

## CHAPTER

## 1

## The SAS Logging Facility

---

<i>Accessibility Features of the SAS Logging Facility</i>	3
<i>Overview of the SAS Logging Facility</i>	4
<i>What Is the Logging Facility?</i>	4
<i>Who Uses the Logging Facility?</i>	4
<i>Comparing the SAS Logging Facility and the SAS Log</i>	4
<i>Logging Facility Terminology</i>	5
<i>How the Logging Facility Works</i>	6
<i>Setting Up the Logging Process</i>	6
<i>The Logging Process</i>	6
<i>Loggers</i>	7
<i>What Is a Logger?</i>	7
<i>XML Elements for Configuring Loggers</i>	7
<i>Hierarchical Logger Names</i>	8
<i>SAS Server Logger Names</i>	9
<i>Loggers in the SAS Language</i>	10
<i>Appenders</i>	10
<i>Appender Overview</i>	10
<i>XML Elements for Configuring Appenders</i>	11
<i>General Appender Syntax</i>	11
<i>SAS Appenders for Server Logging</i>	13
<i>Appenders in the SAS Language</i>	13
<i>Referencing Appenders in a Logger</i>	14
<i>Logging Thresholds</i>	14
<i>Formatting Messages</i>	15
<i>Message Filtering</i>	16
<i>Using the SAS Logging Facility in the SAS Intelligence Platform</i>	17
<i>About the Initial Logging Configuration for SAS Servers</i>	17
<i>Viewing SAS Logging Messages and Adjusting Logging Levels in Client Applications</i>	17
<i>Best Practices for SAS Server Logging</i>	18

---

### Accessibility Features of the SAS Logging Facility

For information about accessibility for any of the products mentioned in this book, see the online Help for that product.

If you have questions or concerns about the accessibility of SAS products, send e-mail to [accessibility@sas.com](mailto:accessibility@sas.com).

---

## Overview of the SAS Logging Facility

---

### What Is the Logging Facility?

The SAS 9.2 logging facility is a flexible, configurable framework that you can use to collect, categorize, and filter events and write them to a variety of output devices. The logging facility supports problem diagnosis and resolution, performance and capacity management, and auditing and regulatory compliance. The logging facility has the following features:

- Log events are categorized using a hierarchical naming system that enables you to configure logging at a broad or a fine-grained level.
- Log events can be directed to multiple output destinations, including files, operating system facilities, and client applications. For each output destination, you can specify the following logging facility components:
  - the categories and levels of log events to report
  - the message layout, including the types of data to be included, the order of the data, and the format of the data
  - filters based on criteria such as diagnostic levels and message content
- Logging levels can be adjusted dynamically without starting and stopping processes.
- Performance-related log events can be generated for processing by the Application Response Measurement (ARM) 4.0 server.

The logging facility is used by most SAS server processes. You can also use the logging facility within SAS programs.

---

### Who Uses the Logging Facility?

This guide is for both administrators, who configure the SAS logging facility, and for programmers, who can use the logging facility in their SAS programs.

---

### Comparing the SAS Logging Facility and the SAS Log

The SAS logging facility and the SAS log are two different logging systems within SAS.

Traditionally, the SAS log displays information, warning, and error messages as a result of executing SAS programs or SAS global statements. Regardless of their origin, all messages are destined for a single log.

By contrast, the SAS logging facility is a framework that categorizes and filters log messages in SAS server and SAS programming environments, and writes log messages to various output devices. In the server environment, the logging facility logs messages based on predefined message categories, such as Admin for administrative messages, App for application messages, and Perf for performance messages. Messages for a category can be written to files, consoles, and other system destinations simultaneously. The logging facility also enables messages to be filtered based on the following thresholds: TRACE, DEBUG, INFO, WARN, ERROR, and FATAL.

In the programming environment, if the logging facility is initialized for SAS server logging, messages are written to logging facility destinations only. If the logging facility is not initialized for SAS server logging, messages are written not only to the SAS log, but also to logging facility destinations that are created in a SAS program.

---

## Logging Facility Terminology

Here are the common terms that this document uses:

### appender

a named entity that represents a specific output destination for messages.

