Overview
Classes
Environment
Errors
Debugging
Other Tasks
Copyrights
Feedback

Caché includes an environment for using objects interactively as well as an API for compiling and manipulating objects and class definitions. This utility is called the Caché Object Utility Library implemented as a calls within %apiOBJ.

This utility allows you to perform the following tasks:

- Manipulate classes
  - Load classes
  - Compile classes
  - Export classes
  - Delete classes
- Manipulate Environment Settings
  - Manipulate Default Flags
  - Manipulate Storage Strategies
  - Manipulate Default Locales
- Error Handling
  - Add an error mssage
  - Display error text
- Debugging Object Applications
  - Display details of an object
  - List all objects in the current process
  - Kill all objects in the current process
- Other Tasks
  - Access a method indirectly
  - Get the value of a property
  - Set the value of a property
  - Get the version number
  - Manipulate concurrency settings
  - Manipulate transaction settings

Revision Date:
May 13, 1999

Revision Date:
May 13, 1999

You can use the Caché Object Utility Library to manipulate classes in four ways:

- Load classes
- Compile classes
- Export classes
- Delete classes

Loading Classes

You can use the Caché Object Utility Library to load classes from CDL files. You can load all classes in a single CDL file or load all classes in all CDL files within a single directory.

Loading All Classes in a Single CDL File

You can use the following command to load all classes in a single CDL file:

```
Do LoadFile^%apiOBJ(filename,flag,.errorlog)
```

This command loads one CDL file into the class dictionary. *filename* is the name of the CDL file to load into class dictionary and optionally compile, depending on the value of *flag*. *flag* and *errorlog* are described in their own sections elsewhere.

Loading all classes in all CDL files in a Single Directory

You can use the following command to load all classes in all CDL files in a single directory:

```
Do LoadDir^%apiOBJ(dir,flag,.errorlog)
```

Loads all CDL files in a directory into the class dictionary. *dir* is the path for a directory containing CDL files. All the CDL files in this directory will be loaded into the class dictionary and optionally compiled, depending on the value of *flag*. *flag* and *errorlog* are described in their own sections elsewhere.

Compiling Classes

You can use the Caché Object Utility Library to compile classes loaded in the Class Dictionary. Classes can also be loaded and compiled in a single command, as described in Loading Classes and The Flag Argument.

You can compile a single class, compile a list of classes, compile all classes in a namespace, compile all system classes, or compile all classes in a namespace and all system classes.

Compiling a Single Class

You can use the following command to compile a single class already loaded into the Class Dictionary:

```
Do Compile^%apiOBJ(classname,flag,.errorlog)
```

Compiles the class *classname*. It is assumed that class *classname* is defined in the class dictionary. *flag* and *errorlog* are described in their own sections elsewhere.

Compiling a List of Classes

You can use the following command to compile a list of classes already loaded into the Class Dictionary:

```
Do CompileList^%apiOBJ(classlist,flag,.errorlog)
```

Compiles the list of classes specified in the comma-delimited string *classlist*. *flag* and *errorlog* are described in their own sections elsewhere.

Compiling All Classes in a Namespace

You can use the following command to compile all classes in a namespace already loaded into the Class Dictionary:

```
Do CompileNameSpace^%apiOBJ(flag,.errorlog)
```

flag and *errorlog* are described in their own sections elsewhere.

Compiling All System Classes

You can use the following command to compile all system classes already loaded into the Class Dictionary:

```
Do CompileSystem^%apiOBJ(flag,.errorlog)
```

Compiles all system classes. These are classes defined with the System keyword. *flag* and *errorlog* are described in their own sections elsewhere.

Compiling All Classes

You can use the following command to compile compile all classes in a namespace and all system classes already loaded into the Class Dictionary:

```
Do CompileAll^%apiOBJ(flag,.errorlog)
```

flag and *errorlog* are described in their own sections elsewhere.

Exporting Classes

The Caché Object Utility Library can export classes from the Class Dictonary into four different types of text files: CDL files, Java Files, ODL files, and TLB files.

Exporting Classes to CDL Files

You can use the following command to export classes from the Class Dictionary to a CDL file:

Do ExportCDLList^%apiOBJ(*classlist*,*file*,*flag*,.*errorlog*)

Exports all of the classes listed in the comma-delimited string *classlist* to the CDL file *file*. This file can then be loaded and compiled into any Caché Objects system. *flag* and *errorlog* are described in their own sections elsewhere.

