Accessing Zen v15 from C# on Windows Using the Btrieve 2 API

A White Paper From

GOLDSTAR SOFTWARE

www.GoldstarSoftware.com

For more information, see our web site at
http://www.goldstarsoftware.com

Accessing Zen v15 from C# on Windows Using Btrieve 2

Last Updated: March 2023

The Actian Zen database engine (formerly known as Actian PSQL) supports a wide variety of application programming interfaces (APIs) to access the data. Some of these interfaces leverage a lower-level interface, commonly known as the Btrieve API to provide the needed performance and flexibility. One of the problems with the Btrieve API is that it uses byte-aligned variables and structures, and it can be cumbersome to use from a newer, object-oriented language. It can also have a bit of a learning curve, which can be daunting to newer developers.

This paper defines the basic steps required to build a simple Btrieve 2 application using the Actian Zen v15 database environment. The example is a bit contrived – it is based on the sample code provided with the Actian SDK downloads, but it should be easy enough for a knowledgeable developer to expand upon these concepts and make more complicated applications.

Important Pre-requisites

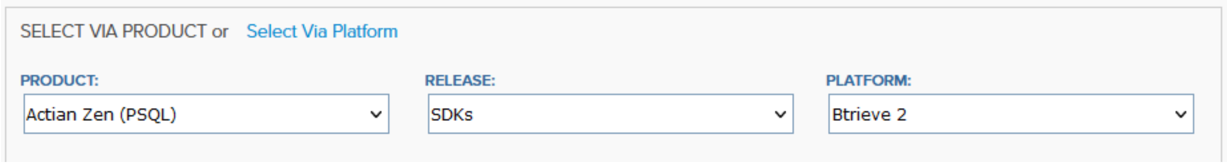
The following pre-requisites should be in place before you start.

1) You should have a Windows operating system properly installed.
2) You should have Microsoft Visual Studio properly installed. Screenshots within this paper are from VS2019, but the requirements should be similar for both newer and older versions. If you need help with this, please see Microsoft's documentation.
3) You should have Actian Zen v15.11 (or newer) installed and running. If you are working on a stand-alone development machine, you can install either the Workgroup Engine or the Server Engine. (Both will work just fine with a 30-day trial.) If you are working within a shared environment, you should have a Zen v15 Workgroup Engine or Server Engine installed where the database files are located, and the Zen v15 Client installed on your development computer. See our install documentation at http://www.goldstarsoftware.com/ineedzen15.asp for complete instructions if needed.
4) We are creating the development environment in a folder C:\Develop. If you are working in a different folder, then you will need to alter some command lines herein.

Download and Install the Btrieve 2 SDK

Actian has a formal SDK download for the Btrieve 2 API components, which you can find on their web site following these directions.

1) Go to https://esd.actian.com/product/Zen_PSQL in a web browser.
2) In the boxes provided, select **SDKs** and **Btrieve 2**.



PRODUCT: Actian Zen (PSQL)
RELEASE: SDKs
PLATFORM: Btrieve 2

3) Scroll down and open up the link for **Btrieve 2**, then click on the **DOWNLOAD** button to download the needed SDK component. (Note that there are multiple versions, as well as downloads for both Windows and Linux. This paper assumes you are building on Windows using the v15.11.014 release.)

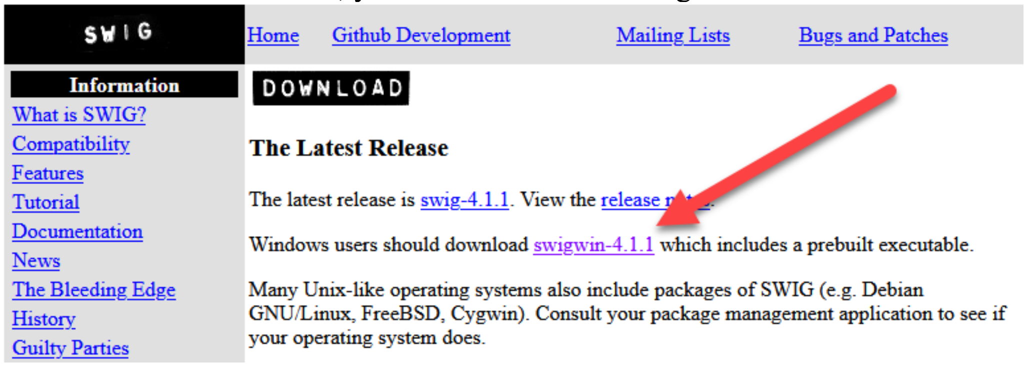
4) After it downloads, double-click the EXE file to launch the installer.
5) Change the **Unzip to Folder** to your project base folder (**C:\Develop\SDK15** in my example**)** and click **Unzip**.

