

# RHOZET

Universal Media Transcoding

## Carbon SDK Samples – Version 2

Revision 1.0

Spring 2009

### Table of Contents

Introduction .....	3
Getting Started.....	4
Installing the SDK .....	6
Sample Code .....	7
Logger.....	7
TimeCode .....	8
Carbon SDK .....	9
Coder.....	9
Jobs .....	9
Preset .....	9
Applications.....	9
Version .....	10
Evaluate.....	11
Queue.....	12
Overriding Preset/Profile Paramets.....	13

CarbonWeb .....	15
Other API.....	16
Revisions .....	17
Changes for Revision 1.0.....	17

## Introduction

This document describes the Rhozet Carbon Coder samples version 2 which interface with Carbon through its API to carry out a series of Tasks described in the following sections. The code is a great starting point for your project covering a range of tasks you can program through the API (See [Revisions](#) for the latest changes to this doc and the sample)

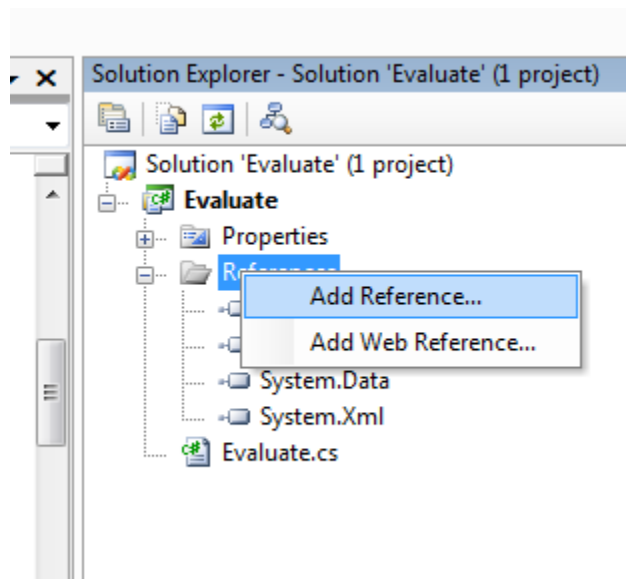
Whilst the code is not intended as production level code – it does abstract and isolate the tasks from the process of rendering the API with Carbon – to this end it provides a common shared SDK framework that abstracts the details away from the application.

The goal of that design is to enable you to utilize the CarbonSDK without having to know anything about carbon API – to this end three or four lines of code will enable you to submit, for example, a job to encode. This is discussed in our first section – Getting Started, go here if you just want to get going straight away, and feel free to use this framework in your own workflow.

## Getting Started

This section gives you the fastest entry point to working with Carbons API. You'll need to have Carbon Coder installed (with its license) as well as Visual Studio C# (free C# copy [here](#)). We're going to create a simple project to evaluate a media file to get some details about our file. Before we do this – take a look at the files in the carbon\sdk\bin directory – in here is a pre-built version of all the projects – if you want to open up a command prompt and just try one out . Otherwise you can read on to build your first sample!

Open up Visual Studio and create a new command line project. Once you are ready, add a reference to the CarbonSDK.DLL located in the SDKs Bin directory – to do this open up the reference on your project as show below and browse to and add CarbonSDK.DLL found in the Bin directory of our installed files.



Once you have added this you are ready to start writing some code. You have just a few things to do :-

1. Add a using statement for the SDK so we can use the sample framework
  - a. `using CarbonSDK;`
2. Setup your connection to Carbon
  - a. `ICoder coder = new Coder("127.0.0.1", 1120);`
3. Create an interface to work with jobs
  - a. `IJob job = coder.GetJobInterface ();`
4. Run a command, lets evaluate a file for this sample
  - a. `Jobs.JobElement q =  
job.JobEvaluate("c:\\carbon\\SDK\\sourcemedi\\sample.wmv");`

If you place a breakpoint on 4.a – and then step through this line, you can review the result of the evaluate call in the new object q – in here a property called string result, this contains the response from Carbon along with details of your media file you can examine. You can even use these later on to, say, maintain the video width from source to target.

