Caché Installation Guide

Version 2018.1
2024-04-03

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

Caché Installation Guide
Caché Version 2018.1 2024-04-03

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)
Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

Table of Contents

List of Tables

About This Book

Caché runs on several different platforms. Please check the online *InterSystems Supported Platforms* document for this release to verify that Caché runs on your particular version of the supported operating systems.

This *Caché Installation Guide* contains the following chapters.

To plan and prepare to install Caché, see the chapter:

- Preparing to Install Caché

Install Caché following the instructions in the appropriate platform-specific installation chapter:

- Installing Caché on Microsoft Windows
- Installing Caché on UNIX®, Linux, and macOS

If you are upgrading from Caché version 5.1 or later, read the following chapter for a list of pre-installation upgrade tasks:

- Upgrading Caché

To learn how to create an installation manifest describing a specific Caché configuration and use it to generate code to configure a Caché instance, read the chapter:

- Creating and Using an Installation Manifest

For detailed information, see the Table of Contents.

For general information, see *Using InterSystems Documentation*.

Note: The InterSystems products Caché and Ensemble share similar underlying technologies. Both products use the instructions in this book for installation. However, there are additional tasks for Ensemble, especially after the initial software installation is complete. Consult the *Ensemble Release Notes* for an overview of these tasks before beginning an Ensemble installation.

1

Preparing to Install Caché

Before you install Caché, read the following sections:

- Cache Installation Planning Considerations
- Managing Caché Memory
- File System and Storage Configuration Recommendations
- Preparing for Caché Security
- Preparing to Install Caché on UNIX®, Linux, and macOS

1.1 Caché Installation Planning Considerations

Read the following Caché installation planning considerations that apply to your installation:

- Install the VS Code - ObjectScript development environment
- Caché Installation Directory
- Setup Type
- Disk Space Requirements
- Supported Platforms and Components
- Caché Private Web Server
- If You Are Upgrading Caché
- Configuring Third-Party Software

1.1.1 Install the VS Code - ObjectScript development environment

Visual Studio Code (VS Code) is a free source code editor made by Microsoft for Windows, Linux and macOS. The InterSystems ObjectScript extension for Visual Studio Code enables you to use VS Code to connect to a Caché server and develop code in ObjectScript. An alternative to VS Code - ObjectScript is Studio, which installs alongside Caché on Windows-based operating systems.

For information on downloading and using VS Code - ObjectScript, see https://intersystems-community.github.io/vscode-objectscript/.

1.1.2 Caché Installation Directory

Throughout the Caché documentation, the directory in which a Caché instance is installed is referred to as *install-dir*. This directory varies by platform, installation type, and user choice, as shown in the following table:

Platform	Installation Type	Directory
Windows	attended	C:\InterSystems\Cache (or Cache*N* when multiple instances exist) unless installing user specifies otherwise.
	unattended	C:\InterSystems\Cache (or Cache*N* when multiple instances exist) unless `INSTALLDIR` property specifies otherwise.
UNIX®, Linux	attended	Installing user must specify. Do not choose the /home directory, or any of its subdirectories.
	unattended	`ISC_PACKAGE_INSTALLDIR` parameter required.
Linux	RPM	Single instance installation, always /usr/cachesys.

Important: The installation directory of a Caché instance cannot be changed following installation. The directory must be a fully resolved physical path, containing no symbolic links.

1.1.2.1 Installation Directory Restrictions

You *cannot* install Caché into a destination directory that has any of the following characteristics:

- It is a UNC (non-local) path.
- It is at the root level of a drive (such as C:\).
- It is anywhere under the \Program Files directory.
- It has a caret (^) in the pathname.
- It has a character that is not in the US ASCII character set.

1.1.3 Setup Type

During installation, you can choose which components of Caché you would like to install. The options are:

- **Development** — Installs the Caché Database Engine (Samples Database, User Database, SQL Gateway, Server Monitoring Tools), Studio, all supported language bindings, and xDBC (ODBC and JDBC) drivers. Select this option if you plan to use this instance to perform both client and server tasks.

  Note: For Ensemble and HealthShare only, the Development Setup Type also installs the EnsDemo Database.

- **Server** — Installs Caché Database Engine (Samples database, User database, SQL Gateway, Server monitoring tools) and CSP Gateway. Select this option if you plan to use this instance as a Caché database server which can be accessed by Caché clients.

  Note: For Ensemble and HealthShare only, the Server Setup Type also installs the EnsDemo Database.

- **Custom** — Installs/uninstalls specific components. Select this option if you would like to install or remove specific Caché components.

On Windows, you can choose from these two additional setup types:

- **Client** — Installs Studio and the xDBC (ODBC and JDBC) drivers. Select this option if you plan to use this instance as a client to a Caché database server on this or another computer.
- **Web Server** — Installs CSP Gateway (IIS, Apache 2.0, Apache 2.2). Select this option if you want to install only those parts of Caché that are required on a CSP Gateway machine.

The following table identifies which component groups are installed for each setup type. The **Custom** setup type lets you select specific component groups/components to install or remove; see the Performing a Caché Custom Installation section of this book.

Table 1–1: Components Installed by Setup Type

Component Group	Components	Development	Server	Client	Web
Caché Database Engine (Caché Server)	Server Monitoring Tools Samples database User database SQL Gateway Agent Service (ISCAgent) Apache FOP (Formatting Objects Processor)	Yes	Yes	No	No
Caché Launcher (cube)		Yes	Yes	Yes	No
Studio		Yes	No	Yes	No
xDBC	ODBC Driver Java Database Connectivity	Yes	No Yes		No
Caché Application Development	ActiveX Connectivity C++ Binding for Caché Light C++ Binding Java Binding for Caché C++ SDK for Caché Caché Engine Link Libraries Perl Binding for Caché [1] Python Binding for Caché [1] .NET Binding for Caché Threaded Server Libraries Other Samples)	Yes	No	No	No

Component Group	Components	Development	Server	Client	Web
Documentation	PDF Documentation Online Documentation	Yes	Yes	No	No
Web Server Gateway (CSP)	CSP for IIS CSP for Apache 2.0.x CSP for Apache 2.2.x	No	Yes	No	Yes

[1] This component is supported only on the x86-32 bit platform.

1.1.4 Caché Character Width

You must select either 8-bit or Unicode support for your installation:

- **8-bit** — The software handles characters in an 8-bit format.
- **Unicode** — The software handles characters in the Unicode (16-bit) format. Select Unicode if your application uses languages that store data in a Unicode format, such as Japanese.

InterSystems recommends 8-bit character support for locales based upon the Latin-1 character set, ISO 8859–1. Use Unicode if the base character set for your locale is not Latin-1, or if you anticipate handling data from locales based upon a different character set. If you use an 8-bit version of Caché, your data is not portable to 8-bit locales based on a different character set.

Caché allows users to upgrade from 8-bit to Unicode instances; however, these installations will not automatically perform data conversions. For more information on converting data in a database, see the *Database Compatibility Considerations* chapter of the *Caché System Administration Guide*.

CAUTION: If you choose a Unicode installation, you cannot revert to an 8-bit version without potential data loss. This is because an 8-bit version of Caché cannot retrieve 16-bit character data from a database.

Important: If you are installing Ensemble, you must select **Unicode**.

1.1.5 Disk Space Requirements

For every platform, the installation kit must be available, either on your computer or on a network. Specific disk space requirements for each platform are:

- Windows:
  - A Caché installation that includes support for Caché Server Pages (CSP) uses approximately 1500 MB (megabytes) of disk storage (not including disk space for user data).
  - Any system that can effectively support Windows should be sufficiently powerful to run Caché. Caché performance greatly improves with increased processor and disk speed.
- UNIX®, Linux, macOS:
  - A standard Caché installation that includes support for Caché Server Pages (CSP) needs 1600 – 1950 MB (megabytes) of disk space depending on the type of installation you choose.
  - In addition, 200 MB of space is required in the Caché installation directory. The installation procedure confirms that this disk space is available in the specified location before installing.

1.1.6 Supported Platforms and Components

For a list of operating systems platforms on which this version of Caché is supported, see the online *InterSystems Supported Platforms* document for this release.

For a list of web servers on which InterSystems CSP technology is supported, see "Supported Web Servers" in the "Supported Technologies" chapter of the online *InterSystems Supported Platforms* document for this release.

If you are using CSP, install the web server before installing Caché to let the Caché installer configure the web server automatically. See the "CSP Architecture" chapter of the *Using Caché Server Pages* guide for more information.

1.1.7 Caché Private Web Server

With each instance, Caché installs a private web server and a private CSP Gateway to serve CSP pages to ensure proper operation of the Management Portal and Caché Online Documentation.

The private web server is installed to ensure that:

1. The Management Portal runs out of the box.
2. An out-of-the-box testing capability is provided for development environments.

The private web server is not supported for any other purpose.

For deployments of http-based applications, including CSP, Zen, and SOAP over http or https, you should not use the private web server for any application other than the Management Portal; instead, you must install and deploy one of the supported web servers (see "Supported Web Servers" in the online *InterSystems Supported Platforms* document for this release).

- Windows: Its Windows service name is "Web Server for *instname*" where *instname* is the instance name you enter when you install Caché. Caché installs the web server into the *install-dir*\httpd directory, where *install-dir* is the Caché installation directory. It is uninstalled when you uninstall the corresponding Caché instance.

The private web server configuration is preserved through upgrades.

1.1.8 If You Are Upgrading Caché

If you are performing an upgrade, first read and perform all necessary procedures described in the "Upgrading Caché" chapter of this book.

When upgrading, back up your old Caché installation after completing all the pre-installation upgrade tasks and before installing Caché.

1.1.9 Configuring Third-Party Software

InterSystems products often run alongside and interact with non-InterSystems tools. For important information about the effects these interactions can have, see the appendix "Configuring Third-Party Software to Work in Conjunction with InterSystems Products" in the *Caché System Administration Guide*.

1.2 Managing Caché Memory

The goal of memory planning and management is to provide enough memory to all of the entities that use it under all normal operating circumstances. This section discusses the following:

- Calculating Initial Memory Requirements
- Vertically Scaling for Memory
- Large and Huge Pages
- Allocating Memory Within Caché

Important: Correct sizing and allocation of system memory are critical in maximizing the performance and availability of Caché and your application. Before installing Caché, be sure to review the guidelines that follow, calculate your initial memory requirements, and determine whether your system has sufficient memory.

For additional information on allocating memory, see Calculating System Parameters for UNIX®, Linux, and macOS and Shared Memory Allocation on Windows in the chapter "Installing Caché on Windows". For an in-depth look at Caché memory planning and allocation by an InterSystems senior technology architect, see InterSystems Data Platforms and Performance Part 4 - Looking at Memory on InterSystems Developer Community.

1.2.1 Sizing System Memory for Caché

Generally, there are four main consumers of memory on a server hosting a Caché instance. At a high level, you can calculate the amount of physical memory required by simply adding up the requirements of each of the items on the following list:

- Operating system, including the file system cache
- Installed applications
- Caché and application processes

  Caché is process-based. If you look at the operating system statistics while your application is running, you will see numerous processes running as part of Caché.
- Caché shared memory, which includes
  - The database and routine caches
  - The generic memory heap (gmheap)
  - Other shared memory structures

For the best possible performance, all four of these should be maintained in physical (system) memory under all normal operating conditions. Virtual memory and mechanisms for using it such as swap space and paging are important because they enable the system to continue operating during a transient memory capacity problem, but the highest priority is to include enough physical memory to avoid the use of virtual memory.

1.2.1.1 Calculating Initial Memory Requirements

Of course, every application is different and any given system may require a series of operational adjustments to optimize memory use. However, the following list provides general guidelines to use as a basis in sizing memory for your application. Benchmarking and performance load testing the application will further influence your estimate of the ideal memory sizing and parameters.

Important: If you have not configured sufficient physical memory on a Linux system and thus regularly come close to capacity, you run the risk that the out of memory killer may misidentify long-running Caché processes that touch a lot of memory in normal operation, such as the write daemon and CSP server processes, as the source of the problem and terminate them. *This will result in an outage of the Caché instance and require crash recovery at the subsequent startup*. Disabling the out of memory killer is *not recommended*, however, as this safety mechanism keeps your operating system from crashing when memory runs short, giving you a chance to intervene and restore Caché to normal operation. The recommended way to avoid this problem is to configure enough physical memory to avoid any chance of the out of memory killer coming into play. (For a detailed discussion of process memory in Caché, see Caché Process Memory.)

General guidelines for sizing memory for your application are as follows:

- System memory to provision (install in a physical server or allocate to a virtual server)

  Start with 4 to 8 GB per CPU core (physical or virtual). This core count does not include any threads such as Intel HyperThreading (HT) or IBM Simultaneous Multi-Threading (SMT). So, for example, if you have an IBM AIX LPAR with 8 cores allocated, the calculation would be 4-8 GB * 8 = 32 to 64 GB of total RAM allocated to that LPAR, even with SMT-4 enabled (which would appear as 32 logical processors).

- Shared memory to allocate within Caché
  - On servers with less than 64 GB of RAM, allocate
    - 50% of total memory to the database cache
    - 256 MB minimum to the routine cache
    - 256 MB minimum to the generic memory heap
  - On servers with more than 64 GB of RAM, allocate
    - 70% of total memory to the database cache
    - 512 MB minimum to the routine cache
    - 384 MB minimum to gmheap

- Swap space or page file to configure

  As a general guideline, configure the smaller of a) 25 to 50% of your physical memory or b) 32 GB as virtual memory. As previously noted, swapping and paging degrade performance and should come into play only when transient memory capacity problems require it. Further, you should configure alerts to notify operators when the system uses virtual memory so they can take immediate action to avoid more severe consequences.

  Note: When large and huge pages are configured, as is highly recommended, Caché shared memory segments are pinned in physical memory and never swapped out; for more information, see Large and Huge Pages.

Note: If you are configuring a data server in an ECP cluster, see Memory Use on Large ECP Systems in the "Developing Distributed Applications" chapter of the *Caché Distributed Data Management Guide* for important information about adjustments to database cache sizes that may be necessary.

1.2.1.2 Vertically Scaling for Memory

Performance problems in production systems are often due to insufficient memory for application needs. Adding memory to the server hosting one or more Caché instances lets you allocate more to the database cache, the routine cache, generic memory, or some combination. A database cache that is too small to hold the workload's working set forces queries to fall back to disk, greatly increasing the number of disk reads required and creating a major performance problem, so this is

often a primary reason to add memory. Increases in generic memory and the routine cache may also be helpful under certain circumstances.

1.2.1.3 Large and Huge Pages

Where supported, the use of large and huge memory pages can be of significant performance benefit and is highly recommended, as described in the following:

- IBM AIX® — The use of large pages is highly recommended, especially when configuring over 16GB of shared memory (the sum of the database cache, the routine cache, and the generic memory heaps, as discussed in Calculating Initial Memory Requirements).

  By default, when large pages are configured, the system automatically uses them in memory allocation. If shared memory cannot be allocated in large pages, it is allocated in standard (small) pages. However, you can use the memlock parameter for finer-grained control over large pages.

  For more information, see Configuring Large and Huge Pages in this chapter and memlock in the *Configuration Parameter File Reference*.

- Linux (all distributions) — The use of static huge pages (2MB) when available is highly recommended for either physical (bare metal) servers or virtualized servers. Using static huge pages for the Caché shared memory segments yields an average CPU utilization reduction of approximately 10-15% depending on the application.

  By default, when huge pages are configured, Caché attempts to provision shared memory in huge pages on startup. If there is not enough space, Caché reverts to standard pages and orphans the allocated huge page space, potentially causing system paging. However, you can use the memlock parameter to control this behavior and fail at startup if huge page allocation fails.

  For more information, see Configuring Large and Huge Pages in this chapter and memlock in the *Configuration Parameter File Reference*.

- Windows

  The use of large pages is recommended to reduce page table entry (PTE) overhead.

  By default, when large pages are configured, Caché attempts to provision shared memory in large pages on startup. If there is not enough space, Caché reverts to standard pages. However, you can use the memlock parameter to control this behavior and fail at startup if large page allocation fails.

  For more information, see Shared Memory Allocation on Windows in the "Installing Caché on Windows" chapter and memlock in the *Configuration Parameter File Reference*.

1.2.2 Allocating Memory Within Caché

There are two primary steps involved in configuring the way in which a Caché instance uses memory, described in the following sections:

- Allocating Memory for Routine and Database Caches
- Configuring Generic Memory Heap (gmheap)

The first action, allocating memory for routine and database caches, determines memory available to hold code and data. The second action, configuring gmheap, determines memory available for all other purposes. These, taken both separately and together, are important factors in the performance and functioning of the instance. For guidelines for initial memory allocations within Caché, see Calculating Initial Memory Requirements.

Two other memory settings are described in the section:

- Other Caché Memory Settings

Important: Some changes on the Memory and Startup page require an instance restart and some do not. If you modify a setting that requires a restart and save your changes, none of the changes take effect until you restart Caché, even those that by themselves do not require a restart. If a restart is required, the message `Modification saved. You must restart system for the new values to take effect.` displays. After you close the page, the warning message does not appear again to remind you that a restart is required, and it is therefore best to restart the instance immediately.

1.2.2.1 Allocating Memory to the Routine and Database Caches

To allocate memory for routine and database caches,

1. On the Management Portal, navigate to the **Memory and Startup** page (**System Administration** > **Configuration** > **System Configuration** > **Memory and Startup**).
2. Select **Manually**.

Important: When Caché is first installed, memory for routine and database caches is set, by default, to be **Automatically** allocated. With this default, Caché allocates a conservative fraction of the available physical memory for the database cache, not to exceed 1 GB. *This setting is not appropriate for production use.* Before deploying the system for production use or before performing any tests or benchmarking intended to simulate production use, you must manually create an appropriate memory allocation for the database cache (typically as much memory as possible after taking into account the needs of application and operating system processes) by selecting **Manually** and following the procedures described in this section.

Allocating Memory to the Routine Cache

Memory Allocated for Routine Cache (MB) — The routine cache specifies the system memory allocated for caching server code.

Caché takes the total amount of memory you allocate for the routine cache and creates buffers of different sizes according to this formula: half the total memory in 64 KB buffers, three-eighths in 16 KB buffers, and one-eighth in 4 KB buffers. For example, if you allocate 500 MB, Caché creates 3906 64 KB buffers (250 MB), 11718 16 KB buffers (187.5 MB), and 15,625 4 KB buffers (62.5 MB). These groups of buffers are sometimes called *pools*, as in "the 16 K buffer pool".

Bear in mind the following points regarding the routine cache memory allocation:

- For typical production instances, a good starting allocation is 350-400 MB. However, the ideal allocation for a given application depends on may factors, and adjustment may be necessary to optimize performance.
- The maximum number of buffers that Caché allocates to any pool is 65,529. The format for Caché routines does not allow more than 32,768 characters for literal strings regardless of the setting for the maximum routine size.
- The minimum allocation is 35 MB; the instance will allocate this much even if you specify less. The minimum number of buffers in a pool is 430.

Note: You can also allocate memory to the routine cache using the **routines** setting in the cache.cpf file; for more information, see routines in the *Caché Parameter File Reference*.

Allocating Memory to the Database Cache

Memory Allocated for [*blocksize*] **Database Cache (MB)** — The database cache specifies the system memory allocated for buffering data; this is also called creating *global buffers*. The database cache and the memory allocated to it are sometimes referred to as the *global buffer pool*.

Enter a separate allocation for each enabled database block size listed. The 8K block size is required and is listed by default. To enable more database block sizes (16K, 32K, 64K), use the **DBSizesAllowed** setting on the **Startup Settings** page (**System**

Administration > **Additional Settings** > **Startup**); see DBSizesAllowed in the *Caché Additional Configuration Settings Reference* for more information.

Both block size and the maximum number of buffers available have implications for performance. To determine how many global buffers Caché will create for databases with a particular block size, divide the allocation for a block size by the block size; the smaller the block size, the larger the number of global buffers that will be created for databases with that block size. For guidelines for selecting the appropriate block sizes for your applications, see "Large Block Size Considerations" in the "Configuring Caché" chapter of the *Caché System Administration Guide*.

Important: If you are configuring a data server in an ECP cluster, see Memory Use on Large ECP Systems in the "Developing Distributed Applications" chapter of the *Caché Distributed Data Management Guide* for important information about adjustments to database cache sizes that may be necessary.

Note: You can also allocate memory to the database cache using the **globals** setting in the cache.cpf file; for more information, see globals in the *Caché Parameter File Reference*.

1.2.2.2 Configuring Generic Memory Heap (gmheap)

You can configure **gmheap** on the **Advanced Memory** page (**System Administration** > **Configuration** > **Additional Settings** > **Advanced Memory**).

gmheap — The generic memory heap (also known as the shared memory heap) determines the memory available to Caché for purposes other than the routine and database caches.

To see details of used and available memory for **gmheap**, use the **Shared Memory Heap Usage** page (**System Operation** > **System Usage** page; click the **Shared Memory Heap Usage** link).

For more information, see gmheap in the "Advanced Memory Settings" section of the *Caché Additional Configuration Settings Reference* and also Generic (Shared) Memory Heap Usage in the "Monitoring Caché Using the Management Portal" chapter of the *Caché Monitoring Guide*.

1.2.2.3 Other Caché Memory Settings

Other memory settings that you can change on the **Memory and Startup** page are:

- **Maximum per Process Memory (KB)** — The maximum memory allocation for a process for this Caché instance. The default is 262144 KB. The allowed range is 128 KB to 2147483647 KB.

  Note: It is not necessary to reset this value unless you have set it lower than its default (262144 KB). If you receive <STORE> errors, increase the size.

  This amount of process private memory, which is used for symbol table allocation and various other memory requirements (for example I/O device access structures and buffers), is allocated in increasing extents as required by the application until the maximum is reached. The initial allocation is 128 KB. Once this memory is allocated to the process, it is not deallocated until the process exits.

- If you select the **Enable Long Strings** check box, Caché allocates a large string stack to handle long strings for each process.

1.3 File System and Storage Configuration Recommendations

This section provides general recommendations in the following areas:

- File System Recommendations
- Storage Configuration Recommendations

In addition, database configuration recommendations are outlined in Configuring Databases section of the "Configuring Caché" chapter of the *Caché System Administration Guide*.

1.3.1 File System Recommendations

For recommendations about the best file system to use for any given operating system, see "Supported File Systems" in the "Supported Technologies" chapter of the online *InterSystems Supported Platforms* document for this release.

In the interests of performance and recoverability, InterSystems recommends a minimum of four separate file systems for Caché, to host the following:

- Installation files, executables, and system databases (including, by default, the write image journal, or WIJ, file)
- Database files (and optionally the WIJ)
- Primary journal directory
- Alternate journal directory

In addition, you can add another separate file system to the configuration for the WIJ file which, by default, is created in the *install—dir*\mgr\ directory. Ensure that such a file system has enough space to allow the WIJ to grow to its maximum size—that is, the size of the database cache as allocated on the **Memory and Startup** page (**System Administration** > **Configuration** > **System Configuration** > **Memory and Startup**) (see Memory and Startup Settings in the "Configuring Caché" chapter of the *Caché System Administration Guide*). For more information on the WIJ, see the "Write Image Journal" chapter of the *Caché Data Integrity Guide*.

Note: On UNIX®, Linux, and macOS platforms, /usr/local/etc/cachesys is the Caché registry directory and therefore must be on a local filesystem.

In the event of a catastrophic disk failure that damages database files, the journal files are a key element in recovering from backup. Therefore, you should place the primary and alternate journal directories on storage devices that are separate from the devices used by database files and the WIJ. (Journals should be separated from the WIJ because damage to the WIJ could compromise database integrity.) Since the alternate journal device allows journaling to continue after an error on the primary journal device, the primary and alternate journal directories should also be on devices separate from each other. For practical reasons, these different devices may be different logical units (LUNs) on the same storage array; the general rule is the more separation the better, with separate sets of physical drives highly recommended. See Journaling Best Practices in the "Journaling" chapter of the *Caché Data Integrity Guide* for more information about separate journal storage.

The journal directories and the WIJ directory are not configured during installation. For information on changing them after you install Caché, see Configuring Journal Settings in the *Caché Data Integrity Guide*.

InterSystems does not support the use of symbolic links for database directories.

Note: Current storage arrays, especially SSD/Flash-based arrays, do not always allow for the type of segregation recommended in the preceding. When using such a technology, consult and follow the storage vendor's recommendations for performance and resiliency.

In addition, this section includes information about the following:

- Supported File Systems on UNIX®, Linux, and macOS Platforms
- File System Mount Options on UNIX®, Linux, and macOS Platforms

1.3.2 Storage Configuration Recommendations

Many storage technologies are available today, from traditional magnetic spinning HDD devices to SSD and PCIe Flash based devices. In addition, multiple storage access technologies include NAS, SAN, FCoE, direct-attached, PCIe, and virtual storage with hyper-converged infrastructure.

The storage technology that is best for your application depends on application access patterns. For example, for applications that predominantly involve random reads, SSD or Flash based storage would be an ideal solution, and for applications that are mostly write intensive, traditional HDD devices might be the best approach.

The sections that follow provide guidelines as general suggestions. Specific storage product providers may specify separate and even contradictory best practices that should be consulted and followed accordingly.

1.3.2.1 Storage Connectivity

The following considerations apply to storage connectivity.

Storage Area Network (SAN) Fibre Channel

Use multiple paths from each host to the SAN switches or storage controllers. The level of protection increases with multiple HBAs to protect from a single card failure, however a minimum recommendation is to use at least a dual-port HBA.

To provide resiliency at the storage array layer, an array with dual controllers in either an active-active or active-passive configuration is recommended to protect from a storage controller failure, and to provide continued access even during maintenance periods for activities such as firmware updates.

If using multiple SAN switches for redundancy, a good general practice is to make each switch a separate SAN fabric to keep errant configuration changes on a single switch from impacting both switches and impeding all storage access.

Network Attached Storage (NAS)

With 10Gb Ethernet commonly available, for best performance 10Gb switches and host network interface cards (NICs) are recommended.

Having dedicated infrastructure is also advised to isolate traffic from normal network traffic on the LAN. This will help ensure predictable NAS performance between the hosts and the storage. -

Jumbo frame support should be included to provide efficient communication between the hosts and storage.

Many network interface cards (NICs) provide TCP Offload Engine (TOE) support. TOE support is not universally considered advantageous. The overhead and gains greatly depend on the server's CPU for available cycles (or lack thereof). Additionally, TOE support has a limited useful lifetime because system processing power rapidly catches up to the TOE performance level of a given NIC, or in many cases exceeds it.

1.3.2.2 Storage Configuration

The storage array landscape is ever-changing in technology features, functionality, and performance options, and multiple options will provide optimal performance and resiliency for Caché. The following guidelines provide general best practices for optimal Caché performance and data resiliency.

In the past, RAID10 was recommended for maximum protection and performance. However, storage controller capacities, RAID types and algorithm efficiencies, and controller features such as inline compression and deduplication provide more options than ever before. Additionally, your application's I/O patterns will help you decide with your storage vendor which storage RAID levels and configuration provide the best solution.

Where possible, it is best to use block sizes similar to that of the file type. While most storage arrays have a lower limit on the block size that can be used for a given volume, you can approach the file type block size as closely as possible; for example, a 32KB or 64KB block size on the storage array is usually a viable option to effectively support CACHE.DAT files with 8KB block format. The goal here is to avoid excessive/wasted I/O on the storage array based on your application's needs.

The following table is provided as a general overview of storage I/O within a Caché installation.

I/O Type	When	How	Notes
Database reads, mostly random	Continuous by user processes	User process initiates disk I/O to read data	Database reads are performed by daemons serving web pages, SQL queries, or direct user processes
Database writes, ordered but non-contiguous	Approx. every 80 seconds or when pending updates reach threshold percentage of database cache, whichever comes first	Database write daemons (8 processes)	Database writes are performed by a set of database system processes known as write daemons. User processes update the database cache and the trigger (time or database cache percent full) commits the updates to disk using the write daemons. Typically expect anywhere from a few MBs to several GBs that must be written during the write cycle depending on update rates.
WIJ writes, sequential	Approx. every 80 seconds or when pending updates reach threshold percentage of database cache, whichever comes first	Database master write daemon (1 process)	The WIJ is used to protect physical database file integrity from system failure during a database write cycle. Writes are approximately 256KB each in size.
Journal writes, sequential	Every 64KB of journal data or 2 seconds, or sync requested by ECP, Ensemble, or application	Database journal daemon (1 process)	Journal writes are sequential and variable in size from 4KB to 4MB. There can be as low as a few dozen writes per second to several thousand per second for very large deployments using ECP and separate application servers.

Bottlenecks in storage are one of the most common problems affecting database system performance. A common error is sizing storage for data capacity only, rather than allocating a high enough number of discrete disks to support expected Input/Output Operations Per Second (IOPS).

