CHAPTER

9

Additional Routine Utilities

This chapter describes additional routine utilities that enable you to examine and manipulate routines from terminal devices. All of the functionality described in this chapter is also available through the Caché Explorer.

These utilities are provided for use in batch jobs or an alternative to use when you do not have a client machine handy.

Selecting Routines

When you start any of the routine utilities described in this section, it will prompt you for the routine name. The simplest way to specify routines is to enter the name of each routine.

For example, the following specification selects the routine LAB:

```
Routine: LAB
```

You can also use the following symbols:

Table 9-1: Options for Selecting Routines

Symbol	Definition
? (question mark)	A single question mark at the Routine(s) prompt displays help text on the wildcard symbols below.
?L	A question mark followed by the character L displays a list of the routines you have selected so far.
; (semicolon)	Separates the routines in a list of routines.
- (dash)	Indicates an inclusive range of globals to be selected.
' (apostrophe)	Indicates globals to be excluded from those to be selected.
= =.INT = :.	In utilities that use the two-column format (e.g., %RCMP, where routines in the first column are compared to those in the second), you can use the equals sign (=) to signify the same routine name as specified in the “From” column. If you specify no namespace, Caché assumes the current namespace. You can use the equals sign followed by an extension (=.INT) to signify the same name as before with the specified extension. If you specify no extension, Caché assumes “.MAC”. You can use the symbol =:. to signify the same routine and extension as in the "From" column.
* (asterisk)	A wildcard character that matches any character(s).
? (question mark)	A wildcard character that matches any single character. Note that the question mark cannot be the first character of a routine specification other than in its use to obtain help or a listing, as described above.

Selecting Routines with Wildcards

Wildcards take some of guesswork out of selecting the right routine as well as allowing you to select multiple routines at a time without making a routine set.

Place an asterisk after a set of alphanumeric characters to specify all routines whose names start with those characters. For example, the following selects all routines whose names start with N:

```
Routine: N*
```

Use an asterisk by itself to select all routines:

```
Routine: *
```

Wildcards for Extensions

Use the asterisk as a file extension to specify all extensions. For example:

```
Routine(s): ROU.*
```

means:

```
ROU.MAC, ROU.INT, ROU.INC, and ROU.OBJ
```

which expands more fully to:

```
ROU.MAC.1, ROU.INT, ROU.INC.1, and ROU.OBJ
```

Wildcards for Version Numbers

You must specify a version number only when you're not using the current version. Use the asterisk for version number to specify all versions.

Enter the indicated specification to select the corresponding set of routines.

Specification	Routines Selected
ROU.MAC.*	ROU.MAC.1, ROU.MAC.2, etc.
ROU.INC.*	ROU.INC.1, ROU.INC.2, etc.
ROU.*.*	All versions of all extensions of routine ROU
..*	All versions of all extensions of all routines

Selecting a Range of Routines

Use a dash to select a range of routines.

This example selects all routines whose names range between XR and YB inclusively:

```
Routine: XR-YB
```

When you use a range, Caché understands an implied wildcard. The above command will include XR, XRS, YArd, but will not include xr or yard because lower case is sorted after uppercase.

This example selects all routines whose names range between XR and YB and xr and yb.

```
Routine: XR-YB
Routine: xr-yb
```

Excluding Routines

Use an apostrophe to indicate that one or more routines are not to be included among the routines already selected. This example excludes the routine NIM:

```
Routine: 'NIM
```

You can combine an apostrophe with a dash or an asterisk to specify a set of routines that should be excluded. This example excludes all routines whose names begin with AB and whose names range from XR to YB inclusively:

```
Routine: 'AB*
Routine: 'XR-YB
```

The following specification selects all routines except the routine ABC:

```
Routine: *
Routine: 'ABC
```

Routine Sets

The Caché %RD Routine Set facility lets you create a list of routines under one name, and use that named routine set at the Routine(s): prompt of the routine utilities. This facility is useful when you have a group of routines on which you commonly perform a particular function, such as compiling all the routines in a particular application.

To create a routine set at the Routine(s) prompt of any of the routine utilities:

1. At the Routine(s): prompt, name all the routines you want to include in the routine set, either by naming them explicitly or using the wildcard syntax.
2. Enter ".F" at the Routine(s): prompt.
3. At the File as Routine Set: prompt, enter a name for the routine set. Routine set names are case sensitive, so “ROUTINESET” is not the same as “routineset”.
4. At the With Description: prompt, enter a description for the set.
5. At the OK to File? prompt, press Return.

