

# Cintel ASL 2.0 tutorial

Copyright© 2001 Cintel Software

Last updated: 20/2/01

For more information please visit [www.cintelsoftware.co.uk/asl.htm](http://www.cintelsoftware.co.uk/asl.htm)

## Introduction

Thank you for your interest in ASL, Cintel's proprietary animation scripting technology. This tutorial applies to both standalone ASL scripts, and also those created for use with ASL WebScript. ASL is a simple to learn, yet powerful scripting language that allows you to design presentations that can be run as an introduction or information source for applications, for entertainment, or embedded to run when someone visits a web page on the Internet.

## Basics

An animation consists of two parts; the script file and its support files, which may include pictures, sounds or other files. The script tells a program called the ASL Runtime Engine what to do and which files to load. The basic structure for an ASL script file is as follows:

### Runtime Engine Version

**Start ASL**

**Code...**

**End ASL**

The **Code** is a set of instructions that tells the Runtime Engine (or "compiler") what files it should load, and in what order operations should be carried-out. It resides between a start instruction '**Start ASL**', and an end instruction '**End ASL**', which tell the Runtime Engine, simply when to start and stop reading the script. The '**End ASL**' instruction is optional, and tells the Runtime Engine to stop processing the script and to shut down, even if there are instructions after it, so it is important that you place it at the very end of your script. If it is omitted, the animation will remain open after it has finished running and must be close by the user.

The **Runtime Engine Version** is a number specifying the version of the Runtime Engine that the script was designed to run on, and must be the first line in the script. Always make sure you design and test your scripts with the latest build of the Runtime Engine or your scripts may not work properly since features may be altered in newer versions of the Runtime Engine. The latest Runtime Engine and its version can always be found on the Cintel ASL website (see top of the page).

## Instructions

An instruction is a structured statement that tells the Runtime Engine what to do. This structure is the same for all instructions in ASL, to make them easy to remember and use. An instruction consists of two main parts; the **Function**, which tells the compiler what to do to the second part of the instruction: the **Object**. The final part of an instruction depends on the function. If the function requires any other information, such as text to display, it is specified in the **Parameters** section of the instruction. A function can require up to 5 parameters, however, most only require one or two.

**Function;Object;Parameter 1;Parameter 2;**

As you can see from above, the separate parts of the instruction are separated by semicolons ';' with one placed at the end of the instruction to tell the Runtime Engine where the instruction stops. There are no spaces between the parts of the instruction, for example after semicolons.

## Making animations

In order to make ASL animations, you will need to use a combination of instructions – some which set properties of objects (such as setting the picture to be displayed), and others which carry-out functions on them (for example, scrolling text around the screen). In the following example a text object is set-up and then scrolled from left to right.

```
Text;Lbl03;Hello world!;
Scroll;Lbl03;60;-230;
```

The text object above would start at the default position of 0 pixels across and 0 pixels down, since we did not specify these properties. In order to make the text object more central, we might place the following lines of code before the Scroll statement:

```
HPos;Lbl03;100;
VPos;Lbl03;75;
```

The order of the instructions is not important – just bear in mind that only properties set **before** the Scroll statement will be applied. It's like reading a book; the author gives you the necessary information about a character before something happens to them – but you're not aware of what will happen to them later because you haven't got there yet.

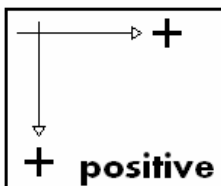
## Language reference

There are four main objects in ASL; the SAL (shell-activated link) object for launching files and internet locations, the 'Pic' object, for displaying images, the 'Lbl' object for printing text on the window, and the 'Window' object itself, which is the visible environment created by the Runtime Engine in which the script is run. Most functions in ASL are 'Object-dependant,' which means that they are all used with reference to an object, and may need specific properties of that object to be set in order to be executed. Each of the three objects has its own functions listed below, some of which are common to more than one object.

