
Physical Storage of Globals and Collation

This appendix describes how Caché stores globals and collates data.

Physical Storage of Globals page C-2

Collation page C-12

Routine Storage page C-14

Physical Storage of Globals

This section describes the physical storage of globals in Caché. While this information is currently accurate, the internal structure may change in future releases. Take when developing applications that rely on the physical structure of the database, as these may need to be updated in the future. %ST node should be used where possible to avoid hard-coded values which must be maintained “by hand.”

Globals Stored in CACHE.DAT or CACHE.EXT Files

Globals are stored in a Caché database file called CACHE.DAT. In a multi-volume database, the first volume is called CACHE.DAT; additional volumes are called CACHE.EXT. Under Caché, each host operating system directory can have an CACHE.DAT or CACHE.EXT file, so there can be more than one Caché database file available on a system. No two globals can have the same name unless they are in different Caché databases (i.e., an CACHE.DAT file and any CACHE.EXT files associated with it).

Globals

Caché databases contain Caché *globals*. Each database include a global directory, which describes where in the database the globals are stored and some attributes describing the individual globals.

Routines

Caché ObjectScript programs are called *routines*. Routines can exist in three forms, all stored in globals:

Routine Form	How and Where Stored
macro code	As nodes in the ^mMAC global,
intermediate code	In the ^ROUTINE global.
object code	In a special global whose name is the single character with the ASCII value of 255. The object code global is not normally included in a display of the globals in a database.

Creating a Caché Database

Before you can assign globals values in an application, you must create a Caché database in the directory in which your process will run.

General Structure of Caché Database Files

Caché stores information in blocks that contain 2,048 bytes each. The blocks are organized in groups of 400, called maps. Maps are numbered starting at zero, within each volume. The final map in a database that has undergone expansion may contain fewer than 400 blocks, if there is no more disk space available. Later, if it expands the database automatically again, Caché first tries to add blocks to a final map if that final map contains fewer than the standard 400 blocks.

Note for OpenVMS systems: One Caché block is equivalent to four OpenVMS blocks.

A block number identifies each block uniquely. The *cumulative block number* is unique within the entire database. The first block is numbered 1, the first block in the second map is numbered 401, the first block in the third map 801, and so on. A block can also be described by its position within a map. The block 0:32 is the 32nd block in the first map. The block 2:400 is the 400th, or last, block in the third map.

A database is a collection of one or more extents which contain all of the maps comprising the database. This extent structure allows the Caché database to expand beyond the boundaries of the physical disk or partition.

On OpenVMS and Windows, an extent is a flat file within a directory.

On UNIX, an extent can be either a flat file within a directory, or a UNIX raw disk partition.

A database always consists of one primary extent and from zero to seven secondary extent. A primary extent is the first or only extent in the database. Any other extents are secondary database extents. The extents are permanently arranged in a definite order, with the primary extent always being the first. Database references are made to the primary extent which transparently manipulates data on the secondary extents.

Block Types

The types of blocks you can have in an Caché database are as follows.

Table C-1: Database Block Types

Block Type		Definition
Decimal	Hexadecimal	
1	1	Global Directory Block
2	2	Routine Directory Block or Global or Routine Upper Level Pointer Block
6	6	Bottom Level Pointer Block
8	8	Global Data Block (strings 512 bytes or less)
12	C	Big Global Data Block (strings up to 32,767 bytes)
170	AA	Map Block

When the Garbage Collector (process GARCOL) has collected a block, it adds 32 (decimal) to the block type number. (The Caché Garbage Collector keeps large areas of free space available in your database.

You can control the location of global pointer blocks, and global data blocks when you create a new database.

Map Organization

Each map contains a special first block, called a map block, that contains a map of all blocks in use and all blocks that are still available, or free, within the 400-block section.

The map block of the *first* map, called map 0, is called the label block. It contains information that other map blocks do not contain. The first map block of the first volume in the volume set includes information that is not in the map 0 map block of other volumes.

Note that in the following table, items marked with an asterisk (*) are in the first map block of the first volume of the volume set only. All other items in the table are in the first map block of every volume in the volume set.