To use the routine set at the Routine(s): prompt, enter the routine set name preceded by the "@" character.

Assume, for example, you want to create the routine set TESTSET:

```
USER> DO ^%RD
Routine(s):  test*.mac
Routine(s):  .F
  File as Routine Set: TESTSET
   with description: Routines for testing
OK to File? Yes= <RETURN>
Filing ... done
```

To then use the routine set TESTSET:

```
USER> DO^%RD

Routine(s): @TESTSET

            --.MAC--

test1 test2 test3 test4 test5
```

Referencing Routines in Other Namespaces

The Caché Studio and the CHUI routine utilities allow you to address routines in namespaces other than the current namespace.

To specify routines in another namespace at the CHUI Routine(s): prompt, use the syntax:

```
Routine(s): ["DIR","SYS"]ROU.MAC
Routine(s): ["NAM"]ABC.INT
```

The first refers to the .MAC version of routine “ROU” in directory “DIR” in system “SYS”. Caché interprets the directory and system as an implied

namespace. The latter refers to .INT version of routine “ABC” in the defined namespace “NAM”. This namespace can be local or remote.

To avoid repeatedly retyping extended references where a list of routines is required, the following syntax can be used to refer to the last explicitly specified namespace or implied namespace:

```
[^]
```

Note: You can also map routines to different locations on the network, much as you can map global subscripts. For more information, see the *Caché Networking Guide*.

Restrictions

Some restrictions exist on extended reference syntax. The CHUI routine utilities do not permit you to alter the contents of a namespace other than the current namespace. The utilities %RCHANGE, %RCOMPILE, and %RDEL do not accept the extended reference syntax. The %RCOPY utility has a limitation: you can copy routines from another namespace into the current namespace, but you cannot copy routines from the current namespace to another namespace. To copy routines from the current namespace to another namespace, use the Caché Studio.

Examples

Routines “AAA”, “BBB”, and “CCC” in namespace "NAM$SYSX" are specified as follows:

```
Routine(s): ["NAM$SYSX"]AAA
Routine(s): [^]BBB
Routine(s): [^]CCC
```

If the two-column format is used, (e.g., in the %RCMP utility, where routines from the first column are compared to those in the second), the [^] syntax is column specific.

In the following example, routine “AAA” in “DIR1” of system “SYS” is compared to routine “BBB” in “DIR2” of the current computer, then routine “XXX” in “DIR1” of “SYS” is compared to “YYY” in “DIR2” of the current computer:

```
Compare Routine(s): ["DIR1","SYS"]AAA  To: ["DIR2"]BBB
Compare Routine(s): [^]XXX  To: [^]YYY
```

Importing and Exporting Routines

Caché provides four utilities for importing and exporting routines for use on other systems and remote storage.

- The %RI utility imports routines and intermediate code (.MAC, .INC and .INT) into Caché.
- The %RO utility exports routines and intermediate code (.MAC, .INC and .INT) to disk or tape.
- The %RIMF utility imports compiled routine object code (.OBJ) into Caché.
- The %ROMF utility exports compiled routine object code (.OBJ) to disk or tape.

Importing Routines with %RI

Use %RI to import source and intermediate code for routines written to a sequential file on disk or magnetic tape. %RI imports source code exported using %RO.

To run %RI:

1. At the Caché prompt, type:

   USER> ***DO ^%RI***

2. At the “Input Routines from Sequential Device" prompt, enter the name of the device to which you wrote the routine. The utility describes the contents of the file or tape and asks if routines which exist should be overwritten.

3. Enter (the first letter of) the function you want:

Function	Effect
All	Load all routines
Select	Select routines one at a time for loading.
Enter	Specify a set of routines to load.
List	Display a list of the routines available for loading on the device you selected.
Quit	Finish selecting routines.

