

Monitoring

Caché Health

Executive Summary

Good monitoring practices are a vital part of an overall strategy for keeping your systems healthy. While Caché is largely self-regulating when properly installed and configured, unforeseen events can happen. Caché provides a suite of tools, processes and APIs for the real-time monitoring necessary to keep watch on Caché appropriately. An effective monitoring process looks not only at Caché, but the application, the operating system, the network, the storage system, and the hardware. While the focus of this document is Caché, these other areas need to be part of your comprehensive approach to monitoring.

Caché includes monitoring utilities and services that are ready to use. This may be sufficient for many applications with only some simple setup required. For sites requiring integration with existing tools or products, or applications that need more sophisticated monitoring processes, Caché provides the framework, APIs and system classes to build custom, powerful and integrated solutions.

The technical details on all of the facilities mentioned in this document are all well covered in the online documentation provided with each Caché install. As such, this document does not go into any detail on the implementation process.

Overview

Caché is quite resilient; however problems can still occur, such as exhausting critical resources, hardware failure or human error. By using the tools that are provided with Caché, process and procedures can be put in place to track and provide notification of issues or trends, allowing you to be proactive instead of reactive. The right mix of these processes, procedures and scripting depends heavily on your application and service level requirements.

Some of areas of Caché to consider in your monitoring program include checking:

- That all critical system daemons are running
- That there is enough disk space or there has been unusual disk space usage
- Whether there have been any access violations or dead jobs
- If there have been any system errors logged
- Whether any critical performance metrics have reached unusual levels
- If there is any evidence of data corruption
- Whether any application errors have been logged
- If the application is close to running out of Caché licenses
- That the backups ran correctly and completely
- That Journaling is running properly

- That Shadowing is running and in synch
- That application activity is within norms

The Console Log

The primary information source for monitoring Caché is the console log (cconsole.log). Caché reports general messages, system errors, certain operating system errors, and network errors through an operator console facility. The console log file is a plain text file and may be viewed with any editor or text viewer. It is found in the MGR subdirectory of the location where Caché was installed. Starting with v5.1, an additional log file, alerts.log, is created whenever a severe or fatal error is detected in the cconsole.log file. This log contains only the severe and fatal error messages.

Console Log Monitoring

Starting with Caché 5.0.20, additional facilities have been added to enhance system monitoring. One such facility monitors the system daemons and creates a console log entry whenever a system daemon is no longer running. Another new process scans the console log looking for level 2 (severe) and level 3 (fatal) events and sends an email alert to specific addresses. Because the console and alert logs are a plain text files, you have the option of creating your own console log scanner, either as a Caché process or an external process, which performs the specific actions desired for your system.

Caché Tools

There are other tools available besides the console log. Some of the tools run within the Caché environment and others work external to Caché. Within Caché, most of the utilities have both a command line and a GUI interface available. Appendix A lists some of the more interesting command line utilities for monitoring. The GUI tools are mostly found in the Caché Control Panel, accessible from the Caché Cube in the system tray (in versions prior to 5.1). In 5.1 these are all available in the System Management Portal.

External Tools

The primary tool used outside of the Caché environment is 'cstat.exe'. This utility is provided with Caché and provides a wealth of information about the running Caché system. This information is used for troubleshooting, performance monitoring and health monitoring. See Appendix B for specific metrics to consider. Caché also has a built in interface to BMC's Patrol Performance Manager (formerly called BMC Patrol). For companies that use this tool, Caché fits well into the overall monitoring strategy. Starting with Caché 5.1, a full SNMP API is also available. For other commercial monitoring tools, custom interfaces can be created to provide data in the method required by those tools.

How to Alert

Caché provides many options for issuing an alert. The MONITOR utility has a built-in mechanism to email its alerts. When Patrol Performance Manager is used, Caché provides the raw metrics and Patrol manages the alarms and triggers. Where SNMP agents are used, Caché 5.1 can generate SNMP messages for the agents. If you build your own custom monitoring solution, Caché provides a variety of options for

communicating alerts, such as: TCP sockets; SOAP and web services; email; ftp; web browser; SQL gateways; flat files and printers.