I/O Type	Average Response Time	Maximum Response Time	Notes
Database block size random read (non-cached)	<=6 ms	<=15 ms	Database blocks are a fixed 8KB, 16KB, 32KB, or 64KB—most reads to disk will not be cached because of large database cache on the host.
Database block size random write (cached)	<=1 ms	<2 ms	All database file writes are expected to be cached by the storage controller cache memory.
4KB to 4MB journal write (without ECP)	<=2 ms	<=5 ms	Journal writes are sequential and variable in size from 4KB to 4MB. Write volume is relatively low when no ECP application servers are used.
4KB to 4MB journal write (with ECP)	<=1 ms	<=2 ms	Journal synchronization requests generated from ECP impose a stringent response time requirement to maintain scalability. The synchronization requests issue can trigger writes to the last block in the journal to ensure data durability.

Please note that these figures are provided as guidelines, and that any given application may have higher or lower tolerances and thresholds for ideal performance. These figures and I/O profiles are to be used as a starting point for your discussions with your storage vendor.

1.4 Preparing for Caché Security

The material in this section is intended for those using Caché security features. For an overview of those features, especially the authentication and authorization options, review the "Introduction" to the *Caché Security Administration Guide*. This material can help you select the security level for your site, which determines the required tasks to prepare the security environment before installing Caché.

This section covers the following topics:

- Preparing the Security Environment for Kerberos — If you are not using the Kerberos authentication method in your environment, you can bypass this section.
- Initial Caché Security Settings — This section describes the characteristics of the different default security settings. It is particularly useful if you are choosing to use Normal or Locked Down Caché security.

Important: If your security environment is more complex than those this document describes, contact the InterSystems Worldwide Response Center (WRC) for guidance in setting up such an environment.

1.4.1 Preparing the Security Environment for Kerberos

These sections describe the installation preparation for three types of environments:

1. Windows-only Environment

   This configuration uses a Windows domain controller for KDC functionality with Caché servers and clients on Windows machines. A domain administrator creates domain accounts for running the Caché services on Caché servers.

   See the Creating Service Accounts on a Windows Domain Controller for Windows Caché Servers section for the requirements of using Windows Caché servers. Depending on the applications in use on your system, you may also need to perform actions described in the Configuring Windows Kerberos Clients section.

2. Mixed Environment Using a Windows Domain Controller

   This configuration uses a Windows domain controller with Caché servers and clients on a mix of Windows and non-Windows machines. See the following sections for the requirements for using both Windows and non-Windows Cache servers:

   - Creating Service Accounts on a Windows Domain Controller for Windows Caché Servers
   - Creating Service Accounts on a Windows Domain Controller for Non-Windows Caché Servers
   - Depending on the applications in use on your system, you may also need to perform actions described in the Configuring Windows Kerberos Clients section.

3. Non-Windows Environment

   This configuration uses a UNIX® or Kerberos KDC with Caché servers and clients all on non-Windows machines. See the following two sections for the requirements for using a UNIX® or macOS KDC and Caché servers:

   - Creating Service Principals on a KDC for Non-Windows Caché Servers
   - Testing Kerberos KDC Functions

All Caché supported platforms have versions of Kerberos supplied and supported by the vendor; see the appropriate operating system documentation for details. If you choose to use Kerberos, you must have a Kerberos key distribution center (KDC) or a Windows domain controller available on your network. Microsoft Windows implements the Kerberos authentication protocol by integrating the KDC with other security services running on the domain controller.

A Note on Terminology

This document refers to related, but distinct entities:

- Service account — An entity within an operating system, such as Windows, that represents a software application or service.
- Service principal — A Kerberos entity that represents a software application or service.

1.4.1.1 Creating Service Accounts on a Windows Domain Controller for Windows Caché Servers

Before installing Caché in a Windows domain, the Windows domain administrator must create a service account for each Caché server instance on a Windows machine using the Windows domain controller.

Account Characteristics

When you create this account on the Windows domain controller, configure it as follows:

- Set the account's **Password never expires** property.
- Make the account a member of the **Administrators** group on the Caché server machine.

- Add the account to the **Log on as a service** policy.

Important: If a domain-wide policy is in effect, you must add this service account to the policy for Caché to function properly.

Names and Naming Conventions

In an environment where clients and servers are exclusively on Windows, there are two choices for naming service principals:

- Follow the standard Kerberos naming conventions. This ensures compatibility with any non-Windows systems in the future.
- Use any unique string.

Each of these choices involves a slightly different process of configuring a connection to a server as described in the following sections.

Names That Follow Kerberos Conventions

For a name that follows Kerberos conventions, the procedure is:

1. Run the Windows **setspn** command, specifying the name of service principal in the form *service_principal*/*fully_qualified_domain_name*, where *service_principal* is typically `cache` and *fully_qualified_domain_name* is the machine name along with its domain. For example, a service principal name might be `cache/cacheserver.example.com`. For detailed information on the **setspn** tool, see the Setspn page in the Microsoft documentation.
2. In the **Caché Server Manager** dialog for adding a new preferred server, choose Kerberos. What you specify for the **Service Principal Name** field should match the principal name specified in **setspn**.

For detailed information on configuring remote server connections, see the "Connecting to Remote Servers" chapter of the *Caché System Administration Guide*.

Names That Are Unique Strings

For a name that uses any unique string, the procedure is:

1. Choose a name for the service principal.
2. In the **Caché Server Manager** dialog for adding a new preferred server, choose Kerberos. Specify the selected name for the service principal in the **Service Principal Name** field.

If you decide not to follow Kerberos conventions, a suggested naming convention for each account representing a Caché server instance is "`cache`*HOST*", which is the literal, `cache`, followed by the host computer name in uppercase. For example, if you are running a Caché server on a Windows machine called WINSRVR, name the domain account `cacheWINSRVR`.

For more information on configuring remote server connections, see the "Connecting to Remote Servers" chapter of the *Caché System Administration Guide* for the detailed procedure.

1.4.1.2 Creating Service Accounts on a Windows Domain Controller for Non-Windows Caché Servers

Before you install Caché in a Windows domain, you need to create a service account on the Windows domain controller for each Caché server on a non-Windows machine. Create one service account for each machine, regardless of the number of Caché server instances on that machine.

A suggested naming convention for these accounts is "`cache`*HOST*," which is the literal, `cache`, followed by the host computer name in uppercase. For example, if you run a Caché server on a non-Windows machine called UNIXSRVR, name

the domain account `cacheUNIXSRVR`. For Caché servers on non-Windows platforms, this is the account that maps to the Kerberos service principal.

Important: When you create this account on the Windows domain controller, Caché requires that you set the **Password never expires** property for the account.

To set up a non-Windows Caché server in the Windows domain, it must have a keytab file from the Windows domain. A keytab file is a file containing the service name for the Caché server and its key.

To accomplish this, map the Windows service account (`cacheUNIXSRVR`, in this example) to a service principal on the Caché server and extract the key from the account using the **ktpass** command-line tool on the domain controller; this is available as part of the Windows support tools from Microsoft.

The command maps the account just set up to an account on the UNIX®/Linux machine; it also generates a key for the account. The command must specify the following parameters:

Parameter	Description
/princ	The principal name (in the form *cache/<fully qualified hostname>@<kerberos realm>*).
/mapuser	The name of the account created (in the form *cache<HOST>*).
/pass	The password specified during account creation.
/crypto	The encryption type to use (use the default unless specified otherwise).
/out	The keytab file you generate to transfer to the Caché server machine and replace or merge with your existing keytab file.

Important: The principal name on UNIX®/Linux platforms must take the form shown in the table with the literal `cache` as the first part.

Once you have generated a key file, move it to a file on the Caché server with the *key file characteristics* described in the following section.

1.4.1.3 Creating Service Principals on a KDC for Non-Windows Caché Servers

In a non-Windows environment, you must create a service principal for each UNIX®/Linux or macOS Caché server that uses a UNIX®/Linux or macOS KDC. The service principal name is of the form *cache/<fully qualified hostname>@<kerberos realm>*.

Key File Characteristics

Once you have created this principal, extract its key to a key file on the Caché server with the following characteristics:

- On most versions of UNIX®, the pathname is *install-dir*/mgr/cache.keytab. On macOS and SUSE Linux, the pathname is /etc/krb5.keytab.
- It is owned by the user that owns the Caché installation and the group `cacheusr`.
- Its permissions are `640`.

1.4.1.4 Configuring Windows Kerberos Clients

If you are using Windows clients with Kerberos, you may also need to configure these so that they do not prompt the user to enter credentials. This is required if you are using a program that cannot prompt for credentials — otherwise, the program is unable to connect.

To configure Windows not to prompt for credentials, the procedure is:

1. On the Windows client machine, start the registry editor, **regedit.exe**.
2. Go to the HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\Kerberos\Parameters key.
3. In that key, set the value of *AllowTgtSessionKey* to 1.

1.4.1.5 Testing Kerberos KDC Functions

When using Kerberos in a system of only non-Windows servers and clients, it is simplest to use a native UNIX®/Linux KDC rather than a Windows domain controller. Consult the vendor documentation on how to install and configure the KDC; these are usually tasks for your system administrator or system manager.

When installing Kerberos, there are two sets of software to install:

- The KDC, which goes on the Kerberos server machine.
- There also may be client software, which goes on all machines hosting Kerberos clients. This set of software can vary widely by operating system. Consult your operating system vendor documentation for what client software exists and how to install it.

After installing the required Kerberos software, you can perform a simple test using the **kadmin**, **kinit**, and **klist** commands to add a user *principal* to the Kerberos database, obtain a TGT (ticket-granting ticket) for this user, and list the TGT.

Once you successfully complete a test to validate that Kerberos is able to provide tickets for registered principals, you are ready to install Caché.

1.4.2 Initial Caché Security Settings

During installation, there is a prompt for one of three sets of initial security settings: Minimal, Normal, and Locked Down. This selection determines the initial authorization configuration settings for Caché services and security, as shown in the following sections:

- Initial User Security Settings
- Initial Service Properties

If you select Normal or Locked Down for your initial security setting, you must provide additional account information to the installation procedure. If you are using Kerberos authentication, you must select Normal or Locked Down mode. See the Configuring User Accounts section for details.

Important: If you are concerned about the visibility of data in memory images (often known as core dumps), see the section "Protecting Sensitive Data in Memory Images" in the "System Management and Security" chapter of the *Caché Security Administration Guide*.

1.4.2.1 Initial User Security Settings

The following tables show the user password requirements and settings for predefined users based on which security level you choose.

Table 1–2: Initial User Security Settings

Security Setting	Minimal	Normal	Locked Down
Password Pattern	3.32ANP	3.32ANP	8.32ANP
Inactive Limit	0	90 days	90 days
Enable _SYSTEM User	Yes	Yes	No
Roles assigned to UnknownUser	%All	None	None

You can maintain both the password pattern and inactive limit values from the **System-wide Security Parameters** page (**System** > **Security Management** > **System Security Settings** > **System-wide Security Parameters**). See the System-wide Security Parameters section of the "System Management and Security" chapter of the *Caché Security Administration Guide* for more information.

After installation, you can view and maintain the user settings at the **Users** page (**System** > **Security Management** > **Users**).

Password Pattern

When Caché is installed, it has a default set of password requirements. For locked-down installations, the initial requirement is that a password be from 8 to 32 characters, and can consist of alphanumeric characters or punctuation; the abbreviation for this is 8.32ANP. Otherwise, the initial requirement is that the password be from 3 to 32 characters, and can consist of alphanumeric characters or punctuation (3.32ANP).

Inactive Limit

This value is the number of days an account can be inactive before it is disabled. For minimal installations, the limit is set to 0 indicating that accounts are not disabled, no matter how long they are inactive. Normal and locked-down installations have the default limit of 90 days.

Enable _SYSTEM User

In versions of Caché prior to 5.1, all installed systems included an SQL System Manager user named _SYSTEM with a password of SYS. This Caché version creates the _SYSTEM and the following additional predefined users, using the password you provide during the installation: _SYSTEM, Admin, SuperUser, CSPSystem, and the instance owner (the installing user on Windows and the username specified by the installer on other platforms).

For more details on these predefined users, see the Predefined User Accounts section of the "Users" chapter of the *Caché Security Administration Guide*.

Roles Assigned to UnknownUser

When an unauthenticated user connects, Caché assigns a special name, `UnknownUser`, to $USERNAME and assigns the roles defined for that user to $ROLES. The UnknownUser is assigned the %All role with a Minimal-security installation; UnknownUser has no roles when choosing a security level other than Minimal.

For more details on the use of $USERNAME and $ROLES, see the "Users" and "Roles" chapters of the *Caché Security Administration Guide*.

1.4.2.2 Initial Service Properties

Services are the primary means by which users and computers connect to Caché. For detailed information about the Caché services see the "Services" chapter of the *Caché Security Administration Guide*.

Table 1–3: Initial Service Properties

Service Property	Minimal	Normal	Locked Down
Use Permission is Public	Yes	Yes	No
Requires Authentication	No	Yes	Yes
Enabled Services	Most	Some	Fewest

Use Permission is Public

If the Use permission on a service resource is Public, any user can employ the service; otherwise, only privileged users can employ the service.

Requires Authentication

For installations with initial settings of locked down or normal, all services require authentication of some kind (Caché login, operating-system–based, or Kerberos). Otherwise, unauthenticated connections are permitted.

Enabled Services

The initial security settings of an installation determine which of certain services are enabled or disabled when Caché first starts. The following table shows these initial settings:

Table 1–4: Initial Enabled Settings for Services

Service	Minimal	Normal	Locked Down
%Service_Bindings	Enabled	Enabled	Disabled
%Service_CSP	Enabled	Enabled	Enabled
%Service_CacheDirect	Enabled	Disabled	Disabled
%Service_CallIn	Enabled	Disabled	Disabled
%Service_ComPort	Disabled	Disabled	Disabled
%Service_Console*	Enabled	Enabled	Enabled
%Service_ECP	Disabled	Disabled	Disabled
%Service_MSMActivate	Disabled	Disabled	Disabled
%Service_Monitor	Disabled	Disabled	Disabled
%Service_Shadow	Disabled	Disabled	Disabled
%Service_Telnet*	Disabled	Disabled	Disabled
%Service_Terminal†	Enabled	Enabled	Enabled
%Service_WebLink	Disabled	Disabled	Disabled

* Service exists on Windows servers only

† Service exists on non-Windows servers only

After installation, you can view and maintain these services at the **Services** page (**System** > **Security Management** > **Services**).

1.4.2.3 Configuring User Accounts

If you select Normal or Locked Down for your initial security setting, you must provide additional information to the installation procedure:

1. **User Credentials** for *Windows* server installations only — Choose an existing Windows user account under which to run the Caché service. You can choose the default system account, which runs Caché as the Windows Local System account, or enter a defined Windows user account.

   Important: If you are using Kerberos, you must enter a defined account that you have set up to run the Caché service. InterSystems recommends you use a separate account specifically set up for this purpose as described in the Creating Service Principals for Windows Caché Servers section.

   If you enter a defined user account, the installation verifies the following :

   - The account exists on the domain.
   - You have supplied the correct password.
   - The account has local administrative privileges on the server machine.

2. **Caché Users Configuration** for *Windows* installations — The installation creates a Caché account with the %All role for the user that is installing Caché to grant that user access to services necessary to administer Caché.

   Owner of the instance for *non-Windows* installations — Enter a username under which to run Caché. Caché creates an account for this user with the %All role.

   Enter and confirm the password for this account. The password must meet the criteria described in the Initial User Security Settings table.

   Setup creates the following Caché accounts for you:_SYSTEM, Admin, SuperUser, CSPSystem, and the instance owner (installing user on Windows or specified user on other platforms) using the password you provide.

Important: If you select Minimal for your initial security setting on a *Windows* installation, but Caché requires network access to shared drives and printers, you must manually change the Windows user account under which to run the Caché service. Choose an existing or create a new account that has local administrative privileges on the server machine.

The instructions in the platform-specific chapters of this book provide details about installing Caché. After reading the *Caché Security Administration Guide* introduction and following the procedures in this section, you are prepared to provide the pertinent security information to these installation procedures.

1.5 Preparing to Install Caché on UNIX®, Linux, and macOS

Read the following sections for information that applies to your platform:

- Supported File Systems on UNIX®, Linux, and macOS Platforms
- File System Mount Options on UNIX®, Linux, and macOS Platforms
- Calculating System Parameters for UNIX®, Linux, and macOS
- Platform Configuration Issues
- Special Considerations

1.5.1 Supported File Systems on UNIX®, Linux, and macOS Platforms

A complete list of file systems supported on UNIX®/Linux platforms, see "Supported File Systems" in the "Supported Technologies" chapter of the online *InterSystems Supported Platforms* document for this release.

1.5.2 File System Mount Options on UNIX®, Linux, and macOS Platforms

This section describes the following mount options::

- Buffered I/O vs. Direct I/O
- noatime Mount Option

1.5.2.1 Buffered I/O vs. Direct I/O

In general, most of the supported UNIX®, Linux, and macOS file systems and operating systems offer two distinct I/O options, using either program control, a mount option, or both:

- Buffered I/O, in which the operating system caches reads and writes, is the default.
- Direct I/O is an option in which reads and writes bypass the operating system cache. Some platforms further distinguish an optimized form of direct I/O, called concurrent I/O which, if offered, is preferred.

The use of buffered and direct I/O in Caché varies by platform, file system, and the nature of the files that are stored on the file system, as follows:

- Journal files

  Some platforms have specific recommendations to use direct or concurrent I/O mount options for optimal performance, as documented in "Supported File Systems" in the online *InterSystems Supported Platforms* document for this release. On other platforms, Caché uses direct I/O automatically for journal files as appropriate and no special consideration is required.

- Installation files, executables, and system databases

  This file system should be mounted to use buffered I/O (the default option, and on some platforms the only option).

- Databases (CACHE.DAT files)

  The use of direct I/O (or concurrent I/O) varies in order to optimize I/O characteristics for database files on each platform, as detailed in the following. In all cases, Caché uses its own database cache, so buffering at the operating system level is not advantageous for database files. You must ensure that sufficient database cache is configured; this is particularly true on platforms on which Caché utilizes direct I/O, since operating system buffering cannot make up for an insufficient database cache.

  – IBM AIX

    Caché uses concurrent I/O for database files regardless of whether the **cio** file system mount option is used.

    Note: On AIX, in unusual configurations in which an external command is used to read a database file while Caché is running, the external command may fail because the file is opened for concurrent I/O by Caché. An example is performing an external backup using the **cp** command instead of a more sophisticated backup or snapshot utility. Mounting the file system with the **cio** option resolves this by forcing all programs to open files with concurrent I/O.

  – Linux

Caché uses buffered I/O for database files. If using the VxFS file system, this can be overridden by mounting the file system for concurrent I/O with the **cio** mount option.

– macOS

Caché uses buffered I/O for database files.

- External application files and streams

  Applications that use external files typically benefit from those files being located on a buffered file system.

1.5.2.2 noatime Mount Option

Generally, it is advisable to disable updates to the file access time when this option is available. This can typically be done using the **noatime** mount option on various file systems.

1.5.3 Calculating System Parameters for UNIX®, Linux, and macOS

This section explains how you can calculate the best parameters for your system in these sections:

- Determining Initial System Configuration — configure large or huge pages and calculate disk requirements, maximum buffers, maximum users, and maximum database size.
- Configuring UNIX® Kernel Parameters — set values for tunable UNIX® parameters and other platform-specific memory management issues.
- Platform Configuration Issues — configuration issues for individual UNIX®/Linux platform-specific issues

For optimal Caché performance, you need to calculate proper values for certain Caché system parameters. These values allow you to determine whether you need to adjust certain system level parameters. The values you choose should minimize swapping and paging that require disk accesses, and thus improve system performance.

Review this section carefully and calculate the proper values for both your operating system and Caché before proceeding. Use the tables provided here to record the current and calculated values for your system level parameters. You can then refer to these tables when you install Caché. After your system is running, you may need to adjust these values to gain optimal performance.

If you are not already familiar with the memory organization at your operating system level, consult the appropriate system documentation.

1.5.3.1 Determining Initial System Configuration

This section covers some basic configuration topics for UNIX, Linux, and macOS systems. Because requirements vary by platform, consult your platform documentation for additional information.

- Configuring Large and Huge Pages
- Calculating Disk Requirements
- Determining the Number of Global Buffers
- Determining the Number of Routine Buffers
- Determining the Maximum Number of Users
- Determining the Maximum Database Size

See the section Managing Caché Memory for information on the two primary ways that you can manage memory in Caché.

Configuring Large and Huge Pages

As noted in Large and Huge Pages, the use of large and huge memory pages where supported can be of significant performance benefit and is highly recommended.

Support for Huge Memory Pages for Linux

The default memory page size on Linux systems is 4 KB. Most current Linux distributions include an option for Huge Pages, that is, a memory page size of 2 MB or 1 GB depending on system configuration. Use of Huge Pages saves memory by saving space in page tables. When Huge Pages are configured, the system automatically uses them in memory allocation. InterSystems recommends the use of Huge Pages on systems hosting Caché under most circumstances.

On Linux platforms, if shared memory is allocated in Huge Pages, they are automatically locked in memory and no further action is required. You can configure Caché to lock the shared memory segment in memory to prevent paging as described in the memlock entry of the *Configuration Parameter File Reference*.

Important: With the 2.6.38 kernel, some Linux distributions have introduced Transparent Huge Pages (THP) to automate the creation, management, and use of HugePages. However, THP does not handle the shared memory segments that make up the majority of Caché's memory allocated, and can cause memory allocation delays at runtime that may affect performance, especially for applications that have a high rate of job or process creation. For these reasons, InterSystems recommends that THP be *disabled* on all systems hosting Caché. For more detailed information on this topic, see Linux Transparent Huge Pages and the impact to Caché on InterSystems Developer Community.

To configure Huge Pages on Linux, do the following:

1. Check the status.

   /proc/meminfo contains Huge Pages information. By default, no Huge Pages are allocated. Default Huge Page size is 2 MB. For example:

   ```
   HugePages_Total:     0
   HugePages_Free:      0
   HugePages_Rsvd:      0
   Hugepagesize:     2048 KB
   ```

2. Change the number of Huge Pages.

   You can change the system parameter directly: For example, to allocate 2056 Huge Pages, execute:

   ```
   # echo 2056 > /proc/sys/vm/nr_hugepages
   ```

   Note: Alternatively, you can use **sysctl(8)** to change it:

   ```
   # sysctl -w vm.nr_hugepages=2056
   ```

   Huge pages must be allocated contiguously, which may require a reboot. Therefore, to guarantee the allocation, as well as to make the change permanent, do the following:

   a. Enter a line in /etc/sysctl.conf file:

      ```
      echo "vm.nr_hugepages=2056" >> /etc/sysctl.conf
      ```

   b. Reboot the system.

   c. Verify **meminfo** after reboot; for example:

      ```
      [root woodcrest grub]# tail -4 /proc/meminfo
      HugePages_Total:  2056
      HugePages_Free:   2056
      HugePages_Rsvd:      0
      Hugepagesize:     2048 KB
      ```

3. Verify the use of Huge Pages by Caché.

   When Caché is started, it reports how much shared memory was allocated; for example, a message similar to the following is displayed (and included in the cconsole.log file):

   ```
   Allocated 3580MB shared memory: 3000MB global buffers, 226MB routine buffers
   ```

   The amount of memory available in Huge Pages should be greater than the total amount of shared memory to be allocated; if it is not greater, Huge Pages are not used.

   Note: Huge Pages are allocated from physical memory. Only applications and processes using Huge Pages can access this memory.

   Physical memory not allocated for Huge Pages is the only memory available to all other applications and processes.

   It is not advisable to specify **HugePages_Total** much higher than the shared memory amount because the unused memory will not be available to other components.

   If Caché fails to allocate Huge Pages on start-up and switches to standard pages, Caché will be allocating shared memory from the same memory pool as all other jobs.

   When Caché is configured to lock the shared memory segment in memory to prevent paging, Huge Pages can provide the required increase in the maximum size that may be locked into memory, as described in the Locked-in Memory section of the *Red Hat Linux Platform Notes* in this chapter.

Support for Large (16 MB) Pages on IBM AIX®

AIX® supports multiple page sizes: 4 KB, 64 KB, 16 MB, and 16 GB. Use of 4 KB and 64 KB pages is transparent to Caché. In order for Caché to use 16 MB large pages, you must configure them within AIX®. AIX® does not automatically change the number of configured large or huge pages based on demand. Currently, Caché does not use 16 GB huge pages.

Large pages should be configured only in high-performance environments because memory allocated to large pages can be used only for large pages.

To allocate large pages, users must have the **CAP_BYPASS_RAC_VMM** and **CAP_PROPAGATE** capabilities or have root authority unless *memlock=64*.

By default, when large pages are configured, the system automatically uses them in memory allocation. If shared memory cannot be allocated in large pages then it is allocated in standard (small) pages. For finer grain control over large pages, see memlock in the *Caché Parameter File Reference*.

Configuring Large Pages for AIX®

Configure large pages using the **vmo** command as follows:

vmo -r -o lgpg_regions=*<LargePages>* -o lgpg_size=*<LargePageSize>*

where *<LargePages>* specifies the number of large pages to reserve, and *<LargePageSize>* specifies the size, in bytes, of the hardware-supported large pages.

Note: On systems that support dynamic Logical PARtitioning (LPAR), you can omit the **-r** option to dynamically configure large pages without a system reboot.

For example, the following command configures 1 GB of large pages:

```
# vmo -r -o lgpg_regions=64 -o lgpg_size=16777216
```

Once you have configured large pages, run the **bosboot** command to save the configuration in the boot image. After the system comes up, enable it for pinned memory using the following **vmo** command:

```
vmo -o v_pinshm=1
```

However, if *memlock=64*, `vmo -o v_pinshm=1` is not required. For more information on *memlock*, see memlock in the *Caché Parameter File Reference*.

Calculating Disk Requirements

Your Caché instance needs disk space for the following items:

- 67 MB for Caché.
- 3 MB for the Caché Server Pages (CSP).
- 3.5 MB for Caché ODBC support.
- 2.5 MB for the Caché manager sources.
- 6.6 MB for the Caché engine link libraries.
- Space for your Caché application database.
- Approximately 12.5% of the buffer pool size for the initial size of the write image journal file. If your disk does not have enough space for the write image journal file, when you start Caché it displays a message indicating that the system did not start.
- Desired space for journal files.

Although you do not need to remove any installation files after completing the installation procedure, you can do so if you are short on disk space. The installation program tells you how much space can be saved, and asks if you want to delete the installation files.

Determining the Number of Global Buffers

Caché supports the following maximum values for the number of global buffers:

- For 32-bit platforms, any 8-KB buffers that are:
  - Less than 2GB for 32-bit platforms

    The 2-GB value is the total address space the operation system allocates for the process data, which includes not only shared memory, but other Caché and operating system data as well. Therefore, it represents an upper limit that is not achievable in practice.
- For 64-bit platforms:

  The number of global buffers is limited only by the operating system and the available memory.

For guidelines for your initial allocation of memory to the database cache (the global buffer pool), see Calculating Initial Memory Requirements; for the procedure for this allocation, see Allocating Memory Within Caché. For further information about the database cache, see globals in the "config" section of the *Caché Parameter File Reference* and Memory and Startup Settings in the "Configuring Caché" chapter of the *Caché System Administration Guide*.

Determining Number of Routine Buffers

Caché supports the following maximum value for the number of routine buffers:

65,535

Set your values to less than this maximum number of buffers.

For guidelines for your initial allocation of memory to the database cache (the global buffer pool), see Calculating Initial Memory Requirements; for the procedure for this allocation, see Allocating Memory Within Caché. For further information

about the database cache, see routines in the "config" section of the *Caché Parameter File Reference* and Memory and Startup Settings in the "Configuring Caché" chapter of the *Caché System Administration Guide*.

Determining Maximum Number of Users

The maximum users allowed by Caché is the *lowest* of the following values:

- License limit
- # of semaphores - 4

For more information, see Determining License Capacity and Usage in the "Managing Caché Licensing" chapter of the *Caché System Administration Guide*.

Determining Maximum Database Size

The *ulimit* parameter in UNIX® determines the maximum file size available to a process. For the Caché Manager group, the value of *ulimit* should either be unlimited or as large as the largest database you may have.

For more information, see Configuring Databases in the "Configuring Caché" chapter of the *Caché System Administration Guide*.

1.5.3.2 Configuring UNIX® Kernel Parameters

The following sections describe issues related to tuning and performance on various UNIX® platforms:

- Setting Values for Tunable UNIX® Parameters
- Adjusting Maximum File Size
- Platform Configuration Issues

Setting Values for Tunable UNIX® Parameters

Caché uses a configurable number of semaphores, in sets whose size you define. The parameters *SEMMNI*, *SEMMNS*, and *SEMMSL* reflect the number of semaphores per set and the total number of semaphores Caché uses. The UNIX®/Linux parameters that govern shared memory allocation are *SHMMAX*, *SHMMNI*, *SHMSEG*, and *SHMALL*. Caché uses shared memory and allocates one segment of shared memory; the size of this segment depends on the area set aside for global buffers and routine buffers. It uses the following formula to determine the segment's minimum size:

```
  space required for routine buffers
+  space required for global buffers
+                              4 MB
_____________________________________
=         Shared memory segment size
```

If you are distributing your data across multiple computers, Caché allocates a second segment; by default, there is no memory allocated for the second segment. (If you plan to use distributed data, contact your vendor or InterSystems support for configuration guidelines.) You can alter *NBUF* and *NHBUF* according to other system requirements. Because Caché does all its own disk buffering, you should keep *NBUF* and *NHBUF* small. The following table lists the most common names of the UNIX® parameters that you may need to change, the minimum value InterSystems recommends for each parameter, and a brief description of each. Verify that your parameter values are set to at least the minimum value. Certain parameters may not be implemented on all platforms or may be referred to differently. Refer to platform-specific tuning notes for more information.