4. Continue to answer the questions the utility asks. If you answered Yes at the Recompile prompt and you answer Yes at the “Display Syntax Errors? Yes =>” prompt, you see the following:

- An explanation of codes that will follow each routine as it is copied into the namespace:
  - “^” indicates routines which will replace those now on file
  - “@” indicates routines which have been [re]compiled
  - “-” indicates routines which have not been filed
- How many routines were copied into the namespace

Example

This example shows a session where an archived copy of an intermediate code version of the routine ACCT is imported into the current namespace with %RI.

```
USER>DO ^%RI

Input routines from Sequential Device: acct.rtn
Parameters: "R"=> RETURN>

File written [Caché 3.2] by %RO
on 03 Apr 99 3:43 PM with extension INT and
with description: ACCT.INT

( All Select Enter List Quit )

Routine Input Option: ALL Routines

If a selected routine has the same name as one already on file,
shall it replace the one on file? No => <RETURN>
Recompile? Yes => <RETURN>
Display Syntax Errors? Yes => <RETURN>

^ indicates routines which will replace those now on file.
@ indicates routines which have been [re]compiled.
- indicates routines which have not been filed.

ACCT.INT^@

1 routines saved.
USER>
```

Exporting Routines with %RO

Use %RO to export source code and intermediate code for one or more routines to a sequential file on disk or magnetic tape. Use %RI to read routines saved by %RO.

The following procedure deal with running %RO. %RO allows you to specify the routine output format:

- To output .MAC files, enter the routine name only, with no extension. The routine will copy the .MAC file if it exists, or the .INT file if there is no .MAC file.
- To output both .MAC and .INT files, enter the routine name followed by ".*", so all files with that name and any extension will be copied.
- Enter *routine_name*.INT to copy just the .INT file.
- Enter *routine_name*.INC to copy just the .INC file. Specifying .INC or using the * wildcard is the only way to select an INC file for export.

To run %RO:

1. At the Caché prompt, type:
   USER>***DO ^%RO***
2. Select output data.
3. The utility continues to prompt you for routine names until you respond by pressing <RETURN>.
4. Enter a description or press <RETURN>.
5. Enter the name of the file or the logical number of the magnetic tape device to which you are writing. You may include a file extension, like .rou.
6. At the "Parameters" prompt you can specify parameters, or press <RETURN> to accept the default that appears.
7. At the "Printer Format" prompt, enter *Y* to store output in printer format, or press <RETURN> for default output format. Printer format requires more space because it produces nicely formatted, tabbed output, similar to what you see when you issue the Print command at the Caché prompt.
8. If saving to magnetic tape, you are prompted to save at current location or rewind tape. Enter *Y* to rewind the tape before you use it, or press <RETURN> to leave the tape in its current position.
9. %RO shows you a list of the routines as it exports them.

This example shows a routine export of all files whose names begin with LOCK to a file on disk with default parameters.

```
USER>DO ^%RO

Routine output (please use %ROMF for object code output)
Routine(s): LOCK*.*
Routine(s):<RETURN>

Description: Lock routines

Output routines to
Device: d:\cachesys\more.rou   Parameters: "WNS"=><RETURN>
Printer Format? No => No
LOCKTAB.MAC LOCKTAB1.MAC LOCKCLR.INT LOCKUX.INT
USER>
```

Importing Compiled Routine Code with %RIMF

Use %RIMF to load Caché ObjectScript object code routines from files produced with the %ROMF utility.

%RIMF prompts you for the following information:

- The device name and file format.
- Which function you want to perform. Enter (the first letter of) the function you want

Function	Effect
All	Load all routines
Select	Select routines one at a time for loading.
Enter	Specify a set of routines to load.
List	Display a list of the routines available for loading on the device you selected.
Quit	Finish selecting routines.

- The name of the routine(s) you want to load (if you requested the Select function).
- Whether you want to overwrite a file with the same name.

Warning: Using %RIMF to restore files you created with %ROMF deletes the source code for those Caché ObjectScript routines, if you have the source code.

Example

This example shows how you would use %RIMF.

```
USER> DO ^%RIMF

Load routines from a %ROMF file

WARNING: This routine will delete the source code (if any)
for existing M routines that are being replaced

Device: c:\cachesys\user test.mro
           file format: ("UR") => <RETURN>

Caché wrote this file on Apr 16 99 3:40 PM.

( All Select Enter List Quit )

Routine Input Option: ALL Routines

If a selected routine has the same name as one already on file, shall it replace
the one on file? No => Yes

. indicates routines which will replace those now on file.

Loading routines ...
zza@.   zzb@.       zzc@.        zzd@.           zzf@.          zzg@.

   8 routines in 0 minutes, 1 second
none of them skipped.
```

Exporting Compiled Routine Code with %ROMF

Use the %ROMF utility to copy Caché ObjectScript object code routines to sequential files.

