Overview

The Gateway service includes a powerful logging framework to enable you to control the logging of events.

This document contains information about logging, including the following key topics:

Loggers configuration files

The Gateway has two logger configuration files:

- **[GATEWAY_INSTALLDIR]/config/dev-logger.properties:** Used in development mode.
- **[GATEWAY_INSTALLDIR]/config/prod-logger.properties:** Used in production mode.

Where *[GATEWAY_INSTALLDIR]* is the root folder of the installation.

Separate files allow you to maintain two custom logger configurations, one for development and for production. To switch between the two logger configurations, simply set the *Netborder.run.mode* parameter in the [GATEWAY_INSTALLDIR]/config/*gw.properties* file to '*production*' or '*development*'.

Logging levels

There are six logging levels, as follows:

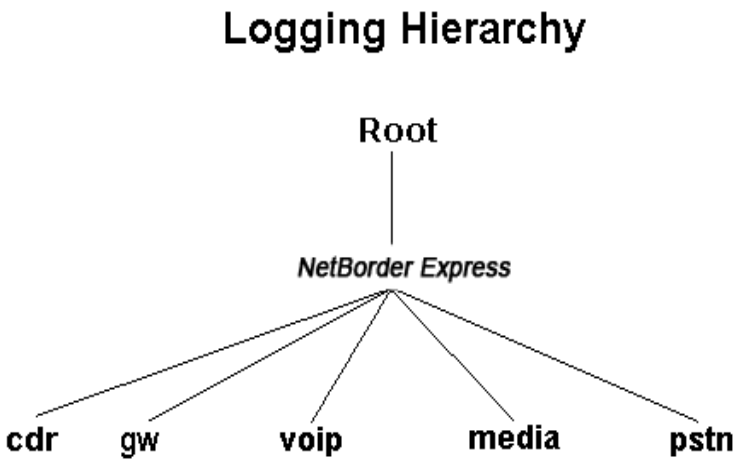
- **FATAL:** Logs very severe error events that may lead the application to abort.
- **ERROR:** Logs only error conditions. The ERROR level provides the smallest amount of logging information.
- **WARN:** Logs information when an operation completes successfully but there are issues with the operation.
- **INFO:** Logs information about workflow. It generally explains how an operation occurs.
- **DEBUG:** Logs all of the details related to a specific operation. This is the highest level of logging.
- **TRACE:** Logs designated finer-grained informational events than DEBUG.

> **CAUTION:** The performance of the system is significantly degraded when the log level is set to TRACE. DEBUG and TRACE should only be used by Technical support when troubleshooting a specific issue.

Logger hierarchy

The logger adheres to a hierarchical structure starting with a “Root” logger. Components that are direct “children” of a logger above them in the hierarchy inherit all of the properties of the parent. The value inherited by a child from the parent can by overridden by modifying the specific property of the child. A change at one level will impact all of the component’s descendants.

The figure below illustrates a subset of the Gateway’s logging hierarchy. Those components that are used solely for troubleshooting purposes by Technical Support have not been included (for example, all of the Gateway’s infrastructure components and utilities).



The “public” logging sub-modules of most importance are as follows:

- **PSTN:** Relates to everything that has to do exclusively with telephony signaling within the context of a PSTN call leg.
- **Media:** Relates to everything that has to do exclusively with media processing (RTP packetization, audio format, echo cancellation, DTMF relay, etc.).
- **VoIP:** Relates to everything that has to do exclusively with signaling within the context of a VoIP call leg.
- **GW:** Relates to everything that has to do with the general Gateway application and is independent of the specifics of the PSTN, Media, VoIP implementations. It includes, for example, the Call Engine and Routing Engine logs.
- **CDR:** Contains high-level information about a call (such as call start time, call end time, ANI, DNIS, etc.), and can be used to generate billing information.

Configuring the logging subsystem

To configure the logging subsystem, follow these steps:

Step 1: Set the logging level and appender

The first step to configuring the logging subsystem is to set the root logger. The root logger can be assigned a logging level and one or more formatting handles.

A *formatting handle*, also called an "*appender*", holds the information on where to redirect the logging output (for example, Windows Event Viewer, console, file, *syslog*, etc.), as well as the type and format of logging information to output.

