InterSystems
IRIS Data Platform

InterSystems IRIS Demo: Connecting with ADO.NET

Version 2025.1
2025-06-03

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

Table of Contents

InterSystems IRIS Demo: Connecting with ADO.NET

This article explains how to connect to InterSystems IRIS® data platform via the InterSystems ADO.NET Managed Provider. Once you have completed the demo that follows, you will have configured a Visual Studio project to use the InterSystems.Data.IRISClient.dll assembly, established an ADO.NET connection to InterSystems IRIS, run several SQL statements from your .NET application, and confirmed the effects of these statements in the InterSystems IRIS System Management Portal.

To give you a taste of the ADO.NET Managed Provider without bogging you down in details, we've kept this exploration simple. These activities are designed to only use the default settings and features, so that you can acquaint yourself with the fundamentals of the feature without having to deal with details that are off-topic or overly complicated. When you bring ADO.NET to your production systems, there may be things you will need to do differently. Be sure not to confuse this exploration of ADO.NET with the real thing! The sources provided at the end of this document will give you a good idea of what is involved in using ADO.NET in production.

For more documentation and other learning resources for ADO.NET and InterSystems IRIS, see Next Steps.

1 Why ADO.NET Is Important

ADO.NET is a data access technology from the Microsoft .NET Framework that provides access to data sources. It is used to establish database connectivity and provides a standard, reliable way for .NET Framework programmers to connect to many types of data sources or perform operations on them with SQL. Connecting to InterSystems IRIS via the ADO.NET Managed Provider is simple, especially if you've used ADO.NET before. Establishing an ADO.NET connection to InterSystems IRIS from a .NET application allows you to run SQL commands against InterSystems IRIS databases from your .NET application.

If you're new to InterSystems IRIS but familiar with .NET and SQL, you can use your existing expertise right away to help you become familiar with the database platform. You can test ADO.NET connections and SQL commands in a development environment with just a few lines of code.

2 ADO.NET and InterSystems IRIS

InterSystems IRIS is a fully compliant implementation of the ADO.NET specification. The InterSystems ADO.NET Managed Provider provides easy relational access to data. It processes ADO.NET method calls from applications and submits SQL requests to InterSystems IRIS. It then returns results to the calling application — in this case, your .NET application.

Connecting to InterSystems IRIS via ADO.NET is a very straightforward process.

In order to use InterSystems IRIS ADO.NET capability, you must first add the InterSystems.Data.IRISClient.dll assembly as a dependency to your Visual Studio project. After confirming a few settings, use our sample code to establish an ADO.NET connection to InterSystems IRIS and to execute SQL queries. Note that the InterSystems.Data.IRISClient.dll assembly is implemented using .NET managed code throughout, making it easy to deploy within a .NET environment. It is thread-safe and can be used within multithreaded .NET applications.

3 Exploring ADO.NET

We have developed a brief demo that shows you how to work with ADO.NET and InterSystems IRIS.

3.1 Before you Begin

To use this procedure, you will need a Windows system to work on, with the .NET framework and Visual Studio installed, and a running InterSystems IRIS instance to connect to. Your choices for InterSystems IRIS include several types of licensed and free evaluation instances; the instance need not be hosted by the system you are working on (although they must have network access to each other). For information on how to deploy each type of instance if you do not already have one to work with, see Deploying InterSystems IRIS in *InterSystems IRIS Basics: Connecting an IDE*. Connect Visual Studio to your InterSystems IRIS instance using the information in InterSystems IRIS Connection Information and .Net IDEs in the same document.

3.2 Configuring the Visual Studio Project

In the Visual Studio main menu, create a new Project by selecting **File** > **New** > **Project**. In the resulting dialog, click the **Visual C#** option, and choose **Console App (.NET Framework)**. For the **Name** field, enter `ADONET`. Click **OK**. This should create a new console application using the .NET Framework.

Next, in the Visual Studio main menu, select **Project** > **ADONET Properties**. Under **Target framework**, select **.NET Framework 4.6.2**.

Note: This demo uses .NET Framework 4.6.2, but it is possible to use any supported version of .NET or .NET Framework if it is installed on your system. For current supported versions and file locations, see Supported .NET Frameworks in *InterSystems Supported Platforms*.

3.2.1 Adding the Assembly Reference

The InterSystems.Data.IRISClient.dll assembly must be installed on your local system. You can download the assembly from the InterSystems IRIS Driver Packages page. If InterSystems IRIS is installed on your local system or another you have access to, the assembly is already installed in the subdirectory *install-dir*\dev\dotnet\bin\v4.6.2, where *install-dir* is the installation directory for the instance.