All jobs in carbon are different but the output follows the same schema as defined in the API documents – some typical output is show below.

```
<Reply Success="TRUE">  
(removed redundant xml...)  
<PublicDescription RZMETA_VideoBitrate_kbps="15000.0" RZMETA_BitsPerPixel="24"  
RZMETA_AudioBitrate_kbps="256.0" Duration.QWD="169263000">  
<Video Size_X.DWD="1280" Size_Y.DWD="720" AverageFrameDuration.QWD="900000" Codec="Windows  
Media Video 9" Interlacing="Progressive" Video_Aspect_X.DWD="16" Video_Aspect_Y.DWD="9"/>  
</PublicDescription>
```

Later on we will see how we can use this evaluation technique to do some cool dynamic updates for encoding files.

## Installing the SDK

In order to build and run these sample ensure you have a development environment (such as a copy of Visual Studio Express – you can download a free copy [here](#)). All the samples are written in C# although the SDK framework could be used in any number of environments by importing the sdk library in the SDK\bin directory

Please make sure that you have either Carbon Server or Carbon Coder installed on the same machine that you will be executing these examples. You can also download the API documentation from [here](#) for reference.

There are several common directories,

- Bin – contains prebuilt dll's and all the executable samples from this package.
- Output – is where we write the output media files from Rhozet by default.
- CarbonSDK – this is a common directory that provides a framework for all the samples.
  - *Note – we post build copy the output DLL to the Bin directory –CarbonSDK.DLL.*

Your first task is to open the CarbonSDK project file - CarbonSDK.sln, and compile this solution which compiles all the sample projects. Once this dll is built successfully a copy is made (regardless of debug vs. release) in the Carbon\SDK\Bin directory – all other projects use this library and reference it directly in this location. If you wish to install the SDK somewhere else then you will need to modify absolute file path references accordingly in each project file.

You can now load each project in turn and compile these. If you make any changes to the CarbonSDK library, don't forget to recompile the library AND the sample project. In the SDK directory is also the samples solution – this loads all the project files into one Visual Studio instance.

Here is the manifest for the installed directories:

Directory	Use
C:\carbon\sdk	This Document
C:\carbon\sdk\bin	Compiled versions of the samples
C:\carbon\sdk\CarbonSDK	Carbon SDK common assembly
C:\carbon\sdk\SampleJobs	Job evaluate and queue sample
C:\carbon\sdk\SampleWS	As Jobs, but deployed via a web service
C:\carbon\sdk\SampleJobMonitor	Windows Forms application for viewing job status

## Sample Code

### Logger

A simple console IO class that writes out a string using a color param as well to enable you to highlight the output – for instance if there is an exception to be printed.

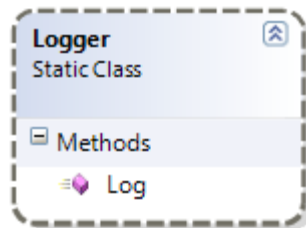


Figure 1 - Class Diagram, Logger

## TimeCode

Carbon makes use of a 27Mhz clock. What this means is there are 27 000 000 ticks per second – we’ve added a helper class to help you more easily figure out things like mark in time from frame time in the class defined below.

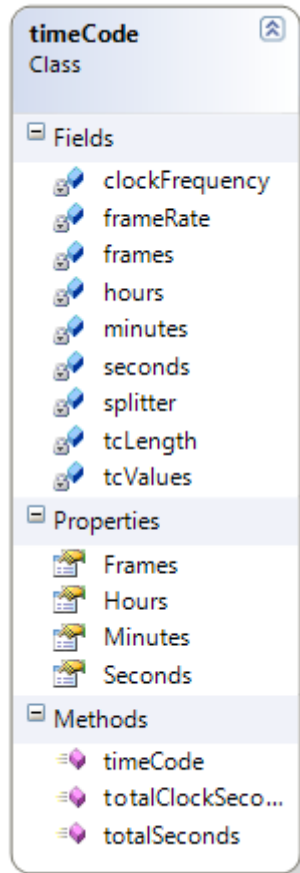


Figure 1 - Class Diagram, timeCode

## Carbon SDK

The Carbon SDK is a common set of classes used by all the following sample projects. This assembly can be directly used in your applications or serve as a guide for developing your own. It makes of several classes – Jobs, Coder, Presets among others. These three form the core work that defines the abilities of the assembly.

