.NET Connector

The .NET (C #) Connector library that supports some features of the ADO.NET driver is provided.

The library location is `$MACHBASE_HOME/lib/` provided as a DLL type. It provides different DLLs depending on the .NET version.

- > .NET Framework 4.0: `machNetConnector.dll`
- > .NET Core 2.0: `machNetConnectorCore.dll`

GitHub

The source code can be found on github site.

https://github.com/MACHBASE/NetConnector

Index

Class

Features not listed below may not be implemented yet or may not work correctly.

If you call a method or field that is not a named instance, it generates NotImplementedException or a NotSupportedException.

MachConnection : DbConnection

This class is responsible for linking with Machbase. Because it inherits IDisposable like DbConnection, it supports disassociation through Dispose () or automatic disposition of object using using () statement.

Constructor	Description
MachConnection(string aConnectionString)	Creates a MachConnection with a Connection String as input.

Method	Description
Open()	Attempts to connect to the connection string.
Close()	Closes the connection when connecting.
BeginDbTransaction (IsolationLevel isolationLevel)	(Not yet implemented) MACHBASE does not support this object because there is no special transaction.
CreateDbCommand()	(Not yet implemented) Explicitly induces MachCommands to be created
ChangeDatabase(string databaseName)	(Not yet implemented) MACHBASE has no DATABASE classification.

- Class
    - MachConnection : DbConnection
    - MachCommand : DbCommand
    - MachDataReader : DbDataReader
    - MachParameterCollection : DbParameterCollection
    - MachParameter : DbParameter
    - MachException :

DbException
    - MachAppendWriter
    - ErrorDelegateFuncType
    - MachAppendException : MachException
    - MachTransaction
- Sample Code
    - Connection
    - Performing Queries
    - Performing Select
    - Parameter Binding
    - APPEND

Field	Description
State	Represents a System.Data.ConnectionState value.
StatusString	Indicates the state to be performed by the connected MachCommand. This is used internally to decorate the Error Message and it is not appropriate to check the status of the query with this value because it indicates the state in which the operation started.
Database	(Not yet implemented)
DataSource	(Not yet implemented)
ServerVersion	(Not yet implemented)

Connection String

Each item is separated by a semicolon (;).

Many of the keywords in the same section have the same meaning.

Keyword	Description	Example	Default Value

DSN
SERVER
HOST
	Hostname	DSN=localhost
		SERVER=192.168.0.1	

PORT PORT_NO	Port No.	PORT=5656	5656
USERID USERNAME USER UID	User ID	USER=SYS	SYS
PASSWORD PWD	User password	PWD=manager	

CONNECT_TIMEOUT ConnectionTimeout connectTimeout	Maximum connection time	CONNECT_TIMEOUT	60 second
COMMAND_TIMEOUT commandTimeout	Maximum time to perform each command	COMMAND_TIMEOUT	60 second

As an example, we can prepare the following string.

```
String sConnString = String.Format("DSN={0};PORT_NO={1};UID=;PWD=MANAGER;CONNECT_TIMEOUT=10000;
COMMAND_TIMEOUT=50000", SERVER_HOST, SERVER_PORT);
```

MachCommand : DbCommand

A class that performs **SQL commands or APPEND** using MachConnection .
Since it inherits IDisposable like DbCommand, it supports object disposal through Dispose () or automatic disposal of object using using () statement.

Constructor	Description
MachCommand(string aQueryString, MachConnection)	Creates by typing the query to be executed along with the MachConnection object to be connected.
MachCommand(MachConnection)	Creates a MachConnection object to connect to. Use only if there is no query to perform (eg APPEND).

Method	Description
void CreateParameter() / void CreateDbParameter()	Creates a new MachParameter.
void Cancel()	(Not yet implemented)
void Prepare()	(Not yet implemented)
MachAppendWriter AppendOpen(aTableName, aErrorCheckCount = 0, MachAppendOption = None)	Starts APPEND. Returns a MachAppendWriter object. • aTableName: Target table name • aErrorCheckCount: Each time the cumulative number of records entered by APPEND-DATA matches, it is checked whether it is sent to the server or not. In other words, you are setting the automatic APPEND-FLUSH point. • MachAppendOption: Currently only one option is provided. ◦ MachAppendOption.None: No options are attached. ◦ MachAppendOption. MicroSecTruncated: When inputting the value of a DateTime object, enter the value expressed only up to microsecond. (The Ticks value of a DateTime object is expressed up to 100 nanoseconds.)