SAL	Pic	Lbl	Window
Object	HPos, VPos	HPos, VPos	HPos, VPos
Launch	Width, Height	HMove, VMove	Width, Height
Number of	HMove, VMove	HScroll, VScroll	Focus
	HScroll, VScroll	Number of	Colour
	Number of	Flash	Title
	Flash	Show, Hide	
	Show, Hide	Link	
	Link	Text	
	Image	Font	

## Function descriptions

The above-mentioned functions are described in more detail below, with the relative syntax and some examples. Please note; when specifying the direction of change (+/-) the number increases *towards* the bottom-right of the window. Also, when specifying a position in pixels, the value can be positive or negative, but **must** be an integer (whole number).



```
HPos;Object;Position;
VPos;Object;Position;
```

<p>Sets the horizontal and vertical position of an object. The parameter required is the position (in pixels) on the window that the object will be placed at. For example, the following code would set the horizontal position of the object 'Pic01' to 100 pixels on the window.</p>
<pre><b>HPos;Pic01;100;</b></pre>
<pre><b>HMove;Object;Change;</b> <b>VMove;Object;Change;</b></pre>
<p>Causes horizontal and vertical movement of an object without reference to a specific position on the window. You don't specify the position to which the object moves; instead you provide the direction (+/-) and amount (in pixels) by which it does so. For example, the following code would cause the object 'Lbl01' to vertically move 200 pixels upwards (negative).</p>
<pre><b>VMove;Lbl01;-200;</b></pre>
<pre><b>HScroll;Object;Speed;Change;</b> <b>VScroll;Object;Speed;Change;</b></pre>
<p>Scrolls an object either horizontally or vertically. You must provide two parameters; the speed of scrolling (in arbitrary units), and the direction (+/-) and amount (in pixels) of change. For example, the following code would horizontally scroll the object 'Pic02' 150 pixels to the right (positive) at speed 60.</p>
<pre><b>HScroll;Pic02;60;150;</b></pre>
<pre><b>Width;Object;Size;</b> <b>Height; Object; Size;</b></pre>
<p>Sets the width and height dimensions of an object, measured in pixels. For example, the following code would set the width of the Pic object, 'Pic01' to 150 pixels.</p>
<pre><b>Width;Pic01;150;</b></pre>
<pre><b>Show;Object;</b> <b>Hide;Object;</b></pre>
<p>Shows or hides an object from view. No parameters are required by this function. If the object is not already in the designated state when the function is called, it will be changed so it is so. For example, the following code makes the object 'Lbl02' visible.</p>
<pre><b>Show;Lbl02;</b></pre>
<pre><b>Flash;Object;Cycles;Interval;</b></pre>
<p>Causes an object to flash for a designated number of cycles at a set interval (in ms). Two parameters are required; the number of show/hide cycles, and the interval between each state-change. For example, the following code makes the object 'Pic01' flash 3 times, changing its state every 500ms (half a second).</p>
<pre><b>Flash;Pic01;3;500;</b></pre>
<pre><b>Object;SAL Object;Link;</b></pre>
<p>Creates a link to a file, folder or internet location inside a SAL object. When the SAL object is activated, the file, folder, or internet address is launched. For example, the following code would link the object 'SAL03' to the specified internet address.</p>
<pre><b>Object;SAL03;http://www.cintelsoftware.co.uk/;</b></pre>
<pre><b>Link;Object;SAL Object;</b></pre>
<p>Links a Pic or Lbl object to a SAL object, so that the SAL object will be activated if the Pic or Lbl object is clicked. For example, the following code would link the object 'Lbl01' to the SAL object 'SAL02,' causing it to be activated if 'Lbl01' is clicked on.</p>
<pre><b>Link;Lbl01;SAL02;</b></pre>
<pre><b>Launch;SAL Object;</b></pre>

Activates a SAL object, and launches its link. For example, the following code would cause the object 'SAL01' to be launched.
<b>Launch;SAL01;</b>
<b>Image;Pic Object;Location;</b>
Sets the image that will be displayed by a Pic object. For example, the following code would cause the image 'Cintel.jpg' to be displayed by the Pic object 'Pic02.'
<b>Image;Pic02;Cintel.jpg;</b>
<b>Text;Lbl Object;Text;</b>
Sets the text that will be displayed by a Lbl object. For example, the following code would cause the specified text to be displayed by the Lbl object 'Lbl01.'
<b>Text;Lbl01;This is some text;</b>
<b>Font;Lbl Object;Name;Size;Colour;</b>
Sets the font name, size and colour in which the text will be displayed by a Lbl object. For the font colour, the choices are Black, White, Red, Blue, Green or Yellow. For example, the following code would cause the Lbl object 'Lbl02' to display its text in red, in the font 'Arial,' size 12.
<b>Font;Lbl02;Arial;12;Red;</b>
<b>Focus;Window;</b>
Sets the Runtime Engine window has the uppermost window on the screen. It will remain in the foreground until the animation ends. The syntax for the instruction is always the same, since there is only one Window object per script.
<b>Colour;Window;Colour;</b>
Sets the background colour of the window, on top of which the images and text will be rendered. The colour choices are Black, White, Red, Blue, Green and Yellow. If no colour is specified, the default, white is used. The following code would set the background colour of the window to Blue.
<b>Colour;Window;Blue;</b>
<b>Title;Window;Text;</b>
Sets the text that will appear in the title-bar of the window. For example, the following code would cause the specified text to be displayed in the title bar of the window.
<b>Title;Window;ASL script demo;</b>

