AUMERIAL

NTi DATA PROVIDER

QUICK INSTALLATION GUIDE

IBM Silver Partner

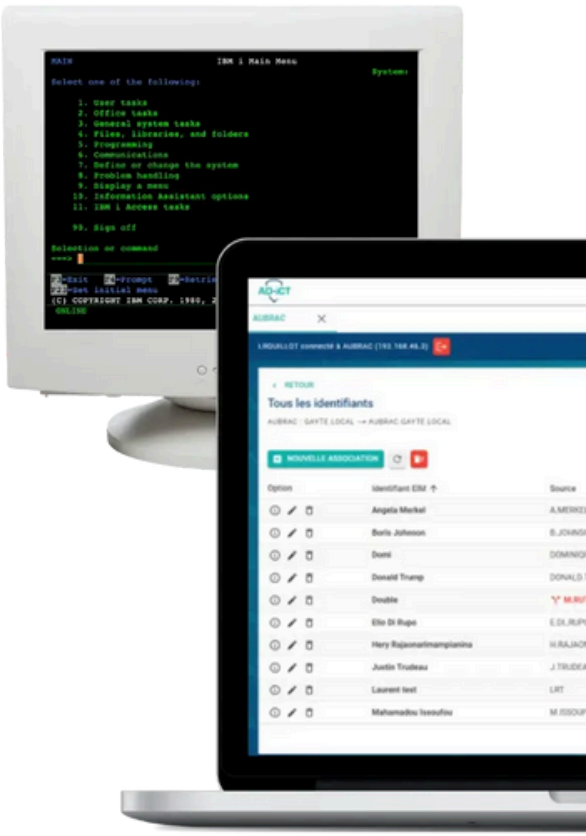
Connect .NET and your IBM i with NTi, for instant access to data, programs, and much more.

INTRODUCTION

NTi is designed to meet a reality that few tools address: you can connect your IBM i to .NET and be operational **in less than 15 minutes.** No disruption in your activities, no training required, no complex configuration.

Everything integrates directly into Visual Studio **through a simple reference**. You code in C# with a familiar syntax and an immediate handling, without tedious preliminaries related to RPG.

Moreover NTi is entirely multi-platform, running as well on **Windows**, **Linux** as **macOS**, and agnostic to IBM i versions. No matter your environment or your system version.



PREREQUISITES

IBM i side:

- TCP/IP services started (*DATABASE, *RMTCMD, *SIGNON)
- License script provided by Aumerial executed via Run SQL Scripts

Once the license installed on IBM i,
everything happens on the .NET side.

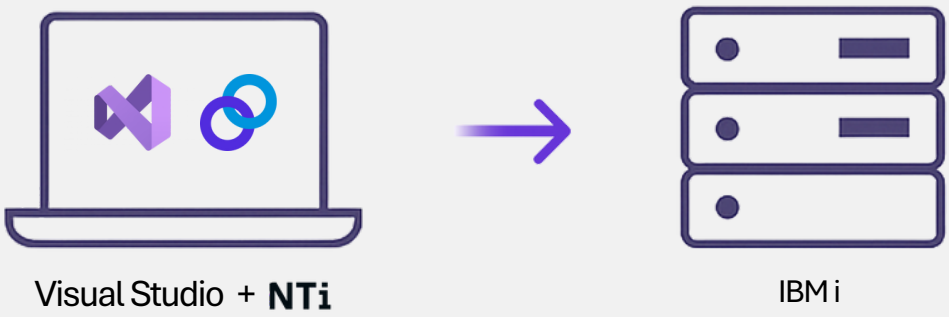
.NET development:

Visual Studio 2022 or higher

.NET 6 or later

NuGet

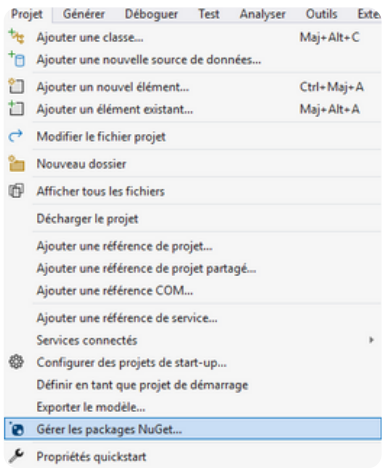
IBM i credentials (host, user, password)



NTi INSTALLATION

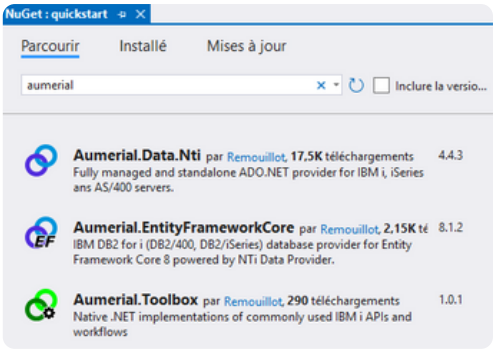
1 Open a Visual Studio project.

