Oracle® TimesTen In-Memory Database

Open Source Languages Support Guide

Release 22.1
F35401-02
June 2024

Oracle TimesTen In-Memory Database Open Source Languages Support Guide, Release 22.1

F35401-02

Contents

About This Content

This document provides usage and reference information for TimesTen support of open source languages.

Audience

This guide is for application developers who access TimesTen through supported open source languages.

In addition to familiarity with the particular programming interface you use, you should be familiar with TimesTen, SQL (Structured Query Language), database operations, and ODBC.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Resources

TimesTen documentation is available on the TimesTen documentation website.

Oracle Database documentation is also available on the Oracle documentation website. This may be especially useful for Oracle Database features that TimesTen supports but does not attempt to fully document, such as OCI and Pro*C/C++.

Conventions

The following text conventions are used in this document.

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New

This section summarizes new features and functionality of TimesTen Release 22.1.

New Features in Release 22.1.1.25.0

- In this release, TimesTen supports the following:
  - Python through python-oracledb 2.2.0
  - Node.js through node-oracledb 6.4.0

New Features in Release 22.1.1.1.0

- In this release, TimesTen supports the following:
  - ODPI-C 4.2
  - Python through cx_Oracle 8.2.1
  - Node.js through node-oracledb 5.2.0

1 Getting Started

Oracle TimesTen In-Memory Database supports Python and Node.js open source languages through the Oracle Database Programming Interface for C (ODPI-C). This support applies to both TimesTen Classic and TimesTen Scaleout.

- ODPI-C and Language-Specific Packages
- Platform Support for Open Source Languages
- Requirements for Open Source Languages
- Restrictions for Open Source Languages

Note:

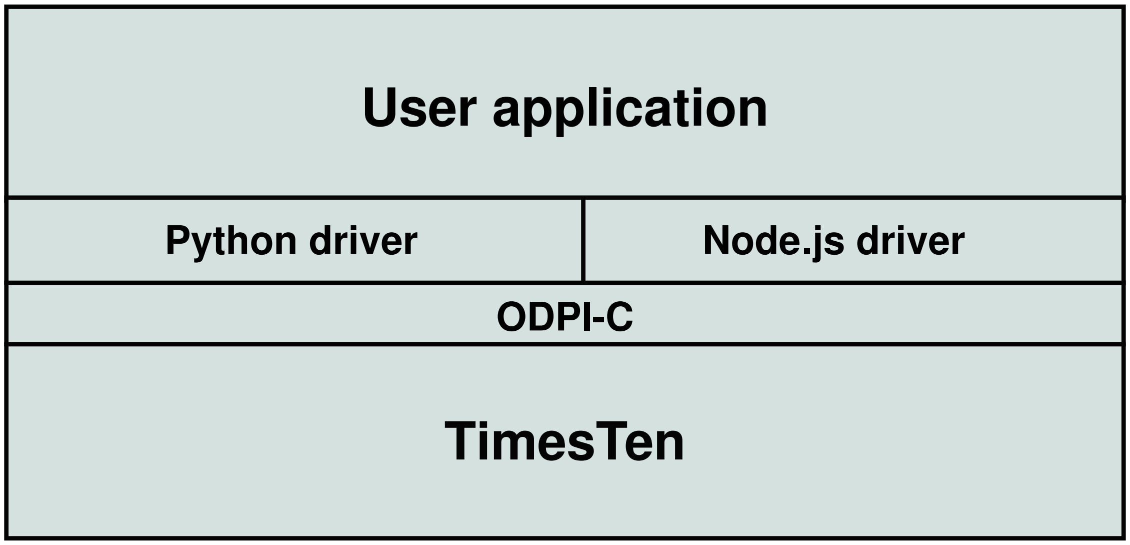
Sample programs for the open source languages are available on GitHub. For an overview of what is available, refer to `README.md` at oracle-timesten-samples in GitHub.

ODPI-C and Language-Specific Packages

TimesTen support of open source languages is through the Oracle Database Programming Interface for C (ODPI-C). ODPI-C is an open source library from Oracle Corporation designed to simplify access to Oracle databases from a variety of programming languages. See ODPI-C. For each language, there is an open source package, or driver, available through GitHub and maintained by Oracle. The open source languages supported by TimesTen are Python and Node.js. ODPI-C is included when you download the driver for each of these languages.