## Coder

This class collects some more common operations on the services running Carbon. Some examples include being able to gather version and license information. You need an interface to this class to work with the job interface – in doing this all initialization for the connection is done in one place.

## Jobs

Note that for our current 3.12 implementation the socket is kept open for all clients – therefore the only safe way to gather a response is to loop and take all the bytes (from the header of the response) until done – there is no other way of doing this safely – the socket class shows an example of this.

## Preset

## Applications

The structure for the applications is very similar – add a reference to bin\CarbonSDK.dll, add a using statement for CarbonSDK, Create an interface to ICoder and that's it!

There are currently three sample projects –

SampleJobMonitor – shows a UI version of a tool to monitor the job process queues.

SampleJobs – how to work with job evaluate and job queue.

SampleWS – exactly as SampleJobs but exposed through a Web service layer.

## Version

Version simply gathers the version number of Coder – it returns this in an XML packet that is written to the console – here is the code from this application – the flow is common for all applications –

```
ICoder coder = new Coder("127.0.0.1", 1120);  
string version = coder.VersionInfo();  
Console.WriteLine (version);
```

This is a great way to verify Carbon is online and running – like a network Ping.

Typical Response:

```
<Reply Version="Rhozet Carbon Coder ver. 03.14.00.00" Success="TRUE"/>
```

## Evaluate

Evaluate takes a source input file and asks Carbon for details of this file which are returned in the JobElement .result string parameter. Note also that this method packs out some of the data in this same structure including aspect ratio, video dimensions, frame rate and so on. This becomes useful when you want to re-encode a file using a preset – but need to maintain the aspect ratio or video size, for example. Here is the simple code to call JobEvaluate;

```
ICoder coder = new Coder("127.0.0.1", 1120);
string version = coder.VersionInfo();

//Job Evaluation - observer values in JobElement
IJob job = coder.GetJobInterface();
Jobs.JobElement q =
job.JobEvaluate("c:\\carbon\\SDK\\sourcemedi\\sample.wmv");
```

The q.result value is the response from Carbon, see a sample below:

```
<Reply Success="TRUE">
<JobEvaluateResult>
<Sources>
<Module_0 Inpoint.QWD="-1" Outpoint.QWD="-1" Duration.QWD="169263000" MultiSource.DWD="1"
OriginalModuleIndex.DWD="0">
<ModuleData>
<StreamTypeTable StreamSelector_0.DWD="0" StreamType_0.DWD="0" StreamPtr_0.DWD="0"/>
<SourceModules>
<MultiSrcModule_0 MultiSource.DWD="0" Filename="c:\carbon\sdk\SourceMedia\Sample.wmv"
CreatorMachine="LAP-J7QJTF1" FullUNCFilename="\\LAP-
J7QJTF1\C$\carbon\sdk\SourceMedia\Sample.wmv">
<ModuleData ForceFixedFrameRate.DWD="0" ForcedFixedFrameRate.DBL="30.000000"
AspectRatio.BIN="AQAAAAEAAAA=" PixelAspect_X.DWD="1" PixelAspect_Y.DWD="1"
FrameMode.DWD="0"/>
</MultiSrcModule_0>
</SourceModules>
</ModuleData>
<Filter_0/>
<Filter_1/>
<PublicDescription RZMETA_VideoBitrate_kbps="15000.0" RZMETA_BitsPerPixel="24"
RZMETA_AudioBitrate_kbps="256.0" Duration.QWD="169263000">
<Video Size_X.DWD="1280" Size_Y.DWD="720" AverageFrameDuration.QWD="900000" Codec="Windows
Media Video 9" Interlacing="Progressive" Video_Aspect_X.DWD="16" Video_Aspect_Y.DWD="9"/>
</PublicDescription>
</Module_0>
</Sources>
<Destinations/>
<ProjectSettings Stitching.DWD="0">
<KernelFlags KP_LetterBoxParam.DWD="0" KP_FrameRateMode.DWD="2"
KP_SpecialMPEGScaling.DWD="1" KP_SpecialDeinterlacing.DWD="1"
KP_SpecialID1toDVScaling.DWD="1" KP_RGB2YUV601.DWD="1" KP_YUV2RGB601.DWD="1"
KP_AdjustDurationLimit.DWD="5" KP_AdjustDurationEnabled.DWD="1" KP_RestrictQuality.DWD="0"
KP_AnamorphicScalingLimit.DWD="50" KP_AnamorphicScalingLimitEnabled.DWD="1"
KP_SpecialVBIScaling.DWD="1" KP_DefaultSTLanguage.DWD="28261"
KP_DefaultAudioLanguage.DWD="0" KP_ForceInternalMPEGDecoder.DWD="1"
KP_SetChapterAtStitch.DWD="0" KP_UseDropFrame.DWD="1" KP_UseMPEGTimeStamp.DWD="0"
KP_MaxMediaCut.DWD="5" KP_UseLegacyMPEG.DWD="0" KP_UseLegacyQT.DWD="0"/>
</ProjectSettings>
</JobEvaluateResult>
</Reply>
```

