

SLM Software

SLM_SDK_2.1 Help Document

Upgrade Notes and User Guide

- SLM_SDK 2.1 supports loading and displaying 8-bit grayscale and 24-bit color images.
- SLM_SDK 2.1 provides a stable 20 Hz image refresh interface.
- SLM_SDK 2.1 adds the Timeout function group for image output in loop code during development. Timeout_ShowImageFromFilePath can display a single image and allows the desired code to be executed after the image is displayed.
- Before calling the functions, check the display scaling and resolution settings of the computer primary display and the SLM display (see Section 2).

CONTENTS

1 Function Calls	1
1.1 Timeout_CreateWindow	1
1.2 Timeout_ShowWindow	2
1.3 Timeout_ShowImageFromFilePath	3
1.4 Timeout_CloseWindow	4
1.5 saShowImageFromFilePath	5
1.6 saShowImageFromFolder	6
1.7 saShowImageFormFolderPath	7
1.8 saShowImageFromSelector	8
1.9 saShowBlock	9
1.10 saShowWindow	10
1.11 saCloseWindow	11
1.12 saPauseShow	12
1.13 saResumeShow	13
1.14 saStopShow	14
2 Precautions	15
3 Demo Instructions	16

1 Function Calls

1.1 Timeout_CreateWindow

`void Timeout_CreateWindow()`

Example Call (C code):

`Timeout_CreateWindow ()`

Description:

Calling this function creates a display window with delay attributes. This function does not display a window on the desktop. However, when `Timeout_ShowImageFromFilePath` is called for the first time, this window must already exist. To show the window, call `Timeout_ShowWindow(true)`.

The window created by this function (the Timeout function group) and the window created by the sa function group are independent of each other. Therefore, if a Timeout window is created by this function, use the Timeout-series window management functions. If images are displayed by the sa function group, use the sa-series window management functions.

1.2 Timeout_ShowWindow

`void Timeout_ShowWindow(bool isShow)` `// isShow: whether to show the window`

Example Call (C code):

```
Timeout_ShowWindow (true);
```

Description:

This function is a window management function in the Timeout function group. Calling this function shows a window with delay attributes. If the window exists, it is displayed. If the window does not exist, the function has no effect. Before displaying images by calling `Timeout_ShowImageFromFilePath`, make sure that this window exists and has already been shown.

Example Call (C code):

```
Timeout_ShowWindow (false);
```

Description:

This function is a window management function in the Timeout function group. Calling this function hides a window with delay attributes. If the window exists, it is hidden. If the window does not exist or has already been hidden, the function has no effect.

1.3 Timeout_ShowImageFromFilePath

```
void Timeout_ShowImageFromFilePath (  
char* szPath,                // image file path  
bool IsStretch,              // whether to stretch the image. true: stretch the image to fill the  
                             // window; false: do not stretch the image and center it in the window  
int DesX,                    // x-coordinate of the display-area origin (screen coordinate system)  
                             // (unit: pixel)  
int DesY,                    // y-coordinate of the display-area origin (screen coordinate system)  
                             // (unit: pixel)  
int DesWidth,                // display-area width (unit: pixel)  
int DesHeight,               // display-area height (unit: pixel)  
bool IsRGB,                  // when IsRGB is true, a color image is displayed  
int uMill)                   //image display duration
```

Example Call (C code):

```
Timeout_ShowImageFromFilePath ("123.jpg", false, 0, 0, 1920, 1080, true, 4000);
```

Description:

This function is a display function in the Timeout function group. Calling this function displays one image in the delayed display window. The uMill parameter specifies the hold time of the image. If only one image is displayed, or if the image is the last one, the image remains displayed until the window is closed. If images are displayed in a for loop, the next image is displayed after the uMill time has elapsed.

Note: Before calling this function, call Timeout_CreateWindow and Timeout_ShowWindow first to create and display the window environment.

Note: To display images in loop code by calling a function, use this function first instead of saShowImageFromFilePath.

1.4 Timeout_CloseWindow

`void Timeout_CloseWindow()`

Example Call (C code):

```
Timeout_CloseWindow ();
```

Description:

This function is a window control function in the Timeout function group. Calling this function closes the Timeout display window.

