Oracle Data Provider for .NET Entity Framework Core

Developers Guide 18c (18.0) Beta 2

Pre-General Availability: 2019-2-7

Introduction

Oracle Data Provider for .NET (ODP.NET) Entity Framework (EF) Core is a database provider that allows Entity Framework Core to be used with Oracle databases.

Entity Framework Core is a cross-platform Microsoft object-relational mapper that enables .NET developers to work with relational databases using .NET objects.

ODP.NET EF Core consists of a single 100% managed code dynamic-link library, Oracle.EntityFrameworkCore.dll, available via a NuGet package. It uses the Oracle.EntityFrameworkCore namespace.

System Requirements

ODP.NET EF Core has similar system requirements as ODP.NET 18.3. ODP.NET EF Core requires the following:

- Operating System
  - Windows x64
    - Windows 8.1 (Pro and Enterprise Editions)
    - Windows 10 x64 (Pro, Enterprise, and Education Editions)
    - Windows Server 2012 R2 x64 (Standard, Datacenter, Essentials, and Foundation Editions)
    - Windows Server 2016 x64 (Standard and Datacenter Editions)
  - Linux x64
    - Oracle Linux 7
    - Red Hat Enterprise Linux 7
- .NET
  - .NET Core 2.1 or higher
  - .NET Framework 4.6.1 and higher
- Entity Framework Core
  - 2.1 or higher
- Dependent .NET Assemblies
  - ODP.NET Core 18.3 or higher
  - Microsoft.EntityFrameworkCore.Relational 2.1 or higher

- Access to Oracle Database 11g Release 2 (11.2) or higher

ODP.NET EF Core is compatible with ASP.NET Core 2.x and ASP.NET.

ODP.NET EF Core is built with AnyCPU. It supports 64-bit .NET and 32-bit .NET.

.NET Core 3.x nor EF Core 3.x are not yet supported.

Application Programming Interface

ODP.NET EF Core supports standard EF Core application programming interfaces. The provider contains additional extension methods specific to the provider.

UseOracle(@"<Connection String>")

This extension method sets the provider and database connection configuration. Developers can set any connection string attributes that are available in ODP.NET Core.

// C#

```
optionsBuilder.UseOracle(@"User Id=blog;Password=<password>;Data Source=pdborcl;");
```

Note: optionsBuilder is of type "DbContextOptionsBuilder".

UseOracleSQLCompatibility(string version)

This extension method specifies the database version generated SQL should be compatible with.

This method accepts either a value of "11" or "12". By default, generated SQL is compatible with database version 12 and higher. Customers using Oracle Database version 11.2 should set UseOracleSQLCompatibility("11").

// C#
optionsBuilder.UseOracle("User Id=hr;Password=<password>;Data Source = inst1", b => b.UseOracleSQLCompatibility("11"));

Note: optionsBuilder is of type "DbContextOptionsBuilder".

UseOracleIdentityColumn()

This extension method specifies the column to be an identity column or have it associated with a sequence and a trigger to have a server generated column value, depending on the value passed to UseOracleSQLCompatibility().

// C #

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
  modelBuilder.Entity<Blog>().Property(p => p.Id).UseOracleIdentityColumn();
}
```

Sample Code

This code sample demonstrates code necessary to that will create a blogging context of Blogs and Posts objects. EF Core will create database schema tables mapping to these two objects. When the application is run, it will add a new blog entry to the Blogs table, then retrieve that entry back to the application.

```
// C#

using Oracle.EntityFrameworkCore;
using System.Collections.Generic;
using Microsoft.EntityFrameworkCore;

namespace OracleBlog
{
    class Program
    {

        public class BloggingContext : DbContext
        {
            public DbSet<Blog> Blogs { get; set; }
            public DbSet<Post> Posts { get; set; }

            protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
            {
                optionsBuilder.UseOracle(@"User Id=blog;Password=<password>;Data
Source=pdborcl;");
            }
        }

        public class Blog
        {
            public int BlogId { get; set; }
            public string Url { get; set; }
            public List<Post> Posts { get; set; }
        }

        public class Post
        {
            public int PostId { get; set; }
            public string Title { get; set; }
            public string Content { get; set; }

            public int BlogId { get; set; }
            public Blog Blog { get; set; }
        }
        static void Main(string[] args)
        {
```

```
            using (var db = new BloggingContext())
            {
                var blog = new Blog { Url = "https://blogs.oracle.com" };
                db.Blogs.Add(blog);
                db.SaveChanges();
            }

            using (var db = new BloggingContext())
            {
                var blogs = db.Blogs;
            }