Routine source code, which normally resides in the ^ROUTINE global, is not always available. When you do not have source code, using %ROMF is the only way you can copy an Caché routine to a sequential file. %RO does not work in this situation.

To restore a file that was created with %ROMF, you must use the %RIMF utility or the Import feature of the Caché Explorer.

%ROMF prompts you for the following information:

- Whether you want all routines; answer *Y* to copy **all** object code routines, or *N* to select individual routines to copy.
- The routines you want to output, if you selected N.
- The device name, file format, and, for disk files, the maximum media size of the device.
- A description of the information you are copying.

The example copies to a sequential file on disk all of the object code routines that start with zz, except zz, in the default file format. Pressing <RETURN> only at the "Maximum media size" prompt means that no maximum is in effect.

```
USER> DO ^%ROMF
%ROMF Fast Object Code Routine Output
All Routines? No => No
Routine: zz*
Routine: 'zz
Routine: <RETURN>
8 routines

Device: c:\cachesys\user test.mro
file format: ("UNW") => <RETURN>
        Maximum media size (bytes): <RETURN>
Description: test of %ROMF

zza zzb zzc zzd zze zzf zzgzzhelp

   8 routines written in   0 minutes,    5 seconds.
```

Copying Routines

The %RCOPY routine is used to:

- Copy macro, intermediate code and include files from another namespace to the current namespace.
- Copy macro, intermediate code and include files from within the current namespace to, giving the new routines different names.
- Compile macro and intermediate code in the current namespace.
- Generate backup versions of macro and include files.

To use %RCOPY:

1. Go to the namespace where you want the routines to reside after being copied, compiled, or converted.
2. Provide the name of the routine or include file you want to copy, compile, or convert. Include the name of the namespace where the routine is located, if it is not in the current namespace.
3. Provide the name of the routine to which you want to copy the routine or routines.
4. Caché then provides you a series of options in the form of questions as shown in the example below.

This example shows all routines beginning with a being copied from the USER namespace into the current namespace.

```
USER> DO ^%RCOPY

Copy routine(s): [USR]A*.MAC To: A*.MAC
Copy routine(s): <RETURN>

Generate backups? No=> <RETURN>
Compile? Yes=> N
Overwrite Existing Routines? Yes=> <RETURN>

Display routine names on
Device: <RETURN> Right margin: 80=> <RETURN>

    AAA.MAC -> AAA.MAC
    ABC.MAC -> ABC.MAC
USER>
```

Deleting Routines

There are three utilities you can use to delete routines from your Caché system.

- %RDELETE - allows you to delete macro, intermediate code, include and compiled object files.
- %RKILL - allows you to delete intermediate code files.
- %RPURGE - allows you to delete backups of macro and include files. This utility is described in “Managing Routine Backups” on page 9-27.

Deleting Routine Files with %RDELETE

The %RDELETE utility deletes macro source routines, include files, intermediate code routines, or object code routines from the current namespace. As routines are deleted, their names appear on the specified output device.

You cannot use %RDELETE to delete routines that reside outside your current namespace.

This example shows the deletion of all versions of all routines (.MAC, .INT, .INC, .OBJ) beginning with the letter Y.

```
USER> DO ^%RDELETE
Delete routines/INCLUDE files.
WARNING: When <rtn>.MAC.0 is deleted, the latest backup is moved to <rtn>.MAC.0,
UNCOMPILED.

Routine(s): Y*.*.*
Routine(s): <RETURN>

Output on
Device: <RETURN> Right margin: 80=> <RETURN>

Delete Selected Routines/Include Files
Files
Apr 13 99 3:32 PM

Namespace: USER

YAZ.MAC.1 YPP.MAC.1 YZZ.MAC.1 YAZ.MAC.2 YPP.MAC.2
YZZ.MAC.2 Y1.INC Y2.INC Y3.INC YAZ.INT
YPP.INT YZZ.INT YAZ.OBJ YPP.OBJ YZZ.OBJ
```

Deleting Intermediate Code with KILL

The intermediate code can also be deleted by using the KILL command on the ^ROUTINE global. Intermediate code is stored in the global ^ROUTINE in the

namespace in which the routine resides. The first subscript of ^ROUTINE is the routine name. To delete the routine MYROUT, type the command:

```
%SYS>KILL ^ROUTINE("MYROUT")
```

Some programs use the $TEXT function to access their source code. If you delete the source code, the $TEXT function returns the empty string ("""). However, you can place text into the m-code so that $TEXT will function even when the source has been deleted with the double semicolon technique.