Table C-2: Map O Map Block Only

%ST Subscripts		Decimal	Hex		
location	size	Location	Location	Length	Contents
gfoic	szoic	1024*	400	2	UIC of Caché Database
gfvoleseq	szvoleseq	1026	402	2	This Volume Number
gfexp	szexp	1028*	404	2	Extend By (Number of Maps)
		1030	406	2	Total Number of Volumes
gfdir	szdir	1036*	40C	4	Location of Global Directory (Byte count within map)
gfgptr	szgptr	1040*	410	4	Location of New Global Pointers
gfrdir	szrdir	1044*	414	4	Location of Routine Directory
gfrgrw	szrgrw	1048*	418	4	Location for New Routine Growth
gfggrw	szggrw	1052*	41C	4	Location for New Global Growth
gfmsz	szmsz	1060	424	2	Number of Blocks per Map
gfmaps	szmaps	1064	428	4	Size of File (Number of Maps)
gfmax	szmax	1068*	42C	4	Maximum Number of Maps
gfvolmaps	szvolmaps	1072	430	4	Maps in this Volume
none		1524*	5F4	8	Map Label "ISM0 UCI"

Executing `D GFS^%ST` defines a number of entries in the `%ST()` array related to the organization of the label block. Applications which access this information should use `%ST()` to locate data to avoid problems caused when the structure of the label block is modified.

Each map block contains the information listed in the following table.

Table C-3: All Map Blocks

%ST Subscripts	Decimal Location	Hex Location	Length	Contents
offset	0-399	0-18F	1	Map Flags: 0 (Available) 1 (Normal Database Block) 128 (Mapped Out Bad Block) 246-254 (Used by Caché) 255 (Reserved)
MAPLABEL	2038	7F6	2	Map Label FFFF (hex) (65535 decimal)
	2040	7F8	2	Map Label 5555 (hex) (21845 decimal)
	2042	7FA	1	Map Label and Block Type AA (hex) (170 decimal)
BLTYPE	2043	7FB	1	Block Type AA (hex) (170 decimal)
	2044	7FC	2	Map Label 8001 (hex) (32769 decimal)
BLOFF	2046	7FE	2	Summary Count (how many blocks are free)

Summary Count

The last two bytes in the map block, locations 2046 and 2047, contain the summary count. These locations normally contain the number of free blocks in that map. However, if the system attempts to free a block that is already marked free in the map (this can happen if there is a physical disk degrade due to a hardware problem), it notes this degradation of the database by setting the two highest order bits in the last word, so that the summary count is 49,152 or greater. Caché then allocates no more blocks in that map.

If this condition occurs, run an integrity check using either the Integrity functions of the GUI Database System Operations utility, and correct any errors using its Repair function or the manual or auto repair function of DIAG. Then you can correct the map summary count by inserting the proper value in decimal location 2046 of the map block. To obtain the correct value, run the CHECKMAP utility.

Non-Map Blocks

A sample non-map block stores information as described in the following table.

Table C-4: Non-Map Block Contents

%ST0	Decimal Location	Contents	Description
Size = 2036 "BLENGTH"	0	First byte is always 0.	Necessary for integrity of Caché data and pointer blocks.
	1-2035	DATA	Pointer Block Format: Global Ref: ^A(123,"TEST") Pointer: 126545 Data Block Format: Global Ref: ^A(123,"TEST") Data: "HELLO"
"BLGAR"	2036-2039	Garbage Pointer	Used by the Garbage Collector.
"BLLNK"	2040-2042	Right Link Pointer	Points to next block in the Link List.
"BLTYPE"	2043	Block Type	Tells if this block contains Data, Upper Level Pointer, or Bottom Level Pointer.
"BLOFF"	2046-2047	Offset	Location of next available space in block.

Global Storage

Caché maps the logical global into physical locations in a balanced binary tree. This organization consists of the global directory plus two or more levels of blocks with a single block at the top. All of the data lies in blocks at the bottom of the tree. The same number of steps is required to get from the top block to any block in the bottom level.