Exporting Classes to Java Files

You can use the following command to export classes from the Class Dictionary to Java files:

Do ExportJavaList^%apiOBJ(*classlist*,*dir*,*flag*,.*errorlog*)

Exports all of the classes listed in the comma-delimited string *classlist* to the directory *dir*. This results in a file for each class called *classname.java* in that directory. *flag* and *errorlog* are described in their own sections elsewhere.

Exporting Classes to ODL Files

You can use the following command to export classes from the Class Dictionary to an ODL file:

Do ExportODLList^%apiOBJ(*classlist*,*file*,*flag*,.*errorlog*)

Exports all of the classes listed in the comma-delimited string *classlist* to the ODL file *file*. This file can then be used in C++. *flag* and *errorlog* are described in their own sections elsewhere.

Exporting Classes to TLB Files

You can use the following command to export classes from the Class Dictionary to a TLB file:

Do ExportTLBList^%apiOBJ(*classlist*,*file*,*flag*,.*errorlog*)

Exports all of the classes listed in the comma-delimited string *classlist* to the Type Library *file*. This file can be used directly by C++. *flag* and *errorlog* are described in their own sections elsewhere.

Deleting Classes

You can use the Caché Objects Command Level Utility to delete classes from the Class Dictionary. You can delete a single class or delete all classes in a namespace.

Deleting a Single Class

You can use the following command to delete a single class from the Class Dictionary:

Do Delete^%apiOBJ(classname)

Deletes the class *classname* and its generated routines from the current namespace.

Deleting All Classes in a Namespace

You can use the following command to delete all classes in a namespace from the Class Dictionary:

Do DeleteAll^%apiOBJ()

Deletes all classes and their generated routines in the current namespace.

The value of *flag* is a string expression containing a series of one or more single character compiler options.

The options for *flag* include:

Code	*Code Name*	*Usage*
a	Application	Specifies to compile non-system classes. This code is ignored for Export* commands.
b	Compile Subclasses	Specifies to compile subclasses of the specified classes.
c	Compile	For the Load* commands this specifies that the class or classes should also be compiled. This code is ignored for the Compile* and Export* commands.
d	Display	The progress of the load, compile, or export is displayed to the screen.
f	Force Compilation	The specified classes will be compiled regardless of whether any objects are in use. If objects of the classes being compiled are open, the will be marked invalid and users will get <CLASS COMPILING> or <CLASSES RECOMPILED> errors.
k	Keep Source	All code generated by the classes during compilation will be stored in Caché. By default, source code is deleted from Caché once compilation is successfully completed. This code is ignored for the Export* commands.

Revision Date:
May 13, 1999

errorlog returns from 0 to *n* error messages in the following format:

```
errorlog=n
errorlog(1)=errormsg1
errorlog(2)=errormsg2
...
errorlog(n)=errormsgn
```

Revision Date:
May 13, 1999

p	Compile %* classes	Compile classes that start with a percent sign (``%"). This flag is used in conjunction with the ``s" flag to compile system classes in the CACHELIB namespace under the direction of the InterSystems Worldwide Response Center.
r	Recursive	All superclasses of the specified classes will be compiled as well as all of the data type classes used by these classes. System classes will only be compiled if the s code is set. This code is ignored for the Export* commands.
s	System	Compiles system classes. This code is ignored for the Export* commands.
u	Update	Only classes that need to be compiled will be compiled. If the class has not changed since last compilation, it will not be compiled. This code is ignored for the Export* commands.
v	Keep objects valid	Any open objects will remain valid after the compilation. These objects should be used with caution since the in-memory structure of the objects may have been changed by the new compilation. This code is ignored for the Export* commands.