Table 1–5: Tunable UNIX® Parameters

Kernel Parameter	Recommended Minimum Value	Definition
CDLIMIT	Number of bytes in largest virtual volume	Maximum size of a file.
MSGMAX	2 KB	Maximum message size, in bytes.

Kernel Parameter	Recommended Minimum Value	Definition
MSGMNI	Number of Caché instances x 3; each Caché instance uses three message queues	Maximum number of uniquely identifiable message queues that may exist simultaneously.
NOFILES	35	Number of open files per process.
SEMMNI	Product of SEMMNI and SEMMSL must be greater than the # of user processes + 4	Number of semaphore identifiers in the kernel; this is the number of unique semaphore sets that can be active at any one time.
SEMMNS	128 or ...	Total number of semaphores in the system. User processes include jobbed processes and all other semaphores required by other software.
	Number of processes expected to run. If the process table might expand, use a larger number to provide for expansion.	
SEMMSL	See SEMMNI	Maximum number of semaphores per identifier list.
SHMALL	60 KB or ...	Maximum total shared memory system-wide. Units should be in KB. 1000 represents the MCOMMON shared region.
	1000 + total global buffer space+ total routine buffer space *	
SHMMNI	3	Maximum number of shared memory identifiers system-wide.
SHMSEG	3	Number of attached shared memory segments per process.
SHMMAX	60 KB or ...	Maximum shared memory segment size in KB.
	1000 + total global buffer space+ total routine buffer space	

* This is the minimum value for *SHMALL* required for Caché UNIX®. You must also take into account any other applications that use shared memory. If you are unsure of other shared memory use, calculate *SHMALL* as *SHMSEG* multiplied by *SHMMAX*, in pages; this larger value suffices in all cases.

Important: Enough swap space must be created to support the memory allocated, unless the operating system documentation explicitly states otherwise. On certain operating systems, Caché creates *locked shared memory segments*, which are not pageable but still may need swap space.

Adjusting Maximum File Size

The hard limit for the maximum file size (*RLIMIT_FSIZE*) on any system running Caché must be *unlimited*. Set the value to *unlimited* on the operating system before installing. Make sure that the limit is set to *unlimited* for both the root user and the user who will run Caché. Caché also sets the process soft limit to *RLIMIT_FSIZE* in its daemons to prevent I/O errors.

Important: Caché will not install or start up if *RLIMIT_FSIZE* is not set to unlimited.

See the operating system documentation for your platform for instructions on how to set the system hard limit for the maximum file size, *RLIMIT_FSIZE*.

1.5.4 Platform Configuration Issues

The following sections contain configuration issues for individual UNIX®/Linux platforms. For more information, consult the system documentation for your platform.

- AIX® Platform Notes
- Red Hat Linux Platform Notes
- SUSE Linux Platform Notes
- Ubuntu Platform Notes

1.5.4.1 AIX® Platform Notes

The default settings of several AIX® parameters can adversely affect performance. The settings and recommendations are detailed for the following:

- I/O Pacing Parameters
- File System Mount Option
- Memory Management Parameters
- AIX® Tunable Parameters

I/O Pacing Parameters

AIX® implements an I/O pacing algorithm that may hinder Caché write daemons. In AIX® 5.2 and AIX® 5.3, I/O pacing is automatically enabled when using HACMP clustering; beginning in AIX® 6.1, however, I/O pacing is enabled on all systems and the default high-water mark is set higher than in earlier releases.

If write daemons are slowing or stalling, you may have to adjust the high-water mark; for information, see the "Using Disk-I/O Pacing" section of the *AIX® Performance Management Guide* at the following IBM web page:http://publib.boulder.ibm.com/infocenter/systems/scope/aix/topic/com.ibm.aix.prftungd/doc/prftungd/disk_io_pacing.htm.

Important: Beginning in AIX® 6.1, you should not have to make any high-water mark adjustments.

If you have questions about the impact to your system, however, contact the InterSystems Worldwide Response Center (WRC) or your AIX® supplier before making any changes. These recommendations are independent of Caché versions and apply to both JFS and Enhanced JFS (JFS2) file systems.

File System Mount Option

Different mount options may improve performance for some workloads.

Note: Non-Caché workloads that benefit from file system caching (for example, operating system-level backups and/or file copies) are slowed by the `cio` mount option.

For more information on JFS2 file systems that contain only journal files, see UNIX® File System Recommendations in the "Journaling" chapter of the *Caché Data Integrity Guide*.

To improve recovery speed using the CACHE.WIJ file after a hard shutdown or system crash, InterSystems recommends a mount option that includes file system buffering (for example, `rw`) for the file system that contains the CACHE.WIJ file.

For information about **mount** options, see the *AIX® Commands Reference* at the following IBM web page: http://publib.boulder.ibm.com/infocenter/systems/scope/aix/topic/com.ibm.aix.cmds/doc/aixcmds3/mount.htm.

Memory Management Parameters

The number of file systems and the amount of activity on them can limit the number of memory structures available to JFS or JFS2, and delay I/O operations waiting for those memory structures.

To monitor these metrics, issue a **vmstat -vs** command, wait two minutes, and issue another **vmstat -vs** command. The output looks similar to the following:

```
# vmstat -vs
              1310720 memory pages
              1217707 lruable pages
               144217 free pages
                    1 memory pools
               106158 pinned pages
                 80.0 maxpin percentage
                 20.0 minperm percentage
                 80.0 maxperm percentage
                 62.8 numperm percentage
               764830 file pages
                  0.0 compressed percentage
                    0 compressed pages
                 32.1 numclient percentage
                 80.0 maxclient percentage
               392036 client pages
                    0 remote pageouts scheduled
                    0 pending disk I/Os blocked with no pbuf
                 5060 paging space I/Os blocked with no psbuf
              5512714 filesystem I/Os blocked with no fsbuf
               194775 client filesystem I/Os blocked with no fsbuf
                    0 external pager filesystem I/Os blocked with no fsbuf
```

If you see an increase in the following parameters, increase the values for better Caché performance:

- *pending disk I/Os blocked with no pbuf*
- *paging space I/Os blocked with no psbuf*
- *filesystem I/Os blocked with no fsbuf*
- *client filesystem I/Os blocked with no fsbuf*
- *external pager filesystem I/Os blocked with no fsbuf*

When increasing these parameters from the default values:

1. Increase the current value by 50%.
2. Check the **vmstat** output.
3. Run **vmstat** twice, two minutes apart.
4. If the field is still increasing, increase again by the same amount; continue this step until the field stops increasing between **vmstat** reports.

Important: Change both the current and the reboot values, and check the **vmstat** output regularly because I/O patterns may change over time (hours, days, or weeks).

See the following IBM web pages for more detailed information:

- For a complete description of each of the fields reported by **vmstat**, see the *vmstat Command* page of *AIX® Commands Reference, Volume 6, v - z* at:

  http://publib.boulder.ibm.com/infocenter/systems/scope/aix/topic/com.ibm.aix.cmds/doc/aixcmds6/vmstat.htm

- For instructions on how to increase these parameters, see the *VMM page replacement tuning* section of the *AIX® Performance Management Guide* at:

  http://publib.boulder.ibm.com/infocenter/systems/scope/aix/topic/com.ibm.aix.prftungd/doc/prftungd/vmm_page_replace_tuning.htm

- For a complete description of managing I/O tunable parameters, see the *ioo Command* page of *AIX® Commands Reference, Volume 3, i - m* at:

  http://publib.boulder.ibm.com/infocenter/systems/scope/aix/topic/com.ibm.aix.cmds/doc/aixcmds3/ioo.htm

AIX® Tunable Parameters

None of the following listed parameters requires tuning because each is dynamically adjusted as needed by the kernel. See the appropriate AIX® operating system documentation for more information.

The following table lists the tunable parameters for the IBM pSeries AIX® 5.2 operating system.

Table 1–6: AIX® Interprocess Communication Tunable Parameters

Parameter	Purpose	Dynamic Values
msgmax	Specifies maximum message size.	Maximum value of 4 MB
msgmnb	Specifies maximum number of bytes on queue.	Maximum value of 4 MB
msgmni	Specifies maximum number of message queue IDs.	Maximum value of 4096
msgmnm	Specifies maximum number of messages per queue.	Maximum value of 524288
semaem	Specifies maximum value for adjustment on exit.	Maximum value of 16384
semmni	Specifies maximum number of semaphore IDs.	Maximum value of 4096
semmsl	Specifies maximum number of semaphores per ID.	Maximum value of 65535
semopm	Specifies maximum number of operations per semop() call.	Maximum value of 1024
semume	Specifies maximum number of undo entries per process.	Maximum value of 1024
semvmx	Specifies maximum value of a semaphore.	Maximum value of 32767
shmmax	Specifies maximum shared memory segment size.	Maximum value of 256 MB for 32-bit processes and 0x80000000u for 64-bit
shmmin	Specifies minimum shared-memory-segment size.	Minimum value of 1
shmmni	Specifies maximum number of shared memory IDs.	Maximum value of 4096

maxuproc

`maxuproc`, which specifies the maximum number of processes than can be started by a single nonroot user, is a tunable parameter that can be adjusted as described in this subsection.

If this parameter is set too low then various components of the operating system can fail as more and more users attempt to start processes; these failures include loss of CSP pages, background tasks failing, etc. Therefore, you should set the `maxuproc` parameter to be higher than the maximum number of processes that might be started by a nonroot user (including interactive users, web server processes, and anything that might start a process).

Note: Do not set the value excessively high because this value protects a server from a runaway application that is creating new processes unnecessarily; however, setting it too low causes unexplained problems.

InterSystems suggests that you set `maxuproc` to be double your expected maximum process count which gives a margin of error but still provides protection from runaway processes. For example, if your system has 1000 interactive users and often runs 500 background processes, then a value of at least 3000 would be a good choice.

The `maxuproc` value can be examined and changed either from the command line or from the **smit**/**smitty** administrator utilities, both as root user, as follows:

- From the command line, view the current setting:

  ```
  # lsattr -E -l sys0 -a maxuproc
  ```

  then modify the value:

  ```
  # chdev -l sys0 -a maxuproc=NNNNNN
  ```

  where *NNNNNN* is the new value.

- From the administrator utility **smit** (or **smitty**) choose **System Environments** > **Change / Show Characteristics of Operating System** > **Maximum number of PROCESSES allowed per user**.

If you increase the value of `maxuproc`, the change is effective immediately. If you decrease the value of `maxuproc`, the change does not take effect until the next system reboot. In both cases the change persists over system reboots.

1.5.4.2 Red Hat Linux Platform Notes

This topic includes the information on the following adjustments:

- Shared Memory Limit
- Locked-in Memory
- Adjustments for Large Number of Concurrent Processes
- Dirty Page Cleanup

Shared Memory Limit

The default shared memory limit (*shmmax*) on Linux platforms is 32 MB. This value is too small for Caché, but it can be changed in the proc file system without a restart.

For example, to allow 128 MB, type the following command:

```
$ echo 134217728 >/proc/sys/kernel/shmmax
```

You can put this command into a startup script.

Alternatively, you can use **sysctl(8)**, if available, to control this parameter. Look for a file called /etc/sysctl.conf and add a line similar to the following:

```
kernel.shmmax = 134217728
```

This file is usually processed at startup, but **sysctl** can also be called explicitly later.

Important: The *msgmni* parameter may also be set too low if you are running more than one instance of Caché on a machine. As stated in the Tunable UNIX® Parameters table, set this value to three times the number of instances of Caché that run simultaneously on your system.

Other parameters are sufficiently sized for a Caché application. To view the values of other parameters, look in the files /usr/src/linux/include/asm-xxx/shmparam.h and /usr/src/linux/include/linux/sem.h.

For more information, reference "The proc File System" chapter of the *Red Hat Enterprise Linux 4: Reference Guide*.

Locked-in Memory

On Linux platforms, you can configure Caché to lock the shared memory segment in memory to prevent paging as described in the memlock entry of the *Caché Parameter File Reference*. If shared memory is allocated in Huge Pages, they are automatically locked in memory and no further action is required. Otherwise, you must increase the maximum size that may be locked into memory. The default value is 32 KB. View the current value using the **ulimit** command.

For example, to display all current limits:

```
bash$ ulimit -a
core file size (blocks, -c) unlimited
data seg size ( KBytes, -d) unlimited
file size (blocks, -f) unlimited
pending signals (-i) 1024
max locked memory (KBytes, -l) 32 <---------- THIS ONE
max memory size (KBytes, -m) unlimited
open files (-n) 1024
pipe size (512 bytes, -p) 8
POSIX message queues (bytes, -q) 819200
stack size ( KBytes, -s) 10240
cpu time (seconds, -t) unlimited
max user processes (-u) 49000
virtual memory ( KBytes, -v) unlimited
file locks (-x) unlimited
```

To display only *max-locked memory*, use the `-l` option:

```
bash$ ulimit -l
32
```

If you have privileges, you can alter the value directly using the **ulimit** command; however, it is better to update the *memlock* parameter in the /etc/security/limits.conf file. If the *memlock* limit is too low, Linux reports a `ENOMEM - "Not enough memory"` error, which does not make the cause obvious. The actual memory is allocated; it is the lock that fails.

For more information, see memlock in the *Caché Parameter File Reference*.

Note: You can achieve the same effect by using Linux Huge Pages for Caché shared memory. See the section "Support for Huge Memory Pages for Linux" in the section Calculating Memory Requirements in this chapter for more information.

Adjusting for Large Number of Concurrent Processes

Make the following adjustments if you are running a system that requires a large number of processes or telnet logins.

1. In the /etc/xinetd.d/telnet file, add the following line:

   ```
   instances = unlimited
   ```

2. In the /etc/xinetd.conf file, add or change the instances setting to:

   ```
   instances = unlimited
   ```

3. After you make these modifications, restart the **xinetd** services with:

   ```
   # service xinetd restart
   ```

4. The default pty (pseudo terminal connection) limit is `4096`. If this is not sufficient, add or change the maximum pty line in the /etc/sysctl.conf file. For example:

   ```
   kernel.pty.max=10000
   ```

Dirty Page Cleanup

On large memory systems (for example, 8GB or larger), when doing numerous flat-file writes (for example, Caché backups or file copies), you can improve performance by adjusting the following parameters, which are located in proc/sys/vm/:

- dirty_background_ratio — Maximum percentage of active that can be filled with dirty pages before pdflush begins to write them. InterSystems recommends setting this parameter to `5`.
- dirty_ratio — Maximum percentage of total memory that can be filled with dirty pages before processes are forced to write dirty buffers themselves during their time slice instead of being allowed to do more writes. InterSystems recommends setting this parameter to `10`

You can set these variables by adding the following to your /etc/sysctl.conf file:

```
vm.dirty_background_ratio=5
vm.dirty_ratio=10
```

These changes force the Linux pdflush daemon to write out dirty pages more often rather than queue large amounts of updates that can potentially flood the storage with a large burst of updates."

1.5.4.3 SUSE Linux Platform Notes

This topic includes the information on the following adjustments:

- Shared Memory Limits
- Locked-in Memory

Shared Memory Limits

The default shared memory limits (*shhmax* and *shmall*) on SUSE Linux 32-bit platforms are too small for Caché, and can be changed in the proc file system without a restart.

Caché uses shared memory for database buffers, global buffers, routine buffers, as well as license use. If the machine is being used only for Caché, InterSystems recommends setting the shared memory to approximately half the total memory. For more information, see the subsections of Determining Initial System Configuration in this chapter, and Determining License Capacity and Usage in the "Managing Caché Licensing" chapter of the *Caché System Administration Guide*.

Note: The recommendations to change the shared memory limits do not apply to SUSE Linux 64-bit systems.

For example, to allow 512 MB, type the following commands:

```
#sets shmall and shmmax shared memory
echo 536870912 >/proc/sys/kernel/shmall     #Sets shmall to 512 MB
echo 536870912 >/proc/sys/kernel/shmmax     #Sets shmmax to 512 MB
```

You can put these commands into a script that is run at startup. The SUSE Linux product documentation recommends you put the commands in the /etc/init.d/boot.local script file.

You can change the settings for the system memory user limits by modifying a file called /etc/profile.local. Add lines similar to the following:

```
#sets user limits (ulimit) for system memory resources
ulimit -v 512000     #set virtual (swap) memory to 512 MB
ulimit -m 512000     #set physical memory to 512 MB
```

In this same file, you can permanently change the values for the *PATH* and *CLASSPATH* parameters by adding lines similar to the following:

```
#sets env values PATH and CLASSPATH
export PATH=$PATH:/usr/cache/bin:/path/to/j2sdk/bin:/.
export CLASSPATH=
      $CLASSPATH:/cache/dev/java/lib/JDK16/cachedb.jar:/path/to/otherjar/file:/.
```

Important: To avoid the risk of losing your changes during system upgrades, do not change the /etc/profile file.

Locked-in Memory

On Linux platforms, you can configure Caché to lock the shared memory segment in memory to prevent paging as described in the memlock entry of the *Caché Parameter File Reference*. If shared memory is allocated in Huge Pages, they are automatically locked in memory and no further action is required. Otherwise, see the Locked-in Memory section of the *Red Hat Linux Platform Notes* in this appendix.

1.5.4.4 Ubuntu Platform Notes

This topic includes the information on the following adjustments:

- Semaphore Deletion Setting

Semaphore Deletion Setting

Under some circumstances, the OS may delete an instance's semaphores when the instance owner connects to an Ubuntu host, for example using SSH. To prevent this, edit the /etc/systemd/logind.conf file and change **RemoveIPC=yes** (the default) to **RemoveIPC=no**.

Updating to a newer version of Ubuntu may revert **RemoveIPC** to the default value of **yes**. After updating Ubuntu, be sure to change **RemoveIPC** to avoid unwanted semaphore deletion.

1.5.5 Special Considerations

The following sections describe particular issues or tasks associated with specific platforms or kinds of installations:

- Maximum User Process Recommendations
- Journal File System Recommendations
- IBM AIX® Considerations
- Red Hat Linux Considerations
- SUSE Linux Considerations
- macOS Considerations

1.5.5.1 Maximum User Process Recommendations

Ensure that the *maximum user processes* is set high enough to allow all Caché processes for a given user, as well as other default processes, to run on the system.

1.5.5.2 Journal File System Recommendations

To achieve optimal journal performance and ensure journal data integrity when there is a system crash, InterSystems recommends various file systems and mount options for journal files. For specific platform details see the UNIX® File System Recommendations section of the "Journaling" chapter of the *Caché Data Integrity Guide*.

1.5.5.3 IBM AIX® Considerations

The default settings of several AIX® parameters can adversely affect performance. For detailed information on the settings and recommendations, see the AIX® Platform Notes section of the chapter "Preparing to Install Caché".

System Requirements

For information about current system requirements, see the "Supported Technologies" chapter of the online *InterSystems Supported Platforms* document for this release.

Required C/C++ Runtime Libraries

You must ensure that the required C/C++ runtime is installed on your IBM AIX® system before installing Caché.

Caché for AIX is compiled using the IBM XL C/C++ for AIX 13.1 compiler. If the system on which you are installing Caché does not have the corresponding version of the runtime already installed, you must install these three runtime file sets from runtime package IBM_XL_CPP_RUNTIME_V13.1.0.0_AIX.tar.Z:

- xlC.aix61.rte 13.1
- xlC.rte 13.1
- xlC.msg.en_US.rte 13.1

If these files are not present, Caché installation will not complete.

Full information about and download of this package is available at IBM XL C/C++ Runtime for AIX 13.1.

Shared Library Environment Variable for Caché Engine Link Libraries

The Caché Engine link libraries contain a batch file that references any installed C linker.

If you have either the standard UNIX® C libraries or any proprietary C libraries defined in the *LIBPATH* environment variable, then your environment is ready.

If not, append the paths for the standard UNIX® C libraries to *LIBPATH*; these paths are /usr/lib and /lib.

Use of Raw Ethernet

In order to use raw Ethernet, an IBM AIX® machine must have the DLPI (Data Link Provider Interface) packages installed. If the machine does not have the DLPI packages, obtain them from your IBM provider and create DLPI devices through the following procedure:

1. Log in as `root`.
2. In the PSE drivers section of the /etc/pse.conf file, uncomment the four lines that refer to the DLPI drivers.
3. Save the file.
4. Restart the computer.

If the DLPI devices are not installed, the **EthernetAddress()** method of the %SYSTEM.INetInfo class returns a null string rather than information about the Ethernet device.

1.5.5.4 Red Hat Linux Considerations

The following considerations may apply to your environment:

- The default shared memory limit (*shmmax*) on Linux platforms is 32 MB, which is too small to install or run Caché. If the installation fails, you can change the value interactively in the proc file system (see the Red Hat Linux Platform Notes section of *Calculating System Parameters for UNIX®, Linux, and macOS* for more information), then reinstall Caché. The new memory limit remains in effect until you restart the Red Hat Linux system.

  Alternatively, you can change the value permanently by editing the /etc/sysctl.conf file, which requires a restart of the Red Hat Linux system for the new value to become effective.

- On Linux platforms with sufficient Huge Pages available, the Caché shared memory segment will be allocated from the Huge Page pool. A beneficial consequence of using Huge Pages is that the Caché shared memory segment will be locked into memory and its pages will not be paged out. See the section "Support for Huge Memory Pages for Linux" in the section Calculating Memory Requirements in this chapter for information about allocating Huge Pages.

- To use Kerberos on the Red Hat Linux platform, you must install the krb5-devel package in addition to the krb5-libs package. Installing krb5-devel establishes the required symbolic links for using Kerberos. The package is required for

production environments, not only development environments. See the Red Hat Network web site for more information about these components.

- Red Hat Enterprise Linux V4 requires Websphere MQ version 7.0 to use the MQ interface.

1.5.5.5 SUSE Linux Considerations

The following considerations may apply to your environment:

- The default shared memory limits (*shhmax* and *shmall*) on SUSE Linux 32-bit platforms are too small for Caché, and can be changed in the proc file system without a restart.
- On Linux platforms with sufficient Huge Pages available, the Caché shared memory segment will be allocated from the Huge Page pool. A beneficial consequence of using Huge Pages is that the Caché shared memory segment will be locked into memory and its pages will not be paged out. See the section "Support for Huge Memory Pages for Linux" in the section Calculating Memory Requirements in this chapter for information about allocating Huge Pages.
- To use Kerberos on the SUSE Linux platform, you must install the krb5-devel package in addition to the krb5-libs package. Installing krb5-devel establishes the required symbolic links for using Kerberos. The package is required for production environments, not only development environments. See the SUSE documentation web site for more information about these components.

See the SUSE Linux Platform Notes section of the chapter "Preparing to Install Caché" for detailed configuration information.

1.5.5.6 macOS Considerations

For the cinstall script procedure, see the section "Performing a Caché UNIX® Installation" in the chapter Installing Caché on UNIX®, Linux, and macOS in this book.

2

Installing Caché on Microsoft Windows

This chapter describes how to install Caché on a Microsoft Windows system. It assumes you are familiar with Windows directory structures, utilities, and commands. Before beginning this installation, be sure you have read all the information that applies to this platform in the chapter "Preparing to Install Caché".

This chapter contains the following major sections:

- Caché Installation
- Unattended Custom Installation
- Post-Installation Tasks
- Special Considerations

2.1 Caché Upgrade Installation

The steps for upgrading each type of a Caché installation are the same. The upgrade installation procedure installs the required components for the selected setup type on the computer.

Note: When upgrading Caché on a Windows system, the USER namespace and database will not be created if the original USER database was deleted in the original instance.

To upgrade an installation follow this procedure:

1. Stop any running Caché server on the computer. Also, close all other Windows applications and shut down the web server if it is installed on the same computer. Ensure that you have access to the installation kit.
2. Execute the installation file, for instance by double-clicking it in Windows Explorer or executing it on the command line as follows:

   ```
   C:\Users\Public\Downloads\cache-2016.2.0.626.0-win_x64.exe
   ```

3. The **Select Instance** dialog box lists the existing installation directories of all Caché instances installed on the machine. Select the instance you want to upgrade. (You can always select **New Instance** to install a new Caché instance, as described in Installing Caché.)
4. The **Welcome to the Caché Installation** dialog box displays the following buttons to let you control the upgrade:
   - Optionally, click **Customize** to add or remove components during the installation upgrade. For more information, see Caché Custom Installation.

Note: When you upgrade a previously-installed instance of Caché, the installer retains all configuration settings unless you customize them.

- Click **Update** to continue to the next dialog box whether or not you have customized the installation.

5. The **InstallShield Wizard Complete** dialog box indicates the installation has completed successfully. Click **Finish**.

2.2 Caché Installation

The steps for installing each type of Caché configuration are fundamentally the same, but diverge slightly depending on the type of installation. The differences are detailed in subsections after the standard installation description.

2.2.1 Installing Caché

The installation procedure installs the required components for the selected setup type on the computer. To perform an installation follow this procedure:

1. Ensure that the installation kit is available on your computer or on a network.
2. Execute the installation file, for instance by double-clicking it in Windows Explorer or executing it on the command line as follows:

   ```
   C:\Users\Public\Downloads\cache-2016.2.0.626.0-win_x64.exe
   ```

   Note: By default, a newly installed Caché instance starts immediately after installation and the Caché launcher (cube) is placed in the system tray. To prevent Caché from starting, set the Windows Installer property ISCSTARTCACHE to `0`; to prevent the launcher from being placed in the system tray, set ISCSTARTLAUNCHER to `0`. You can do this using the **setup.exe** from the multifile installation kit on the command line, for example:

   ```
   C:\Users\Public\Downloads\setup.exe ISCSTARTCACHE=0 ISCSTARTLAUNCHER=0
   ```

   For a description of all Caché Windows installer properties, see Unattended Custom Installation.

3. If there are existing Caché instances installed on the system, the **Select Instance** dialog box lists their installation directories. Select **New Instance** to install a new Caché instance. (You can also select an existing instance to upgrade that instance.)
4. If you are installing a new instance of Caché on this computer, setup displays the **License Agreement** dialog box. Click **I accept the terms in the license agreement** to confirm that you accept the license agreement.
5. The **Caché Instance Name** dialog box lets you assign a name to the new instance you are installing. The default name is CACHE (or if other instances, exist CACHE*n*, where *n* is the number of Caché instances including this new one). Accept the default or enter another name, using alphanumeric characters only. Subsequent updates to this instance maintain the instance name you enter here.
6. The **Destination Folder** dialog box lets you select a destination directory for the Caché software for the new instance; the default location is C:\InterSystems\Cache (or Cache*n* when multiple instances exist).

   You can select or create a directory by clicking **Change**. If the specified directory does not exist, setup lets you create it.

   Important: Be sure to see Installation Directory in the "Preparing to Install" chapter for important information about choosing an installation directory.

7. The **Setup Type** dialog box lets you specify how you intend to use Caché. Review the Setup Type section in Preparing to Install Caché to decide which components of Caché you need.

The next steps of the installation procedure differ based on the **Setup Type** you choose. To finish installing Caché, follow the steps in the section that corresponds to your chosen **Setup Type**:

- **Development** — Installing Caché Development or Server Components Only
- **Server** — Installing Caché Development or Server Components Only
- **Client** — Performing a Caché Client Installation
- **Web Server** — Installing the Web Server (CSP) Gateway Only
- **Custom** — Performing a Caché Custom Installation

2.2.2 Installing Caché Development or Server Components Only

If you wish, you can install only the components of Caché that are required on a development system or on a server system.

To perform a Development or Server installation:

1. Select **Development** or **Server** in the **Setup Type** dialog box, described in the Installing Caché procedure, and click **Next.**
2. The **Install Unicode Support** dialog box lets you select either 8-bit or Unicode character support for your installation (the default depends on your operating system locale).
3. The **Initial Security Settings** dialog box lets you decide how restrictive you want the initial Caché security settings to be. If you choose **Minimal**, the installation continues with the next step.

   Important: If you select **Minimal** for your initial security setting, but Caché requires network access to shared drives and printers, you must manually change the Windows user account under which to run the Caché service, choosing an existing account or creating a new account that has local administrator privileges on the server machine.

   If you select **Normal** or **Locked Down**, the installation includes the following steps:

   a. The **Enter Credentials for Caché Service** dialog lets you choose the credentials under which the Windows Caché service will run. The default is the local default SYSTEM account. You can also specify a defined (existing) Windows user account and password; when you do so, the installer verifies these credentials before proceeding. The installer also creates a Caché account with the same username, with the %All role for unrestricted access to the instance (the password for this account is provided on the next panel).