### Object-independent instructions

Some instructions in ASL are independent of objects, and therefore, since they do not rely on the state of any other objects, they can be invoked at any time between the 'Start ASL' and 'End ASL' statements. These two statements themselves are examples of such instructions. Descriptions and examples of these are given in the table below. They exist in the following format:

**Function;Parameters;**

<b>Wait;Time;</b>
Suspends execution of the script for a designated amount of time, measured in milliseconds. This instruction is used to cause a pause during the execution of the script. For example, the following code would cause a pause of 2000ms (2 seconds).
<b>Wait;2000;</b>
<b>Info;Text;</b> <b>Warning;Text;</b>

<b>Message;Text;</b>
Displays a plain, warning or information message box containing text, and suspends execution until the user clicks OK. For example, the following code would show an information message box, displaying the specified text.
<b>Info;This is an example;</b>
<b>Debug;</b>
Executes the script in Debug Mode, useful for checking for incorrect values or errors that would be otherwise ignored. The processing of every instruction as it is executed and any subsequent errors are shown in a side window, the contents of which are placed in a text file called "Debug.txt" for later reference, in the script directory when it finishes.
<b>Exec Speed;Time;</b>
Sets the speed (1 - 10) at which the execution engine will run. If this instruction is not provided in a script, the default speed of 10 is used. Please note, the slower the computer, the higher the speed should be. Whilst the option to set the speed higher than 10 is available, causing the execution engine to run faster than it was designed to may have undesired results.
<b>Exec Speed;8;</b>
<b>Version;RTE Version;</b>
Specifies the Runtime Engine version for which this script was intended to run on. If the number is higher than the current version of the engine installed, the user will be asked whether they wish to run the script.
<b>Version;300;</b>

### Special characters

Apart from the widely used semicolon ";" character which is used in every instruction, there are two other special characters used in ASL. The use of the single inverted comma, "'" denotes that a line be treated as a comment in a script, and ignored. In other words, placing the inverted comma at the start of a line will cause the Runtime Engine to skip over it as if it didn't exist. This is useful when debugging, where you may want to remove a line, without deleting the code, or simply for placing comments or notes in a script.

The other special character is the "&" ampersand sign, which can be used with certain instructions to cause their simultaneous execution, for example, scrolling or flashing a Pic object and a Lbl object at the same time. Execution will not continue until both have completed their respective animations. There is no limit on the number of instructions that can be 'daisy-chained' in this manner, but keep in mind, the more animation at any one time, the larger the overall performance hit will be. Only instructions of the same type can be simultaneously executed, otherwise timing errors can occur. The option to daisy-chain different animation effects is a feature, which will be implemented in the near future. An example of a daisy-chained instruction is show below. The code would cause the object 'Pic01' to move to the right while 'Lbl03' moves to the left.

```
Scroll;Pic01;40;120;&  
Scroll;Lbl03;60;-230;
```

More instructions could be added, with the ampersand sign placed at the end of every instruction but the last one.

## Example code

A sample script, which displays and flashes an image is shown below. The code would cause the image "Example.jpg" placed in the same folder as the script to be displayed at 30, 30 pixels in the window, and then flashed 3 times. As usual, the first line specifies the Runtime Engine version, and execution of the script begins with the **Start ASL** instruction after the properties are set.

```
Version;262;  
Number of Pics;1;
```

```
HPos;Pic01;30;  
VPos;Pic01;30;  
Width;Pic01;230;  
Height;Pic01;103;  
Image;Pic01;Example.jpg;
```

**Start ASL**

```
Show;Pic01;  
Flash;Pic01;3;500;
```