



# Host Server Logging

## INTRODUCTION TO HOST SERVER LOGGING

October 13

This document describes the Host Server logging framework. The logging framework provides a unified logging mechanism that enables any subcomponent to log information to the same location.



**CORPORATE HEADQUARTERS**

4042 148th Avenue NE  
Redmond, WA 98052  
USA  
PH +1 (425) 883 4200  
FAX +1 (425) 671 0411

**EMEA HEADQUARTERS**

26 Worple Road  
Wimbledon  
London  
UK  
PH +1 44 (0) 845 612 0912  
FAX +1 44 (0) 845 612 0911

**APAC HEADQUARTERS**

9 Shenton Way #06-02  
Singapore  
068813  
PH +1 65 6327 4110  
FAX +1 65 6327 4120

[ [WWW.K2.COM](http://WWW.K2.COM) ]

The information provided relates to pre-release software products, may include features only available after installation of additional add-ins, and may be substantially modified before the commercial release. This information is provided for informational purposes only, and SourceCode Technology Holdings, Inc. makes no warranties, expressed or implied, with respect to this document or the information contained within it.

Copyright © 2008. SourceCode Technology Holdings Inc. All rights reserved. Patents pending. SourceCode and K2 are registered trademarks or trademarks of SourceCode Technology Holdings, Inc. in the United States and/or other countries. The names of actual companies and products mentioned herein may be the trademarks of their respective owners.



**CONTENTS**

- CONTENTS .....3
- > Introduction.....4
  - > Host Server Logging Destinations .....4
- > The Host Server Logging Configuration File.....4
  - > AppSettings .....6
  - > Extensions.....7
  - > ApplicationLevelLogSettings .....8
  - > CategoryList .....9
  - > MessageList ..... 10
- > Configuring Host Server Logging ..... 10
  - > Severity Levels ..... 10
  - > Enabling Logging ..... 10
  - > Asynchronous Queuing ..... 11
  - > Enabling Stack Trace ..... 11
- > Logging and Performance ..... 11
  - > Recommendations ..... 12



## INTRODUCTION

K2 blackpearl has a unified logging framework. This allows any subcomponent of K2 blackpearl to log information to the same location, leveraging a single mechanism and eliminating the need for extra code and overhead. This unified logging framework offers a variety of logging options for the Host Server, from different available log destination/types, to individually set output severity levels, and more. This article discusses the available Host Server logging options, how to configure them, and performance considerations for their use.

## HOST SERVER LOGGING DESTINATIONS

There are five different types of logging available out of the box in K2 blackpearl. You can choose to log to any, all, or none of them. Each destination can have customized logging levels configured. The five available types of logging are:

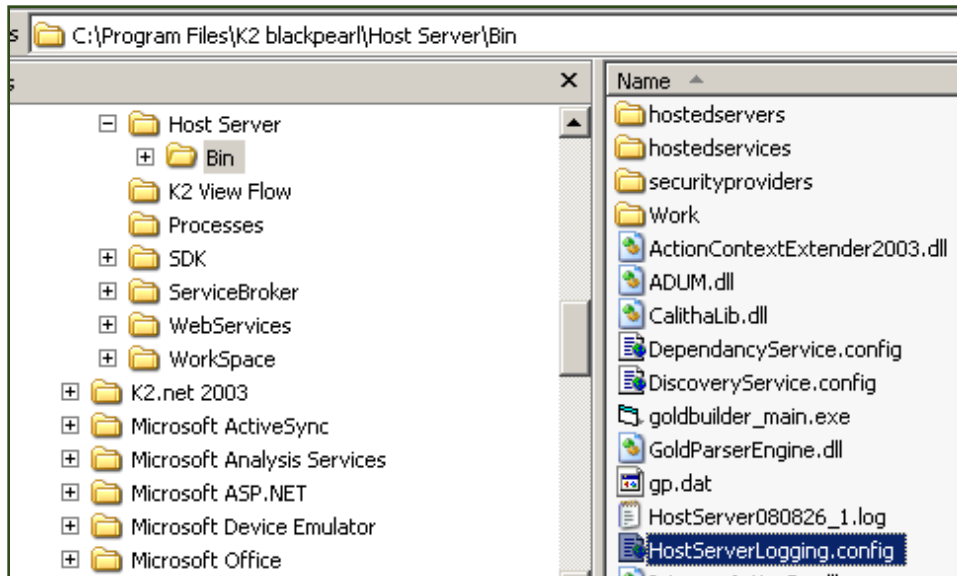
- **Console:** Messages will be logged to the K2 [blackpearl] server console when the server is running in console mode. To run the server in console mode, shut down the K2 Server (Start -> Administration Tools -> Services. Find the K2 [blackpearl] Server and stop it). After the server is stopped, load the console located at Start -> All Programs -> K2 [blackpearl] -> K2 [blackpearl] Server.
- **File:** Messages will be logged to a log file, saved to the location specified in the properties of its configuration declaration.
- **Windows Event Log:** Messages will be recorded to the machine's Windows Event Log.
- **MS Message Queue (MSMQ):** Messages can be logged to the Microsoft Messaging Queue. A logger can then be setup to monitor the Message Queue for messages and take action when necessary. This is useful when running multiple K2 [blackpearl] Servers, in a server farm for example, and one machine is set-up to handle all of the logging. This configuration is beyond the scope of this article.
- **SQL database:** Messages will be logged to SQL server, allowing queries to be run against logged messages. By default, messages will be logged to the HostServerDB connection, but the destination database can be modified by changing the properties associated with this extension.

Each logging destination introduces its own level of overhead to the K2 server. Therefore, it is important to balance the amount of information being logged, the destination of the logging, and server performance. Please see the section "Logging and Performance" later in this document for further information.

## THE HOST SERVER LOGGING CONFIGURATION FILE

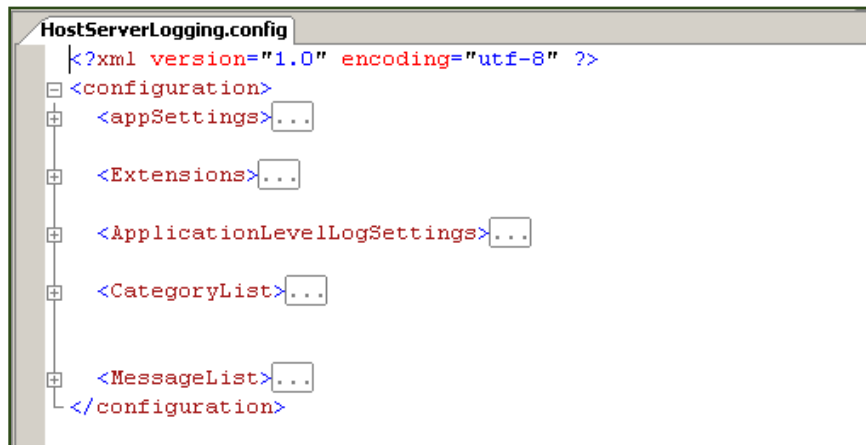
The file used to define the Host Server Logging options is HostServerLogging.config. It is typically located at:

C:\Program Files\K2 blackpearl\Host Server\Bin\HostServerLogging.config



This file can be opened in any text editor. Opening in an editor such as Visual Studio provides a convenient color-coded view. Let's investigate the basic structure of the HostServerLogging.config file. There are 5 main nodes in the file:

- appSettings
- Extensions
- ApplicationLevelLogSettings
- CategoryList
- MessageList



```
HostServerLogging.config
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>...
  <Extensions>...
  <ApplicationLevelLogSettings>...
  <CategoryList>...
  <MessageList>...
</configuration>
```

The screenshot shows a text editor window with the file name HostServerLogging.config. The XML content is displayed with color coding: <?xml is blue, version="1.0" is red, encoding="utf-8" is green, and ?> is blue. The <configuration> root element is blue, and its child elements (<appSettings>, <Extensions>, <ApplicationLevelLogSettings>, <CategoryList>, <MessageList>) are also blue, followed by their respective values (...). The closing tag </configuration> is blue.