ODPI-C is a layer on top of the Oracle Call Interface (referred to in this document as OCI, but not to be confused with Oracle Cloud Infrastructure). Figure 1-1 shows the architecture.

Figure 1-1 TimesTen Architecture for Open Source Languages



Platform Support for Open Source Languages

Platform support for open source languages is of course limited to the platforms, or a subset of the platforms, supported by TimesTen, and may vary between languages.

TimesTen supports client/server access from Linux or macOS or direct access on Linux for both Python and Node.js.

For the latest information about platform support for each language, refer to the applicable `README.md` file at TimesTen Python Samples or TimesTen Node.js Samples in GitHub.

For more information about TimesTen platform support, including specific versions supported, refer to Platforms and Configurations in *Oracle TimesTen In-Memory Database Release Notes*.

Requirements for Open Source Languages

Using open source languages through ODPI-C for access to TimesTen databases requires use of the special OCI client library that is included with the Oracle Instant Client shipped with TimesTen software.

Because you must use the Instant Client that is supplied with TimesTen, your library path must be set so that the TimesTen Instant Client directory (*`installation_dir`*`/ttoracle_home/instantclient`) precedes the path to any Oracle Database libraries. The path is set appropriately when you use the TimesTen `ttenv` script (`ttenv.sh` or `ttenv.csh`), described in Task 1: Install TimesTen.

Restrictions for Open Source Languages

Because ODPI-C is a layer on top of OCI, TimesTen restrictions for OCI also apply to ODPI-C.

These restrictions are documented in TimesTen Restrictions and Limitations in *Oracle TimesTen In-Memory Database C Developer's Guide*.

In addition, these features are not supported:

- Application continuity
- Continuous query notification
- Call timeouts
- Session tagging
- Database management, including startup and shutdown

2 Prepare TimesTen for Python or Node.js

There are several tasks you need to perform before you are able to connect to a TimesTen database through an open source language application.

This tasks include:

- Task 1: Install TimesTen
- Task 2: Set the TimesTen Environment
- Task 3: Define and Manage the TimesTen Database
- Task 4: Configure Connections for TimesTen

Task 1: Install TimesTen

You can download TimesTen with a single ZIP file distribution.

If you do not already have TimesTen, you can download it from TimesTen downloads.

> **Note:**
>
> Full support for open source languages requires a TimesTen release of 18.1.4.1.0 or higher.

There is a single ZIP file distribution for both TimesTen Classic and TimesTen Scaleout, but setup instructions differ from there:

- For prerequisites, environment setup, and installation for TimesTen Classic, refer to Overview of the Installation Process in TimesTen Classic and the applicable installation chapter in *Oracle TimesTen In-Memory Database Installation, Migration, and Upgrade Guide*.

  The installation chapters also describe how to create a TimesTen instance.
- For prerequisites, environment setup, and installation for TimesTen Scaleout, refer to Prerequisites and Installation of TimesTen Scaleout in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

  Setting Up a Grid in that document includes information about creating the initial management instance.

Task 2: Set the TimesTen Environment

After you have installed TimesTen and created a TimesTen instance, use the appropriate `ttenv` script (`ttenv.sh` or `ttenv.csh`) under `*timesten_home*/bin` to finish setting up the environment.

Task 3: Define and Manage the TimesTen Database

You can define and manage databases in either TimesTen Classic or TimesTen Scaleout.

For defining and managing a database for TimesTen Classic, refer to Managing TimesTen Databases in *Oracle TimesTen In-Memory Database Operations Guide*.

For defining and managing a grid and database for TimesTen Scaleout, refer to Setting Up the Membership Service, Setting Up a Grid, and Managing a Database in *Oracle TimesTen In-Memory Database Scaleout User's Guide*.

Task 4: Configure Connections for TimesTen

Because TimesTen support of open source languages goes through OCI, you can connect to TimesTen from Python or Node.js using either the `tnsnames` or the Easy Connect naming method. See Connecting to a TimesTen Database from OCI in *Oracle TimesTen In-Memory Database C Developer's Guide*.
You may have a `tnsnames.ora` entry such as the following, for example, for connecting to database `sampledb`. Then you would reference the TNS name, `sampledbconn`, for the connection string in your code.

```
sampledbconn =
  (DESCRIPTION =
    (CONNECT_DATA =
      (SERVICE_NAME = sampledb)
      (SERVER = timesten_direct)))
```

Or, to avoid using `tnsnames.ora`, you can use an Easy Connect string, such as `myhost/sampledb:timesten_direct`, directly in your code for the connection string.