Note: Call `Timeout_CloseWindow` before the main window exits or before the process terminates. If `Timeout_CloseWindow` is not called, any `Timeout_ShowImageFromFilePath` operation that is still in a loop will not end immediately; the image output will stop only after the loop exits.

1.5 saShowImageFromFilePath

```
bool saShowImageFromFilePath (  
const char* szPath,           // path of the image file (UTF-8 encoded string)  
bool IsStretch,              // whether to stretch the image. true: stretch the image to fill the  
                             // window; false: do not stretch the image and center it in the window  
int DesX,                    // x-coordinate of the display-area origin (screen coordinate  
system) (unit: pixel)  
int DesY,                    // y-coordinate of the display-area origin (screen coordinate  
system) (unit: pixel)  
int DesWidth,                // display-area width (unit: pixel)  
int DesHeight,               // display-area height (unit: pixel)  
bool IsRGB)                  // when IsRGB is true, a color image is displayed
```

Example Call (C code):

```
saShowImageFromFilePath ("D:\\Images\\test.jpg", 0, 1920, 0, 1920, 1080, true);
```

Description: This function directly reads and displays the image file specified by szPath. The function displays the image within the rectangular area defined by the points {DesX, DesY}, {DesX + DesWidth, DesY}, {DesX, DesY + DesHeight}, and {DesX + DesWidth, DesY + DesHeight}. If IsStretch is false (no stretching), an image whose resolution is larger than the display area (DesWidth x DesHeight) will be cropped by the rectangular area. If the image resolution is smaller than the display area, the image will be centered in the rectangular area. If IsRGB is true, the image is displayed in its original colors. If IsRGB is false, the image is displayed in grayscale. Supported image formats typically include bmp, jpg, png, and tif.

Notes:

Non-stretch mode preserves the original image fidelity without distortion. In stretch mode, the image fills the specified display area regardless of its resolution. Excessive stretching or compression may reduce image sharpness and cause distortion.

Do not load more than 3,000 images at a time. Otherwise, system resources may be exhausted and the program may crash.

Do not mix non-image files with image files in the same folder. Otherwise, unexpected errors may occur.

This function does not read subfolders. Therefore, the images to be displayed should be placed directly in the selected folder, not in its subfolders.

1.6 saShowImageFromFolder

```
void saShowImageFromFolder (  
bool IsStretch,           // whether to stretch the image. true: stretch the image to fill the  
                           window; false: do not stretch the image and center it in the window  
int DesX,                 // x-coordinate of the display-area origin (screen coordinate system)  
                           (unit: pixel)  
int DesY,                 // y-coordinate of the display-area origin (screen coordinate system)  
                           (unit: pixel)  
int DesWidth,             // display-area width (unit: pixel)  
int DesHeight             // display-area height (unit: pixel)  
int uMill,                // playback interval (unit: ms)  
bool IsRGB)               // when IsRGB is true, a color image is displayed
```

Example Call (C code):

```
saShowImageFromFolder (0,1920,0,1920,1080,500,true);
```

Description:

Calling this function opens a folder selector. After a folder is selected and confirmed, the function displays all images in the folder in non-stretch mode within the rectangular area defined by the points {1920, 0}, {3840, 0}, {1920, 1080}, and {3840, 1080}, and switches to the next image every 500 ms. If IsRGB is true, the image is displayed in its original colors. If IsRGB is false, the image is displayed in grayscale. If the image resolution is larger than 1920 x 1080, the image will be cropped by the rectangular area. If the image resolution is smaller than 1920 x 1080, the image will be centered in the rectangular area. Supported image formats include bmp, jpg, png, and tif.

Notes:

Non-stretch mode preserves the original image fidelity without distortion. In stretch mode, the image fills the specified display area regardless of its resolution. Excessive stretching or compression may reduce image sharpness and cause distortion.

Do not load more than 3,000 images at a time. Otherwise, system resources may be exhausted and the program may crash.

Do not mix non-image files with image files in the same folder. Otherwise, unexpected errors may occur.

This function does not read subfolders. Therefore, the images to be displayed should be placed directly in the selected folder, not in its subfolders.

