



Using Character Sets with CSP

Version 5.1
15 June 2006

Using Character Sets with CSP

Caché Version 5.1 15 June 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 trademarks of InterSystems Corporation.



The Ensemble product and its logos are 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

Using Character Sets with CSP	1
-------------------------------------	---

Using Character Sets with CSP

The issue of character sets and how this is handled by CSP has come up a few times and so I decided to describe the logic behind the CSP character set choices, what the options are and how it works. Hopefully this will be useful for you.

Each CSP page served Caché will include a set of HTTP headers that informs the browser about the contents of the page. The definition of these headers is contained in RFC 1945 (for the HTTP/1.0 spec which we currently support in CSP). These headers are not displayed in the browser and most browsers in fact do not give you a way of showing them at all. A good way of finding out what the headers are is to use the built in `%Net.HttpRequest` object in Caché. For example to find the headers from a typical CSP page:

```
Set http = ##class(%Net.HttpRequest).%New()  
Set http.Server = "127.0.0.1"  
Set http.Port = 1972  
Do http.Head("/csp/samples/loop.csp")  
Do http.HttpResponse.OutputToDevice()  
Do http.%Close()
```

And it gives the following output (on a Unicode Caché system):

```
HTTP/1.0 200 OK  
CACHE-CONTROL: no-cache  
CONNECTION: Close  
CONTENT-TYPE: text/html; charset=utf-8  
DATE: Mon, 13 Aug 2001 17:34:18 GMT  
EXPIRES: Thu, 29 Oct 1998 17:04:19 GMT  
PRAGMA: no-cache  
SET-COOKIE: CSPSESSIONID=260066677206262462181; path=;
```

Note that the Port is set to 1972 to use the built in mini web server in Caché, although if you have a proper web server setup this can use the default port number of 80. Another way of displaying the headers a CSP will produce is from Caché using:

```
Do ShowPage^%apiCSP("/csp/samples/loop.csp",,,1)
```

The 'Content-Type' header is always sent and this specifies the type of document the browser should expect. Normally this is “text/html” meaning that this is an HTML document so it should render it as such, but if it was a JPEG image then it will be “image/jpeg”. If the content type is “text” then you can also specify a character set to use on the same line with the “charset” modifier, for example:

```
Content-Type: text/html; charset=utf-8
```

This means that the web page that follows is encoded using utf-8, which is a mapping of the unicode standard so that ASCII codes 0-127 are mapped to utf-8 as values 0-127. The browser will lookup each character in the charset table specified in the HTTP header and draw the symbol corresponding to this character in the window.

In CSP it is important to send the correct charset header with your CSP page so that the browser can display it correctly. If you do not specify a value for the charset (I will cover how to set this later) then CSP will use the default value, on a Unicode Caché system this is “utf-8” and on an 8-bit Caché system this is the system default locale which is set using the **CNLS.EXE** utility (installed in the CacheSys/Bin directory along with the other executable files) and can be seen when you run this utility at the top of the “locale” tab. It can also be found by invoking:

```
Write ^%SYS("LOCALE", "CURRENT")
```

from the command line. To translate the name next to the drop-down dialog into the external name used in the HTTP header use the function:

```
Write $$MapExtCharset^%NLS(^%nls("Loc", locale, 0))
```

Where *locale* here is the lower case value from the CNLS.EXE utility.

If you wish to explicitly set the charset for a page rather than using the default there are several ways of doing this:

In the **OnPreHTTP** method set the CharSet property of the *%response* object:

```
Set %response.CharSet="iso-8859-1"
```

then if the content-type is a “text” type (it does not happen for other content types) it will output the “charset” header with this value. This method is generally not recommended as the input can not be translated, it is suggested that you use one of the other mechanisms. Use the `<jsp:content charset="iso-8859-1">` tag in the CSP page. This is matched by a CSP system rule and sets the *CHARSET* parameter of the class that is compiled from this csp page. Then when the page is rendered the *%response.CharSet* value is initialized to the *CHARSET* parameter of this page and so this is the value written out for the charset if the content-type is a 'text' type. Use the `<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">` tag in the `<head></head>` section of the CSP page. This is picked up by a system rule and gets converted into the *CHARSET* parameter in the class that is compiled so the end result is exactly the same as option 1.

Create a class that derives from *%CSP.Page* and in this class specify the charset that you wish to use for the parameter *CHARSET*. On the **Edit CSP Application** in the System Management Portal (**[Home]** > **[Security Management]** > **[CSP Applications]** > **[Edit CSP Application]**),

in the **Default Superclass** field, set the superclass for this application to be the new class you have created. Then recompile all the pages in your CSP application and they will now inherit from this superclass and so they will automatically pick up this default *CHARSET* parameter. This is a useful way to set the charset for a large number of pages at the same time.

Besides putting the “charset” entry in the HTTP header setting this also effects how Caché translates any characters it writes out to the browser and how it interprets input it receives from the browser, either as parameters in the url or from an HTTP post. CSP will always output the HTTP headers using utf-8 because before the browser reads the 'charset' header there is no way it can know what the charset should be and if the RFC ever allows characters other than ASCII in the HTTP header the only logical choice is utf-8 as the ASCII characters are all preserved using this encoding. Once the headers are output CSP then switches to output the rest of the page in the charset specified in the header using the call:

```
Write $$SetIO^%NLS(tablename)
```

This function returns the old table that was in use and changes the current device to use the new table. This tablename is the Caché internal name of the table, for example to turn translation off it is “RAW” and to use utf-8 it is “UTF8” . If you want to find the name of the current table use the following:

```
Write $$GetIO^%NLS()
```

On a Unicode version of Caché if it is using the default charset then this will force all characters output to be translated to their utf-8 representation. For an 8-bit Caché if it is using the default which is the locale specified in Caché then it will not translate the characters at all and so will use the “RAW” table. This is because the characters in the Caché database are already in the default locale (because that is the meaning of the default locale) and so it would just be translating from the default table to the default table and so no translation is needed.