Note:

Use `timesten_direct` for a direct connection to TimesTen or `timesten_client` for a client/server connection.

See Task 3: Configure Connections to TimesTen in Python and Task 3: Configure Connections to TimesTen in Node.js for examples using TNS names and Easy Connect strings in your code.

You will need a TimesTen database user in order to connect to a TimesTen database. For information about creating users in TimesTen, see CREATE USER in *Oracle TimesTen In-Memory Database SQL Reference*.

The following example creates a TimesTen internal user `appuser` and grants `CREATE SESSION` and `CREATE TABLE` privileges so `appuser` can create a user session in connecting to the database and then create a database table.

```
CREATE USER appuser IDENTIFIED BY password;

GRANT CREATE SESSION TO appuser;
GRANT CREATE TABLE TO appuser;
```

3

Support for Python

There are specific tasks you need to perform to access a TimesTen database using Python, such as install the python-oracledb driver.

- About Python
- Set Up Python
- Additional Information for Python

About Python

Python is an interpreted, high-level, general purpose language that includes support for object-oriented programming and a robust standard library. Simple syntax rules make Python code easy to read and support.

Many platforms come with Python installed. Enter `python` at the command prompt to confirm. If it is installed, its response will include the version number.

You can find information about Python and download it from the Python website.

The Python extension module for access to Oracle or TimesTen databases from Python applications is the python-oracledb driver. This driver conforms to the Python Database API v2.0 specification. There is general information about the driver at Python Driver for Oracle Database.

Note:

The python-oracledb driver is the renamed, major version successor to cx_Oracle 8.3. For upgrade information, see Upgrading from cx_Oracle 8.3 to python-oracledb.

For information about versions of Python and the python-oracledb driver supported by TimesTen, refer to the `README.md` file at TimesTen Python Samples.

Set Up Python

You need to install the python-oracledb driver to successfully configure a connection to a TimesTen database in Python.

- Task 1: Install the python-oracledb Driver
- Task 2: Initialize the python-oracledb Driver
- Task 3: Configure Connections to TimesTen in Python

Task 1: Install the python-oracledb Driver

There are several approaches for installing the python-oracledb driver. Choose the approach that is most suitable for your setup and environment.

- Install the python-oracledb driver from PyPI. See Quick Start python-oracledb Installation.
- Install the python-oracledb driver from source code. You can either perform the installation using GitHub, opensource.oracle.com, or PyPI. See Installing from Source Code.
- Install the python-oracledb driver without Internet access. See Installing python-oracledb without Internet Access.

Information for all scenarios is available at Installing python-oracledb.

Task 2: Initialize the python-oracledb Driver

By default, the python-oracledb driver runs in Thin mode which connects directly to Oracle Database. However, to connect to a TimesTen database, the driver needs to be in Thick mode so it uses the Oracle Instant Client library included with TimesTen.
To enable Thick mode for the python-oracledb driver, call the `oracledb.init_oracle_client()` function in your application before creating any connection. See Enabling python-oracledb Thick mode.

This calls the `oracledb.init_oracle_client()` function without a `lib_dir` parameter to ensure it uses the `PATH` environment variable set by `ttenv` script. See Task 2: Set the TimesTen Environment.

```
import oracledb
oracledb.init_oracle_client()
```

Task 3: Configure Connections to TimesTen in Python

You can use a TNS name or an Easy Connect mechanism to connect to your database. See Task 4: Configure Connections for TimesTen.
This example uses a TNS name for a Python connection string:

```
connection = oracledb.connect(user="appuser", password="password",
dsn="sampledbconn")
```

This example uses an Easy Connect string that specifies a direct connection to TimesTen:

```
connection = oracledb.connect(user="appuser", password="password",
dsn="myhost/sampledb:timesten_direct")
```

Additional Information for Python

This section covers these topics:

- Type Mappings for Python Applications
- Failure Modes for Python
- Python Sample: Connect to TimesTen and Execute SQL

- Python Sample Programs for Download

Type Mappings for Python Applications

There are mappings between python-oracledb types (as available through the language extension module) and TimesTen SQL types.