1.7 saShowImageFromFolderPath

```
void saShowImageFromFolderPath (  
char* FolderPath,           // folder path  
bool IsStretch,             // whether to stretch the image. true: stretch the image to fill the  
                             // window; false: do not stretch the image and center it in the window  
int DesX,                   // x-coordinate of the display-area origin (screen coordinate system)  
                             // (unit: pixel)  
int DesY,                   // y-coordinate of the display-area origin (screen coordinate system)  
                             // (unit: pixel)  
int DesWidth,               // display-area width (unit: pixel)  
int DesHeight               // display-area height (unit: pixel)  
int uMill,                  // playback interval (unit: ms)  
bool IsRGB)                 // when IsRGB is true, a color image is displayed
```

Example Call (C code):

```
saShowImageFromFolderPath ("c:\\picture",0,1920,0,1920,1080,500,true);
```

Description:

This function is similar to `saShowImageFromFolder`, except that it does not open a folder selector. Instead, it parses image files by using the specified folder path parameter. The function displays all images in the folder in non-stretch mode within the rectangular area defined by the points {1920, 0}, {3840, 0}, {1920, 1080}, and {3840, 1080}, and switches to the next image every 500 ms. If `IsRGB` is true, the image is displayed in its original colors. If `IsRGB` is false, the image is displayed in grayscale. If the image resolution is larger than 1920 x 1080, the image will be cropped by the rectangular area. If the image resolution is smaller than 1920 x 1080, the image will be centered in the rectangular area. Supported image formats include bmp, jpg, png, and tif.

Notes:

Non-stretch mode preserves the original image fidelity without distortion. In stretch mode, the image fills the specified display area regardless of its resolution. Excessive stretching or compression may reduce image sharpness and cause distortion.

Do not load more than 3,000 images at a time. Otherwise, system resources may be exhausted and the program may crash.

Do not mix non-image files with image files in the same folder. Otherwise, unexpected errors may occur.

This function does not read subfolders. Therefore, the images to be displayed should be placed directly in the selected folder, not in its subfolders.

1.8 saShowImageFromSelector

```
void saShowImageFromSelector (  
bool IsStretch,           // whether to stretch the image. true: stretch the image to fill the  
                           // window; false: do not stretch the image and center it in the window  
int DesX,                 // x-coordinate of the display-area origin (screen coordinate system)  
                           // (unit: pixel)  
int DesY,                 // y-coordinate of the display-area origin (screen coordinate system)  
                           // (unit: pixel)  
int DesWidth,             // display-area width (unit: pixel)  
int DesHeight             // display-area height (unit: pixel)  
int uMill,                //播放间隔 (单位: ms)  
bool IsRGB)               // when IsRGB is true, a color image is displayed
```

Example Call (C code):

```
saShowImageFromSelector (0,1920,0,1920,1080,500,true);
```

Description:

Calling this function opens a file filter dialog. The desired images can be selected freely by left-clicking while holding Ctrl or Shift. After clicking OK, the function displays the selected images in non-stretch mode within the rectangular area defined by the points {1920, 0}, {3840, 0}, {1920, 1080}, and {3840, 1080}, and switches to the next image every 500 ms. If IsRGB is true, the image is displayed in its original colors. If IsRGB is false, the image is displayed in grayscale. If the image resolution is larger than 1920 x 1080, the image will be cropped by the rectangular area. If the image resolution is smaller than 1920 x 1080, the image will be centered in the rectangular area. Supported image formats include bmp, jpg, png, and tif.

Notes:

Non-stretch mode preserves the original image fidelity without distortion. In stretch mode, the image fills the specified display area regardless of its resolution. Excessive stretching or compression may reduce image sharpness and cause distortion.

Do not load more than 3,000 images at a time. Otherwise, system resources may be exhausted and the program may crash.

Do not select non-image files. Otherwise, unexpected errors may occur.

1.9 saShowBlock

```
void saShowBlock (  
double* data,           // double-type array  
unsigned int data_cols, // width of the data block  
unsigned int data_rows, // height of the data block  
int DesX,               // x-coordinate of the display-area origin (screen coordinate system)  
                        (unit: pixel)  
int DesY,               // y-coordinate of the display-area origin (screen coordinate system)  
                        (unit: pixel)  
int DesWidth,           // display-area width (unit: pixel)  
int DesHeight,          // display-area height (unit: pixel)  
bool IsRGB)             // when IsRGB is true, a color image is displayed
```

Example Call (C code):

```
double* data = new double[500*300];  
for(int i=0;i<500*300;i++)  
    data[i]=255;  
saShowBlock (data,500,300,0 ,0,600 ,400,false);  
delete data;
```



Execution Result

Description:

This function sends a 500 x 300 data block to the display area {0, 0}, {600, 0}, {0, 400}, and {600, 400}. In the figure above, the white area is the constructed data block, and the black area is the display area specified by the function parameters 0, 0, 600, 400.