Here is a sample configuration for the root logger. These parameters would be included in the file *[GATEWAY_HOME]/config/dev-loggers.properties* (where *[GATEWAY_HOME]* is the root folder of the installation).

```
# Logger configuration for the Windows Event log, level = INFO
log4cplus.rootLogger=INFO, NTEVENTLOG, ROLLINGFILE
# NTEVENTLOG Appender
log4cplus.appender.NTEVENTLOG=log4cplus::NTEventLogAppender
# ROLLINGFILE Config: Size limited rolling files of 50MB with 20 backups (max 1GB)
log4cplus.appender.ROLLINGFILE=log4cplus::RollingFileAppender
```

In the example provided, the root logging level is set to INFO, and the appender (formatting handle) is NTEVENTLOG. The appender is configured to redirect the output to the Windows Event Viewer using *NTEventLogAppender* and a *RollingFileAppender*.

To set the target redirection, assign the property *log4cplus.appender.<MY_HANDLE>* (where *<MY_HANDLE>* is the name of the logger's redirection handle) to one or more of the values listed in the following table.

Logging Appenders	
Appender	***Description***
log4cplus::NTEventLog Appender	Output redirected to the Windows Event Viewer.
log4cplus::ConsoleAppender	Output redirected to stdout (standard output). Applies only when the Gateway is started from a Console.
log4cplus::RollingFileAppender	Output redirected to a rolling file (rolling executed based on the log file size). If this type of appender is chosen, then the following properties can be configured: log4cplus.appender.<MY_HANDLE>.File=Gateway.log log4cplus.appender.<MY_HANDLE>.MaxFileSize=50MB log4cplus.appender.<MY_HANDLE>.MaxBackupIndex=20 log4cplus.appender.<MY_HANDLE>.ImmediateFlush=true
Log4cplus::DailyRollingFile Appender	Output redirected to a scheduled rolling file (rolling executed based on a schedule – hourly, daily, etc.). **There is no restriction on disk space for this appender, so use with care.** If this type of appender is chosen, then the following properties can be configured: log4cplus.appender.<MY_HANDLE>.File=Gateway.log log4cplus.appender.<MY_HANDLE>.Schedule=HOURLY The Schedule property can take one of the following values: MONTHLY, WEEKLY, DAILY, TWICE_DAILY, HOURLY, MINUTELY
log4cplus::FileAppender	Output redirected to a file (file is overridden each time the service starts). If this type of appender is chosen, then the following properties can be configured: log4cplus.appender.<MY_HANDLE>.File=Gateway.log log4cplus.appender.<MY_HANDLE>.MaxFileSize=50MB log4cplus.appender.<MY_HANDLE>.ImmediateFlush=true

Step 2: Set the pattern layout

After setting the redirection output, the next step is to configure the type and format of the information to appear in the logs. This is achieved through the configuration of a *Pattern Layout*. A pattern layout allows you to format the output of the logs in a similar fashion to a *printf* function in 'C'. The format string contains one or more placeholders, which will be replaced by the logging engine when it is time to log the message.

Here is an example of how a pattern layout would be assigned to a logger and configured. Again, these parameters are included in the file *[GATEWAY_HOME]\config\gw.properties* (where *[GATEWAY_HOME]* is the root folder of the installation).

```
log4cplus.appender.NTEVENTLOG.layout=log4cplus::PatternLayout

# Output 'Log Level' - 'Logger name' : Message
log4cplus.appender.NTEVENTLOG.layout.ConversionPattern=%p - %c : %m%n
```

In the example above, "%p" indicates the logging level or priority (such as "INFO"), "%c" indicates the source of event (such as a global configuration parameter or routing rule), followed by the log message itself, and finally a new line.

The following table lists the special conversion characters available for use within layout pattern strings. Note that each conversion qualifier starts with a percent sign (%).

Pattern Layout Conversion Characters	
Conversion Character	***Description***
%c	Category (name of logger) issuing the event.
%D	Date/time of the logging event. The date can be formatted using `%D[format]` where valid symbols for `[format]` are as follows: %Y: year, %m: month, %d: day, %H: hours, %M: minutes, %S: seconds, %%q: milliseconds.
%l	Location of the logging request (method, file name and line number). WARNING: This qualifier will severely impede performance!
%L	Line number of the logging request. WARNING: This qualifier will severely impede performance!
%m	Message of the log.
%n	New line.
%p	Priority of the logging event (logging level).

Pattern Layout Conversion Characters	
Conversion Character	***Description***
%r	Timestamp in milliseconds from the start of program until the creation of the logging event.
%t	Thread from which the logging request was made.

