InterSystems™
IRIS Data Platform

Installation Guide

Version 2023.1
2024-07-11

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

Installation Guide
InterSystems IRIS Data Platform Version 2023.1 2024-07-11

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

1

Preparing to Install InterSystems IRIS

InterSystems IRIS® supports multiple methods of deployment, and runs on many different platforms.

The *Installation Guide* describes the process of installing InterSystems IRIS from an installation kit. InterSystems IRIS can also be deployed in containers, as described in Running InterSystems Products in Containers, and provides several methods for automated deployment, which are described in Deploying InterSystems IRIS in *InterSystems IRIS Basics: Connecting an IDE*. If you are using one of these methods, consult the documentation listed in the latter.

Before installing, confirm that InterSystems IRIS supports your server or development platform. Review the *InterSystems Supported Platforms* document for the current release, which describes what technologies are supported for each release.

Before installing, you should consider the system resources at your disposal, make sure they will be sufficient for the application you will be developing and running, and manage them to optimize your system's performance. You can read more about this process in System Resource Planning and Management.

1.1 Installation Planning Considerations

There are a few different ways to configure a new installation of InterSystems IRIS. When the installation runs, it prompts for the following information to decide which configuration to install:

- Installation Directory
- Setup Type
- Character Width Setting
- Port Numbers (Optional)
- Security Settings

The following section can be performed before or after installing InterSystems IRIS:

- Install the VS Code - ObjectScript Development Environment

1.1.1 Installation Directory

The installation directory is the directory in which an InterSystems IRIS instance is installed. Throughout the documentation, this is referred to as *install-dir*. Once InterSystems IRIS is installed, it is impossible to change the installation directory.

There are several restrictions to what you can specify as the *install-dir*. The directory must be a fully resolved physical path, containing no symbolic links. The name of the directory can only use characters in the US ASCII character set, and cannot contain a caret (^). Also, InterSystems IRIS *cannot* be installed into a directory that is:

- a UNC (non-local) path.
- at the root level of a drive (such as C:\).
- anywhere under the \Program Files directory.

If unspecified during installation, *install-dir* uses a default value. This default varies by platform, installation type, and user choice, as shown in the following table:

Platform	Installation Type	Default Directory
Windows	attended	C:\InterSystems\Iris (or Iris*N* when multiple instances exist), unless installing user specifies otherwise.
	unattended	C:\InterSystems\Iris (or Iris*N* when multiple instances exist), unless `INSTALLDIR` property specifies otherwise.
UNIX®, Linux, macOS	attended	No default; installing user must specify. Do not choose the /home directory, any of its subdirectories, or the /usr/local/etc/irissys directory.
	unattended	No default; `ISC_PACKAGE_INSTALLDIR` parameter required.

1.1.2 Setup Type

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

- **Development** — Includes the InterSystems IRIS Database Engine (User Database, Language Gateways, Server Monitoring Tools), Studio (on Windows only), all supported language bindings, and database drivers. Select this option if you plan to use this instance to perform both client and server tasks.
- **Server** — Includes InterSystems IRIS Database Engine (User database, Language Gateways, Server monitoring tools) and Web Gateway. Select this option if you plan to use this instance as an InterSystems IRIS database server which can be accessed by InterSystems IRIS clients.
- **Custom** — Allows user to specify specific components to install or uninstall. Select this option if you plan to install or remove specific InterSystems IRIS components.

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

- **Client** — Includes Studio, ODBC and JDBC drivers, and the InterSystems IRIS Application Development components. Select this option if you plan to use this instance as a client to an InterSystems IRIS database server on this or another computer.
- **Web Server** — Includes Web Gateway (IIS, Apache 2.0, Apache 2.2). Select this option if you want to install only those parts of InterSystems IRIS that are required for the Web Gateway.

The following table identifies which components are installed for each setup type. When performing a custom installation, you may include only individual components from a group.

Table 1–1: Components Installed by Setup Type

Component Group	Components	Development	Server	Client	Web
InterSystems IRIS Database Engine (InterSystems IRIS Server)	Server Monitoring Tools User database Language Gateways Agent Service (ISCAgent) Apache FOP (Formatting Objects Processor) InterSystems IntegratedML	X	X		
InterSystems IRIS launcher (on Windows only)		X	X	X	
Studio (on Windows only)		X		X	
Database Drivers	ODBC Driver Java Database Connectivity	X		X	
InterSystems IRIS Application Development	Java Binding for InterSystems IRIS InterSystems IRIS shared library support .NET Binding for InterSystems IRIS Threaded Server Libraries Other Samples	X		X	
Web Gateway	CSP for IIS CSP for Apache 2.0.x CSP for Apache 2.2.x		X		X

1.1.2.1 Preexisting CSP Gateway

If the CSP Gateway is already installed on your system when you install the Web Gateway, the installer automatically upgrades the CSP Gateway to the Web Gateway. However, the installer creates a new copy of the CSP.ini file. To preserve the CSP.ini file from the CSP Gateway, perform the following steps:

1. Create a backup of the CSP.ini file, located in *install-dir*/CSP/bin.
2. Execute the Web Gateway installer.
3. Restore the backup of CSP.ini.

Note: Installing the Web Gateway does not close any old connections. To remove these old connections, you must manually close them from the System Status page.

1.1.3 Character Width Setting

You must choose between 8-bit or Unicode (16-bit) character width for your installation. An 8-bit instance cannot process data in 16-bit format, while a Unicode instance can process both 8-bit and 16-bit data.

If you need your instance to store and process data in a language that uses only Unicode character sets, such as Japanese, you must select Unicode. If the base character set for the locale your instance uses is based on the Latin-1 character set, ISO8859-1, you can select 8-bit, but bear in mind:

- If you select 8-bit, the data stored by your instance is portable only to 8-bit locales based on the same character set.
- If you select Unicode, the data stored by the instance is not portable to an 8-bit instance.

1.1.4 Port Numbers

A standard installation sets the following port numbers for an InterSystems IRIS instance:

- *Superserver port number* — `1972`. If taken, then `51773` or the first available subsequent number.
- *Web server port number* — `52773` or the first available subsequent number

You can assign different port numbers during a new installation by performing a **Custom** installation (see Setup Type). You cannot enter a port number greater than `65535`.

For information about setting the port numbers on an already-installed instance of InterSystems IRIS, see Set Port Numbers in the “Using Multiple Instances of InterSystems IRIS” chapter of the *System Administration Guide*.

1.1.5 Security Setting

An InterSystems IRIS installation has three initial security configurations: **Minimal**, **Normal**, or **Locked Down**. **Minimal** is only available for InterSystems IRIS installations. If you select **Normal** or **Locked Down**, the installer prompts for additional information, such as a password and account information.

For more information, see Initial InterSystems Security Settings.

1.1.6 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 extensions for Visual Studio Code enable you to use VS Code to connect to an InterSystems IRIS server and develop code in ObjectScript. An alternative to VS Code - ObjectScript is Studio, which installs alongside InterSystems IRIS on Windows-based operating systems.

1.2 Memory Planning and Management

Memory management is a very important factor in maximizing the performances of an InterSystems IRIS deployment. Read the following planning considerations:

- Minimum Disk Space Requirements
- Swappiness Recommendations
- Additional Memory Resources

1.2.1 Minimum Disk Space Requirements

The installation kit must be available, either on your computer or on a network, to perform an installation. InterSystems recommends you have at least 1.5 GB of disk space available before installing InterSystems IRIS. The installation procedure confirms that sufficient disk space is available before installing.

1.2.2 Swappiness Recommendations

On Linux platforms, the swappiness value determines how frequently your system will swap memory pages between the physical RAM and swap space. For systems with less than 64 GB of RAM, a swappiness of 5 is recommended. For systems with 64+ GB of RAM, a swappiness of 1 is recommended.

1.2.3 Additional Memory Resources

Correctly sizing system memory is critical for maximizing the performance and availability of InterSystems IRIS and your application. InterSystems IRIS memory usage is discussed in Memory Planning which provides guidelines for calculating initial memory requirements so you can determine whether your system has sufficient memory.

After installation, monitoring how InterSystems IRIS allocates memory is an important performance factor. If InterSystems IRIS is not allocating memory efficiently, you can adjust memory settings for processes, routine and database caches, and the generic heap. For information about how to perform these memory-related tasks, see Reviewing and Monitoring Memory Usage and Memory and Startup Settings.

1.3 Platform-Specific Preparations

The following sections contain platform-specific considerations. Review the following sections that apply to your intended installation platform:

- Supported Platforms and Components
- Maximum User Process Recommendations
- AIX® Platform Notes
- Red Hat Linux Platform Notes
- SUSE Linux Platform Notes
- Ubuntu Platform Notes

1.3.1 Supported Platforms and Components

For information about supported technologies for a release version of InterSystems IRIS, including supported operating systems and web servers, see the Supported Technologies chapter of the online *InterSystems Supported Platforms* document for the release.

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 *InterSystems IRIS Data Integrity Guide*.

Note: With each instance, InterSystems IRIS installs a private web server and a private Web Gateway to serve CSP pages to ensure proper operation of the Management Portal. The private web server ensures that:

- the Management Portal runs out of the box.
- development environments are provided with an out-of-the-box testing capability.

The private web server is not supported for any other purpose. You should not use the private web server for deployments of http-based applications, including CSP and SOAP over http or https; instead, you must install and deploy one of the supported web servers.

The private web server configuration is preserved through upgrades. You can disable the private web server using the WebServer CPF parameter

On Windows, the private web server Windows service name is "Web Server for *instance name*". InterSystems IRIS installs the web server into the *install-dir*\httpd directory. It is uninstalled when you uninstall the corresponding InterSystems IRIS instance.

For more information, see Using or Replacing the Private Web Server.

1.3.2 Maximum User Process Recommendations

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

1.3.3 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
- Required C/C++ Runtime Libraries
- Shared Library Environment Variable for InterSystems IRIS Shared Library Support
- Use of Raw Ethernet

1.3.3.1 I/O Pacing Parameters

AIX® implements an I/O pacing algorithm that may hinder InterSystems IRIS 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 Disk-I/O Pacing in the IBM AIX documentation.

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 apply to both JFS and Enhanced JFS (JFS2) file systems.

1.3.3.2 File System Mount Option

Different mount options may improve performance for some workloads.

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

To improve recovery speed using the IRIS.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 IRIS.WIJ file.

For information about **mount** options, see mount Command in the IBM AIX documentation.

1.3.3.3 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 InterSystems IRIS 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 vmstat Command in the IBM AIX documentation.
- For instructions on how to increase these parameters, see VMM page replacement tuning in the IBM AIX documentation.
- For a complete description of managing I/O tunable parameters, see ioo Command in the IBM AIX documentation.

1.3.3.4 AIX® Tunable Parameters

AIX® Interprocess Communication Tunable Parameters

The following table lists the tunable parameters for the IBM pSeries AIX® 5.2 operating system. None of the following listed parameters require tuning because each is dynamically adjusted as needed by the kernel. For more information, see Interprocess communication tunable parameters in the AIX documentation.

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.

1.3.3.5 Required C/C++ Runtime Libraries

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

InterSystems IRIS for AIX is compiled using the IBM XL C/C++ for AIX 16.1.0 compiler. If the system on which you are installing InterSystems IRIS does not have the corresponding version of the runtime already installed, you must install it. You can download the runtime from IBM XL C/C++ Runtime for AIX 16.1 on the IBM Support website.

1.3.3.6 Shared Library Environment Variable for InterSystems IRIS Shared Library Support

The InterSystems IRIS shared library support 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.

1.3.3.7 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.

1.3.4 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
- Using Kerberos

1.3.4.1 Shared Memory Limit

The default shared memory limit (*shmmax*) on Linux platforms is 32 MB. This value is too small for InterSystems IRIS, but it can be changed in the proc file system without a restart. The new memory limit remains in effect until you restart the Red Hat Linux system.

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 set this parameter permanently. Add a line to the/etc/sysctl.conf 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 InterSystems IRIS on a machine. Set this value to three times the number of instances of InterSystems IRIS that run simultaneously on your system.

Other parameters are sufficiently sized for an InterSystems IRIS 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*.

