First Look: The InterSystems .NET Gateway

Version 2019.4
2020-01-28

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

Table of Contents

First Look: The InterSystems .NET Gateway

This First Look guide introduces you to the fundamentals of using the .NET Gateway for InterSystems IRIS® data platform by giving you a focused overview and a basic, hands-on example. You will learn how InterSystems IRIS interoperates with .NET assemblies, and in the example, you will create a .NET Gateway and make calls from proxy classes in InterSystems IRIS to a basic DLL.

This document is simple by design; when you bring the .NET Gateway to your production systems, there may be things you need to do differently. The resources at the end of this document provide detailed and complete information on using the .NET Gateway in production.

To browse all of the First Looks, including others that can be performed on a free cloud instance or web instance, see InterSystems First Looks.

1 Why the .NET Gateway is Important

The InterSystems IRIS Object Gateway for .NET (also known as “the .NET Gateway”) provides an easy way for InterSystems IRIS to interoperate with Microsoft .NET Framework components. After importing a .NET DLL using the Gateway, you can instantiate an external .NET object and manipulate it as if it were a native object within InterSystems IRIS by means of proxy classes.

Each proxy object communicates with a corresponding .NET object, providing you with access to your .NET classes and methods from within InterSystems IRIS and ObjectScript. A call to any InterSystems IRIS proxy method causes a message to be sent to a .NET Gateway worker thread, which finds the appropriate method or constructor call and invokes it. The results of the invocation are sent back to the proxy object, and the proxy method returns the results to the InterSystems IRIS application.

In general, the best approach to using the .NET Gateway is to build a small wrapper class for the third party DLL in your application, then create a proxy for the wrapper. A wrapper classes exposes only the functionality you want, rather than creating a large number of proxy classes that your application may not need.

2 Exploring the .NET Gateway

In this hands-on example, you will:

- Create a DLL that has sample classes to call from InterSystems IRIS
- Define a .NET Gateway and start the server process
- Create an ObjectScript class to generate proxy classes from the DLL
- Create another ObjectScript class to connect to the Gateway and manipulate the DLL through proxy objects

Want to try an online video-based demo of InterSystems IRIS .NET development and interoperability features? Check out the .NET QuickStart!

2.1 Before You Begin

To run this demo you'll need a Windows 10 system running version 4.5 of the Microsoft .NET Framework, Visual Studio, and an installed instance of InterSystems IRIS. (For instructions for installing InterSystems IRIS, see InterSystems IRIS Basics: Installation.)

For instructions for configuring Visual Studio to connect to your InterSystems IRIS instance, see InterSystems IRIS Connection Information and .Net IDEs in *InterSystems IRIS Basics: Connecting an IDE*.

You will also use Atelier, the Eclipse-based IDE for InterSystems IRIS, to create ObjectScript code in your instance. For instructions for setting up Atelier and connecting it to your instance, see Atelier in *InterSystems IRIS Basics: Connecting an IDE*; more detailed information is available in *First Look: Atelier with InterSystems Products*. (You can also use the Studio IDE from InterSystems, a client application running on Windows systems, to create the ObjectScript code; for more information, see *Using Studio* and Studio in *Connecting an IDE*.)

2.2 Create the DLL

Using Visual Studio, create a class called Person and copy the following C# code. Use .NET 4.5 for this example.

```
public class Person {

    public int age;
    public String name;

    //constructor
    public Person (int startAge, String Name) {

        age = startAge;
        name = Name;

    }

    public void setAge(int newAge) {

        age = newAge;
    }

    public String getName() {

        return name;
    }

    public int getAge() {

        return age;
    }

    public static void main(String []args) {

        Person myPerson = new Person (5, "Tom");
        Console.Out.WriteLine(myPerson.getName());
        Console.Out.WriteLine(myPerson.getAge());

    }
}
```

Compile the Person class and generate a Person.dll file. Note the location of the DLL, as you will need it later.

2.3 Create and Start a .NET Gateway

Create and start the .NET Gateway:

1. Open the Management Portal for your instance in your browser, using the URL described in InterSystems IRIS Connection Information in *InterSystems IRIS Basics: Connecting an IDE*.

