HCL OneDB 2.0.1

OneDB .NET EF Core Provider Reference Guide

Contents

Chapter 1. HCL OneDB™ .NET EF Core Provider Guide

These topics assume you are familiar with the Microsoft™ .NET EF Core specification and using HCL OneDB™ servers and databases.

Overview of the HCL OneDB™ .NET EF Core Provider

The topics in this overview describe the HCL® OneDB® .NET EF Core Provider and provide information on the environment in which to use it. These topics also provide installation and connection information and general information to help you get started using the HCL® OneDB® .NET EF Core Provider.

What is the HCL OneDB™ .NET EF Core Provider?

The HCL® OneDB® .NET EF Core Provider or HCL® OneDB® .NET Entity Framework Core Provider is based on open source and cross platform version of Framework database access technology.

HCL® OneDB® .NET EF Core Provider is popular for the following reasons:

- Cross Platform application support
- Application migration is easy with very limited database provider specific code.

Supported programming environments

The HCL® OneDB® .NET EF Core Provider can be used by any application that can be run by the .NET Core.

The HCL® OneDB® .NET EF Core Provider is shipped with CSDK 2.0.0.0. It is available on Windows x64 and Linux x86_64.

OneDB.EntityFramework.Core.dll is the name of Provider on both platforms and located at `$ONEDB_HOME\bin` and is built on top of OneDB .NET Core & ODBC Driver .

.NET EF Core Provider is built on following environment:

- Microsoft Windows Server 2016 Standard : v1607 Build 14393.2551
- .NET Core SDK Version : v3.1.410
- .NET EF Core version : v5.0.9
- Microsoft Visual Studio Enterprise 2017 : v15.8.6

HCL OneDB™ .NET EF Core Applications

The .NET Core applications using the HCL® OneDB® .NET EF Core Provider need other .NET Core assemblies/libraries from .NET Runtime. Such applications need to download/acquire the required assembly files (.NET Core SDK/Runtime).

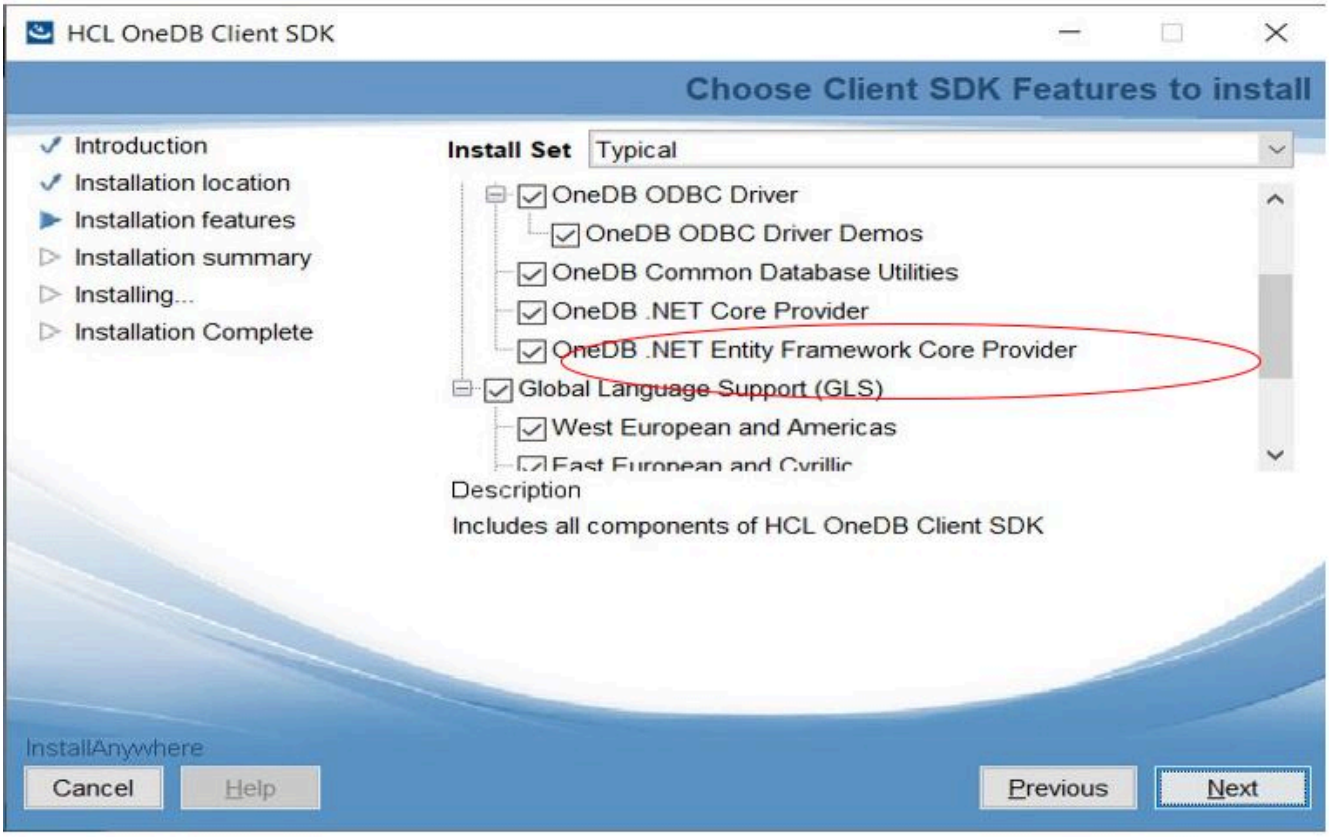
In order to run such applications, all the OneDB related environment variables required i.e. ONEDB_HOME,ONEDB_SQLHOSTS, ONEDB_SERVER, PATH etc must be appropriately set.