Step 3 (optional): Set child-specific behaviour

Logging configuration is performed at the root level, but any action can be overwritten by any "child" logger lower in the hierarchy. This way, you can assign a different logging level and logging format to a specific node in the logger hierarchy. Note that any log statement meeting the minimal logging level selected **will be forwarded to all appenders assigned to that logger** (directly or inherited from higher up in the logger hierarchy). For example, you might use this methodology to direct the log output to both the Windows Event Viewer *and* to a rolling file.

Dynamic call logging

The logger can be configured to redirect information about one particular call to a dedicated file. This "per call" information is actually **duplicated** from the current appenders configured for the Gateway and written to a unique file in the system. The call logger appender is in reality a dynamic appender, similar to a *log4cplus::FileAppender*, except that the output file name is determined dynamically based on the call ID.

To enable the call logger, make the following changes to the *gw.properties* file:

```
# Logger configuration for the Windows Event log, level = INFO
# Also attach the CALL_LOG_APPENDER to duplicate call-specific
# information to a unique file
log4cplus.rootLogger=INFO, NTEVENTLOG, CALL_LOG_APPENDER
log4cplus.appender.CALL_LOG_APPENDER.Directory=test-output
log4cplus.appender.CALL_LOG_APPENDER.ImmediateFlush=false
log4cplus.appender.CALL_LOG_APPENDER.layout=log4cplus::PatternLayout
log4cplus.appender.CALL_LOG_APPENDER.layout.ConversionPattern=%p - %c : %m%n
# Set to true if you want a directory structure with
# year/month/day/hour for your call logs
Netborder.infra.CallLogger.dateTimeDirectory=false
```

If *log4cplus.appender.CALL_LOG_APPENDER.Directory* is set to a valid output directory (with write permission), call logs are written to that directory. Otherwise, call logging is disabled. If, on the other hand, *Netborder.infra.CallLogger.dateTimeDirectory* is set to "*true*", a sub-directory structure with a year, month, day and hour hierarchy is created under the directory specified by the *Directory* property.

The *CALL_LOG_APPENDER* is different from other appenders because it cannot be attached to a logger at initialization time via the properties file. In fact, there is one *CALL_LOG_APPENDER* for each call that takes place, thus the terminology "dynamic appender". For this reason, only code that has been specifically instrumented for call logging can use this appender.

Currently, the following loggers use the *CALL_LOG_APPENDER*:

- **Netborder.cdr.pstn:** Call start/end, ANI and DNIS at *INFO_LOG_LEVEL*. Can be used for generating simple billing information.
- **Netborder.pstn:** All logging statements that take place between call start and call end can be duplicated to the call log, provided that call logging has been enabled and that the logging level of the *Netborder.pstn* logger is high enough. Used for troubleshooting purposes.

To enable all instrumented call logs to appear in the separate file, make sure the *CALL_LOG_APPENDER* is attached to the root logger. For CDR only, ensure that the *Netborder.cdr* logger has this appender enabled at 'INFO' level or more. See the example below.

```
# Logger configuration for the CDR logger, level = INFO
# Attach the CALL_LOG_APPENDER to duplicate call-specific
# high-level information to a unique file

log4cplus.logger.Netborder.cdr=INFO, CALL_LOG_APPENDER
```

Syslog integration

Logging to *syslog* involves adding the *REMOTE_SYSLOG_APPENDER* to the root logger and enabling network logging in *syslogd*.

Step 1: Add a Syslog appender

To use the remote syslog appender, amend the *gw.properties* file as follows:

```
log4cplus.rootLogger=WARN, REMOTE_SYSLOG_APPENDER
log4cplus.appender.REMOTE_SYSLOG_APPENDER=
                Netborder::RemoteSyslogAppender
log4cplus.appender.REMOTE_SYSLOG_APPENDER.layout=
                      log4cplus::PatternLayout
log4cplus.appender.REMOTE_SYSLOG_APPENDER.layout.ConversionPattern=%D{%H:%M:%S:%
%q} [%t] %p - %c : %m%n
log4cplus.appender.REMOTE_SYSLOG_APPENDER.Hostname=hostname
log4cplus.appender.REMOTE_SYSLOG_APPENDER.Port=514
log4cplus.appender.REMOTE_SYSLOG_APPENDER.Facility=8
```

The last two settings are optional and have default values of 514 and 8. 514 is the default syslog *Port*, and the *Facility* property represents the source of the message (8 stands for LOG_USER or random user-level messages).

Step 2: Enable network logging in syslogd

To enable network logging in *syslogd*, you must make certain it is launched with the `-r` option. This option will tell *syslogd* to accept logging messages from remote hosts.