You now have the Btrieve 2 components and sample code on your workstation.

Downloading and Installing SWIG

The SWIG components are the second piece:

1) In your web browser, go to http://www.swig.org/download.html and select the version of SWIG you want to download. At the time of this writing, the current version is 4.1.1.
2) As indicated for Windows, you should select the swigwin-4.1.1 file:



The Latest Release

The latest release is swig-4.1.1. View the release notes.

Windows users should download swigwin-4.1.1 which includes a prebuilt executable.

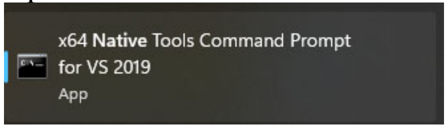
Many Unix-like operating systems also include packages of SWIG (e.g. Debian GNU/Linux, FreeBSD, Cygwin). Consult your package management application to see if your operating system does.

3) There is no installer for SWIGWIN. Instead, simply unzip the entire file into your development folder, creating **C:\Develop\swigwin-4.1.1** in the process.

Generate the Btrieve 2 C# Components

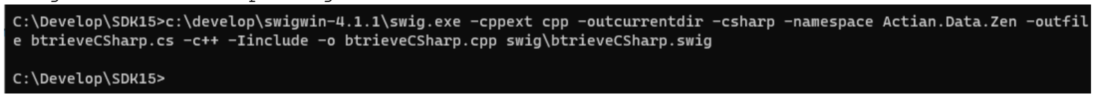
Before you can work in the Btrieve 2 API, you have to generate the needed components to allow the C# environment to see and understand the new API calls.

1) Open the Visual Studio x64 Native Tools Command Prompt.



2) Change the directory to the Btrieve 2 SDK folder that you created above.
3) Generate the Btrieve 2 C# components using the following command line. If your SWIGWIN folder differs, be sure to specify the correct path to the SWIG executable:

```
c:\develop\swigwin-4.1.1\swig.exe -cppext cpp -outcurrentdir -csharp -namespace
Actian.Data.Zen -outfile btrieveCSharp.cs -c++ -Iinclude -o btrieveCSharp.cpp
swig\btrieveCSharp.swig
```



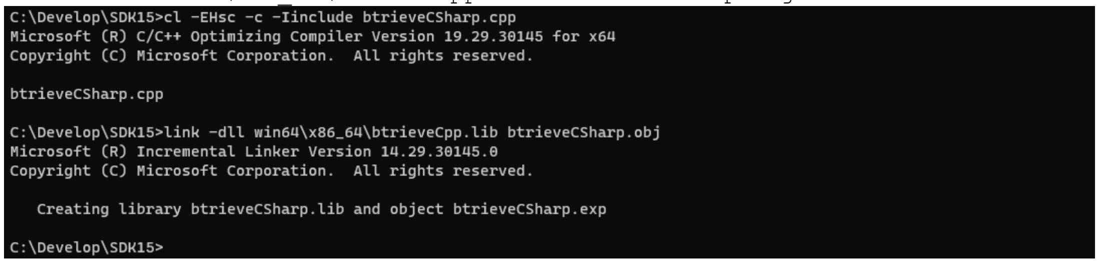
```
C:\Develop\SDK15>c:\develop\swigwin-4.1.1\swig.exe -cppext cpp -outcurrentdir -csharp -namespace Actian.Data.Zen -outfil
e btrieveCSharp.cs -c++ -Iinclude -o btrieveCSharp.cpp swig\btrieveCSharp.swig

C:\Develop\SDK15>
```