Destinations include fixed files, rolling files, operating system facilities, and client applications. You can configure appenders by specifying thresholds, filters, log directories and filenames, pattern layouts, and other parameters that control how messages are written to the destination.

### filter

a set of character strings or thresholds, or a combination of strings and thresholds that you specify. Log events are compared to the filter to determine whether they should be processed.

### level

the diagnostic level that is associated with a log event. The levels, from lowest to highest, are TRACE, DEBUG, INFO, WARN, ERROR, and FATAL.

### log event

an occurrence that is reported by a program for possible inclusion in a log.

### logger

a named entity that identifies a message category. Loggers are named using a hierarchical system that enables you to configure logging at a broad or a fine-grained level.

The logging facility includes a set of high-level loggers for SAS servers, including Audit, Admin, App, IOM, and Perf. Some loggers are subdivided into lower-level (child) loggers. For example, the Audit logger has descendant loggers called Audit.Meta and Audit.Authentication, and Audit.Meta has descendant loggers called Audit.Meta.Security and Audit.Meta.Mgmt. The Root logger is the highest-level logger and does not represent a specific message category.

Loggers inherit settings from their higher-level (ancestor) loggers.

### logging configuration

an XML file or a set of SAS program statements that determines how log events are processed. You use the logging configuration to assign thresholds to loggers, to configure appenders, and to specify which categories and levels of log events are to be written to each appender.

If you perform a planned deployment, then the SAS Deployment Wizard provides default logging configuration files for your SAS servers.

### pattern layout

a template that you create to format messages. The pattern layout identifies the types of data, the order of the data, and the format of the data that is generated in a log event and is delivered as output.

### threshold

the lowest event level that is processed. Log events whose levels are below the threshold are ignored.

---

## How the Logging Facility Works

---

### Setting Up the Logging Process

To use the SAS logging facility, you must set up your logging environment:

- Define a logging configuration, which configures appenders and loggers. You can define the configuration by setting up an XML file or by using SAS language elements. If you perform a planned deployment, then logging configuration files are provided for your SAS servers.
- Specify the LOGCONFIGLOC= system option to enable logging, if you are using configuration files. If you perform a planned deployment, then this system option is included in the SAS configuration files for your SAS servers.
- Issue log events in a format that can be processed by the logging facility, if you are developing your own SAS programs.

---

### The Logging Process

After your logging environment is in place, the SAS logging facility begins processing as follows:

- 1 A SAS process (for example, a SAS server process) issues a log event. Each event includes the following attributes: a name that indicates the message category, a diagnostic level, and a message that describes the context for the event.
- 2 The logging facility receives the log event and determines which logger to assign it to, based on the event's name attribute.
- 3 The log event's level is compared to the threshold that is specified for the logger in the logging configuration. If the event's level is at or above the specified threshold, then processing continues. If the level is below the threshold, then the event is ignored.

If no threshold is specified for the event's logger, then the event inherits the threshold setting of the nearest ancestor logger. For example, if an Audit.Meta.Security event is being processed, then inheritance occurs as follows:

- a The event's level is compared to the threshold for the Audit.Meta.Security logger.
- b If no threshold is specified for Audit.Meta.Security, then the threshold for Audit.Meta is applied.
- c If no threshold is specified for Audit.Meta, then the threshold for Audit is applied.
- d If no threshold is specified for Audit, then the threshold for Root is applied.

If no thresholds are assigned to the logger or its ancestors, then the event is ignored.

- 4 The log event is processed by the appenders that are assigned to the logger and any of its ancestors in the logging configuration. For example, an Audit.Meta.Security event is processed by the appenders that are assigned to the following loggers: Audit.Meta.Security, Audit.Meta, Audit, and Root.

Each of these appenders processes the event according to the appender's configuration as specified in the logging configuration. Appender processing is performed as follows:

- a If the appender configuration includes a threshold, then the event's level is compared to the threshold. If the event's level is at or above the threshold,

- then processing continues. If the level is below the threshold, then processing stops.
- b If the appender configuration includes a filter, then the event is compared to the filtering criteria. Processing either continues or stops depending on the results of the comparison.
  - c The event is written to the output destination using the specifications that are defined in the appender configuration. Appender specifications include parameters such as pattern layouts, log directories, log filenames, rolling policies, locales, and encoding.