1.3.4.2 Locked-in Memory

On Linux platforms, if shared memory is allocated in huge pages, the pages are automatically locked in memory and no further action is required. See the Configuring Huge Pages on Linux section in this chapter for information about allocating huge pages.

If not using huge pages, you can configure InterSystems IRIS to lock the shared memory segment in memory to prevent paging. This is described in the LockSharedMemory section of the “memlock” entry in the *Configuration Parameter File Reference*.

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 *Configuration Parameter File Reference*.

1.3.4.3 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
   ```

1.3.4.4 Dirty Page Cleanup

On large memory systems (for example, 8GB or larger), when doing numerous flat-file writes (for example, InterSystems IRIS 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."

1.3.4.5 Using Kerberos

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.

1.3.5 SUSE Linux Platform Notes

This topic includes the information on the following adjustments:

- I/O Scheduler
- Shared Memory Limits
- Locked-in Memory
- Using Kerberos

1.3.5.1 I/O Scheduler

The I/O scheduler for SUSE Linux is responsible for ordering the I/O requests submitted to a storage device. On SUSE Linux 15, it may default to `BFQ` (Budget Fair Queueing) which is known to cause performance issues with InterSystems IRIS. InterSystems recommends changing this setting to `NONE`. For details on changing the I/O scheduler, see the SUSE documentation: Tuning I/O Performance.

Note: Setting the I/O scheduler to `NONE` may not be optimal for all use cases. Users should test the system's application workload after making any changes.

1.3.5.2 Shared Memory Limits

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

InterSystems IRIS uses shared memory for database buffers, global buffers, routine buffers, as well as license use. If the machine is being used only for InterSystems IRIS, InterSystems recommends setting the shared memory to approximately half the total memory. For more information, see Memory Planning in *System Resource Planning and Managment*, as well as Memory and Startup Settings and Determining License Capacity and Usage in the *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/iris/bin:/path/to/j2sdk/bin:/.
export CLASSPATH=
      $CLASSPATH:/iris/dev/java/lib/JDK18/intersystems-jdbc-3.0.0.jar.
```

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

1.3.5.3 Locked-in Memory

On Linux platforms, if shared memory is allocated in huge pages, they are automatically locked in memory and no further action is required. See Configuring Huge Pages on Linux for information about allocating huge pages.

If not using huge pages, you can configure InterSystems IRIS to lock the shared memory segment in memory to prevent paging. This is described in the LockSharedMemory section of the “memlock” entry in the *Configuration Parameter File Reference*.

Otherwise, you must increase the maximum size that may be locked into memory. See the Locked-in Memory section of the *Red Hat Linux Platform Notes* in this chapter for instructions.

1.3.5.4 Using Kerberos

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.

1.3.6 Ubuntu Platform Notes

This topic includes the information on the following adjustments:

- Semaphore Deletion Setting

1.3.6.1 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.

2

Installing InterSystems IRIS on Microsoft Windows

This chapter describes how to install InterSystems IRIS® data platform on a Microsoft Windows system. It assumes you are familiar with Windows directory structures, utilities, and commands.

Note: It is not possible to perform multiple simultaneous installations on the same machine.

2.1 Before Installing

Before beginning the installation, be sure you have read all the information that applies to Windows in the "Preparing to Install InterSystems IRIS" chapter of this guide. Additionally, review the following topics:

- Windows User Account
- Choosing a Configuration Strategy

2.1.1 Windows User Accounts

When installing InterSystems IRIS, you must choose the Windows user account to run the InterSystems service, *InterSystems IRIS Controller for <instance-name>*. There are two options:

- The default SYSTEM account (Windows Local System account). This is used in **Minimal** security installations.
- A defined Windows user account.

Running the InterSystems 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 an InterSystems IRIS instance—for example due to your network configuration or security arrangements—specify a Windows account for the InterSystems service that has the needed privileges and access, such as a domain administrator account.

For instructions on how to change the service account after installation, see Managing Access to the InterSystems IRIS Instance.

Important: If you are using Kerberos, you must configure a Windows account before installing InterSystems IRIS. InterSystems recommends you use a separate account specifically set up for this purpose as described in Preparing the Security Environment for Kerberos.

Note: If running InterSystems IRIS under a defined Windows user account, be sure to grant that account the Windows "Lock Pages in Memory" (SELockMemory) privilege. See Configuring Large Pages on Windows in the "Preparing to Install InterSystems IRIS" chapter for details.

2.1.2 Choosing a Configuration Strategy

You can define an installation manifest that describes a specific InterSystems IRIS configuration and invoke it when executing the installation file. This is useful when considering automated deployment, as it reduces the need to adjust settings after the installation is complete.

For more information on installation manifests, see Creating and Using an Installation Manifest.

2.2 InterSystems IRIS Installation Procedure

This section contains the procedure to install InterSystems IRIS on a Windows computer. After the initial steps, the procedure diverges depending on which components you would like to install. The differences are described in subsections beneath the main procedure.

To perform an InterSystems IRIS installation:

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

   ```
   C:\Users\Public\Downloads\IRIS-2018.1.0.508.0-win_x64.exe
   ```

   Note: By default, a newly installed InterSystems IRIS instance starts immediately after installation and the InterSystems IRIS launcher is placed in the system tray. You can prevent these actions by setting ISCSTARTIRIS and ISCSTARTLAUNCHER each to `0` using the command line. For example:

   ```
   C:\Users\Public\Downloads\IRIS-<version_number>-win_x64.exe ISCSTARTIRIS=0 ISCSTARTLAUNCHER=0
   ```

   The Command Line Properties table in the *Unattended Installation Procedure* section of this chapter contains a description of all InterSystems IRIS Windows installer properties.

3. The InterSystems IRIS setup begins in the language specified by the Windows **language for non-Unicode programs** setting. Within setup, you can click **Next** to continue to the next dialog box, **Back** to return to the previous dialog box, and **Cancel** to stop the installation.
4. If there is an existing InterSystems IRIS instance installed on the system, the **Select Instance** dialog box lists its installation directories. Select **New Instance** to install a new InterSystems IRIS instance.

   Note: Select an existing instance to reinstall or upgrade that instance.

5. If you are installing a new instance of InterSystems IRIS 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.
6. The **InterSystems IRIS Instance Name** dialog box lets you assign a name to the new instance you are installing. The default name is IRIS (or if other instances exist, IRIS*n*, where *n* is the number of InterSystems IRIS instances including

this new one). Accept the default or enter another name, using only alphanumeric characters, underscores, and dashes. You cannot change the instance name after the instance is installed.

7. The **Destination Folder** dialog box lets you select a destination directory for the InterSystems IRIS software for the new instance. You can select or create a directory by clicking **Change**. If the specified directory does not exist, setup lets you create it.

   For information about the requirements for choosing an installation directory, see Installation Directory in the “Preparing to Install InterSystems IRIS” chapter.

8. The **Setup Type** dialog box lets you specify how you intend to use InterSystems IRIS. Review the Setup Type section in the “Preparing to Install InterSystems IRIS” chapter of this book to decide which components of InterSystems IRIS you need.

   Note: If the CSP Gateway is already installed on your system when you install the Web Gateway, the installer automatically upgrades the CSP Gateway to the Web Gateway. For details, see Preexisting CSP Gateway.

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

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

2.2.1 Installing Development or Server Components Only

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

To perform a Development or Server installation, first complete the steps in the InterSystems IRIS Installation Procedure, then continue:

1. Select **Development** or **Server** in the **Setup Type** dialog box.

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 InterSystems security settings to be: **Minimal**, **Normal**, or **Locked Down**. For a detailed explanation of these security settings, see Initial InterSystems Security Settings.

   Minimal is only available for InterSystems IRIS installations. If you choose **Minimal**, skip to the **Ready to Install** step.

   Important: If you select **Minimal** for your initial security setting but InterSystems IRIS requires network access to shared drives and printers, you must manually change the Windows user account under which to run the InterSystems service to one that has the desired access privileges. For instructions on how to do so, see Managing Access to the InterSystems IRIS Instance.

4. The **Enter Credentials for InterSystems Service** dialog lets you choose the credentials under which the Windows InterSystems service runs.

   The default is the local default SYSTEM account. You can instead specify a defined (existing) Windows user account and password; if you do, the installer verifies that:

- the account exists on the domain.
- the password is correct.

Note: The default SYSTEM account is not always appropriate, and may cause issues relating to file permissions and network security access. Also, if you are using Kerberos, you must enter a defined account that you have set up to run the InterSystems service. To decide what account should run the Windows InterSystems Service, review the Windows User Accounts section of this chapter.

5. The **InterSystems IRIS Users configuration** dialog let you enter the initial password for the predefined InterSystems IRIS user accounts listed in the dialog box. The password must meet the criteria described in the Initial User Security Settings table. After clicking **Next**, the installer prompts you for the initial password for the CSPSystem predefined account.

   If you specified a Windows account in the previous step, the installation creates a "privileged user account" with the same name as the Windows account to grant that user access to services necessary to administer InterSystems IRIS. For more details on all these predefined accounts, see Predefined User Accounts.

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

   You can also click the **License** button to select an InterSystems IRIS license key. If the key is valid, the license is automatically activated and the license key is copied to the instance's *install-dir*/mgr directory as iris.key during installation, and no further activation procedure is required. If you do not select a key, you can activate your InterSystems IRIS license key following installation. See Activating a License Key in the "Managing InterSystems IRIS Licensing" chapter of the *System Administration Guide* for information about licenses, license keys and activation.

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

7. The **InterSystems IRIS Installation Completed** dialog box indicates the installation has completed successfully. Choose whether you want to see the *Getting Started* page and click **Finish**.

By default, the system starts automatically after installation is complete and the InterSystems IRIS launcher icon appears in the system tray area of the Windows tool bar. Click the launcher to bring up the InterSystems IRIS menu. In addition, there is an **InterSystems IRIS** item on the Windows **Programs** menu.

Continue to the Post-Installation Tasks section to complete the installation process.

2.2.2 Installing Client Components Only

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

To perform a client installation, first complete the steps in the InterSystems IRIS Installation Procedure, then continue:

1. Select **Client** in the **Setup Type** dialog box 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 InterSystems IRIS in the selected directory.

3. The **InterSystems IRIS Installation Completed** dialog box indicates the installation has completed successfully. Click **Finish**.

After InterSystems IRIS is installed on a client, the InterSystems IRIS launcher icon appears in the system tray area of the Windows tool bar; it appears dimmed because there is no InterSystems IRIS 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 *System Administration Guide*.

Continue to the Post-Installation Tasks section to complete the installation process.

2.2.3 Installing the Web Gateway Only

If you wish, you can install only those parts of InterSystems IRIS that are required for the Web Gateway.

To perform a Web Gateway installation, first complete the steps in the InterSystems IRIS Installation Procedure, then continue:

1. Select **Web Server** in the **Setup Type** dialog box 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 InterSystems IRIS in the selected directory.
3. The **InterSystems IRIS Installation Completed** 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 Web Gateway. The installer also displays a check box for IIS; if you select this, Web Gateway IIS modules are installed in C:\InetPub\CSPGateway.

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

- /
- /csp
- /<instancename> (by default, /IRIS)

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

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

Continue to the Post-Installation Tasks section to complete the installation process.

2.2.4 Performing a Custom Installation

The InterSystems IRIS installation program allows you to select certain InterSystems IRIS components to install on the system. For example, you may want to install only the Web Gateway. Keep in mind that some selections require that you also install other components.

To perform a custom InterSystems IRIS installation, first complete the steps in the InterSystems IRIS Installation Procedure, then continue:

1. Select **Custom** in the **Setup Type** dialog box.

2. In the **Custom Setup** dialog box, select the components you want to install. Some of the components include sub-items; to view these, select the **+**icon.

   Select a component to see a short description, or view a list of available components in the Components Installed by Setup Type table.

   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).
5. The **Enter Port Numbers** dialog box lets you change the port numbers assigned by InterSystems IRIS. Review the Port Numbers section in the "Preparing to Install InterSystems IRIS" chapter of this book for more information.
6. The **Initial Security Settings** dialog box lets you decide how restrictive you want the initial InterSystems security settings to be: **Minimal**, **Normal**, or **Locked Down**. For a detailed explanation of these security settings, see Initial InterSystems Security Settings.