Where to Start

To implement the proper monitoring solution for your company and application, it is important to understand the goals, objectives and policies applicable to the application. In many cases the built in tools will be sufficient (see, The Basics below). Or, a more sophisticated solution may be needed that will involve possibly some custom coding. Where integrating into an existing monitoring structure is required, the Patrol or SNMP solutions may be the best approach.

The Basics

A good first place to begin is to use the built in MONITOR utility. It sends an email based on errors listed in the console log. This will catch of the critical issues logged in the console log. Some examples of these issues are: identifying when a system daemon is not running, detecting if an access violation has occurred, or detecting if certain types of corruption have occurred. All that is required is to setup the email parameters.

BMC Patrol

If BMC Patrol is part of your overall monitoring strategy, Caché provides a Patrol Knowledge Module. When enabled, Caché delivers dozens of system metrics to Patrol at intervals of your choosing. Patrol also has the ability to perform log scanning. Because the Caché console log and alert log are a plain text files, Patrol can easily include this action into its overall process.

SNMP

To enable monitoring of Caché by a variety of systems management tools and frameworks, support for the Simple Network Management Protocol (SNMP) has been added in Caché v5.1. The %SYSTEM.MonitorTools.SNMP class allows for control of SNMP agents and functions. This class contains methods to start and stop the Caché SNMP agent, as well as the CreateMIB() method which generates a custom MIB file based on an application description in the Monitor Framework.

GUI Tools

Caché has many of the metrics and statuses that should be monitored regularly available in an interactive GUI form. In Caché 5.0 and earlier, they are in the Control Panel, and in 5.1 and following they are in the System Management Portal. The topics available in these tools are listed in a table below. In particular, the System Management Portal has a System Dashboard that can be used to monitor many aspects of Caché. From the dashboard page you can view performance indicators and then drill down to other operations pages for more detailed information. Also available on the dashboard are the capabilities to monitor locks, view log files, view CSP sessions, view background tasks, view ECP and Shadowing statuses, monitor system usage, view license information and usage, and view scheduled tasks in the task manager.

APIs

While the Monitor utility, the Dashboard, SNMP and Patrol offer monitoring capabilities out of the box, the real power in monitoring Caché are the programming APIs available. The APIs and Utilities chart below lists some of the routines, classes and services available to programmers that can be used to tailor a monitoring solution specific to an application. With these tools, Caché monitoring can be tied into an existing enterprise tool, or a custom solution can be built. One such idea is to build a “report card” that collects metrics and status information and then emails this data to the systems manager on a periodic basis. Another valuable capability available to exploit is the raw data from the Dashboard. This data is stored in ^CacheTemp and is updated every few seconds. The details on these metrics are in Appendix D.

The cstat Utility

Another level of monitoring would be to create a process external to Caché that will watch Caché. The ‘cstat’ program would be the primary tool for looking at Caché metrics. This could be scripted using the operating system’s scripting language, some other language of choice (such as Perl), or called by an enterprise monitoring tool. This provides a method for observing Caché from the outside apart from possible effects of the Caché environment itself. An external monitor has the advantage of easily including the concurrent monitoring of non-Caché components.

User Application Monitoring

You may also find it valuable to add application metrics to your monitoring solution. Your application could feed your monitoring tool information about itself, such as number of transactions processed, error conditions, job statuses and the like. Knowing your application’s normal settings will help define the triggers for appropriate alerts.

APIs and Utilities

Below are some of the more important areas to monitor in Caché, and a list of tools that can assist with the identification of alerts. These tools are routines or system classes that can be built in to a custom data collection or monitoring program, or can be built in to the %Monitor class using the %Monitor.Adaptor feature and let Caché perform the alerts. Full documentation on the use of the utilities and system classes is available in the Caché online documentation. Tools in italics are only available in 5.1 or following. In addition, there are Caché provided GUI tools to interactively monitor each of the situations below (except for item 3 and 12). In v5.0 or earlier, these are accessible through the Control Panel section of the Cube in the system tray on Windows systems. In v5.1 and later systems, these are all available through the web based Systems Management Portal.

