

## DS7000/USCC – How the logging component works

### **Introduction**

USCC uses an external logging component called [Apache log4cxx](https://logging.apache.org/log4cxx/) (<https://logging.apache.org/log4cxx/>). It is ported from Java's log4j and is used also in well-known projects like *Apache HTTP Server* which the mostly used http server used for delivering websites to webbrowsers.

The *log4cxx* component supports different *log levels* and it can be configured which log messages of which log level should be logged. Further it supports different *appenders* which are some kind of targets where the log messages should be logged to (e.g. file, console, network, etc.). In USCC different *loggers* are used. Almost every component has its own logger which can be configured differently (e.g. different log levels, filters, files, etc.). In order to simplify configuration, log4cxx supports to build *logger hierarchies* which means that you can nest different loggers in a hierarchy (analog: directories and subdirectories). That means, you can configure a parent logger and all child loggers get the same configuration.

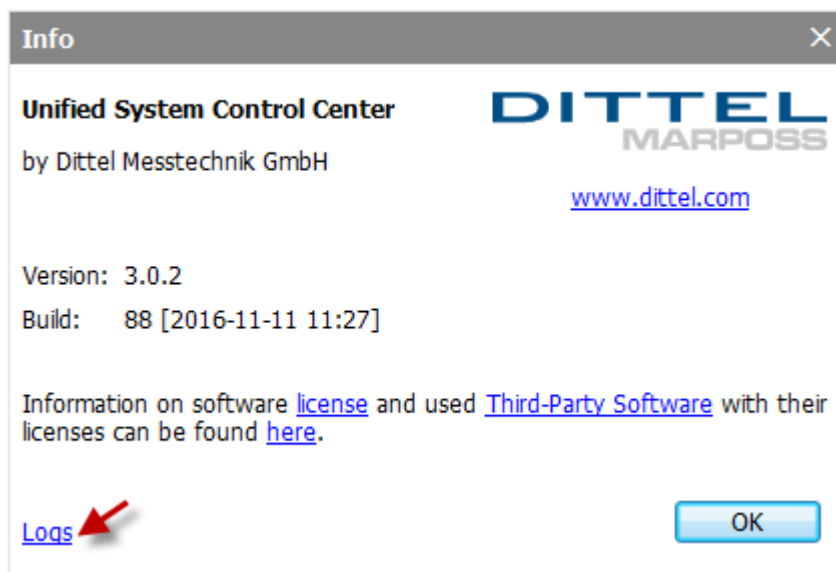
DS7000 devices send their log messages to the USCC instances which then log those messages (timely ordered) next to their own USCC log messages to pre-defined files by default. The next section explains how the default log configuration file looks like and how it is configured.

### **Default Configuration File**

The default configuration file is named *DittelLogConfig.xml* and is located in *settings* directory within the USCC installation directory.

In order to quickly find out the installation directory of a running USCC instance the info dialog can be opened. Exit *Widget Mode* by pressing the menu bar button *Main Menu* and then *Info*.





In the info dialog there's a link to the directory where the log messages are logged to. This directory is a subdirectory of the USCC installation directory.

### Unicode aware editor

For viewing and editing the log configuration file it is recommended to use an editor which supports Unicode, especially UTF-8 encodings and is able to auto-detect the file's encoding. On Windows, [Notepad++](https://notepad-plus-plus.org/) (<https://notepad-plus-plus.org/>) is a good candidate for an editor.

### Log levels

In the following you can find a description of log levels and what should be logged with which level.

Log Level	Description
FATAL	Severe errors that cause premature termination.
ERROR	Other run-time errors or unexpected conditions.
WARN	Run-time situations that are undesirable or unexpected, but not necessarily "wrong".
INFO	Interesting run-time events (start-up/shutdown).
DEBUG	Detailed information on the flow through the system.
TRACE	More detailed information.

## Appenders

### consoleAppender

```
<appender name="consoleAppender" class="org.apache.log4j.ConsoleAppender">
  <param name="Target" value="System.out"/>
  <!--<param name="Threshold" value="TRACE"/>-->
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d{HH:mm:ss,SSS} (%9r) [%30.30c] %5p - %m%n"/>
  </layout>
</appender>
```

The console appender logs to the console. It is relevant for console only applications or on Linux. This appender is not used by default.

### dittelConsoleAppender

```
<appender name="dittelConsoleAppender" class="DittelConsoleAppender">
  <param name="Target" value="System.out"/>
  <!--<param name="Threshold" value="TRACE"/>-->
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d{HH:mm:ss,SSS} (%9r) [%30.30c] %5p - %m%n"/>
  </layout>
</appender>
```

The dittel console appender is only useful for developers and is able to log to the development environment console.

### sessionAppender

```
<appender name="sessionAppender" class="org.apache.log4j.FileAppender">
  <param name="File" value="logs/USCC-session.log" />
  <param name="Append" value="false" />
  <!--<param name="Threshold" value="TRACE"/>-->
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d{yyyy-MM-dd HH:mm:ss,SSS} (%9r) [%10.10t, %x, %40.40c] %5p - %m%n" />
  </layout>
</appender>
```

The Session Appender logs to the file logs/USCC-session.log and its content is always erased when the application starts. That means if multiple application use the same default log configuration and e.g. USCC was started and a while later the Device Configurator, the Device Configurator would truncate the log file and all log messages of USCC logged before the start of the Configurator are lost in that file – but now both, USCC and the Device Configurator log to that same log file.