   Minimal is only available for InterSystems IRIS installations. If you choose **Minimal**, skip to the **Ready to Install** step.

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

7. The **Enter Credentials for InterSystems Service** dialog lets you choose the credentials under which the Windows InterSystems service runs.

   The default is the local default SYSTEM account. You can instead specify a defined (existing) Windows user account and password; if you do, the installer verifies that:

   - the account exists on the domain.
   - the password is correct.

   Note: The default SYSTEM account is not always appropriate, and may cause issues relating to file permissions and network security access. Also, if you are using Kerberos, you must enter a defined account that you have set up to run the InterSystems service. To decide what account should run the Windows InterSystems Service, review the Windows User Accounts section of this chapter.

8. The **InterSystems IRIS Users configuration** dialog let you enter the initial password for the predefined InterSystems IRIS user accounts listed in the dialog box. The password must meet the criteria described in the Initial User Security Settings table. After clicking **Next**, the installer prompts you for the initial password for the CSPSystem predefined account.

   If you specified a Windows account in the previous step, the installation creates a "privileged user account" with the same name as the Windows account to grant that user access to services necessary to administer InterSystems IRIS. For more details on all these predefined accounts, see Predefined User Accounts.

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

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

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

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

10. The **InterSystems IRIS Installation Completed** dialog box indicates the installation has completed successfully. Choose whether you want to see the *Getting Started* page and click **Finish**.

Continue to the Post-Installation Tasks section to complete the installation process.

2.3 Unattended Installation Procedure

The InterSystems IRIS for Windows installer provides a way to perform *unattended* installation, upgrade, reinstallation (repair), and removal (uninstall) of instances of InterSystems IRIS 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 InterSystems IRIS 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 Removal
- 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 InterSystems IRIS, use the following command:

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

You must specify:

Variable	Description
<path>	The path to the InterSystems IRIS installation file.
<installer>.exe	The name of the InterSystems IRIS installation file.
<instancename>	The name for the new InterSystems IRIS instance. If omitted, the default value is IRIS, but you must specify a different value if there are one or more instances already installed on the machine.
/qb or **/qn**	Whether to display a progress bar during the installation (**/qb**) or to perform a fully silent installation (**/qn**).
<properties>	The properties to passing to the installer (see the Command-Line Properties table).

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

```
C:\downloads\IRIS-<version_number>-win_x64.exe /qn INSTALLDIR=C:\InterSystems\MyIris
```

To install an instance of InterSystems IRIS with the instance name IrisA, specify:

```
C:\downloads\IRIS-<version_number>-win_x64.exe /instance IrisA /qn
```

You can custom install a subset of features using the **ADDLOCAL** property; see **ADDLOCAL** in the Command-Line Properties table for more information. As an example, to install only the launcher on a 64–bit Windows system (accepting the default instance name and directory), specify:

```
C:\downloads\IRIS-<version_number>-win_x64.exe /qn ADDLOCAL=cube
```

Note: An unattended installation does not install the Web Gateway by default; this must be specified using the **ADDLOCAL** property.

When the installation is finished, continue to the Post-Installation Tasks section of this chapter.

2.3.2 Running an Unattended Upgrade or Reinstall

In addition to installing a new instance, the InterSystems IRIS installer can be called on an existing installed instance. To do so, you must use the **/instance** flag to specify the name of the target existing instance. The action the installer takes depends on the version of the installation file compared to the version of the 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 IRISB that is an earlier version than the installation file, use the following:

```
C:\downloads\IRIS-<version_number>-win_x64.exe /instance IRISB /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 IRISB, you can use the following command (assuming the installation file and IRISB are the same version):

```
C:\downloads\IRIS-<version_number>-win_x64.exe /instance IRISB /qn REINSTALL=studio
```

2.3.3 Running an Unattended Removal

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

```
<path>\<installer>.exe /instance <instancename> /q[b|n] 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 Web Gateway from instance IrisC, use the command:

```
C:\downloads\IRIS-<version_number>-win_x64.exe /instance IrisC /qn 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 following Registry location:

Processor Type	Registry Location
32–bit	HKEY_LOCAL_MACHINE\SOFTWARE\Intersystems\IRIS\Configurations\<*instance*>
64–bit	HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intersystems\IRIS\Configurations\<*instance*>

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

```
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 (https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc759262(v=ws.10)).

2.3.4 Command Line Reference

The Command-Line Properties table describes the InterSystems IRIS-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:

```
... ISCSTARTIRIS=0 WEBSERVERPORT=57779 INITIALSECURITY=Normal
```

Note: In the following table, the **REINSTALL** and **REMOVE** properties are used with installed instances, as described in Running an Unattended Upgrade or Reinstall and Running an Unattended Removal, respectively.

Table 2–1: Command-Line Properties

Property Name	Description
ADDLOCAL	Use this property to custom install a new instance of InterSystems IRIS 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
CSPSKIPIISCONFIG	Optionally use this property to install IIS CSP binary files. With a value of 1, the files are installed without any changes to the IIS web server configuration. With a value of 0, the installer updates 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 are installed without any changes to the IIS web server configuration. With a value of 0, the installer updates 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 are installed without any changes to the IIS web server configuration. With a value of 0, the installer updates the IIS web server configuration regardless of the existence of the /csp virtual directory.
CSPSYSTEMUSERPASSWORD	If the security level is `Normal` or `LockedDown` (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 **IRISUSERPASSWORD** 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 `LockedDown`. **Note:** Omit this property to accept the default of `None`. See also the **IRISUSERPASSWORD**, **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\IRIS*n*, where *n* is `{empty}`, `1`, `2`, ... `127`.
INSTALLERMANIFEST	If installing with an installation manifest, as described in Using the Manifest in the appendix “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).
INSTALLERMANIFESTLOGFILE	If installing with an installation manifest, as described in Using the Manifest in the appendix “Creating and Using an Installation Manifest”, this property specifies where %Installer prints messages.

Property Name	Description
INSTALLERMANIFESTLOGLEVEL	If installing with 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.

INSTALLERMANIFESTPARAMS	If installing with an installation manifest, as described in Using the Manifest in the appendix "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 configuration parameter file (iris.cpf) and activate the changes before your manifest runs. You can specify these parameters:
	• 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"`

IRISSERVICEDOMAIN	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 InterSystems service login account specified by **IRISSERVICEUSER**. **Note:** If the service credentials are specified as `LocalSystem`, do not use this property.

Property Name	Description
IRISSERVICEPASSWORD	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 InterSystems service account specified by **IRISSERVICEUSER**. **Note:** If the service credentials are specified as `LocalSystem`, do not use this property.
IRISSERVICEUSER	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 InterSystems service. **Note:** If the service credentials are specified as `LocalSystem`, do not use this property.
IRISUSERPASSWORD	Required if the security level is `Normal` or `LockedDown`; see the **INITIALSECURITY** property in this table. Use this property to specify the password for the predefined InterSystems IRIS accounts _SYSTEM, Admin, and SuperUser, as well as the account with the username specified by **IRISSERVICEUSER** if **SERVICECREDENTIALS** is specified as `UserDefined`. **Note:** If the initial security level is `None`, do not use this property.
ISCSTARTIRIS	Optionally set this property to `0` to prevent InterSystems IRIS from starting after installation. The default is `1`, to start InterSystems IRIS.
ISCSTARTLAUNCHER	Optionally set this property to `0` to prevent the InterSystems IRIS launcher from being added to the system tray. The default is `1`, to add the launcher.
REINSTALL	Use this property to reinstall (repair) an installed instance of InterSystems IRIS or to change the custom-installed features (see the Custom-Installable Features table) for an installed instance of InterSystems IRIS: • 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 InterSystems IRIS 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).

Property Name	Description
REMOVE	Use this property to uninstall (remove) an instance of InterSystems IRIS or a subset of custom-installed features (see the Custom-Installable Features table) installed for an installed instance of InterSystems IRIS: • To remove an instance of InterSystems IRIS, specify `ALL`. • To remove a subset of InterSystems IRIS features, 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 `LockedDown` (see the **INITIALSECURITY** property in this table), optionally use this property to specify the credentials under which the Windows InterSystems service runs: `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 Access to the InterSystems IRIS Instance for important information about the InterSystems service account. If you specify `UserDefined` for this property, you must also specify the **IRISSERVICEDOMAIN**, **IRISSERVICEPASSWORD**, and **IRISSERVICEUSER** properties.
SKIPUPGRADECHECK	When upgrading an instance, set this property to `1` to bypass system pre-upgrade checking. The default is `0`. **Note:** Generally, you should leave pre-upgrade checking enabled.
SUPERSERVERPORT	Optionally use this property to specify the Superserver port to be used by the instance being installed. **Note:** By default, this port is set to `1972` (if available). Otherwise, the port is set to `51773` or the first available subsequent number.
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`. 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.

Property Name	Description
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 `52773` and increasing by 1 for each installed instance of InterSystems IRIS.

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, 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 USER database include the following in the command line:

```
ADDLOCAL=server,server_user
```

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:

```
ADDLOCAL=documentation,documentation_pdf,server,development,callin
```

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.)

Table 2–2: Custom-Installable Features

Component Group (featurename)	Components (featurename)
Development (**development**)	Callin (**callin**) Callin, Threaded (**callin_threaded**) Threaded Server Libraries (**server_threaded**) Other Samples (**other_samples**) Other Development Libraries (**development_other**)
Documentation (**documentation**)	PDF Documentation (**documentation_pdf**) Online Documentation (**documentation_online**)
InterSystems IntegratedML (**integratedml**)	
Launcher (**cube**)	
Server (**server**)	User database (**server_user**) SQL Gateway (**sqlgateway**) Apache Formatting Objects Processor (**fop**) Server monitoring tools (**server_monitoring**) Agent Service (**agent_service**)

Component Group (featurename)	**Components (featurename)**
Studio (**studio**)	
Database Drivers (**sqltools**)	ODBC (**odbc**) JDBC (**jdbc**)
Web 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 InterSystems IRIS instance using the Management Portal, which is accessible from the InterSystems IRIS launcher. For more information on this management tool, see the "Using the Management Portal" chapter of the *System Administration Guide*.
- The Windows InterSystems service includes the instance name, and starts automatically when you start your server; this means the InterSystems IRIS instance is automatically configured to autostart (start when the system starts).

  You can configure the instance not to autostart by changing the **Start InterSystems IRIS 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 you plan to connect remotely to other instances of InterSystems IRIS, follow the procedure described in the Define a Remote Server Connection section of the "Connecting to Remote Servers" chapter of the *System Administration Guide*.
- If you are using the Windows IIS Web server, file types must be mapped manually; for information, see Mapping InterSystems IRIS File Extensions in the *Web Gateway Guide*.
- Allocate the system memory to be used by InterSystems IRS, as described in the Memory and Startup Settings section of the "Configuring InterSystems IRIS" chapter in *System Administration Guide*.
- InterSystems products often run alongside and interact with non-InterSystems tools. For important information about the effects these interactions can have, see the Configuring Third-Party Software to Work in Conjunction with InterSystems Products appendix of *System Administration Guide*.
- If you need to migrate data from a different database to your newly installed InterSystems IRIS instance, see Data Migration.
- 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:

- Configuring Multiple InterSystems IRIS Instances
- Changing the InterSystems IRIS Language
- Reinstalling or Uninstalling InterSystems IRIS
- The Write-cache Buffer

2.5.1 Configuring Multiple InterSystems IRIS Instances

You can install and simultaneously run multiple instances on a single Windows machine. To do so, install InterSystems IRIS 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 InterSystems IRIS Instances section of the *System Administration Guide* for more detailed information.

2.5.2 Changing the InterSystems IRIS Language

When you install InterSystems IRIS, 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 an InterSystems IRIS installation, see Using the NLS Settings Page of the Management Portal in the “Configuring InterSystems IRIS” chapter of the *InterSystems IRIS 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.