This function can be used to present any constructed data as an image.

1.10 saShowWindow

```
void saShowWindow (  
bool IsShow)           // Boolean parameter. true: show; false: hide
```

Example Call (C code):

```
saShowWindow(1);        // if the display area has been constructed, show this display area  
saShowWindow(0);        // if the display area has been constructed, hide this display area
```

Note:

This function is a **display-area control function**.

The display area can only be constructed by the following five functions: `saShowImageFromFilePath_Timeout`, `saShowBlock`, `saShowImageFromFolderPath`, `saShowImageFromFolder`, and `saShowImageFromSelector`. If the display area has not been constructed or has been released (see `saCloseWindow`), this function has no effect.

1.11 saCloseWindow

`void saCloseWindow ()`

Example Call (C code):

```
saCloseWindow();           // if the display area has been constructed, release this display area
```

Note:

This function is a **display-area control function.**

The display area can only be constructed by the following five functions: `saShowImageFromFilePath`, `saShowBlock`, `saShowImageFromFolderPath`, `saShowImageFromFolder`, and `saShowImageFromSelector`. If the display area has not been constructed or has been released, this function has no effect.

1.12 saPauseShow

`void saPauseShow ()`

Example Call (C code):

```
saPauseShow();           // pause a continuous image playback process
```

Note:

This function is a **playback control function**.

If a continuous playback process is created by `saShowImageFromFolderPath`, `saShowImageFromFolder`, or `saShowImageFromSelector`, `saPauseShow` can be used to pause the playback process. If no playback process is currently running, this function has no effect.

1.13 saResumeShow

void saResumeShow ()

Example Call (C code):

```
saResumeShow();           // resume a process paused by saPauseShow, or restart playback from the  
beginning if the image content has not been changed
```

Note:

This function is a **playback control function**.

If a continuous playback process is created by saShowImageFromFolderPath, saShowImageFromFolder, or saShowImageFromSelector and has been paused by saPauseShow, ResumeShow resumes playback from the interrupted position.

If playback was first started by saShowImageFromFolderPath, saShowImageFromFolder, or saShowImageFromSelector, and no new images have been loaded by these three functions after playback finishes, saResumeShow can be called directly to play from the beginning again. This is a cache optimization function for loading a large number of images, because loading a large image set can be time-consuming.

If playback was first started by saShowImageFromFolderPath, saShowImageFromFolder, or saShowImageFromSelector and was stopped by saStopShow, calling saResumeShow will restart playback from the beginning.

1.14 saStopShow

`void saStopShow ()`

Example Call (C code):

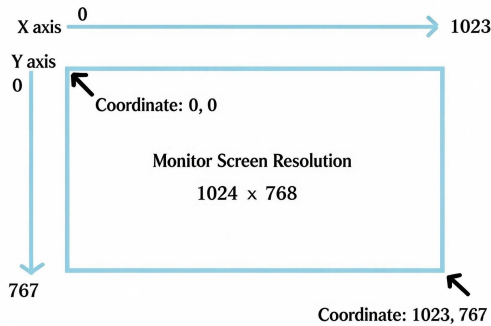
```
saStopShow();           // stop playback
```

Stops an active playback process.

2 Precautions

To ensure that phase patterns loaded to the modulator through the SDK are displayed correctly, configure the display resolution after connecting the modulator.

