InterSystems IRIS In-Place Conversion Guide

Version 2020.1
2020-08-04

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

InterSystems IRIS In-Place Conversion Guide
InterSystems IRIS Data Platform Version 2020.1 2020-08-04

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

Introduction

This document describes how to convert an existing instance of InterSystems Caché®, InterSystems Ensemble®, or InterSystems HealthShare® Health Connect/HSAP (based on the Caché/Ensemble platform) to the appropriate product based on InterSystems IRIS® data platform.

You can use this document to convert an existing instance to:

- InterSystems IRIS
- InterSystems IRIS for Health™
- Health Connect 2019.1 or later (based on the InterSystems IRIS platform)

Not every possible conversion scenario is supported. See Supported Conversion Paths for details.

The conversion process may require some changes to application code, configuration scripts, and other procedures, but will be relatively easy for the majority of cases. As with any major upgrade, you should thoroughly test your custom code, including any business services, processes, and operations, in a test environment before deploying to a live production environment.

Before attempting a conversion, review the *InterSystems IRIS Adoption Guide* for background information on the differences between Caché or Ensemble and InterSystems IRIS. You can download the guide from the InterSystems Worldwide Response Center documents distribution page.

Depending on which features your instance uses, some sections of this document may not apply to your environment.

CAUTION: The conversion process cannot be undone. It is essential that you make a complete backup of your system before attempting the conversion.

If you are uncertain about any aspect of the conversion process, InterSystems recommends that you contact the InterSystems Worldwide Response Center for assistance.

Note: Please send any issues or feedback to conversionft@intersystems.com.

Requirements

This section summarizes the valid conversion paths from Caché/Ensemble-based products to InterSystems IRIS-based products and the supported starting and destination versions for each product. In all cases, you must have the proper installation kit and a valid license key for the destination product to which you are converting.

Supported Conversion Paths

The left column of the following table lists the products that can be converted to an InterSystems IRIS-based product. The remaining columns show, for each starting product, if a conversion can be performed to get to a given destination product.

Table 1: Conversion Matrix

Starting Product	InterSystems IRIS (2019.1.1 or later)	InterSystems IRIS for Health (2020.1 or later)	Health Connect (on InterSystems IRIS)
Caché (2016.2 or later)	YES	YES	NO
Ensemble (2016.2 or later)	YES	YES	YES (2020.1 or later)
Health Connect/HSAP (15.03x on Ensemble 2017.1 or later)	NO	YES	YES (2019.1 or later)

Important: If you are using Ensemble as a platform for healthcare, make note of the following:

- InterSystems IRIS does not support the HL7 and the DICOM features that are available in Ensemble. This support is included in the Health Connect and InterSystems IRIS for Health products.
- If you are using Ensemble as an integration engine and have no plans to use it as a general-purpose data platform for healthcare, perform an in-place conversion to Health Connect 2020.1 or later (based on the InterSystems IRIS platform).
- If you are using Ensemble as a general-purpose data platform for healthcare, or plan to in the future, perform an in-place conversion to InterSystems IRIS for Health 2020.1 or later.

Important: If you are using Health Connect (HSAP), make note of the following:

- If you are using Health Connect as an integration engine and have no plans to use it as a general-purpose data platform for healthcare, perform an in-place conversion to Health Connect 2019.1 or later (based on the InterSystems IRIS platform).
- If you are using Health Connect as a general-purpose data platform for healthcare, or plan to in the future, perform an in-place conversion to InterSystems IRIS for Health 2020.1 or later.

Supported Versions

This section lists the versions for each starting and destination product for which in-place conversion is supported.

The supported versions apply to both mirrored and non-mirrored configurations. Container-only releases are not included in the supported versions.

Starting Products

- Caché 2016.2 or later
- Ensemble 2016.2 or later
- Health Connect (HSAP) 15.03, 15.031, or 15.032 based on Ensemble 2017.1 or later

Destination Products

- InterSystems IRIS 2019.1.1 or later
- InterSystems IRIS for Health 2020.1 or later
- Health Connect 2019.1 or later (based on InterSystems IRIS)

Supported Platforms

Some older platforms that are supported for Caché or Ensemble are no longer supported for InterSystems IRIS. Read Supported Platforms for the version of InterSystems IRIS to which you are converting. You may need to upgrade your operating system to one that InterSystems IRIS supports.

In-place conversion is supported for server platforms only, not for platforms that are supported solely for development purposes.

Limitations

Mirroring

To support mixed mirror configurations between Caché/Ensemble-based instances and InterSystems IRIS-based instances, Caché/Ensemble must have a specific enhancement (SML2736). SML2736 enables a Caché/Ensemble instance to discover an InterSystems IRIS failover mirror and is available with Caché/Ensemble 2018.1.3 or later. If you are running an earlier version, please contact the InterSystems Worldwide Response Center for assistance.

These mixed mirror configurations are useful for a limited number of scenarios, such connecting a Caché/Ensemble-based reporting async to an InterSystems IRIS-based failover mirror. Both the primary and backup members of the failover mirror should be converted to InterSystems IRIS using the procedure described in Performing a Conversion in a Mirrored Environment, which does not require the presence of SML2736.

Enterprise Cache Protocol (ECP)

Connecting an InterSystems IRIS instance to a Caché or Ensemble instance using ECP is fully supported for data (global) access only. Support for routines and classes in mixed ECP configurations is not currently available.

KMIP Server

InterSystems IRIS does not currently support in-place conversion of instances that use the Key Management Interoperability Protocol (KMIP) for managed key encryption.

Windows Cluster

InterSystems IRIS does not currently support in-place conversion of instances running on Windows Cluster.

InterSystems IRIS Already Installed

In certain cases, if you attempt to perform an in-place conversion of an Caché or Ensemble instance on a machine that already contains an instance of InterSystems IRIS, or you attempt to install Caché or Ensemble on a machine that already contains an instance of InterSystems IRIS, conflicts may occur. See Issues with Caché or Ensemble and InterSystems IRIS on the Same Machine for details.

About This Book

This document describes how to:

- Perform any needed pre-conversion tasks.
- Perform a conversion in a non-mirrored environment.
- Perform a conversion in a mirrored environment.
- Use a manifest to load code or files during the in-place conversion process.

The document also contains appendices that:

- Detail the changes made during the conversion process.
- List the classes that are updated or re-implemented in Health Connect 2019.1 and InterSystems IRIS for Health.
- Describe memory considerations that apply when converting from Caché 2016.2 or Ensemble 2016.2 to InterSystems IRIS for Health 2020.1 or later
- Provide guidance on converting a disaster recovery async member
- Describe issues you need to be aware of if you are installing Caché or Ensemble and InterSystems IRIS on the same machine
- Describe how to convert a Health Connect/HSAP instance on Windows that was previously upgraded from a full HealthShare instance
- List the changes made to this document

1

Performing Pre-Conversion Tasks

Perform these tasks before you convert your instance of Caché or Ensemble to InterSystems IRIS:

1. *Read the Supported Platforms for the version of InterSystems IRIS to which you are converting* — You may need to upgrade your operating system to one that InterSystems IRIS supports.
2. *Review the InterSystems IRIS Adoption Guide* — This guide will supply you with background information on the differences between Caché or Ensemble and InterSystems IRIS. You can download the guide from the InterSystems Worldwide Response Center documents distribution page.
3. *Identify any classes you are running in deployed mode* — Before performing a conversion, you must import the source code for any classes that are running in deployed mode, as it must be compiled under InterSystems IRIS. (See Compile Namespaces.) You can again place these classes under deployed mode after the conversion is complete. Alternatively, you can load a new, compiled version of your application with an installation manifest. (See Using an Installation Manifest.) If the source code is not available to recompile, or the vendor has not supplied a new version of the application, you will not be able to run your application after the conversion.
4. *Identify any custom routines (*.mac or *.int) for which you do not have source code loaded* — If you have any custom routines (those not generated from classes), the source code must be available so that it can be compiled under InterSystems IRIS. (See Compile Namespaces.) Alternatively, you can load new, compiled versions of these items with an installation manifest. (See Using an Installation Manifest.) Recompiling routines under InterSystems IRIS ensures optimum performance.
5. *Identify any Caché Server Pages (*.csp) that must be reloaded from disk* — Make sure that source code for any CSP pages is available, as it must be reloaded from disk and compiled under InterSystems IRIS. (See Compile Namespaces.) Alternatively, you can load new, compiled versions of these items with an installation manifest. (See Using an Installation Manifest.)
6. *Decide whether to create an installation manifest* — If your application relies on code libraries, CSS files, JavaScript files, or other components that are not included as part of InterSystems IRIS, you can use a manifest to install them as part of the in-place conversion process. See Using an Installation Manifest.
7. *Determine whether you have 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 InterSystems IRIS and use an installation manifest to install it as part of the in-place conversion process. See Using an Installation Manifest.

   CAUTION: If your application calls any user code at instance startup and you do not install updated code with an installation manifest, the instance will not start up after the installation script or installer completes.

8. *Validate your application in InterSystems IRIS* — Validate your application in a test instance of InterSystems IRIS before performing the conversion. This ensures that your application will run on the new platform.

9. *Obtain a InterSystems IRIS license key* — See the "Managing InterSystems IRIS Licensing" chapter of the *System Administration Guide* for more information.

10. *Validate the Configuration Parameter File* — Validate that the CPF file is syntactically correct. You can validate the CPF file by installing a test instance of the target version of InterSystems IRIS and running the class method ##Class(Config.CPF).Validate(CPFFile) from the Terminal, where CPFFile is the name of the CPF file, including the path.

11. *Check system integrity* — Run a system integrity check on existing directories to ensure there is no corruption in any of the databases. See the "Introduction to Data Integrity" chapter of the *Data Integrity Guide* for more information.

12. *Back up the system* — Make a complete backup of the system that is running the Caché or Ensemble instance. Use your customary full operating system backup procedures. If the conversion is not successful, you may need to restore from this backup. As a best practice, InterSystems recommends that you restore this backup to another system and run a system integrity check to make sure that it is good.

13. *Save custom classes, routines and globals* — To prevent your own classes, routines and globals in the %SYS namespace from being affected by the conversion, 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 during the conversion.

    The CACHELIB, CACHETEMP, and CACHE databases are completely replaced.

    The SAMPLES, ENSDEMO, and DOCBOOK databases are deleted.

    Any .mac or .inc routines are not affected during the upgrade.

14. *Save user files* — To ensure that your user files are not deleted or replaced during the conversion, save them in the *install-dir*/mgr/user directory, which is the only directory that is not subject to modification during conversion (that is, except for the *install-dir*/mgr/user directory, any files and directories in the install directory may be deleted or replaced during conversion).