      Running the Windows Caché service under the default SYSTEM account is appropriate for many installations, but in some cases can cause issues relating to file permissions and network security access. If you anticipate potential problems in these areas for a Caché instance, for example due to your network configuration or security arrangements, specify an account for the Windows Caché service that has the needed privileges and/or access, for example that of a domain administrator.

      To change the service account (in Windows 10): Navigate to **Control Panel** > **Administrative Tools** > **Services**. Next, locate the correct Caché service. Then, right-click to open the properties dialog of the correct instance. Finally, change the credentials on the **Log On Tab** and restart Caché.

      Important: If you are using Kerberos, you must configure a service account before installing Caché; see Preparing the Security Environment for Kerberos in the "Preparing to Install Caché" chapter for more information.

Note: See Shared Memory Allocation on Windows for information about granting the Windows "Lock Pages in Memory" (SELockMemory) privilege when running Caché under credentials other than the local system account.

b. The first panel of the **Caché Users configuration** dialog let you enter the initial password for the following predefined Caché user accounts _SYSTEM, Admin, and SuperUser, as well as the username you specified if you selected a defined Windows account as the service account on the previous panel. The second panel of the **Caché Users configuration** dialog lets you enter the initial password for the CSPSystem predefined account. For more details on these predefined users, see the Predefined User Accounts section of the "Users" chapter of the *Caché Security Administration Guide*.

For a detailed explanation of these security settings, see Managing Windows User Access to the Caché Instance below, and the Initial Caché Security Settings section of the "Preparing to Install Caché" chapter of this guide.

4. The **Ready to Install** dialog box lets you review the installation name, type, and directory, as well as the license key status.

   Click the **License** button to select a Caché license key. If the key is valid, the license is automatically activated, the **License Key** field on the **Ready to Install** dialog is updated to **Valid**, and the license key is copied to the instance's manager directory (*install-dir*/mgr) as cache.key during installation; no further activation procedure is required. If you do not select a key, you can activate your InterSystems Caché license key following installation. See Activating a License Key in the "Managing Caché Licensing" chapter of the *Caché System Administration Guide* for information about licenses, license keys and activation.

   Click **Install** to continue. Setup installs Caché in the selected directory.

5. The **InstallShield Wizard Complete** dialog box indicates the installation has completed successfully. Choose whether you want to see the *Getting Started* page and click **Finish**.

The system starts automatically after installation is complete. The **Caché Cube** icon appears in the system tray area of the Windows tool bar. Click the cube to bring up the Caché menu. In addition, there is a **Caché** item on the Windows **Programs** menu.

2.2.3 Installing Caché Client Components Only

If you wish, you can install only those parts of Caché that are required on a client machine.

To perform a client installation:

1. Select **Client** in the **Setup Type** dialog box, described in the Installing Caché procedure, and click **Next**.
2. The **Ready to Install** dialog box lets you review the installation name, type, and destination directory for the software files.

   Click **Install** to continue. Setup installs Caché in the selected directory.

3. The **InstallShield Wizard Complete** dialog box indicates the installation has completed successfully. Click **Finish**.

After Caché is installed on a client, the **Caché Cube** icon appears in the system tray area of the Windows tool bar; it appears dimmed because there is no Caché server running.

Important: Before you can use the client, you must specify the preferred server for this client; this procedure is described in the Define a Remote Server Connection section of the "Connecting to Remote Servers" chapter of the *Caché System Administration Guide*.

2.2.4 Installing the Web Server (CSP) Gateway Only

If you wish, you can install only those parts of Caché that are required on a CSP Gateway machine.

To perform a CSP Gateway installation:

1. Select **Web Server** in the **Setup Type** dialog box, described in the Installing Caché procedure, and click **Next**.
2. The **Ready to Install** dialog box lets you review the installation name, type, and destination directory for the software files.

   Click **Install** to continue. Setup installs Caché in the selected directory.
3. The **InstallShield Wizard Complete** dialog box indicates the installation has completed successfully. Click **Finish**.

If a web server is running, a dialog box asks if you want to restart the web server. If you click **Yes**, the installation procedure restarts the web server. If you click **No**, the procedure does not restart the web server, in which case it does not start until you restart it manually or restart the system.

If the installer detects an Internet Information Services (IIS) web server installed on the system, it configures the web server for the CSP gateway. The installer also displays a checkbox for IIS; if you select this, CSP gateway IIS modules are installed in C:\InetPub\CSPGateway.

The CSP Gateway configures the following application paths pointing to the server configured for the instance:

- /
- /csp
- /Cache (instance name)

You can change the configurations manually after installation from the CSP Gateway application; for information, see the *CSP Gateway Configuration Guide*.

Note: The installer cannot automatically configure an Apache web server for use with Caché and CSP; for information on the required manual configuration procedures, see the *CSP Gateway Configuration Guide*.

2.2.5 Performing a Custom Caché Installation

The Caché installation program allows you to select certain Caché components to install on the computer. For example, you may want to install only the Web Server (CSP) Gateway. Keep in mind that some selections require that you also install other components.

To perform a custom Caché installation:

1. Select **Custom** in the **Setup Type** dialog box, described in the Installing Caché procedure, and click **Next**.
2. In the **Custom Setup** dialog box, select the components you want to install as described in the Components Installed by Setup Type table.

   Important: If you are custom-installing the Caché Database Engine (Caché Server) component group or any of its components, ActiveX Connectivity (included in the Caché Application Development component group) is a prerequisite.

   If you are custom-installing the Documentation component group or any of its components, the Caché Database Engine (Caché Server) component group is a prerequisite.

   Note: You can remove previously-installed components by selecting the **X** menu item for any component group or component.

3. Optionally, click **Space** to ensure that there is enough space on the disk for the selected components.
4. The **Install Unicode Support** dialog box lets you select either 8-bit or Unicode character support for your installation (the default depends on your operating system locale).

   Important: If you are installing Ensemble, you must select **Unicode**.

5. The **Enter Port Numbers** lets you change the port numbers assigned by Caché.

   Note: You cannot enter a port number greater than `65535` or less than `1`. For information about setting port numbers, see Set Port Numbers in the "Using Multiple Instances of Caché" chapter of the *Caché System Administration Guide*.

   The following port numbers are valid for your Caché instance:

   - SuperServer Port Number — `1972` or the first available subsequent number equal to or higher than `56773`; you can change the superserver port value after installation from the **Memory and Startup** page (**System** > **Configuration** > **Memory and Startup**).
   - WebServerPort number — `57772` or the first available subsequent number; you can change the WebServerPort values after installation from the **Startup Settings** page (**System Administration** > **Configuration** > **Startup Settings**).

6. The **Initial Security Settings** dialog box lets you decide how restrictive you want the initial Caché security settings to be. If you choose **Minimal**, the installation continues with the next step.

   Important: If you select **Minimal** for your initial security setting, but Caché requires network access to shared drives and printers, you must manually change the Windows user account under which to run the Caché service, choosing an existing account or creating a new account that has local administrator privileges on the server machine.

   If you select **Normal** or **Locked Down**, it displays **Enter Credentials for Caché Service**, where you can choose to run Caché service under:

   - The default SYSTEM account, which runs Caché as the Windows Local System account.
   - Defined Windows user account. The installation creates a Caché account with the %All role for the user who is installing Caché to grant that user access to services necessary to administer Caché. Enter and confirm the password for this account. The password must meet the criteria corresponding to the security setting.

     Important: If you are using Kerberos, you must enter a defined account that you have set up to run the Caché service. InterSystems recommends that you use a separate account specifically set up for this purpose.

   When you click **Next**, the installation verifies the following if you enter a defined user account:

   - The account exists on the domain.
   - You have supplied the correct password.

   For a detailed explanation of these settings, see the Initial Caché Security Settings section of the "Preparing for Caché Security" section of this book.

7. The **Ready to Install** dialog box lets you review the installation name, type, and destination directory for the software files.

   You can also click the **License** button to select a Caché license key. If the key is valid, the license is automatically activated and the license key is copied to the instance's manager directory (*install-dir*/mgr) as cache.key during installation, and no further activation procedure is required. If you do not select a key, you can activate your InterSystems

Caché license key following installation. See Activating a License Key in the "Managing Caché Licensing" chapter of the *Caché System Administration Guide* for information about licenses, license keys and activation.

Click **Install** to continue. Setup installs Caché in the selected directory.

8. The **InstallShield Wizard Complete** dialog box indicates the installation has completed successfully. Choose whether you want to see the *Getting Started* page and click **Finish**.

2.3 Unattended Custom Installation

The Caché for Windows installer provides a way to perform *unattended* custom installation, upgrade, reinstallation (repair), and removal (uninstall) of instances of Caché on your computer. A typical install operation obtains necessary input from the user in the form of responses to dialog boxes. An unattended operation, however, does not prompt the user for input; instead, it gets input from properties passed to the Caché installation file on the command line. These properties are described in the Command-Line Reference section.

This section discusses the following topics:

- Running an Unattended Installation
- Running an Unattended Upgrade or Reinstall
- Running an Unattended Uninstallation
- Command-Line Reference

Note: No messages are displayed during unattended installation, upgrade, reinstallation, or uninstallation.

2.3.1 Running an Unattended Installation

To launch unattended installation of a new instance of Caché, use the following command:

```
<path>\<installer>.exe /instance <instancename> /q{b|n} <properties>
```

where:

- <*path*> is the path to the Caché installation file.
- <*installer*>.exe is the name of the Caché installation file.
- **/qb** displays a progress bar during unattended installation, while **/qn** specifies a completely silent installation (one or the other must be included).
- <*instancename*> is the name of the installed Caché instance. (A new instance is installed with the default instance name Cache if the **/instance** parameter is omitted, however the **/instance** parameter must be used when there are one or more instances already installed.)
- <*properties*> are the properties you are passing to the installer (see the Command-Line Properties table).

For example, to install an instance of Caché with the default instance name in an installation directory named C:\InterSystems\MyCache on a 64–bit Windows system, specify the following:

```
C:\downloads\cache-2015.1.0.320.0-win_x64.exe /qn INSTALLDIR=C:\InterSystems\MyCache
```

To install an instance of Caché with the instance name CacheA, specify the following:

```
C:\downloads\cache-2015.1.0.320.0-win_x64.exe /instance CacheA /qn
```

Alternatively, you can custom install a subset of features using the **ADDLOCAL** property; see the Command-Line Properties table, as well as the Custom-Installable Features table for a list of features that can be specified. For example, to install only the Launcher and the PDF version of documentation in an installation that uses the default instance name and default installation directory on a 64–bit Windows system, specify the following:

```
C:\downloads\cache-2015.1.0.320.0-win_x64.exe /qn ADDLOCAL=cube,documentation,documentation_pdf
```

Note: An unattended installation does not install the CSP Gateway by default; this must be specified using the **ADDLOCAL** property (see the Command-Line Properties table).

2.3.2 Running an Unattended Upgrade or Reinstall

The target of the Caché installer can be an existing installed instance as well as a new instance. When there are no installed Caché instances, the target is always a new instance. When one or more instances of Caché are already installed, you must use the **/instance** flag to specify either the name of one of the existing instances or a new name, which installs a new instance.

When an installed instance is the installer target, the action the installer takes depends on the versions of the installation file specified in the command line and the target instance, as follows:

- If the installation file is the same version as the target installed instance, the installer reinstalls (repairs) the instance.
- If the installation file is a later version than the target installed instance, the installer upgrades the instance to the new version.

For example, to run an unattended upgrade of an installed instance CacheB that is an earlier version than the installation file, use the following:

```
C:\downloads\cache-2015.1.0.320.0-win_x64.exe /instance CacheB /qn
```

You can reinstall one or more specific features, as listed in the Custom-Installable Features table, by specifying the target instance and using the **REINSTALL** property (see the Command-Line Properties table). For example, to reinstall Studio for the installed instance CacheB, you can use the following command (assuming the installation file and CacheB are the same version):

```
C:\downloads\cache-2015.1.0.320.0-win_x64.exe /instance CacheB /qn REINSTALL=studio
```

2.3.3 Running an Unattended Uninstallation

To launch an unattended uninstall, specify the instance to be uninstalled and the **REMOVE=ALL** property, as follows:

```
C:\downloads\cache-2015.1.0.320.0-win_x64.exe /instance CacheC /qn REMOVE=ALL
```

You can also use the **REMOVE** property to remove specific features, as described in the Custom-Installable Features table. For example, to remove the Apache 2.0 CSP Gateway from instance CacheC, use the following command:

```
C:\downloads\cache-2015.1.0.320.0-win_x64.exe /instance CacheC REMOVE=cspgateway,cspapache20
```

2.3.3.1 Special Consideration

If you do not have access to the original installation package, you can run uninstall in unattended mode by using the Windows® Installer command-line application (**msiexec**) and information in the Registry, as follows:

```
msiexec /x {<product_guid>} /qn /l <logfile>
```

where <*product_guid*> is the **ProductCode** property value of the version you installed.

You can obtain the **ProductCode** property value from the Registry:

- On 32-bit machines, go to the following Registry location:
  `HKEY_LOCAL_MACHINE\SOFTWARE\InterSystems\Cache\Configurations\<instance>`
- On 64-bit machines, go to the following Registry location:
  `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\InterSystems\Cache\Configurations\<instance>`

where <*instance*> is the name of the Caché instance you want to uninstall in unattended mode. The **ProductCode** property value is displayed in a row similar to the following:

```
ProductCode     REG_SZ   {80E3F658-2D74-4A81-92AD-FD16CD226154}
```

You can also use any of the properties in the Command-Line Properties table with **msiexec**. For information about **msiexec**, see the Microsoft msiexec (command-line options) TechNet article (http://technet.microsoft.com/en-us/library/cc759262.aspx).

2.3.4 Command-Line Reference

The Command-Line Properties table describes the Caché-specific unattended install properties that you can modify via the command-line interface. The property name must be uppercase, but the arguments are not case-sensitive; each property must be separated by one or more spaces, and properties can be specified in any order as *`PROPERTYNAME=argument`*. For example:

```
... ISCSTARTCACHE=0 UNICODE=0 WEBSERVERPORT=57779 INITIALSECURITY=Normal
```

Note: In the following table, only the **REINSTALL** and **REMOVE** properties are used with installed instances, as described in Running an Unattended Upgrade or Reinstall and Running an Unattend Uninstall, respectively. All of the other properties are used only when installing new instances.

Table 2–1: Command-Line Properties

Property Name	Description
ADDLOCAL	Use this property to custom install a new instance of Caché with a subset of features or to omit optional databases (see the Custom-installable Features table) by specifying a comma-separated list of *featurenames* together with their group names, as described in the example following this table. **Note:** In the absence of the **ADDLOCAL** `property, or if ADDLOCAL=ALL` is specified, all features are installed. See also the **REINSTALL** property, for use with installed instances.

Property Name	Description
CACHESERVICEDOMAIN	Required if the service credentials are defined as `UserDefined`; see the **SERVICECREDENTIALS** property in this table. Use this property to specify the domain of the Windows Caché service login account specified by **CACHESERVICEUSER**. **Note:** If the service credentials are specified as `LocalSystem`, do not use this property.
CACHESERVICEPASSWORD	Required if the service credentials are defined as `UserDefined`; see the **SERVICECREDENTIALS** property in this table. Use this property to specify the password for the Windows Caché service account specified by **CACHESERVICEUSER**. **Note:** If the service credentials are specified as `LocalSystem`, do not use this property.
CACHESERVICEUSER	Required if the service credentials are defined as `UserDefined`; see the **SERVICECREDENTIALS** property in this table. Use this property to specify the username of the account under which to run the Windows Caché service. **Note:** If the service credentials are specified as `LocalSystem`, do not use this property.
CACHEUSERPASSWORD	Required if the security level is `Normal`or `Locked Down`; see the **INITIALSECURITY** property in this table. Use this property to specify the password for the predefined Caché accounts _SYSTEM, Admin, and SuperUser, as well as the account with the username specified by **CACHESERVICEUSER** if **SERVICECREDENTIALS** is specified as `UserDefined`. **Note:** If the initial security level is `None`, do not use this property.
CSPSKIPIISCONFIG	Optionally use this property to install IIS CSP binary files. With a value of 1, the files will be installed without any changes to the IIS web server configuration. With a value of 0, the installer will update the IIS web server configuration regardless of the existence of the /csp virtual directory.
CSPSKIPAPACHE20CONFIG	Optionally use this property to install Apache 2.0 CSP binary files. With a value of 1, the files will be installed without any changes to the IIS web server configuration. With a value of 0, the installer will update the IIS web server configuration regardless of the existence of the /csp virtual directory.
CSPSKIPAPACHE22CONFIG	Optionally use this property to install Apache 2.2 CSP binary files. With a value of 1, the files will be installed without any changes to the IIS web server configuration. With a value of 0, the installer will update the IIS web server configuration regardless of the existence of the /csp virtual directory.

Property Name	Description
CSPSYSTEMUSERPASSWORD	If the security level is `Normal`or `Locked Down` (see the **INITIALSECURITY** property in this table), optionally use this property to specify a password for the CSPSystem predefined user. If this property is omitted, the value of **CACHEUSERPASSWORD** is used. **Note:** If the initial security level is `None`, do not use this property.
INITIALSECURITY	Optionally use this property to specify the level of security to be used by the instance being installed. Specify `None`, `Normal`, or `\"Locked Down\"` (the escaped quote characters are needed due to the space character). **Note:** Omit this property to accept the default of `None`. See also the **CACHEUSERPASSWORD**, **CSPSYSTEMUSERPASSWORD**, and **SERVICECREDENTIALS** properties in this table.
INSTALLDIR	Optionally use this property to specify the directory in which the instance is to be installed. **Note:** If the property is omitted, the default installation directory is C:\InterSystems\CACHE*n*, where *n* is `{empty}`, `1`, `2`, ... `127`.
INSTALLERMANIFEST	If installing with an installation manifest, as described in Using the Manifest in the chapter "Creating and Using an Installation Manifest", you must use this property to specify the location of the installation manifest (that is, your exported manifest class) .
INSTALLERMANIFESTLOGLEVEL	If installing with an installation manifest, as described in Using the Manifest in the chapter "Creating and Using an Installation Manifest", optionally use this property to specify the log level of the **setup()** method of your installation manifest class. The default log level is 1.

Property Name	Description

INSTALLERMANIFESTPARAMS	If installing with an installation manifest, as described in Using the Manifest in the chapter "Creating and Using an Installation Manifest", use this property to specify the name/value pairs (*name*=*value*) to be passed to the first argument of the **setup()** method of your installation manifest clas. This property can be used to modify the Caché parameter file (cache.cpf) and activate the changes before your manifest runs. The following parameters can be specified:
	• bbsiz
	• globals4kb, globals8kb, globals16kb, globals32kb, globals64kb
	• gmheap
	• LibPath,
	• locksiz,
	• MaxServerConn,
	• Path
	• routines
	• ZFSize, ZFString
	For example:
	`INSTALLERMANIFESTPARAMS="bbsiz=512000,globals4kb=20, globals8kb=30,globals16kb=40,globals32kb=50, globals64kb=100,routines=40,gmheap=10000, LibPath=c:\libpath\,locksiz=2179648,MaxServerConn=5, Path=c:\lib\,ZFSize=2000,ZFString=3000"`
	The following would be useful in installing and activating 100 MB of 64KB buffers before running a manifest that creates a 64kb block size database:
	`INSTALLERMANIFESTPARAMS="globals64kb=100"`

ISCSTARTCACHE	Optionally set this property to `0` to prevent Caché from starting after installation. The default is `1`, to start Caché.
ISCSTARTLAUNCHER	Optionally set this property to `0` to prevent the Caché launcher (cube) from being added to the system tray. The default is `1`, to add the launcher.

Property Name	Description
REINSTALL	Use this property to reinstall (repair) an installed instance of Caché or to change the custom-installed features (see the Custom-installable Features table) for an installed instance of Caché: • To reinstall whatever features are currently installed for the instance—whether that is a custom-installed subset of features or all features—specify `ALL`. • To reinstall a subset of Caché features that is different from the subset of features currently installed, specify a comma-separated list of *featurenames* together with their group names (as described in the example following this table). See also the **ADDLOCAL** property (for use with new instances) and **REMOVE** property (for uninstalling installed instances).
REMOVE	Use this property to uninstall (remove) an instance of Caché or a subset of custom-installed features (see the Custom-installable Features table) installed for an installed instance of Caché: • To remove an instance of Caché, specify `ALL`. • To remove a subset of Caché features (see the Custom-installable Features table), specify a comma-separated list of *featurename* together with their group names (as described in the example following this table). See also the **ADDLOCAL** (for new instances) and **REMOVE** properties in this table properties in this table.
SERVICECREDENTIALS	If the security level is `Normal`or `Locked Down` (see the **INITIALSECURITY** property in this table), optionally use this property to specify the credentials under which the Windows Caché service will run: `LocalSystem` for the default local system account or `UserDefined` (an existing Windows user account). If you do not specify the property, the default of `LocalSystem` is used. **Note:** If the initial security level is `None`, do not use this property. See Managing Windows User Access to the Caché Instance for important information about the Caché service account. If you specify `UserDefined` for this property, you must also specify the **CACHESERVICEDOMAIN**, **CACHESERVICEPASSWORD**, and **CACHESERVICEUSER** properties.
SUPERSERVERPORT	Optionally use this property to specify the Superserver port to be used by the instance being installed. **Note:** By default, this port is auto-determined, beginning with `1972` (if available), then `56773`, and increasing by 1 for each installed instance of Caché.

Property Name	Description
UNICODE	Optionally use this property to specify whether 8–bit or 16–bit Unicode characters are to be supported by the instance being installed. For 8–bit characters, specify `0`; for 16–bit characters, specify `1`. **Note:** This setting applies to Caché instances only; Ensemble instances are always Unicode. If you omit this property, is 8–bit specified by default for all languages except Chinese, Korean and Japanese; 16–bit is specified by default for Chinese, Korean, and Japanese systems.
WEBSERVERPORT	Optionally use this property to specify the Webserver port to be used by the instance being installed. **Note:** By default, this port is auto-determined, beginning with `57772` and increasing by 1 for each installed instance of Caché.

The Custom-Installable Features table lists component group/component names and the associated *featurename* for each. You can specify "ALL" (to specify all available features) or a comma-separated list (with no spaces) of feature names (to specify individual features).

To specify components in ADDLOCAL, REINSTALL, and REMOVE properties (see the Command-Line Properties table), specify the *featurename* of a component group, followed by the *featurename* of each specific component from that group that you want installed. For example, to install only the SAMPLES and USER databases and the PDF documentation, include the following in the command line:

```
ADDLOCAL=server,server_samples,server_user,documentation,documentation_pdf
```

When specifying a component group, you must also specify at least one associated component; if no components are listed with a component group, the component group is ignored and no components are installed. For example, if you specify the following:

```
ADDLOCAL=documentation,documentation_pdf,server,development,perlbind,pythonbind
```

the **server** component group is ignored and no server components are installed. (This requirement does not apply to the **studio** and **cube** groups as they have no components.)

Note: You can choose not to install any of the three optional databases, ISC_PACKAGE_DOCBOOK_DATABASE, ISC_PACKAGE_SAMPLES_DATABASE, and ISC_PACKAGE_USER_DATABASE, by changing the value to `N` (Y by default). Doing so will ensure these database/namespace definitions are not installed and will not be created in cache.rpf.

Table 2–2: Custom-Installable Features

Component Group (featurename)	Components (featurename)
Server (**server**)	Samples database (**server_samples**) User database (**server_user**) SQL Gateway (**sqlgateway**) Apache Formatting Objects Processor (**fop**) Server monitoring tools (**server_monitoring**) HotJVM RenderServer/QueuingRenderServer for Zen Reports (**renderserver**) Agent Service (**agent_service**) **Note:** For Ensemble and HealthShare only: ENSDemo database (**server_ensdemo**)
Launcher (**cube**)	
Studio (**studio**)	
xDBC (**sqltools**)	ODBC (**odbc**) JDBC (**jdbc**)
Development (**development**)	C++ Binding (**cppbind**) Light C++ Binding (**lcbind**) C++ SDK (**cppsdk**) Callin (**callin**) Callin, Threaded (**callin_threaded**) Perl Binding (**perlbind**) Python Binding (**pythonbind**) Java (**javabind**) Threaded Server Libraries (**server_threaded**) ActiveX Connectivity (**activex**) Other Samples (**other_samples**)
Documentation (**documentation**)	PDF Documentation (**documentation_pdf**) Online Documentation (**documentation_online**)
CSP Gateway (**cspgateway**)	IIS (cspiis) Apache 2.0 (**cspapache20**) Apache 2.2 (**cspapache22**)

2.4 Post-Installation Tasks

Review the following important post-installation tasks:

- You can manage your Caché instance using the Management Portal, which is accessible from the Caché Cube. For more information on this management tool, see the "Using the Management Portal" chapter of the *Caché System Administration Guide*.
- Caché installation on Windows names the Windows Caché service with the instance name, using the name from the **Define Caché Instance Name** dialog box, and configures the service to start automatically when you start your server; this means the Caché instance is automatically configured to autostart (start when the system starts).

  You can configure the instance not to autostart by changing the **Start Caché on System Boot** setting on the Memory and Startup page of the Management Portal (from the home page, select **System Administration** > **Configuration** > **System Configuration** > **Memory and Startup**). If this instance is part of a Windows cluster, you *must* clear this check box to prevent automatic startup, allowing the cluster manager to start Caché.

  Note: You may occasionally need to *prevent* IRIS from starting with the host system starts while the host system is down (for example during restores). This can only be done at the OS level and is dependent on your configuration.

- If you plan to connect remotely to other instances of Caché, follow the procedure described in the Define a Remote Server Connection section of the "Connecting to Remote Servers" chapter of the *Caché System Administration Guide*.
- If you are upgrading from a previous version of Caché, see Post-installation Upgrade Tasks in the "Upgrading Caché" chapter of this book.
- If you are using the Windows IIS Web server, file types must be mapped manually; for information, see the Web Servers for Microsoft Windows chapter of the *CSP Gateway Configuration Guide*.
- If appropriate for your installation, perform any additional tasks described in the Special Considerations section.

2.5 Special Considerations

The following topics describe particular issues or tasks associated with licensing, specific platforms, or kinds of installations:

- Managing Windows User Access to the Caché Instance and Installation Tree
- Using an Installation Manifest
- Multiple Caché Installation Issues
- Changing the Caché Language
- Reinstalling or Uninstalling Caché
- InterSystems Caché Packet Drivers
- Shared Memory Allocation on Windows

2.5.1 Managing Windows User Access to the Caché Instance

For Normal and Locked Down installations, two local user groups control access to the Caché instance. These groups are:

- **CacheServices**, which grants the privileges to start, stop, and control the Caché instance.

- **Cache_Instance_**<*instancename*>, which grants access to the installation tree—the directory in which Caché is installed and all its subdirectories.

Note: Regarding these groups and their privileges:

- When the **CacheServices** group is created, it is granted the **`Replace a process level token`** and **`Adjust memory quotas for a process`** privileges. Do not remove these privileges.
- The **CacheServices** and **Cache_Instance**_*instancename* groups may not grant all the permissions that Caché requires to perform certain actions. To ensure Caché has the needed access to all instance, journal, and log files that are outside the installation tree, grant the **Cache_Instance**_*instancename* group full access to these files and the directories containing them. You may also grant this group additional permissions if necessary.

When you specify a service account other than the default local system account, Caché creates these two groups and adds the service account to each. See the Changing the Caché Service Account section below for details.

By default, any authenticated Windows users can access the installation tree, which may be undesirable. The following command removes the Windows access control entry (ACE) for authenticated users:

```
icacls <install-dir> /remove "NT AUTHORITY\Authenticated Users"
```

After running this command, only users that are administrators or in the **Cache_Instance_**<*instancename*> group have access to the installation tree.

Important: If you do not do this, any user who can log in to the host Windows system can easily modify files, change settings, or disable the Caché instance entirely.

In some cases, you may want to give a Windows account other than the service account access to the installation tree. This could, for example, include accounts running automated tasks, or accounts that log in to the Windows server directly and then access Caché through a local Terminal session, or by invoking a custom callin executable. You can give any such account the needed access by adding it to the **Cache_Instance_**<*instancename*> group.

2.5.1.1 Changing the Caché Service Account

Enter the following in the command line to change the InterSystems service account:

```
<install-dir>\bin\cinstall.exe setserviceusername <instance-name> <username> <password>
```

This command changes the service account to the user you provide. It also adds the user to the **CacheServices** and **Cache_Instance_**<*instancename*> groups, creating these groups if necessary. After running this command and restarting a Caché instance, the instance runs under the new service account.

2.5.2 Using an Installation Manifest

You can define an installation manifest that describes a specific Caché configuration and invoke it when executing the installation file. For more information on installation manifests, see the chapter "Creating and Using an Installation Manifest".

2.5.3 Multiple Caché Installation Issues

You can install and simultaneously run multiple instances (Caché 4.0 and later) on a single Windows machine. Install Caché as for a single installation, giving each instance a unique name, a unique installation directory, and a unique port number.

Please refer to the Multiple Caché Instances section of the *Caché System Administration Guide* for more detailed information.

