InterSystems IRIS Basics: Connecting an IDE

Version 2024.1
2024-07-02

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

InterSystems IRIS Basics: Connecting an IDE
InterSystems IRIS Data Platform Version 2024.1 2024-07-02

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

InterSystems IRIS Basics: Connecting an IDE

This document tells you how to quickly connect your integrated development environment (IDE) for ObjectScript, Java, .NET, Python, or JavaScript/Node.js to an instance of InterSystems IRIS® and verify that you have successfully connected, using template files downloaded from GitHub.

In addition, the next two sections cover the following:

- Deploying InterSystems IRIS

  Lists the options for deploying both licensed and free evaluation instances of InterSystems IRIS. If you do not yet have an instance to work with, review this section.

- InterSystems IRIS connection information

  Details the information you need to connect an IDE to a licensed or free InterSystems IRIS instance, and to interact with it in other ways, and explains get the information for your instance. Review the section covering the type of instance you are working with, regardless of which IDE you want to connect to InterSystems IRIS.

1 Deploying InterSystems IRIS

This section covers both licensed instances and free evaluation instances.

1.1 Deploying Licensed Instances

If you have purchased one or more licenses from InterSystems, you can deploy a licensed InterSystems IRIS instance using one of these methods:

- the InterSystems Kubernetes Operator
- an InterSystems IRIS container image
- an InterSystems IRIS installation kit

When deploying from either a container image or an installation kit, you can use the configuration merge feature to support automated deployment.

1.1.1 InterSystems Kubernetes Operator

Kubernetes is an open-source orchestration engine for automating deployment, scaling, and management of containerized workloads and services. You define the containerized services you want to deploy and the policies you want them to be governed by; Kubernetes transparently provides the needed resources in the most efficient way possible, repairs or restores the deployment when it deviates from spec, and scales automatically or on demand. The InterSystems Kubernetes Operator (IKO) extends the Kubernetes API with the IrisCluster custom resource, which can be deployed as an InterSystems IRIS sharded cluster, distributed cache cluster, or standalone instance (all optionally mirrored) on any Kubernetes platform. The IKO also adds InterSystems IRIS-specific cluster management capabilities to Kubernetes, enabling automation of tasks

like adding nodes to a cluster, which you would otherwise have to do manually by interacting directly with the instances. For information about using the IKO, see Using the InterSystems Kubernetes Operator.

1.1.2 InterSystems IRIS Container Image

Container images from InterSystems let you use your own tools and methods to deploy InterSystems IRIS in containers on Linux hosts — cloud, virtual, or physical. Persistent storage of instance-specific data makes containerized InterSystems IRIS easy to upgrade, so you can always move your existing configurations to the latest version of InterSystems IRIS with little trouble and minimal downtime.

Note: Container images from InterSystems comply with the Open Container Initiative (OCI) specification and are therefore supported on any OCI-compliant runtime engine on Linux-based operating systems. The remainder of this document assume the use of the Docker runtime engine.

For a detailed guide to using InterSystems IRIS images, see *Running InterSystems Products in Containers*. For information on how to obtain an InterSystems IRIS image, see Downloading the InterSystems IRIS Image in *Running InterSystems Products in Containers*.

You can provision a BYOL (bring your own license) cloud node with an InterSystems IRIS container image and Docker installed on the Google Cloud Platform, Microsoft Azure, or Amazon Web Services public cloud platforms, then add your license and run an InterSystems IRIS container from the image as described in *Running InterSystems Products in Containers*. For more information, see Deploy InterSystems IRIS BYOL on a Cloud Node in *Deploy and Explore InterSystems IRIS*. (You can also use these documents with a free InterSystems IRIS Community Edition image.)

1.1.3 InterSystems IRIS Installation Kit

You can install and license a development instance of InterSystems IRIS on your local machine or on another on your network by obtaining the latest installation kit for your platform from the InterSystems Worldwide Response Center (WRC) download area. For installation instructions, see the *Installation Guide*.