For Sample program, see Example on page 5.

Installing the HCL® OneDB® .NET EF Core Provider

You can install the HCL® OneDB® .NET EF Core Provider with the HCL OneDB™ Client Software Development Kit (Client SDK) through a typical or custom installation.

When product is installed on Windows and Linux platforms, .NET EF core provider binary **OneDB.EntityFramework.Core.dll** will be avialble in `%ONEDB_HOME%\bin` directory.



Using HCL® OneDB® .NET EF Core Provider

UseOneDB(<Connection String>) is only override method specific to OneDB, every .NET EF Core application calls method **OnConfiguring**(), in this method we need to call database provider specific connection API. Any database specific parameters needs to be passed as part of connection string.

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
optionsBuilder.UseOneDB("Database=efcoredb; server=onedb;User ID=user1; Password=xxxx; Host=127.0.0.1;
 Protocol=onsoctcp; Service=8909; connectdatabase=no");
}
```

Note: All the connection options are same as .NET Core Provider (ConnectionString property).

Supported Data types

Following data types are supported currently:

- INT
- CHAR, NCHAR
- VARCHAR, NVARCHAR
- DATETIME YEAR to MONTH
- DATE
- MONEY
- DECIMAL
- BIGINT
- SMALLINT
- SERIAL
- BIGSERIAL
- FLOAT
- SMALLFLOAT

Attention: Migration and Scaffolding operations: There are some limitations if tables has PK/FKs.

Example

```
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Migrations;
using Microsoft.EntityFrameworkCore.Storage;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using static EFCoreSql.SchoolContext;

namespace EFCoreSql
{
    class SchoolContext : DbContext
    {
        public SchoolContext() : base()
        { }
        public class Student
        {
            [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
            public int StudentId { get; set; }
            public string Name { get; set; }
        }
        public class Course
        {
            public int CourseId { get; set; }
```