void AppendData(MachAppendWriter aWriter, List<object> aDataList)	Through the MachAppendWriter object, it takes a list containing the data and enters it into the database. • In the order of the data in the List, each datatype must match the datatype of the column represented in the table. • If the data in the List is insufficient or overflows, an error occurs. When representing a time value with a ulong object, simply do not enter the Tick value of the DateTime object. In that value, you must enter a value that excludes the DateTime Tick value that represents 1970-01-01 .
void AppendDataWithTime(MachAppendWriter aWriter, List<object> aDataList, DateTime aArrivalTime)	Method that explicitly puts an _arrival_time value into a DateTime object in AppendData ().
void AppendDataWithTime(MachAppendWriter aWriter, List<object> aDataList, ulong aArrivalTimeLong)	Method that can explicitly put _arrival_time value into a ulong object in AppendData (). Refer to AppendData () above for problems that may occur when typing a ulong value as an _arrival_time value .
void AppendFlush(MachAppendWriter aWriter)	The data entered by AppendData () is immediately sent to the server to force data insert. The more frequently the call is made, the lower the data loss rate due to the system error and the faster the error check, although the performance is lowered. The less frequently the call is made, the more likely the data loss will occur and the error checking will be delayed, but the performance will increase significantly.
void AppendClose(MachAppendWriter aWriter)	Closes APPEND. Internally, after calling AppendFlush (), the actual protocol is internally finished.
int ExecuteNonQuery()	Performs the input query. Returns the number of records affected by the query. It is usually used when performing queries except SELECT.
object ExecuteScalar()	Performs the input query. Returns the first value of the query targetlist as an object. It is usually used when you want to perform a SELECT query, especially a SELECT (Scalar Query) with only one result, and get the result without a DbDataReader
DbDataReader ExecuteDbDataReader (CommandBehavior aBehavior)	Executes the input query, generates a DbDataReader that can read the result of the query, and returns it.

Field	Description
Connection / DbConnection	Connected MachConnection.
ParameterCollection / DbParameterCollection	The MachParameterCollection to use for the Binding purpose.
CommandText	Query string.
CommandTimeout	The amount of time it takes to perform a particular task, waiting for a response from the server. It follows the values set in MachConnection, where you can only reference values.
FetchSize	The number of records to fetch from the server at one time . The default value is 3000.
IsAppendOpened	Determines if Append is already open when APPEND is at work
CommandType	(Not yet implemented)
DesignTimeVisible	(Not yet implemented)
UpdatedRowSource	(Not yet implemented)

MachDataReader : DbDataReader

This is a class that reads fetch results. Only objects created with MachCommand.ExecuteDbDataReader () that can not be explicitly created are available.

Method	Description
string GetName(int ordinal)	Returns the ordinal column name.
string GetDataTypeName(int ordinal)	Returns the datatype name of the ordinal column.
Type GetFieldType(int ordinal)	Returns the datatype of the ordinal column.
int GetOrdinal(string name)	Returns the index at which the column name is located.
object GetValue(int ordinal)	Returns the ordinal value of the current record.
bool IsDBNull(int ordinal)	Returns whether the ordinal value of the current record is NULL.
int GetValues(object[] values)	Sets all the values of the current record and returns the number.
xxxx GetXXXX(int ordinal)	Returns the ordinal column value according to the datatype (XXXX). • Boolean • Byte • Char • Int16 / 32/64 • DateTime • String • Decimal • Double • Float
bool Read()	Reads the next record. Returns False if the result does not exist.
DataTable GetSchemaTable()	(Not supported)
bool NextResult()	(Not supported)

Field	Description
FetchSize	The number of records to fetch from the server at one time. The default is 3000, which can not be modified here.
FieldCount	Number of result columns.
this[int ordinal]	Equivalent to object GetValue (int ordinal).
this[string name]	Equivalent to object GetValue(GetOrdinal(name).
HasRows	Indicates whether the result is present.
RecordsAffected	Unlike MachCommand, here, it represents Fetch Count.

MachParameterCollection : DbParameterCollection

This is a class that binds parameters needed by MachCommand.

If you do this after binding, the values are done together.

Since the concept of Prepared Statement is not implemented, execution performance after Binding is the same as the performance performed first.