Each node is explained separately in the sections that follow.



### APPSETTINGS

Allows you to configure certain system-wide logging settings. Keys included are:

Key Name	Description	Default
<b>DateFormat</b>	define the format for the date/time stamp on events	"yyyy-MM-dd hh:mm:ss"
<b>LogLevel</b>	define the default base log level	"All"
<b>RowHash</b>	specify whether to create Checksum/Hash for each row in log file	"False"
<b>Thread Priority</b>	Adjust Logger Thread Priority Low/Normal/High	1 "Low"
<b>AsyncQueue Enabled</b>	define whether asynchronous queuing is enabled	2 "True"
<b>IncludeStack Trace</b>	If the message logged was an exception, you can choose to include the stack trace	3 "False"
<b>Preserve Sequence</b>	define whether to preserve the log file SequenceNumber between restarts	5 "True"
<b>AddGUID</b>	whether to add a GUID per log entry	"True"
<b>MSMQActive</b>	Whether MSMQ is available as a logging option	"False"
<b>MSMQPath</b>	Set MSMQ Path for MSMQ-based logging	.\private\$\SCQueue

1 Increasing the Thread Priority can have a performance impact on your server(s).

2 By default, logging is performed asynchronously. This means that the K2 Host Server continues without waiting for confirmation that the message was successfully logged.

Disabling asynchronous queuing prevents K2 from performing any additional action until confirmation is received that the event has been logged. This can be used while troubleshooting, to ensure that all possible logging is received.

Disabling asynchronous queuing can have a performance impact on your server(s). It is recommended you only disable asynchronous queuing during troubleshooting.

3 Including the stack trace should ONLY be enabled when troubleshooting K2 Host Server crashes.



- 4 By default, only a service restart will create a new log file.
- 5 By default, Host Server log files are named using the format "HostServer" + "Date" + "\_" + "sequence number" + ".log". The sequence number is incremented for every service restart on the same date.

### EXTENSIONS

Specifies the available logging locations, their source Assembly, and specific configuration values for that extension. Default extensions are:

Extension Name	Property	Default
<b>Console</b>	Shorthand	"True"
<b>Event Log</b>	N/A	N/A
<b>File</b>	LogFileName	"HostServer.log"
	LogFilePath	""
	HashAlgorithm	"CRC32"
	MaxFileSizeKB	"0"
	MaxLifeTimeSpan	"0:0:0:00"
<b>MSMQ</b>	QueuePath	".\private\$\SCQueue"
<b>Archive</b>	HostServerConfigFileName	"K2HostServer.config"
	ConfigDBConnectionName	"HostserverDB"

The Shorthand property under the Console extension determines whether the full timestamp is displayed in events logged to the console. Set Shorthand="False" if you wish to see the full timestamp.



### APPLICATIONLEVELLOGSETTINGS

Each logging extension (destination) can be enabled individually, with its own level of logging output. This allows you, for example, to log Error messages to the Event Log, but Warning and higher messages to file. The ApplicationLevelLogSettings node of the configuration file determines the “Active” (True/False) state of each logging location, as well as the level of messages logged to that location:

```
<ApplicationLevelLogSettings>
  <ApplicationLevelLogSetting Scope="Default">
    <LogLocationSettings>
      <LogLocation Name="ConsoleExtension" Active="True" LogLevel="Info" />
      <LogLocation Name="FileExtension" Active="False" LogLevel="All" />
      <LogLocation Name="EventLogExtension" Active="False" LogLevel="Debug" />
      <LogLocation Name="ArchiveExtension" Active="False" LogLevel="Debug" />
      <LogLocation Name="MSMQExtension" Active="False" LogLevel="Debug" />
    </LogLocationSettings>
  </ApplicationLevelLogSetting>
</ApplicationLevelLogSettings>
```