15. *Make note of whether your operating system is using huge or large memory pages* — Check your cconsole.log file to see if you have configured one of these memory settings. When you restart your instance during the conversion process, you will need to make sure the instance is allocating memory in accordance with your configuration. (If the instance cannot allocate memory using this configuration, a server reboot is required to prevent an error condition from occurring.)

    Linux example:

    ```
    Allocated 1342MB shared memory using Huge Pages: 1024MB global buffers, 35MB routine buffers
    ```

    Windows example:

    ```
    Allocated 8920MB shared memory (large pages): 8192MB global buffers, 256MB routine buffers
    ```

16. *Check the Windows Service controller logon information* — If you are on a Windows system, open Windows Services and find the service Caché Controller (or Ensemble Controller) for *instname*. Right-click the service, click **Properties**, and then click the **Log On** tab. If the tab specifies the Local System account or an account of the form *domainname\username*, no action is necessary. If the account is of the form *username@domainname*, make a note of this account and make sure that you know the password associated with the account. You will need to reenter this information after the installation finishes.

17. *Check user privileges* — For conversions run on Linux, make sure the user who is performing the conversion has **sudo** privileges, as some commands must be run with root access.

18. *Check available disk space* — Your system must have free disk space of at least 200 MB plus the size of the file /<installdir>/mgr/cache.dat prior to performing the conversion. The installation script or installer will not attempt the conversion if it detects that insufficient disk space is available.

19. *Check license server* — If your instances use a remote license server, note that this license server will also need to be converted to InterSystems IRIS, as a Caché or Ensemble-based license server cannot be used with InterSystems IRIS. Alternatively, you can set up a new InterSystems IRIS license server and use it as the remote license server.

20. *If you are converting a Health Connect/HSAP instance on Windows, check to see if the instance was previously upgraded from a full HealthShare instance* — If this situation applies to you, or you think it might, read Converting a Health Connect/HSAP Instance That Was Previously Upgraded from a Full HealthShare Instance before performing the conversion.

2

Performing a Conversion in a Non-Mirrored Environment

Use this checklist to perform a conversion in a non-mirrored environment. InterSystems recommends printing this checklist and marking off each step when you have completed it. Complete all steps in the order specified.

Table 2–1: Non-Mirrored Environment Conversion Checklist

Step	Done	Description and Link
1		Perform pre-conversion tasks
2		Uncompress the installation kit (Linux only)
3		Turn off system-wide production auto-start (Ensemble or Health Connect only)
4		Stop all Ensemble productions (Ensemble or Health Connect only)
5		Shut down the Caché or Ensemble instance
6		Run the installation script or installer
7		Upgrade FHIR Repositories (If converting to InterSystems IRIS for Health only)
8		Update user code, compile namespaces, and start productions
9		Turn on system-wide production auto-start

CAUTION: Make sure you have validated the conversion in a development or test environment and fully tested your application under InterSystems IRIS before running the conversion on a production instance.

Important: After you convert the instance to InterSystems IRIS and it is fully validated, InterSystems recommends that you make a complete backup of the system. Use your customary full operating system backup procedures.

Important: If you are converting from Caché 2016.2 or Ensemble 2016.2 to InterSystems IRIS for Health 2020.1 or later, see InterSystems IRIS for Health Memory Considerations for needed post-conversion steps.

Note: If you are performing the InterSystems IRIS conversion on Linux, use **sudo** to run any commands that require root permissions.

Note: To launch the Terminal in InterSystems IRIS, use **iris terminal <instname>**, instead of **csession <instname>**.

On Windows, run the command from the *install-dir*\bin directory (or with *install-dir*\bin in the PATH environment variable). Or, launch the Terminal using the System Tray icon.

2.1 Perform Pre-Conversion Tasks

To ensure a smooth and efficient conversion, make sure you have performed the pre-conversion tasks before starting the conversion.

2.2 Uncompress the Installation Kit (Linux Only)

If your installation kit is in the form of a .tar file, for example IRIS-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
# gunzip -c /download/IRIS-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-lnxrhx64 in the preceding example.

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

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

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

2.3 Turn Off System-Wide Production Auto-Start (Ensemble or Health Connect Only)

As the conversion process leaves the instance running upon completion, turn off production auto-start so that Ensemble productions remain off after the conversion.

1. In the Management Portal, select **System Administration** > **Configuration** > **Additional Settings** > **Startup**.
2. On the Startup Settings page, next to the **EnsembleAutoStart** setting, click **Edit**.
3. Clear the **EnsembleAutoStart** check box.
4. Click **Save**.

2.4 Stop All Ensemble Productions (Ensemble or Health Connect Only)

To prevent Ensemble productions from entering a problem state, stop all productions before shutting down the instance. This is an InterSystems best practice and allows large transactions to complete and remaining messages in production queues to be processed.

Perform the following procedure in each namespace that contains Ensemble productions:

1. In the Management Portal, select **Ensemble** > **List** > **Productions**.
2. On the Production List page, click the production that you want to stop.
3. Click **Open**.
4. On the Production Configuration page, click **Stop**.
5. When the browser asks if you wish to stop this production, click **OK**.

   Note: If any Terminal windows open as a result of stopping the production, do not close them.
6. When the Stop Production dialog box shows it is "Done," click **OK**.
7. If portal displays a message "Production could not stop, do you want to force a shut down?", click **Yes - Force to Shut Down**.

CAUTION: A production acquires the `Suspended` status when, at the end of the shutdown sequence, some queues still contain any messages. If the production becomes `Suspended`, contact the InterSystems Worldwide Response Center before proceeding.

A production acquires a status of `Troubled` if the production did not shut down properly. On the Production Configuration page, click **Recover** to shut down and clean up the production. If this does not clean up the production, you may need to run the following method in the Terminal:

```
do ##class(Ens.Director).RecoverProduction()
```

Note: If you have a large number of productions and stopping them individually would be very time consuming, run the following command from the Terminal:

```
do ##class(%EnsembleMgr).OnSystemShutdown(.pConsoleLog)
```

Then run the following command to make sure all productions have been stopped:

```
zw pConsoleLog
```

You should see messages similar to the following:

```
pConsoleLog=2
pConsoleLog(1)=$lb(0,"Stopping Ensemble production in namespace 'MYNAMESPACE initiated at
15:03:32")
pConsoleLog(2)=$lb(0,"All Ensemble productions stopped.")
```

2.5 Shut Down the Caché or Ensemble Instance

Shut down the Caché or Ensemble instance from the command line:

```
# ccontrol stop <instname>
```

On Windows, run the command from the *install-dir*\bin directory (or with *install-dir*\bin in the PATH environment variable). Or, stop the instance using the System Tray icon.

CAUTION: Prior to performing a conversion on a Caché or Ensemble instance, it is essential that the instance be shut down cleanly. To verify that the shutdown was clean, examine the cconsole.log file after the shutdown finishes. If the log contains entries similar to the following, then the shutdown was clean:

```
Journal restore not required at next startup
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 conversion.

2.6 Run the Installation Script or Installer

Now that the instance is down, you are ready to perform the installation.

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

Note: If you are using a manifest as part of your conversion process, see the instructions in Using a Manifest, instead of performing the standard installation step.

On Linux

The installation script, **irisinstall**, provides all the functionality required to convert an existing Caché or Ensemble instance to InterSystems IRIS.

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

To perform the conversion:

1. 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, as described in "Uncompress the Installation Kit".

2. The script displays a list of any existing Caché, Ensemble, InterSystems IRIS, and Health Connect instances on the host.

3. At the **Enter instance name:** prompt, type the name of the Caché or Ensemble instance you want to convert to InterSystems IRIS, and press **Enter**.

4. When asked if you want to convert the instance to InterSystems IRIS, press **Enter** to confirm. (An existing Health Connect 15.03 instance may be identified as "HealthShare.")

5. When prompted for a license key file, type *keypath*\iris.key, where *keypath* is the location of the new iris.key file.

   Note: If the installation script finds the new key file in the *install-dir*/mgr directory, the script will proceed without prompting you for the license key file.

6. If you are converting a Health Connect instance, when prompted, optionally configure the Web Gateway.
7. The installation script summarizes the conversion options and again asks you to confirm the conversion. Press **Enter** to continue with the conversion.
8. Examine messages.log (formerly known as cconsole.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 conversion finishes, the instance restarts, and any user installation manifest is run. For more information on installation manifests, see "Using an Installation Manifest".

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.
```

Then, start the InterSystems IRIS instance from the command line:

```
# iris start <instname>
```

On Windows

The installer provides all the functionality required to convert an existing Caché or Ensemble instance to InterSystems IRIS.

To perform the conversion:

1. Double-click the installer file, for example, IRIS-win_x64.exe.
2. The installer displays a list of any existing Caché, Ensemble, InterSystems IRIS, and Health Connect instances on the host. Any instances that can be converted are marked with the text `(CONVERSION)`.
3. Select the name of the Caché or Ensemble instance you want to convert to InterSystems IRIS, and click **OK**.
4. When prompted for an IRIS license key, click **License** and browse for the new iris.key file.

   Note: If the installer finds the new key file in the *install-dir*/mgr directory, it will proceed without prompting you for the license key file.

5. After the installer validates the license key, click **Convert**.
6. After the conversion is completed, click **Finish**.
7. Examine messages.log (formerly known as cconsole.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.

After the conversion finishes, the instance restarts, and any user installation manifest is run. For more information on installation manifests, see "Using an Installation Manifest".

