



# ***S.T.R.I.K.E.7 API.net Specification***

*Specification v1.7*

# Contents

Introduction.....	2
F.A.Q. ....	3
<i>I get a 'Bad Format Exception' error when I try to run my applet. How do I fix it?</i> .....	3
Events .....	4
EventArgs.....	5
AppletActive.....	6
AppletInactive.....	7
AppletTouch.....	8
AppletReset.....	9
Functions .....	10
IsOpen .....	11
AddSupportedDevice.....	12
Open.....	13
Close.....	14
UpdateScreen.....	15
Scroll .....	16
Draw .....	17
Fill.....	18
DrawText.....	19
OnAppletActive .....	20
OnAppletInactive .....	21
OnAppletTouch .....	22
OnAppletReset .....	23
Strike7API.Helpers.Gfx.....	24
GetBitmapSection .....	24
CopyImage .....	25

### Introduction

*The S.T.R.I.K.E.7 API.net is a class library that will allow easy access to the S.T.R.I.K.E.7 API through the .net framework.*

*The API will consist of a private Strike7API core that takes care of the each of the registered applets.*

*The public facing class is called Strike7Applet and will be the entry point for all applications. All C++ interop code will be hidden from the programmer. The code will be fully CLS compliant so that it can be used by any .net language.*

*This interop API can evolve in to a more fully featured system with standard controls to make it easier to build good user interfaces. Maybe using a hidden WPF window that we can inject clicks and drags into whilst grabbing the screen and sending the updates.*

**F.A.Q.**

***I get a 'Bad Format Exception' error when I try to run my applet. How do I fix it?***

*To resolve this issue you need to go in to your project settings (Visual Studio, Right-click on your project->Properties->Build) and change the platform target to x86.*

### *Events*

- `event EventHandler AppletActive;`
- `event EventHandler AppletInactive;`
- `event EventHandler AppletReset;`
- `event EventHandler<AppletTouchEventArgs> AppletTouch;`

### **EventArgs**

AppletActiveEventArgs

```
{  
    int endPointType;  
}
```

AppletTouchEvtArgs

```
{  
    long endPointId;  
    int X - Horizontal position touched.  
    int Y - Vertical position touched.  
    bool Pressed - True when the screen is touched and dragged, false when the screen  
is untouched.  
}
```

## **AppletActive**

### **Definition**

```
event EventHandler<AppletActiveEventArgs> AppletActive;
```

### **Description**

*After an applet has been opened with the OpenApplet function it must wait for the user to activate it, by selecting to view it via the V.E.N.O.M. screen. This event tells the applet that it has been selected and that it can now start rendering to the V.E.N.O.M. screen. This event is sent every time the user chooses to view the applet.*

## ***AppletInactive***

### **Definition**

```
event EventHandler AppletInactive;
```

### **Description**

*At any time a user can navigate away from an applet by pressing the home button on the S.T.R.I.K.E.7 command module. AppletInactiveEvent will be sent to the applet to let it know that it is no longer visible to the user. The applet should stop trying to render to the V.E.N.O.M. screen (all render events will be dropped). The user may choose to view the applet again at some point, so this can be treated like a 'pause' event.*



## ***AppletTouch***

### **Definition**

```
event EventHandler<AppletTouchEventArgs> AppletTouch;
```

### **Description**

*Once an applet is running and active it will receive all touch events from the V.E.N.O.M. screen. This event will be triggered every time the user touches the V.E.N.O.M. screen.*

## ***AppletReset***

### **Definition**

```
event EventHandler AppletReset;
```

### **Description**

*If the user unplugs the S.T.R.I.K.E.7 keyboard from the system. The API will send this event for any applets that were currently running on the V.E.N.O.M. screen. This includes all applets that have called the OpenApplet function, even if they have not been active. This event can be ignored if the developer would like their Applet to keep state between STRIKE7 disconnects and reconnects.*

### Functions

- `bool` IsOpen
- `bool` AddSupportedDevice(`long` deviceType)
- `virtual bool` Open(`string` name, `string` fileName)
- `virtual bool` Open(`string` name, `Bitmap` icon)
- `virtual void` Close()
- `virtual void` UpdateScreen()
- `virtual void` Scroll(`int` fromX, `int` fromY, `int` width, `int` height, `int` distanceX, `int` distanceY)
- `virtual void` Scroll(`Point` from, `Size` size, `Size` distance)
- `virtual void` Scroll(`Rectangle` rectangle, `Size` distance)
- `virtual void` Draw(`int` x, `int` y, `Bitmap` image)
- `virtual void` Draw(`Point` position, `Bitmap` image)
- `virtual void` Draw(`Point` position, `Rectangle` gfxSection, `Bitmap` image)
- `virtual void` Draw(`int` x, `int` y, `int` fromX, `int` fromY, `int` width, `int` height, `Bitmap` image)
- `virtual void` Fill(`int` x, `int` y, `int` width, `int` height, `Color` colour)
- `virtual void` Fill(`Point` position, `Size` size, `Color` colour)
- `virtual void` Fill(`Rectangle` rectangle, `Color` colour)
- `virtual void` DrawText(`int` x, `int` y, `int` size, `Color` colour, `string` text)
- `virtual void` DrawText(`Point` position, `int` size, `Color` colour, `string` text)