2.5.3 Reinstalling or Uninstalling InterSystems IRIS

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

When you run setup as described in InterSystems IRIS Installation Procedure and select an InterSystems IRIS 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 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 InterSystems IRIS installer or Windows Control Panel **Programs and Features** to uninstall InterSystems IRIS. Other uninstall programs are not supported and using them may cause unexpected results.

2.5.4 The Write-cache Buffer

Certain InterSystems IRIS features utilize **Windows write-cache buffer flushing**, which is enabled by default. To verify this option is correctly enabled and InterSystems IRIS 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 InterSystems IRIS on UNIX®, Linux, and macOS

This chapter describes how to install InterSystems IRIS® data platform on a UNIX®, Linux, or macOS system. It assumes that you are familiar with UNIX®, Linux, and macOS directory structures, utilities, and commands.

Note: It is not possible to perform multiple simultaneous installations on the same machine.

3.1 Before Installing

Before beginning the installation, be sure you have read all the information that applies to your platform in the "Preparing to Install InterSystems IRIS" chapter of this guide. Additionally, review the following topics:

- Uncompressing the Installation Kit
- Determining Owners and Groups
- Installing the Required Dependencies
- Choosing a Configuration Strategy

3.1.1 Uncompressing the Installation Kit

If your installation kit is in the form of a .tar file, for example iris-2019.3.0.710.0-lnxrhx64.tar.gz, uncompress the file into a temporary directory to avoid permissions issues, as shown in the following example.

```
# mkdir /tmp/iriskit
# chmod og+rx /tmp/iriskit
# umask 022
# gunzip -c /download/iris-<version_number>-lnxrhx64.tar.gz | ( cd /tmp/iriskit ; tar xf - )
```

The installation files uncompress into a directory with the same name as the .tar file, for example /tmp/iriskit/iris-2019.3.0.710.0-lnxrhx64.

Important: Do not uncompress the file into or run InterSystems IRIS installation from the /home directory, or any of its subdirectories. Additionally, the pathname of the temporary directory cannot contain spaces.

Do not install InterSystems IRIS into the same directory you used to expand the installation kit.

Note: 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**.

3.1.2 Determining Owners and Groups

The installation process prompts for the following user and group information:

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

For more information about these categories, see the UNIX® User and Group Identifications section in the "Using InterSystems IRIS on UNIX®, Linux, and macOS" chapter of *System Administration Guide*.

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

The user account you identify as *Owner of the instance* and the group you identify as *Group allowed to start and stop the instance* must both 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 (or **mkgroup** and **mkuser** on AIX®), the system creates the account for the *effective user for InterSystems IRIS superserver* and the *effective group for InterSystems IRIS 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 (or **mkgroup** and **mkuser** on AIX®) 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 InterSystems IRIS 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 "UNIX® Users, Groups and Permissions" chapter of the *System Administration Guide* for details on how InterSystems IRIS assigns permissions.

3.1.3 Installing the Required Dependencies

As part of the installation process, the InterSystems IRIS installer checks whether the system has the required dependencies. If the system is missing any dependencies, the installation fails with a message specifying which dependencies to install before rerunning the installer.

You can run the requirements checker independently from an installation by running irisinstall with the prechecker option (e.g. **# irisinstall --prechecker**). This is useful when modifying or upgrading an InterSystems IRIS dependency to ensure the requirements are still met after the change.

The requirements checker always runs during instance startup. The startup fails if the requirements are not met.

Note: On macOS, installers for InterSystems IRIS versions prior to version 2022.1 do not have a requirements checker, but there are required dependencies.

When installing or running InterSystems IRIS on macOS, you must install openssl@1.1 via Homebrew (https://formulae.brew.sh/formula/openssl@1.1).

To use Embedded Python within InterSystems IRIS on macOS, you must also install python@3.9 (https://formulae.brew.sh/formula/python@3.9)

3.1.4 Choosing a Configuration Strategy

There are several methods to configure InterSystems IRIS before installation. These methods are particularly useful when considering automated deployment, as they reduce the need to adjust settings after the installation is complete.

Before installing InterSystems IRIS, consider whether to use any of these methods:

- Configuration merge allows you to define an InterSystems IRIS configuration in a *merge file*. A merge file specifies desired settings and can also perform administrative actions such as creating users, namespaces, and databases. You may apply a merge file during or after installation.
- Installation manifests describe a specific InterSystems IRIS configuration, and can be used with either the **irisinstall** or **irisinstall_silent** commands.
- Adding Unix® Installation Packages to an InterSystems IRIS Distribution describes how to add new packages to an InterSystems IRIS distribution.

3.2 InterSystems IRIS Installation

This section contains the procedure to install InterSystems IRIS on a UNIX® platform. There are two ways to run an attended IRIS installation, as explained in the following sections:

- Standard InterSystems IRIS Installation Procedure — Perform the basic InterSystems IRIS installation using irisinstall. You can select various installation options and interactively choose what to install along with the InterSystems IRIS server.
- Client-Only InterSystems IRIS Installation Procedure — Install the appropriate client components using irisinstall_client, without installing the InterSystems IRIS server.

Generally, InterSystems IRIS installations are run using root privileges. It is possible to run the installation as a nonroot user, though this is subject to the limitations described in Installing as a nonroot User.

3.2.1 Standard InterSystems IRIS Installation Procedure

Standard InterSystems IRIS 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.

The installation script, irisinstall, does the following:

- Installs the InterSystems IRIS system manager databases.
- Starts InterSystems IRIS in installation mode.

- Installs InterSystems IRIS system manager globals and routines.
- Shuts down InterSystems IRIS and restarts using the default configuration file (iris.cpf). Upgrade installations restart using their updated configuration files.

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

To perform the standard InterSystems IRIS server installation, follow the installation procedure.

1. Log in as user ID `root`. The standard InterSystems IRIS installation must be run by a user with root privileges. It is acceptable to **su** (superuser) to `root` while logged in from another account.

   Note: If root is unavailable, you can perform a non-standard, limited InterSystems IRIS installation as a nonroot user. For details, read the Installing InterSystems IRIS as a Nonroot User section below before continuing the installation procedure.

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

   ```
   # /<pathname>/irisinstall
   ```

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

3. 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. This may include asking for the “platform name” in the format of the string at the end of the installer kit name. 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.

4. The script displays a list of any existing InterSystems IRIS instances on the host.

5. At the **Enter instance name:** prompt, enter an instance name, using only alphanumeric characters, underscores, and dashes. 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`. Be sure to review the Installation Directory section in the “Preparing to Install InterSystems IRIS” chapter for more important information about choosing an installation directory.

   Note: If you select an existing instance that is of the same InterSystems IRIS 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.

   The InterSystems IRIS registry directory, /usr/local/etc/irissys, is always created along with the InterSystems IRIS installation directory.

6. Next, select the installation type from the choices:

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

   Note: If the CSP Gateway is already installed on your system when you install the Web Gateway, the installer automatically upgrades the CSP Gateway to the Web Gateway. For details, see Preexisting CSP Gateway.

Review the Setup Type section in the "Preparing to Install InterSystems IRIS" chapter of this book to decide which installation type is appropriate. The default for new instances is `Development`, which contains a complete installation of InterSystems IRIS; if you are upgrading an existing instance, the default is the option that was used to originally install the instance.

Note: If you choose the `Custom` option, see the InterSystems IRIS Custom Installation section for additional installation steps.

7. Next, indicate whether to install InterSystems IRIS with 8-bit or Unicode character support. On upgrade, you can convert from 8-bit to Unicode, but not the reverse.
8. For new instances, you must next specify security settings. (Security settings cannot be changed when upgrading.)

   First, decide how restrictive you want the initial InterSystems security settings to be: choose from Minimal (`1`), Normal (`2`), and Locked Down (`3`). For a detailed explanation of these security settings, see Initial InterSystems Security Settings.

   Minimal is only available for InterSystems IRIS installations. If you choose this, you can skip the next step; the installer sets the owner of the instance as `root`.

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

9. 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 InterSystems IRIS processes. See Determining Owners and Groups for information about this account. Once InterSystems IRIS is installed, you cannot change the owner of the instance.

   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. A privileged InterSystems IRIS user account is created for this user with the %All role.

      This password is used not only for the InterSystems IRIS privileged user account, but also for the _SYSTEM, Admin, and SuperUser predefined user accounts. For more details on these predefined users, see Predefined User Accounts.

   c. Password for the CSPSystem predefined user.

   Note: The passwords must meet the criteria described in the Initial User Security Settings table. Passwords entered during this procedure cannot include space, tab, or backslash characters; the installer reject such passwords.

10. At this point in the installation, you are asked which group should be allowed to start and stop InterSystems IRIS. 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; InterSystems IRIS verifies that the group exists before proceeding.
11. Finally, for a new instance, or if the script does not detect an iris.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 InterSystems IRIS Licensing" chapter of the *System Administration Guide* for information about licenses, license keys and activation.

    Note: On macOS, you may receive a prompt regarding network connections for **irisdb**. If so, select **Allow**.

12. Review your installation options and press enter to proceed with the installation. File copying does not begin until you answer `Yes`.

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

Continue to the Post-Installation Tasks section to complete the installation process.

3.2.1.1 InterSystems IRIS Custom Installation

If you choose a custom installation, you must answer additional questions throughout the installation 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 InterSystems IRIS assign the port numbers, or you can enter custom port numbers. Upgrade installs do not offer this choice; they keep the port numbers of the original instance. Review the Port Numbers section in the "Preparing to Install InterSystems IRIS" chapter of this book for more information.
- *Web Gateway and External Web Sever* — You can configure the Web Gateway to use a supported external web server. Answer **`Yes`** and then answer additional web server configuration questions to configure the Web Gateway after the InterSystems IRIS installation completes.
- *Additional Security Options* — If you chose Normal or Locked Down initial security, you have the option of configuring additional security settings (minimal installations use the default IDs):
  - *Effective group for InterSystems IRIS* — InterSystems IRIS 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 `irisusr`.)
  - *Effective user for the InterSystems IRIS 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 `irisusr`.)

  See Determine Owners and Groups for additional information.
- *IntegratedML* — You have the option to install *without* InterSystems IntegratedML. For more information about IntegratedML, see *Using IntegratedML*.
- *Client Components* — You have the option of installing one client component. This client component may not be supported on all platforms.

  ```
  Client component selection
       [*]        1) IRIS shared library support (callin)
                  +) 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.

  Note: UNIX® InterSystems IRIS kits always install client components.

See the following guides for more information on the specific client components in the selection list:

- *Using the InterSystems ODBC Driver*
- *Using the Callin API*

3.2.2 Client-Only InterSystems IRIS Installation Procedure

The InterSystems IRIS distribution contains a separate script to install a client-only version of InterSystems IRIS. 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 InterSystems IRIS client installation:

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

   ```
   # /<pathname>/irisinstall
   ```

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

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. At the **Enter a destination directory for client components** prompt, specify the installation directory. If the directory you specify does not exist, it asks if you want to create it. The default answer is `Yes`. Be sure to review the Installation Directory section in the “Preparing to Install InterSystems IRIS” chapter for more important information about choosing an installation directory. Press **Enter** to continue with the installation.

   Note: The InterSystems IRIS registry directory, /usr/local/etc/irissys, is always created along with the InterSystems IRIS installation directory.

4. Choose from the available client component options. Components that require the InterSystems IRIS 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
             +) 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.

The list of client-only components does not include the shared library support 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 irisinstall script instead.

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

- *Using the InterSystems ODBC Driver*

Continue to the Post-Installation Tasks section to complete the installation process.

3.2.3 Installing as a Nonroot User

When installing InterSystems IRIS in a production environment, InterSystems recommends using root privileges. It is possible to run an InterSystems IRIS installation without root privilege, but these installations have several limitations. The following sections describe these limitations and the differences from a standard InterSystems IRIS installation.

- Why InterSystems IRIS Installation Uses Root
- Nonroot Installation Limitations
- Nonroot Installation Differences

Important: Root privilege should only be used when *installing* InterSystems IRIS. Once the installation is complete, all users should interact with InterSystems IRIS using nonroot privileges.

3.2.3.1 Why InterSystems IRIS Installation Uses Root

InterSystems IRIS is typically installed using root and operated using nonroot privileges. Several features require root access, but the majority of processes run as a user or group that you specify during installation. The purpose of these users and groups, and how they use root, is described in UNIX Users, Groups, and Permissions.

InterSystems IRIS processes that utilize root privileges include:

- The Virtual IP process, which has root as its effective user ID (UID) to modify network settings on the operating system.
- The Control Process, which has the instance owner as its effective UID and root as its real UID. The real UID is used to get large pages at startup and to communicate with other InterSystems IRIS processes.
- The startup executables, which have root as the effective UID.

Installing InterSystems IRIS as root also enhances security by ensuring that only users with root privileges can modify or replace the file structure.

3.2.3.2 Nonroot Installation Limitations

While nonroot installations of InterSystems IRIS are supported, there are several features that cannot be used in instances installed in this way:

- Huge pages are not available.
- The installation mount point cannot be mounted with *nosuid* set.
- The Web Gateway cannot be configured to use an external web server.
- A mirror Virtual IP cannot be used.

  Note: For alternative methods of routing network traffic, such as using a network load balancer or the Web Gateway, see Redirecting Application Connections Following Failover or Disaster Recovery.

- There is no option to specify the instance owner and group allowed to start and stop InterSystems IRIS during installation (as described in Determining Owners and Groups).
- There is no group access. All instance files, including the registry, are owned and can be read, written, and executed by the installing user only.

  For example, where a standard instance might have:

  ```
  -rws--x--- 5 root develop 43282 Aug 28 07:52 irismgr
  -r-x--s--x 1 <nonroot-user> irisusr 23058 Aug 28 07:52 irisuxsession
  ```

a nonroot instance would have:

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

The registry is located in the directory specified by *IRISSYS*, and nonroot instances are found in that registry. (The **iris** executable is also in that directory.) Only nonroot instances may be in the nonroot registry. Any attempt to access a root-installed instance from a nonroot registry fails. 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 fails.

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.

3.2.3.3 Nonroot Installation Differences

Along with the feature limitations described above, there are several apparent differences between root and nonroot InterSystems IRIS installations:

- The *IRISSYS* environment variable must be defined as an existing directory writable by the installing user, and must be present during installation and all instance operations.
- The ISCAgent is installed in the directory specified by *IRISSYS*.

  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 *High Availability Guide*.

- Only the installing user’s account can access and operate the InterSystems IRIS instance.
- All InterSystems IRIS executables and processes run as the installing user.

3.3 Unattended InterSystems IRIS Installation

You can perform unattended installation of InterSystems IRIS instances on your systems using the irisinstall_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 irisinstall_silent command line. Each specified package represents an InterSystems IRIS component; the installation scripts for each component are contained in the packages directory below the directory containing the irisinstall_silent script.

The general format for the irisinstall_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_IRISUSER="<user>" ISC_PACKAGE_IRISGROUP="<group>"
 ISC_PACKAGE_CLIENT_COMPONENTS="<component1> <component2> ..."
 ISC_PACKAGE_STARTIRIS="Y"|"N"
 ./irisinstall_silent [<pkg> ...]
```

Note: If you are attempting to install InterSystems IRIS without root permissions, refer to the Installing as a Nonroot User section for additional details.

This section discusses the following topics:

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

3.3.1 Unattended Installation Parameters

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

Table 3–1: Unattended Installation Parameters

Parameter	Description
`ISC_CPF_MERGE_FILE=`"*<location>*"	Specifies the location of the configuration merge file when performing a configuration merge. For more information, see Automating Configuration of InterSystems IRIS with Configuration Merge.
`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="MyIris"` **Note:** If this a new install, the `ISC_PACKAGE_INSTALLDIR` parameter is required.
`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/MyIris"` If the specified directory does not exist, the installation attempts to create it. This parameter is ignored if you are upgrading an installation. **Note:** Review the Installation Directory section of this book for information about choosing an installation directory.
`ISC_PACKAGE_INSTALL_INTEGRATEDML="Y"|"N"`	Specifies whether or not to install InterSystems IntegratedML. For more information about IntegratedML, see *Using IntegratedML*. **Note:** By default, this is set to `Y`.
`ISC_PACKAGE_UNICODE="Y"|"N"` (Optional)	Specifies whether or not this is a UNICODE installation; valid values are `Y` or `N`. See Character Width Settings for more information.

Parameter	**Description**
`ISC_PACKAGE_INITIAL_SECURITY="Minimal"|"Normal"|"LockedDown"` (Optional)	Specifies the initial security setting for the installation; valid values are: `"Minimal"`, `"Normal"`, or `"LockedDown"`. **Note:** `"Minimal"` is only available for InterSystems IRIS installations. If it is set to `"Normal"` or `"LockedDown"`, `ISC_PACKAGE_USER_PASSWORD` is required.
`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="irisusr"` **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 `"LockedDown"` and this parameter is not specified, the installation fails and an error is thrown.

Parameter	Description
`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 `"LockedDown"` and this parameter is not specified, the value of `ISC_PACKAGE_USER_PASSWORD` is used.
`ISC_PACKAGE_IRISUSER="<user>"` (Optional)	Specifies the effective user for the InterSystems IRIS Superserver. **Note:** By default, this is set to `irisusr`. If the security level is `"Minimal"`, this parameter is ignored and set to the default.
`ISC_PACKAGE_IRISGROUP="<group>"` (Optional)	Specifies the effective user for InterSystems IRIS processes. **Note:** By default, this is set to `irisusr`. 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_WEB_CONFIGURE="Y"|"N"` (Optional)	Specifies whether or not to configure the Web Gateway for an external web server. **Note:** By default, this is set to `N` (do not configured the gateway for an external web server).

Parameter	Description
`ISC_PACKAGE_WEB_SERVERTYPE="Apache"|"SunOne"|"None"` (Optional)	Type of existing web server for the Web Gateway to use. For example: `ISC_PACKAGE_WEB_SERVERTYPE="Apache"` **Note:** By default, this is set to `None`.
`ISC_PACKAGE_WEB_APACHE_VERSION=2.2|2.4` (Optional, with `ISC_PACKAGE_WEB_SERVERTYPE="Apache"`)	Version of Apache web server. **Note:** By default, the version is autodetected.
`ISC_PACKAGE_WEB_APACHE_USER="<username>"` (Optional, with `ISC_PACKAGE_WEB_SERVERTYPE="Apache"`)	Username for Apache web server. **Note:** By default, the username is autodetected.
`ISC_PACKAGE_WEB_APACHE_CONF="<path_to_httpd.conf>"` (Optional, with `ISC_PACKAGE_WEB_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 stardard locations. Installation exits with error if `ISC_PACKAGE_WEB_SERVERTYPE="Apache"` and httpd.conf location is undetermined.
`ISC_PACKAGE_WEB_SUNONE_DIR="<sunone_installation_dir>"` (Optional, with `ISC_PACKAGE_WEB_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_WEB_SUNONE_NUMBER="<number>"` (Optional, with `ISC_PACKAGE_WEB_SERVERTYPE="SunOne"`)	SunOne web server number to configure if multiple servers are installed. **Note:** By default, value is `1`.
`ISC_PACKAGE_WEB_GATEWAY_DIR="<web_gateway_directory>"` (Optional, for new Web Gateway installations only)	Directory to contain the Web Gateway files. **Note:** By default, the directory is /opt/webgateway_.
`ISC_PACKAGE_STARTIRIS="Y"|"N"` (Optional)	Specifies whether or not to start the installed InterSystems IRIS instance following installation. **Note:** By default, this is set to `Y`, to start the instance.

Parameter	Description
`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 appendix "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.
`ISC_INSTALLER_LOGLEVEL="<level>"`	When installing with an installation manifest, specifies the log level, from `-1` ("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 set to `1972` (if available). Otherwise, the port is set to `51773` or the first available subsequent number.
`ISC_PACKAGE_USER_DATABASE=Y|N`	Specifies whether or not to install the optional users database. By default, this is set to Y.
`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 InterSystems IRIS.

3.3.2 Unattended Installation Packages

The installation scripts for each component are contained in the packages directory below the directory containing the irisinstall_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:

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

   For example, in the following manifest.isc file, the value of the package key has been changed to match the directory name (custom_install).

```
#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="MyIris" ./irisinstall_silent custom_install`.

Note: See the Adding UNIX® Installation Packages to an InterSystems IRIS Distribution appendix for information about creating your own UNIX® installation packages and adding them to an InterSystems IRIS distribution.

3.3.3 Unattended Installation Examples

The following examples illustrate how you can use the irisinstall_silent script for unattended installation of InterSystems IRIS on UNIX® platforms.

Example 1

In this example, all packages are installed with minimal security:

```
sudo ISC_PACKAGE_INSTANCENAME="MyIris" ISC_PACKAGE_INSTALLDIR="/opt/MyIris1" ./irisinstall_silent
```

If the MyIris instance already exists, it is upgraded; otherwise, it is installed in the /opt/MyIris1 directory.

Example 2

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

```
sudo ISC_PACKAGE_INSTANCENAME="MyIris" ./irisinstall_silent
```

Example 3

In this example, only the `database_server` and `odbc` packages and the `odbc` client binding are installed with minimal security:

```
sudo ISC_PACKAGE_INSTANCENAME="MyIris" ISC_PACKAGE_INSTALLDIR="/opt/MyIris2"
ISC_PACKAGE_CLIENT_COMPONENTS="odbc" ./irisinstall_silent database_server odbc
```

3.4 Post-Installation Tasks

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

- Starting InterSystems IRIS, as described in the section below.
- If you plan to develop using Studio, see the Install InterSystems IRIS Client on Windows for Development section.
- Allocate the system memory to be used by InterSystems IRIS, as described in the Memory and Startup Settings section of the "Configuring InterSystems IRIS" chapter in *System Administration Guide*.
- InterSystems products often run alongside and interact with non-InterSystems tools. For important information about the effects these interactions can have, see the Configuring Third-Party Software to Work in Conjunction with InterSystems Products appendix of *System Administration Guide*.
- If you need to migrate data from a different database to your newly installed InterSystems IRIS instance, see Data Migration.
- If appropriate for your installation, perform any additional tasks described in the Special Considerations section.

3.4.1 Starting InterSystems IRIS

For most installations, InterSystems IRIS starts automatically when the install completes. If you need to start InterSystems IRIS, first log in to your operating system, then start InterSystems IRIS using the **iris** command:

```
iris start <instname>
```

Where *instname* is the instance name that you chose during the installation. On Red Hat Linux, the start command is:

```
service <instname> start
```

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

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

Once InterSystems IRIS is started, initiate an InterSystems IRIS session using the **iris terminal** command, as described in Connecting to an InterSystems IRIS Instance in the "Using Multiple Instances of InterSystems IRIS" chapter of the *System Administration Guide*:

```
iris terminal <instname> [parameters]
```

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

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

3.4.2 Installing a Development Environment

InterSystems IRIS installs a private Apache Web server so you can access the Management Portal, but does not include a development environment on UNIX®; therefore, you must choose between Studio, which must be run on a Windows client, and the InterSystems ObjectScript extensions for Visual Studio Code.

To install a Windows client, follow the instructions in the Installing Client Components Only section of the "Installing InterSystems IRIS on Microsoft Windows" chapter in this book. Once the Windows client is installed, perform the following tasks to connect Studio to your instance:

1. Open the InterSystems IRIS launcher on the Windows client.
2. Point to **Preferred Server** and click **Add/Edit** to add a remote server connection to the InterSystems IRIS instance just installed. Make sure you specify the appropriate port numbers for this connection.
3. Point to **Remote System Access**, point to **Studio**, and then click the appropriate connection name you entered in the previous step.

3.5 Special Considerations

This section describes the following topics:

- Multiple InterSystems IRIS Instances
- Uninstalling InterSystems IRIS
- Adjustments for Large Number of Concurrent Processes on macOS

3.5.1 Multiple InterSystems IRIS Instances

You can install and simultaneously run multiple instances on a single machine. To do so, install InterSystems IRIS as for a single installation, giving each instance a unique name, a unique installation directory, and a unique port number.

For more information, reference the Configuring Multiple InterSystems IRIS Instances section of the "Using Multiple Instances of InterSystems IRIS" chapter in *System Administration Guide*.

3.5.2 Uninstalling InterSystems IRIS

To safely uninstall InterSystems IRIS, follow this procedure:

1. Find the name of the InterSystems IRIS instance you wish to delete using the **iris list** command to list all the instances on your machine:

   ```
   iris list
   ```

2. Verify the instance is stopped. If it is not, stop it with the **iris stop** command:

   ```
   iris stop <instname>
   ```

   Where *instname* is the instance name that you chose during the installation. If it hangs, force it down using **iris force**:

   ```
   iris force <instname>
   ```

3. Remove the instance using the **iris delete** command:

   ```
   iris 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 (iris.key), the configuration file (iris.cpf), and the user database file (iris.dat).