1.1.4 Deployment Using Configuration Merge

The configuration merge feature lets you vary the configurations of InterSystems IRIS containers deployed from the same image, or local instances installed from the same kit. You can use it in the following ways:

- Deploy with configuration merge

  On Linux and UNIX® systems, you can use configuration merge in deployment to update an instance's *configuration parameter file* (CPF), which contains its configuration settings; these settings are read from the CPF at every startup, including the first one after an instance is deployed. When you apply configuration merge during deployment, you in effect replace the default CPF provided with the instance with your own updated version. You can also use the feature when restarting an existing instance.

- Use configuration merge on a running instance

  On both Linux/UNIX® and Windows systems, you can reconfigure a running instance by using the **iris merge** command to update its CPF with the contents of a merge file.

The InterSystems Kubernetes Operator incorporates the configuration merge feature.

For complete information about using configuration merge, see Automating Configuration of InterSystems IRIS with Configuration Merge.

1.2 Deploying Free Evaluation Instances

If you are not yet an InterSystems IRIS user, you can explore its many features and try it out with your preferred languages and tools and your own code by deploying a free evaluation instance. There are several options for doing this:

- On a public cloud node by creating an account on or logging in to Google Cloud Platform, Microsoft Azure, or Amazon Web Services, going gto the platform's marketplace page, and searching for **InterSystems IRIS**.
- In a container on the system of your choice by pulling the Community Edition image from the InterSystems Container Registry (ICR) or Docker Hub.

1.2.1 InterSystems IRIS Community Edition

InterSystems IRIS Community Edition comes with a free built-in 13-month license (and some functionality restrictions). You can deploy Community Edition in two ways:

- On a public cloud node by creating an account on or logging in to Google Cloud Platform, Microsoft Azure, or Amazon Web Services, going gto the platform's marketplace page, and searching for **InterSystems IRIS**.
- In a container on the system of your choice by pulling the Community Edition image from the InterSystems Container Registry (ICR) or Docker Hub.

For instructions for deploying and using InterSystems IRIS Community Edition in the cloud or on your own system, see *Deploy and Explore InterSystems IRIS*.

1.2.2 InterSystems Learning Labs

The InterSystems Learning Labs web page lets you quickly and easily launch a lab instance of InterSystems IRIS for training, development, and testing, with a free 30-day license. Your InterSystems Learning Labs instance includes an integrated IDE and plenty of samples to work with, and you can connect your own IDE.

To launch an InterSystems Learning Labs instance, you must be logged in to learning.intersystems.com; you can easily create an account if you do not have one. (Like Community Edition, the Learning Labs instance has some functionality restrictions.)

2 InterSystems IRIS Connection Information

To connect an IDE to an InterSystems IRIS instance, you first need to select the instance, then determine the needed connection information, as follows:

- Host identifier: the IP address or hostname of the instance's host.
- The superserver port number for the instance.
- Valid credentials for the instance.
- The name of an existing namespace on the instance.

It is also helpful to know the URL of the instance's web-based Management Portal, and how to open the InterSystems Terminal on the instance, which requires the instance's name.

How you determine this information depends on how you deployed the instance you are working with, as follows:

- InterSystems Kubernetes Operator
- InterSystems IRIS Deployed in a Container
- InterSystems IRIS Installed from a Kit
- InterSystems IRIS Community Edition
- InterSystems Learning Labs

2.1 InterSystems Kubernetes Operator

Kubernetes deployments can include services that expose pods and the containers they contain to the network, as needed, through external IP addresses. Regardless of the InterSystems IRIS topology you deploy with the IKO, a service is always created to expose the primary data node—data node 1 in a sharded cluster, the data server in a distributed cache cluster, or a standalone instance. For information about using this service to connect to the superserver port (1972) or web server port (80) of the InterSystems IRIS instance on the primary data node, see Connect to the IrisCluster in *Using the InterSystems Kubernetes Operator*.

The credentials for an IDE connection or the Management Portal are one of the predefined user accounts and the default password you set using the `PasswordHash` parameter in a configuration merge file specified in the `configSource` field, or an account and password you previously added to the instance.

To open the Terminal for the primary data node instance, use the following command:

```
kubectl exec -it IrisCluster_name-data-0 iris terminal IRIS
```

If the deployment is mirrored, use the pod name *IrisCluster_name*-data-0–0 instead to open the Terminal on the current primary. The instance name is always **`IRIS`** in any container from InterSystems. Commands issued from outside an InterSystems IRIS container using **docker exec**, and thus **kubectl exec**, are executed inside the container as **`irisowner`** and do not require authentication, so you do not need to log in to the Terminal.