---

## Loggers

---

### What Is a Logger?

A logger is a named entity that identifies a message category. A logger's attributes consist of a level and one or more appenders that process the log events for the message category. The level indicates the threshold, or lowest event level, that will be processed for this message category.

Loggers are specified in log events to associate the log event with a message category. By categorizing log events, the logger can write messages of the same category to the same destinations. When a log event occurs, the log event message is processed by the appender that is associated with the logger that is named in the event log if the log event level is the same or higher than the level that is specified for the logger.

Loggers are organized hierarchically and inherit the attributes of their ancestor logger. Hierarchical logger names are separated by a period (.) (for example, Admin.Meta.Security). The root logger is the highest level logger. All loggers inherit the root logger's attributes. The logging configuration file defines several message categories that are immediate descendants of the root logger. These high-level categories, Admin, App, Audit, IOM, and Perf, are used for SAS server logging and can be referenced by log events in SAS programs.

You configure loggers in a logging configuration file for SAS server logging or by using SAS language elements in a DATA step or macro program. If you perform a planned deployment, then the SAS Deployment Wizard provides logging configuration files for your SAS servers. You can dynamically adjust thresholds by using the server management features of SAS Management Console. For more information, see "Administering Logging for SAS Servers" in the *SAS Intelligence Platform: System Administration Guide*.

For more information, see "Logging Thresholds" on page 14 and "Appendors" on page 10.

---

### XML Elements for Configuring Loggers

In a logging configuration file, a logger has the following structure:

```
<logger name="logger-name">
  <level value=threshold/>
  <appender-ref ref="appender-name"/>
</logger>
```

**Syntax Description:**

`name="logger-name"`

specifies the name of a message category name. The logger name is specified in a log event to associate a message with a message category.

`<level value="threshold"/>`

specifies one of the following levels, from lowest to highest: TRACE, DEBUG, INFO, WARN, ERROR, FATAL. You use the threshold to filter log events. If the log event diagnostic level is the same or higher than the threshold that is specified for the log event's logger, the logging facility continues to process the log event. If the log event diagnostic level is lower than the logger's threshold, the log event is ignored.

`<appender-ref ref="appender-name"/>`

specifies the name of an appender to record messages for this logger's message category.

## Hierarchical Logger Names

The logger architecture enables logger names to be multiple levels so that descendant loggers can inherit thresholds and appender references from their parent loggers, therefore omitting the appender reference and threshold in the descendant logger definition. You separate hierarchical logger names with a period (.).

As an example, suppose that your logging facility configuration file defines the Admin logger with an appender reference value of **MyRollingFile** and a threshold of **Info**. A second logger definition, Admin.MyPgm, specifies the logger name and a threshold of **Debug**. Because no appender reference is specified in Admin.MyPgm, the appender reference is inherited from its parent, the Admin logger. The appender reference **MyRollingFile** logs messages for Admin log events whose level is INFO or higher, as well as Admin.MyPgm log events whose level is DEBUG or higher.

These loggers might be defined using the following logger elements in the logging configuration file:

```
<logger name="Admin">
  <level value="Info"/>
  <appender-ref ref="MyRollingFile"/>
</logger>

<logger name="Admin.MyPgm">
  <level value="Debug"/>
</logger>

<root>
  <level value="Error"/>
  <appender-ref ref="SystemRollingFile">
</root>
```

If a log event specifies a hierarchical logger name that does not exist, the logging facility checks for a parent logger definition. If the parent logger exists, the log event is processed by the parent logger. If a logger definition does not exist for the parent, the root logger processes the log event.

Consider the example logger definitions in this section. If a log event specifies the logger Admin.Special, the logging facility determines that the logger Admin.Special does not exist. The logging facility then checks for the Admin logger. In this case, the Admin logger exists and the log event is processed by the Admin logger. If the Admin logger was not defined, the root logger would process the log event.