Installing multiple Caché instances is limited by components where only one exists on a system. For example, typically there is only one web server on a system; and as such, the Caché installation configures CSP for the most recent installation.

Caché client components stored in the registry encounter the same issue. Caché stores its ODBC driver and ActiveX components in the registry using one name for each. Currently, the last installation updates these components to point to the last instance installed. If you are adding this release of Caché to your machine and keeping older (Caché 4.1 and earlier) versions running, you need to register the latest components. See Registering Files for details.

InterSystems makes an effort to move common components to a common directory that can be shared across Caché instances. Unfortunately, because of backward compatibility issues, not all current Caché components support Caché 5.0 and 4.1 instances and are even less likely to support Caché 4.0 instances on the same machine.

As a work-around, you can take advantage of a feature Microsoft introduced with Windows 2000 and later. You can force your executable to ignore the registry paths to an executable by creating an empty file of the same name with .local appended to the executable name.

For example: CStudio.exe would need an empty file called CStudio.exe.local to force the Studio program to look in the current directory, before using the registry path. By creating these empty .local files, you enable a previous Caché instance to use the compatible local files, rather than a newly installed current Caché set of registered executables.

To create .local files for all the executables in a directory type the following at a DOS prompt:

```
for %c in (*.exe) do set tempvariable= >%c.local
```

For more information on .local files, see the MSDN library article *Dynamic-Link Library Redirection*.

2.5.3.1 Registering Files

The Caché installation contains a Regfiles.bat script file in the *install-dir*/Bin directory that reregisters object tool files in a common Caché directory. You require Administrator privileges to run RegFiles.bat.

You need to run this script only if you install or uninstall any instance of Caché 4.1 or earlier; if you are running only instances of Caché 5.0 or later, the object tools function properly with multiple instances.

Here is an example of running the script:

```
C:\MyCache\Bin>regfiles.bat

C:\MyCache\Bin>rem Register Cache Self-Registering executables in
 common directory
.
.
```

If after running RegFiles.bat, you still receive errors similar to the following when you start Studio:

```
Cannot create class factory for COM_SLSID_TNodes
```

Run the script with the ALL argument; this reregisters all JCOM DLLs in addition to the other files.

For example:

```
C:\MyCache\Bin>RegFiles.bat ALL

C:\MyCache\Bin>rem Register Cache Self-Registering executables in
common directory
.
.
C:\MyCache\Bin>rem Register JCOM dlls

C:\MyCache\Bin>set CacheDir=C:\MyCache\bin\
.
.
```

Important: Running **RegFiles.bat ALL** when it is not necessary will prevent the uninstall procedure from removing some registry keys under HKLM\Software\InterSystems when you uninstall the last Caché instance.

2.5.4 Changing the Caché Language

When you install Caché, all supported language-specific utility DLLs are installed in the *install-dir*\Bin directory. Each DLL contains localized strings and messages.

The format of the name of the DLL is UTILaaa.DLL, where *aaa* is a 3-letter code that signifies the following languages:

Code	Language
CHS	Chinese (Simplified)
DEU	German (Standard)
ENU	English (United States)
ESP	Spanish (Spain)
FRA	French
ITA	Italian (Standard)
JPN	Japanese
KOR	Korean
NLD	Dutch (Standard)
PTB	Portuguese (Brazilian)
RUS	Russian

For information about changing the locale of a Caché installation, see Using the NLS Settings Page of the Management Portal in the "Configuring Caché" chapter of the *Caché System Administration Guide.*

Note: You can change only among 8-bit locales or Unicode locales, not from 8-bit to Unicode or vice versa. See the %SYS.NLS entry in the *InterSystems Class Reference* for more information.

2.5.5 Reinstalling or Uninstalling Caché

By running setup and selecting a Caché instance of the same version as the installer, or by selecting **Programs and Features** from Windows Control Panel and selecting a Caché instance, you can make changes to or uninstall the instance.

When you run setup as described in Caché Installation and select a Caché instance of the same version as the installer in the **Select Instance** box, or select an instance in **Programs and Features** and use the **Change** or **Repair** buttons, the **Updating Instance** *instancename* dialog box displays.

Note: When you select the **Uninstall** button in **Programs and Features**, the uninstall operation begins immediately.

Click **Next** to display the **Modify, repair or remove the program** dialog box, then select the appropriate option on this dialog to change, repair, or uninstall the instance.

- Select **Modify** to display the **Custom Setup** dialog box described in Performing a Caché Custom Installation. Using this dialog box, you can select the component groups or components you want to add or remove. Components are described in the Components Installed by Setup Type table.
- Select **Repair** to repair problems with the instance such as missing or corrupt files or registry entries.
- Select **Remove** to uninstall the instance.

Important: Use only the Caché installer or Windows Control Panel **Programs and Features** to uninstall Caché. Other uninstall programs are not supported and using them may cause unexpected results.

2.5.6 InterSystems Caché Packet Drivers

Install Caché Packet Driver for Windows 2000, XP, and 2003 Server

To use Raw Ethernet with Caché on Windows systems, you must install the appropriate packet driver as described below. First ensure that the appropriate driver file is available on your computer or on a network.

For Windows 2000, XP, and 2003 Server systems, install the InterSystems Packet Protocol Driver as follows:

1. Right-click **My Network Places** on the desktop and click **Properties**.

   You can also click **Start**, point to **Settings** and click **Network and Dial-up Connections**.
2. Right-click **Local Area Connection** and click **Properties**.
3. Click **Install**.
4. Click **Protocol** in the **Select Network Component Type** dialog box and then click **Add**.
5. Click **Have Disk** in the **Select Network Protocol** dialog box.
6. Enter the path to the packet driver kit and click **OK**.

   You can also enter the appropriate drive letter and click **Browse** to search for the correct path, \drivers\win2k, that contains the file ispkt2k.inf. Click **Open** and then click **OK**.
7. Select the appropriate driver for your operating system, **InterSystems Packet Driver for Windows 2000, XP and 2003 Server**, for example, and click **OK**.
8. After the driver is installed, click **Close**.

After you restart Windows, Caché is fully available to you.

2.5.7 Shared Memory Allocation on Windows

InterSystems recommends enabling Caché to allocate its memory as large pages on Windows systems, which provides more efficient use of memory and paging space. To do so, grant the Windows "Lock Pages in Memory" (SELockMemory) privilege to the user account used to run Caché, which allows Caché to request its memory as large pages. For finer grain control over the memlock parameters, see memlock in the *Caché Parameter File Reference*.

Note: If Caché is running under the default SYSTEM account, it has the "Lock Pages in Memory" privilege by default.

When you restart Caché while using large pages, you typically also need to restart Windows, to guarantee that all configured memory is allocated. If startup is unable to allocate the full amount of configured memory, it attempts to start up with less memory, and may or may not use large pages. You can check the actual memory allocated by reviewing the most recent Caché/Ensemble startup in the console log.

Specifically, Caché allocates shared memory at Windows startup as follows:

1. Request large pages, if allowed.
2. If unable to allocate the full amount of configured memory in large pages, or if large pages are not allowed, request small pages.
3. If unable to allocate the full amount of configured memory in small pages, reduce the allocation by one eighth (1/8) and begin again with step 1.

Note: On Windows versions prior to Windows Server 2008 and Windows Vista, which are not supported by this version Caché, allocation of small pages bigger than 2 GB is not allowed. The Caché memory allocation algorithm was therefore different, as follows:

1. If large pages are allowed,
   a. Request large pages.
   b. If unable to allocate the full amount of configured memory in large pages, allocate small pages if amount of configured memory is smaller than 2 GB.
   c. If configured memory is larger than 2 GB, or if unable to allocate the full amount of configured memory in small pages, reduce the allocation by one eighth (1/8) and begin again with step 1a.
2. If large pages are not allowed,
   a. Reduce memory request size to 2 GB.
   b. Request small pages.
   c. If unable to allocate the full amount of configured memory in small pages, reduce the allocation by one eighth (1/8) and begin again with step 2b.

If the total amount of shared memory configured for Caché exceeds 2 GB on one of these Windows versions and is not allocated as Windows large pages, Caché or Ensemble can become slow or unresponsive. In these circumstances, key Caché/Ensemble processes, such as the Write daemon, spend more and more time in the Windows kernel, visible as percentage of Privileged time in Windows Performance Monitor, **perfmon.exe**.

2.5.8 The Write-cache Buffer

Certain Caché features utilize **Windows write-cache buffer flushing**, which is enabled by default. To verify this option is correctly enabled and Caché receives the full benefit of these features:

1. Open the **Device Manager** from the **Control Panel**.
2. Select the storage device from the **Disk Drives** section.
3. Click on the **Policies** tab.
4. Ensure that **Turn off Windows write-cache buffer flushing on this device** is *not* selected.

3

Installing Caché on UNIX®, Linux, and macOS

This chapter describes how to install Caché on a UNIX®, Linux, or macOS system. It assumes that you are familiar with UNIX®, Linux, and macOS directory structures, utilities, and commands. Before beginning this installation, be sure you have read all the information that applies to this platform in the chapter "Preparing to Install Caché".

This chapter contains the following major sections:

- Caché Installation
- Unattended Caché Installation
- Post-installation Tasks
- Adding UNIX® Installation Packages to a Caché Distribution
- Special Considerations

You can consult the section on Calculating System Parameters for UNIX®, Linux, and macOS to verify and adjust your parameter settings before you begin the installation.

3.1 Caché Installation

There are four main starting points to run an attended Caché installation on a UNIX® platform. The following sections explain each in detail:

- Performing a Standard Caché Installation — Performs the basic Caché installation, in which you can interactively choose what to install along with the Caché server and select various installation options.
- Installing Caché Components on Linux RPM — Uses the **rpm** command to install Caché separate components without user input.
- Installing Caché as a Nonroot User — Performs a limited Caché installation by a nonroot user when root is not available.
- Installing Client-only Caché — Lets you choose the appropriate client components to install without installing the Caché server.

3.1.1 Performing a Standard Caché Installation

Standard Caché installation consists of a set of modular package scripts. The scripts conditionally prompt for information based on input to previous steps, your system environment, and whether or not you are upgrading an existing instance. The first stage of the installation stores all gathered information about the install in a parameter file. You then confirm the specifics of the installation before the actual install takes place. The final phase performs the operations that are contingent upon a successful install, such as instance startup.

To install a Caché server, log in as user ID `root`. It is acceptable to **su** (superuser) to `root` while logged in from another account.

Once you are logged into your operating system, perform the steps in the following sections:

1. Uncompressing the Installation Kit (if necessary).
2. Determining the Installation Directory
3. Determining Owners and Groups
4. Running the Installation Script

3.1.1.1 Uncompressing the Installation Kit

If your installation kit is in the form of a .tar file, for example cache-2015.2.0.600.0-lnxrhx64.tar.gz, uncompress the file into a temporary directory to avoid permissions issues, as shown in the following example.

Important: Do not uncompress the file into or run Caché installation from the /home directory, or any of its subdirectories.

```
# mkdir /tmp/cachekit
# chmod og+rx /tmp/cachekit
# umask 022
# gunzip -c /download/cache-2016.2.0.600.0-lnxrhx64.tar.gz | ( cd /tmp/cachekit ; tar xf - )
```

The installation files uncompress into a directory with the same name as the .tar file, for example /tmp/cachekit/cache-2016.2.0.600.0-lnxrhx64 in the preceding example.

Note: The pathname of the temporary directory cannot contain spaces.

Because legacy **tar** commands may fail silently if they encounter long pathnames, InterSystems recommends that you use **GNU tar** to untar this file. To determine if your **tar** command is a GNU **tar**, run **tar --version**.

3.1.1.2 Determining the Installation Directory

You must choose a fully resolved physical path, containing no symbolic links, for the Caché installation directory; there is no default location.

Important: Be sure to see Installation Directory in the "Preparing to Install" chapter for important information about choosing an installation directory.

Note: The Caché registry directory, /usr/local/etc/cachesys, is always created along with the Caché installation directory.

3.1.1.3 Determining Owners and Groups

The installation process prompts for the following user and group information (see the UNIX® User and Group Identifications section in the "Using Caché on UNIX®, Linux, and macOS" chapter of the *Caché System Administration Guide*):

- *Owner of the instance*
- *Effective user for the Caché superserver and its jobs*
- *Effective group for Caché processes*
- *Group allowed to start and stop the instance*

Important: The user you identify as the owner of the instance needs read and execute access to the installation kit. To that end, the installing user should have a **umask** value of `022` (or less restrictive) when they untar the installation kit.

Additionally, Caché must set user, group, and other permissions on files that it installs. To accomplish this, Caché sets **umask** to `022` for the installation process - do *not* modify the **umask** until the installation is complete.

The user account you identify as the instance owner and the group you identify as the instance start/stop group must exist before you begin installation. If an entry you provide at one of these prompts does not exist, the prompt is repeated, so verify that the user and group you intend to provide exist before you begin installation.

If your operating system contains the **useradd** and **groupadd** utilities, the system creates the account for the effective user for Caché superserver and the effective group for Caché processes, if the entries you provide do not exist. However, if these utilities are not present and an entry you provide does not exist, the prompt is repeated. If you are not sure that your system has these utilities, verify that the user and group you intend to provide exist before you begin installation.

Note: If your operating system uses Network Information Services (NIS) or another form of network-based user/group database, the **groupadd** and **useradd** utilities may create a local user and/or group that could conflict with existing entries in the network database. To avoid this problem, it may be best to create the Caché effective group and effective user in your network database using the appropriate administration tools prior to beginning installation, rather than allowing the utilities to create them.

Tools used on UNIX® operating systems to display process ownership may or may not show effective versus real ownership. See the "Using Caché on UNIX®, Linux, and macOS" chapter of the *Caché System Administration Guide* for details on how Caché assigns permissions.

3.1.1.4 Running the Installation Script

The installation script, **cinstall**, which must be run by a user with root privileges, automatically does the following:

- Installs the Caché system manager databases.
- Starts Caché in installation mode.
- Installs Caché system manager globals and routines.
- Shuts down Caché and restarts using the default configuration file (cache.cpf). Upgrade installations restart using their updated configuration files.

Note: If the profile of the user executing **cinstall** has a value set for the *CDPATH* variable, the installation fails.

A user without root privileges can perform a non-standard, limited Caché installation; see Installing Caché as a Nonroot User for details.

Install Procedures

To perform the standard Caché server installation:

1. As a user with root privileges, start the installation procedure by running the cinstall script, located at the top level of the installation files:

   ```
   # /<pathname>/cinstall
   ```

   where *pathname* is the location of the installation kit, typically the temporary directory to which you have extracted the kit, as described in Uncompressing the Installation Kit.

2. The installation script identifies your system type and validates it against the installation type on the distribution media. If your system supports more than one type, for example, 32-bit and 64-bit, or if the install script cannot identify your system type, it prompts you with additional questions. If your system type does not match that on the distribution media, the installation stops. Contact the InterSystems Worldwide Response Center (WRC) for help in obtaining the correct distribution.

3. The script displays a list of any existing Caché instances on the host.

4. At the **Enter instance name:** prompt, enter an instance name, using alphanumeric characters only. If an instance with this name already exists, the program asks if you wish to upgrade it. If no such instance exists, it asks if you wish to create it and asks you to specify its installation directory. If the directory you specify does not exist, it asks if you want to create it. The default answer to each of these questions is `Yes`; press **Enter** to continue with the installation.

   Note: If you select an existing instance that is of the same Caché version as the installation kit, the installation is considered an upgrade, and you can use the `Custom` selection, described in the following step, to modify the installed client components and certain settings.

5. Next select the setup type.

   The core of a functioning Caché instance is the Caché server. You can install the server and the CSP web server gateway, plus the tools needed for development work, including Studio, the ODBC and JDBC drivers, and all supported language bindings. You can also install the server and the CSP gateway only, without the development tools. Finally, you can do a custom install in which you choose the components to install and specify some additional installation options. These choices are shown in the following:

   ```
   Select installation type.
       1) Development - Install Cache server and all language bindings
       2) Server only - Install Cache server
       3) Custom - Choose components to install
   Setup type <1>?
   ```

   The default for new instances is `Development`; if you are upgrading an existing instance, the default is the option that was used to originally install the instance. See Caché Custom Installation section for details on the `Custom` option.

6. Next, indicate whether to install Caché with 8-bit or Unicode character support. On upgrade, you can convert from 8-bit to Unicode, but not the reverse.

7. For new instances, you must next specify security settings. (Security settings cannot be changed when upgrading.)

   First, decide how restrictive you want the initial Caché security settings to be: choose from Minimal (`1`), Normal (`2`), and Locked Down (`3`). The default is Minimal; if you choose this, the owner of the instance becomes `root` and the installation continues with the next step.

   If you enter `2` or `3`, the script asks for additional information:

   a. Instance owner — Enter the username of the account under which to run Caché processes. Once Caché is installed, you cannot change the owner of the instance. See Determining Owners and Groups for information about this account. A Caché user account is created with this username and the password you enter at the next prompt.

   b. Password for the instance owner — Enter the password for the username you entered at the previous prompt, and enter it again to confirm it.

This password is used not only for the Caché user account is created for the instance owner username, but also for the _SYSTEM, Admin, and SuperUser predefined user accounts. For more details on these predefined users, see the Predefined User Accounts section of the "Users" chapter of the *Caché Security Administration Guide*.

c. Password for the CSPSystem predefined user.

Note: Passwords entered during this procedure cannot include space, tab, or backslash characters; the installer reject such passwords.

For a detailed explanation of these settings, see the Initial Caché Security Settings section in the chapter "Preparing for Caché Security" om this guide.

Important: There are additional security settings that you can choose only through a custom install. See Caché Custom Installation for details.

8. At this point in the installation, you are asked which group should be allowed to start and stop Caché. Only one group can have these privileges, and it must be a valid group on the machine; see Determine Owners and Groups for more information. Enter the name or group ID number of an existing group; Caché verifies that the group exists before proceeding.
9. Finally, for a new instance, or if the script does not detect a cache.key file in the mgr directory of an existing instance when upgrading, you are prompted for a license key file. If you specify a valid key, the license is automatically activated and the license key copied to the instance's mgr directory during installation, and no further activation procedure is required. If you do not specify a license key, you can activate a license key following installation. See Activating a License Key in the "Managing Caché Licensing" chapter of the *Caché System Administration Guide* for information about licenses, license keys and activation.
10. Review your installation options and press enter to proceed with the installation. File copying does not begin until you answer `Yes`.

    A standard installation sets the following port numbers for a Caché instance:

    - *Superserver port number* — `1972` or the first available subsequent number equal to or higher than `56773`
    - *Web server port number* — `57772` or the first available subsequent number

    If you want to assign different port numbers during a new installation, you must choose the custom option. See the Caché Custom Installation section for details.

    You can change the Superserver port number after installation from the **Memory and Startup** page (**System Administration** > **Configuration** > **Memory and Startup**).

    You can change the Web server port number after installation from the **Startup Settings** page (**System Administration** > **Configuration** > **Startup Settings**).

When the installation completes, you are directed to the appropriate URL for the Management Portal to manage your Caché system. See the "Using the Management Portal" chapter of the *Caché System Administration Guide* for more information.

Caché Custom Installation

If you choose a custom installation, you must answer additional questions throughout the procedure about installing several individual components. The defaults appear in brackets before the question mark (?); press **Enter** to accept the default.

- *Superserver and Web Server Ports* — You can choose to let Caché assign the port numbers as described in the standard installation procedure or you can enter custom port numbers. Upgrade installs do not offer this choice; they keep the port numbers of the original instance.

- *CSP Gateway and External Web Sever* — You can configure the CSP gateway to use a supported external web server. Answer **Yes** and then answer additional web server configuration questions to configure the CSP gateway after the Caché installation completes.
- *Additional Security Options* — If you chose Normal or Locked Down initial security, you have the option of onfiguring additional security settings (minimal installations use the default IDs):
  - *Effective group for Caché* — Caché internal effective group ID, which also has all privileges to all files and executables in the installation. For maximum security, no actual users should belong to this group. (Defaults to `cacheusr`.)
  - *Effective user for the Caché superserver* — Effective user ID for processes started by the superserver and Job servers. Again, for maximum security, no actual users should have this user ID. (Defaults to `cacheusr`.)

  See Determine Owners and Groups for additional information.
- *Client Components* — You have the option of installing one or more client components. The list of components to choose from depends on the platform; the illustration below shows all possible components, some of which are not supported on some platforms.

```
Client component selection

   [*]    1) ODBC client
   [*]    2) C++ binding
   [*]    3) C++ SDK
   [*]    4) Perl binding
   [*]    5) Python binding
   [*]    6) Cache engine link libraries (callin)
   [*]    7) Light C++ binding
          +) Select all

Enter the number of each component you wish to install.
Enter the number of an already selected component to deselect it.
Multiple selections can be separated by spaces.
Enter a blank line to continue.
```

  You can use the plus sign (+) to select all components. If this is an upgrade and you clear a previously installed component, that component is neither upgraded nor uninstalled.

See the following guides in the *Caché Language Bindings* set for more information on the specific client components in the selection list:

- *Using Caché ODBC*
- *Using C++ with Caché*
- *Using Perl with Caché*
- *Using Python with Caché*
- *Using the Caché Callin API*

3.1.2 Installing Caché Components on Linux RPM

For Linux platforms only, InterSystems provides RPM installation kits for component installations. Using these kits, you can perform the following types of Caché installs:

- Server (see Run the Installation Script)
- Development (see Run the Installation Script)
- CSP gateway (see Caché Custom Installation)
- ODBC client (see Caché Custom Installation)

- ISCAgent (see ISCAgent and Configuring the ISCAgent in the "Mirroring" chapter of the *Caché High Availability Guide*)

Please note the following points about RPM installation:

- RPM installation can install only one Caché instance per host and does not provide any installation options.
- The instance name is always CACHE; the installation directory is always /usr/cachesys.
- The Unicode version of Caché is always installed.
- The Minimal initial security setting is always used.
- The installation does not prompt for a license key.
- Having installed the server component, you can use the other package files to add the other components.

Install from the appropriate RPM package file using the following command:

```
# rpm -ivh cache-development-2016.2.0.569.0-1.rh.x86_64.rpm
```

For an upgrade from an RPM package file, use the following command:

```
# rpm -U cache-development-2016.2.0.569.0-1.rh.x86_64.rpm
```

Note: Caché does not support the `--relocate` option of the **rpm** command, due to the complexity of the package file.

Important: RPM installation of a new Caché instance does not start the instance following installation.When you start Caché the first time after the installation, it performs the required initialization actions including initializing the databases and creating the configuration file. See the Starting Caché section for details on how to start Caché after the installation.

An RPM upgrade install stops the Caché instance before it replaces files and starts it afterwards, if it was running before the upgrade.

3.1.3 Installing Caché as a Nonroot User

Although Caché is typically installed as root, it is possible for an instance to be installed and run by another user on a system on which root is not available. The operation of Caché in such an environment is identical to that of standard Caché instances, except for the following:

- The *CACHESYS* environment variable must be defined as an existing directory writable by the installing user, and must be present during installation and all instance operations (**cstart**, **cstop**, **csession**).
- The instance can be operated only by the installing user.
- All Caché executables and processes run as the installing user.
- All instance files, including the registry, are owned and can be read, written, and executed by the installing user only. There is no group access.

  For example, where a standard instance might have:

```
-rws--x--- 5 root develop 43282 Aug 28 07:52 cmgr
-r-x--s--x 1 <nonroot-user> cacheusr 23058 Aug 28 07:52 cuxsession
```

  a nonroot instance would have:

```
-rwx------ 5 <installing-user> develop 43282 Aug 28 07:52 cmgr
-r-x------ 1 <installing-user> develop 23058 Aug 28 07:52 cuxsession
```

The registry is located in the directory specified by *CACHESYS*, and nonroot instances are found in that registry. (The **ccontrol** and **csession** executables are in that directory as well.) Users who employ *CACHESYS* as a registry location for root-installed instances are unaffected. Note, however, that only nonroot instances may be in the nonroot registry. Any attempt to access a standard instance from a nonroot registry will fail. Conversely, a nonroot instance may be defined in a root-registry, but an attempt to access the instance by any user other than the owner will fail.

Note: InterSystems recommends that the registry be placed in a directory that is local to the machine on which the instance is installed, not an NFS directory. Note that the standard location /usr/local/etc is such a directory.

Nonroot installation differs from standard installation in the following ways:

- There is no option to configure the CSP Gateway to use an external web server.
- The ISCAgent (see ISCAgent and Configuring the ISCAgent in the "Mirroring" chapter of the *Caché High Availability Guide*) is installed in the directory specified by *CACHESYS*.

  Note: For information about starting the ISCAgent for a nonroot instance, see Starting the ISCAgent for Nonroot Instances on UNIX®/Linux and macOS Systems in the "Mirroring" chapter of the *Caché High Availability Guide*.

  Use of a mirror Virtual IP, as described in Planning a Mirror Virtual IP (VIP) in the "Mirroring" chapter, is not possible with a nonroot instance.

- The instance owner and the group allowed to start and stop Caché are not specified.

3.1.4 Installing Client-only Caché

The Caché distribution contains a separate script to install a client-only version of Caché. The installation process is fairly simple. You do not need to install as `root`. The files from this install have the user and group permissions of the installing user. To perform the Caché client installation:

1. Start the installation procedure by running the cinstall_client script, located at the top level of the installation files:

   ```
   # /pathname/cinstall_client
   ```

2. Choose from the available client component options. Components that require the Caché server on the same machine do not appear in the list. For example:

   ```
   Client component selection

      [*]    1) ODBC client
      [*]    2) C++ binding
      [*]    3) C++ SDK
      [*]    4) Perl binding
      [*]    5) Python binding
             +) Select all

   Enter the number of each component you wish to install.
   Enter the number of an already selected component to deselect it.
   Multiple selections can be separated by spaces.
   Enter a blank line to continue.
   ```

   You can use the plus sign (+) to select all components. If this is an upgrade and you clear a previously installed component, that component is neither upgraded nor uninstalled.

The list of client-only components does not include the engine link libraries or the light C++ binding because these components require a server installation.

You cannot use this script to update client components in server installations. Use the cinstall script instead.

See the following guides in the *Caché Language Bindings* set for more information on the specific client components in the selection list:

- *Using Caché ODBC*
- *Using C++ with Caché*
- *Using Perl with Caché*
- *Using Python with Caché*

3.2 Unattended Caché Installation

You can perform unattended installation and upgrade of Caché instances on your systems using the cinstall_silent script. Whereas a standard install operation obtains the required specifications and selections in the form of user responses to prompts, an unattended operation gets this information from the configuration parameters and the packages specified on the cinstall_silent command line. Each specified package represents a Caché component; the installation scripts for each component are contained in the packages directory below the directory containing the cinstall_silent script.

The general format for the cinstall_silent command line is to precede the command itself by setting environment variables to define the installation parameters, as follows:

```
sudo ISC_PACKAGE_INSTANCENAME="<instancename>"
 ISC_PACKAGE_INSTALLDIR="<installdir>"
 ISC_PACKAGE_PLATFORM="<platform>" ISC_PACKAGE_UNICODE="Y"|"N"
 ISC_PACKAGE_INITIAL_SECURITY="Minimal"|"Normal"|"Locked Down"
 ISC_PACKAGE_MGRUSER="<instanceowner>" ISC_PACKAGE_MGRGROUP="<group>"
 ISC_PACKAGE_USER_PASSWORD="<pwd>"  ISC_PACKAGE_CSPSYSTEM_PASSWORD="<pwd>"
 ISC_PACKAGE_CACHEUSER="<user>" ISC_PACKAGE_CACHEGROUP="<group>"
 ISC_PACKAGE_CLIENT_COMPONENTS="<component1> <component2> ..."
 ISC_PACKAGE_STARTCACHE="Y"|"N"
 ./cinstall_silent [<pkg> ...]]
```

This section contains the following subsections:

- Unattended Installation Parameters
- Unattended Installation Packages
- Unattended Installation Examples

3.2.1 Unattended Installation Parameters

The following table describes the parameters used with the cinstall_silent script in unattended installation.

Table 3–1: Unattended Installation Parameters

Parameter	Description
`ISC_PACKAGE_INSTANCENAME="<instancename>"` (Required)	Specifies the name of the instance to be installed or upgraded: if the instance does not exist, this is a new installation; if it does exist, this is an upgrade. For example: `ISC_PACKAGE_INSTANCENAME="MyCache"` **Note:** If this a new install, the `ISC_PACKAGE_INSTALLDIR` parameter is required.

