AUMERIAL

NTi
DATA PROVIDER

GUIDE D'INSTALLATION RAPIDE

IBM Silver Partner

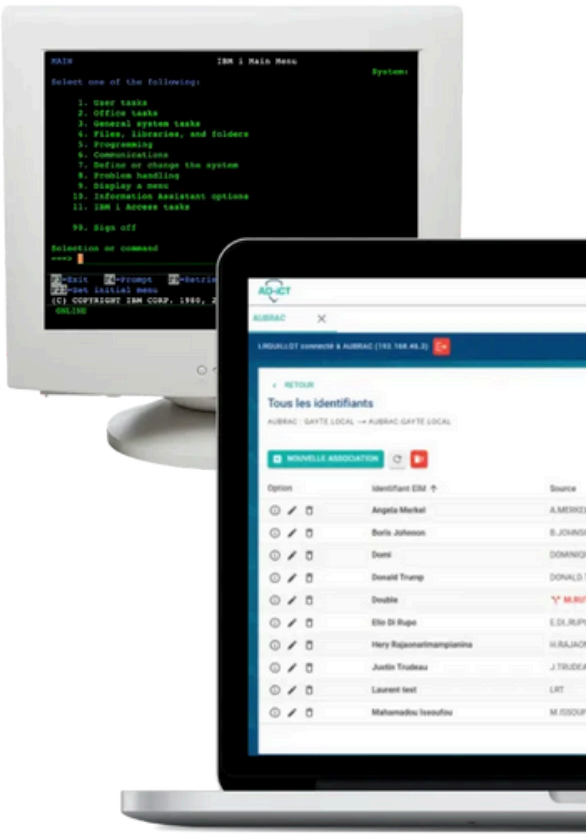
Connectez .NET et votre IBM i avec NTi, pour un accès immédiat aux données, programmes et bien plus encore.

INTRODUCTION

NTi est conçu pour répondre à une réalité que peu d'outils adressent : vous pouvez connecter votre IBM i à .NET et être opérationnel en **moins de 15 minutes**. Aucune rupture dans vos activités, aucune formation nécessaire, aucune configuration complexe.

Tout s'intègre directement à Visual Studio **par simple référencement.** Vous codez en C# avec une syntaxe familière et une prise en main immédiate, sans préambules fastidieux liés au RPG.

De plus NTi est entièrement multiplateforme, fonctionnant aussi bien sur **Windows**, **Linux** que **macOS**, et agnostique aux versions IBM i. Peu importe votre environnement ou la version de votre système.



PRÉ-REQUIS

Côté IBM i:

- Services TCP/IP démarrés (*DATABASE, *RMTCMD, *SIGNON)
- Script de licence fourni par Aumerial éxécuté via Run SQL Scripts

Une fois la licence installé sur IBM i,
tout se déroule côté .NET

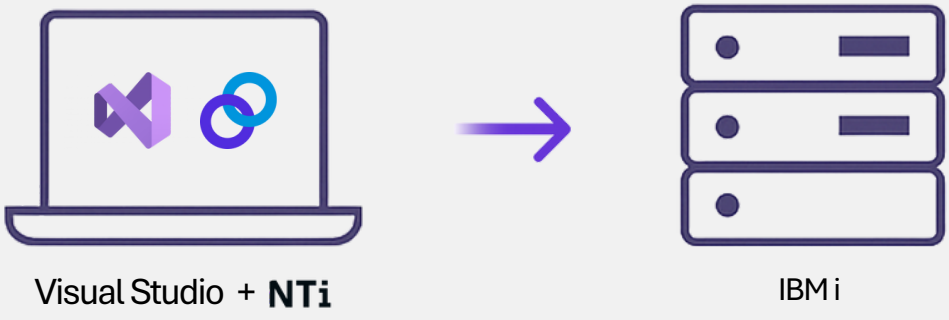
Développement .NET:

Visual Studio 2022 ou supérieur

.NET 6 ou ultérieur

NuGet

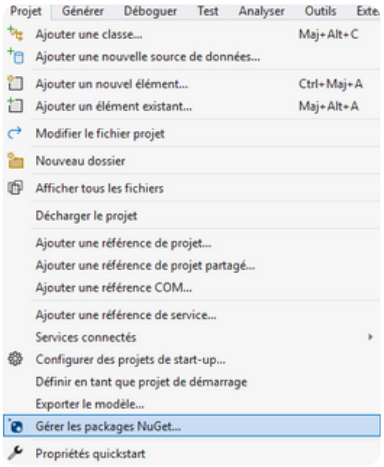
Identifiants IBM i (host, user, password)



INSTALLATION NTi

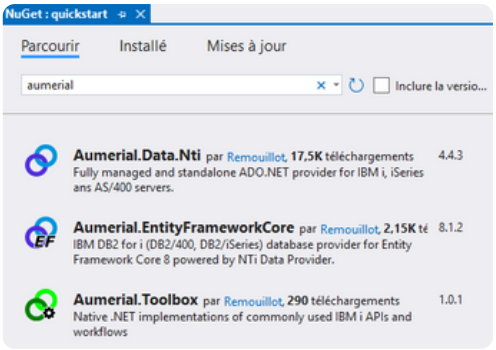
1 Ouvrez un projet Visual Studio.

Faites un clic droit sur la solution, puis sélectionnez Gérer les packages NuGet.



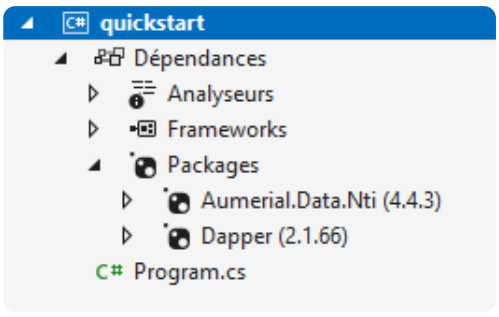
2 Installez NTi et Dapper

NTi s'installe comme un package NuGet standard. Nous recommandons aussi **Dapper**, un ORM léger qui simplifie l'exécution de requête SQL et le mapping d'objets.



3 Vérifiez l'installation dans Visual Studio

Assurez-vous que **Aumerial.Data.NTi** et **Dapper** apparaissent bien dans l'arborescence du projet.



CRÉER UNE CONNEXION

1. Référencez NTi dans le code

Ajoutez cette ligne au début de votre fichier C#.

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

2. Créez la chaîne de connexion

Implémentez la chaîne de connexion à votre IBM i.

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

3. Testez la connexion

Utilisez un bloc *try/catch* pour vérifier si la connexion est bien établie et récupérer d'éventuelles erreurs

```
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 AVEC DAPPER

1. CREATE - Insérer des données

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

2. READ - Lire des données

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

3. UPDATE - Mettre à jour des données

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

4. DELETE - Supprimer des données

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

ADMINISTRATION IBM i

1. Exécuter une commande CL

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

2. Appeler un programme avec paramètre

```
// 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. Appeler une procédure stockée

```
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

Une équipe à votre service.
Demandez votre licence d'essai gratuitement dès aujourd'hui.

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

Le futur de l'IBM i s'écrit en .NET

AUMERIAL

www.aumerial.com

www.documentation.aumerial.com