## fileSizeLimitedRollingFileAppender

```
<appender name="fileSizeLimitedRollingFileAppender" class="org.apache.log4j.RollingFileAppender">
  <param name="File" value="logs/USCC.log" />
  <param name="Append" value="true" />
  <param name="MaxBackupIndex" value="3" />
  <param name="MaxFileSize" value="50MB" />
  <!--param name="Threshold" value="TRACE"/-->
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d{yyyy-MM-dd HH:mm:ss,SSS} (%9r) [%10.10t, %x, %40.40c] %5p - %m%n" />
  </layout>
</appender>
```

This appender uses log rotation and logs to logs/USCC.log and backup log files logs/USCC.N.log. That means if a threshold was reached, the log file will be renamed and the current log file is empty until the next log message is logged. Further another threshold can tell the logger, how many log files should be kept.

For USCC the rotation threshold is reached when the log file size has reached 50 MB. Further there are at maximum 3 backup log files with 50 MB and the current log file with less than 50 MB. That means the log files of this appender are always less than 200 MB and no one needs to take care of cleaning up log files.

Further the log files do always append to the existing log file which means the log messages in the log file survive restarts of the USCC or other programs using the USCC log configuration file.

## monthlyErrorsNotGzipped

```
<appender name="monthlyErrorsNotGzipped" class="org.apache.log4j.DailyRollingFileAppender">
  <param name="File" value="logs/USCC-monthly-errors.log" />
  <param name="Append" value="true" />
  <param name="DatePattern" value=".yyyy-MM" />
  <param name="Threshold" value="WARN"/>
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d{yyyy-MM-dd HH:mm:ss,SSS} (%9r) [%10.10t, %x, %40.40c] %5p - %m%n" />
  </layout>
</appender>
```

This appender logs only logs messages that have warning, error or fatal level to the log file logs/USCC-monthly-errors.log. Further, it archives logs with a suffix .yyyy-MM which means there will be one log file full of warnings, errors and fatal errors per month. Having that long time error history can be useful to understand how long and what kind of problems are present. Of course there could also be some false positives like expected connection loss errors because of cable disconnects but other errors especially from the devices are useful information for debugging.

This appender is not active by default.

## monthlyErrorsGzipped

```
<appender name="monthlyErrorsGzipped" class="org.apache.log4j.rolling.RollingFileAppender">
  <param name="File" value="logs/USCC-monthly-errors.log" />
  <param name="Append" value="true" />
  <param name="Threshold" value="WARN"/>
  <rollingPolicy class="org.apache.log4j.rolling.TimeBasedRollingPolicy">
    <param name="FileNamePattern" value="logs/USCC-monthly-errors-%d{yyyy-MM}.log.gz"/>
  </rollingPolicy>
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%d{HH:mm:ss,SSS} %-6r [%15.15t] %-5p %40.40c %x - %m%n" />
  </layout>
</appender>
```

This appender is equal to the monthlyErrorsNotGzipped appender except that archived log files are also compressed with gzip in order to save disk space. This appender is active by default.

## Loggers

```
<root>
  <priority value="INFO" />
  <!--<appender-ref ref="consoleAppender"/>-->
  <appender-ref ref="dittelConsoleAppender"/>
  <appender-ref ref="sessionAppender"/>
  <appender-ref ref="fileSizeLimitedRollingFileAppender"/>
  <appender-ref ref="monthlyErrorsGzipped"/>
</root>

<category name="dittel.Application.DeviceLogger" additivity="true">
  <priority value="INFO" />
</category>
```

By default there are only two loggers configured. The root logger and the logger which logs log messages received from DS7000 devices.

Due to the logger hierarchy, the settings of a logger higher in the hierarchy also apply to loggers lower in the hierarchy. In the default configuration, the root logger only logs messages with log level INFO and the appenders dittelConsoleAppender, sessionAppender, fileSizeLimitedRollingFileAppender, monthlyErrorsGzipped are used. This configuration applies to all other loggers by default because of the log hierarchy.

Further there is a configuration node in the xml file that defines the device logger which logs messages received from the device. It is called dittel.Application.DeviceLogger. So it is quickly possible to change the log level threshold of device log messages to debug or trace for debugging purpose.

Dots define hierarchy levels. That means that Dittel inherits from the root logger, dittel.Application logger inherits its configuration from the dittel logger and

dittel.Application.DeviceLogger inherits its configuration from dittel.Application.

