



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

*Specification v2.1*

## Contents

<i>Connection lifecycle</i> .....	2
<i>Events (Callbacks)</i> .....	3
<i>AppletActiveEvent</i> .....	4
<i>AppletInactiveEvent</i> .....	5
<i>AppletTouchEvent</i> .....	6
<i>AppletResetEvent</i> .....	7
<i>Functions (Entry points)</i> .....	8
<i>RegisterAppletActiveEvent</i> .....	9
<i>RegisterAppletInactiveEvent</i> .....	10
<i>RegisterAppletTouchEvent</i> .....	11
<i>RegisterAppletResetEvent</i> .....	12
<i>OpenApplet</i> .....	13
<i>CloseApplet</i> .....	14
<i>AppletUpdateScreen</i> .....	15
<i>AppletScroll</i> .....	16
<i>AppletDraw</i> .....	17
<i>AppletFill</i> .....	18
<i>AppletDrawText</i> .....	19

### *Connection lifecycle*

- *Client code loads the API DLL.*
- *DLL Setup. Registering for events.*
- *The Client code registers itself with the STRIKE API (OpenApplet). This includes a Name and Icon. The call will return a unique identifier for this Applet. More than one applet can be registered with the API.*
- *The client code then waits for a AppletActive event. This will include the applet Id. At this point the client code can start sending images.*
- *The applet will receive touch screen events.*
- *When the user presses the home button on the S.T.R.I.K.E.7 the applet will receive AppletInactive (Paused) event.*
- *If the Client program closes it must call the CloseApplet function.*

### *Events (Callbacks)*

- `void AppletActiveEvent( UInt16 appletId, UInt16 endPointId, UInt32 endPointType)`
- `void AppletInactiveEvent( UInt16 appletId, UInt16 endPointId )`
- `void AppletTouchEvent( UInt16 appletId, UInt16 endPointId, UInt32 x, UInt32 y, bool pressed )`
- `void AppletResetEvent( UInt16 appletId, UInt16 endPointId )`

### **AppletActiveEvent**

#### **Definition**

```
void AppletActiveEvent( UInt16 appletId, UInt32 endPointType, UInt16 endPointId)
```

#### **Parameters**

*appletId* – This is unique identifier which is used to identify an applet. The appletId is auto generated and passed the back from the API via the OpenApplet function call.

*endPointType* – This identifies what type of endpoint has made your applet active. This is the VID / PID of the device concatenated together so for example VID: 0x0738 PID: 0x1234 would equate to an endpoint type of 0x07381234.

*endPointId* – This is the same as appletId but for endpoints. End point Id 0 (Zero) is special and means broadcast to all endpoints.

#### **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.

### ***AppletInactiveEvent***

#### **Definition**

```
void AppletInactiveEvent( UInt16 appletId, UInt16 endPointId )
```

#### **Parameters**

*appletId* – This is unique identifier which is used to identify an applet. The appletId is auto generated and passed the back from the API via the OpenApplet function call.

*endPointId* – This is the same as appletId but for end points. End point Id 0 (Zero) is special and means broadcast to all endpoints.

#### **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.

## **AppletTouchEvent**

### **Definition**

```
void AppletTouchEvent( UInt16 appletId, , UInt16 endPointId, UInt32 x, UInt32 y, bool  
pressed )
```

### **Parameters**

*appletId* – This is unique identifier which is used to identify an applet. The appletId is auto generated and passed the back from the API via the OpenApplet function call.

*endPointId* – This is the same as appletId but for end points. End point Id 0 (Zero) is special and means broadcast to all endpoints.

*x* – The horizontal position of the touch event.

*y* – The vertical position of the touch event.

*pressed* – True when the touch event refers to either a finger being placed on the screen or dragged across the screen. False when the event refers to a finger being removed from the screen.

### **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.

### **AppletResetEvent**

#### **Definition**

```
void AppletResetEvent( UInt16 appletId, UInt16 endPointId )
```

#### **Parameters**

*appletId* – This is unique identifier which is used to identify an applet. The appletId is auto generated and passed the back from the API via the OpenApplet function call.

*endPointId* – This is the same as appletId but for end points. End point Id 0 (Zero) is special and means broadcast to all endpoints.

#### **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 S.T.R.I.K.E.7 disconnects and reconnects.*



### Functions (Entry points)

- void RegisterAppletActiveEvent( void ( AppletActiveEvent \* )( UInt16 ) func, UInt16 id=0 )
- void RegisterAppletInactiveEvent( void ( AppletInactiveEvent \* )( UInt16 ) func, UInt16 id=0 )
- void RegisterAppletTouchEvent( void ( AppletTouchEvent \* )( UInt16, UInt32, UInt32, bool ) func, UInt16 id=0 )
- void RegisterAppletResetEvent( void ( AppletResetEvent \* )( UInt16 ) func, UInt16 id=0 )
- UInt16 OpenApplet( const TCHAR\* string name, void \*iconData, UInt32[] supportedDevices )
- void CloseApplet( UInt16 appletId, UInt16 endPointId )
- void AppletUpdateScreen( UInt16 appletId, UInt16 endPointId )
- void AppletScroll(UInt16 appletId, , UInt16 endPointId, Int32 distanceX, Int32 distanceY, UInt32 fromX, UInt32 fromY, UInt32 width, UInt32 height)
- void AppletDraw(UInt16 appletId, UInt16 endPointId, Int32 x, Int32 y, UInt32 width, UInt32 height, void \*data)
- void AppletFill(UInt16 appletId, UInt16 endPointId, Int32 x, Int32 y, UInt32 width, UInt32 height, UInt32 colour)
- void AppletDrawText(UInt16 appletId, UInt16 endPointId, Int32 x, Int32 y, UInt32 size, UInt32 colour, TCHAR\* text)

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

## ***RegisterAppletActiveEvent***

### **Definition**

```
void RegisterAppletActiveEvent( void ( AppletActiveEvent * )( UInt16 ) func, UInt16  
appletId=0 )
```

### **Parameters**

*func* – A pointer to the function that is being registered.

