



Using CNLS to Set Up Customized National Language Translations

Version 5.1
15 June 2006

Using CNLS to Set Up Customized National Language Translations

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 CNLS to Set Up Customized National Language Translations	1
1 NLS TRANSLATION	1
2 Additional Notes	3
2.1 Entry Points for General Functionality	5
2.2 Entry Points for Modal Transactions	8
2.3 Entry Points for Managing Replacement Attributes for I/O Translations	9

Using CNLS to Set Up Customized National Language Translations

This document describes how to use the cnls utility (cnls.exe), detailing how to set up customized language translation, test it, and apply it to your application.

1 NLS TRANSLATION

To set up a customized NLS translation, follow these steps:

1. Start up the NLS utility by double-clicking the cnls.exe application (alphabet block icon), located in the CacheSys\Bin directory.
2. From the Choose Server Connection window, either select the default LOCALTCP or select another server from the pull-down list. Click OK.
3. This displays the Caché National Language Support window. Select the Locale tab. View the existing locales using the Locale pull-down list. Select the locale that most closely resembles the new locale you intend to create. Begin the process of creating this new locale by clicking "New" in the lower right corner.

This displays the Add an NLS Locale window:

- Description: Type a text description of the new locale.
- Copy From: Select an existing locale from the pull-down list to serve as a template. For example, "English, United States, Latin1 (ISO 8859-1)."
- Character Set: Select a character set from the pull-down list or type in a character set. For example, "Latin1." The default value is the character set for the locale from which you are copying.
- Locale Name: Specify a four-character code. A customized locale name should always begin with "y" and end in either "8" for 8 bit applications, or "W" for Unicode.

When you have specified all of the above, the **OK** button in the lower right corner should no longer be grayed out. Click **OK**.

This displays the Caché National Language Support window for your new locale. Choose the Options menu located at the top left of the window and select "Install Locale." Alternatively, you can use the NLS "Import Locale" option to import a locale from another Caché system.

Note: If selecting another standard locale as your current locale, you must use the NLS utility "Import Locale" option to copy the locale definition from the Caché installation CD into the CacheSys\Bin directory. Standard locales are located in the \nt\install\eightbit or \nt\install\unicode directories. Locale definitions are the .goq files. Locales are identified by a four-character locale name. Always import the _src.goq (source code) version of the locale. For example, "English, United States, Latin1 (ISO 8859-1)" is enu8_src.goq. The Install Locale option also prompts you for these files from disk. Browse to the CacheSys\Bin directory or the NLS subdirectory.

4. Go to the Translation Tab and set up the Translation Table choosing "New" in the lower right corner "Copy From," is left blank. i.e. For printing use an "Output Only" table where "Translate From" would be "Latin 1" and "Translate To" is the new translation table which should always start with "y". If the translation is for terminal input, it would be an "Input Only" table and "Translate From", would be the new translation table and "Translate To" would be "Latin 1". "Mapping Type" is primarily "Single to Single" unless you have multiple characters you want to translate into one, or multiple to multiple which would then be a "Multiple to Single" or "Multiple to Multiple" table. The characters are translated in the table on on the far right. By default they are displayed in decimal, which are the ascii values. The table can also be displayed in hex or character mode. This is adjusted in the top left corner under the "Help" option. To set up your translation, you simply change the values in either row or add new values at the bottom next to the *.
5. Once the translation table is set up go back to the "Locale Tab" to update it with the new translation table. In the "Translation Name Grid (the middle grid), go to the * to add your translation. Here you can call it anything you want. i.e. "sueprint" Then move over to the "Output" tab, or "Input" tab, whichever you are using, the drop down list should contain your new translation table, which you then choose. Depending on what you are translating "input" vs "output", you should always have a default table as a placeholder in either the "input" or "output" column. The Translation Name name is critical, as this is the name used by Caché to specify I/O translation. Each translation name is associated with an Output and an Input character set.
6. Go to the "Translation Type" grid (the bottom grid), and change the "Programmer Window" to your new "Translation Name" i.e. "sueprint" (the name you assigned in the second grid) so that you can test the translation. The "Translation Name" is the name used by Caché to specify I/O translation. Each translation name is associated with an Output an Input character set.