Each node in the database contains two parts. The first part is a global reference like ^GLO(1,3,4). The second part is the data. Blocks at the bottom of the tree are called *data blocks*. Blocks above this level are called *pointer blocks*. Both data and pointer blocks contain: <global reference> <value>.

- pointer block: <value> = a block number in the database
- data block: <value> = actual value assigned to the global reference.

Within each block, the entries are ordered by subscript reference sorted according to the collation sequence for the global.

Global Directory (Type 1)

Each Caché database has a global directory. For each global, there is an entry in the global directory. If the number of globals and/or the length of global names is large, the global directory may span multiple blocks.

An entry in the global directory node has the format:

^global-name, type, protection, growth-area, block-number

The block to which *block-number* points is a pointer block. The entries in the global directory are not sorted. They are organized in the order the globals are created. The following table describes the contents of a global directory entry.

Table C-5: Global Directory Node

Field	Description
<i>type</i>	A two-byte value. The lower byte contains a 4 if the global is journaled. The upper byte specifies the collation type. Note: Information about whether the global is replicated to other systems is no longer stored in the global directory. As of Version 5.10, it is stored in system tables.
<i>protection</i>	A two-byte value. This value is interpreted as four sets of two binary digits. From left (high-order) to right (low-order), the sets represent the Network, World, Group, and then Owner, as follows: 00 —N 01 —R 10 —RW 11 —RWD
<i>growth-area</i>	Where search for new pointer blocks will begin.
<i>block-number</i>	Location of first pointer.

Global Pointer Blocks

The highest blocks, which normally comprise less than 1% of the block total, are all pointer blocks.

The first pointer block for a global holds the uppermost node of the global's balanced tree. Depending on the size of the global, it may have multiple levels of pointer blocks.

The logical structure of a global pointer node is (*global-reference, block-number*). Pointer nodes are stored in a collating order based upon the global name and subscript.

It is best to allow enough space for the global pointer blocks when you first create the Caché database. For most databases, the default values are sufficient. For very large databases (2 gigabytes or more), you may want to adjust these values. You can use the routine %GCREATE to control where Caché begins to search for new pointer blocks on a per global basis.

Bottom-level Pointer Blocks (Type 6)

Blocks in the level immediately above the data blocks are called bottom-level pointer blocks, or Type 6 blocks. These blocks contain nodes equal to the first global reference in each data block. Each node in a bottom-level pointer block contains the block number of the corresponding data block. Therefore, pointer blocks on this level have one node for each data block, and you can use these nodes to find the data block with information you are looking for.

Upper Level Pointer Blocks (Type 2)

If there is more than one block on the bottom level of pointer blocks, then there is a higher level of pointer blocks. These higher level blocks point to the blocks in the bottom level just as the bottom-level pointer blocks point to the data blocks. If necessary, this process continues with even higher pointer levels until the very top-level consists of a single block, referred to as the top-level pointer block. In a small global with fewer than about 200 data blocks (about 400 Kb), the top-level pointer block may also be the bottom pointer level (that is, only one level of pointer blocks is needed), but in larger globals there may be two or three pointer levels.

Global Data Blocks (Type 8)

The logical structure of a global data node is (*node-reference, data*). Data blocks, like global pointer blocks, store data nodes in collating order based on global names and subscripts. Type 8 blocks hold strings up to 756 bytes long.

Note: If Caché cannot find enough global data blocks, it attempts to create a new map before using free pointer blocks in the current map. This reserves space near the beginning of the database for pointer blocks to minimize fragmentation.

Big Global Data Blocks (Type 12)

Type 12 blocks are identical to type 8 blocks, with the exception that they hold references to strings up to 32,767 bytes long.

Right Link Pointers

Each block (data or pointer) contains a link to the next block on its right at the same level in the tree. The right link pointers on a block level connect that level's blocks in collation sequence. Thus, the tree includes a redundant set of pointers; the downward pointers in a pointer block specify the same structure as the right-link pointers of the next lower level.

The first node of every global is a dummy entry that appears as an unsubscripted global whose name is the space character. This node, inaccessible through the Caché ObjectScript language, is present so that Caché never has to insert a node before the first node.