Table 3-1 documents mappings between python-oracledb types (as available through the language extension module) and TimesTen SQL types.

Note:

Additional TimesTen SQL types can be mapped to these python-oracledb types as a result of TimesTen implicit data type conversions. See Data Type Conversion in *Oracle TimesTen In-Memory Database SQL Reference*.

For the latest information about data type support in the current release, refer to the `README.md` file at TimesTen Python Samples.

Table 3-1 Type Mappings for python-oracledb

python-oracledb Type	TimesTen Type
`oracledb.DB_TYPE_BINARY_DOUBLE`	`BINARY_DOUBLE`
`oracledb.DB_TYPE_BINARY_FLOAT`	`BINARY_FLOAT`
`oracledb.DB_TYPE_BLOB`	`BLOB`
`oracledb.DB_TYPE_CHAR`	`CHAR`
`oracledb.DB_TYPE_CLOB`	`CLOB`
`oracledb.DB_TYPE_CURSOR`	`REF CURSOR`
`oracledb.DB_TYPE_DATE`	`DATE`
`oracledb.DB_TYPE_NCHAR`	`NCHAR` (In TimesTen, this type is UTF-16 only.)
`oracledb.DB_TYPE_NCLOB`	`NCLOB`
`oracledb.DB_TYPE_NUMBER`	`NUMBER`, `TT_BIGINT`, `TT_INTEGER`, `TT_SMALLINT`, `TT_TINYINT`
`oracledb.DB_TYPE_NVARCHAR`	`NVARCHAR2` (In TimesTen, this type is UTF-16 only.)
`oracledb.DB_TYPE_RAW`	`BINARY`, `VARBINARY`
`oracledb.DB_TYPE_ROWID`	`ROWID`
`oracledb.DB_TYPE_TIMESTAMP`	`TIMESTAMP`, `TT_TIMESTAMP`
`oracledb.DB_TYPE_VARCHAR`	`VARCHAR2`

Failure Modes for Python

There are several failure modes for Python.

Be aware of the following:

- Error messages may differ slightly from those produced by Oracle under the same conditions.
- For the Python method `Cursor.executemany()`, TimesTen ignores the `batcherrors` option and therefore supports only the default `batcherrors=false` setting. An error is always returned if any row in the batch encounters an error. (By contrast, Oracle Database supports `batcherrors=true`, in which case success is always returned if any row in the batch is successful.) Also, against a TimesTen database, if multiple errors are encountered in a batch, TimesTen makes no attempt to order the errors as they would be ordered if encountered against an Oracle database.

Python Sample: Connect to TimesTen and Execute SQL

This sample program does the following:

- Connects to a TimesTen database.
- Creates a table named `employees`.
- Inserts three rows into the table.
- Selects and displays the rows.
- Drops the table.
- Disconnects from the database.

```
import oracledb

def run():
  try:
    oracledb.init_oracle_client();
    connection = oracledb.connect(
        user='appuser',
        password='password',
        dsn='localhost/sampledb:timesten_direct')
    cursor = connection.cursor()
    cursor.execute('''
        CREATE TABLE employees(first_name VARCHAR2(20), last_name 
VARCHAR2(20))''')
    print('Table created')
    values = [['ROBERT', 'ROBERTSON'], ['ANDY', 'ANDREWS'], ['MICHAEL', 
'MICHAELSON']]
    cursor.executemany('INSERT INTO employees VALUES (:1, :2)', values)
    print('Inserted', len(values), 'employees into the table')
    cursor.execute('''
        SELECT first_name, last_name FROM employees''')
    for fname, lname in cursor:
      print('Selected employee:', fname, lname)
    cursor.execute('DROP TABLE employees')
    print('Table dropped')
    cursor.close()
    connection.close()
    print('Connection closed')

  except Exception as e:
    print('An error occurred', str(e))
```

```
run()
```

Running this sample program to connect to a TimesTen database should return the following output:

```
Table created
Inserted 3 employees into the table
Selected employee: ROBERT ROBERTSON
Selected employee: ANDY ANDREWS
Selected employee: MICHAEL MICHAELSON
Table dropped
Connection closed
```

See Task 4: Configure Connections for TimesTen for information on the Easy Connect method used to connect to the database in the sample program.

Python Sample Programs for Download

TimesTen provides sample programs for Python through the python-oracledb driver. These sample programs are available at TimesTen Python Samples.