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.
```

Then, start the InterSystems IRIS instance from the System Tray icon.

Important: If messages.log contains entries like the following, you need to reenter the logon information for the Windows Service controller:

```
Unable to update Windows Service controller with username <username>@<domain>,
   now running as 'Local System Account' user
You will need to update the Windows Service named 'InterSystems IRIS controller
   for <instname> with username <username>@<domain> and password
```

1. Open Windows Services and find the service InterSystems IRIS (or Health Connect) Controller for *instname*.
2. Right-click the service, click **Properties**, and then click the **Log On** tab.
3. Click **This Account**, and then enter the correct username and password.
4. Click **OK**.
5. Stop and restart the service.

2.7 Upgrade FHIR Repositories (If Converting to InterSystems IRIS for Health Only)

If you are converting from Health Connect to InterSystems IRIS for Health, and are using a FHIR repository or FHIR productions, run the FHIR upgrade utility. This regenerates the FHIR repository search tables and converts the FHIR repository streamlets and productions so that they use the re-implemented FHIR classes.

Run the following command from the Terminal, in each FHIR namespace:

```
do ##class(HS.HC.Util.Installer.Upgrade.FHIR1503).RunAll()
```

2.8 Update User Code, Compile Namespaces, and Start Productions

This step involves three distinct tasks that need to be performed for each namespace:

1. Updating User Code
2. Compiling Namespaces
3. Starting Productions (if converting an Ensemble or Health Connect instance)

You can take one of two approaches:

- To minimize downtime, you can run these three tasks sequentially on each namespace, starting with the highest priority namespace.
- To perform like tasks together, you can run each task on each namespace before proceeding to the next task.

CAUTION: You may need to perform these three tasks in the following namespaces:

- ENSEMBLE
- USER
- HSCUSTOM (Health Connect only)
- any user-created namespace

Any user classes or routines (with names Z*, z*, %Z*, and %z*) that are mapped to the %SYS namespace are automatically converted and recompiled. Other user code or data stored in system directories and databases may be lost in the conversion process.

2.8.1 Update User Code

Note: If you have previously performed and validated a conversion in another environment, importing any updated classes or routines exported from that environment will allow you to update your code quickly, minimizing downtime. Make sure you import each of your classes or routines into the appropriate namespace.

You can also load previously updated code (or an updated and pre-compiled database) using a manifest as part of your conversion process. See the instructions in Using an Installation Manifest.

After performing the InterSystems IRIS conversion, you may need to update the user code in each namespace, due to system artifacts that have been renamed or removed between Caché/Ensemble and InterSystems IRIS.

This task may vary, depending on your application's specific implementation. Review the *InterSystems IRIS Adoption Guide* for information on how your code may be affected. For Health Connect, also review the list of "Updated and Re-Implemented Health Connect Classes".

2.8.2 Compile Namespaces

Note: If you are using a manifest as part of your conversion process, you can use the manifest to compile your code (or to install an updated and pre-compiled database). See the instructions in Using an Installation Manifest.

Next, you must compile the code in each namespace so that it runs under InterSystems IRIS.

Class Definitions

First, upgrade your class definitions to prevent `Class dictionary out of date` errors from occurring while compiling your code.

To upgrade the class definitions in all namespaces, at once, from the Terminal:

```
do $system.OBJ.UpgradeAll()
```

To upgrade the class definitions in a single namespace, from the Terminal:

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

Note: If you have code databases that are mapped to your namespaces, use `$system.OBJ.UpgradeAll("/mapped")` or `$system.OBJ.Upgrade("/mapped")`.

Classes

To compile the classes in all namespaces, at once, 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")
```

Routines

Only custom routines need to be compiled manually. Routines generated from classes are compiled automatically when you compile the classes.

To compile the routines in all namespaces, at once, 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()
```

CSP Pages

Any CSP pages stored on disk need to be reloaded and compiled. This applies to CSP pages defined as .csp files in the *install-dir*/csp directory, not to CSP pages defined in .cls files.

To reload and compile the CSP pages in a single namespace, from the Terminal:

```
set $namespace = "<namespace>"
do $system.CSP.LoadPageDir("/csp/<namespace>")
```

Note: To launch the Terminal in InterSystems IRIS, use **iris terminal <instname>**, instead of **csession <instname>**.

On Windows, run the command from the *install-dir*\bin directory (or with *install-dir*\bin in the PATH environment variable). Or, launch the Terminal using the System Tray icon.

Note: If the compiler detects any errors, you may need to re-compile one or more times for the compiler to resolve all dependencies.

If errors persist, review the *InterSystems IRIS Adoption Guide* and make sure you have not missed anything. For Health Connect, also review the list of "Updated and Re-Implemented Health Connect Classes".

Note: After your code compiles successfully, you may want to export any updated classes or routines, in case you need to run the conversion in any additional environments. Importing these classes or routines during future conversions will allow you to update your code quickly, minimizing downtime.

2.8.3 Start Productions (If Converting an Ensemble or Health Connect Instance)

After updating your user code and recompiling each namespace, you are ready to start any Interoperability productions in the namespace.

1. In the Management Portal, select **Interoperability** > **List** > **Productions**.
2. On the Production List page, click the production that you want to start.
3. Click **Open**.
4. On the Production Configuration page, click **Start**.

5. In the confirmation dialog box, click **OK**.

   Note: If any Terminal windows open as a result of starting the production, do not close them.

6. When the progress dialog box shows it is "Done," click **OK**.

2.9 Turn On System-Wide Production Auto-Start

After the instance is fully converted, turn on production auto-start.

1. In the Management Portal, select **System Administration** > **Configuration** > **Additional Settings** > **Startup** .
2. On the Startup Settings page, click **Edit** next to the **EnsembleAutoStart** setting.
3. Enable the check box.
4. Click **Save**.

3

Performing a Conversion in a Mirrored Environment

Use this checklist to perform a conversion in a mirrored environment. InterSystems recommends printing this checklist and marking off each step when you have completed it. Complete all steps in the order specified.

At a high level, you will:

1. Convert the backup failover member to InterSystems IRIS.
2. Fail over to the backup failover member and validate InterSystems IRIS in a non-mirrored environment.
3. Convert the original primary failover member to InterSystems IRIS.
4. Optionally, fail back to the original primary failover member.

Important: If your mirror configuration includes a disaster recovery async member, read Converting a Disaster Recovery Async Member before starting the conversion process.

Table 3–1: Mirrored Environment Conversion Checklist

Step	Done	Server A (Original Primary Failover Member)	Server B (Original Backup Failover Member)	Downtime
1		Perform pre-conversion tasks	Perform pre-conversion tasks	
2			Uncompress the installation kit (Linux only)	
3			Set mirror No Failover	
4			Turn off system-wide production auto-start (Ensemble or Health Connect only)	
5			Delete %ALL namespace (if using)	
6			Shut down the Caché or Ensemble instance	
7			Stop and disable ISCAgent (Linux only)	
8			Run the installation script or installer	
9			Start and enable ISCAgent (Linux only)	

Step	Done	Server A (Original Primary Failover Member)	Server B (Original Backup Failover Member)	Downtime
10			Make sure Server B is backup member	
11			Clear mirror No Failover	
12		Turn off system-wide production auto-start (Ensemble or Health Connect only)		
13		Stop all Ensemble productions (Ensemble or Health Connect only)		Y
14		Delete %ALL namespace (if using)		Y
15		Shut down the Caché or Ensemble instance		Y
16			Make sure mirror failed over to Server B	Y
17			Create %ALL namespace (if using)	Y
18			Upgrade FHIR Repositories (If converting to InterSystems IRIS for Health only)	Y
19			Update user code, compile namespaces, and start productions	Y
20			Set mirror No Failover	
21			Validate non-mirrored InterSystems IRIS instance	
22		Uncompress the installation kit (Linux only)		
23		Stop and disable ISCAgent (Linux only)		
24		Run the installation script or installer		
25		Create %ALL namespace (if using)		
26		Start and enable ISCAgent (Linux only)		
27		Make sure Server A is backup member		
28		Turn on system-wide production auto-start		
29			Clear mirror No Failover	
30			Turn on system-wide production auto-start	
		If you want to run with Server B as the primary member, stop here.	*To fail back to Server A, complete the remaining steps.*	
31			Stop all Interoperability productions (Ensemble or Health Connect only)	Y
32			Shut down the InterSystems IRIS instance	Y

Step	Done	Server A (Original Primary Failover Member)	Server B (Original Backup Failover Member)	Downtime
33		Make sure mirror failed over to Server A		Y
34		Compile code in non-mirrored databases		
35			Start the InterSystems IRIS instance	
36			Make sure Server B is backup member	

CAUTION: Make sure you have validated the conversion in a development or test environment and fully tested your application under InterSystems IRIS before running the conversion on a production instance.

Important: After you convert the instance to InterSystems IRIS and it is fully validated, InterSystems recommends that you make a complete backup of the system. Use your customary full operating system backup procedures.

Important: If you are converting from Caché 2016.2 or Ensemble 2016.2 to InterSystems IRIS for Health 2020.1 or later, see InterSystems IRIS for Health Memory Considerations for needed post-conversion steps.

Note: If you are performing the InterSystems IRIS conversion on Linux, use **sudo** to run any commands that require root permissions.

Note: To launch the Terminal in InterSystems IRIS, use **iris terminal <instname>**, instead of **csession <instname>**.

On Windows, run the command from the *install-dir*\bin directory (or with *install-dir*\bin in the PATH environment variable). Or, launch the Terminal using the System Tray icon.

3.1 Perform Pre-Conversion Tasks

To ensure a smooth and efficient conversion, make sure you have performed the pre-conversion tasks before starting the conversion.

3.2 Uncompress the Installation Kit (Linux Only) — Server B

You will start the conversion process by working on the backup failover member first.

If your installation kit is in the form of a .tar file, for example IRIS-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
# gunzip -c /download/IRIS-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-lnxrhx64 in the preceding example.

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

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

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

3.3 Set Mirror No Failover — Server B

To prevent unwanted failover during the conversion process, set No Failover.

1. In the Management Portal, select **System Operation** >**Mirror Monitor**.
2. On the Mirror Monitor page, click **Set No Failover**.

3.4 Turn Off System-Wide Production Auto-Start (Ensemble or Health Connect Only) — Server B

As the conversion process leaves the instance running upon completion, turn off production auto-start so that Ensemble productions remain off after the conversion.

1. In the Management Portal, select **System Administration** > **Configuration** > **Additional Settings** > **Startup**.
2. On the Startup Settings page, next to the **EnsembleAutoStart** setting, click **Edit**.
3. Clear the **EnsembleAutoStart** check box.
4. Click **Save**.

3.5 Delete %ALL Namespace (If Using) — Server B

CAUTION: Do not perform a conversion in a mirrored environment where the %ALL namespace is defined. The conversion will fail and leave the instance in a non-recoverable state.

If you are using the %ALL namespace, you must delete it before running the installation script or installer. Then, create the %ALL namespace again, when instructed.

If the %ALL namespace is defined for the instance:

1. In the Management Portal, select **System Administration** > **Configuration** > **System Configuration** > **Namespaces**.
2. On the Namespaces page, check the mappings defined for the %ALL namespace and record them for future use.
3. At the end of the %ALL row, click **Delete**.
4. When asked to confirm the action, click **Finish**.

3.6 Shut Down the Caché or Ensemble Instance — Server B

Shut down the Caché or Ensemble instance from the command line:

```
# ccontrol stop <instname>
```

On Windows, run the command from the *install-dir*\bin directory (or with *install-dir*\bin in the PATH environment variable). Or, stop the instance using the System Tray icon.

CAUTION: Prior to performing a conversion on a Caché or Ensemble instance, it is essential that the instance be shut down cleanly. To verify that the shutdown was clean, examine the cconsole.log file after the shutdown finishes. If the log contains entries similar to the following, then the shutdown was clean:

```
Journal restore not required at next startup
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 conversion.

3.7 Stop and Disable ISCAgent (Linux Only) — Server B

On Linux systems, the location of the ISCAgent service has changed. Before running the installation script, stop and disable the old version of the service from the command line.