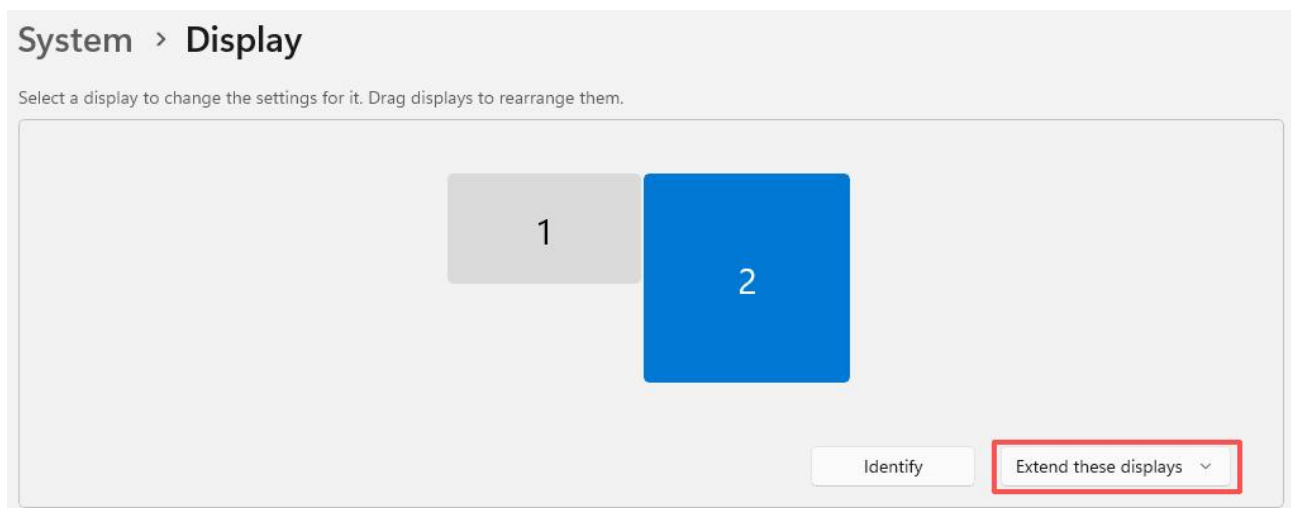
The screen coordinate system and the modulator screen settings on the computer are illustrated below.

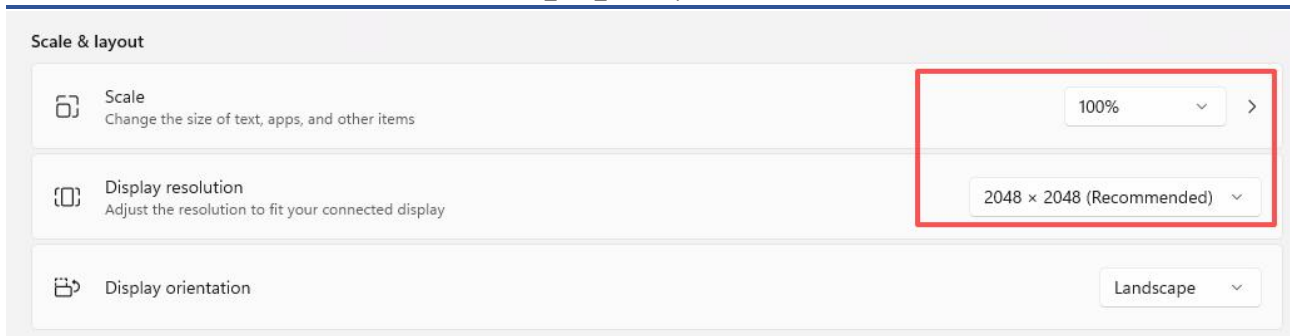


After the modulator is connected and powered on, right-click the desktop and select <Display settings> to check the settings of the computer primary display and the SLM display.

To ensure that the image is completely loaded to the modulator, verify the following:

- The secondary display is set to extended display.
- The primary display resolution is set to 1920 x 1080, and the modulator resolution is set to its corresponding resolution.
- The scaling ratios of the primary display and the modulator are both set to 100%.





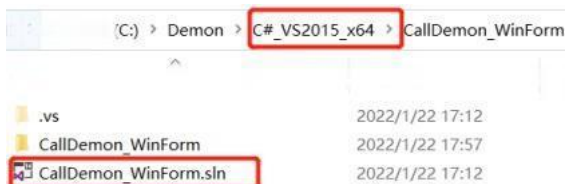
According to the setup steps above, if the resolution of Display 1 on the computer is 1920 x 1080 and the modulator resolution is 2048 x 2048, then to create a display area that fully covers the modulator screen, the function parameters should be 1920, 0, 2048, 2048.

