



Identifying Memory Fragmentation within the Microsoft IIS Environment

Version 5.2
01 September 2006

Identifying Memory Fragmentation within the Microsoft IIS Environment

Caché Version 5.2 01 September 2006

Copyright © 2006 InterSystems Corporation.

All rights reserved.

This book was assembled and formatted in Adobe Page Description Format (PDF) using tools and information from the following sources: Sun Microsystems, RenderX, Inc., Adobe Systems, and the World Wide Web Consortium at www.w3c.org. The primary document development tools were special-purpose XML-processing applications built by InterSystems using Caché and Java.



The Caché product and its logos are registered trademarks of InterSystems Corporation.



The Ensemble product and its logos are registered trademarks of InterSystems Corporation.



The InterSystems name and logo are trademarks of InterSystems Corporation.

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.

Caché, InterSystems Caché, Caché SQL, Caché ObjectScript, Caché Object, Ensemble, InterSystems Ensemble, Ensemble Object, and Ensemble Production are trademarks of InterSystems Corporation. All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Customer Support

Tel: +1 617 621-0700

Fax: +1 617 374-9391

Email: support@InterSystems.com

Table of Contents

- Identifying Memory Fragmentation within the Microsoft IIS Environment..... 1**
- 1 Technology Overview 1
- 2 Problem Description 3
- 3 Investigation and Analysis 4
- 4 Remedies 5
- 5 A Special Note for WebLink Users 6
- 6 Conclusion 8

List of Figures

- Connectivity Options for CSP and WebLink 3

Identifying Memory Fragmentation within the Microsoft IIS Environment

This article describes an issue with the Caché WebLink and CSP technologies. The problem has manifested itself in the IIS/ISAPI (Internet Information Services/Internet Server application programming interface) connectivity option in both products. The result of the analysis is described together with the putative diagnosis. Finally, remedial action taken within the WebLink and CSP software is discussed.

The article contains the following sections:

- [Technology Overview](#)
- [Problem Description](#)
- [Investigation and Analysis](#)
- [Remedies](#)
- [A Special Note for WebLink Users](#)
- [Conclusion](#)

1 Technology Overview

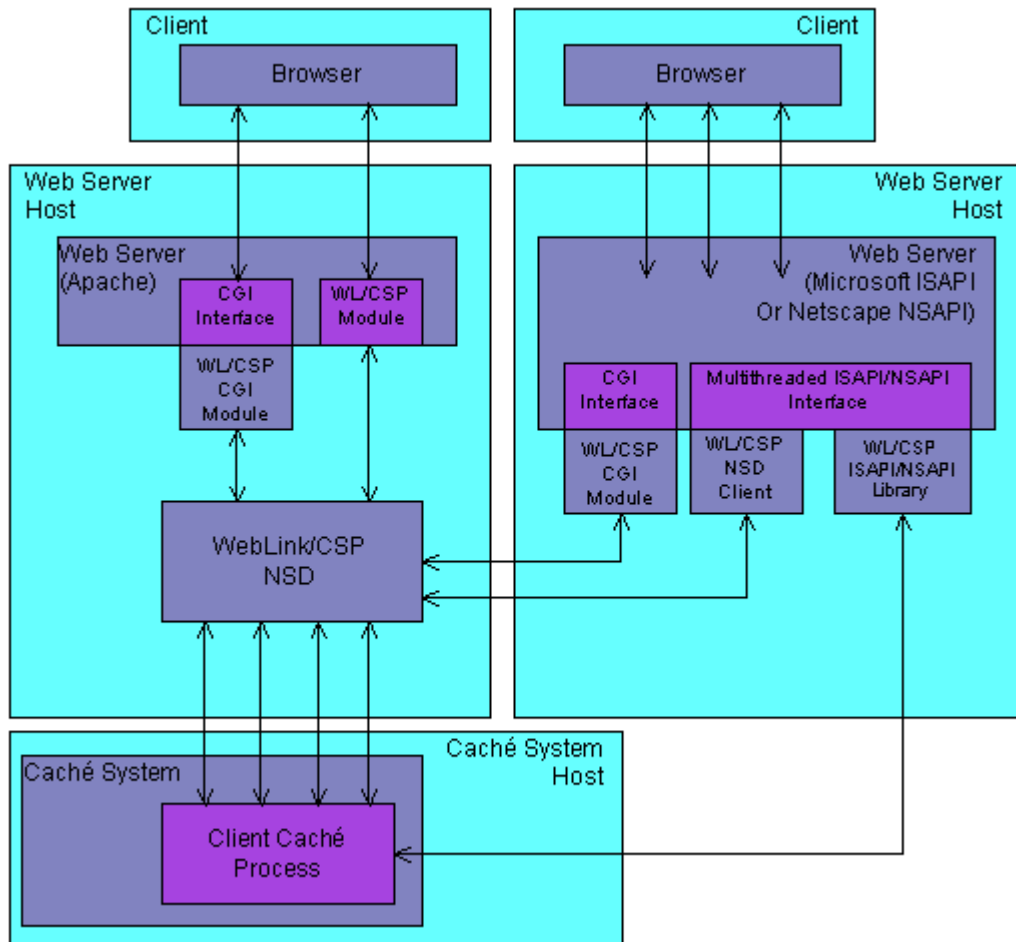
WebLink and CSP are software products that provide network-based connectivity between most commercial Web servers and the Caché application development and database environment.