This section can also be modified to filter the logged messages by namespace. By adding <ApplicationLevelLogSetting> sections with a scope other than “Default”, you can override the default logging level for certain namespaces. This allows you to set a more restrictive logging level for some namespaces, thus reducing the messages you consider “chatter”. This helps both to reduce the size of log files (or databases) and to simplify troubleshooting by weeding out unnecessary messages. Below is an example of how to do this.

```
<ApplicationLevelLogSetting Scope="SourceCode.Hosting">
  <LogLocationSettings>
    <LogLocation Name="ConsoleExtension" Active="True" LogLevel="Error" />
    <LogLocation Name="FileExtension" Active="True" LogLevel="Error" />
  </LogLocationSettings>
</ApplicationLevelLogSetting>
<ApplicationLevelLogSetting Scope="SourceCode.Workflow.Runtime.ClientConnection">
  <LogLocationSettings>
    <LogLocation Name="ConsoleExtension" Active="True" LogLevel="Error" />
    <LogLocation Name="FileExtension" Active="True" LogLevel="Error" />
  </LogLocationSettings>
</ApplicationLevelLogSetting>
<ApplicationLevelLogSetting Scope="SourceCode.Workflow.Runtime.SocketListener">
  <LogLocationSettings>
    <LogLocation Name="ConsoleExtension" Active="True" LogLevel="Error" />
    <LogLocation Name="FileExtension" Active="True" LogLevel="Error" />
  </LogLocationSettings>
</ApplicationLevelLogSetting>
<ApplicationLevelLogSetting Scope="SourceCode.EventBus">
  <LogLocationSettings>
    <LogLocation Name="ConsoleExtension" Active="True" LogLevel="Error" />
    <LogLocation Name="FileExtension" Active="True" LogLevel="Error" />
  </LogLocationSettings>
</ApplicationLevelLogSetting>
```



**CATEGORYLIST**

Includes all the available categories with which to label a log entry. Currently, there are 17 defined categories within K2 blackpearl Host Server logging.

<b>Cat ID</b>	<b>Category Name</b>	<b>Friendly Text</b>
0	General	General Events
1	System	System Events
2	Client	Client Events
3	Security	Security Events
3	Communication	Communication Events
6	SmartFunctions	SmartFunctions
7	Authorization Provider	Authorization Provider
8	SmartObjects	SmartObjects
9	DependencyService	Dependency Service
10	UserRoleManager	UserRoleManager Server
11	CategoryServer	Category Server
12	WorkflowServer	Workflow Server
13	LiveCommServer	Live Comm Server
14	EventBus	Event Bus
15	EnvironmentServer	Environment Server
16	SystemAudit	System Audit
17	WorkActivity	Work Activity



### MESSAGELIST

The full list of messages available to the K2 Host Server's Logging Framework. This section includes the definition of each message, sorted by numeric message code. Some messages include variables (specified in the "{0}" format). The message definition must include the severity level, category (see Category List, above), and message name, as well as the message output format, including the expected variables.

It is possible to define custom error messages. Other articles discuss how to attach to the EventBus, and similar. You can develop your own components that can use the Unified Logging Framework within K2 blackpearl.

```
<MessageList>
  <!-- Severity Levels Are: Ignore, Debug, Info, Warning, Error -->
  <!-- Custom Logging Targets per entry: ToFile="True" ToConsole="True" ToEventLog="False"
ToCustom="False" -->
  <Message MsgID="0" Severity="Debug" Category="0" Name="Unknown" ToFile="True" ToConsole="True"
ToEventLog="True" ToCustom="False">
    Unknown Event Occured.
  </Message>
  <Message MsgID="1" Severity="Error" Category="0" Name="GeneralErrorMessage">
    {0}
  </Message>
```

### CONFIGURING HOST SERVER LOGGING

There are really only two sections of the HostServerLogging.config file you will need to modify to configure logging. For most situations, you need only set the specific log destination's Active flag to true or false, and specify the severity of messages to log. In rare circumstances, you may need to modify the "appSettings" section of the file, particularly to disable asynchronous queuing and to include a stack trace.

All changes to the HostServerLogging.config file require a restart of the K2 service to take effect.

