



Setting Up Structured Logging

Version 2019.4
2020-01-28

Setting Up Structured Logging

InterSystems IRIS Data Platform Version 2019.4 2020-01-28

Copyright © 2020 InterSystems Corporation

All rights reserved.

InterSystems, InterSystems IRIS, InterSystems Caché, InterSystems Ensemble, and InterSystems HealthShare are registered trademarks of InterSystems Corporation.

All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Memorial Drive, Cambridge, MA 02142, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)

Tel: +1-617-621-0700

Tel: +44 (0) 844 854 2917

Email: support@InterSystems.com

Table of Contents

Setting Up Structured Logging	1
1 Information Available in the Structured Log	1
2 Example Output	1
2.1 Name/Value Pairs	1
2.2 JSON	2
3 Enabling Structured Logging	3
4 Class-Based API for Structured Logging	4
5 Other Options of irislogd	4

Setting Up Structured Logging

InterSystems IRIS now supports *structured logging*.

InterSystems IRIS creates multiple logs, each for different purposes. Customers who have migrated from previous products can take advantage of these logs as in the past, but now it is also possible to channel all the log information into a single, central, machine-readable log file — a *structured log*. Then you can use this file with third-party analysis tools.

This article provides an overview of the [information in the structured log](#), shows [examples of the log](#), and describes how to [enable](#) structured logging.

1 Information Available in the Structured Log

In InterSystems IRIS, when you enable structured logging, the system writes the same data to the structured log that it also writes to the other log (whichever that is). For example, the system writes the same lines to `messages.log` and to the structured log.

When you have enabled structured logging, the structured log contains all the following information:

- The information that is written to `messages.log`. This *includes* alerts that require attention, information about the system startup and shutdown, high-level information about journal files and WIJ files, information about configuration changes (the CPF), and information related to licensing. See “[Monitoring InterSystems IRIS Logs](#)” in the chapter “[Monitoring InterSystems IRIS Using the Management Portal](#)” in *Monitoring Guide*.
- The information that is written to the audit database. The details depend on which events you are auditing. See “[Auditing](#)” in *Security Administration Guide*.

2 Example Output

This section shows example output from the structured logging utility for [name/value pair format](#) and [JSON format](#).

2.1 Name/Value Pairs

The following output uses the format option `NVP` (name/value pairs). This sample was edited for display purposes; in the actual output, each entry takes only a single line, and there are no blank lines between entries.

```
when="2019-08-01 18:43:02.216" pid=8240 level=SEVERE event=Utility.Event
text="Previous system shutdown was abnormal, system forced down or crashed"

when="2019-08-01 18:43:05.290" pid=8240 level=SEVERE event=Utility.Event
text="LMF Error: No valid license key. Local key file not found and LicenseID not defined."

when="2019-08-01 18:43:05.493" pid=8240 level=WARNING event=Generic.Event
text="Warning: Alternate and primary journal directories are the same"

when="2019-08-01 18:46:10.493" pid=11948 level=WARNING event=System.Monitor
text="CPUusage Warning: CPUusage = 79 ( Warnvalue is 75)."
```

In this format, each line in the file contains a set of name/value pairs separated by spaces. Each name/value pair has the form `name=value`, and if `value` includes a space character, then `value` is enclosed in parentheses. The lines in the log file include some or all of the following name/value pairs:

Name	Value
host	Name of the host on which ^LOGDMN is running, if provided within the pipe command.
instance	Name of the instance on which ^LOGDMN is running, if provided within the pipe command.
when	<i>Always included.</i> The time stamp of the entry in the format <code>yyyy-mm-dd hh:mm:ss.sss</code>
pid	<i>Always included.</i> The ID of the process that generated the entry
level	<i>Always included.</i> The log level of this entry. This has one of the following values: <ul style="list-style-type: none"> DEBUG2 is used for detailed debug messages (such as hex dumps). DEBUG is used for less detailed debug messages. INFO is used for informational messages, including all audit events. WARNING is used to indicate problems that may need attention but that have not disrupted operations. SEVERE is used for severe errors, which indicate problems that have disrupted operations. FATAL is used for fatal errors, which indicate problems have caused the system not to run.
event	<i>Always included.</i> Identifier for the code that generated the entry, typically the class name.
text	<i>Always included.</i> Descriptive string that explains the entry.
source	The component that is the source of audit event. For InterSystems components, this is always <code>%System</code> . When your application code writes to the event log, <code>source</code> indicates the component in your application code.
type	Categorizing information for the audit event.
group	Group of the audit event, if any.
namespace	Namespace in which the entry was generated. This is useful for examining namespace-specific activity such as application errors and the activity of interoperability productions.