7. Install or Validate the locale again with the new translation. Validate compiles the sources for the tables and loads the objects into shared memory. Install validates the locale and makes it the current one.
8. Open up a new terminal window:

```
w $zm
```

should list the translation name that you updated your locale with.

```
>w $zm
>RY\Latin1\K\SUEPRINT
```

Note: i.e. Translation Name is "sueprint"

2 Additional Notes

To set translation for a printer, you set the translation when the device is opened.

```
u x s zzz=$$SetIO^%NLS("sueprint")
```

Use the following syntax to test it:

```
>s x="|PRN|"
```

```
>o x
```

```
>u x s zzz=$$SetIO^%NLS("sueprint") w $C(146),!, $C(147),!
```

```
>c x
```

The **\$ZCVT** function allows you to test your translation with the following syntax where "ySue" is the output translation table.

```
>f I=0:1:255 s x=$ZCVT($C(i),"o","ySue") if x'=$c(i) w I," ", $a(x),!
>177 49
 178 50
 180 52
 183 55
 184 56
 231 103
```

These are the translated characters.

The **Do Dump^%NLSMISC** command lists the %NLS settings. An example of the sections of this appears below.

Additional Notes

I/O Table Name	Offset:Type	Source Global
UnicodeLittleIn	4A0030:Struct8	
Out	4A0038:Struct8	
UnicodeBig In	4A0020:Struct8	
Out	4A0028:Struct8	
SAME In	4A0000:Struct8	
Out	4A0008:Struct8	
UTF8 In	4A0AAC:Struct8	%nls("Src", "XLT", "UTF8", "Latin1")
Out	4A0AB8:Struct8	%nls("Src", "XLT", "Latin1", "UTF8")
BIN In	4A0000:Struct8	
Out	4A0008:Struct8	
RAW In	4A0000:Struct8	
Out	4A0008:Struct8	
Ricky Out	4A0098:Struct8	%nls("Src", "XLT", "Latin1", "yAquis")

\$X/\$Y Table Name	Offset:Type	Source Global
Latin1	4A0074:Struct8	

Collation Table	Offset:Type	Source Global
0 Enc	Old ANSI	
Dec		
1 Enc	New ANSI	
Dec		
2 Enc		
Dec		
3 Enc		
Dec		
4 Enc		
Dec		
5 Enc	Unicode	
Dec		
128 Enc	Old string	
Dec		
129 Enc	New string	
Dec		
133 Enc	Unicode string	
Dec		

Default	Offset:Type	Source Global
Pattern	4A0040:Struct8	
Ident	4A004C:Struct8	
UpperCase	4A0080:Struct8	
LowerCase	4A008C:Struct8	
TitleCase	4A0080:Struct8	

I/O Default	Offset:Type	Name
Process	490233:ByteStr	Ricky
Direct Term	490233:ByteStr	Ricky
Telnet/LAT	490002:ByteStr	RAW
File	490002:ByteStr	RAW
Magtape	490002:ByteStr	RAW
Network	490002:ByteStr	RAW
DSM-DDP	490002:ByteStr	RAW
DTM-DCP	490002:ByteStr	RAW
SysCalls	490002:ByteStr	RAW

<code>\$X/\$Y Default</code>	<code>Offset:Type</code>	<code>Name</code>
<code>Process</code>	<code>49002D:ByteStr</code>	<code>Latin1</code>
<code>I/O Slots</code>	<code>All Empty</code>	
<code>\$X/\$Y Slots</code>	<code>All Empty</code>	
<code>Process</code>	<code>Offset:Type</code>	<code>Source Global</code>
<code>Pattern</code>	<code>4A0040:Struct8</code>	
<code>Ident</code>	<code>4A004C:Struct8</code>	
<code>UpperCase</code>	<code>4A0080:Struct8</code>	
<code>LowerCase</code>	<code>4A008C:Struct8</code>	
<code>TitleCase</code>	<code>4A0080:Struct8</code>	
<code>Process I/O</code>	<code>Offset:Type</code>	<code>Name</code>
<code>Process</code>	<code>490233:ByteStr</code>	<code>Ricky</code>
<code>Direct Term</code>	<code>490233:ByteStr</code>	<code>Ricky</code>
<code>Telnet/LAT</code>	<code>490002:ByteStr</code>	<code>RAW</code>
<code>File</code>	<code>490002:ByteStr</code>	<code>RAW</code>
<code>Magtape</code>	<code>490002:ByteStr</code>	<code>RAW</code>
<code>Network</code>	<code>490002:ByteStr</code>	<code>RAW</code>
<code>DSM-DDP</code>	<code>490002:ByteStr</code>	<code>RAW</code>
<code>DTM-DCP</code>	<code>490002:ByteStr</code>	<code>RAW</code>
<code>SysCalls</code>	<code>490002:ByteStr</code>	<code>RAW</code>
<code>Process \$X/\$Y</code>	<code>Offset:Type</code>	<code>Name</code>
<code>Process</code>	<code>49002D:ByteStr</code>	<code>Latin1</code>