You can always connect your IDE to the **`USER`** namespace that exists on all InterSystems IRIS instances, but you can also connect to a different namespace you previously created using the Management Portal.

2.2 InterSystems IRIS Deployed in a Container

As explained in detail in Web Access Using the Web Gateway Container in *Running InterSystems Products in Containers*, the InterSystems IRIS Management Portal is a built-in web application, therefore a web server and the InterSystems Web Gateway are required to load it in your browser. However, a single Web Gateway instance configured to interact with multiple InterSystems IRIS containers cannot direct a request for an application common to all of the instances to a specific instance.

One simple way to enable Management Portal access to a containerized InterSystems IRIS instance is to run a dedicated Web Gateway container (which also contains a web server) with each InterSystems IRIS container. A dedicated Web Gateway container is configured to interact only with the InterSystems IRIS container with which it is paired; for deployments requiring a web server tier, additional Web Gateway containers must be included. For more information about and instructions for running dedicated and web server node Web Gateway containers and other approaches to providing Management Portal access to containerized InterSystems IRIS instances, see Options for Running Web Gateway Containers.

Important: InterSystems IRIS Community Edition containers are an exception to the above. For information about accessing the Management Portal of a Community Edition instance, see Interact Using the Management Portal in *Deploy and Explore InterSystems IRIS*.

The credentials for an IDE connection or the Management Portal are one of the predefined user accounts and the default password you set when creating the container, or an account and password you have previously added to the instance.

To open the Terminal for an instance in a container, you can use the **docker exec** command to run the **iris terminal** command within the container, and you can also use **docker exec** to open a shell within the container and run **iris terminal** from that; for examples, see Interacting Using the InterSystems Terminal in *Deploy and Explore InterSystems IRIS*. The instance name is always **`IRIS`** in any container from InterSystems. Commands issued from outside an InterSystems IRIS container using **docker exec** are executed inside the container as **`irisowner`** and do not require authentication, so you do not need to log in to the Terminal.

You can always connect your IDE to the **USER** namespace that exists in all InterSystems IRIS instances, but you can also connect to a different namespace you previously created using the Management Portal.

2.3 InterSystems IRIS Installed from a Kit

For an InterSystems IRIS instance installed from a kit, the host identifier is the hostname or IP address of the system the instance is running on; you can use **localhost** if it is installed locally.

InterSystems IRIS installation sets the superserver port to 1972 by default. However, if you have more than one instance of InterSystems IRIS installed on a system, the superserver port of the instances vary; to display the port numbers for all of the instances, you can use the iris list command on the operating system command line.

On all systems, you can use the URL described in Access the Management Portal and Other System Applications in the *Web Gateway Guide* to open the Management Portal. On a Windows system, assuming a correctly configured Web Gateway and web server, you can open the Management Portal by clicking the InterSystems IRIS icon in the system tray and selecting **Management Portal**.

To open the Terminal for an installed instance:

- On Windows systems, you can select the **Terminal** option on the InterSystems IRIS launcher menu.
- On all systems, you can use the iris terminal command on the operating system command line, with the instance name as an argument. The instance name is set when you install the instance and cannot be changed; you can use the **iris list** command to display it.

The credentials for all purposes are one of the predefined user accounts and the default password **SYS** (you are prompted to immediately change the password after logging in to one of these accounts), or an account and password you have previously added to the instance.

You can always connect your IDE to the **USER** namespace that exists in all InterSystems IRIS instances, but you can also connect to a different namespace you previously created using the Management Portal.

2.4 InterSystems IRIS Community Edition

For information about accessing the Management Portal for a Community Edition instance, which is unlike other types of instances in this regard, see Interact Using the Management Portal in *Deploy and Explore InterSystems IRIS*.

To information about opening the Terminal for any Community Edition instance, see Interacting Using InterSystems Terminal in *Deploy and Explore InterSystems IRIS*.

The credentials for an IDE connection and the Management Portal are one of the predefined user accounts and either the new default password you provided when changing the password on first connecting to the cloud node or the default password **SYS**, if you are on your own system or logging in to the cloud instance without having connected to the node first. When you use **SYS**, you are prompted to immediately change the password for the account you are using. You can also use an account you previously created on the instance.