The uninstall procedure removes all files installed and created during normal InterSystems IRIS 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 InterSystems IRIS stops before it removes files.

3.5.3 Adjustments for Large Number of Concurrent Processes on macOS

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.

4

Upgrading from an Earlier Version

This topic is intended for customers who are upgrading from an earlier version of InterSystems IRIS®, InterSystems IRIS for Health™, or HealthShare® HealthConnect. Customers looking to migrate from InterSystems Caché® should see Why Migrate to InterSystems IRIS?

Important: InterSystems recommends that each application be thoroughly tested in the upgraded environment **before** it is deployed to customers and begins processing live data.

4.1 Compatibility Goals

The goal of each release is a smooth upgrade to new and improved functionality with no changes required to existing applications. However, because of error corrections, significant enhancements, or changes to applicable standards, this goal cannot always be met. In this case, InterSystems discloses all identified instances where changes to applications are necessary so that customers can gauge effort required to move to the new release.

You may, after reading about the release-specific changes (linked below), conclude that none of them affect your application. Even in this case, regardless of how robustly designed and how well implemented your application is, there is no substitute for quality assurance test results that confirm your judgement and demonstrate the application remains unaffected by the upgrade.

4.2 Before Upgrading

The following upgrade tasks are necessary on all platforms. Perform these tasks before you run the upgrade procedures:

Note: If you are upgrading to a maintenance release, the tasks marked with an asterisk (*) are optional. However, performing all tasks will ensure optimum performance.

1. *Review the* InterSystems Upgrade Impact Checklist (or the Incompatibility History) and make sure to account for incompatibilities as needed.
2. *Read the upgrade chapter in the Release Notes for your product* and make sure to account for incompatibilities as needed.
   - Upgrading to This Release in the *InterSystems IRIS Release Notes*
   - Upgrading to This Release in the *InterSystems IRIS for Health Release Notes*

- Upgrading to This Release in the *HealthShare Health Connect Release Notes*

Follow any specific recommendations that should be performed before installing.

3. *Read the InterSystems Supported Platforms for the version of InterSystems IRIS to which you are updating* — Check that your technologies are supported in the new version of InterSystems IRIS.
4. *Ensure that all source code for the application is available** — After a major upgrade, you must recompile all class code under the new version of InterSystems IRIS or else provide class code that has already been compiled under the new version. Otherwise, you will not be able to run your application after the upgrade. It is recommended that you recompile any routine code.

   Check that you have the code for any classes running in deployed mode (which you can place under deployed mode again when the upgrade is complete), as well as the code for custom routines not generated from classes (*.mac or *.int).
5. *Save custom classes, routines and globals** — On an upgrade, the IRISSYS 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 IRISLIB, IRISTEMP, and ENSLIB databases are completely replaced.
6. *Save user files** — To ensure that your user files are not deleted or replaced during the upgrade, save them in the *install-dir*\mgr\User directory, which is the only directory that is not subject to modification during an upgrade.
7. *Decide whether to create an installation manifest** — A manifest is useful for silent installs, as you can invoke the post-upgrade tasks to run as soon as the upgrade is complete, eliminating the need for interactive steps. See Creating and Using an Installation Manifest for more information.
8. *Precompile any user code that runs on startup** — If your application calls any user code at instance startup (using **%ZSTART**, **ZAUTHENTICATE**, or other means), you must precompile it on an instance of the new version of InterSystems IRIS and use an installation manifest to install it as part of the upgrade process. If you do not, the instance will fail to start up.
9. *Obtain an updated license key** — See the "Managing InterSystems IRIS Licensing" chapter of the *System Administration Guide* for more information.
10. *Check system integrity* — Run a system integrity check on existing directories to ensure there is no corruption in any of the databases. For more information, see the "Introduction to Data Integrity" chapter of *Data Integrity Guide*.
11. *Back up the system* — Before upgrading, InterSystems recommends that you run a complete backup of your system using your customary full operating system backup procedures. If the upgrade is not successful, you may need to restore from this backup. For information about backups, see the "Backup and Restore" chapter of *Data Integrity Guide*.
12. *Stop all productions and disable auto-start of productions* — If your system is running any productions, follow the instructions in the Stopping a Production section of the "Starting and Stopping Productions" chapter in *Managing Productions*.

    It is important to disable auto-start so that you can recompile code before starting productions up again.

Important: If you have removed all external language gateway configurations, you may encounter validation errors during the upgrade process. These validation errors occur when upgrading *from* an affected version. They will occur regardless of the version *to* which you are upgrading. The affected versions include: 2021.1.0, 2021.1.1, 2021.1.2, 2022.1.0, and 2022.1.1.

To prevent these errors, add a single gateway configuration of type **Remote** that points to the local gateway with an arbitrary port number. For example, you can set the **Server Name / IP Address** to `127.0.0.1` and set the **Port** to `1`, naming it `ForUpgrade`. This can be done at any point prior to upgrading and will not impact normal system operation. This configuration can be deleted after the upgrade is completed.

Note: For users of HealthShare Health Connect or InterSystems IRIS for Health, if your instance was previously migrated from Health Connect/HSAP based on the Caché/Ensemble platform, the components.ini file in your installation directory may have a reference to the database MPRLLIB, which is no longer used by the product. Comment out this reference prior to the upgrade by inserting a semicolon at the start of each line. This will prevent a misleading error message in messages.log saying that this database does not exist.

Example:

```
;[MPRLLIB]
;Version=15.032.9686
[HSLIB]
Version=2018.1.0
Compatibility_HSAALIB=15.0
Compatibility_HSPILIB=14.0
Compatibility_VIEWERLIB=17.0
```

4.3 Upgrading an Instance

This section describes the procedure to upgrade a single instance of InterSystems IRIS, InterSystems IRIS for Health, or HealthShare HealthConnect. Before proceeding, be sure to perform the tasks in the Before Upgrading section of this chapter. If you are upgrading a mirror or ECP configuration, also review the information in the Upgrading a Mirror and Upgrading ECP Configurations sections of this chapter.

The procedure you should follow depends on your operating system:

- Windows Upgrade Procedure
- UNIX®, Linux, and macOS Upgrade Procedure

If you are using a manifest as part of your upgrade process, see the instructions in Creating and Using an Installation Manifest before performing the standard installation steps.

CAUTION: Prior to beginning an upgrade of an instance, it is essential that the instance be shut down cleanly. To verify that the shutdown was clean, examine the messages.log file after the shutdown finishes. If the log contains entries similar to the following, then the shutdown was clean:

```
...
05/03/19-14:24:13:234 (5204) 0 Journal restore not required at next startup
05/03/19-14:24:13:234 (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.

4.3.1 Windows Upgrade Procedure

The installer provides all the functionality required to upgrade to a new version of InterSystems IRIS. To instead perform a silent upgrade, which is necessary when using a manifest, review the information in the Running an Unattended Upgrade or Reinstall section of the “Installing on Microsoft Windows” chapter of this guide.

To perform the upgrade:

1. Double-click the installer file, for example, IRIS-win_x64.exe.
2. The installer displays a list of all existing InterSystems IRIS instances on the host. Select the name of the instance you want to upgrade and click **OK**.

3. When prompted for an InterSystems IRIS license key, click **License** and browse for the new iris.key file. If the installer detects a new key file in the *install-dir*/mgr directory, it proceeds without prompting you for the license key.
4. After the installer validates the license key, click **Upgrade**.

   Important: Do not interrupt the installation while it is in progress. If the upgrade fails with any error messages, correct the issues and restart the upgrade installation.

5. After the upgrade is completed, click **Finish**.
6. Examine messages.log, iboot.log, and ensinstall.log in the *install-dir*\mgr directory for any errors. If any fatal error is found, correct the error, and then run the installer again.

Important: If your operating system is configured to use large memory pages, check the startup messages to make sure shared memory is being allocated in accordance with these settings. If you see a message similar to the following, reboot your server to avoid an out-of-memory situation.

```
Failed to allocate 592MB shared memory using large pages.  Switching to small pages.
```

4.3.2 UNIX®, Linux, and macOS Upgrade Procedure

The installation script, **irisinstall**, provides all the functionality required to upgrade to a new version of InterSystems IRIS. To instead perform a silent upgrade, review the information in the Unattended InterSystems IRIS Installation section of the “Installing on UNIX®, Linux, and macOS” chapter of this guide.

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

To perform the upgrade:

1. If your installation kit is in the form of a .tar file, you must first uncompress it (see the “Uncompress the Installation Kit” section in the “Installing on UNIX®, Linux, and macOS” chapter of this book).
2. As a user with root privileges, start the installation procedure by running the irisinstall script, located at the top level of the installation files:

   ```
   sudo sh /<pathname>/irisinstall
   ```

   where <*pathname*> is the location of the installation kit, typically the temporary directory to which you have extracted the kit.

3. The script displays a list of all existing InterSystems IRIS instances on the host.
4. At the **Enter instance name:** prompt, type the name of the InterSystems IRIS instance you want to upgrade, and press **Enter**.
5. When prompted for a license key file, type *keypath*\iris.key, where *keypath* is the location of the new iris.key file. If the installation script detects the new license key in the *install-dir*/mgr directory, the script proceeds without prompting you for the license key file.
6. The installation script summarizes the upgrade options and again asks you to confirm the upgrade. Press **Enter** to continue with the upgrade.

   Important: Do not interrupt the installation while it is in progress. If the upgrade fails with any error messages, correct the issues and restart the upgrade installation.

7. Examine messages.log, iboot.log, and ensinstall.log in the *install-dir*/mgr directory for any errors. If any fatal error is found, correct the error, and then run the installation script again.

After the upgrade finishes, the instance restarts and any installation manifest is run.

Important: If your operating system is configured to use huge memory pages, check the startup messages to make sure shared memory is being allocated in accordance with these settings. If you see a message similar to the following, reboot your server to avoid an out-of-memory situation.