1. Stop the ISCAgent service:

   ```
   # systemctl stop ISCAgent.service
   ```

2. Disable the ISCAgent service:

   ```
   # sudo systemctl disable ISCAgent.service
   ```

3. Check the status of the ISCAgent service and make sure it shows that the service is inactive:

   ```
   # systemctl status ISCAgent
   ```

3.8 Run the Installation Script or Installer — Server B

Now that the instance is down, you are ready to perform the installation.

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

Note: If you are using a manifest as part of your conversion process, see the instructions in Using a Manifest, instead of performing the standard installation step.

On Linux

The installation script, **irisinstall**, provides all the functionality required to convert an existing Caché or Ensemble instance to InterSystems IRIS.

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

To perform the conversion:

1. 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, as described in "Uncompress the Installation Kit".

2. The script displays a list of any existing Caché, Ensemble, InterSystems IRIS, and Health Connect instances on the host.

3. At the **Enter instance name:** prompt, type the name of the Caché or Ensemble instance you want to convert to InterSystems IRIS, and press **Enter**.

4. When asked if you want to convert the instance to InterSystems IRIS, press **Enter** to confirm. (An existing Health Connect 15.03 instance may be identified as "HealthShare.")

5. When prompted for a license key file, type *keypath*\iris.key, where *keypath* is the location of the new iris.key file.

   Note: If the installation script finds the new key file in the *install-dir*/mgr directory, the script will proceed without prompting you for the license key file.

6. If you are converting a Health Connect instance, when prompted, optionally configure the Web Gateway.

7. The installation script summarizes the conversion options and again asks you to confirm the conversion. Press **Enter** to continue with the conversion.

8. Examine messages.log (formerly known as cconsole.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 conversion finishes, the instance restarts, and any user installation manifest is run. For more information on installation manifests, see "Using an Installation Manifest".

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.
```

Then, start the InterSystems IRIS instance from the command line:

```
# iris start <instname>
```

Note: When the instance starts after conversion, you may notice an alert:

```
Failed to become either Primary or Backup at startup
```

This alert occurs because ISCAgent is not running. The problem will correct itself after ISCAgent is enabled and the instance becomes the backup failover member.

On Windows

The installer provides all the functionality required to convert an existing Caché or Ensemble instance to InterSystems IRIS.

To perform the conversion:

1. Double-click the installer file, for example, IRIS-win_x64.exe.
2. The installer displays a list of any existing Caché, Ensemble, InterSystems IRIS, and Health Connect instances on the host. Any instances that can be converted are marked with the text `(CONVERSION)`.
3. Select the name of the Caché or Ensemble instance you want to convert to InterSystems IRIS, and click **OK**.
4. When prompted for an IRIS license key, click **License** and browse for the new iris.key file.

   Note: If the installer finds the new key file in the *install-dir*/mgr directory, it will proceed without prompting you for the license key file.

5. After the installer validates the license key, click **Convert**.
6. After the conversion is completed, click **Finish**.
7. Examine messages.log (formerly known as cconsole.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.

After the conversion finishes, the instance restarts, and any user installation manifest is run. For more information on installation manifests, see "Using an Installation Manifest".

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.
```

Then, start the InterSystems IRIS instance from the System Tray icon.

Important: If messages.log contains entries like the following, you need to reenter the logon information for the Windows Service controller:

```
Unable to update Windows Service controller with username <username>@<domain>,
   now running as 'Local System Account' user
You will need to update the Windows Service named 'InterSystems IRIS controller
   for <instname> with username <username>@<domain> and password
```

1. Open Windows Services and find the service InterSystems IRIS (or Health Connect) Controller for *instname*.
2. Right-click the service, click **Properties**, and then click the **Log On** tab.
3. Click **This Account**, and then enter the correct username and password.
4. Click **OK**.
5. Stop and restart the service.

Note: When the instance starts after conversion, you may notice an alert:

```
Failed to become either Primary or Backup at startup
```

This alert occurs because ISCAgent is updated and the service is temporarily not running. The problem will correct itself as the updated ISCAgent service is enabled and the instance becomes the backup failover member.

3.9 Start and Enable ISCAgent (Linux Only) — Server B

After the installation script finishes, start and then enable the updated version of the ISCAgent service from the command line.

```
# systemctl start ISCAgent.service
# sudo systemctl enable ISCAgent.service
```

Important: If you run the installation script without first stopping and disabling ISCAgent, follow these steps to make sure you are running the correct version of the service.

1. Run the following command:

   ```
   # systemctl status ISCAgent
   ```

2. The output of the command contains this warning:

   ```
   Warning: Unit file changed on disk, 'systemctl daemon-reload' recommended.
   ```

3. Run the following command:

   ```
   # systemctl daemon-reload
   ```

4. Then stop and restart the ISCAgent:

   ```
   # systemctl stop ISCAgent.service
   # systemctl start ISCAgent.service
   ```

3.10 Make Sure Server is the Backup Failover Member — Server B

Now that InterSystems IRIS is installed on the backup failover member, prepare to make it the primary failover member. Start by confirming that it is currently the backup failover member.

1. In the Management Portal, select **System Operation** >**Mirror Monitor**.
2. On the Mirror Monitor page, under Mirror Member Status, make sure the correct server appears with the Status of **Backup**.

Note: Check this from the Mirror Monitor page on the backup failover member. If the primary and the backup are on different versions of ISCAgent, the Mirror Monitor page on the primary failover member will show the backup failover member as having a Status of **Down**.

3.11 Clear Mirror No Failover — Server B

Next, clear No Failover. This will allow the backup failover member to take over as the primary failover member when the original primary failover member is shut down.

1. In the Management Portal, select **System Operation** >**Mirror Monitor**.

2. On the Mirror Monitor page, click **Clear No Failover**.

3.12 Turn Off System-Wide Production Auto-Start (Ensemble or Health Connect Only) — Server A

Before shutting down the original primary failover member, prepare it to also be converted.

As the conversion process leaves the instance running upon completion, turn off production auto-start so that Ensemble productions remain off after the conversion.

For details, see "Turn Off System-Wide Production Auto-Start — Server B".

3.13 Stop All Ensemble Productions (Ensemble or Health Connect Only) — Server A

To prevent Ensemble productions from entering a problem state, stop all productions before shutting down the instance. This is an InterSystems best practice and allows large transactions to complete and remaining messages in production queues to be processed.

Perform the following procedure in each namespace that contains Ensemble productions:

1. In the Management Portal, select **Ensemble** > **List** > **Productions**.
2. On the Production List page, click the production that you want to stop.
3. Click **Open**.
4. On the Production Configuration page, click **Stop**.
5. When the browser asks if you wish to stop this production, click **OK**.

   Note: If any Terminal windows open as a result of stopping the production, do not close them.

6. When the Stop Production dialog box shows it is "Done," click **OK**.
7. If portal displays a message "Production could not stop, do you want to force a shut down?", click **Yes - Force to Shut Down**.

CAUTION: A production acquires the `Suspended` status when, at the end of the shutdown sequence, some queues still contain any messages. If the production becomes `Suspended`, contact the InterSystems Worldwide Response Center before proceeding.

A production acquires a status of `Troubled` if the production did not shut down properly. On the Production Configuration page, click **Recover** to shut down and clean up the production. If this does not clean up the production, you may need to run the following method in the Terminal:

```
do ##class(Ens.Director).RecoverProduction()
```

Note: If you have a large number of productions and stopping them individually would be very time consuming, run the following command from the Terminal:

```
do ##class(%EnsembleMgr).OnSystemShutdown(.pConsoleLog)
```

Then run the following command to make sure all productions have been stopped:

```
zw pConsoleLog
```

You should see messages similar to the following:

```
pConsoleLog=2
pConsoleLog(1)=$lb(0,"Stopping Ensemble production in namespace 'MYNAMESPACE initiated at
15:03:32")
pConsoleLog(2)=$lb(0,"All Ensemble productions stopped.")
```

3.14 Delete %ALL Namespace (If Using) — Server A

Again, if you are using the %ALL namespace, you will need to delete it before this instance can be converted to InterSystems IRIS.

For details, see "Delete %ALL Namespace (If Using) — Server B".

3.15 Shut Down the Caché or Ensemble Instance — Server A

Shut down the original primary failover member to cause a failover to the original backup failover member.

For details, see "Shut Down the Caché or Ensemble Instance — Server B".

3.16 Make Sure Mirror Failed Over — Server B

After mirror failover, make sure that the correct server is the primary failover member.

1. In the Management Portal, select **System Operation** >**Mirror Monitor**.
2. On the Mirror Monitor page, under Mirror Member Status, make sure the correct server appears with the Status of **Primary**.

3.17 Create %ALL Namespace (If Using) — Server B

If you deleted the %ALL namespace before running the installation script or installer, re-create it now.

1. In the Management Portal, select **System Administration** > **Configuration** > **System Configuration** > **Namespaces**.
2. On the Namespaces page, click **Create New Namespace**.

3. On the New Namepace page, in the **Name of the Namespace** box, type %ALL.
4. Click **Save**.
5. When asked to confirm, click **OK**.
6. In the **%ALL** row, re-create the mappings you recorded earlier.

3.18 Upgrade FHIR Repositories (If Converting to InterSystems IRIS for Health Only) — Server B

If you are converting from Health Connect to InterSystems IRIS for Health, and are using a FHIR repository or FHIR productions, run the FHIR upgrade utility. This regenerates the FHIR repository search tables and converts the FHIR repository streamlets and productions so that they use the re-implemented FHIR classes.

Run the following command from the Terminal, in each FHIR namespace:

```
do ##class(HS.HC.Util.Installer.Upgrade.FHIR1503).RunAll()
```

3.19 Update User Code, Compile Namespaces, and Start Productions — Server B

This step involves three distinct tasks that need to be performed for each namespace:

1. Updating User Code
2. Compiling Namespaces
3. Starting Productions (if converting an Ensemble or Health Connect instance)

You can take one of two approaches:

- To minimize downtime, you can run these three tasks sequentially on each namespace, starting with the highest priority namespace.
- To perform like tasks together, you can run each task on each namespace before proceeding to the next task.

CAUTION: You may need to perform these three tasks in the following namespaces:

- ENSEMBLE
- USER
- HSCUSTOM (Health Connect only)
- any user-created namespace

Any user classes or routines (with names Z*, z*, %Z*, and %z*) that are mapped to the %SYS namespace are automatically converted and recompiled. Other user code or data stored in system directories and databases may be lost in the conversion process.

3.19.1 Update User Code

Note: If you have previously performed and validated a conversion in another environment, importing any updated classes or routines exported from that environment will allow you to update your code quickly, minimizing downtime. Make sure you import each of your classes or routines into the appropriate namespace.

You can also load previously updated code (or an updated and pre-compiled database) using a manifest as part of your conversion process. See the instructions in Using an Installation Manifest.

After performing the InterSystems IRIS conversion, you may need to update the user code in each namespace, due to system artifacts that have been renamed or removed between Caché/Ensemble and InterSystems IRIS.