You can always connect your IDE to the **USER** namespace that exists in all InterSystems IRIS instances, but you can also connect to a different namespace you previously created using the Management Portal.

2.5 InterSystems Learning Labs

All connections to the containerized InterSystems IRIS instance that is part of your InterSystems Learning Labs configuration are set up during the Learning Labs launch, and all the connection information you need is displayed on the launch page, as follows:

- The URL for the integrated IDE.

- A set of credentials for all purposes.
- The Management Portal URL.
- The web server port number (always 80).
- Under **External Connections**, the server's IP address and the superserver port number (labeled **InterSystems IP** and **InterSystems Port**); you can use these to connect an IDE such as InterSystems Studio or Visual Studio Code with the ObjectScript extension to the instance.

A Terminal option is available on the **InterSystems** menu in the integrated IDE (as is a **Management Portal** option), so you don't need to keep track of the instance name.

3 ObjectScript IDEs

This section covers the IDEs you can use to develop ObjectScript code on an InterSystems IRIS instance: Visual Studio Code (VS Code) with the ObjectScript extension, and Studio.

3.1 Visual Studio Code

Visual Studio Code (VS Code) is a free source code editor made by Microsoft for Windows, Linux, and macOS. While it can be used out of the box as a .NET, Python, or JavaScript IDE with InterSystems IRIS, available extensions enable you to use VS Code to develop code in ObjectScript, which is stored and versioned on the client system and compiled, run, and debugged on an InterSystems IRIS server. VS Code can connect to any InterSystems IRIS instance regardless of the platform the instance is running on, using the DNS name or IP address of the host, the name of the instance, and the instance's webserver port number. For installation and usage instructions, see Use Visual Studio Code as a Development Environment for InterSystems Applications.

3.2 Studio

InterSystems IRIS application developers can also the Studio IDE, a client application running on Windows systems; see *Using Studio* for detailed information. Studio can connect to any InterSystems IRIS instance regardless of the platform the instance is running on, using the DNS name or IP address of the host, the name of the instance, and the instance's superserver port number, which is 1972 by default (but may be different is more than one instance is installed on the system).

4 Java IDEs

This section provides connection instructions for Visual Studio Code (with the Coding Pack for Java or the Java Extension Pack), Eclipse, IntelliJ, and NetBeans, which you can use to develop Java code that interacts with InterSystems IRIS using the InterSystems Native SDK and JDBC.

4.1 Visual Studio Code

To connect Visual Studio Code to InterSystems IRIS, use the following steps. (Log in to GitHub if necessary.)

1. In your web browser, download or clone https://github.com/intersystems/Samples-java-helloworld.
2. In Visual Studio Code, use **File > Open Folder** to open the folder you recently downloaded or cloned from GitHub.
3. Select and open src/main/java.com.intersystems.samples.helloworld.java.

4. Configure the username, password, IP address, and superserver port for your instance (see InterSystems IRIS Connection Information) in the variable declarations at the top of the main method. These values are used to construct the InterSystems JDBC connection string, in the form jdbc:IRIS://*ipAddress*:*webserverPort*/*namespace*, for example jdbc:IRIS://12.345.678.910:1972/USER.
5. Use **View > Terminal** to open the terminal and execute the following commands:

```
javac -cp ".:intersystems-jdbc-3.2.0.jar" HelloWorld.java
java -cp ".:intersystems-jdbc-3.2.0.jar" HelloWorld
```

6. The Output pane displays the message `Hello World! You have successfully connected to InterSystems IRIS via JDBC.`

4.2 Eclipse

To connect Eclipse to InterSystems IRIS, use the following steps. (Click **Next** to advance to the next panel of each dialog as needed.)

First, import the sample package,

1. Select **File > Import > Git > Projects from Git**.
2. Choose **Clone URI**. Copy and paste https://github.com/intersystems/Samples-java-helloworld into the **URI** field.
3. Select the **master** branch, configure local storage, then choose **Import existing Eclipse projects**.
4. Confirm the import by clicking **Finish**.

Next, update the sample code and run it.