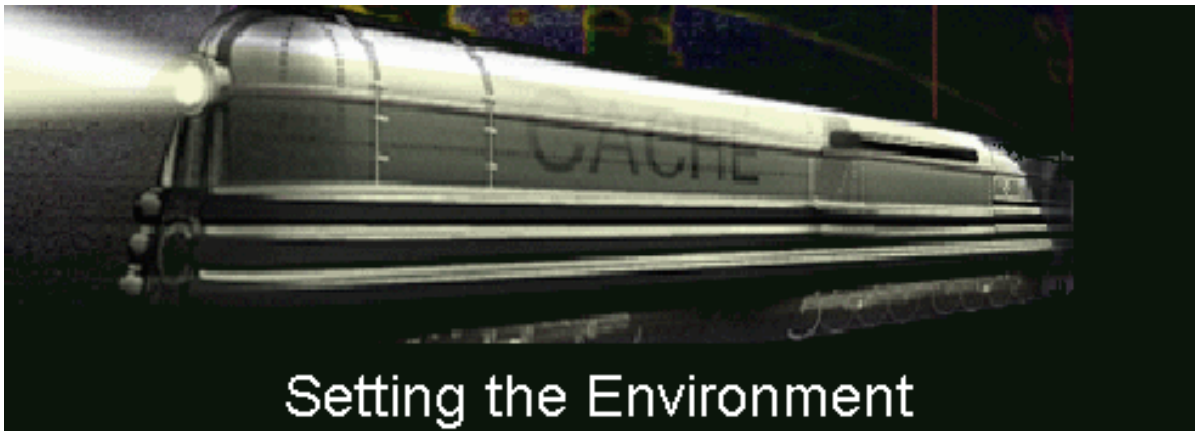
The *flag* options are not case sensitive. By default, *Caché Objects* compiles application classes and displays status information during loading and compilation. The Load* commands do not compile by default. You must use the ``c" option if you want a Load* command to compile.

You can suppress a code by preceding it with ``-``. For example, a flag setting ``cr-d" would cause a recursive compile without displaying any status information.

Examples:

```
Do LoadFile^%apiOBJ("C:\MyClasses\Car.cdl","cr")

Do LoadDir^%apiOBJ("C:\MyClasses","c")
```



The Caché Object Utility Library provides a mechanism for customizing your Caché environment using the SetEnvironment^%apiOBJ() call. You can also determine the current environment settings by using GetEnvironment^%apiOBJ.

Currently, you can:

- Set default compilation flags
- Set storage strategy aliases
- Set the default locale for error messages

Revision Date:
May 13, 1999

You can use the Caché Object Utility Library to set or display your own default compilation flags. Any of the supported flags can be specified on by default.

Assigning the Default Flags

You can use the following command to assign default compilation flags for your system:

```
Do SetEnvironment^%apiOBJ($$$cENVdefaultflags,myflags)
```

Displaying the Default Flags

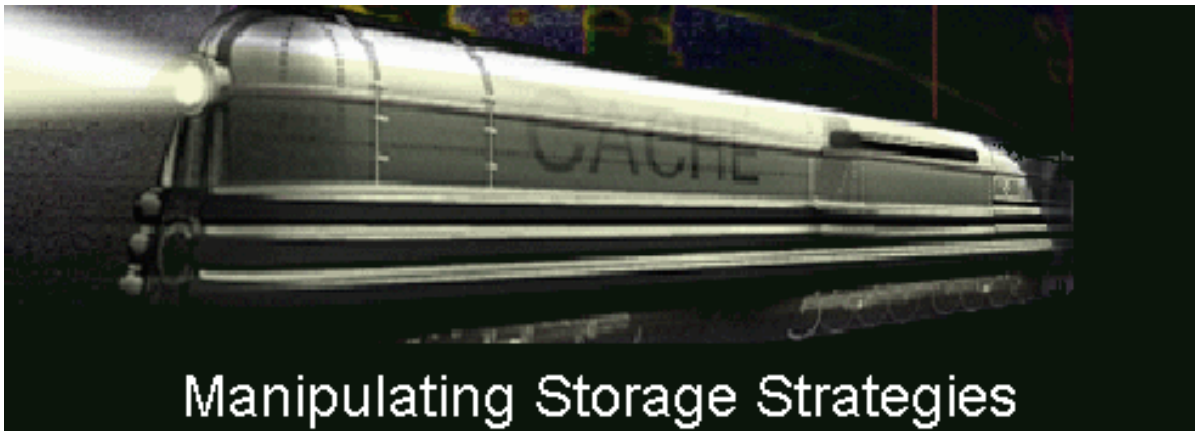
You can use the following command to display the current default flags:

```
Do GetEnvironment^%apiOBJ($$$cENVdefaultflags,.myflags)
```

This returns the current default flags, *myflags*.