If you specify a charset to use then Caché will pick the NLS table so that the characters stored inside the Caché database will be converted to characters in this specified charset. To find the name of the internal Caché table from the charset name use the function:

```
Write $$MapCharset^%NLS(charset)
```

If when displaying a page from CSP you get a “Character Set TEST not installed” error message then it means that you are requesting a charset that Caché does have available. For example if you invoke:

```
Write $$MapCharset^%NLS("TEST")
```

then it returns "" because there is no such charset as “TEST” . When you get this error you have two choices, either load the table into Caché using the **CNLS.EXE** utility (or create a

table with this name if we do not supply this table) or change the CSP page to use another charset that is supported.

Any characters that can not be converted using the specified table are output as “?” or whatever error character is specified in the **CNLS.EXE** utility. So if you start seeing “?” appearing in either your web pages or in the data posted back to Caché then it is likely that there are characters that were not translated correctly.

When we receive data back from the web browser in the url or in an HTTP post CSP will convert the characters from the external charset to the Caché default locale. As the HTTP headers sent by the web browser do not specify which charset the browser is using (a fairly serious oversight in the HTTP/1.0 and 1.1 specs) we are forced to make an estimated guess as to which charset to use, that in practice appears to work well. We assume that the charset sent by the browser is the same as the charset specified on the page that it is submitting the information to. So if I have a form.csp page that uses charset 'iso-8859-1' and it has a form that submits the information to result.csp which also uses charset “iso-8859-1” then we will convert the information sent by the browser from the charset of the result.csp page, i.e. “iso-8859-1”, into the Caché default locale. From testing the major browsers it seems that if you supply a page with a particular charset then any information submitted by the browser from this page is encoded using this same charset. This means that you should make sure that both the page that is going to submit information to Caché and the page that receives this information must be using the same charset, however as entire applications tend to use the same charset this is not a difficult requirement. CSP converts strings from the external to the internal charset using the **\$ZConvert** function:

```
Write $ZConvert(string,"I",table)
```

Where *string* is the string to convert and *table* is the Caché internal table name.

The one last piece to this puzzle is how CSP reads in the csp files from the filesystem and how this deals with this charset issue. The compiler reads csp files using the %FileCharacterStream object. It first checks if the file is a Unicode file and if so whether it is big or little Endian. This done by looking at the leading two bytes in the file, if they are \$char(255,254) or \$char(254,255) then this is a Unicode file and it reads it in as such. If the file is not a Unicode file then it uses the default translation setup in **CNLS.EXE** for reading and writing to files from Caché (on the “Locale” tab under the translation type for sequential devices).

So for example assume you have a Unicode Caché system and wish to use this in Japan where the charset will be Shift-JIS. When the CSP compiler loads the csp file it will use the default sequential device translation which will be setup to convert from Shift-JIS to Unicode and so the CSP class compiled in Caché will be stored internally in Unicode. Using one of the mechanisms described above you tell CSP to output this page with Shift-JIS charset, so when

a request comes in for this page CSP sets the translation on the TCP/IP device to convert from the Unicode internal representation into Shift-JIS externally and it will also add the HTTP header 'charset=Shift-JIS' so the browser will interpret the page correctly. If this page contains a form where some Shift-JIS characters are entered and then submitted the data sent by the browser will be using the Shift-JIS charset. As the page it is being submitted to also has a charset of Shift-JIS then CSP will convert from Shift-JIS back to the internal Unicode representation of this character. Note that there is a slight logical inconsistency in the <meta http-equiv> method of setting the charset in a csp file. If you create a html file with a <meta http-equiv> tag in it on a web server then when this page is output to a browser the browser will see this <meta http-equiv> tag and interpret the data in the page as being from the charset specified in this tag, just as if the 'Content-Type' HTTP header had included the 'charset'. So it tells the browser the charset of the html file. However if you put a <meta http-equiv> tag in a csp file then this will be loaded into Caché using the default device translation for a sequential file and then when it is output to the browser it will convert from the Caché default locale to the charset specified in the <meta http-equiv> tag. So an HTML editing tool that saves a page with a <meta http-equiv> specifying a charset of 'utf-8' will save the whole page as utf-8. When Caché loads this page it should translate from utf-8 to the Caché default internal locale however it just uses the default device translation instead which will cause problems if the device translation does not match the charset of the page. Often the charset specified in the <meta http-equiv> is the same as the default device translation (e.g. both are Shift-JIS on a Japanese system) and so this will work correctly.

If you wish to get around the translations of the page on output to the browser and on input from the browser we also provide a parameter of the compiled class called *NOCHARSETCONVERT* if this is set to 1 then CSP will still output the charset in the HTTP "Content-Type" header but it will not set the device translation to convert from the Caché default locale to this charset and it will not convert any values sent by the browser from this charset to the Caché default locale. This means that all charset conversion issues must be handled by your application rather than being done by the CSP engine itself. You can also set the *NOCHARSETCONVERT* option in a csp page with the tag <csp:content nocharset-convert="1">.

NOCHARSETCONVERT was developed for situations such as Thailand that have no locale defined and therefore use English default locale but actually store 8-bit Thai characters in the database.