Monitoring Topics	Tools Available
1. Check that all critical system daemons are running	<i>^MONITOR</i> , <i>cstat.exe</i> (for some of the daemons), %SS

2. Check that there is enough disk space or there has been unusual disk space usage	cstat.exe, ^%FREECNT, %Sys.GlobalQuery, %Monitor.System.Database, %Monitor.System.Freespace
3. Check whether there have been any access violations or dead jobs	^MONITOR, %Library.File, cstat.exe
4. Check if there have been any system errors logged	^MONITOR, cstat.exe
5. Check whether any critical performance metrics have reached unusual levels	^mgstat, cstat.exe, %Monitor.System.Dashboard
6. Check if there is any evidence of data corruption	^Integrity, %Sys.IntegrityCheck
7. Check whether any application errors have been logged	^%ERN
8. Check if the application is close to running out of Caché licenses	%System.License
9. Check that the backups ran correctly and completely	<i>Backup.General</i>
10. Check that Journaling is running properly	<i>%Monitor.System.Dashboard</i>
11. Check that Shadowing is running and in synch	Customer provided heartbeat utility (see WRC for assistance)
12. Check that application activity is within norms	<i>%Monitor.Adaptor</i>

References and Additional Resources

Details on the routines and classes mentioned in this document are available in the online Caché documentation. In addition, there are other whitepapers and good practice documents available that may be of interest. These are available at the InterSystems website, within the documentation, or from your local InterSystems contact.

- Best Practices on Journaling Caché 5.0 (whitepaper)
- Best Practices for Checking Database Integrity (whitepaper)
- Using BMC Patrol To Monitor Caché (in the online documentation)
- Caché High Availability Guide (in the online documentation)
- The CHUI-Based Management Routines (in the 5.1 online documentation)
- Caché Monitoring Guide (in the 5.1 online documentation)

Appendices

Appendix A – Command Line Utilities Helpful for Caché Health Monitoring

The following command line utilities all run from the Caché command prompt or from within a Caché ObjectScript routine except for cstat.exe. The cstat utility is to be run from the operating system command line, or within an operating system script.

- ^Integrity – is a utility that evaluates the internal structure of a Caché database and reports any possible corruption.
- ^MONITOR – scans the console log and emails alerts when level 2 (severe) or level 3 (fatal) events are found (5.0.20 or greater) and is started automatically at startup. The command line is used to setup the email alerts and start/stop the background job.
- ^SYSLOG – dumps the internal Caché system error log (used more for debugging than for health monitoring).
- ^%SS – system status display of all Caché processes; can be used to verify all system processes or required application processes are running; it can be called interactively or programmatically.
- ^%ERN – displays any errors that occurred in ObjectScript routines.
- ^Integrity – assesses the integrity of the db, ideally run once a day.
- ^%FREECNT – shows free space within a Caché database.
- cstat.exe – is an executable that runs outside of Caché. Details on its operation can be provided by the WRC.
- ^%UTIL (v5.0), or \$\$\$LOGMSG (v5.1+) – is the utility for an application to call to insert messages into the cconsole.log file, and specify a severity level.
- ^mgstat – is the utility heavily used in troubleshooting performance issues. Details on its use are available from the WRC.

Appendix B – System Classes Available for Health Monitoring

- %Monitor.Manager – sets up the Monitor email settings, and in Caché 5.1 it sets up and runs the collection of performance and system metrics.
- Backup.General (Caché 5.1) – provides information on last backup statuses.
- %Library.File – this class is useful for scanning directories to look for file based alerts, such as the existence of a core file that indicates a job has had an access violation.
- %Monitor.System.Dashboard (Caché 5.1) – returns information on: journal space availability, standard metrics, custom metrics.
- %Monitor.System.Database (Caché 5.1) – returns information on databases, including disk space available, free space available and database current size.

- %Monitor.Adaptor (Caché 5.1) – provides a way to build in custom alerts into the Monitor subsystem. This is one way to include application metrics into Caché’s built-in monitoring and alerting facilities, or any other Caché metric not already in %Monitor.
- %System.License – provides information on licenses including the number licenses consumed, licenses available and the maximum consumed.
- %Sys.GlobalQuery (Caché 5.1) – calculates the size of all globals in a database.
- %Sys.Task.IntegrityCheck (Caché 5.1) – provides a way to run integrity checks in the background using the %Sys.Task facility.
- %SYSTEM.MonitorTools.SNMP (Caché 5.1) - allows for control of SNMP agents and functions. This class contains methods to start and stop the Caché SNMP agent, as well as the CreateMIB() method which generates a custom MIB file based on an application description in the Monitor Framework.