4
Support for Node.js

There are specific tasks you need to perform to access a TimesTen database using Node.js, such as install the node-oracledb driver.

- About Node.js
- Setup for Node.js
- Additional Information for Node.js

About Node.js

Node.js is an open source JavaScript runtime environment you can use to build scalable network applications and tools, such as web servers. It includes modules to reduce the complexity of writing server applications by handling basic functions such as file system I/O, networking, data streams, and cryptography.

You can find information about Node.js and download it from the Node.js website.

The Node.js module for access to Oracle or TimesTen databases from Node.js applications is the node-oracledb driver. There is general information about the driver at Node.js Driver for Oracle Database.

For information about the versions of Node.js and the node-oracledb driver supported by TimesTen, refer to the `README.md` file at TimesTen Node.js Samples.

Setup for Node.js

You need to install the node-oracledb driver to successfully configure a connection to a TimesTen database in Node.js.

- Task 1: Install the node-oracledb Driver
- Task 2: Initialize the node-oracledb Driver
- Task 3: Configure Connections to TimesTen in Node.js

Task 1: Install the node-oracledb Driver

There are several approaches for installing the node-oracledb driver. Choose the approach that is most suitable for your setup and environment.

- Install the node-oracledb driver from the npm package manager. See Quick Start node-oracledb Installation.
- Install the node-oracledb driver from source code. You can either perform the installation using GitHub or opensource.oracle.com. See Installing node-oracledb from Source Code.
- Install the python-oracledb driver without Internet access. See Installing node-oracledb Without Internet Access.

Information for all scenarios is available at Installing node-oracledb.

Task 2: Initialize the node-oracledb Driver

By default, the node-oracledb driver runs in Thin mode which connects directly to Oracle Database. However, to connect to a TimesTen database, the driver needs to be in Thick mode so it uses the Oracle Instant Client library included with TimesTen.
To enable Thick mode for the node-oracledb driver, call the `oracledb.init_oracle_client()` synchronous function in your application before creating any connection. See Enabling node-oracledb Thick Mode.

This example calls the `oracledb.init_oracle_client()` function without a `libDir` attribute to ensure it uses the `PATH` environment variable set by `ttenv` script. See Task 2: Set the TimesTen Environment.

```
var oracledb = require('oracledb');
oracledb.initOracleClient();
```

Task 3: Configure Connections to TimesTen in Node.js

You can use a TNS name or an Easy Connect mechanism to connect to your database. See Task 4: Configure Connections for TimesTen.
This example uses a TNS name for a Node.js connection string:

```
function connect(cb) {
    oracledb.getConnection({
      user          :  "appuser",
      password      :  "password",
      connectString :  "sampledbconn"
    } ,
    cb);
}
```

This example uses an Easy Connect string that specifies a direct connection to TimesTen:

```
function connect(cb) {
    oracledb.getConnection({
      user          : "appuser",
      password      : "password",
      connectString : "myhost/sampledb:timesten_direct"
    } ,
    cb);
}
```

Additional Information for Node.js

This section covers these topics:

- Type Mappings for Node.js Applications
- Failure Modes for Node.js
- Node.js Sample: Connect to TimesTen and Execute SQL
- Node.js Sample Programs for Download

Type Mappings for Node.js Applications

There are mappings between node-oracledb types (as available through the language module) and TimesTen SQL types

Note:

Additional TimesTen SQL types can be mapped to these Oracle Database types and node-oracledb alias types as a result of TimesTen implicit data type conversions. See Data Type Conversion in *Oracle TimesTen In-Memory Database SQL Reference*.

For the latest information about data type support in the current release, refer to the `README.md` file at TimesTen Node.js Samples.