To check the locale setting do the following:

```
>d ^%G
>Global ^%SYS("LOCALE","CURRENT") -- NOTE: translation in effect
>^%SYS("LOCALE","CURRENT")=yaq8
```

2.1 Entry Points for General Functionality

The following is a list of the entry points in `^%NLS`. Each entry point is in the form **EntryPoint(Arg)**, where the full name of the call is **\$\$EntryPoint^%NLS(Arg)**.

- `DefCOL` — Return system default
- `DefIO[(type)]` — Return system default, where valid values of *type* are 0 for process; 1 for M terminal (default type); 2 for other terminal; 3 for sequential file; 4 for mag tape; and 5 for network device
- `DefIdent` — Return system default
- `DefLower` — Return system default
- `DefPM` — Return system default
- `DefTitle` — Return system default

- DefUpper — Return system default
- DefXY — Return system default
- Def("C") — Collation table name
- Def("E") — Identifier table name
- Def("I"[,type]) — I/O Translation table name (see DefIO for types)
- Def("L") — Lower Case Conversion tbl name
- Def("P") — Pattern Match table name
- Def("T") — Title Case Conversion tbl name
- Def("U") — Upper Case Conversion tbl name
- Def("X") — \$X/\$Y Action table name
- GetCOL — Return Collation
- GetCursorFlag — Get current value of SetCursorFlag
- GetIO — Return I/O Translation
- GetIdent — Return Identifier
- GetLower — Return Lower Case Conversion
- GetPDefIO[(type)] — Return process default
- GetPDefXY — Return process default
- GetPDef("I"[,type]) — I/O Translation table name (see DefIO for types)
- GetPDef("X") — \$X/\$Y Action table name
- GetPIO — Return I/O Translation
- GetPM — Return Pattern Match
- GetPitch — Get pitch value for current device
- GetTitle — Return Title Case Conversion
- GetUpper — Return Upper Case Conversion
- GetXY — Return \$X/\$Y Action
- Get("C") — Specify the table name for calling process
- Get("E") — Specify the table name for calling process
- Get("I") — Specify the table name for current device

- Get("J") — Specify the table name for calling process
- Get("L") — Specify the table name for calling process
- Get("P") — Specify the table name for calling process
- Get("T") — Specify the table name for calling process
- Get("U") — Specify the table name for calling process
- Get("X") — Specify the table name for current device
- OffIO — Turn I/O xlate OFF for current device and return previous state
- OffPIO — Turn I/O xlate OFF for calling process and return previous state
- OnIO — Turn I/O xlate ON for current device and return previous state
- OnPIO — Turn I/O xlate ON for calling process and return previous state
- SetCOL(tblname) — Select Collation
- SetCursorFlag(0) — Makes current device compatible with the logical setting on the terminal
- SetCursorFlag(1) — Makes current device compatible with the physical setting on the terminal
- SetIO(tblname) — Select I/O Translation
- SetIO(tblname) — Select I/O Translation
- SetIdent(tblname) — Select Identifier
- SetLower(tblname) — Select Lower Case Conversion
- SetPDefIO(tblname[,type]) — Select process default
- SetPDefXY(tblname) — Select process default
- SetPDef("I",tblname[,type]) — I/O Xlate table name (see DefIO for types)
- SetPDef("X",tblname) — \$X/\$Y Action table name
- SetPIO(tblname) — Select I/O Translation
- SetPM(tblname) — Select Pattern Match
- SetPitch(value) — Set pitch value for current device
- SetTitle(tblname) — Select Title Case Conversion
- SetUpper(tblname) — Select Upper Case Conversion