Revision Date:
May 13, 1999

Site Map



Overview
Classes
Environment
Errors
Debugging
Other Tasks
Copyrights
Feedback

You can use the Caché Object Utility Library to assign the storage strategy associated with a particular storage strategy alias or to display the storage strategy currently associated with a particular storage strategy alias..

Assigning a Storage Strategy

You can use the following command to assign a storage strategy to a particular storage strategy alias:

Do SetEnvironment^%apiOBJ($$$cENVstoragestrategy,*aliasname*,*storagestrategy*)

This assigns the storage strategy *storagestrategy* to the alias *aliasname*. *aliasname* must start with $. *storagestrategy* must be defined in every class which defines its storage strategy as *aliasname*.

Displaying a Storage Strategy

You can use the following command todisplay the storage strategy associated with a particular storage strategy alias:

Do GetEnvironment^%apiOBJ($$$cENVstoragestrategy,*aliasname*,.*storagestrategy*)

This returns the storage strategy *storagestrategy* associated with the alias *aliasname*.

Revision Date:
May 13, 1999

You can use the Caché Object Utility Library to set or display the current default locale. This locale will be used to display error messages. On Windows, the default locale will be used only if the error messages are not available in the locale used for Windows.

Assigning the Default Locale

You can use the following command to assign default compilation flags for your system:

```
Do SetEnvironment^%apiOBJ($$$cENVdefaultlocale,mylocale)
```

where *mylocale* is the three letter code for the desired locale.

Displaying the Default Flags

You can use the following command to display the current default locale:

```
Do GetEnvironment^%apiOBJ($$$cENVdefaultlocale,mylocale)
```

This returns the three letter abbreviation for the current default locale, *mylocale*.

Revision Date:
May 13, 1999

The Caché Object Utility Library provides several calls to add user-defined error messages to Caché and to display the error messages returned within status codes:

- Add new error messages
- Display error messages

Add Error Messages to Caché

You can use the Caché Object Utility Library to add new error messages to Caché using the following call:

```
DO AddError^%apiOBJ(errorname,errorcode,errortext,locale))
```

where *errorname* is the name of the error, *errorcode* is the number assigned to the error, *errortext* is the error message, and *locale* is the locale for the error.

The error name becomes a macro used to return the error from methods that return status. User-defined errors must have numbers above 10000. You must separately define the error in each locale you wish to use it in.

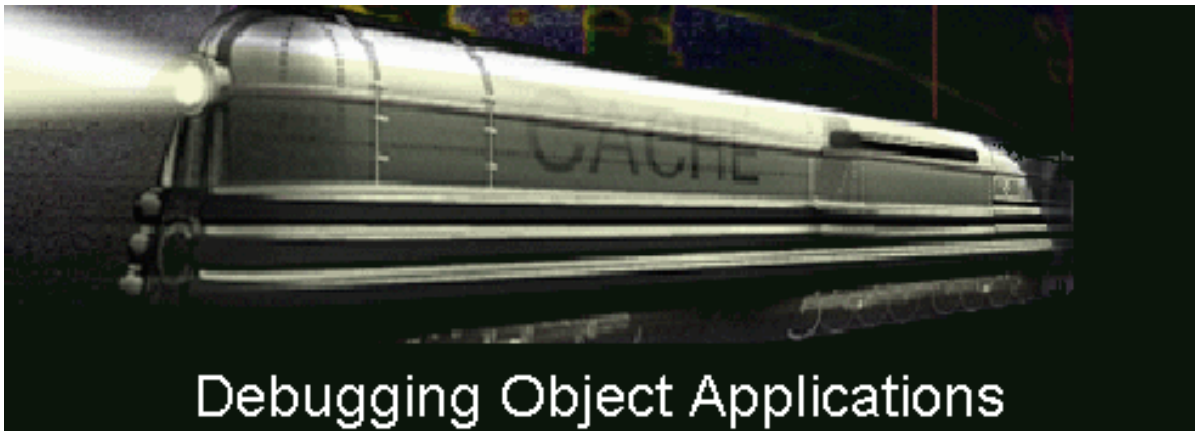
The combination of error number and locale must be unique for each error, but the same error code can be used for multiple locales.