**NOTE: All GFX functions operate on an internal buffer. To make the screen updates visible you must call UpdateScreen() after you have finished applying your GFX updates.**

## ***IsOpen***

### **Definition**

`bool` IsOpen

### **Description**

*Returns true if the applet is open, false otherwise.*

### **AddSupportedDevice**

#### **Definition**

```
bool AddSupportedDevice(long deviceType)
```

#### **Description**

*Adds a supported device to the Applet. All supported devices must be added before the Open call is made. Any additions to the supported devices will not be propagated once the Open call is made.*

*If no supported devices are added to the Applet it will default to support the STRIKE7 only.*

*Returns true if the device was added to the supported list. It will return false if the device was already in the list.*

## Open

### Definition

```
virtual bool Open(string name, Bitmap icon) <- Main function
```

```
virtual bool Open(string name, string fileName) <- Overload
```

### Parameters

*name* – The name of the applet that is being opened.

*fileName* – A string containing the filename of a valid image file that will be used as the icon for the applet on the V.E.N.O.M. screen.

*icon* – Bitmap data for the icon.

### Returns

*bool* – True on success false otherwise.

### Description

This tells the API that an applet is now open. The function call includes an icon to display on the V.E.N.O.M. screen. The Open function is overloaded. Each overload calls back in to the main function.

**NOTE: The icon will resize to 100x70.**

## **Close**

### **Definition**

```
virtual void Close()
```

### **Description**

*Tells the API that the applet is now closed. The applet icon will be removed from the V.E.N.O.M. screen.*

## **UpdateScreen**

### **Definition**

```
virtual void UpdateScreen()
```

### **Description**

*Updates the screen with the latest GFX function.*

*To allow for a seamless screen updates, all GFX functions will not update the V.E.N.O.M. screen until this function is called. This means that a programmer can perform several GFX operations before the VENOM screen is updated. This is essentially -double buffering.*



**Scroll****Definition**

```
virtual void Scroll(int fromX, int fromY, int width, int height, int distanceX, int distanceY) <- Main function
```

```
virtual void Scroll(Point from, Size size, Size distance) <- Overload
```

```
virtual void Scroll(Rectangle rectangle, Size distance) <- Overload
```

**Parameters**

*fromX* – The position in the X-Axis where to start scrolling from.

*fromY* – The position in the Y-Axis where to start scrolling from.

*width* – The width of the rectangle to scroll.

*height* – The height of the rectangle to scroll.

*distanceX* – The number of pixels to shift the selection along the X-Axis.

*distanceY* – The number of pixels to shift the selection along the Y-Axis.

*from* – The point to scroll from.

*size* – The size of the rectangle to scroll.

*distance* – The number of pixels to shift the selection.

*rectangle* – Specifies the rectangle that will be scrolled.

**Description**

Scrolls a rectangular section of the screen to another location on the V.E.N.O.M. screen. Please note that this is not a move function. The scroll happens within the rectangle. The Scroll function is overloaded. Each overload calls back in to the main function.

**Draw****Definition**

```
virtual void Draw(int x, int y, int fromX, int fromY, int width, int height, Bitmap image) <- Main function
```

```
virtual void Draw(int x, int y, Bitmap image) <- Overload
```

```
virtual void Draw(Point position, Bitmap image) <- Overload
```

```
virtual void Draw(Point position, Rectangle gfxSection, Bitmap image) <- Overload
```

**Parameters**

*x* – Where to render the image horizontally on the V.E.N.O.M. screen.

*y* – Where to render the image vertically on the V.E.N.O.M. screen.

*fromX* – Where in the image to take the data from.

*fromY* – Where in the image to take the data from.

*width* – The width of the image being sent.

*height* – The height of the image being sent.

*image* – Bitmap data for the image that is being sent to the V.E.N.O.M. screen.

*Position* – Where to render the image on the V.E.N.O.M. screen.

*gfxSection* – The rectangle within the image data to render.

**Description**

Draws a piece of GFX at a specific location on the V.E.N.O.M. screen. The Draw function is overloaded. Each overload calls back in to the main function.

**Fill****Definition**

```
virtual void Fill(int x, int y, int width, int height, Color colour) <- Main function
```

```
virtual void Fill(Point position, Size size, Color colour) <- Overload
```

```
virtual void Fill(Rectangle rectangle, Color colour) <- Overload
```

**Parameters**

*x* – Left point of the fill rectangle.