Figure C-1, “Portion of Global Balanced Tree Block Structure,” on page C-11, illustrates the physical structure of a Caché database. The figure shows a portion of a balanced tree structure with right links. The portion of the global directory block shown provides for three globals: ^A, ^X, and ^C. The global ^X is shown pointing to block 70 which points to subscripts to ^X in blocks 431, 214, and others. Each of these blocks contains pointers to nodes of ^X. For example:

- Block 431 points to:
 - node ^X(“L”) (and others) in block 171
 - node ^X(“O”) (and others) in block 240
 - and so on
- Block 214 points to:
 - node ^X(“U”) (and others) in block 371
 - node ^X(“W”) (and others) in block 97
 - and so on

Each subscript is shown in the shaded Data Block with its data value in the form (“Subscript”) Data value. For example:

- Node ^X(“L”) stores the value 1, shown in the figure as (“L”)1 in the upper left cell of the Data Block
- Node ^X(“Y”) stores the value 25, shown as (“Y”)25 in the upper right cell of the Data Block

The values for the nodes of ^X are shown in Figure C-2, “^X Node Values Stored in Global Structure,” on page C-12.

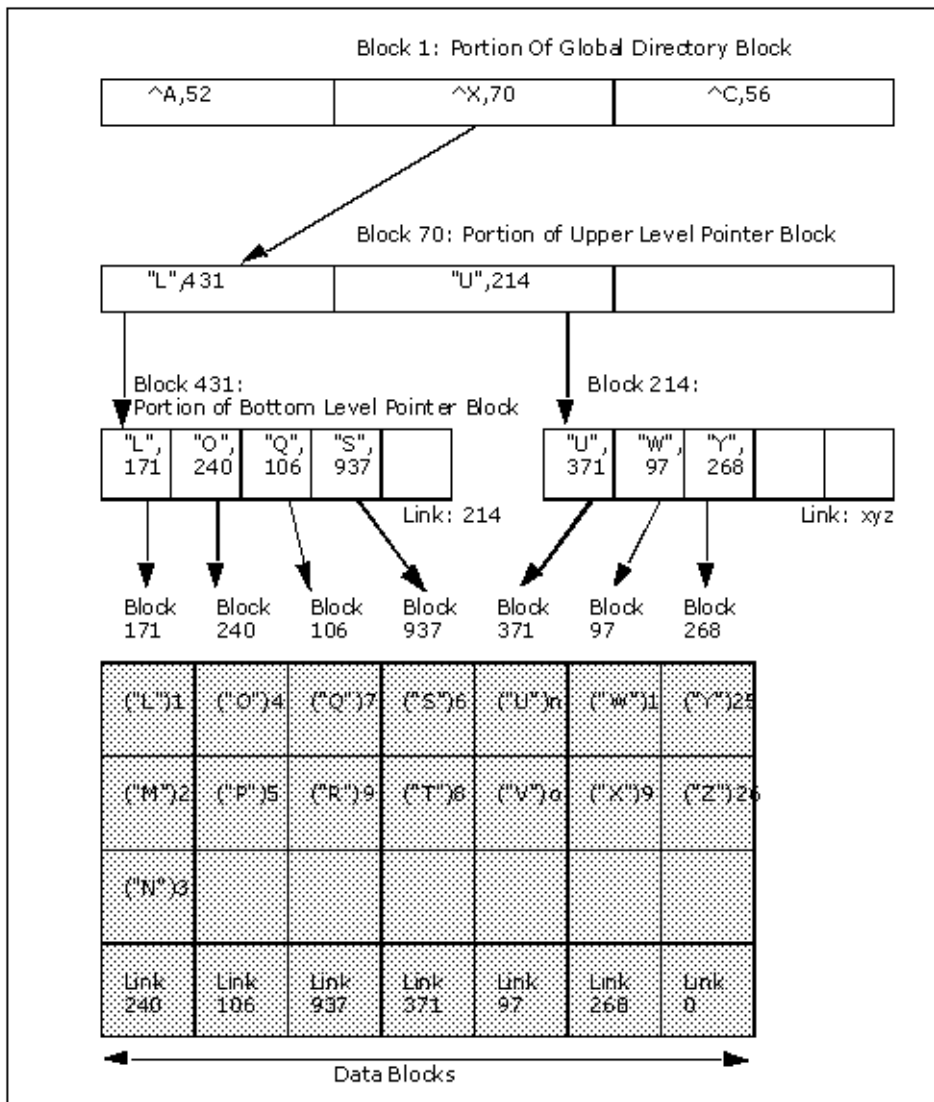


Figure C-1: Portion of Global Balanced Tree Block Structure

$\wedge X("L") = 1$	$\wedge X("O") = 4$	$\wedge X("R") = 9$	$\wedge X("U") = n$	$\wedge X("X") = 9$
$\wedge X("M") = 2$	$\wedge X("P") = 5$	$\wedge X("S") = 6$	$\wedge X("V") = o$	$\wedge X("Y") = 25$
$\wedge X("N") = 3$	$\wedge X("Q") = 7$	$\wedge X("T") = 8$	$\wedge X("W") = 1$	$\wedge X("Z") = 26$

Figure C-2: $\wedge X$ Node Values Stored in Global Structure

Collation

This section describes how data is collated in Caché.

Collation Sequence

A character set and a collation rule together make up a collation sequence. The character set can be the native character set of the computer (such as ASCII) or any other character set the system manager chooses. Caché supports several collation rules, including unicode, ANSI M, and string. National language support (NLS) allows user defined collation in extended character sets.

ANSI Standard M Rule

By this rule, canonic numeric subscripts come first in numeric order, followed by all others in string order. String order is the order of characters in the character set. You can think of canonic numeric subscripts being ordered by the Caché ">" operator, and all others being ordered by the Caché "J" operator. In the following example, -2.40 lies after the other numbers because its trailing zero makes it non-canonic. This example shows the non-canonic subscripts in ASCII order:

-5 -2.4 1 2 19 -2.40 AA BB

String Rule