---

## SAS Server Logger Names

Log events for SAS servers use a hierarchical logger name where each name in the hierarchy identifies a category such as an operation, a server, and a server operation. For example, log events that specify the Admin.OLAP.Security logger indicate that the message is an OLAP server security message that is intended for a system administrator or computer operator.

SAS server logger names begin with one of the following logger categories:

### Admin

processes log events that are relevant to system administrators or computer operators.

### App

processes log events that are related to specific applications. For example, metadata servers, OLAP servers, stored process servers, and workspace servers use loggers that are named *App.class.interface.method* to record method calls that are issued to the server.

### Audit

processes log events that are related to user authentication (including accepted and rejected authentication requests) and to security administration (including the administration of users, groups, and access controls).

### IOM

processes log events for servers that use the Integrated Object Model (IOM) workspace interface. The IOM interface provides access to Foundation SAS features such as the SAS language, SAS libraries, the server file system, results content, and formatting services. IOM servers include metadata servers, OLAP servers, stored process servers, and workspace servers.

### Perf

processes log events that are related to system performance.

The second category in a hierarchical logger name can indicate a type of server or some type of event, such as authentication. In most cases, however, the categories are self-explanatory. The following list gives some examples of server categories for the logging facility.

Logging Facility Server Category	SAS Server
Connect	SAS/CONNECT Server
Meta	SAS Metadata Server
ObjectSpawner	SAS Object Spawner
OLAP	SAS OLAP Server
Table	SAS Table Server

In most cases, the categories are self-explanatory. Here is a list of some of the loggers that the logging facility uses for SAS servers:

### Admin.Operations

processes log events that are related to server operations, such as starting, pausing, and stopping an instance of a workspace server.

**Admin.Session**

processes log events that are related to starting and stopping batch, windowing, and SAS/CONNECT server sessions.

**Audit.Authentication**

processes log events for server authentication requests.

**App.Program**

processes log events that are related to running a program using the SAS language.

**IOM**

processes log events that are related to client interactions.

**IOM.PE**

processes log events that are related to packets that are processed by the BRIDGE and COM protocol engines.

**Perf.ARM**

processes log events that are related to ARM 4.0 transactions.

---

## Loggers in the SAS Language

You create loggers in SAS programs by using the following SAS language elements:

- %log4sas\_logger( ) autocall macro for macro programming
- log4sas\_logger function in a DATA step
- DCL Logger object constructor in a DATA step

See the following reference documents for information about defining loggers in the SAS language:

- “%LOG4SAS\_LOGGER Autocall Macro” on page 99
- “LOG4SAS\_LOGGER Function” on page 113
- “DECLARE Statement, Logger Object” on page 128

If you are writing SAS programs, you can write log events for loggers that are defined in one of the logging configuration files or you can write log events for loggers that you create by using the SAS language.

Loggers that are created by using the SAS language exist for the duration of the SAS session.

---

## Appenders

---

### Appender Overview

An appender is a named entity that is referenced by a logger. An appender specifies the destination for the message, specifies how the message is formatted, specifies attributes for the appender class, and provides additional filtering capabilities.

When a log event occurs, the logging facility processes the message by using the appender that is named in the logger’s <appender-ref> element in a logging facility configuration file, or in the APPENDER-REF argument of a logger language element in a SAS program.

SAS has several appender classes for processing messages:

- appenders to log messages to an operating system console



- an IOM server appender to log messages from any IOM server
- file appenders for writing log messages to a file on disk
- appenders to write to Windows, UNIX, and z/OS operating system logs

For a complete list and description of the SAS server appenders, see “SAS Appenders for Server Logging” on page 13.

You define appenders in the logging configuration file or in a SAS program by using a SAS function, autocall macro, or DATA step component object. An appender definition requires an appender class and name and the required parameters for the appender class. To customize the message, you specify the message layout within the appender definition. In a logging facility configuration file, you can include additional filtering arguments in the appender definition.

Logger definitions in SAS programs can reference appenders that are defined in a SAS program or any of the SAS server appenders.

For more information, see Chapter 5, “Appender Reference,” on page 33 and “Creating and Using Appenders in a SAS Program” on page 88.