Listing Routines

Listing Routines in a Namespace with %RD

The %RD utility lists the routines in any namespace. Additionally, you can specify:

- any set of .MAC, .INT, .INC, or .OBJ routines to display using the wildcard syntax.
- short display form or long display form. The short form displays the routine names, extensions, and version numbers. The long form displays, in addition to the short form information, the date and time when routines were last saved, the size in bytes of each routine, and the block(s) that object code routines occupy.
- a range of dates during which the routines were last modified.

To list a routine from a namespace other than the current namespace, type the namespace name inside brackets and within quotation marks ["""], as shown in the following example.

The example shows the short form display of all versions of macro source and intermediate code routines that reside in the BUSINESS namespace (not the

current namespace), begin with the letter Y, and were modified between April 1, 1999 and April 12, 1999.

```
USER> DO ^%RD

Routine(s): [BUSINESS]Y*.MAC
Routine(s): Y*.INT
Routine(s): <RETURN>

Long or Short form (L or S)? S=> <RETURN>
Find routines last modified since date: 4/1/99
    and on or before date: t
Display on
Device: <RETURN> Right margin: 80=> <RETURN>

    Short Listing of Selected Routines/Include Files
    Namespace: USER
    Apr 12 99 11:30 AM  Page 1

--.MAC--

YAAA.MAC.1   YAAA.MAC.2   YAAA.MAC.3   YBBB.MAC   YDDD.MAC

--.INT--
YAAA.INT   YCCC.INT
```

View the First Line of Selected Routines with %RFIRST

The %RFIRST utility lists the first line of a the selected macro, include or intermediate code files. %RFIRST does not operate on object code routines. This utility displays each routine's name, followed by an indentation and the first line in the routine. You can also specify a range of dates between which changes were made.

The example below shows the display of the first line of all intermediate code routines beginning with the letter H in the current namespace.

```
USER> DO ^%RFIRST

Print first lines of selected routines or include files.
Routine(s): H*.int
Routine(s): <RETURN>

Find routines last modified since date: <RETURN>
       and on or before date: <RETURN>

Output on
Device: <RETURN> Right margin: 80=> <RETURN>

           First Line of Selected Routines Files

HEDIT.INTHEDIT ; edit stats; 06 Apr 99 12:22 PM

HPRINT.INTHPRING(POS) ; print stats; 14 Apr 99 11:21 AM

HREAD.INTHREAD ; data entry program ; 05 Jan 99 7:16 PM
```

Comparing Routines

Caché provides two character-based utilities to compare routines:

- %RCMP - compares similar routines installed on your Caché system.
- %RCMPSEQ - compares an in-memory routine with the identical routine stored on disk.

Comparing Routines with %RCMP

The %RCMP utility compares pairs of routines, identifies how they differ, then prints those lines that are different. It is useful for comparing routines in which only a few lines differ.

The following example illustrates two difference routine comparisons.

- The first example shows a comparison of a MAC routine in a difference namespace with a MAC routine in the current namespace.
- The second example shows a comparison of two INT routines in the current namespace.The third example compares the intermediate code routine ABC2.INT with the routine ABC3.INT in the same namespace. The lines that differ are output.

```
USER> DO ^%RCMP
Compare: ["ACCOUNT"]FIELDS.MAC with: FIELDS.MAC
Compare: ABC2.INT with: ABC3.INTCompare:

Ignore Comment Differences? No => No

Display Results on
Device:      Right margin: 80=>

Routine Comparison May 14 99 2:50 PM
From namespace: %SYS

["ACCOUNT"]FIELDS.MAC     FIELDS.MAC
*********************************************************************
         ["ACCOUNT"]FIELDS.MAC  - Date last edited: 11 Apr 99 10:20AM
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
         FIELDS.MAC - Date last edited: 07 Apr 99 01:33PM
["ACCOUNT"]FIELDS.MAC
+12   #include A
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
FIELDS.MAC
+12   #include X

["ACCOUNT"]FIELDS.MAC
+22   #include B
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
FIELDS.MAC
+22   #include Y
*********************************************************************
ABC2.INT               ABC3.INT
ABC2.INT  -  Date last edited: 15 Jan 99 10:00AM
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
         ABC3.INT  -  Date last edited: 15 Jan 99 11:15AM
ABC2.INT
+2    s x=6,y=7,z=8
 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
ABC3.INT
+2    s x=7,y=7,z=9
*********************************************************************
USER>
```