Parameter	Description
`ISC_PACKAGE_INSTALLDIR="<installdir>"` (Required for new instances)	Specifies the installation directory for the new instance to be installed; for example: `ISC_PACKAGE_INSTALLDIR="/opt/MyCache"` If the specified directory does not exist, the installation attempts to create it. This parameter is ignored if you are upgrading an installation. **Note:** If this a new install, but this parameter is not specified, the installation fails and an error is thrown.
`ISC_PACKAGE_PLATFORM="<platform>"` (Optional)	Specifies a platform when installing Caché on a system that supports multiple platforms on the same hardware. **Important:** Currently, the only ISC-supported platform that supports this feature is IBM AIX®; valid values are `ppc` (32–bit system) and `ppc64` (64–bit system). For example: `ISC_PACKAGE_PLATFORM="ppc"` **Note:** By default, the platform is auto-detected. If the parameter is omitted and no supported platform is detected, unattended installation terminates with an error.
`ISC_PACKAGE_UNICODE="Y"|"N"` (Optional)	Specifies whether or not this is a UNICODE installation; valid values are `Y` or `N`. **Note:** By default, this is set to `N` (8–bit); see Caché Character Width for more information.
`ISC_PACKAGE_INITIAL_SECURITY="Minimal"|"Normal"|"Locked Down"` (Optional)	Specifies the initial security setting for the installation; valid values are: `"Minimal"`, `"Normal"`, or `"Locked Down"`. **Note:** By default, the security level is set to `"Minimal"`. If it is set to `"Normal"` or `"Locked Down"`, `ISC_PACKAGE_USER_PASSWORD` is required.

Parameter	Description
ISC_PACKAGE_MGRUSER="<*user*>" (Optional)	Specifies the login name of the owner of the installation. For example: ISC_PACKAGE_MGRUSER="jcsmith" **Note:** By default, this is set to the username of the user who is installing the instance. If the security level is "Minimal", this parameter is ignored, and ISC_PACKAGE_MGRUSER is set to "root".
ISC_PACKAGE_MGRGROUP="<*group*>" (Optional)	Specifies the group that is allowed to start and stop the instance. For example: ISC_PACKAGE_INITIAL_MGRGROUP="cacheusr" **Note:** By default, this is set to the group of the user who is installing the instance.
ISC_PACKAGE_USER_PASSWORD="<*password*>" (Required for installation of secure instances.)	Specifies the required password for an instance with normal or locked down security. **Note:** If the security level is "Minimal", this parameter is ignored. If the security level is "Normal" or "Locked Down" and this parameter is not specified, the installation fails and an error is thrown.
ISC_PACKAGE_CSPSYSTEM_PASSWORD="<*password*>"	Specifies the password for the CSPSystem user. **Note:** If the security level is "Minimal", this parameter is ignored. If the security level is "Normal" or "Locked Down" and this parameter is not specified, the value of ISC_PACKAGE_USER_PASSWORD is used.
ISC_PACKAGE_CACHEUSER="<*user*>" (Optional)	Specifies the effective user for the Caché SuperServer. **Note:** By default, this is set to cacheusr. If the security level is "Minimal", this parameter is ignored and set to the default.

Parameter	Description
ISC_PACKAGE_CACHEGROUP="*group*" (Optional)	Specifies the effective user for Caché processes. **Note:** By default, this is set to cacheusr. If the security level is "Minimal", this parameter is ignored and set to the default.
ISC_PACKAGE_CLIENT_COMPONENTS="*component1* *component2* ..." (Optional)	Specifies the client bindings to be installed from the client_components package (see Unattended Installation Packages). **Note:** By default, all client bindings are installed. Specified components (bindings) must be space-delimited. Available components are listed in package/client_components/manifest.isc. Installation validates the specified components and removes those that do not exist or are not supported on a particular system.
ISC_PACKAGE_CSP_CONFIGURE="Y"|"N" (Optional)	Specifies whether or not to configure the CSP Gateway for an external web server. **Note:** By default, this is set to N (do not configured the gateway for an external web server).
ISC_PACKAGE_CSP_SERVERTYPE="Apache"|"SunOne"|"None" (Optional)	Type of existing web server for the CSP Gateway to use. For example: ISC_PACKAGE_CSP_SERVERTYPE="Apache" **Note:** By default, this is set to None.
ISC_PACKAGE_CSP_APACHE_VERSION=2.2|2.4 (Optional, with ISC_PACKAGE_CSP_SERVERTYPE="Apache")	Version of Apache web server. **Note:** By default, the version is autodetected.
ISC_PACKAGE_CSP_APACHE_USER="*username*" (Optional, with ISC_PACKAGE_CSP_SERVERTYPE="Apache")	Username for Apache web server. **Note:** By default, the username is autodetected.

Parameter	Description
ISC_PACKAGE_CSP_APACHE_CONF="*<path_to_httpd.conf>*" (Optional, with ISC_PACKAGE_CSP_SERVERTYPE="Apache")	Location of the Apache Web server configuration file, for example: /etc/httpd/conf/httpd.conf **Note:** By default, installation attempts to autodetect file in one of several standard locations. Installation exits with error if ISC_PACKAGE_CSP_SERVERTYPE="Apache" and httpd.conf location is undetermined.
ISC_PACKAGE_CSP_SUNONE_DIR="*<sunone_installation_dir>*" (Optional, with ISC_PACKAGE_CSP_SERVERTYPE="SunOne")	Installation directory of the SunOne web server, for example: /usr/netscape/server4/httpd-production **Note:** By default, the installation directory is autodetected.
ISC_PACKAGE_CSP_SUNONE_NUMBER="<number>" (Optional, with ISC_PACKAGE_CSP_SERVERTYPE="SunOne")	SunOne web server number to configure if multiple servers are installed. **Note:** By default, value is 1.
ISC_PACKAGE_CSP_GATEWAY_DIR="<csp_gateway_directory>" (Optional, for new CSP Gateway installations only)	Directory to contain the CSP Gateway files. **Note:** By default, the directory is /opt/cspgateway_.
ISC_PACKAGE_STARTCACHE="Y"|"N" (Optional)	Specifies whether or not to start the installed Caché instance following installation. **Note:** By default, this is set to Y, to start the instance.
ISC_INSTALLER_MANIFEST="<location>"	When installing with an installation manifest, specifies the location of the exported manifest class. **Note:** See Using the Manifest in the chapter "Creating and Using an Installation Manifest" for more information.
ISC_INSTALLER_PARAMETERS="<var>=<value>,<var>=<value>..."	When installing with an installation manifest, specifies variable name/value pairs.
ISC_INSTALLER_LOGFILE="<filename>"	When installing with an installation manifest, specifies the log file name.

Parameter	Description
`ISC_INSTALLER_LOGLEVEL="<level>"`	When installing with an installation manifest, specifies the log level, from `0` (“none”) to `3` (“verbose”).
`ISC_PACKAGE_SUPERSERVER_PORT="<port_number>"` (Optional)	Specifies the Superserver port to be used by the instance being installed. **Note:** By default, this port is auto-determined, beginning with `51773` and increasing by 1 for each installed instance of Caché.
`ISC_PACKAGE_WEBSERVER_PORT="<port_number>"` (Optional)	Specifies the Webserver port to be used by the instance being installed. **Note:** By default, this port is auto-determined, beginning with `52773` and increasing by 1 for each installed instance of Caché.

3.2.2 Unattended Installation Packages

The installation scripts for each component are contained in the packages directory below the directory containing the cinstall_silent script. Each package is in its own directory, and each package directory contains a manifest.isc file defining prerequisite packages for the package in that directory.

The standard_install package is the starting point for a server install in which all packages are installed. To define a custom package, you can use the manifest.isc file for the standard_install package as a template, as follows:

1. Copy the standard_install directory to a new directory.

   For example, copy it to a directory named custom_install; initially, the manifest.isc file in the new directory is similar to the following:

   ```
   #This is the target for a standard (non-client-only) install
   package: standard_install
   prerequisite: install_mode
   prerequisite: database_server
   prerequisite: databases
   prerequisite: gadget
   prerequisite: fop
   prerequisite: renderserver
   prerequisite: printserver
   prerequisite: excelexporter
   prerequisite: callin_components
   prerequisite: client_components
   prerequisite: addenda
   prerequisite: install_confirmation
   prerequisite: copyright
   ```

2. In the new directory, modify the manifest.isc file as follows:

   - Change the package key to match the directory name (**required**).
   - Add and/or remove prerequisites for your custom installation.

   For example, in the following manifest.isc file, the package key has been changed to match the directory name (custom_install) and the `databases` prerequisite has been removed to exclude the Samples and Docbook databases from the installation:

```
#This is the target for a custom (non-client-only) install
package: custom_install
prerequisite: install_mode
prerequisite: database_server
prerequisite: gadget
prerequisite: fop
prerequisite: renderserver
prerequisite: printserver
prerequisite: excelexporter
prerequisite: callin_components
prerequisite: client_components
prerequisite: addenda
prerequisite: install_confirmation
prerequisite: copyright
```

Then you can specify the new custom package when performing unattended installations; for example: `sudo ISC_PACKAGE_INSTANCENAME="MyCache" ./cinstall_silent custom_install`.

Note: See Adding UNIX® Installation Packages to a Caché Distribution for information about creating your own UNIX® installation packages and adding them to a Caché distribution.

3.2.3 Unattended Installation Examples

The following examples illustrate how you can use the cinstall_silent script for unattended installation of Caché on UNIX® platforms.

Example 1

In this example, the 8-bit versions of all packages are installed with minimal security:

```
sudo ISC_PACKAGE_INSTANCENAME="MyCache" ISC_PACKAGE_INSTALLDIR="/opt/mycache1" ./cinstall_silent
```

If the MyCache instance already exists, it is upgraded; otherwise, it is installed in the /opt/mycache1 directory.

Example 2

In this example, the installation is aborted and an error is thrown if the instance named MyCache does not already exist:

```
sudo ISC_PACKAGE_INSTANCENAME="MyCache" ./cinstall_silent
```

Example 3

In this example, the 8–bit versions of only the `database_server` and `odbc` packages and the `odbc` client binding are installed with minimal security:

```
sudo ISC_PACKAGE_INSTANCENAME="MyCache" ISC_PACKAGE_INSTALLDIR="/opt/mycache2"
ISC_PACKAGE_CLIENT_COMPONENTS="odbc" ./cinstall_silent database_server odbc
```

3.3 Post-installation Tasks

Once you have completed running the installation procedure, perform the following tasks:

- Start Caché
- If you are upgrading from a previous version of Caché, see Post-installation Upgrade Tasks in the "Upgrading Caché" chapter of this book.
- If you plan to develop using Studio, see the Install Caché Client on Windows for Development section.
- If appropriate for your installation, perform any additional tasks described in the Special Considerations section.

3.3.1 Starting Caché

The evaluation RPM Linux install does not start Caché when it completes. Start and stop Caché using the standard Linux methods. For example:

```
/etc/init.d/cache start
```

On Red Hat Linux, you can also use the following:

```
service <instname> start
```

Where *instname* is the instance name that you chose during the installation.

All other installs leave Caché running. If you need to start Caché, first log into your operating system, then start Caché using the **ccontrol** command:

```
ccontrol start <instname>
```

Where *instname* is the instance name that you chose during the installation.

Use the **ccontrol** command to start and stop Caché. It is described in greater detail in the Controlling Caché Instances section of the *Caché System Administration Guide*.

Note: If the permissions on all elements of the path to the mgr subdirectory do not provide read access to the `cacheusr` group (at a minimum), the instance fails to fully start and the following message is recorded in cconsole.log: `Element of path manager_subdirectory could not be read (errno 2).`

Once Caché is started, initiate a Caché session using the **csession** command:

```
csession <instname> [parameters]
```

Where *instname* is the instance name that you chose during the installation.

For more information, see the "Using Caché on UNIX®, Linux, and macOS" chapter of the *Caché System Administration Guide*.

3.3.2 Installing Caché Client on Windows for Development

Caché installs a private Apache Web server so that you can access the Management Portal; therefore, a UNIX® system does not require a Caché client on a Windows machine to perform system configuration and management tasks.

You do, however, require a Windows client to use the Studio development tool. See Performing a Caché Client Installation in the "Installing Caché on Microsoft Windows" chapter of this book for information on installing the Windows client only. Once installed, from the Caché Cube of the Windows client, perform the following tasks:

- Point to **Preferred Server** and click **Add/Edit** to add a remote server connection to the Caché instance just installed. Make sure you specify the appropriate port numbers for this connection.
- Point to **Remote System Access**, point to **Studio**, and then click the appropriate connection name you entered in the previous step.

3.4 Installing Caché on Mac

This chapter describes how to install Caché on a macOS system. It assumes that you are familiar with Mac directory structures, utilities, and commands. This chapter contains the following major sections:

- Installation Requirements
- Caché UNIX® Installation

3.4.1 Installation Requirements

This section describes the hardware and software requirements for installations of Caché.

3.4.1.1 Disk Space Requirements

A standard Caché installation that includes support for Caché Server Pages (CSP), needs approximately 1 GB (gigabytes) of disk space depending on the type of installation you choose.

3.4.1.2 Supported Platforms and Web Servers

The latest version of Caché is supported on the macOS operating system on Intel. For macOS, the Caché Server Pages (CSP) technology is supported on the Apache and Nginx web servers. For current information, see the "Supported Technologies" chapter of the online *InterSystems Supported Platforms* document for this release.

Note: When installing Caché and Ensemble, a private version of Apache is installed to ensure that:

1. The Management Portal runs out of the box.
2. An out-of-the-box testing capability is provided for development environments.

The private Apache web server is not supported for any other purpose.

For deployments of http-based applications, including CSP, Zen, and SOAP over http or https, you should not use the private web server for any application other than the Management Portal; instead, you must install and deploy one of the supported web servers (for information, see "Supported Web Servers" in the "Supported Technologies" chapter of the online *InterSystems Supported Platforms* document for this release).

If using CSP, you must install the web server before installing Caché for the installation to configure the web server. Its support on each operating system is dependent on the operating system vendor and is subject to change. See the Web Server Configuration section of the "CSP Configuration" chapter of the *Using Caché Server Pages* guide for more information.

Important: For up-to-date information about platforms, versions, and features supported for each operating system, see the "Supported Technologies" chapter of the online *InterSystems Supported Platforms* document for this release.

3.4.2 Caché UNIX® Installation

The installation of Caché on the macOS is much like the installation on any UNIX® platform.

To install Caché:

1. Obtain the installation kit from InterSystems and install it on the desktop.
2. Log in as user ID `root`. It is acceptable to **su** (superuser) to `root` while logged in from another account.

3. See Adjustments for Large Number of Concurrent Processes and make adjustments if needed.
4. Follow the instructions in the Run the Installation Script section and subsequent sections of the "Installing Caché on UNIX and Linux" chapter of this guide.
5. For post-installation tasks and other information about Caché on macOS, see the Installing Caché on a UNIX® or Linux sections of the "Installing Caché on UNIX and Linux" chapter of this guide.

3.4.3 Adjustments for Large Number of Concurrent Processes

Make the following adjustments if you are running a system that requires a large number of processes or telnet logins:

1. *Remote connections* — The number of pty (pseudo terminal) connections is limited to 128 system-wide. If your applications count on telnet or other pty-using connections for users to access, keep this in mind.
2. *Number of processes* — If the pty limit is not a problem, but you need to run a larger number of processes, there are limits to that as well.
   - *System-wide process limits* — The *kern.maxproc*, *kern.maxprocperuid* parameters are set to `532` and `100` by default. You can change them using the following commands:

     ```
     administrator$ sudo sysctl -w kern.maxproc=2500
     kern.maxproc: 2065 -> 2500
     administrator$ sudo sysctl -w kern.maxprocperuid=2500
     kern.maxprocperuid: 2000 -> 2500

     administrator$ sysctl -a | grep maxproc
     kern.maxproc = 2500
     kern.maxprocperuid = 2500
     ```

     Note, however, that `2500` is the absolute unchangeable upper limit.

3.5 Adding UNIX® Installation Packages to a Caché Distribution

This section describes how to add a new UNIX® installation package to an existing Caché distribution. It is presented in the form of a tutorial in which we create a simple package that copies additional files into the Caché instance directory.

Note: Because install packages are implemented through UNIX® shell scripts, you can also write packages that perform much more complex operations.

Suppose we have written a Caché Callout shared library (see the "Creating a Caché Callout Library" chapter in *Using the Caché Callout Gateway*) to connect to an imaging device named Foo9000. We compile this library as libfoo9000.so and want to install it with Caché. In addition, we want the installation to prompt users to provide the network server name for the device (Foo9000) to which we want the library to connect. This information will be stored in a configuration file in the Caché instance manager's directory *(install-dir*\mgr).

We start with an existing Caché kit:

```
~/kit:>ls
cinstall   cplatname docs  lgpl.txt NOTICE
copyright.pdf dist   kitlist LICENSE package
```

... and our compiled library (libfoo9000.so):

```
~/lib:>ls
libfoo9000.so
```

First, we need to choose a location in the kit to store our library, then copy the library to that location. By convention, platform-specific libraries go in dist/*package*/*platform* directories (for example, ~/kit/dist/foo9000/lnxsusex64):

```
~/kit:>cd dist
~/kit/dist:>mkdir foo9000
~/kit/dist:>cd foo9000
~/kit/dist/foo9000:>mkdir lnxsusex64
~/kit/dist/foo9000:>cd lnxsusex64
~/kit/dist/foo9000/lnxsusex64:>cp ~/lib/libfoo9000.so .
```

Next, we need to create the installation package directory and add the manifest.isc file (which describes the package) to it. In its simplest form, the manifest.isc file includes only the name of the package, which must be identical to the name of the package directory (foo9000).

```
~/kit/package:>mkdir foo9000
~/kit/package:>cd foo9000
~/kit/package/foo9000:>emacs manifest.isc
package: foo9000
```

Without any content the package does not do anything, but in this tutorial we want to do the following:

1. Prompt users for the name of the server hosting the Foo9000.
2. Save this information in a configuration file in the manager's directory (*install-dir*\mgr).
3. Copy the library (libfoo9000.so) into the instance binary directory.

The package installer performs actions in phases, the most important of which are the following:

- “parameters” phase
- “install” phase

Note: Packages can contain Bourne shell scripts, with the same name as the phase, for each phase. The package installer runs the script for each package at the appropriate time during the phase. If your package script successfully completes its given phase, it returns an error code of 0 explicitly or implicitly via its final command; otherwise it returns a non-zero error code.

The “parameters” phase collects information necessary for the package's installation, typically by prompting users, and should not make any permanent changes to the system. Users are typically given the opportunity to cancel the installation after the “parameters” phase; if they do so, the installation should have had no effect on their system.

The “install” phase modifies the system. During the install phase users should *not* be prompted for information because the install may be unattended or automated.

Some packages do not require information from users and, therefore, do not need a “parameters” script. If the script for a particular phase is not included in a package, no actions are performed for that package during the phase.

Here is our first attempt at a “parameters” script for the foo9000 package:

```
~/kit/package/foo9000:>emacs parameters
#!/bin/sh
echo "Please enter host name of the Foo9000 imaging server: "
read host
echo "Host $host entered."
```

If we try running this script, as follows, we see that it does indeed prompt us for the host name. which it records in the *host* variable:

```
~/kit/package/foo9000:>sh parameters
Please enter host name of the Foo9000 imaging server:
host1
Host host1 entered.
```

However, what do we do with the *host* value once we've acquired it? When the script is finished running, it will be lost and unavailable when we need to write it to the configuration file during the "install" phase.

Note: Remember that we do not want to create the configuration file now because the "parameters" phase should have no effect on the user's system.

The package installer provides a convenient pair of functions – **Import** and **Export** – that let multiple phases and multiple packages share information. We can use these functions by including them in the parameters.include file through the usual shell script mechanism:

```
#!/bin/sh
. parameters.include
echo "Please enter host name of the Foo9000 imaging server: "
read host
echo "Host $host entered."
Export foo9000.host $host
```

The **Export** function takes the name of a parameter variable to export and its value, typically from a variable local to the script. The **Import** function works in reverse: the first argument is the local variable into which you want to import the previously exported value, and the second argument is the name of the parameter variable to which it was exported.

Note: By convention, parameter variables are given a name of *package name.local variable name* (for example, foo9000.host).

Since our "parameters" script now collects all the Foo9000 information needed to complete the installation, we can turn to writing the "install" script:

```
~/kit/package/foo9000:>emacs install
#!/bin/sh
. parameters.include
Import host foo9000.host
echo host=$host > ????/mgr/foo9000.cfg
cp ????/dist/foo9000/????/libfoo9000.so ????/bin
```

There are a few details (???? in the preceding script) we need to provide:

- Where is the instance directory in which the install is being created?
- Where is the kit we're installing from?
- Which platform is being installed?

Although we could include these questions in the "parameters" script, that may confuse users because they already entered that information earlier in the install. Instead, we import parameter variables from other packages that can provide the information we need. This is possible because each successful installation using the cinstall, cinstall_client or cinstall_silent scripts (as described in Caché Installation) creates the parameters.isc file, which contains these variables and their values, in the installation directory. The variables in the parameters.isc file are listed in the Caché Installation Parameter File Variables table at the end of this chapter.

Note: For security reasons, the parameters.isc file is accessible only by the root user.

In order to use the parameter variables from a particular package, we must inform the package installer that our package (foo9000) depends on the other package and, therefore, our package must be processed later in each phase than the other package. We do this by adding "prerequisite" values to our package's manifest.isc file:

```
~/kit/package/foo9000:>emacs manifest.isc
package: foo9000
prerequisite: server_location
prerequisite: legacy_dist
prerequisite: platform_selection
```

Now we can import parameter variables from these packages and use them to complete our install script:

```
~/kit/package/foo9000:>emacs install
#!/bin/sh
. parameters.include
Import host foo9000.host
Import tgtdir "server_location.target_dir"
Import srcdir "legacy_dist.source_dir"
Import platform_family "platform_selection.platform_family"
echo host=$host > $tgtdir/mgr/foo9000.cfg
cp $srcdir/dist/foo9000/$platform_family/libfoo9000.so $tgtdir/bin
```

Our package (foo9000) is nearly complete. The final task is to add our package to the prerequisite list for an appropriate preexisting package. Then, to complete installation of that package, the package installer will process ours. In this case, we want our library to be installed and configured any time a Caché server is installed, so we add our new package to the "database_server" package's prerequisite list inside its manifest.isc file:

```
~/kit/package/database_server:>emacs manifest.isc
package: database_server
prerequisite: legacy_dist
prerequisite: platform_selection
prerequisite: server
prerequisite: server_location
prerequisite: upgrade
prerequisite: available_disk_space
prerequisite: posix_tools
...
prerequisite: isql
prerequisite: zlib
prerequisite: udp
prerequisite: bi
prerequisite: foo9000
```

As you can see, many packages are required to create a server installation, but now, when we run **cinstall**, our package (foo9000) is configured and installed:

```
~/kit:>sudo ./cinstall
Your system type is 'SuSE Linux Enterprise Server 10 (x64)'.
Currently defined instances:
Cache instance 'INSTANCE1'
directory: /home/user/INSTANCE1
versionid: 2008.2.0.456.0
conf file: cache.cpf (SuperServer port = 1972, WebServer = 57785)
status:  crashed, last used Sat Sep 13 08:37:32 2008
Enter instance name: INSTANCEPACK1
Do you want to create Cache instance 'INSTANCEPACK1' ? Y
...
Please enter host name of the Foo9000 imaging server:
host1
Host host1 entered.
...
Do you want to proceed with the installation ? Y
...
Installation completed successfully
~/INSTANCEPACK1/bin:>ls libfoo*
libfoo9000.so
~/INSTANCEPACK1/mgr:>cat foo9000.cfg
host=host1
```

Contents of the parameters.isc File

The following table lists the variables in the parameters.isc file with a description and an example value or a list of valid values.

Table 3–2: Caché Installation Parameter File Variables

Variable name	Description (Valid values) or Example
dist.source_dir	Source directory of the installation media. `/cachekit`
legacy_dist.source_dir	For legacy purposes, source directory of the installation media. `/cachekit`
product_info.version	InterSystems product version number. `2012.2.0.100.0`
product_info.name	Name of InterSystems product. `(Cache/Ensemble/HealthShare)`
platform_selection.platform	InterSystems abbreviation for install platform. `lnxrhx64`
platform_selection.platform_family	InterSystems abbreviation for install platform family. `lnxrhx64`
platform_selection.endianness	Platform endian byte order. `(big/little)`
platform_selection.os	Platform operating system; value of **uname** command. `Linux`
posix_tools.user_add	Portable Operating System Interface (POSIX)-compliant user add tool. `/usr/sbin/useradd`
posix_tools.group_add	POSIX-compliant group add tool. `/usr/sbin/groupadd`
posix_tools.grep	POSIX-compliant grep utility. `grep`
posix_tools.id	POSIX-compliant id utility. `id`
posix_tools.ps_opt	Extend full options for process listing. `-ef`
posix_tools.gzip	Gnu-compatible zip utility. `gzip`
posix_tools.shared_ext	Extension for shared library files. `so`

Variable name	Description (Valid values) or Example
posix_tools.symbolic_copy	POSIX-compliant symbolic copy command. `cp -Rfp`
posix_tools.tr	POSIX-compliant translation utility. `tr`
posix_tools.shared_ext1	Alternate extension for shared library files. `so`
posix_tools.permission	POSIX-compliant permissions applied to selected files. `755`
posix_tools.dir_permission	POSIX-compliant permissions applied to selected directories. `775`
server_location.target_dir	Target directory of server installation. `/test/CACHE`
server_location.is_server_install	Indicates whether or not this is a server installation. `(N/Y)`
server_location.is_nonroot_install	Indicates whether or not this is a nonroot install. `(N/Y)`
server_location.instance_name	Instance name. `CACHE`
server_location.is_new_install	Indicates whether or not this is a new install. (`(N`=upgrade/`Y`=new)
server_location.is_new_directory	Indicates whether or not to create a new directory. `(N/Y)`
server_location.registry_dir	Location of the Caché registry directory (must be on a local filesystem). `/usr/local/etc/cachesys`
server_location.convert_ensemble_registry	If you are upgrading a pre-4.0 Ensemble instance this indicates to consolidate the old Ensemble registry with the Caché registry. `(N/Y)`
server_location.ccontrol	Directory in which **ccontrol** resides during installation. `/cachekit/dist/lnxrhx64/bin/shared/ccontrol`
*postinstall**	Specifies packages to run after parameter file phase. `upgrade`

Variable name	Description (Valid values) or Example
install_mode.setup_type	Type of installation. (Development/Server/Custom)
unicode_selection.binary_type	Binary type of install. (unicode/eightbit)
unicode_selection.install_unicode	Indicates whether or not to install the Unicode version of the product. (N/Y)
security_settings.cache_user	Effective user for the Caché superserver cacheusr
security_settings.cache_group	Effective group for Caché. cacheusr
security_settings.manager_user	Owner of the instance. root
security_settings.manager_group	Group allowed to start and stop the instance. develop
security_settings.dbencrypted	Whether to enable an encryption key at startup (0/1)
security_settings.dbenckeyfile	The path of the encryption key. This parameter may be blank.
security_settings.dbenckeyuser	The name of an administrator who can activate the key. This parameter may be blank.
security_settings.dbenckeypassword	The password for the key administrator. This parameter may be blank.
security_settings.personal_database	Indicates whether or not to use the Personal Database feature. (N/Y)
security_settings.initial_level	Initial security settings. (Minimal/Normal/Locked Down)
security_settings.already_secured	If this is an upgrade from a pre-5.1 instance, indicates the need for security settings. (N/Y)
security_settings.password	Password field cleared before the parameter file is stored if running from **cinstall**.

Variable name	Description (Valid values) or Example
installer.manifest	Location of the `DefaultInstallerClass.xml` (the exported %Installer class); for example: `/home/user/Downloads/DefaultInstallerClass.xml`
installer.manifest_parameters	Location of installer manifest parameters. `SourceDir=/home/user/Downloads`
installer.manifest_loglevel	Specifies the log level of the manifest. `(0/1/2/3)`
installer.manifest_logfile	Specifies the log file name. `/manifests/CACHE-installManifestLog.txt`
manager_source_code.available	Indicates whether or not to install the manager utility source code. `(N/Y)`
port_selection.superserver_port	Superserver port number. `1972`
port_selection.webserver_port	Web server port number. `57772`
port_selection.jdbcgateway_port	Java Database Connectivity (jdbc) gateway port number. `62972`
csp_gateway.configure	Indicates whether or not to configure the CSP Gateway for an external web server. `(N/Y)`
csp_gateway.configure_apache	`(N/Y)`
csp_gateway.web_server_type	Type of existing web server for the CSP Gateway to use. `(Apache/SunOne/None)`
csp_gateway.apache_version	Version of Apache web server
csp_gateway.apache_user	Username for Apache web server
csp_gateway.apache_conf_file	Location of the Apache Web server configuration file. `/etc/httpd/conf/httpd.conf`
csp_gateway.apache_pid_file	File that records the process id of the Apache web server daemon. `/usr/local/apache/logs/httpd.pid`
csp_gateway.apache_use32bit	Indicates whether 32–bit architecture is used for the Apache web server. `Y/N`

Variable name	Description (Valid values) or Example
csp_gateway.sunone_server	Location of the Sun ONE server for the CSP Gateway to use. /usr/netscape/server4/httpd-production
csp_gateway.directory	Directory to contain the CSP Gateway files.
license_key.enter_key	Indicates whether or not to install the key during installation. N/Y
license_key.license_file	Location of the key file information if the value of *enter_key* is Y.
agent.user_account	Username for ISCAgent. iscagent
agent.user_group	Group name for ISCAgent. iscagent
agent.install	Indicates whether or not ISCAgent is installed. (N/Y)
client_location.target_dir	Target directory of a client-only installation. test/CACHE
client_location.is_client_install	Indicates whether or not it is a client install. (N/Y)
*install**	database_server
*postinstall**	database_server
*install**	Used to initiate the installation of the SAMPLES database. samples
samples.install	Indicates whether or not to install the SAMPLES database. (N/Y)
japanese_docs.install	Indicates whether or not to install the Japanese documentation sources. (N/Y)
*install**	Used to initiate the installation of the online documentation. docbook
docbook.install	Indicates whether or not to install the DOCBOOK database for the online documentation. (N/Y)
*install**	Component name to install.