## Queue

The IJob interface also support queuing of Jobs to carbon. There are three primary inputs required to do this – source file name, destination directory and the GUID of a preset you defined (outside in Carbon say) – with these three values we can queue a job for carbon as follows:

```
ICoder coder = new Coder("127.0.0.1", 1120);
string version = coder.VersionInfo();

IJob job = coder.GetJobInterface();
//WM 100kpbs 15fps QCIF
q.guid = "{19DCB198-F005-46EF-8D93-6E707D51F5D7}";
q.output = "d:\\output";
q.writename.Add("MyFileOut1.wmv");
//And finally add the source file
q.filename.Add("d:\\sourcemia\\2.wmv");
//Queue the job/s
job.JobQueue(ref q);
```

Interestingly note that the filename and writename properties are arrays, so we support multi sources and destination names - and in fact later on you will see even how to stich sources together.

Typical response in q.result.

```
<Reply Guid="{DC212831-A4F0-4525-9729-88FC11B9EFE9}" GUID="{DC212831-A4F0-4525-9729-88FC11B9EFE9}" Success="TRUE"/>
```

## Overriding Preset/Profile Params

As we mentioned earlier, through the IJob interface its possible to set the preset GUID and then to override the values in the JobElement used to provide source details to the interface. There are several supported parameters as follows:

```
/// The Carbon Coder job name
public string name;
/// The Carbon Coder job description
public string description;
/// User ID
public string user;
/// Guid for the Target output
public string guid;
/// ID returned from JobQueue
public string jobGUID;
/// LotID, only used, when part of an other job
public int lotID;
/// Priority, integer between 1 (highest) and 10 (lowest)
public int priority = 5;
/// Integer indicating what the capabilities of this job is
public int capabilities;
/// The Check in time of the job
public string checkInTime;
/// deleted when conversion is completed
public bool deleteSource;
/// Source filename
public IList<string> filename = new List<string>();
/// Mark in time
public TimeCode timeIn = new TimeCode("00:00:00:00");
/// Mark out time
public TimeCode timeOut = new TimeCode("00:00:00:00");
/// Output destination
public string output;
/// Output filenames
public IList<string> writename = new List<string>();
/// Source size x of video
public string size_x;
/// Source size y of video
public string size_y;
/// Frame Duration (use to get frame rate on 27MHz clock)
public string frameRate;
/// Aspect X
public string aspectX;
/// Aspect Y
public string aspectY;
/// Return XML from JE
public string result;//the actual xml back from Carbon API Call
/// Source XML to Queue
public string sourceXml;
/// Desintation XML to queue
public string destinationXml;
/// Use to hand off a URL to the post notify task
public string url;
/// Flag to stich sources
public bool stitch;
```