Comparing Routines on Disk with %RCMPSEQ

The %RCMPSEQ utility compares macro source, intermediate code routines, and include files saved by the utility %RO with the routines of the same name installed on your Caché system. This allows you to easily check for the difference between a current version and a saved backup copy.

The utility displays any lines that have changed. If a routine has too many discrepancies, %RCMPSEQ stops displaying discrepancies and goes to the next routine.

This example shows a comparison between routines stored to the sequential file "PROUTS.OLD" and the same routines stored on disk.

```
USER> DO ^%RCMPSEQ

(Compare routines in sequential file with routines of same name and extension.)

Use Sequential Device:
PROUTS.OLD     Parameters: "R"=> <RETURN>
Display differences on
Device: <RETURN> Right margin: 80=> <RETURN>

File written by %RO on 13 Apr 99 7:48 PM
with comment All P Intermediate Code Routines
OK to proceed? Y=> <RETURN>

Routine Comparisons from Sequential File to Disk
Apr 15 99 2:00 PM

PAA.INT

**********************************************************************
PAA.INT
Seq. file:Date last edited: 08 Apr 99 10:45AM
    Disk:Date last edited: 15 Apr 99 09:12AM

Line+1
Seq. file:fields d ^field2
Disk:fields ;from macro source

Line+6
Seq. file:i lev=0 s cap="NO"
    Disk: i lev=1 s cap="YES"

Line+22
Seq. file: Q
    Disk:;
**********************************************************************
PAB.INT20         PAC.INT             PRR.INT20         PAQ.INT
```

Finding and Replacing Text in Routines

Caché provides character-based utilities that let you search macro, intermediate and include files for:

- *all* of a specified set of strings
- at least one of a specified set of strings
- a specified string for the purpose of replacing it

Finding All Specified Strings with %RFAND

The %RFAND utility searches through macro source code, intermediate code, and include files and returns routine lines that contain *all* of a specified set of strings. %RFAND does not operate on object code routines.

The example below shows the utility searching for occurrences of the strings "Hello" and "Goodbye" in all macro source and intermediate code routines in the current namespace whose names begin with the letter A. The utility outputs the names of all routines through which it searches. When it finds all of the specified strings in one line of a routine, it outputs the tag numbers and lines in which the strings are located.

```
USER> DO ^%RFAND

Find routine lines that contain all of a set of strings.

1. Search for: Hello
2. And Search for: Goodbye
3. And Search for: <RETURN>

Exact Upper/Lowercase match? Y=> N

Routine(s): A*.MAC
Routine(s): A*.INT
Routine(s): <RETURN>

Display results
Device: <RETURN> Right Margin: 132=> <RETURN>

ABC.INT AQQ.INT AYY.INT AZZ.INT

ABC.MAC AQQ.MAC AYY.INT AZZ.INT

USER>
```

Finding at Least One String of a Set of Strings: Use %RFIND

The %RFIND utility searches through macro source code, intermediate code, and include files and returns all routine lines that contain at least one of a specified set of strings. %RFIND does not operate on object code routines.

The example below shows the utility searching for occurrences of the strings "Hello" and "Goodbye" in all macro source and intermediate code routines in the current namespace whose names begin with the letter A. The utility outputs the names of all routines through which it searches. When it finds one of the specified strings, it outputs the tag numbers and lines in which that string is located.

```
USER> DO ^%RFIND

Find routine lines that contain at least one of a set of strings.

1. Search for: Hello
2. Search for: Goodbye
3. Search for: <RETURN>

Exact Upper/Lowercase Match? Yes=> N

Routine(s): A*.MAC
Routine(s): A*.INT
Routine(s): <RETURN>

Display results on
Device: <RETURN> Right Margin: 132=> <RETURN>

ABC.INT
ABC+4  S ^ABC(x) = "Hello"

AQQ.INT AYY.INT AZZ.INT
AZZ+6  S ^ABC(z) = "Goodbye"

ABC.MAC
ABC+4  S ^ABC(x) = "Hello"

AQQ.MAC AYY.MAC AZZ.MAC
AZZ+6  S ^ABC(z) = "Goodbye"
USER>
```