* The install variable appears several times in the parameter file, once for every component to install. A custom or client-only install conditionally generates any or all of the following:

- dev_kit
- odbc
- cpp_binding
- cpp_sdk
- perl_binding
- python_binding
- engine_link_libraries
- light_cpp_binding
- addenda
- install_confirmation
- copyright

3.6 Special Considerations

This section describes the following topics:

- Multiple Caché Instances
- Uninstalling Caché
- Using an Installation Manifest

3.6.1 Multiple Caché Instances

You can install and simultaneously run multiple instances (Caché 4.0 and later) on a single machine. Install Caché as for a single installation, giving each instance a unique name, a unique installation directory, and a unique port number.

Please reference the Configuring Multiple Caché Instances section of the *Caché System Administration Guide*.

3.6.2 Uninstalling Caché

To safely uninstall Caché, perform the following tasks:

1. Find the name of the Caché instance you wish to delete using the **ccontrol** command to list all the instances on your machine:

   ```
   ccontrol list
   ```

2. Verify the instance is stopped. If it is not, stop it with the following command:

   ```
   ccontrol stop <instname>
   ```

Where *instname* is the instance name that you chose during the installation. If it hangs, use the following command to force it down:

```
ccontrol force <instname>
```

3. Remove the instance using the following command:

```
ccontrol delete <instname>
```

4. Remove the installation directory using the following operating system command:

```
rm -r <directory>
```

Important: Be aware that this removes files you may wish to keep. For example: the license key (cache.key), the configuration file (cache.cpf), and the user database file (cache.dat).

RPM Uninstall

If you installed Caché using the RPM package, uninstall using the following option:

```
# rpm -e cache-server
```

The uninstall procedure removes all files installed and created during normal Caché processing, including journal and temporary database files.

Important: The SUSE Linux Enterprise Server 9 platform uses asynchronous scriptlets, so the uninstall process cannot guarantee that Caché stops before it removes files. InterSystems recommends you stop Caché on this platform before you run the RPM uninstall command.

3.6.3 Using an Installation Manifest

You can define an installation manifest that describes a specific Caché configuration and use it with either of the **cinstall** or **cinstall_silent** commands. For more information on installation manifests, see the chapter "Creating and Using an Installation Manifest" in this book.

4

Upgrading Caché

This chapter is intended for customers who are upgrading from Caché release 4.1 and later. Topics in this chapter include:

- Supported Upgrade Paths
- Upgrading to a Maintenance Release
- Performing Upgrade Tasks
- Post-Installation Upgrade Tasks

Important: Before upgrading, review the following documents:

- *Caché Release Notes and Upgrade Checklist*, for general upgrade information and recommendations and for release-specific issues that may apply to your site.

  You may also need to see the *Caché Release Note and Upgrade Checklist Archive*.
- the online *InterSystems Supported Platforms* document for this release, for up-to-date information about supported technologies.

CAUTION: Prior to beginning an upgrade of a Caché or Ensemble instance, it is essential that the instance be shut down cleanly. To verify that the shutdown was clean, examine the cconsole.log file after the shutdown finishes. If the log contains entries similar to the following, then the shutdown was clean:

```
...
05/03/11-14:24:13:234 (5204) 0 Journal restore not required at next startup
05/03/11-14:24:13:234 (5204) 0 Transaction rollback not required at next startup
...
```

If these entries are not present, the instance did not shut down cleanly. Please contact the InterSystems Worldwide Response Center before proceeding with the upgrade.

Note: If you run Caché MultiValue, some files may be overwritten during upgrade; see the section "Notes on Files Overwritten During Upgrade" in the chapter "Installing Caché" in *Using the MultiValue Features of Caché* for more information.

4.1 Supported Upgrade Paths

You can upgrade directly to this release of Caché from release 2009.1 or any later release. However, if you are upgrading a pre-2012.1 instance containing any 2K block size databases, these must be converted to 8K format before upgrading to version 2012.x or beyond. In this situation, use the following procedure:

1. Upgrade the instance to version 2011.1, if necessary.
2. Use the **SYS.Database.Copy()** method to convert all 2K databases to 8K format.

   Note: This method requires the database to be mounted. Only versions prior to 2014.1 support mounting 2K databases. Versions 2014.1 and later do not support this operation.

3. Upgrade the instance from 2011.1 to the current release.

See Configuring Databases in the "Configuring Caché" chapter of the *Caché System Administration Guide* for information about database block size.

Note: To upgrade from a version older than 2009.1, you must first upgrade to 2009.1 or later, then upgrade to the current release. Use the procedure for converting 2K block size databases if applicable. For assistance with upgrading from earlier versions, contact the InterSystems Worldwide Response Center (WRC).

4.2 Upgrading to a Maintenance Release

Most of the information in this chapter applies to upgrading Caché to a new major version. If you are upgrading to a maintenance release of the installed version of Caché, review the following:

- The list of tasks at the beginning of Upgrade Tasks (only some will apply).
- If you are upgrading a Caché mirror, Upgrading Mirroring with Minimum Downtime.
- Maintenance Release Post-Installation Tasks.

4.3 Performing Upgrade Tasks

The following upgrade tasks are necessary regardless of the platform on which you are upgrading and running Caché. Perform these tasks before you run the Caché installation procedures:

1. *Obtain an updated license key* — the key structure changed in Caché 5.1; upgrades from Caché 5.0 or earlier require an updated key.
2. *Back up the system* — before upgrading, InterSystems recommends that you run a complete backup of your system. Use your customary full operating system backup procedures.
3. *Check system integrity* — run a system integrity check on existing directories to ensure there is no corruption in any of the databases.
4. *Save custom classes, routines and globals* — on an upgrade, the CACHESYS database is modified. To prevent your own classes, routines and globals in the %SYS namespace from being affected by the upgrade installation, ensure that

they have names that begin with "Z", "z", "%Z", or "%z". All .int and .obj routines (except for Z*, z*, %Z*, and %z*) are deleted from the %SYS namespace when upgrading.

Similarly, on an upgrade, the CACHELIB, CACHETEMP, DOCBOOK, and SAMPLES databases are completely replaced.

5. *Save user files* — to ensure that your user files are not deleted or replaced during upgrades, save them in the *install-dir*\Mgr\User directory, which is the only directory that is not subject to modification during upgrades (that is, except for the *install-dir*\Mgr\User directory, any files and directories in the Caché install directory may be deleted or replaced during upgrades).
6. *Review encryption settings* — because of a limitation in prior releases, if you are upgrading from release 2010.1 or earlier and have encryption set to interactively ask for the username, password, and key file, you should temporarily change this setting to load encryption information automatically before upgrading (see Configuring Startup with Unattended Key Activation in the "Managed Key Encryption" chapter of the *Caché Security Administration Guide* for more information). After you perform the upgrade, you should then reenable the interactive setting. (Failure to configure startup with unattended key activation may cause the upgrade to fail; if this happens, start the system normally, change the startup configuration to unattended mode, and then reinstall the new version.)

   If you are upgrading using an .rpm file, you must disable database encryption before upgrading and reenable it when the upgrade is complete.

In addition, there are other tasks that may be necessary depending on your environment and the components as described the following sections:

- Upgrading a Cluster
- Upgrading Mirroring with Minimum Downtime
- Upgrading ECP Configurations

4.3.1 Upgrading a Cluster

This section describes general considerations to take into account when upgrading Caché in a Caché failover cluster.

Note: The information in this section does not apply to systems in which ECP application servers are deployed. For information about upgrading ECP configurations, see Upgrading ECP Configurations.

Important: At a general level, it is important to:

- Ensure that the shared-disk resources are not impacted or taken offline during a Caché upgrade.
- Prevent cluster failover while Caché is being upgraded in a failover cluster.
- Prevent Caché from starting automatically while being upgraded in a failover cluster.

In Windows, for example, to prevent cluster failover/automatic Caché startup, always tale the Caché cluster resource offline prior to the upgrade; when ready to resume operation, bring the resource back online. Similar steps should be followed on all other platforms; refer to the failover software product documentation that is relevant to your installation.

For platform-specific information, see the following:

- Upgrading Caché on Windows Failover Clusters
- Upgrading Caché on UNIX®, Linux, and macOS Failover Clusters

4.3.1.1 Upgrading Caché on Windows Failover Clusters

On Microsoft Windows clusters, use the following procedure to upgrade Caché.

Important: Do not start and stop Caché from the Caché Launcher. Instead, using Failover Cluster Management, take the Caché cluster resource offline to stop Caché, and bring the Caché cluster resource online to start Caché.

It is important to upgrade Caché on both systems in a Windows cluster before resuming cluster operation to ensure that the correct registry settings are maintained. For this reason, you cannot upgrade a Windows cluster without making the cluster unavailable for a period of time.

1. After ensuring that a cluster failover will not occur automatically, shut down on the currently running (primary) system (for example, System A) by taking the Caché cluster resource offline. This step is recommended to quiesce the system prior to an upgrade.
2. Upgrade Caché on System A. While the upgrade process will start Caché on System A after the upgrade is successful, it is recommended that users (direct-connect, background processes, ECP application servers, etc.) not be allowed back on to the system until the entire upgrade process (cluster-wide) is complete.
3. Shut down Caché on System A by taking the Caché cluster resource offline.
4. Fail the cluster over to the secondary system (for example, System B).
5. Upgrade Caché on System B. While the upgrade process will start Caché on System B after the upgrade is successful, it is recommended that users (direct-connect, background processes, ECP application servers, etc.) not be allowed back on to the system until the entire upgrade process (cluster-wide) is complete.
6. Fail the cluster over to System A.
7. Start Caché on System A by bringing the Caché cluster resource online.
8. Ensure that the cluster resource type (for example, **Caché ISCCres2003**) is brought online so that automatic failover can occur. Users (direct-connect, background processes, ECP application servers, etc.) can be permitted to reconnect to Caché (on System A) at this point.

4.3.1.2 Upgrading Caché on UNIX®, Linux, and macOS Failover Clusters

On UNIX/Linux systems clusters, use the following procedure to upgrade Caché:

1. Gather the list of Caché instances on the two failover nodes. This can be done by using the **ccontrol list** command. Note the list of systems (and paths, etc.), if any, that are unique to the second (standby) failover node.
2. After ensuring that a cluster failover will not occur automatically, shut down Caché on the currently running (primary) system (for example, System A).
3. Upgrade Caché on System A. The upgrade process should start Caché once the upgrade is complete. Users (direct-connect, background processes, ECP application servers, etc.) can be permitted to reconnect to Caché (on System A) at this point.
4. Copy the /usr/local/etc/cachesys/ directory from System A to System B. Note: please ensure that the permissions and ownership for the cachesys directory as well as the files within that directory remain intact.
5. If there are any Caché instances that were unique to System B, you must recreate that information manually using the **ccontrol create** command to add the instance information to the registry. The correct usage of the command can be found by typing **ccontrol create help** at the OS prompt.
6. Ensure that the appropriate cluster resource associated with Caché is brought back online so that automatic failover can occur.

4.3.2 Upgrading Mirroring with Minimum Downtime

This section provides instructions for upgrading Caché or Ensemble instances, an application, or both on the members of a Caché mirror.

As noted in the "Mirroring" chapter of the *Caché High Availability Guide*, all failover and DR async members of a mirror must be of the same Caché version, and can differ only for the duration of an upgrade. Once an upgraded member becomes primary, you cannot make use of the other failover member and any DR async members (and in particular cannot allow them to become the primary) until the upgrade is completed.

Mirroring does not require reporting async members to be of the same Caché version as the failover members, although application functionality may require it; see the "Supported Version Interoperability" chapter of the online *InterSystems Supported Platforms* document for this release for information about reporting async version requirements.

There are four mirror upgrade procedures, as follows. The first three procedures are designed to let you perform the upgrade you need with the shortest possible application downtime; they differ to accommodate different upgrade circumstances. The last, which is a bit simpler, can be used for any type of upgrade when minimizing downtime is not a priority.

- Upgrading to Caché Maintenance Release
- Upgrading to Major Caché Version with Mirrored Database Changes
- Upgrading to Major Caché Version with No Mirrored Database Changes
- Upgrading to Major Caché Version with Planned Downtime

Choosing Mirror Upgrade Procedure provides an overview of the factors that determine which procedure you should use. Your circumstances may call for additional steps and details in a given procedure. If you are unsure which procedure is appropriate for you or whether a particular procedure is appropriate as stated, please contact the InterSystems Worldwide Response Center (WRC).

To avoid repetition, Minimum Downtime Upgrade Terms defines the terms that are used in multiple procedures.

Important: On Linux systems supporting **systemd**, although it is possible to use either **systemctl** commands or the /etc/init.d/ISCAgent script to manage the ISCAgent, you must choose one method and use it exclusively; the ISCAgent should always be stopped using the method with which it was started (see Starting the ISCAgent on Linux Systems in the "Mirroring" chapter of the *Caché High Availability Guide*).

When you upgrade Caché on such a Linux system, a running ISCAgent is automatically restarted using **systemd**. If you are using the /etc/init.d/ISCAgent script to manage the ISCAgent, stop the agent before performing the upgrade so that it is not automatically restarted, then restart it using the script after the upgrade.

4.3.2.1 Choosing Mirror Upgrade Procedure

There are two primary factors to consider in choosing the procedure appropriate for your mirror upgrade:

- Are you upgrading to a maintenance release of the installed version of Caché (for example, from 2015.1.0 to 2015.1.1), or to a new major version of Caché (such as 2014.1.0 to 2015.1.0)?
- Does the upgrade involve changes to mirrored databases?

Whenever you upgrade to a maintenance release and are not making any application upgrades, you do not have to consider the second question; you can always use the simple Maintenance Release Upgrade procedure, which renders the mirror unavailable only for the time it takes to execute a planned failover.

However, when you upgrade from one major Caché version to another, perform application upgrades, or both, there is the possibility that your upgrade will involve changes to mirrored databases. Such changes *must occur on the functioning primary*

failover member as part of the upgrade procedure, while application access is disabled. As a result, the upgrade requires more downtime than if you are not making any mirrored database changes. (The changes will then be replicated by the mirror on the backup and any async members.)

Following a major upgrade, InterSystems recommends that you recompile all classes in all application namespaces on the instance, including Ensemble namespaces, and some routines must also be recompiled (see Upgrade Specifics in the chapter General Upgrade Information in the *Caché Release Notes and Upgrade Checklist*). Any application upgrade you perform may require changes to application data. In either of these situations, your upgrade may result in changes to mirrored databases. Therefore, if you are upgrading to a major release, are performing application upgrades, or both, you must determine whether any of the following conditions apply:

- Classes and routines are stored in mirrored databases that also contain application data; these must be recompiled on the primary (even if the application has not been upgraded).
- Data in mirrored databases must be upgraded due to an application upgrade; these changes must take place on the primary.
- Mirrored databases are used with Ensemble or DeepSee; all classes in all namespaces mapped to these databases must be recompiled on the primary.

If your major upgrade and/or application upgrade involves any of these conditions, use the Major Upgrade (Mirrored Database Changes) procedure, which renders the mirror unavailable only for the time it takes to execute a planned failover and make the required mirrored database changes.

If your upgrade does *not* involve any of the listed mirrored database changes, consider the Major Upgrade (No Mirrored Database Changes) procedure, which renders the mirror unavailable only for the time it takes to execute a planned failover.

If you have a significant planned maintenance window and minimizing mirror downtime is not a major concern, you may choose to use the simpler Major Upgrade with Planned Downtime procedure instead.

In the case of major upgrades, the general sequence of each procedure applies to upgrades of the mirror's Caché instances, application code, or both; adapt individual steps depending on what you are upgrading.

4.3.2.2 Adding Mirror Members During an Upgrade

If you plan to add members to a mirror, you may want to defer this until you plan to perform one of the upgrades described in this section and add members of the new version during the upgrade, so that you will not have to upgrade them later. You can always add members of a newer version to a mirror during an upgrade, provided you immediately continue with and complete the upgrade as described, with one restriction: once a member of the newer version becomes primary, it must remain primary until all other members have been upgraded.

Adding new members during an upgrade can be very helpful when migrating a mirror to new hardware. Rather than upgrading one of the failover members before failing over to it, you can install a new instance of the new version on the target system, add it to the mirror as a DR async member, promote it to backup and then fail over to it, thus migrating the mirror primary to the new system. By repeating this technique you can then migrate the remaining failover member. The procedures in this section include steps for adding a new backup of the new version as an alternative to upgrading the existing backup, to illustrate this approach.

4.3.2.3 Minimum Downtime Upgrade Terms

In the procedures in this section, the following terms are used:

- *Upgrade classes and routines* refers to the recompilation of all classes in all application namespaces on the instance after a Caché major version upgrade (as described in Post-Installation Upgrade Tasks) and/or application upgrade, which may involve one or more of the following operations, depending on your situation:
  - As noted in the foregoing, when classes and routines exist in any mirrored database that also contains application data, they must be recompiled locally on the functioning primary failover member (regardless of whether they are modified by an application upgrade).

  - Classes and routines stored in non-mirrored routine databases that are separate from application data can be recompiled on a mirror member regardless of whether it is the current primary.
  - Classes and routines stored in separate non-mirrored routine database can also be "precompiled" by recompiling a copy of the database on a system that has already been upgraded to the target Caché release, then distributed to each mirror member following the upgrade.

  The methods you use to upgrade classes and routines will depend on your circumstances, the procedure you are using, and the member on which you are recompiling.

- The name *system A* refers to the mirror member that is *initially* the primary failover member, and *system B* to the mirror member that is *initially* the backup failover member. *System C* refers to a newly-added DR async of the new Caché version that has been promoted to failover member.
- *Set no failover* refers to the use of the **^MIRROR** routine to set the no failover state so that failover cannot occur; see Avoiding Unwanted Failover During Maintenance of Failover Members in the "Mirroring" chapter of the *Caché High Availability Guide* for instructions.
- *Perform a graceful shutdown* refers to the use of the **ccontrol stop** command (see Controlling Caché Instances in the "Using Multiple Instances of Caché" chapter of the *Caché System Administration Guide*).
- *Promote to failover member* refers to the procedure for promoting a DR async mirror member to failover member as described in Promoting a DR Async Member to Failover Member in the "Mirroring" chapter of the *Caché High Availability Guide*.
- *Viewing the Mirror Monitor* refers to using the Mirror Monitor page in the instance's Management Portal to review the status of the mirror and its members; see Using the Mirror Monitor in the "Mirroring" chapter of the *Caché High Availability Guide* for more information.

4.3.2.4 Upgrading to Caché Maintenance Release

If you are upgrading to a Caché maintenance release rather than a new major Caché version and are not making any application changes, use the following procedure, which renders the mirror unavailable only for the time required to execute a planned failover:

1. To prevent failover from occurring until the backup is fully upgraded, set no failover.
2. Perform one of the following two operations:
   - Upgrade the existing backup:
     a. Perform a graceful shutdown of the backup (B).
     b. Upgrade the backup (B) with the new version of Caché. System B becomes backup.
   - Add a new backup of the new version:
     a. Install the new version of Caché on system C.
     b. Add system C to the mirror as a DR async member.
     c. Promote system C to failover member. System C becomes backup and System B becomes a DR async.
3. Ensure that the backup (B or C) has become active by viewing the Mirror Monitor.
4. Clear no failover.
5. Perform a graceful shutdown of the primary (A). The mirror fails over and the backup (B or C) takes over as primary.
6. Upgrade system A with the new version of Caché. System A becomes backup.

7. If system C became primary and system B was demoted to DR async, upgrade system B.

4.3.2.5 Upgrading to Major Caché Version with Mirrored Database Changes

If you are upgrading to a new major Caché version and/or performing an application upgrade and have determined that changes to mirrored databases are required (as described in Choosing a Mirror Upgrade Procedure), use the following procedure, which renders the mirror unavailable only for the time it takes to execute a planned failover and make the required mirrored database changes:

1. To prevent failover from occurring until the backup is fully upgraded, set no failover.
2. Perform one of the following two operations:
   - Upgrade the existing backup:
     a. Perform a graceful shutdown of the backup (B).
     b. Upgrade the backup (B) with the new version of Caché. System B becomes backup.
     c. Upgrade non-mirrored classes and routines on system B, if any.
   - Add a new backup of the new version:
     a. Install the new version of Caché on system C.
     b. Add system C to the mirror as a DR async member.
     c. Promote system C to failover member. System C becomes backup and System B becomes a DR async.
3. Ensure that the backup (B or C) has become active by viewing the Mirror Monitor.
4. Disallow new user access to the mirror using established practices at your site. Additionally, you must disable any application jobs that would normally start on system B when it becomes primary. If using Ensemble, disable production auto-start.
5. Clear no failover.
6. Perform a graceful shutdown of the primary (A). The mirror fails over and the backup (B or C) takes over as primary
7. Upgrade mirrored classes and routines on the new primary (B or C). If an application upgrade requires further changes to mirrored databases, make these changes.
8. Restore user access to the mirror. If using Ensemble, enable production auto-start and start productions.
9. To prevent system A from taking over as primary until fully upgraded, set no failover
10. Upgrade system A with the new version of Caché. System A becomes backup.
11. Upgrade non-mirrored classes and routines on system A, if any.
12. Clear no failover.
13. If system C became primary and system B was demoted to DR async, upgrade system B and upgrade non-mirrored classes and routines on system B, if any.

4.3.2.6 Upgrading to Major Caché Version with No Mirrored Database Changes

If you are upgrading to a new major Caché version and/or performing an application upgrade and have determined that changes to mirrored databases are *not* required (as described in Choosing a Mirror Upgrade Procedure), you may be able to use the following procedure.

This procedure is simplest for static applications where separate non-mirrored routine databases are always maintained. It can, however, be used even if the separate routine databases are normally mirrored by removing these databases from the mirror for the duration of the upgrade.

Important: If the routine databases are normally mirrored, ensure that they do not contain any application data before removing them from the mirror.

This procedure cannot be used with normally mirrored routine databases if ECP application servers connect to the mirror.

1. Enact a code freeze and application configuration freeze to disallow application changes during the upgrade, using established procedures at your site, to ensure that routine databases are not modified during the upgrade.
2. To prevent failover from occurring until the backup is fully upgraded, set no failover.
3. Perform one of the following two operations:
   - Upgrade the existing backup:
     a. If the routine databases are mirrored, remove them from the mirror on system B.
     b. Ensure that the backup (B) has become active by viewing the Mirror Monitor.
     c. Perform a graceful shutdown of the backup (B).
     d. Upgrade the backup (B) with the new version of Caché. System B becomes backup.
     e. Upgrade classes and routines on system B.
   - Add a new backup of the new version:
     a. Install the new version of Caché on system C.
     b. Add system C to the mirror as a DR async member.
     c. Promote system C to failover member. System C becomes backup and System B becomes a DR async.
     d. If the routine databases are mirrored, remove them from the mirror on systems C and B.
     e. Ensure that the backup (C) has become active by viewing the Mirror Monitor.
4. Clear no failover.
5. Perform a graceful shutdown of the primary (A). The mirror fails over and the backup (B or C) takes over as primary.
6. To prevent system A from taking over as primary until fully upgraded, set no failover,
7. Upgrade system A with the new version of Caché. System A becomes backup.
8. Upgrade any non-mirrored classes and routines on system A.
9. If system C became primary and system B was demoted to DR async, upgrade system B and upgrade non-mirrored classes and routines on system B, if any.
10. For any mirrored routine databases that you removed from the mirror on backup (B or C) before it became the new primary, do the following:
    a. Remove the routine databases from the mirror on system A.
    b. Add the databases to the mirror on the new primary (B or C), then back them up and restore them on the backup (A) and DR async (B, if C is the primary), using the procedures for adding an existing database to a mirror provided in Adding Databases to a Mirror in the "Mirror" chapter of the *Caché High Availability Guide.* Retain these backups, as they represent the first backups of newly-added mirrored databases; older backups made prior to the upgrade cannot be used to restore these databases in the event of disaster.

11. Clear no failover.

4.3.2.7 Upgrading to Major Caché Version with Planned Downtime

If you are upgrading to a new major Caché version and/or performing an application upgrade and have a significant planned maintenance window that obviates the need to minimize mirror downtime, you may prefer to use the following simpler procedure, regardless of whether changes to mirrored databases on the primary are required:

1. Disallow all user access to the mirror using established practices at your site. Additionally, you must disable any application jobs that would normally start on instance startup. If using Ensemble, disable production auto-start.
2. Perform a graceful shutdown of the backup (B).
3. Perform a graceful shutdown of the primary (A).
4. Upgrade system A with the new version of Caché. System A becomes primary.
5. Upgrade mirrored classes and routines on system A. If an application upgrade requires further changes to mirrored databases, make these changes.
6. Upgrade non-mirrored classes and routines on system A, if any.
7. Perform one of the following two operations:
   - Upgrade the existing backup:
     a. Upgrade system B with the new version of Caché. System B becomes backup.
     b. Upgrade non-mirrored classes and routines on system B, if any.
   - Add a new backup of the new version:
     a. Install the new version of Caché on system C.
     b. Add system C to the mirror as a DR async member.
     c. Promote system C to failover member. System C becomes backup.
     d. Upgrade system B with the new version of Caché. System B becomes a DR async.
     e. Upgrade non-mirrored classes and routines on system B, if any.
8. If system C became primary and system B was demoted to DR async, upgrade system B and upgrade non-mirrored classes and routines on system B, if any.
9. Restore user access to the mirror.

4.3.3 Upgrading ECP Configurations

In general, ECP application servers should be upgraded before the data servers they connect to. Application servers must have either local routines databases to recompile following upgrade or access to “precompiled” routines database (previously recompiled on a separate system running the target Caché version) on the data server.

Downtime can be minimized using rolling upgrades with users connecting to both upgraded and unupgraded application servers as well as the database server if one of the following is true:

- There are no application changes.
- There are application changes, but the new code does not generate any new data structures, meaning that old code can work with data generated by new code.

If the new application code can work against old data, but can generate new data structures not understood by the old code, follow this procedure:

1. Upgrade the application servers on a rolling basis, but do not allow users to connect to an application server once it is upgraded. (Application server capacity will be gradually reduced.)
2. When enough application servers have been upgraded, restore user access to the upgraded application servers and end all user connections to the remaining (not upgraded) application servers and the data server (if need be), ensuring that users have no access to these systems until you enable it.
3. Upgrade the remaining application servers, restoring user access to each application server after it is upgraded.
4. Upgrade the data server and restore user access as needed. (Upgrading the data server will cause a pause in application activity, the length of which will depend on the amount of downtime involved in the upgrade.)

If you have questions or concerns about how to upgrade your ECP configuration, please contact InterSystems Worldwide Customer Support (WRC).

4.4 Post-installation Upgrade Tasks

The post-installation tasks required depend on whether you have upgraded to a new major version of Caché or to a maintenance release of the installed version.

4.4.1 Major Version Post-Installation Tasks

After upgrading Caché to a new major version, for example, from 2014.1.0 to 2015.1.0, or 2016.1 to 2016.2, you must perform the following tasks (if you have not already done so as part of one of the previous procedures in this chapter):

- *Recompile Classes and Routines* — InterSystems recommends that customers recompile all classes contained in each namespace. Some routines also require recompilation. You may have your own tools and procedures for this purpose, taking into account any dependencies and other needs particular to the instance that was upgraded. If you do not, and are upgrading to version 2014.1 or later, see Upgrade Specifics in the chapter General Upgrade Information in the *Caché Release Notes and Upgrade Checklist* for full information and instructions. (If you are upgrading to an earlier version, see the "Pre-2014.1 Upgrade Information" appendix in *Caché Release Note and Upgrade Checklist Archive* information and instructions.)
- *Regenerate Proxy Classes* — You must regenerate any proxy classes you had generated in the upgraded instance by following the instructions in the appropriate guides in the *Caché Language Bindings* set. For more information, see Upgrade Specifics in the chapter General Upgrade Information in the *Caché Release Notes and Upgrade Checklist*.

  This item does not apply to web services and web clients; it is not necessary to reimport any Web Service Definition (WSDL) files.
- *Clear browser cache* — Your browser cache may contain JavaScript files that are no longer compatible with the installed version of Caché and will cause errors; clear your browser cache immediately after completing the upgrade.