Display Error Messages

You can use the Caché Object Utility Library to display the error message associated with the status code of a method which did not successfully execute. If *sc* is the status code returned by a method which did not execute successfully, you can display the error text using the following call:

```
DO $$DisplayError^%apiOBJ(sc)
```

Revision Date:
May 13, 1999



The Caché Object Utility Library provides three calls for debugging object applications. These calls provide the following functionality:

- Display an object's details
- List all objects in current process
- Kill all objects in current process

Display an Object's Details

The Caché Object Utility Library allows you to display details about any object in memory using the following call:

```
DO DumpObject^%apiOBJ(oref)
```

This call displays the OREF, class name, reference count, OID, concurrency, and the value of each property of the object.

List All Objects in the Current Process

The Caché Object Utility Library allows you to display information about all objects in the current process using the following call:

```
DO ListAllObjects^%apiOBJ()
```

This call displays the OREF and class name of every object in the current process.

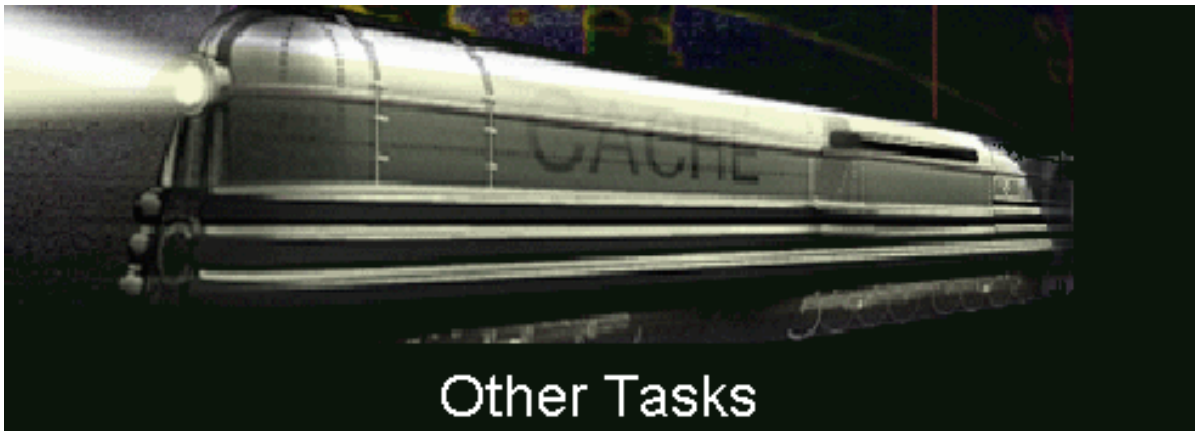
Kill All Objects in the Current Process

The Caché Object Utility Library allows you to kill all objects in the current process using the following call:

```
DO KillAllObjects^%apiOBJ()
```

This call displays removes all objects in the current process from memory. %Close() and %OnClose() methods are not called for these objects..

Revision Date:
May 13, 1999



Revision Date:
May 13, 1999

You can use the Caché Object Utility Library to perform these additional tasks:

- Access a method indirectly
- Get the value of a property
- Set the value of a property
- Get the version number
- Manipulate concurrency settings
- Manipulate transaction settings

Accessing a Method Indirectly

The Caché Object Utility Library allows you to run a method of a particular object indirectly. You can use the following commands to respectively run methods which return a value and methods which do not return a value:

```
Set x=$$InvokeMethod^%apiOBJ(class,oref,method,parm1,parm2,...,parm9)
Do InvokeMethod^%apiOBJ(class,oref,method,parm1,parm2,...,parm9)
```

Runs the method *method* of the instance *oref* of class *class* using the parameters passed in *parmn*

Getting the Value of a Property

You can use the Caché Object Utility Library to get the value of a property in any object in memory using the following call:

```
SET name=$$GetProperty^%apiOBJ(oref, propname)
```

where *oref* is the OREF of the object and *propname* is the name of the property to return.

Setting the Value of a Property

You can use the Caché Object Utility Library to set the value of a property in any object in memory using the following call:

```
DO SetProperty^%apiOBJ(oref, propname,value)
```

where *oref* is the OREF of the object, *propname* is the name of the property to change, and *value* is the new value to set..