So typically to override some values on the preset, here is how you would do this in code:

```
//Some simple connection tests
ICoder coder = new Coder("127.0.0.1", 1120);
string version = coder.VersionInfo();

Jobs.JobElement q = new Jobs.JobElement ();

////WM 100kpbs 15fps QCIF
q.guid = "{19DCB198-F005-46EF-8D93-6E707D51F5D7}";
//Make things more urgent
q.priority = 2;
//Set the destination directory
q.output = "d:\\output";
q.writename.Add("MyFileOut1.wmv");

//Grab a slice of the source video
q.timeIn = new TimeCode("00:00:00:00");
q.timeOut = new TimeCode("00:00:08:00");
//Lets change the Video Size of the output
q.size_x = "200";
q.size_y = "100";
//Define a location for a notification when the job is complete
q.url = "http:\\\\rhonet.com?%jobguid%";
//lets add a second file and stitch them
q.filename.Add("d:\\sourcemedi\\2.wmv");
q.stitch = true;

//Queue the job/s
job.JobQueue(ref q);
```

Theres a lot going on here – but the comments above should explain how we pack out values in the JobElement with the values we want to alter on the preset. CarbonSDK takes care of overriding these in the xml it eventually generates to create the target/s.

Essentially we take a single source, crop out 8 seconds form this source, encode it to a size 200\*100, put the output in the output directory and call a notification URL (q.url) when the job is complete (or error'ed).

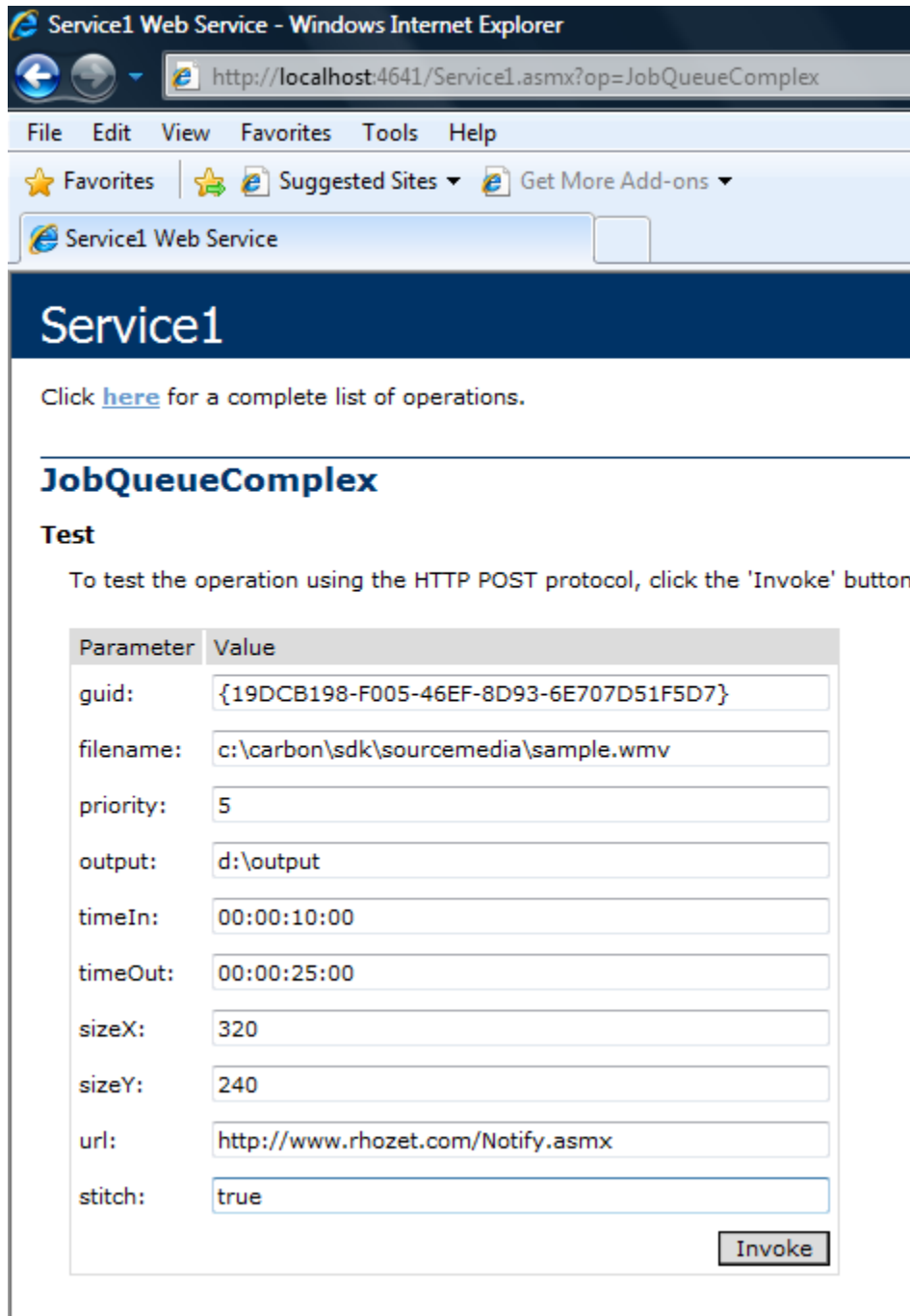
We support multiple input source files as well as multiple destinations from the profile GUID through the assembly and will override these values for all destinations. Just ensure you add sufficient names in the “writename” property to cover all the destinations in the output preset or profile.