Method	Description
MachParameter Add(string parameterName, DbType dbType)	Adds the MachParameter, specifying the parameter name and type. Returns the added MachParameter object.
int Add(object value)	Adds a value. Returns the index added.

void AddRange(Array values)	Adds an array of simple values.
MachParameter AddWithValue(string parameterName, object value)	Adds the parameter name and its value. Returns the added MachParameter object.
bool Contains(object value)	Determines whether or not the corresponding value is added.
bool Contains(string value)	Determines whether or not the corresponding parameter name is added.
void Clear()	Deletes all parameters.
int IndexOf(object value)	Returns the index of the corresponding value.
int IndexOf(string parameterName)	Returns the index of the corresponding parameter name.
void Insert(int index, object value)	Adds the value to a specific index.
void Remove(object value)	Deletes the parameter including the value.
void RemoveAt(int index)	Deletes the parameter located at the index.
void RemoveAt(string parameterName)	Deletes the parameter with that name.

Field	Description
Count	Number of parameters
this[int index]	Indicates the MachParameter at index.
this[string name]	Indicates the MachParameter of the order in which the parameter names match.

MachParameter : DbParameter

This is a class that contains the information that binds the necessary parameters to each MachCommand.

No special methods are supported.

Field	Description
ParameterName	Parameter name
Value	Value
Size	Value size
Direction	ParameterDirection (Input / Output / InputOutput / ReturnValue) The default value is Input.
DbType	DB Type
MachDbType	MACHBASE DB Type May differ from DB Type.
IsNullable	Whether nullable
HasSetDbType	Whether DB Type is specified

MachException : DbException

This is a class that displays errors that appear in Machbase.

An error message is set, and all error messages can be found in *MachErrorMsg* .

Field	Description
int MachErrorCode	Error code provided by MACHBASE

MachAppendWriter

APPEND is supported as a separate class using MachCommand.

This is a class to support MACHBASE Append Protocol, not ADO.NET standard.
It is created with MachCommand's AppendOpen () without a separate constructor.

Method	Description
void SetErrorDelegator(ErrorDelegateFuncType aFunc)	Specifies the ErrorDelegateFunc to call when an error occurs.

Field	Description
SuccessCount	Number of successful records. Is set after AppendClose ().
FailureCount	The number of records that failed input. Set after AppendClose ().
Option	MachAppendOption received input during AppendOpen()

ErrorDelegateFuncType

```
public delegate void ErrorDelegateFuncType(MachAppendException e);
```

In MachAppendWriter, you can specify a function to detect errors occurring on the MACHBASE server side during APPEND.

In .NET, this function type is specified as a Delegator Function.

MachAppendException : MachException

Same as MachException, except that:

- An error message is received from the server side.
- A data buffer in which an error has occurred can be obtained. (comma-separated) can be used to process and re-append or record data.

The exception is only available within the ErrorDelegateFunc.

Method	Description
GetRowBuffer()	A data buffer in which an error has occurred can be obtained.

MachTransaction

Not supported.

Sample Code

Connection

You can create a MachConnection and use Open () - Close ().

```
String sConnString = String.Format("DSN={0};PORT_NO={1};UID=;PWD=MANAGER;", SERVER_HOST, SERVER_PORT);
MachConnection sConn = new MachConnection(sConnString);
sConn.Open();
//... do something
sConn.Close();
```

If you use the using statement, you do not need to call Close (), which is a connection closing task.

```
String sConnString = String.Format("DSN={0};PORT_NO={1};UID=;PWD=MANAGER;", SERVER_HOST, SERVER_PORT);
using (MachConnection sConn = new MachConnection(sConnString))
{
    sConn.Open();
    //... do something
} // you don't need to call sConn.Close();
```

Performing Queries

Create a MachCommand and perform the query.

```
String sConnString = String.Format("DSN={0};PORT_NO={1};UID=;PWD=MANAGER;", SERVER_HOST, SERVER_PORT);
using (MachConnection sConn = new MachConnection(sConnString))
{
    String sQueryString = "CREATE TABLE tab1 ( col1 INTEGER, col2 VARCHAR(20) )";
    MachCommand sCommand = new MachCommand(sQueryString , sConn)
    try
    {
        sCommand.ExecuteNonQuery();
    }
    catch (MachException me)
    {
        throw me;
    }
}
```

Again, using the using statement, MachCommand release can be done immediately.

```
String sConnString = String.Format("DSN={0};PORT_NO={1};UID=;PWD=MANAGER;", SERVER_HOST, SERVER_PORT);
using (MachConnection sConn = new MachConnection(sConnString))
{
    String sQueryString = "CREATE TABLE tab1 ( col1 INTEGER, col2 VARCHAR(20) )";
    using(MachCommand sCommand = new MachCommand(sQueryString , sConn))
    {
        try
        {
            sCommand.ExecuteNonQuery();
        }
        catch (MachException me)
        {
            throw me;
        }
    }
}
```

Performing Select

You can get a MachDataReader by executing a MachCommand with a SELECT query.