- SetXY(tblname) — Select \$X/\$Y Action
- Set("C",tblname) — Specify the table for calling process
- Set("E",tblname) — Specify the table for calling process
- Set("I",tblname) — Specify the table for current device
- Set("J",tblname) — Specify the table for calling process
- Set("L",tblname) — Specify the table for calling process
- Set("P",tblname) — Specify the table for calling process
- Set("T",tblname) — Specify the table for calling process
- Set("U",tblname) — Specify the table for calling process
- Set("X",tblname) — Specify the table for current device
- StatIO — Return I/O xlate state for current device
- StatPIO — Return I/O xlate state for calling process

2.2 Entry Points for Modal Transactions

The following operations are for modal translations only:

- GetInpMode() — Get input mode string index for current device
- GetInpModeStr(nbr) — Get input mode string value for current device
- GetOutMode() — Get output mode string index for current device
- GetOutModeStr(nbr) — Get output mode string value for current device
- GetPInpMode() — Get input mode string index for calling process
- GetPInpModeStr(nbr) — Get input mode string value for calling process
- GetPOutMode() — Get output mode string index for calling process
- GetPOutModeStr(nbr) — Get output mode string value for calling process
- SetInpMode(nbr) — Set input mode string index for current device
- SetInpModeStr(nbr,value) — Set input mode string value for current device; that is, mode(nbr) = value
- SetOutMode(nbr) — Set output mode string index for current device
- SetOutModeStr(nbr,value) — Set output mode string value for current device

- `SetPInpMode(nbr)` — Set input mode string index for calling process
- `SetPInpModeStr(nbr,value)` — Set input mode string value for calling process
- `SetPOutMode(nbr)` — Set output mode string index for calling process
- `SetPOutModeStr(nbr,value)` — Set output mode string value for calling process

2.3 Entry Points for Managing Replacement Attributes for I/O Translations

These functions set or get replacement attributes for I/O translations. These functions return a replacement status, where:

- 0 means normal (no substitutions)
- 1 means substitution has occurred

Some of the functions accept arguments, which are either:

- `ReplType` — The replacement type for characters/strings which do not have a valid translation, where valid values are:
 - 0 = generate error
 - 1 = substitute (value or string)
 - 2 = ignore (use original character)
- `ReplVal` — The value/string to be used if `ReplType=1` (substitute)

The functions are:

- `GetInpReplStat` — Get input replacement status for current device
- `GetInpReplType` — Get input replacement type for current device
- `GetInpReplVal` — Get input replacement value for current device
- `GetOutReplStat` — Get output replacement status for current device
- `GetOutReplType` — Get output replacement type for current device
- `GetOutReplVal` — Get output replacement value for current device
- `GetPInpReplStat` — Get input replacement status for process
- `GetPInpReplType` — Get input replacement type for process

- GetPInpReplVal — Get input replacement value for process
- GetPOutReplStat — Get output replacement status
- GetPOutReplType — Get output replacement type for process
- GetPOutReplVal — Get output replacement value for process
- SetInpReplStat(ReplType) — Set input replacement status for current device
- SetInpReplType(ReplType) — Set input replacement type for current device
- SetInpReplVal(ReplVal) — Set input replacement value for current device
- SetOutReplStat(ReplType) — Set output replacement status for current device
- SetOutReplType(ReplType) — Set output replacement type for current device
- SetOutReplVal(ReplVal) — Set output replacement value for current device
- SetPInpReplStat(ReplType) — Set input replacement status for process
- SetPInpReplType(ReplType) — Set input replacement type for process
- SetPInpReplVal(ReplVal) — Set input replacement value for process
- SetPOutReplStat(ReplType) — Set output replacement status for process
- SetPOutReplType(ReplType) — Set output replacement type for process
- SetPOutReplVal(ReplVal) — Set output replacement value for process

The “Set” functions typically return the prior setting, when this value is meaningful.