4) Build the Btrieve 2 C# components with the following two commands:

```
cl -EHsc -c -Iinclude btrieveCSharp.cpp
```

```
link -dll win64\x86_64\btrieveCpp.lib btrieveCSharp.obj
```



```
C:\Develop\SDK15>cl -EHsc -c -Iinclude btrieveCSharp.cpp
Microsoft (R) C/C++ Optimizing Compiler Version 19.29.30145 for x64
Copyright (C) Microsoft Corporation.  All rights reserved.

btrieveCSharp.cpp

C:\Develop\SDK15>link -dll win64\x86_64\btrieveCpp.lib btrieveCSharp.obj
Microsoft (R) Incremental Linker Version 14.29.30145.0
Copyright (C) Microsoft Corporation.  All rights reserved.

   Creating library btrieveCSharp.lib and object btrieveCSharp.exp

C:\Develop\SDK15>
```

5) From an **administrative** Command Prompt, install the Btrieve 2 C# components into the Zen\Bin folder.

```
copy C:\Develop\SDK15\btrieveCSharp.cs "C:\Program Files\Actian\Zen\bin"
copy C:\Develop\SDK15\btrieveCSharp.dll "C:\Program Files\Actian\Zen\bin"
```

Compile and Run the Provided Sample Application

Actian provides a simple test application as part of the Btrieve 2 SDK download. (A copy of that application source code is included in the appendix for completeness.) You should compile and run this sample application first to confirm that the environment is configured correctly.

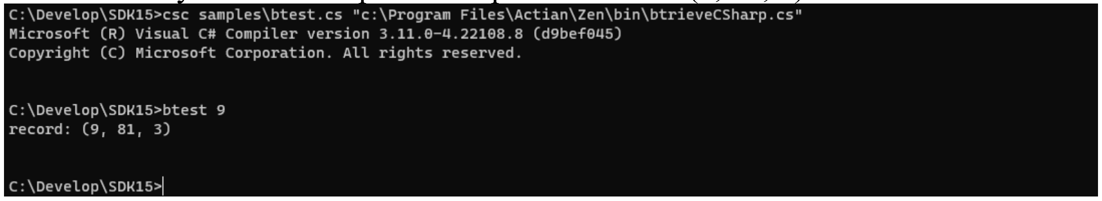
1) Compile the test application with this command:

```
csc samples\btest.cs "c:\Program Files\Actian\Zen\bin\btrieveCSharp.cs"
```

2) Execute the compiled code with this command:

```
btest.exe 9
```

3) Confirm that you see the expected output of "record: (9, 81, 3)":



```
C:\Develop\SDK15>csc samples\btest.cs "c:\Program Files\Actian\Zen\bin\btrieveCSharp.cs"
Microsoft (R) Visual C# Compiler version 3.11.0-4.22108.8 (d9bef045)
Copyright (C) Microsoft Corporation. All rights reserved.


C:\Develop\SDK15>btest 9
record: (9, 81, 3)


C:\Develop\SDK15>
```

If you get a permission error message, then you may lack sufficient rights to the current directory.

At this point, you now have the entire environment built and ready to use. You should be able to modify the sample code as needed and build your own applications from here.

Create a New Visual Studio Project

In an ideal world, we would have simple instructions on how to configure the Visual Studio 2019 environment so that you can build applications directly from the GUI. However, we don't have those instructions yet. Check back to our web site later on to see if this document has been updated.

A Few Notes About the Source Code

Now that we have built the code and it is working, let's review a few key components of that source code:

- Line 4 includes the needed components for the compiler to find the Btrieve 2 API definitions.
- Line 10 defines the file name that will be created.

- Lines 15 through 21 define the data structure to be used for the table. Notice that this data is packed on a Byte boundary – this is very important for Btrieve-based structures, which are typically byte-packed!
- The Serialize and Deserialize functions (lines 23-42) are used to ensure that the data is available in a global memory block via a pointer, which is required for Btrieve 2 to access the data in memory. If you elect to NOT byte-pack your data structures in memory, then you can add dedicated packing/unpacking routines into these functions instead.
- The functions included in this module (createFile, openFile, etc.) are only examples. They all include error handling to the console and GOTO statements, which you can easily remove if needed. You may also elect to simply include the desired Btrieve 2 API calls directly in the main module and forgo all of these extra functions entirely.
- When you run this application, it creates a new database file, loads that file with 256 data records, creates an index on the table, and retrieves the requested record. After that is completed, the file is closed and deleted. Again, this is a simple example, but it shows many of the basics.