        }

    }
}
```

Using ODP.NET Core Classes

Developers can use the OracleConfiguration class and other ODP.NET Core classes in Entity Framework Core to access ODP.NET Core-specific functionality, such as the TNS ADMIN location or tracing settings. The ODP.NET Core assembly will already be part of any Oracle EF Core project since it is a dependency of Oracle.EntityFrameworkCore. Most commonly, developers will add the ODP.NET Core namespace to the project:

```
// C#

using Oracle.ManagedDataAccess.Client;
```

Then, add the desired OracleConfiguration property settings. These properties should be set prior to any EF Core code as OracleConfiguration settings must be made prior to opening an ODP.NET connection. The below example turns on tracing and sets a TNS ADMIN location which should contain the application's tnsnames.ora and sqlnet.ora files:

```
// C#

static void Main(string[] args)
{

        OracleConfiguration.TraceFileLocation = @"D:\traces";
        OracleConfiguration.TraceLevel = 7;
        OracleConfiguration.TnsAdmin = @"D:\tnsadmin";

        <Start Entity Framework Core code>

}
```

Oracle EF Core applications can use all the properties and behavior available in ODP.NET Core.

Code First Data Type Mapping

By convention, ODP.NET EF Core maps an appropriate database data type based on the .NET data type and its characteristics. This table shows the default mappings. Fluent APIs/Annotations can be used to map the .NET types to any valid Oracle datatype.

Table: ODP.NET Entity Framework Core Code First Data Type Default Mappings

.NET Type Alias	.NET Data Type	Required Fluent API(s)*	Oracle Database Type
bool	System.Boolean	None	NUMBER(1)
sbyte	System.Sbyte	None	NUMBER(3)
byte	System.Byte	None	NUMBER(4)
short/int16	System.Int16	None	NUMBER(5)
ushort/uint16	System.UInt16	None	NUMBER(5)
int/int32	System.Int32	None	NUMBER(10)
uint32	System.UInt32	None	NUMBER(10)
decimal	System.Decimal	None	NUMBER(18,2)
long/int64	System.Int64	None	NUMBER(19)
uint64	System.UInt64	None	NUMBER(20)
float	System.Float	None	BINARY_FLOAT
double	System.Double	None	BINARY_DOUBLE
DateTime	System.DateTime	None	TIMESTAMP(7)
DateTimeOffset	System.DateTimeOffset	None	TIMESTAMP(3) WITH TIME ZONE
TimeSpan	System.TimeSpan	None	INTERVAL DAY(8) TO SECOND(7)
char	System.Char	None	NVARCHAR2(1)
byte[]	System.Byte[]	None	RAW(2000)
byte[]	System.Byte[]	HasMaxLength(x <= 2000)	RAW(x)
byte[]	System.Byte[]	HasMaxLength(x > 2000)	BLOB
string	System.String	None	NVARCHAR2(2000)
string	System.String	IsUnicode(false) && IsFixedLength(false) && HasMaxLength(x > 4000)	CLOB
string	System.String	IsUnicode(true) && IsFixedLength(false) && HasMaxLength(x > 2000)	NCLOB

string	System.String	IsUnicode(false) && IsFixedLength(false) && HasMaxLength(x <= 4000)	VARCHAR2(size)
string	System.String	IsUnicode(true) && IsFixedLength(false) && HasMaxLength(x <= 2000)	NVARCHAR2(size)
string	System.String	IsUnicode(false) && IsFixedLength(true) && HasMaxLength(x < 2000)	CHAR(size)
string	System.String	IsUnicode(true) && IsFixedLength(true) && HasMaxLength(x < 1000)	NCHAR(size)

*- Corresponding data annotations can also be used instead of the specified fluent APIs.

Scaffolding/Reverse Engineering Data Type Mapping

By convention, ODP.NET EF Core maps an appropriate .NET data type based on the Oracle Database data type and its characteristics. This table shows the default mappings.

Table: ODP.NET Entity Framework Core Reverse Engineering Data Type Default Mappings

Oracle Database Type	.NET Type Alias	.NET Data Type
NUMBER(1)	bool	System.Boolean
NUMBER(2) to NUMBER(4)	byte	System.Byte
NUMBER(5)	short/int16	System.Int16
NUMBER(6) to NUMBER(10)	int/int32	System.Int32
NUMBER(11) to NUMBER(19)	long/int64	System.Int64
NUMBER(>19)	decimal	System.Decimal
NUMBER(p,s)	decimal	System.Decimal
NUMBER	decimal	System.Decimal
BINARY_FLOAT	float	System.Float
BINARY_DOUBLE	double	System.Double
TIMESTAMP	DateTime	System.DateTime
DATE	Date	System.Date
INTERVAL DAY TO SECOND	TimeSpan	System.TimeSpan
INTERVAL YEAR TO MONTH	string	System.String
VARCHAR2	string	System.String
NVARCHAR2	string	System.String
CHAR	string	System.String
NCHAR	string	System.String
CLOB	string	System.String
NCLOB	string	System.String
RAW	byte[]	System.Byte[]

BLOB	byte[]	System.Byte[]
XMLTYPE	string	System.String
ROWID	string	System.String
UROWID	string	System.String
LONG	string	System.String
BFILE	byte[]	System.Byte[]
LONG RAW	byte[]	System.Byte[]

Identifier Name Length and Uniqueness

Oracle Database prior to version 12.2 limit identifier names, such as table names, column names, and primary key names, to 30 characters. Oracle Database 12.2 and higher have a default limit of 128 characters. In Entity Framework Core Code First, these identifier lengths should be constrained to prevent creating identifier names longer than what the Oracle Database version supports. Attempting to create an identifier longer than the database can support generally results in an "ORA-00972: IDENTIFIER IS TOO LONG" error.

Use RelationalModelAnnotations.MaxIdentifierLength property to set the maximum identifier length the target database version can handle. For example, if Oracle Database 11cR2 is used, it needs to be set to 30 (or less). Once set, Entity Framework Core will automatically truncate identifier names that are too long to the specified length.