WebLink and CSP implement two methods for connecting Web servers to Caché. The choice of method depends on whether or not the hosting Web server provides a multithreaded application programming interface (API):

- *Web Servers offering multithreaded APIs* (Microsoft Internet Information Services (IIS) and Personal Web Server (PWS), and the Netscape Enterprise and FastTrack servers, for example) — use a dedicated WebLink library to connect directly to the Caché environment. WebLink/CSP libraries that work with the Microsoft Internet Server Application Programming Interface (ISAPI) and the Netscape Server Application Programming Interface (NSAPI) are supplied.
- *Web Servers not offering a multithreaded API* (Apache, for example) — communicate with Caché via the WebLink/CSP network service daemon (NSD). The NSD is responsible for managing connectivity to the Caché environment. The Web Server communicates with the NSD through a small CGI (Common Gateway Interface) module, or, optionally, in the case of Apache, a module compiled into the Apache core. The Microsoft ISAPI and Netscape NSAPI Web servers also communicate with Caché through the NSD, but better performance is achieved by using the dedicated solutions. The WebLink/CSP CGI-NSD connectivity option can be used with any Web server supporting the CGI interface. The NSD does not necessarily have to run on the Web server's host machine.

The connectivity options for WebLink (WL) and CSP are illustrated in the following diagram:

Connectivity Options for CSP and WebLink



2 Problem Description

In all reported cases the classic symptoms are as follows:

- After a varying period of normal operation, IIS gets into a state whereby user requests are serviced very slowly; a drop in the overall performance of the hosting operating system is noticed.
- After operating slowly for a while, IIS becomes totally unresponsive and the operating system monitor shows IIS consuming 100% CPU time.

- Occasionally, an exception is logged in the operating system event log. The event logged usually describes a not-handled exception while processing an ISAPI extension. Sometimes the WebLink/CSP ISAPI extension is explicitly named as the culprit. Other times, the error recorded simply describes some fatal condition within the overall IIS environment. However, the timings studied in these latter cases indicate that most of these errors are caused by hands-on attempts to recover the Web server, usually by initiating a restart.

Reports from the field suggest that though it is sometimes possible to recover the Web server environment by restarting IIS, it is often necessary to completely reboot the hosting server. Clearly, both these remedies result in severe disruption to operations.

Frustratingly, and despite the intense efforts of the InterSystems support staff, it has not been possible to reproduce this problem under test conditions. The only precipitating factor appears to be the posting of large amounts of data from the Web browser to Caché. Web server or application load does not appear to be a factor; failures occur at times when the Web server is virtually idle.

3 Investigation and Analysis

The first approach in identifying a cause involved trying to identify the source of the errors recorded in the operating system event log. If these failures are caused by a mechanical defect in the Caché ISAPI DLLs, it is straightforward to quickly supply a solution.