For a complete description of the Btrieve 2 API, please go to Actian's web site here:
https://docs.actian.com/zen/btrieve2v15/html/index.html

Expanding Your Knowledge

Want to know if you understand what is going on? Try these modifications:

1) Remove the **deleteFile** call from the code and recompile and run the code again. Examine the resulting database file with **BUTIL -STAT** or the *Maintenance Utility* to see what was actually created. Note the record length and key definition of this table.
2) Prior to retrieving the record, display the number of records in the table to the screen. This will require building a new function that calls the **GetInformation()** API and returns the correct value from the returned structure.
3) The **retrieveRecord** function returns ONLY the requested value provided on the command line. Create a new function called **retrieveNextRecords** that returns the NEXT 10 records and displays their values on-screen as well. *Hint*: Start with the **retrieveRecord** function, but then switch it up to use **RecordRetrieveNext()**. For bonus points, specify the number of records to return in the function definition and display that number of records. (Don't forget error handling in case you try to read past the end of the file.)
4) Extend the data structure to change the field "**x**" from a 1-byte field to a 2-byte (WORD) value and increase the value of MAX_X from 255 to 4096. This is a bit more complicated. You will need to change the record structure definition, of course. Do you need to change anything inside the **createFile** function? What did you have to change within **loadFile**? Did it work? *Hint*: If you got a Duplicate Key error, you missed changing something in **createIndex**.

From here, you're only limited by your imagination!

Finding More Help

If you have other problems getting this to work, you can contact Actian directly through their web forums at https://communities.actian.com/s/ for more help. If you need some additional hand-holding, Goldstar Software may be able to assist you as well through our Developer Jump Start Program. You can contact us at 1-708-647-7665or via the web at http://www.goldstarsoftware.com.