Typical response:

```
<Reply Guid="{DC212831-A4F0-4525-9729-88FC11B9EFE9}" GUID="{DC212831-A4F0-4525-9729-88FC11B9EFE9}" Success="TRUE"/>
```

## CarbonWeb

Carbon web is essentially a replication of the previous sample but wrapped into a Web Service so that you can more easily access the interface remotely. If you run this application through a debugger you can navigate in your browser to a test page for Service1.aspx – from there you should see and can fill out the fields to test the service as shown below:



Click [here](#) for a complete list of operations.

---

### JobQueueComplex

**Test**

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
guid:	{19DCB198-F005-46EF-8D93-6E707D51F5D7}
filename:	c:\carbon\sdk\sourcemia\sample.wmv
priority:	5
output:	d:\output
timeIn:	00:00:10:00
timeOut:	00:00:25:00
sizeX:	320
sizeY:	240
url:	http://www.rhozet.com/Notify.aspx
stitch:	true

This service is hosted and processed by the ASP.NET library – so you need to make sure that your IIS service has this option enabled.

Once you hit the invoke button on the IE page – you should get a new IE window pop up with the following response (just the same xml) you get when you run the console application.

```
<?xml version="1.0" encoding="utf-8" ?>
  <string xmlns="http://rhozet.com/samplesSDK"><?xml version="1.0" encoding="UTF-8"?><Reply
    Guid="{B7CA12AF-0CB8-4EFC-B6CF-01ED9B3F6C35}" GUID="{B7CA12AF-0CB8-4EFC-B6CF-
    01ED9B3F6C35}" Success="TRUE" /></string>
```

## Other API

Here are some other methods on the IJob interface you can use to manage jobs in Carbon:

```
Jobs.JobElement[] JobList();
    Returns a List of all jobs in Carbon along with some high level
    information in the JobElement array.

Jobs.JobStatusElement JobStatus(string guid);
    Returns the status for a particular job in the JobStatusElement object.

Jobs.JobStatusElement[] JobStatusList(Jobs.JobListType type);
    Returns a filtered list of Jobs based on their state, for instance all
    currently Started Jobs.

string JobRequeue(string guid);
    Requeue a job - typically you would do this if you spotted no progress
    on a job over a long period of time.

string JobRemove(string guid);
    Remove a Job from any of the queues.

string JobStop(string guid);
    Stop a job from progressing any further.

string JobSetPriority(string guid, int priority);
    Alter the priority of a job in the queue - perhaps so it executes more
    quickly.
```

## Revisions

March 15 <sup>th</sup> 09	First revision for SDK v2	nickvh
March 24 <sup>th</sup>	Added multi source output and fixed a bug in the dest. XPATH query	nickvh

## Changes for Revision 1.0

- 1) First revision.