This task may vary, depending on your application's specific implementation. Review the *InterSystems IRIS Adoption Guide* for information on how your code may be affected. For Health Connect, also review the list of "Updated and Re-Implemented Health Connect Classes".

3.19.2 Compile Namespaces

Note: If you are using a manifest as part of your conversion process, you can use the manifest to compile your code (or to install an updated and pre-compiled database). See the instructions in Using an Installation Manifest.

Next, you must compile the code in each namespace so that it runs under InterSystems IRIS.

Class Definitions

First, upgrade your class definitions to prevent `Class dictionary out of date` errors from occurring while compiling your code.

To upgrade the class definitions in all namespaces, at once, from the Terminal:

```
do $system.OBJ.UpgradeAll()
```

To upgrade the class definitions in a single namespace, from the Terminal:

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

Note: If you have code databases that are mapped to your namespaces, use `$system.OBJ.UpgradeAll("/mapped")` or `$system.OBJ.Upgrade("/mapped")`.

Classes

To compile the classes in all namespaces, at once, 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")
```

Routines

Only custom routines need to be compiled manually. Routines generated from classes are compiled automatically when you compile the classes.

To compile the routines in all namespaces, at once, 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()
```

CSP Pages

Any CSP pages stored on disk need to be reloaded and compiled. This applies to CSP pages defined as .csp files in the *install-dir*/csp directory, not to CSP pages defined in .cls files.

To reload and compile the CSP pages in a single namespace, from the Terminal:

```
set $namespace = "<namespace>"
do $system.CSP.LoadPageDir("/csp/<namespace>")
```

Note: To launch the Terminal in InterSystems IRIS, use **iris terminal <instname>**, instead of **csession <instname>**.

On Windows, run the command from the *install-dir*\bin directory (or with *install-dir*\bin in the PATH environment variable). Or, launch the Terminal using the System Tray icon.

Note: If the compiler detects any errors, you may need to re-compile one or more times for the compiler to resolve all dependencies.

If errors persist, review the *InterSystems IRIS Adoption Guide* and make sure you have not missed anything. For Health Connect, also review the list of "Updated and Re-Implemented Health Connect Classes".

Note: After your code compiles successfully, you may want to export any updated classes or routines, in case you need to run the conversion in any additional environments. Importing these classes or routines during future conversions will allow you to update your code quickly, minimizing downtime.

3.19.3 Start Productions (If Converting an Ensemble or Health Connect Instance)

After updating your user code and recompiling each namespace, you are ready to start the Interoperability productions in the namespace.

1. In the Management Portal, select **Interoperability** > **List** > **Productions**.
2. On the Production List page, click the production that you want to start.
3. Click **Open**.
4. On the Production Configuration page, click **Start**.
5. In the confirmation dialog box, click **OK**.

   Note: If any Terminal windows open as a result of starting the production, do not close them.

6. When the progress dialog box shows it is "Done," click **OK**.

3.20 Set Mirror No Failover — Server B

The original backup failover member is now fully converted to InterSystems IRIS. To prevent it from failing back to the original primary failover member before it is converted to InterSystems IRIS, set No Failover.

For details, see "Set Mirror No Failover — Server B".

3.21 Validate Non-Mirrored InterSystems IRIS — Server B

Once your application is up and running under InterSystems IRIS on the original backup failover member, validate the updated instance according to your company's best practices before running the installer on the original primary failover member.

CAUTION: Do not run the installer on the original primary failover member (Server A) until you are confident that the original backup failover member (Server B) is running InterSystems IRIS without any issues. Once the original primary failover member has also been converted to InterSystems IRIS, there is no path back to Caché or Ensemble.

Important: You cannot failover the original backup failover member (Server B) back to original primary failover member (Server A) until it has also been converted to InterSystems IRIS. This rule does not apply if Server A is running Caché or Ensemble 2018.1.3 or later.

3.22 Uncompress the Installation Kit (Linux Only) — Server A

While the original primary failover member is down, convert it to InterSystems IRIS, starting by uncompressing the installation kit.

For details, see "Uncompress the Installation Kit (Linux Only) — Server B".

3.23 Stop and Disable ISCAgent (Linux Only) — Server A

On Linux systems, the location of the ISCAgent service has changed. Before running the installation script, stop and disable the old version of the service from the command line.

For details, see "Stop and Disable ISCAgent (Linux Only) — Server B".

3.24 Run the Installation Script or Installer — Server A

Now you are ready to perform the installation on the original primary failover member.

For details, see "Run the Installation Script or Installer — Server B".

3.25 Create %ALL Namespace (If Using) — Server A

If you deleted the %ALL namespace before running the installation script or installer, re-create it now.

For details, see “Create %ALL Namespace (If Using) — Server B”.

3.26 Start and Enable ISCAgent (Linux Only) — Server A

Start and then enable the updated version of the ISCAgent service from the command line.

For details, see “Start and Enable ISCAgent (Linux Only) — Server B”.

3.27 Make Sure Server is the Backup Failover Member — Server A

Now that InterSystems IRIS is installed on the original primary failover member, prepare to make it the primary failover member again. Start by confirming that it is currently the backup failover member.

For details, see “Make Sure Server is the Backup Failover Member — Server B”.

3.28 Turn On System-Wide Production Auto-Start — Server A

Before failing back to the original primary failover member, turn on production auto-start.

1. In the Management Portal, select **System Administration** > **Configuration** > **Additional Settings** > **Startup** .
2. On the Startup Settings page, click **Edit** next to the **EnsembleAutoStart** setting.
3. Enable the check box.
4. Click **Save**.

3.29 Clear Mirror No Failover — Server B

On the original backup failover member, clear No Failover. This will allow the original primary failover member to take over as the primary failover member when the original backup failover member is shut down.

For details, see “Clear Mirror No Failover — Server B”.

3.30 Turn On System-Wide Production Auto-Start — Server B

Turn on system-wide production auto-start on the original backup failover member to prepare for future failovers.

For details, see "Turn On System-Wide Production Auto-Start — Server A".

3.31 Stop All Interoperability Productions (Ensemble or Health Connect Only) — Server B

To prevent Interoperability productions from entering a problem state, stop all productions before shutting down the instance. This is an InterSystems best practice and allows large transactions to complete and remaining messages in production queues to be processed.

Perform the following procedure in each namespace that contains Interoperability productions:

1. In the Management Portal, select **Interoperability** > **List** > **Productions**.
2. On the Production List page, click the production that you want to stop.
3. Click **Open**.
4. On the Production Configuration page, click **Stop**.
5. When the browser asks if you wish to stop this production, click **OK**.

   Note: If any Terminal windows open as a result of stopping the production, do not close them.

6. When the Stop Production dialog box shows it is "Done," click **OK**.
7. If portal displays a message "Production could not stop, do you want to force a shut down?", click **Yes - Force to Shut Down**.

CAUTION: A production acquires the `Suspended` status when, at the end of the shutdown sequence, some queues still contain any messages. If the production becomes `Suspended`, contact the InterSystems Worldwide Response Center before proceeding.

A production acquires a status of `Troubled` if the production did not shut down properly. On the Production Configuration page, click **Recover** to shut down and clean up the production. If this does not clean up the production, you may need to run the following method in the Terminal:

```
do ##class(Ens.Director).RecoverProduction()
```

Note: If you have a large number of productions and stopping them individually would be very time consuming, run the following command from the Terminal:

```
do ##class(%EnsembleMgr).OnSystemShutdown(.pConsoleLog)
```

Then run the following command to make sure all productions have been stopped:

```
zw pConsoleLog
```

You should see messages similar to the following:

```
pConsoleLog=2
pConsoleLog(1)=$lb(0,"Stopping Ensemble production in namespace 'MYNAMESPACE initiated at
15:03:32")
pConsoleLog(2)=$lb(0,"All Ensemble productions stopped.")
```

3.32 Shut Down the InterSystems IRIS Instance — Server B

Shut down the original backup failover member to fail back to the original primary failover member.

From the command line:

```
# iris stop <instname>
```

On Windows, run the command from the *install-dir*\bin directory (or with *install-dir*\bin in the PATH environment variable). Or, stop the instance using the System Tray icon.

CAUTION: Prior to performing a conversion on a Caché or Ensemble instance, it is essential that the instance be shut down cleanly. To verify that the shutdown was clean, examine the cconsole.log file after the shutdown finishes. If the log contains entries similar to the following, then the shutdown was clean:

```
Journal restore not required at next startup
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 conversion.

3.33 Make Sure Mirror Failed Over — Server A

Make sure the original primary failover member is again primary.

For details, see “Make Sure Mirror Failed Over — Server B”.

3.34 Compile Code in Non-Mirrored Databases — Server A

When the mirror fails over to the original primary failover member, the updated code stored in any mirrored databases is copied over. If you have any code stored in any non-mirrored databases, update and recompile it now.

This step involves two distinct tasks that need to be performed for each namespace in a non-mirrored database:

1. Updating User Code
2. Compiling Namespaces

3.34.1 Update User Code

Note: If you have previously performed and validated a conversion in another environment, importing any updated classes or routines exported from that environment will allow you to update your code quickly, minimizing downtime. Make sure you import each of your classes or routines into the appropriate namespace.

You can also load previously updated code (or an updated and pre-compiled database) using a manifest as part of your conversion process. See the instructions in Using an Installation Manifest.

After performing the InterSystems IRIS conversion, you may need to update the user code in each namespace, due to system artifacts that have been renamed or removed between Caché/Ensemble and InterSystems IRIS.

This task may vary, depending on your application's specific implementation. Review the *InterSystems IRIS Adoption Guide* for information on how your code may be affected. For Health Connect, also review the list of "Updated and Re-Implemented Health Connect Classes".

3.34.2 Compile Namespaces

Note: If you are using a manifest as part of your conversion process, you can use the manifest to compile your code (or to install an updated and pre-compiled database). See the instructions in Using an Installation Manifest.

Next, you must compile the code in each namespace stored in a non-mirrored database.

Class Definitions

First, upgrade your class definitions to prevent `Class dictionary out of date` errors from occurring while compiling your code.

For each namespace stored in a non-mirrored database, from the Terminal:

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

Note: If you have code databases that are mapped to your namespace, use `$system.OBJ.Upgrade("/mapped")`.

Classes

For each namespace stored in a non-mirrored database, from the Terminal:

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

Routines

For each namespace stored in a non-mirrored database, from the Terminal:

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

CSP Pages

Any CSP pages stored on disk need to be reloaded and compiled. This applies to CSP pages defined as .csp files in the *install-dir*/csp directory, not to CSP pages defined in .cls files.

For each namespace stored in a non-mirrored database, from the Terminal:

```
set $namespace = "<namespace>"
do $system.CSP.LoadPageDir("/csp/<namespace>")
```

Note: To launch the Terminal in InterSystems IRIS, use **iris terminal <instname>**, instead of **csession <instname>**.

On Windows, run the command from the *install-dir*\bin directory (or with *install-dir*\bin in the PATH environment variable). Or, launch the Terminal using the System Tray icon.

Note: If the compiler detects any errors. you may need to re-compile one or more times for the compiler to resolve all dependencies.

If errors persist, review the *InterSystems IRIS Adoption Guide* and make sure you have not missed anything. For Health Connect, also review the list of "Updated and Re-Implemented Health Connect Classes".

Note: After your code compiles successfully, you may want to export any updated classes or routines, in case you need to run the conversion in any additional environments. Importing these classes or routines during future conversions will allow you to update your code quickly, minimizing downtime.

3.35 Start the InterSystems IRIS Instance — Server B

Once the original primary failover member is fully converted to InterSystems IRIS, start InterSystems IRIS on the original backup failover member from the command line:

```
# iris start <instname>
```

On Windows, run the command from the *install-dir*\bin directory (or with *install-dir*\bin in the PATH environment variable). Or, start the instance using the System Tray icon.

Important: If your operating system is configured to use huge or 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 1468MB shared memory using Huge Pages. Startup will retry with standard
 pages.
```

Then, start the InterSystems IRIS instance from the command line:

```
# iris start <instname>
```

Note: When the instance starts after conversion, you may notice an alert:

```
Preserving all mirror journal files for offline failover member
```

This alert is normal and can safely be ignored.

