ACTIAN

How to use the high speed Btrieve 2 API from Node.js

Application Note

Contents

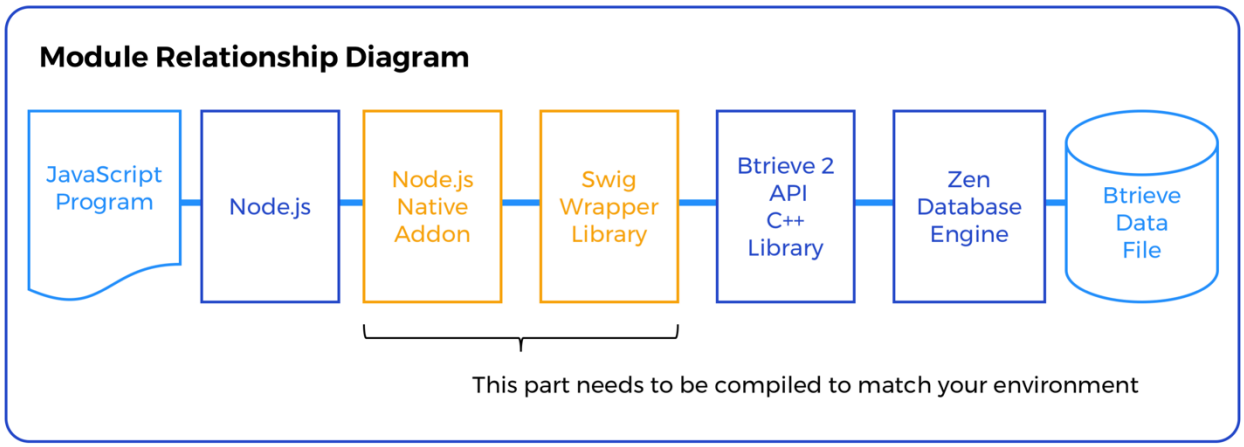
Introduction

The simplest way to use Actian Zen from Node.js is to access it using SQL via ODBC as described in the appendix.

However, to take full advantage of Actian Zen's competition beating speed, it is better to call Zen's native API the Btrieve2 API rather than use a standard SQL interface such as ODBC. By making use of the Btrieve2 API, you can obtain the fastest data access performance to Zen from Node.js.

The Btrieve2 API consists of a library for C/C++, but Actian provides SWIG (Simplified Wrapper and Interface Generator) definition files, making it possible to use the Btrieve2 API from other scripting languages or programming languages as well too. The diagram below illustrates how a JavaScript code can interact through Node.js and the Btrieve 2 API to interact with Zen.

To use the Btrieve2 API from Node.js, you first need to prepare a Btrieve2 API library interface for Node.js.



In this article, we will show you how to create a native Node.js add-on interface and then show its use in a simple sample Btrieve2 API program.

The instructions in this article were written and tested on CentOS 7.4 x64 and Ubuntu 18.04.4 LTS. If there is a difference in commands used on these two OSs, that will be indicated in the following text, if not please assume that the commands are the same for both OSs.

You can download and run the "bcreate_insert_read_sample2.js" Node.js JavaScript program to observe the Btrieve 2 API in action and learn how to prepare a JavaScript application to interact with Zen via the Btrieve 2 API.

※Note: bcreate_insert_read_sample2.js can be downloaded here.

This program performs the following operations: -

(1) It first generates a Btrieve data file capable of storing 10 bytes of data per record (the first 8 bytes of which are type AUTOTIMESTAMP, followed by a two-byte column of type UInt16)