```
Failed to allocate 1468MB shared memory using Huge Pages. Startup will retry with standard
 pages.
```

4.4 Upgrading a Mirror

This section provides instructions for upgrading InterSystems IRIS instances, an application, or both on the members of an InterSystems IRIS mirror.

As noted in the "Mirroring" chapter of the *InterSystems IRIS High Availability Guide*, all failover and DR async members of a mirror must be of the same InterSystems IRIS 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 InterSystems IRIS version as the failover members, although application functionality may require it; for more information, see InterSystems IRIS Instance Compatibility in the "Mirroring" chapter of *High Availability Guide*.

There are four mirror upgrade paths to choose from. To determine which procedure you should follow, review the factors in the Choosing Mirror Upgrade Procedure section below. The four procedures are located in the Mirror Upgrade Procedures section below.

Also review the following two sections about upgrading a mirror:

- Mirror Upgrade Terms, which defines some of the terms used in the mirror upgrade procedures.
- Adding Mirror Members During an Upgrade, which discusses when you should add members to a mirror.

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. For more information, see Starting the ISCAgent on Linux Systems in the "Mirroring" chapter of the *InterSystems High Availability Guide*.

When you upgrade InterSystems IRIS 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.

4.4.1 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 InterSystems IRIS, or to a new major version of InterSystems IRIS?
- 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 InterSystems IRIS version to another, perform application upgrades, or both, there is the possibility that your upgrade involves 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 are then replicated by the mirror on the backup and any async members.)

Important: Following a major upgrade, InterSystems recommends that you recompile all classes in all application namespaces on the instance and some routines must also be recompiled. 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.

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. Classes must be recompiled on the primary (even if the application has not been upgraded). Recompiling routines on the primary is recommended.
- Data in mirrored databases must be upgraded due to an application upgrade; these changes must take place on the primary.

Note: Since all Interoperability productions have mirrored application code, these conditions will always apply for mirrored environments with Interoperability productions.

If your major upgrade and/or application upgrade involves any of these conditions, use the Major Version 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 Version 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 Version 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 InterSystems IRIS instances, application code, or both; adapt individual steps depending on what you are upgrading.

Note: Your circumstances may call for additional steps and details in a given procedure. If you are still unsure which procedure is appropriate for you, or whether a particular procedure is appropriate as stated, please contact the InterSystems Worldwide Response Center (WRC).

4.4.2 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 do 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.

4.4.3 Mirror Upgrade Terms

In the procedures in this section, the following terms are used:

Upgrade classes and routines

Recompile all classes in all application namespaces on the instance. This should be done after an InterSystems IRIS major version upgrade (as described in Post-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 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). It is recommended that any existing routines are recompiled.
- Classes and routines stored in nonmirrored routines 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 nonmirrored routines database can also be “precompiled” by recompiling a copy of the database on a system that has already been upgraded to the target InterSystems IRIS release, then distributed to each mirror member following the upgrade.

System A

The mirror member that is *initially* the primary failover member.

System B

The mirror member that is *initially* the backup failover member.

System C

A newly-added DR async of the new InterSystems IRIS version that has been promoted to failover member.

Set no failover

Use 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 *InterSystems IRIS High Availability Guide* for instructions.

Perform a graceful shutdown

Use the **iris stop** command to shut the instance down cleanly (see Controlling InterSystems IRIS Instances in the “Using Multiple Instances of InterSystems IRIS” chapter of the *InterSystems IRIS System Administration Guide*).

Promote to failover member

Follow 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 *InterSystems IRIS High Availability Guide*.

Viewing the Mirror Monitor

Use 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 *InterSystems IRIS High Availability Guide* for more information.

4.4.4 Mirror Upgrade Procedures

There are four mirror upgrade paths to choose from. To determine which procedure you should follow, review the factors in the Choosing Mirror Upgrade Procedure section. The four procedures are:

- Maintenance Release Upgrade
- Major Version Upgrade (Mirrored Database Changes)
- Major Version Upgrade (No Mirrored Database Changes)
- Major Version Upgrade with Planned Downtime

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.

4.4.4.1 Maintenance Release Upgrade

If you are upgrading to an InterSystems IRIS maintenance release rather than a new major InterSystems IRIS 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 InterSystems IRIS. System B becomes backup.
    - Add a new backup of the new version:
        a. Install the new version of InterSystems IRIS 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 InterSystems IRIS. System A becomes backup.
7. If system C became primary and system B was demoted to DR async, upgrade system B.

4.4.4.2 Major Version Upgrade (Mirrored Database Changes)

If you are upgrading to a new major InterSystems IRIS 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 InterSystems IRIS. System B becomes backup.
  c. Upgrade nonmirrored classes and routines on system B, if any.
- Add a new backup of the new version:
  a. Install the new version of InterSystems IRIS 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.
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.
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 InterSystems IRIS. System A becomes backup.
11. Upgrade nonmirrored 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 nonmirrored classes and routines on system B, if any.

4.4.4.3 Major Version Upgrade (No Mirrored Database Changes)

If you are upgrading to a new major InterSystems IRIS 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 nonmirrored routines databases are always maintained. It can, however, be used even if the separate routines databases are normally mirrored by removing these databases from the mirror for the duration of the upgrade.

Important: If the routines 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 routines 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 routines 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 routines 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 InterSystems IRIS. 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 InterSystems IRIS 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 routines 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 InterSystems IRIS. System A becomes backup.
8. Upgrade any nonmirrored 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 nonmirrored classes and routines on system B, if any.
10. For any mirrored routines databases that you removed from the mirror on backup (B or C) before it became the new primary, do the following:
    a. Remove the routines 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 *InterSystems IRIS 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.

4.4.4.4 Major Version Upgrade with Planned Downtime

If you are upgrading to a new major InterSystems IRIS 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.
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 InterSystems IRIS. 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 nonmirrored 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 InterSystems IRIS. System B becomes backup.
     b. Upgrade nonmirrored classes and routines on system B, if any.
   - Add a new backup of the new version:
     a. Install the new version of InterSystems IRIS 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 InterSystems IRIS. System B becomes a DR async.
     e. Upgrade nonmirrored 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 nonmirrored classes and routines on system B, if any.
9. Restore user access to the mirror.

4.5 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 InterSystems IRIS 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 is 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 causes a pause in application activity, the length of which depends 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).

4.6 Post-Upgrade Tasks

The post-upgrade tasks required depend on whether you have upgraded to a new major version of InterSystems IRIS or to a maintenance release of the installed version. Also note the following points:

- If you changed an instance from 8–bit to Unicode during the upgrade, InterSystems IRIS does not automatically convert your databases. You need to do this conversion yourself; for more information, see the Database Portability section in the "Namespaces and Databases" chapter of *Orientation Guide for Server-Side Programming*.
- It is not necessary to re-import any OpenAPI 2.0 specification files.
- It is not necessary to re-import any Web Service Definition (WSDL) files.
- Cached queries are always purged during upgrade. They are recompiled and cached as needed.

4.6.1 Maintenance Release Post-Upgrade Tasks

When the upgrade is complete, if your system runs any productions, follow the instructions in the Starting a Production section of the "Starting and Stopping Productions" chapter in *Managing Productions* to restart the productions.

Typically, maintenance release upgrades do not require you to change external files and clients, or to recompile classes and routines. However, the Studio version on a client must be the same or later than the InterSystems IRIS server version to which it connects.

If you choose to recompile, review the information in the Major Version Post-Installation Tasks section.

4.6.2 Major Version Post-Installation Tasks

After upgrading InterSystems IRIS to a new major version, it is important to follow the guidance provided in the General Upgrade Information in the *Release Notes*. You must also 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 and all routines in each namespace. See How to Compile Namespaces for instructions. 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.
- *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 *InterSystems IRIS Language Bindings* set.

  This item does not apply to web services and web clients; it is not necessary to re-import 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 InterSystems IRIS and may 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:

- *Review query plans.* When you upgrade to a new major version, existing Query Plans are automatically frozen. This ensures that a major software upgrade will never degrade the performance of an existing query. For performance-critical queries, you should test if you can achieve improved performance.

- *Restart productions* — If your system runs any productions, follow the instructions in the Starting a Production section of the “Starting and Stopping Productions” chapter in *Managing Productions*.
- *Upgrade Studio clients* — The Studio version on a client must be the same or later than the InterSystems IRIS server version to which it connects.
- *Upgrade the Web Gateway* — If your Web Gateway is on a separate machine from the InterSystems IRIS server you are upgrading, you must also upgrade the Web Gateway on that separate machine. You can accomplish this by following the Windows Upgrade Procedure in this chapter, or performing a silent installation with the **ADDLOCAL** property as discussed in Running an Unattended Installation section of the “Installing InterSystems IRIS on Microsoft Windows” chapter of this book.
- *Update web server files* — Following upgrades you must deploy these using established practices at your site.

4.6.2.1 How to Compile Namespaces

After an upgrade, you must compile the code in each namespace so that it runs under the new version of InterSystems IRIS. If the compiler detects any errors, you may need to recompile one or more times for the compiler to resolve all dependencies.

After your code compiles successfully, you may want to export any updated classes or routines, in case you need to run the upgrade in any additional environments. Importing these classes or routines during future upgrades allows you to update your code quickly, minimizing downtime.

Note: If you are using a manifest as part of your upgrade process, you can compile your code from within the manifest. See the instructions in the Perform Post-Upgrade Tasks section of the Creating and Using an Installation Manifest appendix of this guide.

Compiling Classes

To compile the classes in all namespaces from the Terminal:

```
do $system.OBJ.CompileAllNamespaces("u")
```

To compile the classes in a single namespace from the Terminal:

```
set $namespace = "<namespace>"
do $system.OBJ.CompileAll("u")
```

Note: If your namespaces contain mapped classes, include the `/mapped` qualifier in the call to **CompileAllNamespaces()** or **CompileAll()**:

```
do $system.OBJ.CompileAllNamespaces("u /mapped")
do $system.OBJ.CompileAll("u /mapped")
```

Class compiler version utility

To assist customers in determining which class compiler version a class or classes in a namespace have been compiled with, InterSystems provides two assists

- Method – $System.OBJ.CompileInfoClass(<classname>)

  This method returns the version of the class compiler used to compile this <classname> and the datetime the class was compiled

- Query – $System.OBJ.CompileInfo(<sortby>)

  This query generates a report for the current namespace that includes all classes, the version of the compiler used to compile each one, and the datetime each was compiled. The first argument <sortby> may have the following values:

  - 0 – the time the class was compiled

  - 1 – the class name
  - 2 – the version of InterSystems IRIS the class was compiled in

Compiling Routines

To compile the routines in all namespaces from the Terminal:

```
do ##Class(%Routine).CompileAllNamespaces()
```

To compile the routines in a single namespace from the Terminal:

```
set $namespace = "<namespace>"
do ##Class(%Routine).CompileAll()
```

A

Creating and Using an Installation Manifest

This topic describes how to use the %Installer utility to create an installation manifest that describes a specific InterSystems IRIS® data platform configuration and use it to generate code to configure an InterSystems IRIS instance.

A.1 Overview of an Installation Manifest

The %Installer utility lets you define an installation manifest that describes and configures a specific InterSystems IRIS 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 the 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. This appendix provides installation manifest examples that you can copy and paste to get started.

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.

A.2 Creating an Installation Manifest

This section contains the information needed to create an installation manifest, broken into the following subjects:

- Manifest Class Definition
- Common Manifest Options
- Variables Within <Manifest> Tags
- List of <Manifest> Tags

A.2.1 Manifest Class Definition

To create a class that defines an installation manifest, create a class that meets the following criteria (or start from the blank template in the Installation Manifest Examples section below):

- The class must include the %occInclude include file.
- The class must contain an XData block specifying the details of the installation. The name of the XData block will be used later as an argument. The root element in the XData block must be `<Manifest>`. For details, see "List of <Manifest> Tags".

  In Studio, if you include `[XMLNamespace = INSTALLER]` after the name of the XData block, Studio provides assistance as you type the XData block.
- The class must define a **setup()** method that refers to the XData block by name, as seen in the Installation Manifest Examples section.

A.2.2 Common Manifest Options

This section describes some of the common tasks an installation manifest is used to perform. The following options are described:

- Define Namespaces
- Add Users and Passwords
- Write to the Manifest Log
- Perform Post-Upgrade Tasks

A.2.2.1 Define Namespaces

To define namespaces, add any number of `<Namespace>` tags within the `<Manifest>` tag.

If you want to define databases or mappings for a namespace, put a `<Configuration>` tag inside the `<Namespace>` tag. For each database the namespace contains, add a `<Database>` tag within the `<Configuration>` tag. To define mappings, add `<GlobalMapping>`, `<RoutineMapping>`, and `<ClassMapping>` tags beneath the appropriate `<Database>` tag. The `</Configuration>` tag activates the defined mappings.

Also within a `<Namespace>` tag, you can load globals, routines, and classes using the `<Import>` tag. You can also invoke class methods, which in turn can execute routines and access globals that have been imported, with the `<Invoke>` tag.

You can define variables with the `<Var>` tag. To reference a variable within the manifest, use the `${var_name}` syntax. For more information, see Variables Within <Manifest> Tags below.

A.2.2.2 Add Users and Passwords

There are multiple ways to add users (including their roles and passwords) to the installed instance:

- Include a `<User>` tag in an installation manifest, as described in the List of <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 InterSystems IRIS or a web service; the problem with these approaches is that the server that is installing InterSystems IRIS 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.
- Edit users in the Management Portal after installation is complete, as described in Users.
- Using the Security.Users class on a staging instance of InterSystems IRIS, 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.

A.2.2.3 Write to the Manifest Log

You can define messages to be added to the manifest 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 -1 to 3, where -1 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 appendix). 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.)

A.2.2.4 Perform Post-Upgrade Tasks

When upgrading an instance of InterSystems IRIS, you can use an installation manifest to automatically perform any necessary post-upgrade tasks, such as recompiling code. You do this by using the `<Invoke>` tag to call the class methods described in the How to Compile Namespaces section of the “Upgrading InterSystems IRIS” chapter of this book.

See the Installation Manifest Examples section of this appendix for an example of a manifest that can be used during an upgrade.

A.2.3 Variables Within <Manifest> Tags

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>}` — Variables

Expands to the value of the variable. In addition to the predefined variables described in this section, you can specify additional variables with the <Var> tag.

`${#<param_name>}` — Class Parameters

Expands to the value of the specified parameter for the manifest class.

`#{<ObjectScript_expression>}` — ObjectScript Expressions

Expands to the specified InterSystems IRIS Object Script expression (which must be properly quoted).

Note: Parameter expressions are expanded at compile time, which means they can be nested within variable and ObjectScript expressions. Additionally, variable expressions are expanded before ObjectScript expressions and thus can be nested within the latter.

The following table lists the predefined variables that are available to use in the manifest:

Variable Name	Description
SourceDir	(Available only when the installer is run) Directory from which the installation (setup_irisdb.exe or irisinstall) 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 InterSystems IRIS is installed.
MGRDIR	Manager (mgr) directory.
PLATFORM	Operating system.
PORT	InterSystems IRIS superserver port.
PROCESSOR	Processor chip.
VERSION	InterSystems IRIS version number.

A.2.4 List of <Manifest> Tags

All the information for code generation is contained in the outermost XML tag, `<Manifest>`. The tags within `<Manifest>` describe a specific configuration of InterSystems IRIS. This section contains a list of these tags and their function. Not all tags and attributes are listed here; for a complete list, see the %Installer class reference documentation. The default value for an attribute, if any, is listed in square brackets next to each attribute.

Important: Null arguments cannot be passed in tags in a manifest class. For example, the `<Arg/>` tag passes an empty string, equivalent to `<Arg=""/>`, not null.

- <Arg>
- <ClassMapping>
- <Compile>
- <Configuration>
- <CopyClass>
- <CopyDir>
- <CopyFile>
- <CSPApplication>

- <Credential>
- <Database>
- <Default>
- <Else>
- <Error>
- <ForEach>
- <GlobalMapping>
- <If>
- <IfDef>
- <IfNotDef>
- <Import>
- <Invoke>
- <LoadPage>
- <Log>
- <Manifest>
- <Namespace>
- <Production>
- <Resource>
- <Role>
- <RoutineMapping>
- <Setting>
- <SystemSetting>
- <User>
- <Var>

`<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}\messages.log"
    Target="F:\${INSTANCE}_log"
    IgnoreErrors="0"/>
```

`<CSPApplication>`

Parent tag: <Namespace>

Defines one or more web applications, as defined within the **Security.Applications** class. (Also see Create and Edit Applications 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 in to 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\IRIS\CSP\Democode1"
    Resource=""
    Grant="%DB_%DEFAULT"
    Recurse="1"
    LoginClass=""
    CookiePath="/csp/demo1"
    AuthenticationMethods="64"/>
```

`<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"/>
```

`<Database>`

Parent tag: <Configuration>

Defines a database within a namespace.

See Configuring Databases in the “Configuring InterSystems IRIS” chapter of the *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\iris.key")}
'>
<Else/>
<CopyFile Src="C:\\InterSystems\key_files\iris300.key" Target="${MGRDIR}\iris.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,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 [`IRIS 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="iris.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 -1 (none) to 3 (verbose).
- Text—Log message (string up to 32,000 characters in length).

See Write to the Manifest Log 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—interoperability-enabled namespace? [0]

(Other properties are applicable to Interoperability 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 a 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 routine 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 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 inherits its **Modify()** method from Config.CommonSingleMethods.

- Name—*Class.property* of the Config class.
- Value—Value to assign to property.

`<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 (see <Var> below).
- 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 is 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}"/>
```

A.3 Using the Manifest

You can run a manifest manually, or as part of an installation.

A.3.1 Manually

In the `%SYS` namespace, enter the following command in the Terminal:

```
%SYS>do ##class(MyPackage.MyInstaller).setup()
```

You can pass an array of variables to the **setup()** method by reference. Each subscript is used as a variable name, and the node as the value. For example:

```
%SYS>set vars("SourceDir")="c:\myinstaller"
%SYS>set vars("Updated")="Yes"
%SYS>do ##class(MyPackage.MyInstaller).setup(.vars,3)
```

The second argument in this example (`3`) is the log level.

A.3.2 During Installation

Export the manifest class as `DefaultInstallerClass.xml` to the same directory where the InterSystems IRIS install (either .msi, setup_irisdb.exe, or irisinstall) is run. It is imported into `%SYS` and compiled, and the **setup()** method is executed.

Note: Make sure that the user running the installation has write permission to the manifest log file directory. The installation does not proceed if the log file cannot be written.

In addition, if you are installing with a single-file kit, you will need to extract the files from the .exe file into the installation directory to get the expected behavior.

Note that if you use the export technique, you cannot pass arguments directly to the **setup()** method. The following sections describe how to pass arguments on Windows or on UNIX®, Linux, and macOS:

A.3.2.1 On 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, variables, 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.

For more information, see the Command Line Reference section in the “Installing InterSystems IRIS on Microsoft Windows” chapter of this book.

A.3.2.2 On UNIX®, Linux, and macOS

On UNIX®, Linux, and macOS systems, you can set environment variables to define the location of the exported manifest class, variables, log file name, and log level prior to running either **irisinstall** or **irisinstall_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"
./irisinstall
```

For more information, see the Unattended Installation Parameters section in the “Installing InterSystems IRIS on UNIX®, Linux, and macOS” chapter of this book.

A.4 Installation Manifest Examples

This section contains examples of installation manifests that do the following:

- Blank Template
- Create a Namespace
- Compile After an Upgrade

A.4.1 Blank Template

The following is a basic manifest template:

Class Definition

```
Include %occInclude

/// Simple example of installation instructions (Manifest)
Class MyInstallerPackage.SimpleManifest
{

XData SimpleManifest [ XMLNamespace = INSTALLER ]
{
<Manifest>
   <Log Level="3" Text="Start manifest" />
   <Namespace Name="">
      <Import File="" />
      <Invoke Class="" Method="" CheckStatus="" Return="">
         <Arg Value=""/>
      </Invoke>
   </Namespace>
   <CopyFile Src="" Target="" IgnoreErrors=""/>
   <CopyDir Src="" Target="" IgnoreErrors=""/>
   <Log Level="3" Text="End manifest" />
</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")
}

}
```

A.4.2 Create a Namespace

The manifest in this example creates a namespace (MyNamespace) and three databases. The MyDataDB and MyRoutinesDB databases are the default databases for globals and routines, respectively, while the MyMappingDB database is another globals database. The `<GlobalMapping>` tag creates a mapping to MyMappingDB for `t*` globals in the MyNamespace namespace, which overrides the default mapping.

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")
}

}
```

A.4.3 Compile After an Upgrade

The manifest in this example recompiles code in a namespace called APPTEST, and is intended to be run after an upgrade. It performs the following tasks:

1. Writes a message to the manifest log indicating the start of the manifest installation process.
2. Sets the current namespace to APPTEST, which contains the code that needs to be recompiled after the upgrade.
3. Invokes the method **%SYSTEM.OBJ.CompileAll()** to compile the classes in the namespace.
4. Invokes the method **%Routine.CompileAll()** to compile the custom routines in the namespace.
5. Writes a message to the manifest log indicating the end of the manifest installation process.

Class Definition

```
Include %occInclude

/// Simple example of installation instructions (Manifest)
Class MyInstallerPackage.SimpleManifest
{

XData SimpleManifest [ XMLNamespace = INSTALLER ]
{
<Manifest>
    <Log Level="3" Text="The Installer Manifest is running."/>
    <Namespace Name="APPTEST">
        <Invoke Class="%SYSTEM.OBJ" Method="CompileAll" CheckStatus="1">
            <Arg Value="u"/>
        </Invoke>
        <Invoke Class="%Routine" Method="CompileAll" CheckStatus="1"/>
        </Namespace>
    <Log Level="3" Text="The Installer Manifest has completed."/>
</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")
}

}
```

B

Adding UNIX® Installation Packages to an InterSystems IRIS Distribution

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

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

B.1 Tutorial

Suppose we have written an InterSystems Callout shared library (see the "Creating an InterSystems Callout Library" chapter in *Using the Callout Gateway*) to connect to an imaging device named Foo9000. We compile this library as libfoo9000.so and want to install it with InterSystems IRIS. 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 is stored in a configuration file in the InterSystems IRIS instance's *install-dir*\mgr directory.

We start with an existing InterSystems IRIS kit:

```
~/kit:>ls
irisinstall   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 *install-dir*\mgr directory.
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 irisinstall, irisinstall_client or irisinstall_silent scripts (as described in InterSystems IRIS 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 InterSystems IRIS Installation Parameter File Variables table at the end of this appendix.

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 processes ours. In this case, we want our library to be installed and configured any time an InterSystems IRIS 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 **irisinstall**, our package (foo9000) is configured and installed:

```
~/kit:>sudo ./irisinstall
Your system type is 'SuSE Linux Enterprise Server 10 (x64)'.
Currently defined instances:
IRIS instance 'INSTANCE1'
directory: /home/testUser/INSTANCE1
versionid: 2018.1.0.508.0
conf file: iris.cpf (SuperServer port = 1972, WebServer = 57785)
status:  crashed, last used Sat Sep 22 08:37:32 2018
Enter instance name: INSTANCEPACK1
Do you want to create IRIS 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
```

B.2 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 II–1: InterSystems IRIS Installation Parameter File Variables

Variable name	**Description** **`(Valid values) or Example`**
dist.source_dir	Source directory of the installation media. `/iriskit`
legacy_dist.source_dir	For legacy purposes, source directory of the installation media. `/iriskit`
product_info.version	InterSystems product version number. `2018.1.0.100.0`
product_info.name	Name of InterSystems product. `InterSystems IRIS`
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/IRIS`
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. `IRIS`
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 InterSystems IRIS registry directory (must be on a local filesystem). `/usr/local/etc/irissys`
server_location.iris	Directory in which **iris** resides during installation. `/iriskit/dist/lnxrhx64/bin/shared/iris`
server_location.is_dr_mirror	Whether this instance is a disaster recovery (DR) mirror member. `(N/Y)`
*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.iris_user	Effective user for the InterSystems IRIS superserver `irisusr`
security_settings.iris_group	Effective group for InterSystems IRIS. `irisusr`
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 is cleared before the parameter file is stored. 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/LockedDown)`
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 **irisinstall**.

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. `(-1/0/1/2/3)`
installer.manifest_logfile	Specifies the log file name. `/manifests/IRIS-installManifestLog.txt`
port_selection.superserver_port	Superserver port number. `1972`
port_selection.webserver_port	Web server port number. `52773`
port_selection.jdbcgateway_port	Java Database Connectivity (jdbc) gateway port number. `62972`
csp_gateway.configure	Indicates whether or not to configure the Web Gateway for an external web server. `(N/Y)`
csp_gateway.web_server_type	Type of existing web server for the Web 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`
csp_gateway.sunone_server	Location of the Sun ONE server for the Web Gateway to use. `/usr/netscape/server4/httpd-production`
csp_gateway.directory	Directory to contain the Web Gateway files.

Variable name	**Description** **`(Valid values) or Example`**
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/IRIS`
client_location.is_client_install	Indicates whether or not it is a client install. `(N/Y)`
*install**	`database_server`
*postinstall**	`database_server`
japanese_docs.install	Indicates whether or not to install the Japanese documentation sources. `(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
- engine_link_libraries
- light_cpp_binding
- addenda
- install_confirmation
- copyright