3.36 Make Sure Server is the Backup Failover Member — Server B

Make sure that the original backup failover member is again the backup failover member.

For details, see “Make Sure Server is the Backup Failover Member — Server B”.

4

Using an Installation Manifest

If your application relies on code libraries, CSS files, JavaScript files, or other components that are not included as part of InterSystems IRIS, you can use a manifest to install them as an additional option to the in-place conversion process. A manifest is also useful for silent installs, as you can invoke methods from the manifest, eliminating the need for interactive steps.

Creating a manifest is done with the help of the %Installer utility.

4.1 Creating a Class That Defines a Manifest

To create a class that defines an installation manifest, create a class according to the following rules:

- The class must include the %occInclude include file.
- It must contain an XData block specifying the details of the installation.

  You can specify any legal name for the XData block; you use this name later as an argument.

  If you use `[XMLNamespace = INSTALLER]` after the name of the XData block, Studio provides assistance as you type the XData block.

  The root element in the XData block must be `<Manifest>`.
- It must define the **setup()** method.
- The **setup()** method must refer to the XData block by name.

You can use the following code as a template:

```
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")
}

}
```

For a comprehensive list of tags that can be used in a manifest, see "List of <Manifest> Tags" in the *Installation Guide*.

4.2 Running the Manifest

4.2.1 Running the Manifest Manually

To run a manifest manually, enter the following command in the Terminal.

```
do ##class(MyInstallerPackage.SimpleManifest).setup()
```

4.2.2 Running the Manifest with the In-Place Conversion

Note: Before you use the manifest along with the InterSystems in-place conversion, InterSystems recommends that you run it manually it in a test instance of InterSystems IRIS to make sure it functions as expected.

If you are running the manifest as part of a silent installation, make sure to copy the new InterSystems IRIS key to the mgr directory of the instance to be converted before performing the installation, as you will not be able to specify the key location interactively.

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

To run the manifest in conjunction with the InterSystems in-place conversion, export the manifest class as an XML file. You can name the XML file `DefaultInstallerClass.xml` and copy it to the same directory where the installation script or installer is run, or you can specify the location and name of the file on the command line. The manifest will be run at the end of the installation process. The manifest class is imported into the %SYS namespace and compiled, and the **setup()** method is executed.

To run the manifest in conjunction with the InterSystems in-place conversion, use a command similar to the following examples.

On Linux

Set the necessary environment variables on the command line prior to running either **irisinstall** or **irisinstall_silent**, as shown in the following example:

```
sudo ISC_PACKAGE_INSTANCENAME="CACHECONV"
ISC_INSTALLER_MANIFEST="/Conversion/manifest.xml"
ISC_INSTALLER_LOGFILE="/Conversion/ManifestLogs/manifest.log"
ISC_INSTALLER_LOGLEVEL="3" ./irisinstall_silent
```

The log level must be in the range of 0 to 3, where 0 is "none" and 3 is "verbose."

For more information, see the complete list of Linux unattended installation command-line options in the *Installation Guide*.

On Windows

Pass command-line arguments to the installer as shown in the following example. Use the /qn option if you want a silent installation. If you are running a silent installation, run the Windows command prompt in Administrator mode.

```
IRISUPG.85.0-win_x64.exe /instance CACHECONV /qn
INSTALLDIR="C:\InterSystems\CacheConv"
INSTALLERMANIFEST="C:\InterSystems\Conversion\manifest.xml"
INSTALLERMANIFESTLOGFILE="C:\InterSystems\Conversion\ManifestLogs\manifest.log"
INSTALLERMANIFESTLOGLEVEL="3"
```

The log level must be in the range of 0 to 3, where 0 is “none” and 3 is “verbose.”

For more information, see the complete list of Windows unattended installation command-line options in the *Installation Guide*.

4.3 Manifest Examples

This section presents some sample manifests to illustrate concepts that should be useful in a range of situations. These examples are not intended to provide a comprehensive overview of the capabilities of manifests. For more information, see the “Creating and Using an Installation Manifest” chapter in the *Installation Guide*.

4.3.1 Copying an IRIS Code-Only Database

In this scenario, you have successfully converted an instance to IRIS, updated your application code, and compiled it under InterSystems IRIS. Now, you want to perform conversions on other instances and install this pre-compiled code by copying the entire database. This example uses Windows path names.

This example assumes:

- Your code is contained in a code-only database, in this example, AppTestCode.
- You have compiled the code in under InterSystems IRIS and copied the IRIS.DAT file to the directory C:\InterSystems\Conversion\AppTestCode.

The XData block of the manifest class:

1. Writes a message to the manifest log indicating the start of the manifest installation process.
2. Invokes the method **SYS.Database.DismountDatabase()** to dismount the code-only database AppTestCode.
3. Copies the pre-compiled IRIS.DAT file to the proper namespace directory.
4. Invokes **SYS.Database.MountDatabase()** to mount the database again.
5. Writes a message to the manifest log indicating the end of the manifest installation process.

```
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."/>
    <Invoke Class="SYS.Database" Method="DismountDatabase" CheckStatus="1">
        <Arg Value="C:\InterSystems\InstallDir\mgr\AppTestCode\"/>
    </Invoke>
    <CopyFile Src="C:\InterSystems\Conversion\AppTestCode\IRIS.DAT"
        Target="C:\InterSystems\InstallDir\mgr\AppTestCode\"
        IgnoreErrors="1"/>
    <Invoke Class="SYS.Database" Method="MountDatabase" CheckStatus="1">
```

```
        <Arg Value="C:\InterSystems\InstallDir\mgr\AppTestCode\"/>
    </Invoke>
    <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")
}

}
```

4.3.2 Importing Code from an XML Export

In this scenario, you have successfully converted an instance to IRIS, updated your application code, and compiled it under InterSystems IRIS. Now, you want to perform conversions on other instances and install this updated code by importing your code from an XML export. This example uses Linux path names.

This example assumes:

- Your code is contained in the namespace APPTEST.
- You have compiled the code under InterSystems IRIS, exported the code into an XML file, and copied the file to the directory /intersystems/conversion.

The XData block of the manifest class:

1. Writes a message to the manifest log indicating the start of the manifest installation process.
2. Sets the current namespace to APPTEST.
3. Invokes the method **%SYSTEM.OBJ.Upgrade()** to upgrade the class definitions in the namespace. Specifying the `/mapped` qualifier will also upgrade class definitions in mapped databases.
4. Imports the XML file (or files) in the directory /intersystems/conversion.
5. Invokes the method **%SYSTEM.OBJ.CompileAll()** to compile the classes in the namespace. (Included for completeness. May not be needed if the code is already compiled under InterSystems IRIS.)
6. Invokes the method **%Routine.CompileAll()** to compile the custom routines in the namespace.
7. Invokes the method **%SYSTEM.CSP.LoadPageDir()** to load and compile the CSP pages in the namespace.
8. Writes a message to the manifest log indicating the end of the manifest installation process.

```
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="Upgrade" CheckStatus="1">
            <Arg Value="/mapped"/>
        </Invoke>
        <Import File ="/intersystems/conversion/"/>
        <Invoke Class="%SYSTEM.OBJ" Method="CompileAll" CheckStatus="1">
            <Arg Value="u"/>
        </Invoke>
        <Invoke Class="%Routine" Method="CompileAll" CheckStatus="1"/>
        <Invoke Class="%SYSTEM.CSP" Method="LoadPageDir" CheckStatus="1">
            <Arg Value="/csp/apptest"/>
```

```
        </Invoke>
    </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")
}

}
```

4.3.3 Upgrading a FHIR Repository

In this scenario, you are converting a Health Connect instance to IRIS for Health and you want to automatically upgrade an existing FHIR repository for use with IRIS for Health. This example uses Windows path names.

This example assumes that your FHIR namespace is called FHIRSERVER.

The XData block of the manifest class:

1. Writes a message to the manifest log indicating the start of the manifest installation process.
2. Sets the current namespace to FHIRSERVER.
3. Invokes the method **HS.HC.Util.Installer.Upgrade.FHIR1503.RunAll()** to update the FHIR repository in this namespace.
4. Writes a message to the manifest log indicating the end of the manifest installation process.

```
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="FHIRSERVER">
      <Invoke Class="HS.HC.Util.Installer.Upgrade.FHIR1503" Method="RunAll"/>
   </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")
}

}
```

4.3.4 Loading Library Code into a Read-Only Database

In this scenario, you want to load a set of library classes into a read-only database and copy a number of associated images, CSS files, and JavaScript files to the instance. This example uses Linux path names.

This example assumes:

- You have downloaded an export of the library classes to the directory /Conversion/Classes.
- You have downloaded the associated images, CSS files, and JavaScript files to the directory /Conversion/AuxFiles.