*y* – Top point of the fill rectangle.

*width* – Width of the fill rectangle.

*height* – Height of the fill rectangle.

*colour* – The colour of the fill. Using ARGB colour space (red = 0xffff0000, green = 0xff00ff00, blue = 0xff0000ff).

*position* – The start position of the fill rectangle.

*size* – The size of the fill rectangle.

*rectangle* – The fill rectangle.

**Description**

Fills in the section of the screen specified by the parameters with the stated colour. The Fill function is overloaded. Each overload calls back in to the main function.

## **DrawText**

### **Definition**

```
virtual void DrawText(int x, int y, int size, Color colour, string text) <- Main  
function  
  
virtual void DrawText(Point position, int size, Color colour, string text) <- Overload
```

### **Parameters**

*x* – Top point of where to render the text.

*y* – Left point of where to render the text.

*size* – The size of the text in points.

*colour* – The colour of the fill. Using ARGB colour space (red = 0xffff0000, green = 0xff00ff00, blue = 0xff0000ff).

*text* – Text to render.

*position* – Where to render the text.

### **Description**

*Provides a quick and easy way to output some text on to the V.E.N.O.M. screen. The DrawText function is overloaded. Each overload calls back in to the main function.*

## ***OnAppletActive***

### **Definition**

```
virtual void OnAppletActive(int endPointId, long endPointType)
```

### **Description**

*After an applet has been opened with the OpenApplet function it must wait for the user to activate it, by selecting to view it via the V.E.N.O.M. screen. This function is called by the underlying API telling the applet that it has been selected and that it can now start rendering to the V.E.N.O.M. screen. This function calls the AppletActive event.*

## ***OnAppletInactive***

### **Definition**

```
virtual void OnAppletInactive(int endPointId)
```

### **Description**

*At any time a user can navigate away from an applet by pressing the home button on the S.T.R.I.K.E.7 command module. This function is called by the underlying API telling the applet that it is no longer visible to the user. The applet should stop trying to render to the V.E.N.O.M. screen (all render events will be dropped). The user may choose to view the applet again at some point, so this can be treated like a 'pause' event. This function calls the AppletInactive event.*

## ***OnAppletTouch***

### **Definition**

```
virtual void OnAppletTouch(int endPointId, int x, int y, bool pressed)
```

### **Parameters**

*x* – Horizontal position touched.

*y* – Vertical position touched.

*pressed* – True when the screen is touched and dragged, false when the screen is untouched.

### **Description**

Once an applet is running and active it will receive all touch events from the V.E.N.O.M. screen. This function is called every time the user touches the V.E.N.O.M. screen. This function calls the *AppletTouch* event.

## ***OnAppletReset***

### **Definition**

```
virtual void OnAppletReset(int endPointId)
```

### **Description**

*If the user unplugs the S.T.R.I.K.E.7 keyboard from the system. The API will call this function for every applet that was currently running on the V.E.N.O.M. screen. This includes all applets that have called the OpenApplet function, even if they have not been active. This function calls the AppletReset event.*



### ***Strike7API.Helpers.Gfx***

*When using the STRIKE7 API you will quickly find that you will be doing a lot of bitmap manipulation. All functions concerning bitmaps will be located in this namespace.*

### ***GetBitmapSection***

#### **Definition**

```
public static Bitmap GetBitmapSection(int x, int y, int width, int height, Bitmap image)
```

#### **Parameters**

*The parameters define a rectangular section to retrieve from the specified bitmap image.*

*x – Horizontal position of the top left part of the rectangle to get.*

*y – Vertical position of the top left part of the rectangle to get.*

*width – The width of the rectangle to get.*

*height – The height of the rectangle to get.*

*image – The Bitmap to retrieve the rectangular section from.*

#### **Returns**

*A Bitmap containing the GFX bound by the rectangle specified by the parameters.*

#### **Description**

*Gets a new Bitmap containing the GFX specified by the parameters. No range checking is done on the parameters.*

## ***CopyImage***

### **Definition**

```
public static void CopyImage(Bitmap from, Bitmap to, int destinationX, int  
desitnationY)  
  
public static void CopyImage(Bitmap from, Bitmap to, int destinationX, int  
desitnationY, bool alphaBlend)  
  
public static void CopyImage(Bitmap from, Bitmap to, int destinationX, int  
desitnationY, int fromX, int fromY, int width, int height, bool alphaBlend)
```

### **Parameters**

*From* – Where to take the image from.

*To* – Where to copy the 'from' image to.

*destinationX* – X Destination of the copy.

*destinationY* – Y Destination of the copy.

*fromX* – X Position of the copy rectangle in the from image.

*fromY* – Y Position of the copy rectangle in the from image.

*width* – Width of the image section being copied.

*height* – Height of the image section being copied.

*alphaBlend* – Apply alpha blending to the copy.

### **Description**

Copy one image on to another.