Both WebLink and CSP have built-in exception handling that is largely focused on dealing with, and responding to, bogus and malformed Web (HTTP) requests. Indeed, it is not impossible that the Caché ISAPI DLL could potentially fail to cope with an unusual, though otherwise valid, HTTP request.

To analyze this failure, rigorous exception handling was added to every function in the DLL regardless of the level of complexity or the nature of the operation performed by individual functions. Exceptions that occurred within the context of the DLL were successfully trapped and recorded, though there were still occasions where exceptions appeared in the operating system event log. This finding indicated that the failures had to be related to a common resource shared between IIS and its ISAPI extensions.

More detailed investigation revealed that the exceptions were occurring as a result of claiming memory from the processes' primary heap. The operating system functions responsible for allocating memory (specifically **HeapAlloc**) were failing internally. There are three main possibilities:

- *Heap corruption* — Some illegal operation within the overall IIS environment could be corrupting the IIS primary heap.
- *Unserialized access to the heap* — IIS is a multithreaded environment and, as such, access to the primary heap of processes must be serialized with respect to requests from individual threads. However, it is possible to programmatically turn off serialization to improve the performance of individual claims for memory. Of course, this is an irresponsible thing to do within an environment like IIS, but it cannot be excluded. Such an operation leads to simultaneous and potentially conflicting claims on the primary heap, which eventually leads to corruption.
- *Heap fragmentation* — Web requests come in many different shapes and sizes, therefore, claims on memory within a Web server environment are likely to be very mixed in terms of the amount of memory requested in individual calls. This, together with pressure within the industry to make Web servers run as efficiently as possible, adds more stress on the operating system memory management facility. The net result is that the Web server's primary heap can become very fragmented. Fragmentation, in turn, puts extra pressure on the memory management system and makes failures due to obscure bugs more likely.

Considering these possibilities, the most likely explanation for the failures appear to be an obscure problem deep within the operating system memory management facility that is somehow provoked or exacerbated by fragmentation of the heap.

4 Remedies

As a result of the preliminary analysis, the core of the Caché ISAPI DLLs were reworked to take pressure off the memory management facility of the operating system. The following schemes have been implemented:

- *Use a separate heap* — Microsoft allows modules (DLLs in particular) to create and manage their own heap. This completely removes the need to request memory from the primary heap of the hosting process (IIS) thereby avoiding, though not completely eliminating, problems within the primary heap. The integrity of the primary heap is preserved as a result of reducing the burden placed upon it.
- *Reducing the number of calls to memory allocation functions* — reduces the pressure on the memory management system.
- *Reducing the amount of memory used* — again, reduces the pressure on the memory management system and reduces the overall resource usage.

The preceding changes have been made in a way that does not result in a loss of performance in either WebLink or CSP. Indeed, early tests with WebLink indicate a performance boost. CSP, having been reworked the same way, enjoys similar improvements.

5 A Special Note for WebLink Users

To significantly reduce the amount of memory used per request, WebLink attempts to stream the output from the Web server directly to the appropriate Caché system with minimal intermediary buffering. To do this, WebLink needs to know where to send the request as soon as possible without having to read large volumes of request data. Traditionally, the reserved form/URI variables prefixed with “MGW” (particularly *MGWLPN* and *MGWCHD*) are used to specify the target Caché server.

For HTTP **GET** methods (hyperlinks), these variables are present in the URI query string. The complete query string is ideally always available within the hosting Web server environment. For example:

```
<A HREF="/scripts/mgwms32.dll?MGWLPN=LOCAL&formID=3">
```

For HTTP **POST** methods (complete form submissions), these variables are transmitted in the URI specified within the form’s “ACTION” attribute. For example:

```
<FORM METHOD=POST ACTION="/scripts/mgwms32.dll?MGWLPN=LOCAL&formID=3">
```

Again, this is ideal because the whole query string is always available within the Web server environment. The target Caché system is determined before the incoming form data stream is read from the Web server. However, it should be noted that this technique does not work for the Opera Web browser. This particular browser fails to transmit query strings appended to URIs in the “ACTION” attribute.