You can fetch the records one by one through the MachDataReader.

```
String sConnString = String.Format("DSN={0};PORT_NO={1};UID=;PWD=MANAGER;", SERVER_HOST, SERVER_PORT);
using (MachConnection sConn = new MachConnection(sConnString))
{
    String sQueryString = "SELECT * FROM tab1;";
    using(MachCommand sCommand = new MachCommand(sQueryString , sConn))
    {
        try
        {
            MachDataReader sDataReader = sCommand.ExecuteReader();
            while (sDataReader.Read())
            {
                for (int i = 0; i < sDataReader.FieldCount; i++)
                {
                    Console.WriteLine(String.Format("{0} : {1}",
                                                    sDataReader.GetName(i),
                                                    sDataReader.GetValue(i)));
                }
            }
        }
        catch (MachException me)
        {
            throw me;
        }
    }
}
```

Parameter Binding

You can create a MachParameterCollection and then link it to a MachCommand.

```
String sConnString = String.Format("DSN={0};PORT_NO={1};UID=;PWD=MANAGER;", SERVER_HOST, SERVER_PORT);
using (MachConnection sConn = new MachConnection(sConnString))
{
    string sSelectQuery = @"SELECT *
        FROM tab2
        WHERE CreatedDateTime < @CurrentTime
        AND CreatedDateTime >= @PastTime";

    using (MachCommand sCommand = new MachCommand(sSelectQuery, sConn))
    {
        DateTime sCurrtime = DateTime.Now;
        DateTime sPastTime = sCurrtime.AddMinutes(-1);

        try
        {
            sCommand.ParameterCollection.Add(new MachParameter { ParameterName = "@CurrentTime", Value =
sCurrtime });
            sCommand.ParameterCollection.Add(new MachParameter { ParameterName = "@PastTime", Value =
sPastTime });

            MachDataReader sDataReader = sCommand.ExecuteReader();

            while (sDataReader.Read())
            {
                for (int i = 0; i < sDataReader.FieldCount; i++)
                {
                    Console.WriteLine(String.Format("{0} : {1}",
                                                    sDataReader.GetName(i),
                                                    sDataReader.GetValue(i)));
                }
            }
        }
        catch (MachException me)
        {
            throw me;
        }
    }
}
```

APPEND

When you run AppendOpen () on a MachCommand, you get a MachAppendWriter object.

Using this object and MachCommand, you can get a list of one input record and perform an AppendData ().
AppendFlush () will reflect the input of all records, and AppendClose () will end the entire Append process.

```
String sConnString = String.Format("DSN={0};PORT_NO={1};UID=;PWD=MANAGER;", SERVER_HOST, SERVER_PORT);
using (MachConnection sConn = new MachConnection(sConnString))
{
    using (MachCommand sAppendCommand = new MachCommand(sConn))
    {
        MachAppendWriter sWriter = sAppendCommand.AppendOpen("tab2");
        sWriter.SetErrorDelegator(AppendErrorDelegator);

        var sList = new List<object>();
        for (int i = 1; i <= 100000; i++)
        {
            sList.Add(i);
            sList.Add(String.Format("NAME_{0}", i % 100));

            sAppendCommand.AppendData(sWriter, sList);

            sList.Clear();

            if (i % 1000 == 0)
            {
                sAppendCommand.AppendFlush();
            }
        }

        sAppendCommand.AppendClose(sWriter);
        Console.WriteLine(String.Format("Success Count : {0}", sWriter.SuccessCount));
        Console.WriteLine(String.Format("Failure Count : {0}", sWriter.FailureCount));
    }
}
```

```
        private static void AppendErrorDelegator(MachAppendException e)
        {
            Console.WriteLine("{0}", e.Message);
            Console.WriteLine("{0}", e.GetRowBuffer());
        }
```