoiner.out $!";
select TRACE; $| =1; # Unbuffered output for TRACE
}

#Open unix socket to issue syslog messages
myinitlog();

#Main loop to reassemble and issue syslog messages
my $req = "";
my $retry=0;
while(1)
{
my $line=<STDIN> || die;

```

```

if($line =~ /^[A-Z]/){
$req = $line;
}elseif($line =~ /\s/){
chomp($req,$line);
do {
# eval will trap error like "die" if syslog is not available
eval { syslog('info', $req.$line); };
# If it fails, $@ will have the output of the "die" command issued by syslog
if ($@) {
++$retry;
if ($retry>=$maxretry) {
mylogger("Sleeping [$delaybetweenretry] seconds after syslog received error [$@]");
sleep($delaybetweenretry);
}
else {
mylogger("Too many retries [$retry], message lost [$req$line]");
}
myinitlog();
}
else {
# Everything was fine
mylogger("syslog(info, '$req$line')");
$retry=0;
}
} until ($retry>$maxretry or not $retry);
}
}
mylogger("End of perljoiner-syslogd");

```

```

closelog;

#####

# POD DOCUMENT FINAL SECTION #

#####

=pod

=head1 DOCUMENTATION

=cut

```

Script Example - perljoiner-syslogr

```

#!/usr/bin/perl -w

use strict;

#####

# POD DOCUMENTATION HEADER #

#####

=pod

=head1 NAME

```

perljoiner-syslogr - Perl joiner for AIX stream audit that send direct syslog messages to an destination address and port using UDP

=head1 DESCRIPTION

This script will join multiple aix auditpr lines into a single line as required to produce a valid syslog messages.

This version of the script will send the reassembled message directly to an ip address using the UDP protocol.

The syslog messages will not be given to the local syslog system using unix socket.

However, if your local syslog daemon is configured to receive on port 514, it is possible to specify the destination address as 127.0.0.1

Please note that stock aix syslogd does not support listening and forwarding.

Should you require to keep a local copy of the audit trail, either install syslog-ng, rsyslog, NxLogs or any other service

that provide listening for syslog messages and supports forwarding to remote system such as a SmartConnector.

Messages will be dropped silently if the destination is unavailable, whether it be a remote or local system.

Messages will also be dropped silently if the system encounter memory shortage, its UDP in all its simplicity.

This script requires Net Syslog as found on <http://search.cpan.org/~lhoward/Net-Syslog-0.04/Syslog.pm>

```
perl -MCPAN -e "install Net::Syslog"
```

```
=head1 PLATFORM
```

```
AIX
```

```
=head1 VERSION
```

```
1.3
```

```
=head1 USAGE
```

```
perljoiner-syslogr
```

```
=head1 AUTHOR
```

Adapted from original script found on Protect724 post #39629

```
=head1 REVISIONS
```

20160420 FMerchant Packaged to be presented at Protect 2016 from client contribution

```
=cut
```

```
#####
```

```
# END OF POD DOCUMENTATION HEADER #
```

```
#####
```

```
use POSIX qw(strftime);
```

```
use Net::Syslog;
```

```
#####
```

```
# USER DEFINABLE VARIABLES
```

```
#####
```

```
### The variables in this section are the only things that you should need to modify in the script
```

```
# Set this value for the destination port where to send the log
```

```

my $syslogPort ="514";

# Set this value for the destination IP address where to send the log
my $syslogHost = "127.0.0.1";

# Set this to 1 to enable a debug log to /tmp/perljoiner.out
my $debug = 0;

#####

# FUNCTIONS

#####

# Debug function to print an message with a timestamp prefix
sub mylogger {
    return unless ($debug);
    my ( $sec, $min, $hr, $day, $mon, $year ) = localtime;
    printf TRACE "%02d/%02d/%04d %02d:%02d:%02d @_\\n",
    $day, $mon + 1, 1900 + $year, $hr, $min, $sec;
}

#####

# MAIN

#####

# If debug is enabled, open trace file and redirect stdout to it
if ($debug) {
    open( TRACE,">/tmp/perljoiner.out" ) or die "Cannot open perljoiner.out $!";
    select TRACE; $| =1; # Unbuffered output for TRACE
}

# Initialize the syslog instance
my $syslog=new Net::Syslog(
    Name => 'auditpr',
    Facility => 'local0', # Facility and Severity are not used by the smartconnector when parsing
    messages
    Priority => 'informational',

```

```

SyslogPort => $syslogPort, # Set this value for the destination port where to send the log
SyslogHost => $syslogHost # Set this value for the destination IP address where to send the log
);
# Main loop to reassemble and issue syslog messages
my $req = "";
while(1) {
my $line = <STDIN> || die;
chomp( $line );
if($line =~ /^[A-Z]/){
$req = $line;
}
elsif( $line =~ /\s/ ){
$req .= $line;
mylogger( $req );
$req->send( $req, rfc3164 => 1 );
}
}
mylogger( "End of perljoiner-syslogr" );
#####
# POD DOCUMENT FINAL SECTION #
#####
=pod
=head1 DOCUMENTATION
=cut

```

Script Example - streamcmds

```

#!/usr/bin/ksh
#

```

```
#NAME
#
# streamcmds - format and send audit events as stream to a syslog handler
#
#DESCRIPTION
#
# Start auditstream and send event to a syslog handler
#
#PLATFORM
#
# AIX
#
#VERSION
#
# 1.0
#
#USAGE
#
# streamcmds
#
#AUTHOR
#
# Adapted from HPE ArcSight
#
#REVISIONS
#
# 20160420 FMerchant Packaged to be presented at Protect 2016 from client contribution
```



```
#####  
# END OF POD DOCUMENTATION HEADER #  
#####  
#####  
# MAIN  
#####  
#Use this to send to the local syslog daemon  
/usr/sbin/auditstream | /usr/sbin/auditpr -v -t0 -h e,l,R,t,c,p,P | perl  
/etc/security/audit/perljoiner-syslogd &  
#Use this to send to a remote destination such as smartconnector  
#/usr/sbin/auditstream | /usr/sbin/auditpr -v -t0 -h e,l,R,t,c,p,P | perl  
/etc/security/audit/perljoiner-syslogr &
```

Send Documentation Feedback

If you have comments about this document, you can [contact the documentation team](#) by email. If an email client is configured on this computer, click the link above and an email window opens with the following information in the subject line:

Feedback on Configuration Guide for IBM AIX Audit Syslog SmartConnector (SmartConnector CE 24.1)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to MFI-Documentation-Feedback@opentext.com.

We appreciate your feedback!