### SEVERITY LEVELS

There are five severity levels. The five levels, in order of severity (highest to lowest) are:

- Error
- Warning
- Info
- Debug
- Ignore

The K2 Host Server will log messages with at least the severity level specified, to the defined destination. So, based on the list above, setting the severity level to "Info" will result in all Info, Warning, and Error messages being logged.

### ENABLING LOGGING

The two most common forms of logging are "to Console" and "to File". Logging to the console is very useful for troubleshooting and validating general issues. However, from a performance perspective, logging to the console has high overhead, and should only be enabled when needed. Similarly, logging to file has some additional overhead. It is therefore recommended to only log "All" events to file during periods of troubleshooting. For more regular monitoring purposes, you should scale back file-based logging to only log events of severity Warning or higher.



### Enable a Specific Logging Destination

1. Open HostServerLogging.config.
2. Find the "<ApplicationLevelLogSettings>" section .
3. Find the line for the specific Extension type you wish to enable.
4. Modify the Active setting from:

```
Active="False"
```

to:

```
Active="True"
```

### Specify Severity Level

1. Open HostServerLogging.config.
2. Find the "<ApplicationLevelLogSettings>" section.
3. Find the line for the specific Extension type you wish to configure.
4. Modify the LogLevel setting:

```
LogLevel="All"
```

By default, console logging is turned on, but at the "Info" severity level. Thus, if you start K2 server in console mode without modifying the HostServerLogging.config file, you will see Informational, Warning, and Error messages. File-based logging is turned off by default.



**Warning:** The following settings can have a significant impact on K2 Server performance. Do not change these settings unless requested to do so by a K2 Support Professional.

### ASYNCHRONOUS QUEUING

By default, logging is performed asynchronously. This means that the K2 Host Server continues without waiting for confirmation that the message was successfully logged. This is generally a good idea, since the log queue generally receives lower thread priority than other K2 Server processing.

Disabling asynchronous queuing prevents the K2 Host Server from performing any additional action until confirmation is received that the event has been logged. This can be used while troubleshooting, to ensure that all possible logging is received. Disabling asynchronous queuing can have a performance impact on your server(s). It is recommended you only disable asynchronous queuing during troubleshooting, and generally only under the guidance of K2 Support.

### ENABLING STACK TRACE

If you are experiencing unhandled exceptions in K2, leading to server/service crashes, you may be asked by K2 Support to enable the Stack Trace feature of logging. This will include a stack trace at the time the exception occurs. You will have to disable asynchronous queuing to enable the stack trace functionality.

### LOGGING AND PERFORMANCE

As previously mentioned, each type of Host Server logging has its own performance implications. With stack trace disabled, and asynchronous queuing in use, only the log destination-type determines performance impacts. Informal tests have shown the performance quality to be (from best to worst):



- Logging OFF
- Console
- SQL
- Event Logs
- File

Console logging offers no means of storing the logs past the buffer of the console window. Event Logs and SQL have similar performance; the distinguishing characteristic between them is ease of consumption. Event Logs tend to be difficult to traverse and can bog down heavily based on other OS-level conditions, whereas SQL is a much more flexible means of storage – offering much greater storage and filtering capacity. File logging is convenient for portability during troubleshooting, but has the highest overhead.

#### RECOMMENDATIONS

As a general rule, logging should only be enabled at the severity level necessary, and to the most efficient destination possible. Here are some very general guidelines for K2 Host Server logging. Bear in mind that your performance impacts and needs should play a significant part in your logging decisions.

During setup, testing, and troubleshooting, increase the level of logging to Debug. This will allow you to see the maximum amount of information, and more easily determine exactly what is/isn't working. Console and/or File logging are typically best suited to short-term monitoring that fits these use models.

During normal operations, scale back the severity of messages being logged, especially to the slower destinations. File logging, if needed, should be reserved for only the most severe messages (Error). Use SQL logging for storing larger volumes of information, and/or for consolidating data from multiple K2 farm nodes. Normal operating conditions should generally not need more than Warning level messages.

Synchronous logging and stack trace inclusion should ONLY be used under the supervision of K2 Support staff. These options should not be used under normal operating conditions.