2. Navigate to **System Administration** > **Configuration** > **Connectivity** > **Object Gateways**.

3. Select **Create New Gateway**.
4. In the **Object Gateway For** field, select **.NET**.
5. Enter a **Gateway Name**.
6. Accept the default **Server Name/IP Address** of `"127.0.0.1"`.
7. In the **Port** field, enter `55000`.
8. For **.NET Version**, select `4.5`.
9. Select **Save**.
10. On the **Object Gateways** page, locate the Gateway you just defined and select **Start**.

2.4 Generate Proxy Classes

Using Atelier, create a new ObjectScript class in the USER namespace of your instance called CreateProxyClasses.cls and paste in the following code, substituting the host identifier for your instance for `server` in `gwyConn.%Connect`, and replacing `YOUR FILEPATH HERE` with the complete file path of your Person.dll file, enclosed in double quotes.

```
Class User.CreateProxyClasses Extends %Persistent
{

ClassMethod run()
{
    // get a connection to the .NET Gateway
    set gwyConn = ##class(%Net.Remote.Gateway).%New()
    set status = gwyConn.%Connect("127.0.0.1", 55000, "USER")
    if $$$ISERR(status) {
        write !,"error: "_$system.OBJ.DisplayError(status)
        quit
    }

    // add the DLL to the classpath
    set classpath = ##class(%ListOfDataTypes).%New()
    do classpath.Insert("YOUR FILEPATH HERE")
    set status = gwyConn.%AddToCurrentClassPath(classpath)
    if $$$ISERR(status) {
        write !,"error: "_$system.OBJ.DisplayError(status)
        quit
    }

    // create the proxy ObjectScript classes corresponding to the .NET classes in the DLL
    set status = gwyConn.%Import("Person",,,,1)
    if $$$ISERR(status) {
        write !,"error: "_$system.OBJ.DisplayError(status)
        quit
    }

    //close the connection to the .NET Gateway
    set status = gwyConn.%Disconnect()
    if $$$ISERR(status) {
        write !,"error: "_$system.OBJ.DisplayError(status)
        quit
    }

}

}
```

Compile and build the class. Then open the InterSystems IRIS Terminal in the USER namespace, using the instructions for your instance in *InterSystems IRIS Basics: Connecting an IDE*, and execute the following command:

```
do ##class(User.CreateProxyClasses).run()
```

2.5 Use ObjectScript to Manipulate .NET Objects

Create another ObjectScript class in the USER namespace called ManipulateObjects.cls and paste in the following code (again substituting the host identifier for your instance for `server` in `gwyConn.%Connect`):

```
Class User.ManipulateObjects Extends %Persistent
{

ClassMethod run()
{
    // get a connection to the .NET gateway
    set gwyConn = ##class(%Net.Remote.Gateway).%New()
    set status = gwyConn.%Connect("127.0.0.1", 55000, "USER")
    if $$$ISERR(status) {
        write !,"error: "_$system.OBJ.DisplayError(status)
        quit
    }

    // manipulate some proxy objects
    set person = ##class(User.Person).%New(gwyConn,5,"Tom")
    write !,"Name: "_person.getName()
    write !,"Age: "_person.getAge()
    do person.setAge(100)
    write !,"Age: "_person.getAge()

    // close the connection to the .NET gateway
    set status = gwyConn.%Disconnect()
    if $$$ISERR(status) {
        write !,"error: "_$system.OBJ.DisplayError(status)
        quit
    }

}

}
```

Compile and build the class, and then execute the following command in Terminal, in the USER namespace:

```
do ##class(User.ManipulateObjects).run()
```

You should see the following output:

```
Name: Tom

Age: 5

setting age to 100

Age: 100
```

Now that you have successfully completed the exercise, stop the .NET Gateway that you created. Return to the **Object Gateways** page in the Management Portal, locate the Gateway, and select **Stop**.

3 Learn More About the .NET Gateway

From here, you are able to continue exploring the .NET Gateway and InterSystems IRIS. Consult the documentation and resources below to learn about .NET, interoperability, application development, and more.

- *Using the Gateway for .NET* — Learn more about interoperability between InterSystems IRIS and Microsoft .NET Framework components.
- Skyrocket Your .NET Application Development — A presentation from InterSystems on modeling and accessing data as objects in .NET.
- .NET Documentation — Documentation from Microsoft on .NET, including architectural concepts, tutorials, and development guides.