*appletId* – The id of the applet that the event is being registered for. AppletId = 0 is the default and means all applets.

### **Description**

Registers the function that will be called when the AppletActiveEvent is fired.

## ***RegisterAppletInactiveEvent***

### **Definition**

```
void RegisterAppletInactiveEvent( void ( AppletInactiveEvent * )( UInt16, UInt32)  
func, UInt16 id=0 )
```

### **Parameters**

*func* – A pointer to the function that is being registered.

*appletId* – The id of the applet that the event is being registered for. AppletId = 0 is the default and means all applets.

### **Description**

Registers the function that will be called when the AppletInactiveEvent is fired.

## **RegisterAppletTouchEvent**

### **Definition**

```
void RegisterAppletTouchEvent( void ( AppletTouchEvent * )( UInt16, UInt32, UInt32,  
bool ) func, UInt16 id=0 )
```

### **Parameters**

*func* – A pointer to the function that is being registered.

*appletId* – The id of the applet that the event is being registered for. AppletId = 0 is the default and means all applets.

### **Description**

Registers the function that will be called when the AppletTouchEvent is fired.

## ***RegisterAppletResetEvent***

### **Definition**

```
void RegisterAppletResetEvent( void ( AppletResetEvent * )( UInt16 ) func, UInt16 id=0 )
```

### **Parameters**

*func* – A pointer to the function that is being registered.

*appletId* – The id of the applet that the event is being registered for. AppletId = 0 is the default and means all applets.

### **Description**

Registers the function that will be called when the AppletResetEvent is fired.

### OpenApplet

#### Definition

```
UInt16 OpenApplet( const TCHAR* string name, UInt32 iconWidth, UInt32 iconHeight, void  
*iconData, UInt32[] supportedDevices )
```

#### Parameters

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

*iconWidth* – The width of the icon.

*iconHeight* – The height of the icon.

*iconData* – Raw bitmap data for the icon.

*supportedDevices* – An array of device ids that this applet supports. This enables applets to target the devices it was developed for.

#### 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.

## **CloseApplet**

### **Definition**

```
void CloseApplet( UInt16 appletId, UInt16 endPointId )
```

### **Parameters**

*appletId* – The id of the applet that is being closed.

*endPointId* – This is the same as *appletId* but for end points. End point Id 0 (Zero) is special and means broadcast to all endpoints.

### **Description**

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

## **AppletUpdateScreen**

### **Definition**

```
void AppletUpdateScreen(UInt16 appletId, UInt16 endPointId)
```

### **Parameters**

*appletId* – The id of the applet that this command is for.

*endPointId* – This is the same as *appletId* but for end points. End point Id 0 (Zero) is special and means broadcast to all endpoints.

### **Description**

Updates the screen with the latest GFX operations.

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 V.E.N.O.M. screen is updated. This is essentially double buffering.



## AppletScroll

### Definition

```
void AppletScroll(UInt16 appletId, UInt16 endPointId, UInt32 fromX, UInt32 fromY,  
UInt32 width, UInt32 height, Int32 distanceX, Int32 distanceY)
```

### Parameters

*appletId* – The id of the applet that this command is for.

*endPointId* – This is the same as *appletId* but for end points. End point Id 0 (Zero) is special and means broadcast to all endpoints.

*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.

### Description

Scrolls a rectangular section of the screen to another location on the V.E.N.O.M. screen.

**NOTE: This is not a move function. The scroll happens within the specified rectangle.**

## **AppletDraw**

### **Definition**

```
void AppletDraw(UInt16 appletId, UInt16 endPointId, Int32 x, Int32 y, UInt32 width,
UInt32 height, void *data)
```

### **Parameters**

*appletId* – The id of the applet

*endPointId* – This is the same as *appletId* but for end points. End point Id 0 (Zero) is special and means broadcast to all endpoints.

*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.

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

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

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

### **Description**

*Draws a piece of GFX at a specific location on the V.E.N.O.M. screen.*

## AppletFill

### Definition

```
void Fill(UInt16 appletId, UInt16 endPointId, Int32 x, Int32 y, UInt32 width, UInt32 height, UInt32 colour)
```

### Parameters

*appletId* – The id of the applet

*endPointId* – This is the same as *appletId* but for end points. End point Id 0 (Zero) is special and means broadcast to all endpoints.

*x* – Left point of the fill rectangle.

*y* – Top point of the fill rectangle.

*width* – The width of the fill rectangle.

*height* – The height of the fill rectangle.

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

### Description

Fills in the section of the screen specified by the parameters with the stated colour.

## **AppletDrawText**

### **Definition**

```
void AppletDrawText(UInt16 appletId, UInt16 endPointId, Int32 x, Int32 y, UInt32 size, UInt32 colour, TCHAR* text)
```

### **Parameters**

*appletId* – The id of the applet

*endPointId* – This is the same as *appletId* but for end points. End point Id 0 (Zero) is special and means broadcast to all endpoints.

*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* – The text to render.

### **Description**

Provides a quick and easy way to output some text on to the V.E.N.O.M. screen.