2.2 JSON

The following output uses the format option `JSON`. This sample was edited for display purposes; in the actual output, each entry takes only a single line, and there are no blank lines between entries.

```
{ "when": "2019-08-07 14:11:04.904", "pid": "8540", "level": "SEVERE", "event": "Utility.Event",
"text": "Previous system shutdown was abnormal, system forced down or crashed" }

{ "when": "2019-08-07 14:11:08.155", "pid": "8540", "level": "SEVERE", "event": "Utility.Event",
"text": "LMF Error: No valid license key. Local key file not found and LicenseID not defined." }

{ "when": "2019-08-07 14:11:08.311", "pid": "8540", "level": "WARNING", "event": "Generic.Event",
"text": "Warning: Alternate and primary journal directories are the same" }

{ "when": "2019-08-07 14:16:13.843", "pid": "10816", "level": "WARNING", "event": "System.Monitor",
"text": "CPUUsage Warning: CPUUsage = 84 ( Warnvalue is 75)."}

```

In this format, each line in the file is a JSON object with a set of properties. The names of the properties (and the values contained in the properties) are the same as listed for the name/value pairs in the previous section.

3 Enabling Structured Logging

The `^LOGDMN` routine lets you manage structured logging; there is also a class-based API, described in the [next section](#).

To use `^LOGDMN` to enable structured logging:

1. Open the Terminal and enter the following commands:

```
set $namespace="%sys"
do ^LOGDMN
```

This starts a routine with the following prompts:

```
1) Enable logging
2) Disable logging
3) Display configuration
4) Edit configuration
5) Set default configuration
6) Display logging status
7) Start logging
8) Stop logging
9) Restart logging
```

```
LOGDMN option?
```

2. Press 4 so that you can specify the configuration details. The routine then prompts you for the following items:
 - a. The minimum log level, one of the following:
 - -2 — detailed debug messages (such as hex dumps).
 - -1 — less detailed debug messages.
 - 0 — informational messages, including all audit events.
 - 1 (the default) — warnings, which indicate problems that may need attention but that have not disrupted operations.
 - 2— severe errors, which indicate problems that have disrupted operations.
 - 3— fatal errors, which indicate problems have caused the system not to run.
 - b. The pipe command, which specifies where the system will send the structured log. Enter a response of the following form:


```
irislogd -f c:/myfilename.log
```

But replace `c:/myfilename.log` with the fully qualified path name of the destination log file. In this command, `irislogd` is the name of an InterSystems executable file that will receive the log data and write it to the specified file (via the `-f` option).

For the pipe command, the easiest option is to use the executable mentioned here (`irislogd.exe`), but you can substitute a different target. For other options for `irislogd.exe`, see the [last section](#).
 - c. The format of the data sent to the pipe. Specify either `NVP` (the default) or `JSON`. The option `NVP` sends data that consists of name-value pairs, separated by spaces. The option `JSON` sends data in JSON output. For examples, see “[Example Output](#),” earlier in this article.
 - d. The interval in seconds between successive calls to the pipe command. The default is 10 seconds.
3. When the routine displays the main prompt again (`LOGDMN option?`), press 1 to enable logging.
4. Press 7 to start logging.

4 Class-Based API for Structured Logging

To manage structured logging, you can use the class `SYS.LogDmn` in the `%SYS` namespace instead of using the `^LOGDMN` routine. For details, see the class reference.

5 Other Options of `irislogd`

When you invoke the `irislogd.exe` executable via `^LOGDMN`, you can pass the following arguments to the executable:

Argument	Purpose
<code>-d</code>	Emit diagnostic and error messages
<code>-e <i>errfilename</i></code>	Write errors and diagnostic messages to the given file.
<code>-f <i>logfilename</i></code>	Write log messages to the given file.
<code>-h <i>hostname</i></code>	Includes the given host name in the structured log file.
<code>-i <i>irisinstance</i></code>	Includes the given instance name in the structured log file.
<code>-s</code>	Write log messages to the Unix® syslog facility (Unix® only)

Also, you can write the output to stdout. To do on Unix, omit both `-f` and `-s` arguments. To do so on Windows, omit the `-s` argument.