Appendix A: Sample btest.cs Code

The following test application is provided by Actian as part of the SDK and is provided for completeness.

```
using System;
using System.Linq;
using System.Runtime.InteropServices;
using Actian.Data.Zen;

namespace btest
{
    class Program
    {
        static string btrieveFileName = "squaresAndSquareRoots.btr";

        const int MIN_X = 0;
        const int MAX_X = 255;

        [StructLayout(LayoutKind.Sequential, Pack = 1)]
        struct record_t
        {
            public byte x;
            public UInt16 xSquared;
            public double xSquareRoot;
        };

        public static byte[] Serialize<T>(T s) where T : struct
        {
            var size = Marshal.SizeOf(typeof(T));
            var array = new byte[size];
            var ptr = Marshal.AllocHGlobal(size);
            Marshal.StructureToPtr(s, ptr, true);
            Marshal.Copy(ptr, array, 0, size);
            Marshal.FreeHGlobal(ptr);
            return array;
        }

        public static T Deserialize<T>(byte[] array) where T : struct
        {
            var size = Marshal.SizeOf(typeof(T));
            var ptr = Marshal.AllocHGlobal(size);
            Marshal.Copy(array, 0, ptr, size);
            var s = (T)Marshal.PtrToStructure(ptr, typeof(T));
            Marshal.FreeHGlobal(ptr);
            return s;
        }

        static Btrieve.StatusCode createFile(ref BtrieveClient btrieveClient)
        {
            Btrieve.StatusCode status;
            BtrieveFileAttributes btrieveFileAttributes = new BtrieveFileAttributes();
            record_t record = new record_t();

            // If SetFixedRecordLength() fails.
            if ((status = btrieveFileAttributes.SetFixedRecordLength(Marshal.SizeOf(record))) !=
    Btrieve.StatusCode.STATUS_CODE_NO_ERROR)
            {
                Console.WriteLine("Error: BtrieveFileAttributes.SetFixedRecordLength():{0}:{1}.\n", status,
    Btrieve.StatusCodeToString(status));
                goto leave;
            }

            // If FileCreate() fails.
            if ((status = btrieveClient.FileCreate(btrieveFileAttributes, btrieveFileName,
    Btrieve.CreateMode.CREATE_MODE_OVERWRITE)) != Btrieve.StatusCode.STATUS_CODE_NO_ERROR)
            {
                Console.WriteLine("Error: BtrieveClient.FileCreate():{0}:{1}.\n", status, Btrieve.StatusCodeToString(status));
                goto leave;
            }

            leave:
            return status;
        }

        static Btrieve.StatusCode openFile(ref BtrieveClient btrieveClient, ref BtrieveFile btrieveFile)
        {
            Btrieve.StatusCode status;

            // If FileOpen() fails.
        if ((status = btrieveClient.FileOpen(btrieveFile, btrieveFileName, null, Btrieve.OpenMode.OPEN_MODE_NORMAL)) !=
Btrieve.StatusCode.STATUS_CODE_NO_ERROR)
        {
            Console.WriteLine("Error: BtrieveClient.FileOpen():{0}:{1}.\n", status, Btrieve.StatusCodeToString(status));
            goto leave;
        }

        leave:
        return status;
    }

    static Btrieve.StatusCode loadFile(ref BtrieveFile btrieveFile)
    {
        Btrieve.StatusCode status = Btrieve.StatusCode.STATUS_CODE_NO_ERROR;
        int i;
        record_t record = new record_t();
        byte[] recordBytes;

        // For all the values of x.
        for (i = MIN_X; i <= MAX_X; i++)
        {
            record.x = (byte)i;
            record.xSquared = (UInt16)(i * i);
            record.xSquareRoot = Math.Sqrt((double)i);
            recordBytes = Serialize(record);

            // If RecordCreate() fails.
            if ((status = btrieveFile.RecordCreate(recordBytes, Marshal.SizeOf(record))) !=
Btrieve.StatusCode.STATUS_CODE_NO_ERROR)
            {
                Console.WriteLine("Error: BtrieveFile.RecordCreate():{0}:{1}.\n", status,
Btrieve.StatusCodeToString(status));
                goto leave;
            }
        }

        leave:
        return status;
    }

    static Btrieve.StatusCode closeFile(ref BtrieveClient btrieveClient, ref BtrieveFile btrieveFile)
    {
        Btrieve.StatusCode status;

        // If FileClose() fails.
        if ((status = btrieveClient.FileClose(btrieveFile)) != Btrieve.StatusCode.STATUS_CODE_NO_ERROR)
        {
            Console.WriteLine("Error: BtrieveClient.FileClose():{0}:{1}.\n", status, Btrieve.StatusCodeToString(status));
            goto leave;
        }

        leave:
        return status;
    }

    static Btrieve.StatusCode deleteFile(ref BtrieveClient btrieveClient)
    {
        Btrieve.StatusCode status;

        // If FileDelete() fails.
        if ((status = btrieveClient.FileDelete(btrieveFileName)) != Btrieve.StatusCode.STATUS_CODE_NO_ERROR)
        {
            Console.WriteLine("Error: BtrieveClient.FileDelete():{0}:{1}.\n", status, Btrieve.StatusCodeToString(status));