---

## XML Elements for Configuring Appenders

### General Appender Syntax

In a logging configuration file, the appender has the following general structure:

```
<appender class="appender-class" name="appender-name">
  [ <param name="parameter-name" value="parameter-value"/>-1
    ... <param name="parameter-name" value="parameter-value"/>-n ]
  [ <layout>
    <param name="Header" value="header-text"/>
    <param name="HeaderPattern" value="conversion-pattern"/>
    <param name="ConversionPattern" value="conversion-pattern"/>
    <param name="Footer" value="footer-text"/>
    <param name="FooterPattern" value="conversion-pattern"/>
    <param name="XMLEscape" value="TRUE | FALSE"/>
  </layout> ]
  [ <filter>
    <filter-definitions>
  </filter> ]
</appender>
```

The brackets ([ ]) indicate that the element is optional.

Syntax Description:

`class="appender-class"`

The appender class is a type of appender. The following appender classes can be used in the logging facility:

- ARMAppender
- ConsoleAppender
- FileAppender
- IOMServerAppender
- RollingFileAppender
- sLogAppender

- UNXFacilityAppender
- WindowsEventAppender
- ZOSFacilityAppender
- ZOSWtoAppender

For more information about logging facility appenders, see Chapter 5, “Appender Reference,” on page 33.

`name="appender-name"`

The appender name is a user-specified name for the appender. An appender is associated with a logger when the appender name is specified as the value of the logger’s `appender-ref` attribute.

`<param name="parameter-name" value="parameter-value"/>`

Most appenders have required parameters for specifying information that is relevant to the appender. Many parameter names are unique for individual appenders. The `name=` attribute specifies the name of the parameter, and the `value=` attribute specifies the value for the parameter.

For example, appenders that write messages to a user-specified file use the `File` parameter, where the value of the `File` parameter specifies the location and name of the log file.

See Chapter 5, “Appender Reference,” on page 33 for each appender’s required parameters.

`<layout>`

`<param name="ConversionPattern" value="conversion-pattern"/>`

`<param name="Header" value="literal-string"/>`

`<param name="HeaderPattern" value="conversion-pattern"/>`

`<param name="Footer" value="literal-string"/>`

`<param name="FooterPattern" value="conversion-pattern"/>`

`<param name="XMLEscape" value="true | false"/>`

`</layout>`

You use the `<layout>` elements to specify how messages should be formatted in the log. The conversion pattern is a series of conversion characters that represent the types of data to include in the log. For example, use the conversion characters `%d %t %m` to include the date, the time, and the message, respectively, in the log.

You can use the `Header`, `HeaderPattern`, `Footer`, and `FooterPattern` parameters to specify the conversion characters that should appear at the top and the bottom of the log. You can use the `XMLEscape` parameter to specify whether certain characters (for example, `"<"`) is converted to its entity representation, which in this case would be `"&lt;"`.

For more information, see “Formatting Messages” on page 15.

`<filter>`

`<filter-definitions>`

`</filter>`

You can use filters to accept or deny messages based on the following:

- a character string in the message
- a range of message thresholds
- a single message threshold
- a combination of character string, single message threshold, or a range of message thresholds

For more information, see Chapter 7, “Filters,” on page 75.

---

## SAS Appenders for Server Logging

The following appenders can be configured as the value of the <appender> "class" attribute in the XML configuration files for SAS servers:

### ARMAppender

ARMAppender processes all Application Response Measurement (ARM) messages that are submitted by an external ARM agent or by the SAS ARM agent. See "ARMAppender" on page 34.

### ConsoleAppender

ConsoleAppender writes messages to the UNIX and Windows operating system consoles. See "ConsoleAppender" on page 38.

### FileAppender

FileAppender writes messages to the specified file in the specified path. See "FileAppender" on page 39.

### IOMServerAppender

IOMServerAppender commits messages from any IOM server to a volatile runtime cache. See "IOMServerAppender" on page 43.

### RollingFileAppender

RollingFileAppender writes messages to the specified file in the specified path, and begins writing messages to a new file that has a different name when specified criteria are met. See "RollingFileAppender" on page 45.