String rule orders all subscripts as strings in the character set, just as the Caché "J" operator does. This example shows the same subscripts in the ASCII character set, collated using the string rule:

-2.4 -2.40 -5 1 19 2 AA BB

How Collation Sequence is Selected

Caché normally uses the computer's native character set and the Unicode standard collation rule. The system manager can, however, select the collation sequence for individual globals. Because the collation sequence is the storage sequence, you must set it before any data is stored in the global. You select the collation sequence of a new global using the %GCREATE utility.

Collation for National Language Support

Caché's national language support allows for more sophisticated collation. Data is encoded so that it will collate appropriately based on its binary representation. Subscripts may include any character, including control characters, except \$C(0).

A collation rule contains two kinds of information: a rule for collating numeric values and a rule for collating string values. The following two tables show the possible values of the fields that specify this information.

Table C-6: Numeric Collation Types

Value	Meaning
0	New ANSI (ANSI standard collation) New ANSI compacts numeric subscripts more efficiently than Old ANSI.
1	Old ANSI (ANSI collation in earlier releases) Useful if you need to port global data to systems running older versions.
3	DTM Compatibility collation (DTM collation) Useful if you need to port global data from DTM systems.
5	UNICODE (uses 8 or 16-bit characters) Unicode collation uses New ANSI numeric subscript encoding for future support of unicode character sets (default).

Table C-7: String Collation Types

Value	Meaning (see note below)
0	No conversion, quote initial < 32. For example, you might want string rule 0, if all characters in subscripts will be control characters. This is used for local arrays.
1	Quote 0-3 and initial < 32. This is used for global arrays.
2	Disallow 0 or initial < 32. If any character in the subscript is 0 or the first character is a control character, a <SUBSCRIPT> error occurs.
3	Many-to-many 256-byte encoding table, then quote 0-3 and initial < 32, as in rule 1. This rule allows multi-character collation, such as for characters that are stored internally as two separate characters, such as, for example, Spanish "ch" and "ll".

Note: When Caché stores control characters, it inserts a leading character of decimal 31 before the actual character. A “quote” indicates that an extra character has been inserted, but that it does not affect the collation order.

Routine Storage

Caché routines are stored as binary “big strings” as nodes in a special global. The global is named “rOBJ.”