            goto leave;
        }

        leave:
        return status;
    }

    static Btrieve.StatusCode createIndex(ref BtrieveFile btrieveFile)
    {
        Btrieve.StatusCode status;
        BtrieveIndexAttributes btrieveIndexAttributes = new BtrieveIndexAttributes();
        BtrieveKeySegment btrieveKeySegment = new BtrieveKeySegment();

        // If SetField() fails.
        if ((status = btrieveKeySegment.SetField(0, 1, Btrieve.DataType.DATA_TYPE_UNSIGNED_BINARY)) !=
Btrieve.StatusCode.STATUS_CODE_NO_ERROR)
        {
            Console.WriteLine("Error: BtrieveKeySegment.SetField():{0}:{1}.\n", status,
Btrieve.StatusCodeToString(status));
            goto leave;
        }

        // If AddKeySegment() fails.
        if ((status = btrieveIndexAttributes.AddKeySegment(btrieveKeySegment)) != Btrieve.StatusCode.STATUS_CODE_NO_ERROR)
        {
            Console.WriteLine("Error: BtrieveIndexAttributes.AddKeySegment():{0}:{1}.\n", status,
Btrieve.StatusCodeToString(status));
            goto leave;
        }

        // If IndexCreate() fails.
        if ((status = btrieveFile.IndexCreate(btrieveIndexAttributes)) != Btrieve.StatusCode.STATUS_CODE_NO_ERROR)
        {
            Console.WriteLine("Error: BtrieveFile.IndexCreate():{0}:{1}.\n", status, Btrieve.StatusCodeToString(status));
            goto leave;
        }

        leave:
        return status;
    }

    static Btrieve.StatusCode retrieveRecord(ref BtrieveFile btrieveFile, ref byte key)
    {
        Btrieve.StatusCode status = Btrieve.StatusCode.STATUS_CODE_NO_ERROR;
        record_t record = new record_t();
        byte[] recordBytes = Serialize(record);
        byte[] keyBytes = Serialize(key);

        // If RecordRetrieve() fails.
        if (btrieveFile.RecordRetrieve(Btrieve.Comparison.COMPARISON_EQUAL, Btrieve.Index.INDEX_1, keyBytes,
Marshal.SizeOf(key), recordBytes, Marshal.SizeOf(record)) != Marshal.SizeOf(record))
        {
            status = btrieveFile.GetLastStatusCode();
            Console.WriteLine("Error: BtrieveFile.RecordRetrieve():{0}:{1}.\n", status,
Btrieve.StatusCodeToString(status));
            goto leave;
        }

        record = Deserialize<record_t>(recordBytes);
        Console.WriteLine("record: ({0}, {1}, {2})\n", record.x, record.xSquared, record.xSquareRoot);

        leave:
        return status;
    }

    static int Main(string[] args)
    {
        BtrieveClient btrieveClient = new BtrieveClient();
        Btrieve.StatusCode status = Btrieve.StatusCode.STATUS_CODE_UNKNOWN;
        BtrieveFile btrieveFile = new BtrieveFile();
        byte key;
        UInt64 integerValue;

        // If the incorrect number of arguments were given.
        if (args.Count() != 1)
        {
            Console.WriteLine("Usage: {0} uint8_value\n", Environment.GetCommandLineArgs()[0]);
            goto leave;
        }

        integerValue = UInt64.Parse(args[0]);

        // If integerValue is out of range.
        if ((integerValue < MIN_X) || (integerValue > MAX_X))
        {
            Console.WriteLine("Usage: {0} uint8_value\n", Environment.GetCommandLineArgs()[0]);
            goto leave;
        }

        key = (byte)integerValue;

        // If createFile() fails.
        if ((status = createFile(ref btrieveClient)) != Btrieve.StatusCode.STATUS_CODE_NO_ERROR)
        {
            goto leave;
        }

        // If openFile() fails.
        if ((status = openFile(ref btrieveClient, ref btrieveFile)) != Btrieve.StatusCode.STATUS_CODE_NO_ERROR)
        {
            goto leave;
        }
```

```
            // If loadFile() fails.
            if ((status = loadFile(ref btrieveFile)) != Btrieve.StatusCode.STATUS_CODE_NO_ERROR)
            {
                goto leave;
            }

            // If createIndex() fails.
            if ((status = createIndex(ref btrieveFile)) != Btrieve.StatusCode.STATUS_CODE_NO_ERROR)
            {
                goto leave;
            }

            // If retrieveRecord() fails.
            if ((status = retrieveRecord(ref btrieveFile, ref key)) != Btrieve.StatusCode.STATUS_CODE_NO_ERROR)
            {
                goto leave;
            }

            // If closeFile() fails.
            if ((status = closeFile(ref btrieveClient, ref btrieveFile)) != Btrieve.StatusCode.STATUS_CODE_NO_ERROR)
            {
                goto leave;
            }

            // If deleteFile() fails.
            if ((status = deleteFile(ref btrieveClient)) != Btrieve.StatusCode.STATUS_CODE_NO_ERROR)
            {
                goto leave;
            }

            leave:
            // If there wasn't a failure.
            if (status == Btrieve.StatusCode.STATUS_CODE_NO_ERROR)
                return 0;

            return 1;
        }
    }
}
```