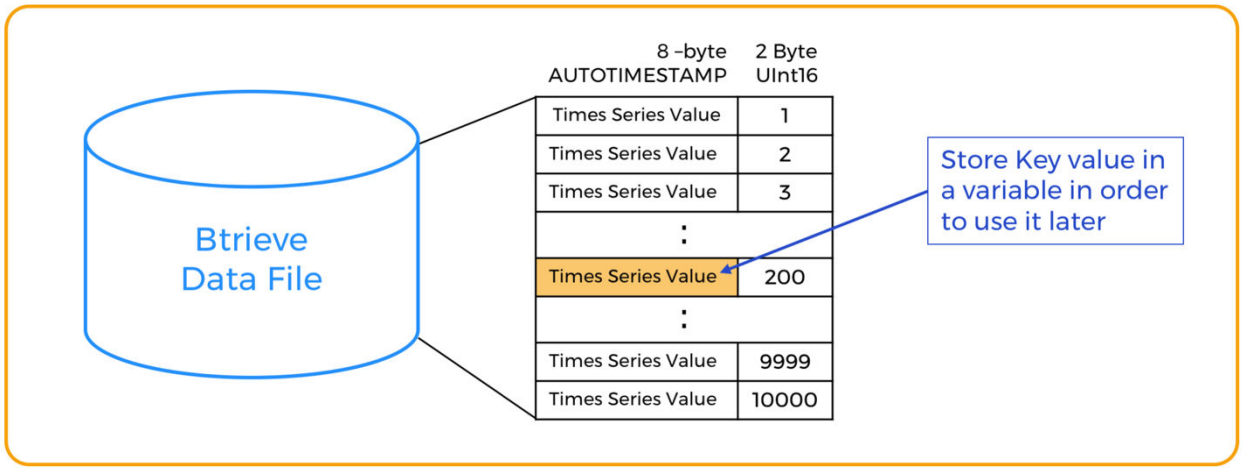
(2) The first 8-byte AUTOTIMESTAMP column is set as an index to the file

(3) The program inserts 10,000 records in a data file

(4) When doing this insertion, the program stores away for later use the 8-byte AUTOTIMESTAMP index for the 200th record

(5) The program then extracts the last written record from the data file and displays the value recorded in that record

(6) Finally, the program uses the timestamp stored during the loop for the 200th record to extract the value stored at that timestamp value



Sample Database Structure Diagram

1 About node-gyp

Although Node.js runs on a variety of different platforms, platform specific functionality such as the Btrieve 2 API require a native add-on interface to Node.JS. **node-gyp** is a tool for building such native add-ons. You can download node-gyp from https://github.com/nodejs/node-gyp.

※ **Note:** In order to run node-gyp, Python3, C++, and **make** are also required.

2

2 How to create the Node.js AddOn

The basic steps to create the Node.js Add-In are as follows: -

(i) Install **node-gyp** (and other necessary tools) on the system
(ii) Run "**node-gyp configure**" – this generates the add-on project's build files (**makefile** etc.)
(iii) Run "**node-gyp build**" to build the add-on

Once the add-on is created, you can build an environment in which to run Node.js scripts that use the Btrieve 2 API.

3

3 Building the Environment

First, let's make sure your OS is up to date. If you don't do this, you may not be able to install the various packages correctly. This can be done with the following commands:

CentOS:

```
$ sudo yum update
```

Ubuntu:

```
$ sudo apt update
$ sudo apt upgrade
```

※ Note: The command examples below use "test" as the account name and /home/test/ as the target subdirectory. Please replace /home/test/ in all of the following commands with appropriate personalized directory names.

Next, install the Actian Zen.

```
$ sudo su
# cd /usr/local
# tar -zxf /home/test/Zen-EnterpriseServer-jajp-linux-x86_64-
  14.11.014.000.tar.gz
# cd actianzen/etc
# ./preinstall.sh
# ./postinstall.sh
```

To check that Actian Zen is installed and running properly, run the following commands. If you see the version information of the Actian Zen engine displayed, then the database engine is running properly.

```
# su zen-svc
$ isql demodata
```

```
SQL> select @@version
SQL> quit
$ exit
```

4

4 Install Node.js

Next, we need to install Node.js.

CentOS:

```
# curl -sL https://rpm.nodesource.com/setup_10.x | bash -
# yum remove -y nodejs npm
# yum install -y nodejs
# node -v
```

Ubuntu:

```
# apt install nodejs
# node -v
```

In either environment, if the node -v command returns a Node.js version number, then Node.js has been installed successfully.