For example when you like to see all log messages with at least debug level from loggers which names begin with dittel.Application you can define a new xml logger configuration node by copy and pasting the existing device logger node and adjusting its name and level like it was done in the following screenshot.

```
<root>
  <priority value="WARN" />
  <!--<appender-ref ref="consoleAppender"/>-->
  <appender-ref ref="dittelConsoleAppender"/>
  <appender-ref ref="sessionAppender"/>
  <appender-ref ref="fileSizeLimitedRollingFileAppender"/>
  <appender-ref ref="monthlyErrorsGzipped"/>
</root>

<category name="dittel.Application" additivity="true">
  <priority value="DEBUG" />
</category>

<category name="dittel.Application.DeviceLogger" additivity="true">
  <priority value="INFO" />
</category>
```

In this example, by default only warning and above are logged but for loggers starting with dittel.Application we also log messages with DEBUG level and above. Further, we are only interested in messages from DS7000 devices that have INFO level and above which means we restrict the DEBUG level again to INFO and above for that more specific logger. Other loggers like dittel.Application.Settings are still logging with DEBUG however because they still inherit DEBUG level configuration from dittel.Application logger.

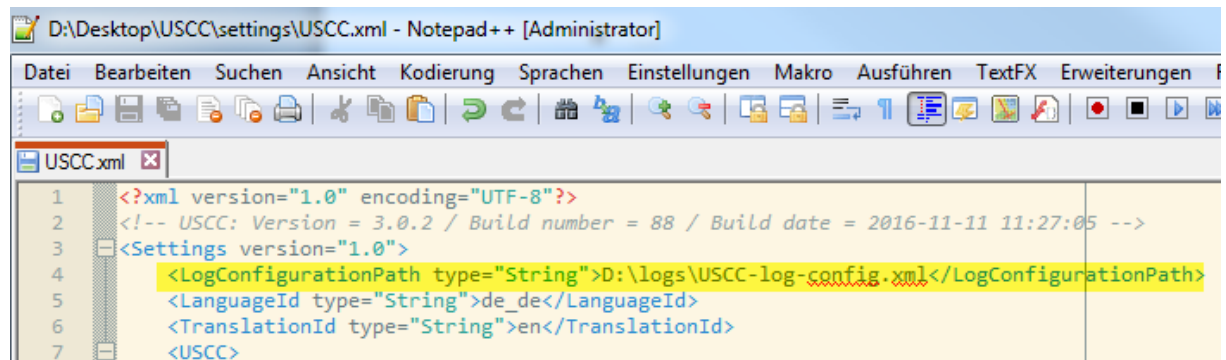
## Use cases

### How to use alternative log configuration files?

In the USCC settings file it is possible to provide a path to another log configuration file with the configuration key LogConfigurationPath. An example is provided in the screenshot below. You could for example copy the default log configuration file, make your adjustments and then link to the new log configuration file in the USCC settings. Due to the fact that USCC can be started with different scopes having their own configuration files, it is possible to define an alternative log configuration file in

each scope configuration file (e.g. embedded ActiveX control uses another configuration file than the USCC.exe standalone application).

```
<LogConfigurationPath type="String">
    D:\The\Path\To\TheAlternativeLogConfigFile.xml
</LogConfigurationPath>
```



## How to change log levels in general?

Open the log configuration file in an [editor that is Unicode aware](#).

Scroll down to the bottom of the file until you see the root.



Now you can edit the priority nodes of the loggers to the log level threshold you like to use. For example if you like to log everything with DEBUG level, change the priority node value from INFO to DEBUG in the root logger.

Due to the fact, that loggers are structured in a hierarchy and loggers deeper in the hierarchy inherit configuration from their parents all loggers will get the root log level priority set except those loggers that overwrite that level. In the screenshot above you either need to remove the `dittel.Application.DeviceLogger` XML node or also set its log level priority to DEBUG. Otherwise everything would be logged in debug level except the `dittel.Application.DeviceLogger`.

In case that the device running the USCC application is not powerful enough, it is recommended to only increase the log level for pre-defined loggers. Due to the fact that there are a lot of different loggers the best configuration for specific problems needs to be asked and a log configuration file for that issue will be provided.

In the following there are however some pre-defined solutions for general problems.

### What needs to be set in case of connection problems?

In case of connection problems it recommended to set the following loggers to debug or even trace level:

- `dittel.DeviceFinder`
- `dittel.DS7000Protocol`
- `dittel.Application`

In case that the device running USCC has performance problems after increasing the log level of those components, remove the `dittel.Application` node from log configuration and define sub-loggers of `dittel.Application` instead:

- `dittel.Application.Application`
- `dittel.Application.BalancingApplication`
- `dittel.Application.DataMonitoringApplication`
- `dittel.Application.DataMonitoringModule`
- `dittel.Application.DeviceLogger`

### What to set in case of problems on device side?

In case of problems on the device side, increase the log level for `dittel.Application.DeviceLogger` to DEBUG or even TRACE.

```
<root>
  <priority value="INFO" />
  <!--<appender-ref ref="consoleAppender"/>-->
  <appender-ref ref="dittelConsoleAppender"/>
  <appender-ref ref="sessionAppender"/>
  <appender-ref ref="fileSizeLimitedRollingFileAppender"/>
  <appender-ref ref="monthlyErrorsGzipped"/>
</root>

<category name="dittel.Application.DeviceLogger" additivity="true">
  <priority value="TRACE" />
</category>
```