Finding and Replacing a String with %RCHANGE

The %RCHANGE utility changes all occurrences of a specified string in macro source routines, intermediate code routines, and include files to a new value. The utility can generates backups for each changed routine, recompile each changed routine, and verify each change before it is made. %RCHANGE works only on routines in your current namespace.

The example shows every occurrence of the global ^ABD changed to ^ABC in all routines beginning with the letter Z in the USER namespace.

```
USER> DO ^%RCHANGE
This routine changes all occurrences of a string in routines/include files.
1.  Change every: ^ABD  to: ^ABC
2.  Change every:

Routine(s): Z*
Routine(s): <RETURN>

Generate backups? N=> <RETURN>
Recompile? Y=> <RETURN>
Verify Each Change?  No=> <RETURN>

Display changes on
Device: <RETURN> Right Margin: 80=> <RETURN>

        Routine Change Jul 17 99 2:42 PM

Changing "^ABD" to "^ABC"

ZBB.INT
ZBB+5   set ^ABC(0)=41250

ZBB.MAC
ZBB+5   set ^ABC(0)=41250
USER>
```

Compiling Routines

The following sections describe how to compile routines. Compiling routines generates .INT and .OBJ code. You can only compile routines in your current namespace.

Source Code, Compiled Code, and Comment Lines

Intermediate code (Caché ObjectScript source code) and code tokens are stored separately. Thus you can delete the source code of a routine and still retain a form that can be executed.

You can execute most of the object code without referring to the original source, except for the $TEXT function and the PRINT command. Applications using these two features require special care if their source is to be deleted.

When you store a routine, Caché stores its source form as nodes of a global called ^ROUTINE, for which you can modify the protection codes just as with any other global. This global has the following form:

```
^ROUTINE(routine-name,0,0)=number of lines
                         ,1)=source code of first line
                         ,2)=source code of second line
... etc.
```

When you use the PRINT command or the $TEXT function, Caché finds the relevant source text in the ^ROUTINE global. You can also refer to and modify this global just like any other global. However, it rarely makes sense to change the value of a node with a SET command, since the source form of the routine will no longer correspond to its object form.

If you kill the nodes for a particular routine, the $TEXT function returns null values and the PRINT command will print null lines. Similarly, if the protection codes for the ^ROUTINE global are set so that you cannot read the ^ROUTINE global, executing a PRINT command or a $TEXT function will also result in null strings. Because the source code is stored in the global ^ROUTINE, use of $TEXT requires a global reference (which does not affect the naked indicator).

Suppose you want to delete the source code for a routine from ^ROUTINE or prevent others from reading or changing ^ROUTINE and still use $TEXT in routines. You can do this with a special form of routine line that contains only a comment that begins with two consecutive semicolons:

```
;; Caché's special form of comment line
```

Such lines are always embedded in the m-code, and $TEXT can always read them; $TEXT looks first for these embedded comments in the object code before

searching the ^ROUTINE global. Thus this Caché ObjectScript works as expected even when its source code in the ^ROUTINE global is killed:

```
    W !,$P($T(MSG),";;",2,99),!,$P($T(MSG+1),";;",2,99),! Q
MSG ;;First line of the message
    ;;Second line of the message
```

Note that embedded comment lines may have a label. Any line that contains a Caché ObjectScript command that precedes the double semicolon, however, is treated as a normal comment line. Since the double semicolon technique is faster and works even when the source code is deleted or unreadable, you should use this technique for function driver programs or any program requiring access to text embedded as comments.

If you kill the source code in the ^ROUTINE global, these utilities no longer work. Since you can't PRINT a routine without source code, killing the source code makes it impossible to retrieve a routine's source form. If you want to protect your routines in this way, be sure to save them first with the Caché Explorer, Caché Studio or %RO.