Appendix C – Caché Metrics Helpful for Health Monitoring

The following internal Caché metrics can be used as part of a comprehensive health monitoring solution. Each metric must be individually considered based on the application to be monitored as to whether it would provide value or not. It is suggested that a system be profiled during a busy time and that ‘picture’ is used for comparison as the system is monitored. In this way “normal” can be defined specifically for each site, and an alarm can be set based on some threshold related to “normal”. These metrics may be accessed with cstat (C), BMC Patrol (P), SNMP (S), or a Caché System class (SC).

- gavailbuf (C, P, S, SC) – this is the number of buffers free and available for use.
- wdstop (C, P, S) – when this is set to a ‘2’, it indicates that Caché stopped the write daemon to prevent further harm.
- gwdqsize (C, P) – this is the number of ‘dirty’ blocks to be written to the disk. If this continues to grow with no shrinking, there is a write daemon or disk problem.
- glbrefs (P, S, SC)– this is the total number of global references performed by Caché. If this number significantly jumps up or declines from the norm, there may be an application issue to research.
- physblkread (P, S, SC) – this is the number of physical reads from disk. If this number starts getting high (‘high’ is application dependent), then you may need more global buffers.
- gminbuf (C) threshold of available buffers below which Cache will freeze. Check that gavailbuf is some percentage higher than gminbuf.

Many others exist and can be viewed by looking in the SNMP MIB file or looking in the Caché Patrol KM file. Both of these files are text files. InterSystems will also provide more detailed information on the cstat.exe utility upon request.

Appendix D – Dashboard Metrics Available in CacheTemp

In 5.1, the SMP Dashboard is backed by globals written to CacheTemp. These globals and their meaning can be found in the documentation for the class

%Monitor.System.Dashboard:

^CacheTemp.SysMetrics("LastBackup")	date/time of last backup
^CacheTemp.SysMetrics("Processes",i)	cpu time for 5 highest processes
^CacheTemp.SysMetrics("CSPSessions")	number of current CSP sessions
^CacheTemp.SysMetrics("DatabaseSpace")	OK, Troubled, Warning based on space available > 5MB, < 5MB, <2MB
^CacheTemp.SysMetrics("DatabaseSpace", directory)	directory of low database
^CacheTemp.SysMetrics("SeriousAlerts")	text of alerts found in alerts.log
^CacheTemp.SysMetrics("ECPServers")	status of ECP servers if system is a client; OK, Troubled, Warning based on status
^CacheTemp.SysMetrics("ECPServers",name)	name of troubled server
^CacheTemp.SysMetrics("ECPClients")	status of connections if system is a server; OK, Troubled, Warning based on status
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client
^CacheTemp.SysMetrics("ECPClients",name)	name of troubled client

<code>^CacheTemp.SysMetrics("WriteDaemon")</code>	"Troubled" If WD has been on same pass, with a non-zero write queue, and a non-zero write buffer size, for 1 minute.
<code>^CacheTemp.SysMetrics("RoutineRefs")</code>	Routine References
<code>^CacheTemp.SysMetrics("GlobalRefs")</code>	Global references
<code>^CacheTemp.SysMetrics("GlobalSetKill")</code>	Global sets and kills
<code>^CacheTemp.SysMetrics("LogicalBlockRequests")</code>	Logical block requests, incremented BLKRD
<code>^CacheTemp.SysMetrics("PhysicalBlockReads")</code>	Physical block reads due to globals or routines
<code>^CacheTemp.SysMetrics("PhysicalBlockWrites")</code>	Physical block writes due to globals or routines
<code>^CacheTemp.SysMetrics("JournalEntries")</code>	Journal Entries
<code>^CacheTemp.SysMetrics("RoutineLines")</code>	Routine Lines
<code>^CacheTemp.SysMetrics("CacheEfficiency")</code>	Cache Efficiency: global references / (phys read + writes)
<code>^CacheTemp.SysMetrics("GlobalRefsPerSecond")</code>	Global refs per second