3 Demo Instructions

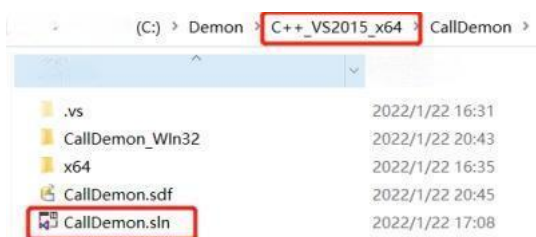
The Demo folder contains the following six demos:

C#_VS2015_x64	2022/1/22 17:12
C++_VS2015_x64	2022/1/22 16:31
LabView_2015_x32	2022/1/21 16:48
Matlab_2016a_x64	2022/1/22 17:10
python_VS2015_x64	2022/1/23 0:17
VB.net_VS2015_x64	2022/1/22 15:30

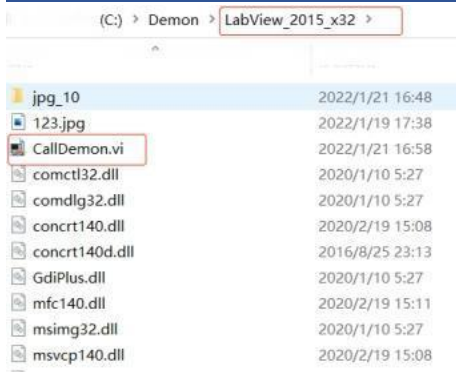
1. For C#, open CallDemon_WinForm.sln and run it.



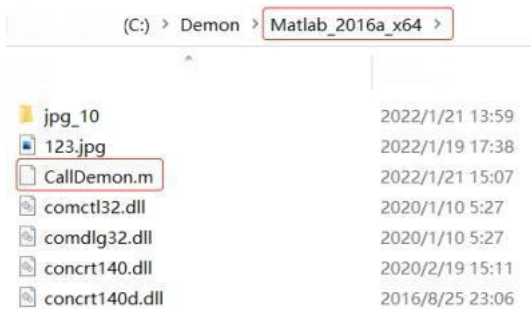
2. For C++, open CallDemon.sln and run it.



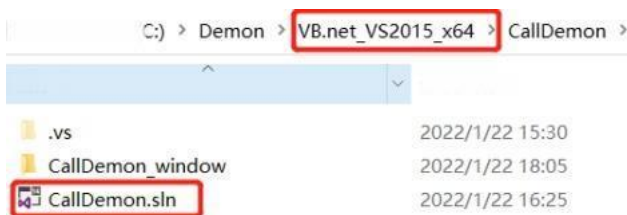
3. For LabVIEW, open CallDemon.vi and run it.



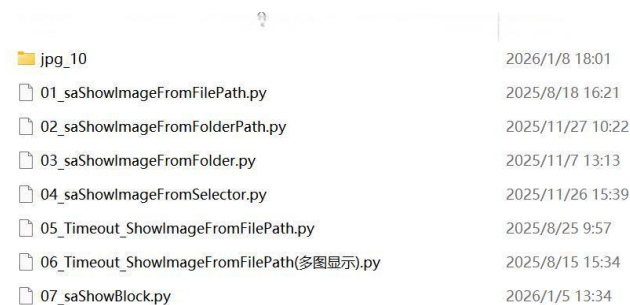
4. For MATLAB, open the CallDemon.m file. When prompted by MATLAB, click Change Folder to run it.

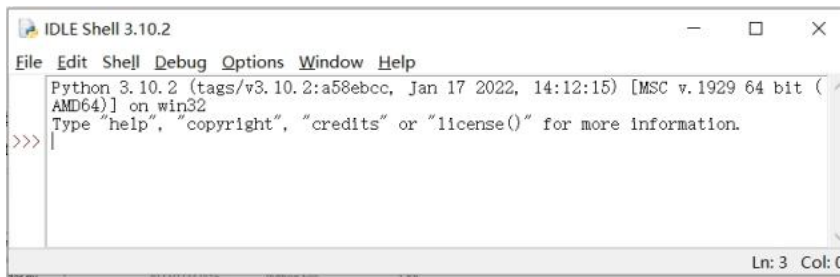


5. For VB, open CallDemon.Sln and run it.



6. For Python, only seven .py files are provided. Because the SDK functions are designed for concurrent execution, directly running the .py files will not produce the expected result. Open Python IDLE or the Python command-line tool, paste the code from the .py files, and run it there.





Python IDLE Tool