The XData block of the manifest class:

1. Writes a message to the manifest log indicating the start of the manifest installation process.
2. Sets the current namespace to USERLIB, where the library classes are to be stored.
3. Invokes a method in the manifest class, **MyInstallerPackage.SimpleManifest.SetDBReadOnly()**, to set the userlib database to not be read-only.
4. Imports the library classes.
5. Invokes **SetDBReadOnly()** to set the userlib database to be read-only again.
6. Sets the current namespace to USER, where the application using the library is stored.
7. Invokes the method **%SYSTEM.OBJ.CompileAll()** to compile all classes in the namespace to run under InterSystems IRIS.
8. Copies the images, CSS files, and JavaScript files to /<installdir>/csp/broker.
9. Writes a message to the manifest log indicating the end of the manifest installation process.

```
Include %occInclude

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

XData SimpleManifest [ XMLNamespace = INSTALLER ]
{
<Manifest>
   <Log Level="3" Text="Start manifest" />
   <Namespace Name="USERLIB">
      <Invoke Class="MyInstallerPackage.SimpleManifest" Method="SetDBReadOnly" CheckStatus="1">
         <Arg Value="userlib"/>
         <Arg Value="0"/>
      </Invoke>
      <Import File="/Conversion/Classes" />
      <Invoke Class="MyInstallerPackage.SimpleManifest" Method="SetDBReadOnly" CheckStatus="1">
         <Arg Value="userlib"/>
         <Arg Value="1"/>
      </Invoke>
   </Namespace>
   <Namespace Name="USER">
      <Invoke Class="%SYSTEM.OBJ" Method="CompileAll" CheckStatus="1">
         <Arg Value="u"/>
      </Invoke>
   </Namespace>
  <CopyDir Src="/Conversion/AuxFiles/" Target="/InterSystems/InstallDir/csp/broker/" IgnoreErrors="1"
 />
   <Log Level="3" Text="End manifest" />
</Manifest>
}

/// Set ReadOnly property of a database.
ClassMethod setDBReadOnly(db As %String, ReadOnly As %Boolean) As %Status
{
   Set db = ##Class(SYS.Database).%OpenId(db)
   Set db.ReadOnly = ReadOnly
   Set status = db.%Save()
   Return status
}


/// 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")
```

```
}
}
```

Note: As Zen Mojo is not included in InterSystems IRIS, this example can be tailored to install the Zen Mojo core classes and copy the associated plugins to the directory /<installdir>/csp/broker.

To reinstall Zen Mojo you will need to download it from the InterSystems Worldwide Response Center components distribution page:

- Download the Zen Mojo zip file and copy the export file of the Zen Mojo core classes, Zen Mojo 1.1.2.xml, to a directory of your choosing.
- Download the Zen Mojo demo zip file, and unzip and copy the plugins to a directory of your choosing.

Then make the following changes to the manifest example:

- In the first <Namespace> tag, change the namespace from USERLIB to %SYS.
- In the first two <Invoke> tags, change the first argument from userlib to irislib.
- In the <Import> tag, specify the directory where you copied the Zen Mojo core classes.
- In the second <Namespace> tag, change the namespace from USER to the namespace where your Zen Mojo application is stored.
- In the <CopyDir> tag, specify the directory where you copied the Zen Mojo plugins.

A

Conversion Details

A.1 InterSystems IRIS Conversion Details

The InterSystems IRIS conversion performs the following tasks within the install directory:

- Deletes all InterSystems supplied binaries, DLLs, Locales, CSP files, Dev, and Java Libraries and replaces them with InterSystems IRIS files. As entire directories are deleted and replaced, be careful not to store any user code or databases in these InterSystems directories.
- Deletes and recreates the mgr/Temp directory, which holds temporary system and user files.
- Retains all user databases, renaming them iris.dat. It does not alter the contents of the user databases.
- Retains the csp.ini file, which contains user CSP settings.
- Retains the journal directories and files. The journal files in InterSystems IRIS are backwards compatible with existing journal files.
- Converts the cacheodbc.ini file to InterSystems IRIS format and renames it irisodbc.ini, retaining any user settings.
- Converts cache.cpf to InterSystems IRIS format and renames it iris.cpf. A list of the conversions is appended to the end of the CPF file.
- Removes the SAMPLES, ENSDEMO, and DOCBOOK databases and namespaces. The USER and ENSEMBLE databases are retained.
- Updates the Caché and Ensemble System databases to their InterSystems IRIS counterparts (for example, CACHELIB to IRISLIB).
- Converts and renames the CACHESYS database (mgr) to IRISSYS. User data, routines and classes stored in the %SYS namespace are retained and recompiled.
- Loads the current locale definitions and NLS settings, including custom locales created by the user.
- Changes the %DB_CACHE* resource names in the system databases to %DB_IRIS*.
- If the audit directory was previously mgr/cacheaudit, changes the audit directory is to mgr/irisaudit. If auditing was performed to an alternate directory, that directory will remain the audit directory.
- If the CACHE directory (now IRISLOCALDATA) was previously mgr/cache, changes the directory to mgr/irislocaldata. If it was an alternate directory, that directory will be the IRISLOCALDATA directory.
- If the CACHETEMP directory (now IRISTEMP) was previously mgr/cachetemp, changes the directory to mgr/iristemp. If it was an alternate directory, that directory will be the IRISTEMP directory.

- Renames the log file cconsole.log to messages.log.
- Applies any database updates contained in the write image journaling file CACHE.WIJ and renames the file IRIS.WIJ.
- In Windows, updates all system and user ODBC data source names (DSNs) to use the InterSystems IRIS ODBC driver. All DSNs that use the SAMPLES namespace are deleted.

A.2 Security Database Conversion Details

The security database is converted as follows:

- The audit database stored in the `^CacheAuditD` global is converted to `^IRIS.AuditD`
- LDAP configuration information is moved into its own class, Security.LDAPConfigs, in order to support multiple LDAP domains.
- The following roles and resources have been removed:
  - `%DB_DOCBOOK`
  - `%DB_SAMPLES`
  - `%DB_ENSDEMO`
  - `%DB_CACHESYS`
  - `%DB_CACHELIB`
  - `%DB_CACHETEMP`
  - `%DB_CACHEAUDIT`
  - `%DB_CACHE`
- The following roles and resources have been added:
  - `%DB_IRISSYS`
  - `%DB_IRISLIB`
  - `%DB_IRISTEMP`
  - `%DB_IRISAUDIT`
  - `%DB_IRISLOCALDATA`
- The following Service has been removed:
  - `%Service_MSMActivate`

B

Updated and Re-Implemented Health Connect Classes

Depending on your application, you may need to update any references to any Health Connect 15.03 classes and files that have been renamed or re-implemented in Health Connect 2019.1 or later (based on InterSystems IRIS) and InterSystems IRIS for Health.

Check any namespace where user code is to be updated (such as, HSCUSTOM, USER, and any user-created namespace), and rename or remediate any affected references, as listed in the table below:

Table II–1: Updated and Re-Implemented Health Connect Classes

Health Connect 15.03 Class	New Class
HS.Audit.ConsolidationService	HS.HC.Audit.ConsolidationService
HS.FHIR.Utils.SearchTableBuilder	HS.FHIR.vDSTU2.SearchTableBuilder
HS.FHIR.Operation.Process	HS.FHIR.vDSTU2.Repository.OperationProcessor
HS.IHE.DSUB.Publisher.Process	HS.HC.IHE.DSUB.Publisher.Process
HS.IHE.XDR.Recipient.CommonProcess	HS.HC.IHE.XDR.Recipient.CommonProcess
HS.IHE.XDSb.Consumer.Operations	HS.HC.IHE.XDSb.Consumer.Operations
HS.IHE.XDSb.Registry.Operations	HS.HC.IHE.XDSb.Registry.Operations
HS.Message.ECRUpdateRequest	Removed. If HS.Message.ECRUpdateRequest was previously used as an input message type for HS.FHIR.FromSDA.DTL.Transaction.Process, then please modify the input to use HS.Message.XMLMessage or Ens.StreamContainer, instead.
HS.MPI.Manager	HS.HC.MPI.Manager
HS.UI.AssigningAuthorities	HS.HC.UI.AssigningAuthorities
HS.UI.Installer.Welcome	HS.HC.UI.Installer.Welcome
HS.UI.Home	HS.HC.UI.Home
HS.Util.Installer	HS.HC.Util.Installer
HS.Util.Trace.Helper	HS.HC.Util.Trace.Helper

Table II–2: Other Updated and Re-Implemented Health Connect Files

Health Connect 15.03 File	New File
HS.Common.inc	HS.HC.inc
HS.Errors.inc	HS.HC.Errors.inc

Note: Certain additional classes available in Health Connect 15.03 are no longer available in Health Connect 2019.1 or later (based on InterSystems IRIS) or InterSystems IRIS for Health. InterSystems believes that these classes are rarely used by Health Connect customers. If you see any error messages saying that certain classes or databases cannot be found, please email conversionft@intersystems.com for assistance.

C

InterSystems IRIS for Health Memory Considerations

If you are converting from Caché 2016.2 or Ensemble 2016.2 to InterSystems IRIS for Health 2020.1 or later, you may need to perform some post-conversion steps to adjust certain memory settings on your instance. The settings carried over from your Caché or Ensemble instance may not be sufficient to make use of some InterSystems IRIS for Health functionality, such as the FHIR data load utility.

Guidelines for the important settings are as follows:

- Maximum Per-Process Memory (should be at least 262144KB)
- Memory Allocated for 8KB Database Cache (should be at least 165MB)
- Maximum Lock Table Size (should be at least 16777216 bytes)
- Memory Allocated for Routine Cache (should be at least 299MB)

Follow the steps in this appendix to adjust your memory settings, if necessary. You can use either Terminal or the Management Portal to make these changes.

C.1 Use Terminal

To use Terminal to adjust the key memory settings:

1. In Terminal, change to the %SYS namespace and get the current properties:

   ```
   Set $Namespace = "%SYS"
   Do ##Class(Config.config).Get(.Properties)
   ```

2. Write the current properties:

   ```
   Zwrite Properties
   ```

3. Examine the output, and set any properties that do not meet the guidelines listed above, for example:

   ```
   Set Properties("bbsiz") = 262144
   Set Properties("globals8kb") = 165
   Set Properties("locksiz") = 16777216
   Set Properties("routines") = 299
   ```

4. Store the new values:

```
Do ##Class(Config.config).Modify(.Properties)
```

5. Restart the instance.

C.2 Use the Management Portal

To use the Management Portal to adjust the key memory settings:

1. In the Management Portal, select **System Administration** > **Configuration** > **Additional Settings** > **Advanced Memory**.
2. On the Advanced Memory Settings screen, next to `locksiz`, click **Edit**.
3. On the Edit: locksiz screen, check the current value.
4. If the value is less than in the guidelines listed above, enter `16777216`, and click **Save**.
5. Select **System Administration** > **Configuration** > **System Configruation** > **Memory and Startup**.
6. On the Memory and Startup screen, examine the current memory settings to see if they meet the guidelines, using one of the following methods:
   - If **Configure Memory Settings** is set to `Automatically`, the actual memory settings may not be reflected in the values shown on the screen. Follow steps 1 and 2 in the Use Terminal section to view the actual values.
   - If **Configure Memory Settings** is set to `Manually`, check the values shown on the screen.
7. Set **Configure Memory Settings** to `Manually`.
8. Set any values that do not meet the guidelines, for example:
   - **Memory Allocated for Routine Cache (MB):** 299
   - **Memory Allocated for 8KB Database Cache (MB):** 165
   - **Maximum Per-Process Memory (KB):** 262144
9. Click **Save**.
10. Restart the instance.