### sLogAppender

sLogAppender is a reserved appender. You should not define new instances of this appender. See "sLogAppender" on page 53.

### UNXFacilityAppender

UNXFacilityAppender writes messages to the syslogd logging facility in UNIX operating systems. See "UNXFacilityAppender" on page 54.

### WindowsEventAppender

WindowsEventAppender writes messages to the Windows Event log. See "WindowsEventAppender" on page 56.

### ZOSFacilityAppender

ZOSFacilityAppender enables multiple instances of SAS in the z/OS operating system to write messages to a common location. See "ZOSFacilityAppender" on page 57.

### ZOSWtoAppender

ZOSWtoAppender directs SAS application messages to the z/OS operating system console. See "ZOSWtoAppender" on page 60.

---

## Appenders in the SAS Language

When you specify an appender reference in a logger language element, you can use any of the appenders that are defined for SAS server logging and the appender FileRefAppender.

FileRefAppender is an appender that you create only in the SAS language, and only by using a SAS function, DATA step object, or autocall macro. As the name indicates, FileRefAppender names a fileref that defines a location to store messages. FileRefAppender is the only appender that can be created by using the SAS language.

When you create an appender in a DATA step, the appender is available only for the duration of the DATA step. After the DATA step has run, the appender is no longer available.

For more information, see “Creating and Using Appenders in a SAS Program” on page 88.

---

## Referencing Appenders in a Logger

After an appender is defined, it can be referenced by a logger. To reference an appender in a logging configuration file, you include the appender name in the logger’s `<appender-ref>` element. In the following logger and appender definitions, the appender `WinEvtVwr` is referenced by the logger `WEVLogger`:

```
<appender class="WindowsEventAppender" name="WinEvtVwr">
  <param name="Appname" value="myApp"/>
</appender>

<logger name="WEVLogger">
  <level="error"/>
  <appender-ref ref="WinEvtVwr"/>
</logger>
```

To reference an appender in a logger language element, you specify the appender name as the value of the `APPENDER-REF` argument:

```
%log4sas_logger(myLogger, appender-ref=(myAppender), level=error);

rc= log4sas_logger("myLogger" "appender-ref=(myAppender) level=error");

declare logger logobj("myLogger");
logobj.appenderref="myAppender";
```

To write the same message in multiple logs, you can specify multiple appender references in a configuration file logger definition:

```
<logger name="MyLoggers">
  <level="error"/>
  <appender-ref ref="WinEvtVwr"/>
  <appender-ref ref="RollingFileAppender"/>
</logger>
```

In a SAS program, you can add multiple appender names separated by a space in the `APPENDER-REF` argument:

```
%log4sas_logger(myLogger, appender-ref=(myAppender myRollingFile), level=error);
```

---

## Logging Thresholds

The SAS logging facility provides six thresholds: `TRACE`, `DEBUG`, `INFO`, `WARN`, `ERROR`, and `FATAL`. Thresholds are used to ignore log events that are lower than a particular level, or to filter messages so that only a single message level is logged.

When a log event occurs, up to three levels of filtering can take place:

- 1 filtering log events by comparing the log event level to the log event’s logger level
- 2 filtering log events by comparing the log event level to the appender’s threshold

- 3 filtering log events by comparing the log event level to the threshold that is specified in the filter definition, which is a part of the appender configuration

In the first two cases, if the log event level is lower than the logger or appender threshold, the logging facility ignores the log event. Otherwise, processing of the log event continues.

In the third case, the log event level is compared to the filter threshold. If there is a match, the log event can be either accepted or denied. If there is no match, the filtering process continues to the next filter in the filtering policy. For more information, see Chapter 7, “Filters,” on page 75.

The logging levels, from the lowest to the highest, are as follows:

TRACE	produces the most detailed information about your application. This level is primarily used by SAS Technical Support or development.
DEBUG	produces detailed information that you use to debug your application. This level is primarily used by SAS Technical Support or development.
INFO	provides information that highlights the progress of an application.
WARN	provides messages that identify potentially harmful situations.
ERROR	provides messages that might allow the application to continue running.
FATAL	provides messages that indicate that severe errors have occurred. These errors will probably cause the application to end.