```
            public string CourseName { get; set; }
        }
        public DbSet<Student> Students { get; set; }
        public DbSet<Course> Courses { get; set; }
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseOneDB("Database=efcoredb; server=onedb;User ID=user1; Password=xxxx;
Host=127.0.0.1; Protocol=onsoctcp; Service=8909; connectdatabase=no");
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            int rowCount = 1;

            using (var schoolContext = new SchoolContext())
            {
                schoolContext.ChangeTracker.LazyLoadingEnabled = false;

                bool _canCo = schoolContext.Database.CanConnect();
                if (_canCo == true)
                    Console.WriteLine("OneDB database is good");
                else
                    Console.WriteLine("Not able to connect to OneDB database!!");

                bool _Created = schoolContext.Database.EnsureCreated();
                if (_Created == false)
                    Console.WriteLine("Database exist in the OneDB");
                else
                    Console.WriteLine("Database either got created or became the current database " +
                        "since connection string is using connectdatabase=no, and tables got " +
                        "created if it didn't exist in OneDB!!");

                List<Student> stds = new List<Student>()
                {
                    new Student(){StudentId=1},
                    new Student(){StudentId=2},
                    new Student(){StudentId=3}
                };

                Console.WriteLine("Deleting all records from students table...");
                schoolContext.Students.RemoveRange(stds);
                schoolContext.SaveChanges();

                Console.WriteLine("Selecting record(s) from students table...");
                var _students = schoolContext.Students.FromSqlRaw("Select * from students;");
                var counter = _students.GetEnumerator();
                bool hasRow = counter.MoveNext();
                if (hasRow is false)
                    Console.WriteLine("No records found - Expected");
                else
                {
                    Console.WriteLine("Records found - NOT Expected!!");
                    rowCount = 1;
                    foreach (var row in _students)
                    {
```

```
                    Console.WriteLine("Row #" + rowCount + " = StudentId : " + row.StudentId
                        + ", Name : " + row.Name);
                    rowCount++;
                }
            }

            Console.WriteLine("Inserting 3 records into students table...");
            // Insert records into Students table.
            var std1 = new Student() { StudentId = 1, Name = "Sheshnarayan" };
            var std2 = new Student() { StudentId = 2, Name = "Vardaan" };
            var std3 = new Student() { StudentId = 3, Name = "Samriddhi" };

            schoolContext.Students.AddRange(std1, std2, std3);
            schoolContext.SaveChanges();

            Console.WriteLine("Selecting record(s) from students table...");
            _students = schoolContext.Students.FromSqlRaw("Select * from students;");
            counter = _students.GetEnumerator();
            hasRow = counter.MoveNext();
            if (hasRow is false)
                Console.WriteLine("No records found - NOT Expected!!");
            else
            {
                Console.WriteLine("Records found - Expected");
                rowCount = 1;
                foreach (var row in _students)
                {
                    Console.WriteLine("Row #" + rowCount + " = StudentId : " + row.StudentId + ", Name : "
+ row.Name);
                    rowCount++;
                }
            }
        }

        using (var updateschoolContext = new SchoolContext())
        {
            updateschoolContext.ChangeTracker.LazyLoadingEnabled = false;

            bool _Created = updateschoolContext.Database.EnsureCreated();
            if (_Created == false)
                Console.WriteLine("Database exist in the OneDB");
            else
                Console.WriteLine("Database either got created or became the " +
                    "current database since connection string is using " +
                    "connectdatabase=no, and tables got created if it didn't exist in OneDB!!");

            Console.WriteLine("Updating Name column value to Monika of Students table " +
                "where StudentID=1...");
            // Update records into Students table.
            var std4 = new Student() { StudentId = 1, Name = "Monika" };

            updateschoolContext.Students.Update(std4);
            updateschoolContext.SaveChanges();

            Console.WriteLine("Selecting just updated record to ensure its updated correctly from students
table...");
```

```
                var _students = updateschoolContext.Students.FromSqlRaw("Select * from students where
 studentid=1;");
                var counter = _students.GetEnumerator();
                bool hasRow = counter.MoveNext();
                if (hasRow is false)
                    Console.WriteLine("No records found - NOT Expected!!");
                else
                {
                    Console.WriteLine("Records found - Expected");
                    rowCount = 1;
                    foreach (var row in _students)
                    {
                        Console.WriteLine("Row #" + rowCount + " = StudentId : " + row.StudentId + ", Name : "
 + row.Name);
                        rowCount++;
                    }
                }
            }
        }
    }
}
```

Index