D

Converting a Disaster Recovery Async Member

If your mirror configuration includes a disaster recovery async (D/R Async) member, isolate it from the rest of the network before starting the conversion process to prevent changes from being written to it when the primary and backup failover members are converted. This allows you to use the D/R Async member for disaster recovery should an unrecoverable error occur while converting the primary and backup failover members. For information on disaster recovery, see "Disaster Recovery Procedures" in the *High Availability Guide*.

For configurations that include a D/R Async member, follow this outline:

1. Isolate the D/R Async member from the network using one of the following methods:
   - From the Mirror Monitor on the D/R Async member, click **Stop Mirror on This Member**.
   - Disable the network between the D/R Async member and the primary and backup failover members.
2. Convert the primary and backup failover members by performing the steps in Performing a Conversion in a Mirrored Environment.
3. Convert the D/R Async member by performing the following steps:
   a. Delete %ALL Namespace (If Using)
   b. Shut down the Caché or Ensemble instance
   c. Uncompress the installation kit (Linux only)
   d. Stop and disable ISCAgent (Linux only)
   e. Run the installation script or installer
   f. Create %ALL Namespace (If Using)
   g. Start and Enable ISCAgent (Linux Only)
4. Reconnect the D/R Async member to the network using the appropriate method:
   - From the Mirror Monitor on the D/R Async member, click **Start Mirror on This Member**.
   - Enable the network between the D/R Async member and the primary and backup failover members.

E

Issues with Caché or Ensemble and InterSystems IRIS on the Same Machine

Several potential issues can occur if you attempt to install Caché or Ensemble and InterSystems IRIS on the same machine. In general, these issues occur because InterSystems IRIS has a different registry than Caché and Ensemble. Installers that pre-dated in-place conversion are not aware of instances defined in the other registry:

- Caché and Ensemble installers prior to version 2019.1.4
- Health Connect installers prior to version 2019.1
- InterSystems IRIS installers prior to 2019.1.1
- InterSystems IRIS for Health installers prior to version 2020.1

The ISCAgent version conflict is a separate issue.

E.1 ISCAgent Version Conflict (Linux Only)

Under certain circumstances, if you perform an in-place conversion of Caché or Ensemble on a machine that already contains an instance of InterSystems IRIS, it may cause a mismatch in versions of ISCAgent. This mismatch can prevent mirror members from communicating with one another.

This issue arises in the following scenario:

1. You install InterSystems IRIS on the machine, for example, version 2020.1.
2. You then install, on the same machine, a version of Caché or Ensemble earlier than version 2018.1, for example, version 2017.2.2.
3. You then convert the instance of Caché or Ensemble to a version of InterSystems IRIS earlier than the one already installed on the machine, for example, 2019.1.1.

In this scenario, the installation of Caché or Ensemble overwrites the version of ISCAgent already installed by InterSystems IRIS, and the conversion sees the later version of InterSystems IRIS and does not update ISCAgent.

To fix this issue, perform a standalone installation of ISCAgent on the machine. You must install a version of ISCAgent equal to or later than the latest instance of InterSystems IRIS on the machine. Download the ISCAgent installer from the InterSystems Worldwide Distribution Center components distribution page.

E.2 Port Number Conflict

Caché and Ensemble use a different default series of numbers for the Web Server and Superserver ports than InterSystems IRIS does. However, if you perform a conversion, these port numbers are retained. If you then install another instance of Caché or Ensemble prior to version 2019.1.4 on that machine, the installer will not be aware of the InterSystems IRIS instance and will reuse the same port numbers, resulting in a conflict.

To prevent this scenario from occurring, either change the Web Server and Superserver port numbers of the InterSystems IRIS instance prior to installing the Caché or Ensemble instance, or perform a custom installation of Caché or Ensemble and specify unique port numbers.

E.3 Instance Name Conflict

If you attempt to install an instance of Caché or Ensemble prior to version 2019.1.4 using the same instance name as an existing instance of InterSystems IRIS, one or both instances may become inoperable. A similar scenario can also occur when installing an instance of InterSystems IRIS with the same name as an existing Caché or Ensemble instance, where the InterSystems IRIS installer does not have in-place conversion capability.

When installing an instance of any InterSystems product, avoid specifying an instance name that is the same as an existing instance name, except in the cases of performing an in-place conversion or an upgrade to the existing instance.

F

Converting a Health Connect/HSAP Instance That Was Previously Upgraded from a Full HealthShare Instance

If you attempt to convert a Health Connect/HSAP instance on Windows that was previously upgraded from a full HealthShare instance, the HealthConnect 2019+ or InterSystems IRIS for Health 2020+ installer may not recognize the instance as being a candidate for conversion. (When you run the installer, any instances that can be converted are marked with the text `(CONVERSION)`.)

If your instance was upgraded from a full HealthShare instance, or you have reason to believe that it might have been, perform the following procedure before beginning the conversion.

Important: If the procedure asks you to make any changes to the Windows registry, back up the registry appropriately before doing so.

1. Launch the Windows registry editor.
2. If you find any of the following registry keys, delete them:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intersystems\Cache\Configurations\<InstanceName>\HSAA]
[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intersystems\Cache\Configurations\<InstanceName>\HSDocbook]
[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intersystems\Cache\Configurations\<InstanceName>\HSPI]
[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intersystems\Cache\Configurations\<InstanceName>\HSViewer]
```

3. Make sure that the following registry keys are present:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intersystems\Cache\Configurations\<InstanceName>\HSCore]
[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intersystems\Cache\Configurations\<InstanceName>\MPRL]
[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Intersystems\Cache\Configurations\<InstanceName>\HSFDocbook]
```

4. If any of the above registry keys is missing, create an empty key with the same name as the missing key.

Once you have confirmed that the registry is in order, you can perform the conversion normally, as described in this document.

G

Version History

Table VII–1: History of Document Versions and Changes Made

Version	Date	Changes
DRAFT 1.3	June 17, 2019	Draft distributed to first-round testers.
DRAFT 1.4	June 21, 2019	Added more information on using **$system.OBJ.Upgrade()** to upgrade the class definitions in a database prior to compiling code. Added information on loading and compiling CSP pages with **$system.CSP.LoadPageDir()**. Updated the "Importing Code" manifest example to make it more robust. In the Pre-Conversion Tasks chapter, noted that the SAMPLES, ENSDEMO, and DOCBOOK databases are deleted during the conversion. (This was previously noted only in Appendix A.) Made minor fixes and clarifications.
DRAFT 1.5	June 26, 2019	In the Introduction, added (1) feedback email address and (2) information on mirroring and ECP limitations. In the Pre-Conversion Tasks chapter, added information related to (1) instances that call user code on startup and (2) disk space requirements.
DRAFT 1.6	July 30, 2019	Added information on checking/restoring the logon information for the Windows controller service for Caché/InterSystems IRIS. In the Pre-Conversion Tasks chapter, added tasks to identify any custom routines or Caché Server Pages for which you do not have source code loaded. In the Pre-Conversion Tasks chapter, added information to the task on making a complete backup. In the Conversion Details, removed the item on renaming the user and group `cacheusr` to `irisusr`, as this is no longer part of the conversion process. Made minor clarifications.
DRAFT 1.7	August 27, 2019	In the conversion instructions, added information on running **##class(%EnsembleMgr).OnSystemShutdown()** if you have a large number of Ensemble productions to stop. In the pre-conversion tasks, added a more accurate calculation of the disk space required to do the conversion. Made minor clarifications.
1.8	September 16, 2019	Version for 2019.1.1 preview. Added HealthShare Health Connect conversion information. Added clarifications to Supported Versions section. In Limitations section, clarified ECP information and added KMIP Server information. Removed document version history.

Version	Date	Changes
1.9	November 4, 2019	Version for InterSystems IRIS 2019.1.1 general release. Remove preview notice. Added back document version history. In Appendix B, added HC.Common.inc as a file that has been re-implemented in Health Connect 2019.1 and IRIS for Health. Fixed minor typo.
DRAFT 2.0	November 4, 2019	Draft for InterSystems IRIS for Health conversion field test. In the Introduction, added information on the valid conversion paths and supported versions. Added information on how to upgrade a FHIR repository and added a manifest example. Made minor clarifications.
DRAFT 2.1	November 7, 2019	Added note that the InterSystems IRIS for Health field test installation build is for evaluation purposes only and should not be used to convert production instances. Made minor clarifications.
2.2	December 19, 2019	In the Introduction, added the limitation that in-place conversion is not currently supported for instances running on Windows Cluster. Added information on applicable products at the beginning of document, not just under Requirements. In Appendix A, added details on WIJ files. Updated trademarks. Made minor clarifications.
2.3	January 23, 2020	Version for 2020.1 preview. In the mirrored environment section, added information on converting a disaster recovery async member and more descriptive text regarding mirror failover strategy. In Appendix B, added a note regarding additional classes available in Health Connect 15.03 that are not available in Health Connect 2019.1 or later (based on InterSystems IRIS) or in InterSystems IRIS for Health.
2.4	February 5, 2020	In the Introduction, clarified the limitations on mixed mirror configurations and added a note that in-place conversion is not supported for platforms that are supported only for development purposes. In the Pre-Conversion Tasks chapter, added a note that a Caché or Ensemble-based license server cannot be used with InterSystems IRIS. In Appendix B, added HS.Errors.inc as a renamed Health Connect file.
2.5	March 30, 2020	Version for 2020.1 general release. In the Introduction, added HSAP to the places Health Connect is mentioned. If you know your product as HSAP, follow the instructions for Health Connect. Clarified Health Connect conversion paths. Clarified that supported versions apply to both mirrored and non-mirrored configurations. Removed 2020.1 preview notice. Added Appendix C on InterSystems IRIS for Health memory considerations.
2.6	June 1, 2020	In the Introduction, updated the Supported Conversion Paths section to include support for converting Ensemble to Health Connect 2020.1 or later and made additional clarifications. Moved the disaster recovery async member conversion guidelines to Appendix D. Added Appendix E to cover considerations regarding machines where both Caché or Ensemble and InterSystems IRIS are installed. Changed references of **zn** to **set $namespace** in accordance with InterSystems best practices.
2.7	June 3, 2020	In Appendix E, clarified the ISCAgent version conflict issue.
2.8	August 4, 2020	In Appendix A, mentioned that the Windows conversion updates all ODBC DSNs to the InterSystems IRIS drivers. Added Appendix F to cover an issue that can arise if converting a Health Connect/HSAP instance on Windows that was previously upgraded from a full HealthShare instance.