Right-click on the solution, then select Manage NuGet Packages.



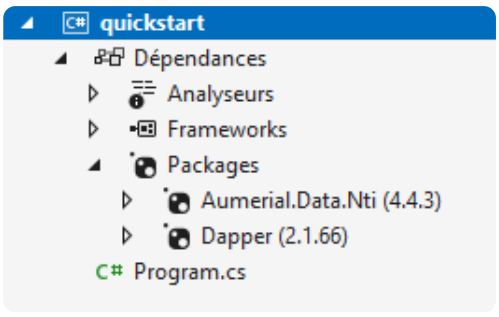
2 Install NTi and Dapper

NTi installs like a standard NuGet package. We also recommend **Dapper**, a lightweight ORM which simplifies SQL query execution and object mapping.



3 Check the installation in Visual Studio

Make sure that **Aumerial.Data.NTi** and **Dapper** appear correctly in the project tree.



CREATE A CONNECTION

1. Reference NTi in the code

Add this line at the beginning of your C# file.

```
using Aumerial.Data.Nti;
using Dapper;
```

2. Create the connection string

Implement the connection string to your IBM i.

```
var conn = new NTiConnection("server=serverName;user=userName;password=password");
conn.Open();
```

3. Test the connection

Use a *try/catch* block to check if the connection is well established and to retrieve possible errors

```
try
{
    using var conn = new NTiConnection("server=server;user=user;password=password");
    conn.Open();

    Console.WriteLine(conn.State == ConnectionState.Open
        ? "Connection to IBM i and NTi license OK."
        : "Connection failed.");
}
catch (Exception ex)
{
    Console.WriteLine("Error while connecting: " + ex.Message);
}
```

CRUD SQL WITH DAPPER

1. CREATE - Insert data

```
conn.Execute("INSERT INTO MYLIB.MYTABLE (ID, NAME) VALUES (@Id, @Name)",
             new { Id = 1002, Name = "DOE" });
```

2. READ - Read data

```
var rows = conn.Query("SELECT * FROM MYLIB.MYTABLE");
foreach (var row in rows)
{
    Console.WriteLine($"{row.ID} - {row.Name}");
}
```

3. UPDATE - Update data

```
conn.Execute("UPDATE MYLIB.MYTABLE SET NAME = @Name WHERE ID = @Id",
             new { Id = 100, Name = "John" });
```

4. DELETE - Delete data

```
conn.Execute("DELETE FROM MYLIB.MYTABLE WHERE ID = @Id",
             new { Id = 1002 });
```

IBM i ADMINISTRATION

1. Execute a CL command

```
conn.ExecuteClCommand("CRTLIB LIB(MYLIB) TEXT('My new lib')");
```

2. Call a program with parameter

```
// Create parameter list
List<NTiProgramParameter> parms = new List<NTiProgramParameter>() {
    new NTiProgramParameter("Hello", 10).AsInput(),       // CHAR(10) INPUT
    new NTiProgramParameter("World", 10).AsInput(),       // CHAR(10) INPUT
    new NTiProgramParameter(new byte[] {0x00}).AsInput(), // CHAR(1) INPUT
    new NTiProgramParameter(128).AsInput(),               // INTEGER (4 bytes) INPUT
    new NTiProgramParameter("", 128).AsOutput(),          // CHAR(128) OUTPUT
    new NTiProgramParameter("", 50)                       // CHAR(50) I/O
};
// Call the program
conn.CallProgram("MYLIB", "MYPGM", parms);

// Retrieve output parameters
string message1 = parms[4].GetString(0, 64);
string message2 = parms[4].GetString(64, 64);

// Close connection
conn.Close();
```

3. Call a stored procedure

```
var result = conn.Query("MYLIB.MYPROC", new { Param1 = 123, Param2 = "ABC" },
                        commandType: CommandType.StoredProcedure);

foreach (var row in result)
{
    Console.WriteLine($"{row.Col1} - {row.Col2}");
}
```

CONTACT

A team at your service.
Request your trial license for free today.

France - WW

Rémi ROUILLOT

+33 6 86 83 91 09

remi.rouillot@aumerial.com

DACH - PL

Tomasz AFELTOWICZ-SCHULTZ

+49 179 421 6006

tafs@aumerial.com

www.linkedin.com/company/aumerial

www.youtube.com/@aumerial

hub.docker.com/orgs/aumerial/repositories

github.com/Aumerial

The future of IBM i is written in .NET

AUMERIAL

www.aumerial.com

www.documentation.aumerial.com