5

5 Create a sample folder

Create an “ActianZen” directory as a working directory for the sample application. Within it, create two subdirectories — “Swig4” and “Demo.” You will place various files in each of these subdirectories during subsequent steps.

```
# exit
$ cd /home/test
$ mkdir ActianZen
$ cd ActianZen
$ mkdir Swig4
$ mkdir Demo
```

We will now place files in each directory.

6

6 Installing SWIG

Copy the latest SWIG file from [www.swig.org/download.html.](www.swig.org/download.html) You will need to install SWIG in the **/home/test/ActianZen/Swig4** directory just created in order to generate a wrapper library that Node.js's JavaScript can use to call the Btrieve2 API, which is a C / C++ library.

※ **Note:** If you already have an older version of SWIG in your environment, you should replace it with the latest files. You can remove older versions with the following command:

CentOS:

```
$ sudo yum erase swig
```

Ubuntu:

```
$ sudo apt remove swig
```

Next, extract the swig source files: -

```
$ cd /home/test/ActianZen/Swig4
$ tar -zxf swig-4.0.2.tar.gz
$ cd swig-4.0.2
```

Next, install php and python3, which are needed for swig to work :-

CentOS:

```
$ sudo yum install -y php
$ sudo yum install -y python3
```

Ubuntu:

```
$ sudo apt install php-dev
$ sudo apt install python3-dev
```

Configure your environment, run make and then install.

```
$ ./configure --without-python --without-perl5 --with-php
$ make -j4
$ sudo make install
$ swig -version
```

If SWIG's version information is then displayed, you have succeeded in installing SWIG.

7

7 Compile Native Add-On

Now that we have swig ready, we can compile the native add-on for Node.js.

First, we'll extract the Btrieve2 API SDK files. We'll assume that the **Zen-SDK-Btrieve-2API-linux-noarch-14.11.014.000.tar.gz** file is located in the **ActianZen/Demo** directory.

```
$ cd /home/test/ActianZen/Demo
$ tar -zxf Zen-SDK-Btrieve2API-linux-noarch-14.11.014.000.tar.gz
$ cd Zen-SDK-Btrieve2API-000-linux-noarch-
  14.11.014/swig/btrieveJavascript
```

Next, we'll use SWIG to generate a Btrieve2 API wrapper file.
(creating a file named **btrieveJavascript_wrap.cxx**)

```
$ swig -javascript -node -I../../include -c++ btrieveJavascript.swig
```

Finally, we'll create the Node.js native add-on.

To do this, first we'll install the add-on creation tool node-gyp.

CentOS:

```
$ sudo npm install -g node-gyp
```

Ubuntu:

```
$ sudo apt install aptitude
$ sudo aptitude install node-gyp
```

Now that node-gyp is installed, we're ready to create a native Node.js add-on.

```
$ node-gyp configure
$ node-gyp build
```

8 Running the sample program

First, we assign the necessary access permissions to the folder and add the Zen library folder to the path.

```
$ chmod a+rwx .
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/actianzen/lib64
```

Next copy the sample program **bcreate_insert_read_sample2.js** to the current directory: -

/home/test/ActianZen/BtrieveJavascript/Zen-SDK-Btrieve2API-000-linux-noarch-14.11.014/swig/btrieveJavascript

Now we can run it by executing: -

```
$ node bcreate_insert_read_sample2.js
```

The location where the data file is created is defined at the beginning of the script as follows : -

```
btrieveFileName = "/usr/local/actianzen/data/test_cr_ins_read.btr"
```

The file must be created in a location that the Zen Engine can read and write to.

※ Note: The account under which the Zen engine operates is named **zen-svc**, group name **zen-data**.

Also, although the sample program includes a command to delete the data file created in the last step, these last lines have been commented out so that the data file created remains so that you can verify the data created.

You can access the **test_cr_ins_read.btr** file using Function Executor from a Windows client to see what was written to the file.

ACTIAN

2300 Geng Rd, Suite 150, Palo Alto, CA 94303
+1 888 446 4737 [Toll Free] | +1 650 587 5500 [Tel]