1. Open **samples-java-helloworld > src > main.java.com.intersystems.samples > HelloWorld.java** and make the following changes:
   a. Change the package declaration at the top to **package main.java.com.intersystems.samples;**.
   b. Configure the username, password, IP address, and port for your instance (see InterSystems IRIS Connection Information) in the variable declarations at the top of the main method. These values are used to construct the InterSystems JDBC connection string, in the form jdbc:IRIS://*ipAddress*:*superserverPort*/*namespace*, for example jdbc:IRIS://12.345.678.910:1972/USER.
2. Run the code by selecting **Run > Run**.
3. The **Console** tab displays the message `Hello World! You have successfully connected to InterSystems IRIS via JDBC.`

If you are not successful, confirm that the Java execution environment is set properly by doing the following:

1. Right-click the **samples-java-helloworld** project in the Package Explorer pane and choose **Build Path > Configure Build Path...**.
2. On the **Libraries** tab, choose **Add Library** > **JRE System Library** > **Execution environment** > and choose an execution environment, such as **JRE-1.1 (jre 1.8.0_172)**. Click **Finish** and then **Apply and Close**.

4.3 IntelliJ

To connect IntelliJ to InterSystems IRIS, use the following steps. (Click **Next** to advance to the next panel of each dialog as needed.)

First, create the sample project.

1. Select **VCS > Checkout from Version Control > Git**.
2. On the Clone Repository dialog,
   a. Copy and paste https://github.com/intersystems/Samples-java-helloworld into the **Clone URI** field.
   b. For the **Directory** field, enter the path to the location where you want the local GitHub repository created.
3. Click **Clone**, then choose **Yes** in the popup to create a project based on this source.

Next, update the sample code and run it.

1. Open src/main/java.com.intersystems.samples.helloworld.java and configure the username, password, IP address, and port for your instance (see InterSystems IRIS Connection Information) in the variable declarations at the top of the main method. These values are used to construct the InterSystems JDBC connection string, in the form jdbc:IRIS://*ipAddress*:*superserverPort*/*namespace*, for example jdbc:IRIS://12.345.678.910:1972/USER.
2. In the **Project** pane, right-click **HelloWorld.java** and choose **Run 'HelloWorld.main()'**.
3. The **Output** pane displays the message `Hello World! You have successfully connected to InterSystems IRIS via JDBC.`

If you are not successful, confirm that the Java execution environment is set properly by doing the following:

1. Choose **Build Path > Build Project**.
2. Edit the configuration and add a new application configuration, selecting **com.intersystems.samples.HelloWorld** for the main class.
3. If the error `Error:java: invalid source release: 9` is displayed, change the project SDK and project language level to **1.8** in the following locations:
   - **File > Project Structure** > **Project Settings**
   - **File** > **Project Structure** > **Module Settings** > **Sources: Language Level** tab
   - **File** > **Project Structure** > **Module Settings** > **Dependencies: Module SDK** tab

4.4 NetBeans

To connect NetBeans to InterSystems IRIS, use the following steps. (Click **Next** to advance to the next panel of each dialog as needed.)

First, create the sample project.

1. Select **Team > Git > Clone**.
2. Copy and paste https://github.com/intersystems/Samples-java-helloworld into the **Repository URL** field.
3. Select **Master** as the branch to be fetched, choose the location where you want the local GitHub repository to be created, and click **Finish**.

Next, update the sample code and run it.

1. Open HelloWorld.java and configure the username, password, IP address, and port for your instance (see InterSystems IRIS Connection Information) in the variable declarations at the top of the main method. These values are used to construct the InterSystems JDBC connection string, in the form jdbc:IRIS://*ipAddress*:*superserverPort*/*namespace*, for example jdbc:IRIS://12.345.678.910:1972/USER.
2. In the **Project** pane, open the Dependencies folder, then right-click **intersystems-jdbc-3.0.0.jar** and choose **Manually install artifact**. Navigate to the folder you recently cloned, select **intersystems-jdbc-3.0.0.jar**, and click **Install Locally**.