To add an assembly reference to InterSystems.Data.IRISClient.dll to a project:

1. From the Visual Studio main menu, select **Project** > **Add Reference...**
2. In the resulting window, click **Browse...**.
3. Browse to the location of the InterSystems.Data.IRISClient.dll file.
4. Select the file and click **Add**.
5. Click **OK**.

In the Visual Studio Solution Explorer, the InterSystems.Data.IRISClient.dll assembly should now be listed under **References**.

3.3 Connecting via ADO.NET

At this point, you are ready to connect to InterSystems IRIS from your .NET application. The connection string for the InterSystems ADO.NET Managed Provider is made up of key-value pairs that define the connection properties. The connection string syntax is:

`Server=`*host_IP*`; Port=`*superserverPort*`; Namespace=`*namespace*`; Password=`*password*`; User ID=`*username*`;`

where the variables represent the InterSystems IRIS instance host's IP address, the instance's superserver port, a namespace on the instance, and credentials for the instance. This is the same information you used to connect Visual Studio to your instance, as described in Before You Begin.

Update this information in the code that follows after you paste it into Visual Studio. You can set *namespace* to the predefined namespace **USER**, as shown, or to another namespace you have created on your installed instance.

```
using System;
using InterSystems.Data.IRISClient;

namespace ADONET
{
    class Program
    {
        static void Main(string[] args)
        {

            String host = "<host>";
            String port = "<port>";
            String username = "<username>";
            String password = "<password>";
            String Namespace = "USER";

            IRISConnection IRISConnect = new IRISConnection();
            IRISConnect.ConnectionString = "Server = " + host
                + "; Port = " + port + "; Namespace = " + Namespace
                + "; Password = " + password + "; User ID = " + username;

            IRISConnect.Open();


            String sqlStatement1 = "CREATE TABLE People(ID int, FirstName varchar(255), LastName 
varchar(255))";
            String sqlStatement2 = "INSERT INTO People VALUES (1, 'John', 'Smith')";
            String sqlStatement3 = "INSERT INTO People VALUES (2, 'Jane', 'Doe')";
            String queryString = "SELECT * FROM People";


            IRISCommand cmd1 = new IRISCommand(sqlStatement1, IRISConnect);
            IRISCommand cmd2 = new IRISCommand(sqlStatement2, IRISConnect);
            IRISCommand cmd3 = new IRISCommand(sqlStatement3, IRISConnect);
            IRISCommand cmd4 = new IRISCommand(queryString, IRISConnect);


            //ExecuteNonQuery() is used for CREATE, INSERT, UPDATE, and DELETE SQL Statements
            cmd1.ExecuteNonQuery();
            cmd2.ExecuteNonQuery();
            cmd3.ExecuteNonQuery();

            //ExecuteReader() is used for SELECT
            IRISDataReader Reader = cmd4.ExecuteReader();

            Console.WriteLine("Printing out contents of SELECT query: ");
            while (Reader.Read())
            {

              Console.WriteLine(Reader.GetValue(0).ToString() + ", " + Reader.GetValue(1).ToString() 
+ ", " \

                   + Reader.GetValue(2).ToString());

            }

            Reader.Close();
            cmd1.Dispose();
            cmd2.Dispose();
            cmd3.Dispose();
            cmd4.Dispose();
            IRISConnect.Close();

            Console.WriteLine("Press any key to continue...");
            Console.ReadKey();

        }
    }
}
```

Run the code by clicking the **Start** button, or by pressing F5.

If the connection and queries have completed successfully, you should see a console window containing the results of the SELECT query.

3.4 Confirming the Changes in the Management Portal

Next, confirm your results in the Management Portal, using the following procedure:

1. Open the Management Portal for your instance in your browser, using the URL described for your instance in *InterSystems IRIS Basics: Connecting an IDE*.
2. If you are not in the namespace you specified in the code, switch to it (click **%SYS**, or whatever namespace is shown, in the **Namespace:** indicator at the top of the page).
3. Navigate to the **SQL** page (**System Explorer** > **SQL**), then click the **Execute Query** tab and paste in the following SQL query:

   ```
   SELECT
   ID, FirstName, LastName
   FROM SQLUser.People
   ```

   Click **Execute**. The page should display the contents of the People table created in the sample code.

4 Next Steps

To learn more about ADO.NET, SQL, and InterSystems IRIS, see:

- Using the InterSystems Managed Provider for .NET
- Using InterSystems SQL
- ADO.NET Overview

To learn about all of the available technologies for connecting .NET applications to InterSystems IRIS, see Connecting .NET Applications to InterSystems Products, a learning path providing access to videos, courses, and exercises as well as documentation.