You can edit a routine that has no source code even though you cannot examine the code. You must edit the material only by using ZINSERT to add or ZREMOVE to delete entire lines at a time. The system stores the new lines of code in the ^ROUTINE global, forming an audit trail of the inserted lines.

Compiling Code with %RCOMPIL

The Caché Studio is the preferred utility for compiling routines. However, you can use the utility %RCOMPIL to compile either the macro source level or the intermediate code level of a routine. When you invoke %RCOMPIL on macro source, the following takes place:

1. The preprocessor phase of the compiler creates intermediate code.
2. The main compiler creates object code.

If you invoke %RCOMPIL on the intermediate level, the main compiler engages to produce object code.

Some intermediate level routines cannot be compiled. When this happens, a compiler abort message occurs and no modification takes place. The following two conditions can cause this to happen:

1. Source lines are missing from the intermediate code level of ^ROUTINE.
2. The intermediate code contains SQL code.

In both cases, compilation must be performed at the macro source level.

Syntax errors in Caché ObjectScript and most SQL code do not abort compilation.

Compilation can also occur while executing the utilities %RCHANGE and %RCOPY or the Caché Studio. These utilities prompt you to specify whether or not you want to compile.

Note that %RCOMPIL has an NALL entry point that lets you recompile all routines in all defined namespaces; for example:

```
USER>DO NALL^%RCOMPIL
```

This example shows the macro source routine “EMPLIST” being compiled.

```
USER>DO ^%RCOMPIL

Routine(s): EMPLIST.MAC
Routine(s): <RETURN>

Check for Syntax Errors? Yes => YES
Display on
Device: <RETURN> Right margin: 80=> <RETURN>

EMPLIST.MAC
USER>
```

Routine Copying and Compiling Synchronization

Caché does not force you to keep .MAC, .INT, and .OBJ levels in synch for a given routine. Copying or editing a routine at either the .MAC or .INT level does not automatically result in compiling that routine, although the utilities for copying and editing do give you the option of compiling. You choose when a routine should be compiled; the system does not do it automatically and does not attempt to keep compiled routines in synch with source code.

Sometimes you may want to keep the different versions of a routine out of synch. For example, you may want to edit one or more macro source routines for several days and not disturb the .INT and .OBJ levels until all editing is complete.

Although copying of .MAC file extensions to .INT and .INT file extensions to .MAC is permitted, .INT files do not always contain all of the information necessary to produce corresponding .MAC files; source lines may be missing or embedded SQL code may exist (for which there is no source code). If a .MAC routine includes preprocessor statements (such as #if or macros) or embedded SQL, do not copy it to the .INT level because it cannot be compiled there.

Managing Routine Backups

Caché provides two utilities to help you manage backups.

- %RVERMAX - allows you to specify how many backup copies to retain.
- %RPURGE - allows you to delete all backup copies.

Specifying the Maximum Number of Backups with %RVERMAX

The %RVERMAX utility specifies (on a per-namespace basis) the maximum number of backup versions of macro source routines and include files that the namespace will maintain. The default is 2 (one current version and one backup version). The maximum number of versions that can be specified is 9 (one current version and 8 backup versions).

The example below shows the maximum number of versions being reset from the default value of 2 to a new value of 5.

```
USER> DO ^%RVERMAX

Number of versions to keep for .MAC: 2=> 5
Number of version to keep for .INC: 2=> 5
USER>
```

Deleting Backups with %RPURGE

You can use the %RPURGE utility to delete some or all backups for.routine source and include files (.MAC and .INC). The utility asks you to specify how many versions should be maintained, and purges all others. The default number of versions to be maintained is 1. If you wish to keep more than the current version, you must change the default.

The example below shows all backup versions of the macro source routines "TEST" and "MATT" in the current namespace being purged. As the backups are purged, the routines and version numbers are printed to the specified output device

```
USER> DO ^%RPURGE

Purge backups, keeping how many versions: 1=> <RETURN>

Routine(s): TEST.MAC
Routine(s): MATT*.MAC

Device: <RETURN> Right margin: 79 => <RETURN>

    Purge Selected Routines/Include Files
    Retaining 1 Version
    Apr 15 99 12:22 PM

TEST.MAC.2 TEST.MAC.3 MATT1.INC.2 MATT1.INC.3 MATT1.INC.4

MATT2.INC.2 MATT2.INC.3 MATT3.INC.3

8 routines Purged.
USER>
```