Requirement: The level must be enclosed in quotation marks.

An appender can be configured to have a threshold. By default, however, appenders do not have a threshold. When set, all log events that have a level lower than the threshold are ignored by the appender.

---

## Formatting Messages

The format of a message can be customized by specifying a unique *pattern layout* for each appender class in the SAS logging facility. To create a pattern layout for an appender class, you use *conversion characters* that represent the types of data to include in the message. You can also control the sequence of the data and the alignment of the data in columns in the message.

*Note:* The conversion patterns that are used in the SAS logging facility and in the C language PRINTF statement are similar.

In addition to the set of pattern layouts that are used by the appender classes in the SAS logging facility, a different set of pattern layouts is used by ARMAppender. For more information, see ARMAppender Pattern Layouts in *SAS Interface to Application Response Measurement (ARM): Reference*.  $\Delta$

Here is an excerpt of an XML file that contains a pattern layout:

```
<layout>
  <param name="ConversionPattern" value="%d; %-5p; %t; %c; (%F:%L); %m"/>
</layout>
```

Each data item to be included in the message is represented by a conversion character. Also, literal text and alignment commands can be specified to enhance the message format. In this example, the data items are the date, the logging level, the

thread, the logger, the log file, the line number in the calling program that specified the logging request, and the message.

Here is an example of a message :

```
2008--06--25--10:24:22,234; WARN; 3; Appender.IOMContext; (yn14.sas.c:149);
    Numeric maximum was larger than 8, am setting to 8.
```

For more information, see Chapter 6, “Pattern Layout,” on page 63.

---

## Message Filtering

In addition to filtering log events based on thresholds that are assigned to loggers or appender definitions, the logging facility enables you to use filter classes to filter log events based on the following:

- a character string in the message
- a single threshold
- a range of thresholds
- a combination of strings and thresholds

Here is a list of the filter classes:

Filter Class Name	Description
StringMatchFilter	filters messages based on a character string in the message.
LevelRangeFilter	filters messages based on a range of thresholds.
LevelMatchFilter	filters messages based on a single threshold.
AndFilter	filters messages based on the results of a list of other filters.
DenyAllFilter	denies log events that did not meet the criteria of previous filters in a filter policy.

You can define one or more filters within the <appender> definition in the logging configuration file. Filters are not available in the logging facility language elements for SAS programs.

Filters are processed in the order that they appear in the <appender> definition, creating a *filtering policy* for the appender. The filters either accept the filtering criteria and process the log event, deny the filtering criteria and deny the log event, or accept the filtering criteria, and the filtering process checks the next filter in the filtering policy. If the log event has not been denied, and if there are no other filters in the filtering policy, the appender accepts and processes the log event.

For more information, see Chapter 7, “Filters,” on page 75.

## Using the SAS Logging Facility in the SAS Intelligence Platform

### About the Initial Logging Configuration for SAS Servers

When you install the SAS Intelligence Platform, the installation process performs the following configuration steps:

- It enables logging for each server by specifying the LOGCONFIGLOC= option in the server's configuration file.
- For each server, it provides a logging configuration file called **logconfig.xml** that is located in the server's configuration directory.
- For each server, it provides an alternative logging configuration file called **logconfig\_trace.xml** that can be used for troubleshooting.

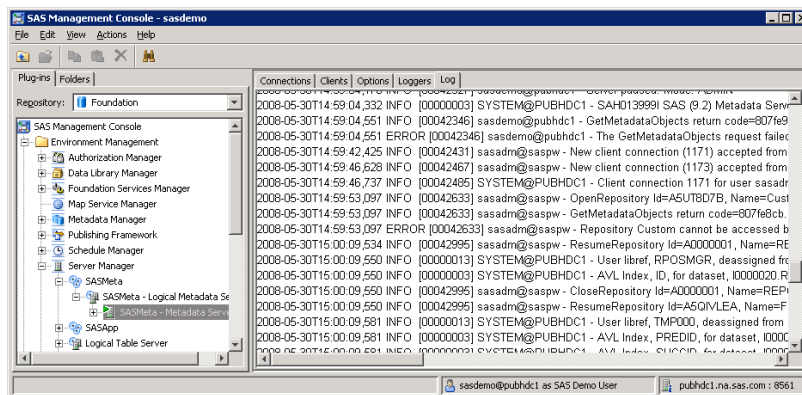
For more information, see the following topics in the *SAS Intelligence Platform: System Administration Guide*:

- “Initial Logging Configuration for SAS Servers”
- “Default Locations for Server Logs”

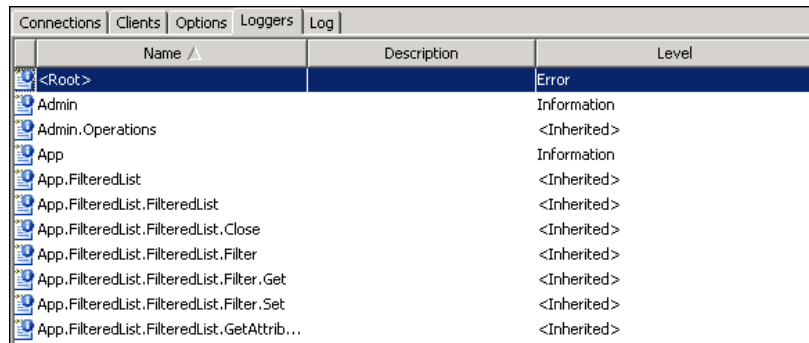
### Viewing SAS Logging Messages and Adjusting Logging Levels in Client Applications

The initial logging configurations for some SAS servers include appender definitions that make logging messages available in the following client applications:

- SAS Management Console. From this application, you can view logging messages for metadata servers, object spawners, OLAP servers, pooled workspace servers, stored process servers, and table servers. In this example, the metadata server log is displayed:



You can also use the Loggers tab in SAS Management Console to dynamically adjust server logging levels, without the need to restart the server. The following example shows the Loggers tab for the metadata server:



Name ▲	Description	Level
<Root>		Error
Admin		Information
Admin.Operations		<Inherited>
App		Information
App.FilteredList		<Inherited>
App.FilteredList.FilteredList		<Inherited>
App.FilteredList.FilteredList.Close		<Inherited>
App.FilteredList.FilteredList.Filter		<Inherited>
App.FilteredList.FilteredList.Filter.Get		<Inherited>
App.FilteredList.FilteredList.Filter.Set		<Inherited>
App.FilteredList.FilteredList.GetAttrib...		<Inherited>

For more information, see “Using SAS Management Console to Monitor SAS Servers” in the *SAS Intelligence Platform: System Administration Guide*.

- SAS Data Integration Studio. From this application, you can view performance-related events that are associated with a SAS Data Integration Studio job. For more information, see the product Help.

You can also use enterprise systems management products to view server logging messages and dynamically adjust server logging levels. For more information, see the Enterprise Management Integration Web page at <http://support.sas.com/rnd/emi>.

---

## Best Practices for SAS Server Logging

When using the logging facility for SAS servers, follow these best practices:

- Use the initial logging configuration files that are created during installation. These files provide a good starting point for server logging.
- If you need to change a server’s logging configuration, back up the initial configuration file before making changes. Make configuration changes incrementally, and evaluate the effect of each change before making additional changes.
- Do not use the TRACE and DEBUG levels unless you are directed to do so by SAS Technical Support, since these logging levels can affect performance. You can use either of these methods to adjust logging levels for SAS Technical Support:
  - Enable the `logconfig_trace.xml` file that is provided for the server.
  - Use the server manager features of SAS Management Console to adjust levels temporarily and avoid having to restart the servers.

For more information, see the following documents:

- “Administering Logging for SAS Servers” in the *SAS Intelligence Platform: System Administration Guide*.
- *SAS Interface to Application Response Measurement (ARM): Reference*, which describes SAS features that are compliant with the ARM 2.0 and ARM 4.0 standards and that enable you to monitor the performance of SAS applications.