3. In the **Project** pane, right-click HelloWorld.java and choose **Run File**.
4. The **Output** pane displays the message `Hello World! You have successfully connected to InterSystems IRIS via JDBC.`

5 .NET IDEs

This section provides connection instructions for Microsoft's Visual Studio and Visual Studio Code, which you can use to develop .NET code that interacts with InterSystems IRIS through the InterSystems Native SDK, the ADO.NET Managed Provider for .NET, andInterSystems XEP.

5.1 Visual Studio

To connect Visual Studio to InterSystems IRIS, use the following steps. (Log in to GitHub if necessary.)

First, create the sample solution.

1. Select **View > Team Explorer**.
2. In the Team Explorer – Connect pane, select **Local Git Repositories > Clone**, copy and paste https://github.com/intersystems/Samples-dotnet-helloworld into the URL box, and click **Clone**.

Next, update the sample code and run it.

1. Double-click HelloWorld.sln to see the files in the solution, then open helloworld.cs.
2. Configure the username, password, IP address, and port for your instance (see InterSystems IRIS Connection Information) using the variable declarations,
3. Press the F5 key to run the program.
4. A command window displays the message `Hello World! You have successfully connected to InterSystems IRIS. Press any key to continue.`

5.2 Visual Studio Code

With the ObjectScript extension, Visual Studio Code can be used to develop ObjectScript Code on InterSystems IRIS; see ObjectScript IDEs.

To connect Visual Studio Code to InterSystems IRIS, use the following steps. (Log in to GitHub if necessary.)

1. In your web browser, download or clone https://github.com/intersystems/Samples-dotnet-helloworld.
2. In Visual Studio Code, use **File > Open Folder** to open the folder you recently downloaded or cloned from GitHub.
3. Expand **samples-dotnet-helloworld** and select **helloworld.cs**.
4. Configure the username, password, IP address, and port for your instance (see InterSystems IRIS Connection Information) using the variable declarations.
5. Use **View > Terminal** to open the terminal and execute the following commands:

```
csc /reference:InterSystems.Data.IRISClient.dll helloworld.cs
.\helloworld.exe
```

6. The Output pane displays the message `Hello World! You have successfully connected to InterSystems IRIS. Press any key to continue.`

6 Python IDEs

This section provides instructions for connecting your favorite Python-focused IDE, such as Visual Studio Code, PyCharm, Spyder, IDLE/IdleX, or Vim to InterSystems IRIS.

1. In your web browser, download or clone https://github.com/intersystems/Samples-python-helloworld.
2. In your IDE, open the folder you recently downloaded or cloned from GitHub, then open hello_world.py.
3. Configure the username, password, IP address, and port for your instance (see InterSystems IRIS Connection Information) using the variable declarations.
4. In the Terminal of your IDE, do the following:
   a. Install the InterSystems Native SDK for Python using one of the following commands:
      - Microsoft Windows:

        pip install wheel\irisnative-1.0.0-cp34.cp35.cp36.cp37-none-win_amd64.whl

      - UNIX®/Linux:

        pip install wheel/irisnative-1.0.0-cp34-abi3-linux_x86_64.whl

      - Apple macOS:

        pip install wheel/irisnative-1.0.0-cp34-abi3-macosx_10_13_x86_64.macosx_10_14_x86_64.whl

   b. Run the code with the command `python hello_world.py`.
5. The Output pane displays the message `Hello World! You have successfully connected to InterSystems IRIS.`

Note: The Python code sample and supporting wheel files are designed for Python 3 because Python 2 will retire in 2020; Python 2 versions are available from InterSystems Learning Services.

7 JavaScript IDEs

This section provides instructions for connecting your favorite IDE for JavaScript and Node.js, such as Visual Studio Code, Vim, or Webstorm, to InterSystems IRIS.

1. In your web browser, download or clone https://github.com/intersystems/Samples-nodejs-helloworld.
2. In your IDE, open the folder you recently downloaded or cloned from GitHub, then open hello_world.js.
3. Configure the username, password, IP address, and port for your instance (see InterSystems IRIS Connection Information) using the variable declarations.
4. In the Terminal of your IDE, do the following:
   a. Install the InterSystems Native SDK for Node.js with the command `npm install --save intersystems-iris-native`.
   b. Run the code with the command `node hello_world.js`.

5. The Output pane displays the message `Hello World! You have successfully connected to InterSystems IRIS.`