The following tasks may be necessary depending on your environment and the components you use:

- *Update web server files* — If you are using Zen, recompilation generates external files; in addition, you may have your own external Zen and CSP files in place. Following upgrades you must deploy these using established practices at your site.
- *Upgrade CSP Gateway* — If your CSP Gateway is on a separate machine from the Caché server you are upgrading, you must also upgrade the CSP Gateway on that separate machine. You accomplish this by performing a custom Caché install (see the CSP Gateway Installation section of the "Installing Caché on Microsoft Windows" chapter of this

book) on your web server machine and choosing only to install the CSP Gateway. See the "Connecting to Remote Servers" chapter of the *Caché System Administration Guide* for details.

- *Upgrade Studio Clients* — The Studio version on a client must be the same or later than the Caché server version to which it connects; this also applies to maintenance releases. See the "Introduction to Studio" chapter of *Using Studio* for details.

4.4.2 Maintenance Release Post-Installation Tasks

After upgrading to a maintenance release of the installed version of Caché—for example, from 2015.1.0 to 2015.1.1—you are not required to recompile any classes or routines. However, if you choose to recompile, it is important to follow the guidance provided in Upgrade Specifics in the chapter General Upgrade Information in the *Caché Release Notes and Upgrade Checklist*.

Typically, no changes to external files or clients are required following a maintenance release upgrade.

5

Creating and Using an Installation Manifest

This chapter describes how to use the %Installer utility to create an installation manifest describing a specific Caché configuration and use it to generate code to configure a Caché instance. The chapter discusses the following topics:

5.1 Overview of an Installation Manifest

The %Installer utility lets you define an installation manifest that describes a specific Caché configuration, rather than a step-by-step installation process. To do so, you create a class that contains an XData block describing the configuration you want, using variables that contain information usually provided during installation (superserver port, operating system, and so on). You also include in the class a method that uses the XData block to generate code to configure the instance. (You can copy and paste this method from this book.)

Once you have defined the manifest, you can call it during installation, from a Terminal session, or from code. The manifest must be run in the %SYS namespace.

5.2 Creating a Class That Defines a Manifest

To create a class that defines an installation manifest, create a class as follows:

- It must include the %occInclude include file.
- It must contain an XData block specifying the details of the installation.

  You can specify any legal name for the XData block; you use this name later as an argument.

  If you use `[XMLNamespace = INSTALLER]` after the name of the XData block, Studio provides assistance as you type the XData block.

  The root element in the XData block must be `<Manifest>`. For details, see "<Manifest> Tags".
- It must define the **setup()** method, as shown in the following example.
- The **setup()** method must refer to the XData block by name as shown in the following example.

The following is an example:

```
Include %occInclude
Class MyPackage.MyInstaller
{
XData MyInstall [ XMLNamespace = INSTALLER ]
   {
   <Manifest>
      <Var/>
      <If/>
      <Log/>
      <User/>
      <Resource/>
      <Role/>
      <SystemSetting/>
      <ForEach>
         <!--Code for each iteration of named Index key while looping through values-->
      </ForEach>
      <CopyDir/>
      <CopyFile/>
      <Namespace>
         <Configuration>
            <Database/>
            <ForEach>
               <!--Code for each iteration of named Index key while looping through values-->
            </ForEach>
            <Log/>
            <GlobalMapping/>
            <RoutineMapping/>
            <ClassMapping/>
         </Configuration>
         <Compile/>
         <CopyClass/>
         <CSPApplication/>
         <ForEach>
            <!--Code for each iteration of named Index key while looping through values-->
         </ForEach>
         <If/>
         <Import/>
         <Invoke/>
      </Namespace>
   </Manifest>
   }

ClassMethod setup(ByRef pVars, pLogLevel As %Integer = 3,
   pInstaller As %Installer.Installer,
   pLogger As %Installer.AbstractLogger)
   As %Status [ CodeMode = objectgenerator, Internal ]
   {
   #; Let XGL document generate code for this method.
   Quit ##class(%Installer.Manifest).%Generate(%compiledclass,
      %code, "MyInstall")
   }
}
```

5.3 Key Options

The outermost XML tag, <Manifest>, contains all the information for code generation. There can be as many <Namespace> tags as needed within the <Manifest> tag to define namespaces in the manifest.

A <Configuration> tag, which is located inside a <Namespace> tag, is required only if databases and/or mappings are being defined. There must be as many <Database> tags within the <Configuration> tag as are needed to define the namespace. In addition, following each <Database> tag, you can add <GlobalMapping>, <RoutineMapping>, and <ClassMapping> tags as required. The </Configuration> tag activates the defined mappings.

Within the context of a <Namespace> tag — after the databases and their mappings have been defined — globals can be loaded, and routines and classes can be loaded and compiled using <Import> tag. Class methods can be invoked using the <Invoke> tag. Invoked class methods can execute routines and access globals that have been imported.

Optionally, you can define variable pairs with the <Var> tag; each variable must specify a name and value. When the value for the <Var> is needed, the name is referenced by the ${NameAssigned} syntax.

For an example, see "Example" later in this chapter.

5.4 Adding Users and Passwords

There are several ways to add users (including their roles and passwords) to the installed instance, as follows:

- Using the Management Portal after installation is complete, as described in the *Caché Security Administration Guide*.
- Using the <User> tag in the manifest, as described in in <Manifest> Tags.

  The *PasswordVar* parameter of the <User> tag specifies the variable containing the password for the user; for example, by defining PasswordVar="Pwd", you are specifying that the value of the variable *Pwd* is the password for a user. There are a variety of ways to populate this variable, but it is ultimately up to you to do this. You might consider using a remote method invocation to another instance of Caché or a web service; the problem with these approaches is that the server that is installing Caché may need internet access. Other possibilities include importing the method you are using to the instance you are installing, or adding a client-side form to the install that prompts for users and passwords, which can be passed to your manifest.
- Using the Security.Users class on a staging instance of Caché or Ensemble, as follows:

  1. Export the user information by using the **Security.Users.Export()** method.
  2. Import the user information by adding the following at the beginning of your manifest class (in the %SYS namespace):

     ```
     <Invoke Class="Security.Users" Method="Import" CheckStatus="true">
     <Arg Value="PathToExportedUserInformation"/>
     </Invoke>
     ```

     where *PathToExportedUserInformation* is the location of the output file specified in the **Security.Users.Export()** method.

5.5 Adding Messages

You can define messages to be added to the log by incorporating <Log> tags in your class, in the following format:

```
<Log Level="<level>" Text="<text>"/>
```

The log level must be in the range of 0 to 3, where 0 is “none” and 3 is “verbose”; if the log level specified in the **setup()** method is equal to or greater than the Level property in the <Log> tag, the message is logged. The text is limited to 32,000 characters.

You set the log level via the second argument of the **setup()** method (for more information, see “Using the Manifest” later in this chapter). Since this cannot be passed to the manifest from the install, you must set it in the class as follows:

```
ClassMethod setup(ByRef pVars, pLogLevel As %Integer = 3,
   pInstaller As %Installer.Installer,
   pLogger As %Installer.AbstractLogger)
   As %Status [ CodeMode = objectgenerator, Internal ]
```

You can direct where messages are displayed via the fourth argument (**%Installer.AbstractLogger**) of the **setup()** method; for example, if you instantiate a **%Installer.FileLogger** object referencing an operating system file, all %Installer and log messages are directed to that file. (Without this argument, all messages are written to the primary device if **setup()** is called directly, and are ignored if it is executed by the installer.)

5.6 <Manifest> Tags

Following are descriptions of the tags (in alphabetical order) you can use within the XData block of your manifest class.

Some tags can contain expressions (strings) that are expanded when the manifest is executed. There are three types of expressions that can be expanded, as follows:

- `${<var_name>}`—

  Expands to the value of the specified user variable (defined with the <Var> tag, as described in this section) or predefined variable (listed in the section Variables Available within <Manifest>). Examples of this type of expression can be found within this section.

- `${#<param_name>}`

  Expands to the value of the specified parameter for the class containing the XDATA.

- `#{<COS_expression>}`

  Expands to the specified Caché Object Script expression (which must be properly quoted).

Parameter expressions are expanded at compile time, which means they can be nested within variable and COS expressions. Addtionally, variable expressions are expanded before COS expressions and thus can be nested within the latter.

Important: Null arguments cannot be passed in tags in a manifest class. The <Arg/> tag passes an empty string, equivalent to <Arg=""/>, not null.

Note: The value shown in square brackets ([]) is the default used for a property if it is not specified explicitly.

<Arg>

Parent tags: <Invoke>, <Error>

Passes an argument into a method called via <Invoke> or <Error>.

- Value—Value of an argument.

```
<Error Status="$$$NamespaceDoesNotExist">
        <Arg Value="${NAMESPACE}"/>
/>
```

<ClassMapping>

Parent tag: <Configuration>

Creates a class mapping from a database to the namespace which contains this <Configuration> item.

- Package—Package to be mapped.
- From—Source database used for mapping.

```
<ClassMapping Package="MYAPP"
     From="MYDB"/>
```

<Compile>

Parent tag: <Namespace>

Compiles the specified class name by calling **%SYSTEM.OBJ.Compile(*Class, Flags*)**.

- Class—Name of class to be compiled.
- Flags—Compilation flags [ck].
- IgnoreErrors—Continue on error? [0]

```
<Compile
    Class="MyPackage.MyClass"
    Flags="ck"
    IgnoreErrors=0/>
```

<Configuration>

Parent tag: <Namespace>

Child tags: <ClassMapping>, <Database>, <GlobalMapping>, <RoutineMapping>

Required as parent tag of configuration tags within <Namespace>. The closing tag (</Configuration>) activates the mappings for the databases in the namespace and updates the .cpf file.

No properties.

```
<Configuration>
    <Database> . . . />
    <ClassMapping> . . . />
/>
```

<CopyClass>

Parent tag: <Namespace>

Copies or moves the source class definition to the target.

- Src—Source class.
- Target—Target class.

- Replace—Overwrite target class? [0]

```
<CopyClass
    Src="MyPackage.MyClass"
    Target="NewPackage.NewClass"
    Replace="0"/>
```

<CopyDir>

Parent tag: <Manifest>

Copies the source directory to a target.

- Src—Source directory.
- Target—Target directory.
- IgnoreErrors—Continue on error? [0]

```
<CopyDir
    Src="${MGRDIR}"
    Target="F:\MyTargetDir"
    IgnoreErrors="0"/>
```

<CopyFile>

Parent tag: <Manifest>

Copies the source file to a target.

- Src—Source file.
- Target—Target file.
- IgnoreErrors—Continue on error? [0]

```
<CopyFile
    Src="${MGRDIR}\cconsole.log"
    Target="F:\${INSTANCE}_log"
    IgnoreErrors="0"/>
```

<Credential>

Parent tag: <Production>

Creates or overrides the access credentials

- Name—Name of the access credentials.
- Username—User name.
- Password—User password.
- Overwrite—Overwrite if the account already exists.

```
<Credential
    Name="Admin"
    Username="administrator"
    Password="123jUgT540!f3B$#"
    Overwrite="0"/>
```

<CSPApplication>

Parent tag: <Namespace>

Defines one or more CSP applications, as defined within the **Security.Applications** class. (Also see Editing a CSP Application: The General Tab in the *Caché Security Administration Guide* for more details on each of these fields.)

- AuthenticationMethods—Enabled authentication methods. (For supported authentication methods and the corresponding values to provide, see the AutheEnabled property in Security.Applications. For example, commonly used values are 4=`Kerberos`, 32=`password`, and 64=`unauthenticated`.
- AutoCompile—Automatic compilation (in CSP settings)? [`1`]
- CSPZENEnabled—CSP/ZEN enabled? [`1`]
- ChangePasswordPage—Path to change password page.
- CookiePath—Session cookie path.
- CustomErrorPage—Path to custom error page.
- DefaultSuperclass—Default superclass.
- DefaultTimeout—Session timeout.
- Description—Description.
- Directory—Path to CSP files.
- EventClass—Event class name.
- Grant—List of roles assigned upon logging into the system.
- GroupById—Group by ID?
- InboundWebServicesEnabled—Inbound web services enabled? [`1`]
- IsNamespaceDefault—Default application for the namespace? [`0`]
- LockCSPName—Lock CSP name? [`1`]
- LoginClass—Path to login page.
- PackageName—Package name.
- PermittedClasses—Permitted classes.
- Recurse—Recurse (serve subdirectories)? [`0`]
- Resource—Resource required to access web app.
- ServeFiles—Service files? [`1`]
  - 0—No
  - 1—Always
  - 2—Always and cached
  - 3—Use CSP security
- ServeFilesTimeout—Time, in seconds, of how long to cache static files.
- TwoFactorEnabled—Two-step authentication enabled? [`0`]
- Url—Name of the web application.

- UseSessionCookie—Use cookies for the session?

```
<CSPApplication
    Url="/csp/foo/bar"
    Description=""
    Directory="C\InterSystems\Cache\CSP\Democode1"
    Resource=""
    Grant="%DB_%DEFAULT"
    Recurse="1"
    LoginClass=""
    CookiePath="/csp/demo1"
    AuthenticationMethods="64"/>
```

<Database>

Parent tag: <Configuration>

Defines a database within a namespace.

See Configuring Databases in the "Configuring Caché" chapter of the *Caché System Administration Guide* for information about the database settings controlled by the following properties.

- BlockSize—Block size for database (4096, 8192, 16384, 32768, 65536).
- ClusterMountMode—Mount database as a part of a cluster at startup?
- Collation—Default collation for globals created in the database.
- Create—Create a new database (yes/no/overwrite)? [yes]
- Dir—Database directory.
- Encrypted—Encrypt database?
- EncryptionKeyID—ID of encryption key.
- InitialSize—Initial size of the database, in MB.
- ExpansionSize—Size in MB to expand the database by when required.
- MaximumSize—Maximum size the database can expand to.
- MountAtStartup—Mount when launching the installed instance?
- MountRequired—Require mounting of database at every instance startup?
- Name—Database name.
- PublicPermissions—The Permission value to be assigned to the Resource if it must be created. It is ignored if the Resource already exists. Read-only or read-write.
- Resource—Resource controlling access to the database.

- StreamLocation—Directory in which streams associated with the database are stored.

```
<Database Name="${DBNAME}"
    Dir="${MGRDIR}/${DBNAME}"
    Create="yes"
    Resource="${DBRESOURCE}"
    Blocksize="8192"
    ClusterMountMode="0"
    Collation="5"
    Encrypted="0"
    EncryptionKeyID=
    ExpansionSize="0"
    InitialSize="1"
    MaximumSize="0"
    MountAtStartup="0"
    MountRequired="0"
    StreamLocation=
    PublicPermissions=""/>
<Database Name="MYAPP"
    Dir="${MYAPPDIR}/db"
    Create="no"
    Resource="${MYAPPRESOURCE}"
    Blocksize="8192"
    ClusterMountMode="0"
    Collation="5"
    Encrypted="0"
    EncryptionKeyID=
    ExpansionSize="0"
    InitialSize="1"
    MaximumSize="0"
    MountAtStartup="0"
    MountRequired="0"
    StreamLocation=
    PublicPermissions=""/>
```

<Default>

Parent tag: <Manifest>

Sets the variable value if it is not already set.

- Name—Variable name.
- Value—Variable value.
- Dir—Variable value, if a path to a folder or file

```
<Default Name="blksiz"
    Value="8192" />
```

<Else>

Parent tags: <If>

Executed when the conditional defined by the preceding <If> statement is false.

No properties.

```
<If Condition='#
{##class(%File).Exists(INSTALL_"mgr\cache.key")}
'>
<Else/>
<CopyFile Src="C:\\InterSystems\key_files\cache300.key" Target="${MGRDIR}\cache.key"
IgnoreErrors="0"/>
</If>
```

<Error>

Parent tag: <Manifest>

Child tag: <arg>

Generates an error.

- Status: Error code.
- Source: Source of the error.

```
<Error Status="$$$NamespaceDoesNotExist" Source=>
    <Arg Value="${NAMESPACE}"/>
</Error>
```

<ForEach>

Parent tag: <Manifest>

Defines values for iterations of the specified index key.

- Index—Variable name.
- Values—List of variable values.

```
<ForEach
    Index="TargetNameSpace"
    Values="%SYS,Samples,User">
    <!--Code for each iteration of TargetNameSpace-->
</ForEach>
```

<GlobalMapping>

Parent tag: <Configuration>

Maps a global to the current namespace.

- Global: Global name.
- From: Source database of the global.
- Collation: Global collation [Caché Standard]

```
<GlobalMapping Global="MyAppData.*"
    From="MYAPP" Collation="30"/>
<GlobalMapping Global="cspRule"
    From="MYAPP"/>
```

<If>

Parent tags: <Manifest>, <Namespace>

Child tag: <Else>

Specifies a conditional action.

- Condition: Conditional statement.

```
<If Condition='$L("${NAMESPACE}")=0'>
    <Error Status="$$$NamespaceDoesNotExist" Source=>
        <Arg Value="${NAMESPACE}"/>
    </Error>
</If>
```

<IfDef>

Parent tags: <Manifest>, <Namespace>

Specifies a conditional action if a variable has been set.

- Var: Variable name.

```
<IfDef Var="DBCreateName">
   <Database Name="${DBNAME}"
       Dir="${MGRDIR}/${DBNAME}"
       Create="yes"
       ...
</IfDef>
```

<IfNotDef>

Parent tags: <Manifest>, <Namespace>

Specifies a conditional action if a variable has not been set.

- Var: Variable name.

<Import>

Parent tag: <Namespace>

Imports files by calling **%SYSTEM.OBJ.ImportDir**(*File,Flags,Recurse*) or **%SYSTEM.OBJ.Load(*File,Flags*)**.

- File—File or folder for import.
- Flags—Compilation flags [ck].
- IgnoreErrors—Continue on error? [0].
- Recurse—Import recursively? [0].

<Invoke>

Parent tag: <Namespace>

Child tag: <arg>

Calls a class method and returns the execution result as the value of a variable.

- Class—Class name.
- Method—Method name.
- CheckStatus—Check the returned status?
- Return—Name of variable to write result to.

```
<Invoke Class="SECURITY.SSLConfigs" Method="GetCertificate" CheckStatus="1" Return="CertContents">
    <Arg Value="Cache.cer"/>
</Invoke>
```

<LoadPage>

Parent tag: <Namespace>

Loads a CSP page by calling **%SYSTEM.CSP.LoadPage**(*PageURL,Flags*) or **%SYSTEM.CSP.LoadPageDir(*DirURL,Flags*)**.

- Name—URL of CSP page.
- Dir—URL of directory containing CSP pages.
- Flags—Compilation flags [ck].
- IgnoreErrors—Continue on error? [0].

<Log>

Parent tag: <Manifest>

Writes a message to the log specified by the **setup()** method if the log level specified by the **setup()** method is greater or equal to the level property provided.

- Level—Log level, from 0 (none) to 3 (verbose).
- Text—Log message (string up to 32,000 characters in length).

See Adding Messages for more information.

<Manifest>

Parent tag: n/a (Root tag, containing all other tags.)

Child tags: <CopyDir>, <CopyFile>, <Default>, <Else>, <Error>, <ForEach>, <If>, <IfDef>, <IfNotDef>, <Log>, <Namespace>, <Resource>, <Role>, <SystemSetting>, <User>, <Var>

No properties.

```
<Manifest>
    <Namespace ... >
        <Configuration>
            <Database .../>
            <Database .../>
        </Configuration>
    </Namespace>
</Manifest>
```

<Namespace>

Parent tag: <Manifest>

Child tags: <Compile>, <Configuration>, <CopyClass>, <CSPApplication>, <Else>, <If>, <IfDef>, <IfNotDef>, <Import>, <Invoke>, <LoadPage>, <Production>

Defines a namespace.

- Name—Name of the namespace.
- Create—Create a new namespace (yes/no/overwrite)? [yes]
- Code—Database for code.
- Data—Database for data.
- Ensemble—Ensemble-enabled namespace? [0]

(Other properties are applicable to Ensemble web applications.)

```
<Namespace Name="${NAMESPACE}"
     Create="yes"
     Code="${NAMESPACE}"
     Data="${NAMESPACE}">
          <Configuration>
               <Database Name="${NAMESPACE}" . . . />
          </Configuration>
</Namespace>
```

<Production>

Parent tag: <Namespace>

Child tags: <Credential>, <Setting>

Defines an Ensemble production.

- Name—Production name.
- AutoStart—Automatically launch production? [0]

```
<Production Name="${NAMESPACE}"
     AutoStart="1" />
```

<Resource>

Parent tag: <Manifest>

Defines a resource.

- Name—Resource name.
- Description—Resource description.
- Permission—Public permissions.

```
<Resource
    Name="%accounting_user"
    Description="Accounting"
    Permission="RW"/>
```

<Role>

Parent tag: <Manifest>

Defines a role.

- Name—Role name.
- Description—Role description (cannot contain commas).
- Resources—Privileges (resource-privilege pairs) held by the role.
- RolesGranted—Roles granted by the named role.

```
<Role
    Name="%DB_USER"
    Description="Database user"
    Resources="MyResource:RW,MyResource1:RWU"
    RolesGranted= />
```

<RoutineMapping>

Parent tag: <Configuration>

Defines a routing mapping.

- Routines: Routine name.
- Type: Routine type (MAC, INT, INC, OBJ, ALL).
- From: Source database of the routine.

```
<RoutineMapping Routines="MyRoutine"
    Type="ALL" From="${NAMESPACE}" />
```

<Setting>

Parent tag: <Production>

Configures an item in the Ensemble production by calling the **Ens.Production.ApplySettings**() method.

- Item—Item name.

- Target—Setting type (Item, Host, Adapter).
- Setting—Setting name.
- Value—Value to configure for setting.

```
<Production Name="Demo.ComplexMap.SemesterProduction">
    <Setting Item="Semester_Data_FileService"
        Target="Item"
        Setting="PoolSize"
        Value="1"/>
    <Setting Item="Semester_Data_FileService"
        Target="Host"
        Setting="ComplexMap"
        Value="Demo.ComplexMap.Semester.SemesterData"/>
    <Setting Item="Semester_Data_FileService"
        Target="Adapter"
        Setting="FilePath"
        Value="C:\Practice\in\"/>
</Production>
```

<SystemSetting>

Parent tag: <Manifest>

Sets the value for a property of any Config class that implements the **Modify** method.

- Name—Class.property of the Config class.
- Value—Value to assign to property.

```
<SystemSetting
    Name="Config.Miscellaneous.EnableLongStrings"
    Value="1"/>
```

<User>

Parent tag: <Manifest>

Defines a user; if PasswordVar is included, the user's password must be provided in the specified variable name.

- Username—Username.
- PasswordVar—Name of variable containing user password.
- Roles—List of roles to which user is assigned.
- Fullname—User's full name.
- Namespace—User's startup namespace.
- Routine—User's startup routine.
- ExpirationDate—Date after which user login will be disabled.
- ChangePassword—Require user to change password on next login?
- Enabled—Is user enabled?

- Comment—Optional comment.

```
<User
    Username="Clerk1"
    PasswordVar="clerk1pw"
    Roles="Dataentry"
    Fullname="Data Entry Clerk"
    Namespace=
    Routine=
    ExpirationDate=
    ChangePassword=
    Enabled=
    Comment=""/>
```

<Var>

Parent tag: <Manifest>

Defines and sets variables that can be used in the manifest/

- Name—Variable name.
- Value—Value to assign to variable.

```
<Var Name="Namespace" Value="MUSIC"/>
<Var Name="GlobalDatabase" Value="${Namespace}G"/>
<Var Name="RoutineDatabase" Value="${Namespace}R"/>
<Var Name="AppDir" Value="C:\MyApp\${CFGNAME}"/>
<Var Name="GlobalDatabaseDir" Value="${AppDir}\${GlobalDatabase}"/>
<Var Name="RoutineDatabaseDir" Value="${AppDir}\${RoutineDatabase}"/>
<Var Name="Resource" Value="%DB_${Namespace}"/>
<Var Name="Role" Value="${Namespace}"/>
<Var Name="CSPResource" Value="CSP_${Namespace}"/>
```

For a comprehensive example of a user-defined installer class, see Sample.Installer in the SAMPLES namespace.

For an up-to-date list of tags and attributes, see the %Installer class reference documentation.

5.7 Variables Available within <Manifest>

The following table lists the predefined variables that you can use in the manifest:

Variable Name	Description
SourceDir	(Available only when the installer is run) Directory from which the installation (setup_cache.exe or cinstall) is running.
ISCUpgrade	(Available only when the installer is run) Indicates whether this is a new installation or an upgrade. This variable is either 0 (new installation) or 1 (upgrade).
CFGDIR	See *INSTALLDIR*.
CFGNAME	Instance name.
CPUCOUNT	Number of operating system CPUs.
CSPDIR	CSP directory.
HOSTNAME	Name of the host server.
HTTPPORT	Web server port.
INSTALLDIR	Directory into which Caché is installed.

Variable Name	Description
MGRDIR	Manager (mgr) directory.
PLATFORM	Operating system.
PORT	Caché superserver port.
PROCESSOR	Processor chip.
VERSION	Caché version number.

5.8 Using the Manifest

You can use the manifest as follows:

- In the %SYS namespace, enter the following command in the Terminal:

  ```
  %SYS>Do ##class(MyPackage.MyInstaller).setup()
  ```

  You can pass an array by reference to the **setup()** method where the subscript is used as a variable in the `<Manifest>` and the node value is the variable <*value*>. For example:

  ```
  %SYS>Set vars("SourceDir")="c:\myinstaller"
  %SYS>Set vars("Updated")="Yes"
  %SYS>Do ##class(MyPackage.MyInstaller).setup(.vars,3)
  ```

  In this example, the second argument (3) is the log level.

- Export the manifest class as `DefaultInstallerClass.xml` to the same directory where the Caché install (either .msi, setup_cache.exe, or cinstall) is run. It is imported into %SYS and compiled, and the **setup()** method is executed.

  Note that if you use the export technique, you cannot pass arguments directly to the **setup()** method. However, you can do the following:

  – On Microsoft Windows systems (see the chapter "Installing Caché on Microsoft Windows"), you can modify the .msi install package to pass variable name/value pairs to the **setup()** method. You can also use command-line arguments with the installer to pass the location of the exported manifest class, variable pairs, log file name, and log level, as shown in the following example:

  ```
  setup.exe INSTALLERMANIFEST="c:\MyStuff\MyInstaller.xml"
  INSTALLERMANIFESTPARAMS="SourceDir=c:\mysourcedir,Updated=Yes"
  INSTALLERMANIFESTLOGFILE="installer_log" INSTALLERMANIFESTLOGLEVEL="2"
  ```

  Note: Variable names passed using the `INSTALLERMANIFESTPARAMS` argument may contain only alphabetic and numeric characters (`A-Za-z0-9`) and underscores (_), and may not start with an underscore.

  – On UNIX®, Linux, macOS systems (see the chapter "Installing Caché on UNIX®, Linux, and macOS"), you can set environment variables to define the location of the exported manifest class, variable name/value pairs, log file name, and log level prior to running either **cinstall** or **cinstall_silent**, as shown in the following example:

  ```
  ISC_INSTALLER_MANIFEST="/MyStuff/MyInstaller.xml"
  ISC_INSTALLER_PARAMETERS="SourceDir=/mysourcedir,Updated=Yes"
  ISC_INSTALLER_LOGFILE="installer_log"
  ISC_INSTALLER_LOGLEVEL="2"
  ./cinstall
  ```

5.9 Example

The following class creates a namespace (MyNamespace) and three databases; two of the databases are defined as the default databases for globals (MyDataDB) and routines (MyRoutinesDB), while the third database (MyMappingDB) is a globals database that overrides the default database mapping for `t*` globals in the MyNamespace namespace.

Note: For a more comprehensive example, see the Sample.Installer class in the SAMPLES namespace.

Class Definition

```
Include %occInclude

/// Simple example of installation instructions (Manifest)
Class MyInstallerPackage.SimpleManifest
{

XData SimpleManifest [ XMLNamespace = INSTALLER ]
{
<Manifest>
    <Namespace Name="MyNamespace" Create="yes"
        Code="MyRoutinesDB" Data="MyDataDB">
      <Configuration>
        <Database Name="MyRoutinesDB" Create="yes"
            Dir="C:\MyInstallerDir\MyRoutinesDB"/>
        <Database Name="MyDataDB" Create="yes"
            Dir="C:\MyInstallerDir\MyDataDB"/>
        <Database Name="MyMappingDB" Create="yes"
            Dir="C:\MyInstallerDir\MyMappingDB"/>
        <GlobalMapping Global="t*" From="MyMappingDB"/>
      </Configuration>
    </Namespace>
</Manifest>
}

/// This is a method generator whose code is generated by XGL.
ClassMethod setup(ByRef pVars, pLogLevel As %Integer = 3,
   pInstaller As %Installer.Installer,
   pLogger As %Installer.AbstractLogger)
   As %Status [ CodeMode = objectgenerator, Internal ]
{
    #; Let our XGL document generate code for this method.
    Quit ##class(%Installer.Manifest).%Generate(%compiledclass,
    %code, "SimpleManifest")
}

}
```

After the class is compiled, you can invoke it in the Terminal as shown below (assuming you created the class in the USER namespace):

```
%SYS>znspace "USER"
USER>do ##class(MyInstallerPackage.SimpleManifest).setup()
```