Getting the Version Number

You can use the Caché Object Utility Library to get the version number of your Caché Objects installation. You can use the following command to get the version number:

```
DO $$GetVersion^%apiOBJ()
```

This prints the version number of the *Caché Objects* system that is currently installed.

Alternately, you can use the following command to just return the build number:

```
SET build=$$GetBuildNumber^%apiOBJ()
```

Manipulating Concurrency Settings

You can use the Caché Object Utility Library to get the current concurrency mode or to set the concurrency mode.

Getting the Current Concurrency Mode

You can use the following command to get the current concurrency mode:

```
SET mode=$$GetConcurrencyMode^%apiOBJ()
```

This returns the current setting of the default concurrency mode.

Setting the Concurrency Mode

You can use the following command to set the concurrency mode:

```
SETsc=$$SetConcurrencyMode^%apiOBJ(mode)
```

This function sets the default concurrency mode for the current process to *mode*. It returns the previous default concurrency mode. Valid values for *mode* include 0,1,2,3, and 4. If *mode* has an invalid value the function returns ``InvalidConcurrencyMode". For example:

```
Set oldsetting=$$SetConcurrencyMode^%apiOBJ(1)
```

Manipulating Transaction Settings

You can use the Caché Object Utility Library to get the current transaction mode or to set the transaction mode.

Getting the Transaction Mode

You can use the following command to get the transaction mode:

```
SET mode=$$GetTransactionMode^%apiOBJ()
```

This returns the current setting of the default transaction mode.

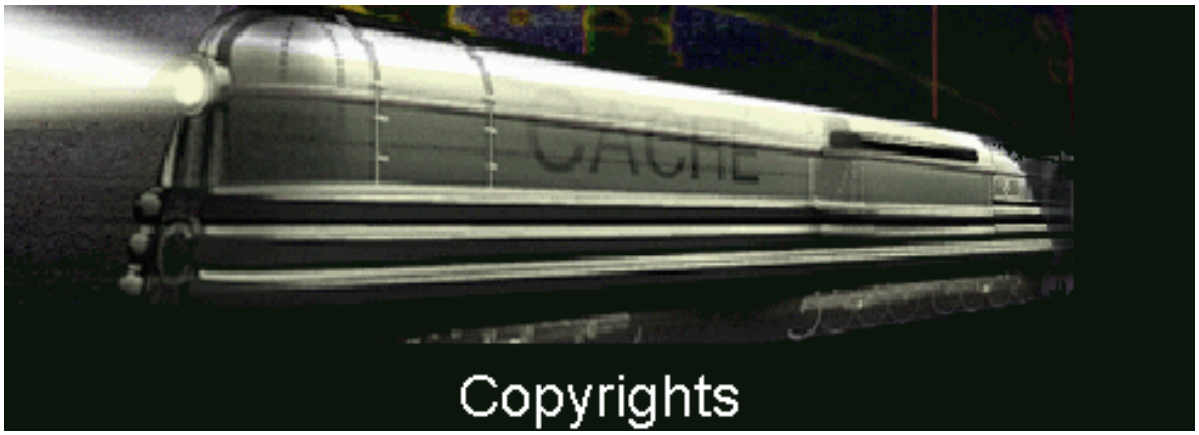
Setting the Transaction Mode

You can use the following command to set the transaction mode:

```
SET sc=$$SetTransactionMode^%apiOBJ(mode)
```

This function sets the default transaction mode for the current process to *mode*. It returns the previous default transaction mode. Valid values for *mode* include 0 and 1. If *mode* has an invalid value the function returns ``InvalidTransactionMode". For example:

```
Set oldsetting=$$SetTransactionMode^%apiOBJ(1)
```



Caché
Object Utility Library Reference

For support questions about any InterSystems products, contact the InterSystems Worldwide Response Center:

Phone: US: +1 617 621-0700 Europe: +44 (0) 1753 830-077
Fax: US: +1 617 374-9391 Europe: +44 (0) 1753 861-311

Internet — support@intersys.com **FTP Site** — ftp.intersys.com **World Wide Web** — www.intersys.com

Revision Date:
May 13, 1999