Table 4-1 Type Mappings for node-oracledb

Node.js Type	Oracle Database Type	node-oracledb Alias Type	TimesTen Type
`Number`	`oracledb.DB_TYPE_BINARY_DOUBLE`	N/A	`BINARY_DOUBLE`
`Number`	`oracledb.DB_TYPE_BINARY_FLOAT`	N/A	`BINARY_FLOAT`
`Lob`	`oracledb.DB_TYPE_BLOB`	`oracledb.BLOB`	`BLOB`
`String`	`oracledb.DB_TYPE_CHAR`	N/A	`CHAR`
`Lob`	`oracledb.DB_TYPE_CLOB`	`oracledb.CLOB`	`CLOB`
`ResultSet`	`oracledb.DB_TYPE_CURSOR`	`oracledb.CURSOR`	`REF CURSOR`
`Date`	`oracledb.DB_TYPE_DATE`	`oracledb.DATE`	`DATE`
`Number`	`oracledb.DB_TYPE_INTEGER`	N/A	`NUMBER`, `TT_BIGINT`, `TT_INTEGER`, `TT_SMALLINT`, `TT_TINYINT`
`String`	`oracledb.DB_TYPE_NCHAR`	N/A	`NCHAR` (In TimesTen, this type is UTF-16 only.)
`Lob`	`oracledb.DB_TYPE_NCLOB`	`oracledb.NCLOB`	`NCLOB`
`Number`	`oracledb.DB_TYPE_NUMBER`	`oracledb.NUMBER`	`NUMBER`, `TT_BIGINT`, `TT_INTEGER`, `TT_SMALLINT`, `TT_TINYINT`
`String`	`oracledb.DB_TYPE_NVARCHAR`	N/A	`NVARCHAR2` (In TimesTen, this type is UTF-16 only.)
`Buffer`	`oracledb.DB_TYPE_RAW`	`oracledb.BUFFER`	`BINARY`, `VARBINARY`
`String`	`oracledb.DB_TYPE_ROWID`	N/A	`ROWID`
`Date`	`oracledb.DB_TYPE_TIMESTAMP`	N/A	`TIMESTAMP`, `TT_TIMESTAMP`
`String`	`oracledb.DB_TYPE_VARCHAR`	`oracledb.STRING`	`VARCHAR`

Failure Modes for Node.js

There are several failure modes for Node.js.

Be aware of the following:

- Error messages may differ slightly from those produced by Oracle under the same conditions.
- For the Node.js method `connection.executeMany()`, TimesTen ignores the `batcherrors` option and therefore supports only the default `batcherrors=false` setting. An error is always returned if any row in the batch encounters an error. (By contrast, Oracle Database supports `batcherrors=true`, in which case success is always returned if any row in the batch is successful.) Also, against a TimesTen database, if multiple errors are encountered in a batch, TimesTen makes no attempt to order the errors as they would be ordered if encountered against an Oracle database.

Node.js Sample: Connect to TimesTen and Execute SQL

This sample program does the following:

- Connects to a TimesTen database.
- Creates a table named `employees`.
- Inserts three rows into the table.
- Selects and displays the rows.
- Drops the table.
- Disconnects from the database.

```
'use strict';
var oracledb = require('oracledb');

oracledb.initOracleClient();

async function run() {

  let connection;

  try {
    connection = await oracledb.getConnection({
      user: 'appuser',
      password: 'password',
      connectString: 'localhost/sampledb:timesten_direct'
    });

    let result;

    await connection.execute('CREATE TABLE employees(first_name VARCHAR2(20),
last_name VARCHAR2(20))');
    console.log('Table created');
    const values = [['ROBERT', 'ROBERTSON'], ['ANDY', 'ANDREWS'], ['MICHAEL',
'MICHAELSON']];
    await connection.executeMany('INSERT INTO employees VALUES(:1, :2)',
values);
```

```
    console.log('Inserted', values.length, 'employees into the table');
    result = await connection.execute('SELECT first_name, last_name FROM
employees');
    result.rows.forEach(function(row){
      console.log('Selected employee:', row[0], row[1]);
    });
    await connection.execute('DROP TABLE employees');
    console.log('Table dropped');
  }
  catch (err) {
    console.log(err);
  }
  finally {
    if (connection) {
      try {
        connection.close();
        console.log('Connection closed');
      }
      catch (err) {
        console.log(err);
      }
    }
  }
}

run();
```

Running this sample program to connect to a TimesTen database should result the following output:

```
Table created
Inserted 3 employees into the table
Selected employee: ROBERT ROBERTSON
Selected employee: ANDY ANDREWS
Selected employee: MICHAEL MICHAELSON
Table dropped
Connection closed
```

See Task 4: Configure Connections for TimesTen for information on the Easy Connect method used to connect to the database in the sample program.

Node.js Sample Programs for Download

TimesTen provides sample programs for Node.js through the node-oracledb driver. These sample programs are available at TimesTen Node.js Samples.