An alternative method for HTTP **POST** is to include the *MGW** variables as hidden fields within the form’s data. For example:

```
<HTML>
<HEAD><TITLE>My Form</TITLE></HEAD>
<BODY>
<FORM METHOD=POST ACTION="/scripts/mgwms32.dll">
<INPUT TYPE=HIDDEN NAME="MGWLPN" VALUE="LOCAL">
.
.
.
</FORM>
</BODY>
</HTML>
```

This works well for cases where the variables are placed at the top of the form. WebLink can determine the target Caché system after reading only a small amount of incoming form data.

However, for cases where these variables are placed at the end of the form (as shown in the following example) WebLink must read and buffer the entire data stream to identify the target Caché system.

```
<HTML>
<HEAD><TITLE>My Form</TITLE></HEAD>
<BODY>
<FORM METHOD=POST ACTION="/scripts/mgwms32.dll">
.
.
.
<INPUT TYPE=HIDDEN NAME="MGWLPN" VALUE="LOCAL">
</FORM>
</BODY>
</HTML>
```

While steps have been taken to use as little memory as possible in these latter cases, it is clearly much better to always include the *MGW** variables at the top of the form.

In practice, WebLink expects to see the *MGW** variables arrive first in the data stream if it is to use the optimal method of servicing the request. It starts streaming the incoming form data directly to Caché as soon as it reads the first non-*MGW** variable. However, this exposes an ambiguity within the protocol.

When WebLink has not read any *MGW** variables in the first section of data, it has no way of knowing whether no *MGW** variables are specified (that is, the default Caché server is indicated) or whether the *MGW** variables are, in fact, at the end of the transmission. Many WebLink applications do not specify any *MGW** variables and rely on the default settings held within the configuration. For these applications to benefit from the optimal memory/request management scheme, a new configuration parameter has been introduced: *Optimise_Memory_Usage* (TRUE or FALSE).

If this parameter is set to FALSE (the default), WebLink buffers the entire request data if it cannot read the *MGW** variables either in the query string or as the first fields in the posted data stream. This allows *MGW** variables to be specified at the end of the form.

If this parameter is set to TRUE and WebLink cannot read the *MGW** variables, either in the query string or as the first fields in the posted data stream, it assumes use of the default Caché server. This mode of operation is highly recommended but should only be used if you are sure that your application complies with the requirements regarding the placement of *MGW** variables. Applications developed using WebLink Developer always place the *MGW** variables at the top of each form; therefore, these applications can safely take advantage of this mode of operation.

A warning is written to the WebLink event log if an *MGW** variable is not processed as a result of arriving at the end of the incoming data stream. For example:

```
>>> Fri Sep 07 15:46:13 2001; Thread ID: 182
```

```
WARNING: Reserved Variable 'MGWLPN' was not processed because it was not  
found amongst the first fields in the submitted data.
```

```
/scripts/mgwms32.dll?FormID=1
```

Note: The information in this section also applies to WebLink operating in PDQWeb compatibility mode. In PDQWeb mode, the critical reserved variable is *EP* as opposed to the WebLink *MGW** variables.

6 Conclusion

Tests in the field have been encouraging. Early adopters of the new DLLs have experienced no failures.

The decision to make the software changes described in this document has been made as a result of analyzing the symptoms and the event logs generated by our DLLs and the operating system. It would be ideal to be able to categorically put forward a definitive diagnosis; however, this has not been possible because of the inability to reproduce the failures under test conditions. For the same reason, it has not been possible to open a dialog with Microsoft to obtain a fix or a recommended workaround from them; historically, they only work with problems that can be demonstrated.

Long-term, InterSystems plans to offer alternative connectivity options, namely the NSD-based options shown in the diagram in the [Technology Overview](#) section. The central feature of these options is that the core WebLink/CSP functions are separated from the Web server environment. This, in itself, lowers the risk associated with open interfaces like ISAPI and provides a more stable and manageable platform for high-end Web operations. However, the all-in-one ISAPI connectivity solution continues to play an important role in small to medium-sized Web operations and InterSystems is committed to providing a robust implementation.