```
// C# Sample Code: Setting maximum identifier length to 30 characters; By default, it's set to 128.
modelBuilder.Model.Relational().MaxIdentifierLength = 30;
```

Entity Framework Core 2.x has a known issue in which identifier names longer than the maximum identifier length are merely truncated, but not made unique. Also, the MaxIdentifierLength has no impact on controlling the length of table names that are created based on the entity class names. These issues will be resolved in a future EF Core release by Microsoft.

In the meantime, either rename the class names, property names, etc. to work around this issue or use the ToTable()/HasColumnName() fluent APIs or their related annotations to specify a shorter and/or unique names for the tables/columns that are to be created in the Oracle database.

If the identifiers use multi-byte characters, the MaxIdentifierLength may need to be set with character expansion ratio in mind to assure that all identifiers can be created in the Oracle database. For example, if the Oracle database character set is UTF8, a single character may require up to 4 bytes. Thus, to guarantee that all identifiers can be created in an Oracle database that does not support long identifiers, the MaxIdentifierLength should be set to 7 characters (i.e. 30 characters divided by 4).

Limitations and Known Issues

Code First

- The HasIndex() fluent API cannot be invoked on an entity property that will result in a primary key in the Oracle database. Oracle Database does not support the creation of indexes for primary keys, since an index is implicitly created for all primary keys.
- The 11.2 Oracle databases do not support default expression to reference any PL/SQL functions nor any pseudocolumns such as *'<sequence>*.NEXTVAL'. As such, HasDefaultValue() and HasDefaultValueSql() fluent APIs cannot be used in conjunction with 'sequence.NETVAL' as the default value, for example, if the Oracle database is 11.2. However, the application can use the UseOracleIdentityColumn() extension method to have the column be populated with server generated values, even if the Oracle database is 11.2. Please read about UseOracleIdentityColumn() for more details.

Reverse Engineering a Database

The following types are not support for reverse engineering a database:
INTERVAL YEAR TO MONTH
INTERVAL DAY TO SECOND
TIMESTAMP WITH LOCAL TIME ZONE
XMLTYPE

Migrations

The following types are not supported for code migrations
DateTimeOffset
TimeSpan
UInt16
UInt32
UInt64

Sequences

- A sequence cannot be restarted.
- Extension methods related to SequenceHiLo is not supported.

Computed Columns

- Literal values used for computed columns must be encapsulated by two single-quotes. In the example below, the literal string is the comma. It needs to be surrounded by two single-quotes as shown below.

```
// C# - computed columns code sample
modelBuilder.Entity<Blog>()
.Property(b => b.BlogOwner)
.HasComputedColumnSql("\"LastName\" || '','' || \"FirstName\"");
```

This documentation is in pre-General Availability status and is intended for demonstration and preliminary use only. It may not be specific to the hardware on which you are using the software. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to this documentation and will not be responsible for any loss, costs, or damages incurred due to the use of this documentation.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.