



Introduction to Intel[®] Software Guard Extensions (Intel[®] SGX)

April 18, 2017

Why Use Intel® SGX?

Many applications have secrets that need to be protected.

Examples of secrets include:



Healthcare records



Personally identifiable information (PII)



Biometric factors and templates



Passwords



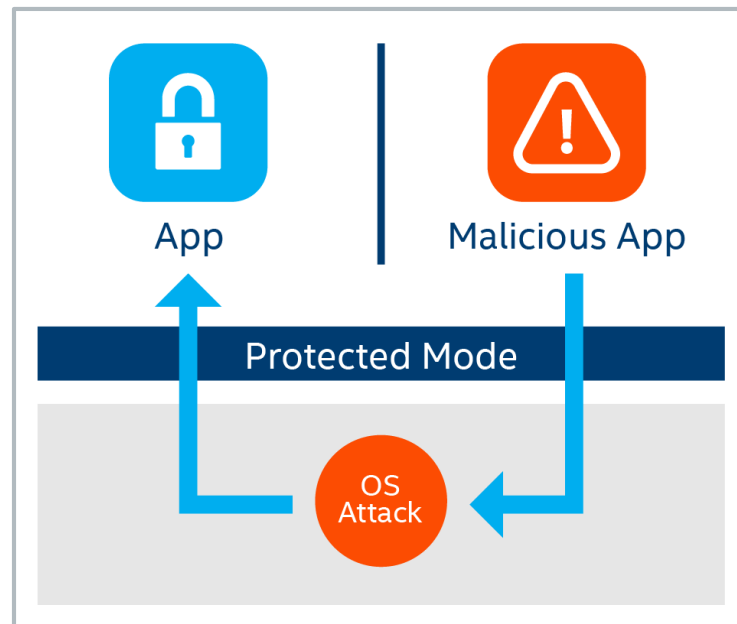
Encryption keys



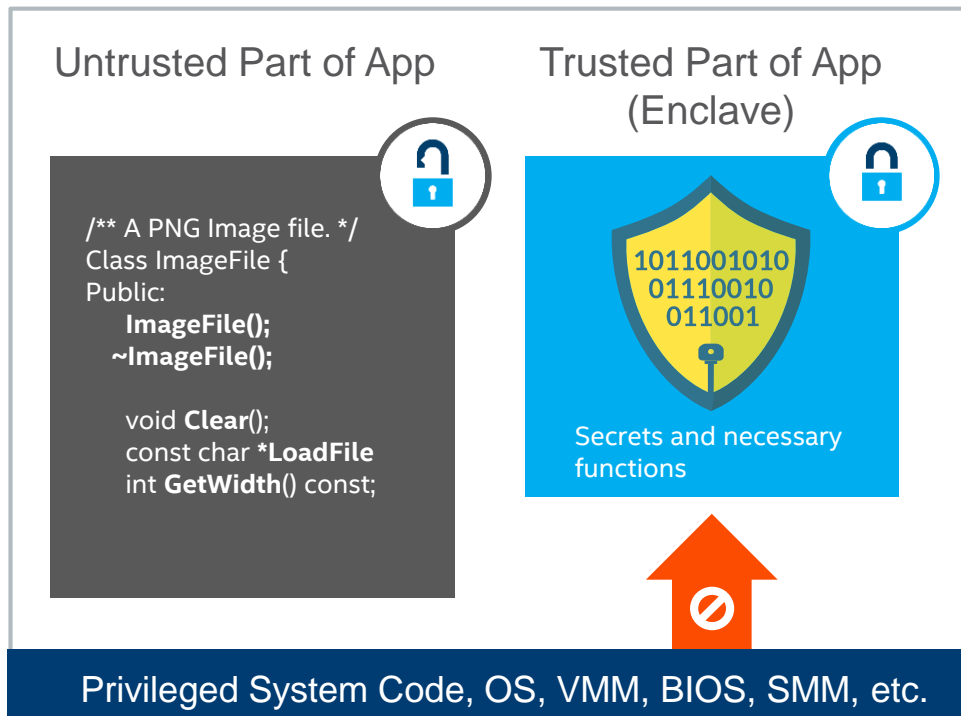
Intellectual property (IP)

What Is Intel® SGX?

- A Trusted Execution Environment from Intel for applications
- Trusted hardware: the CPU die
- Isolates a portion of physical memory to protect select code and data from view or modification
- In Intel SGX, these isolated portions of memory are called “enclaves”



Trusted and Untrusted Partitions

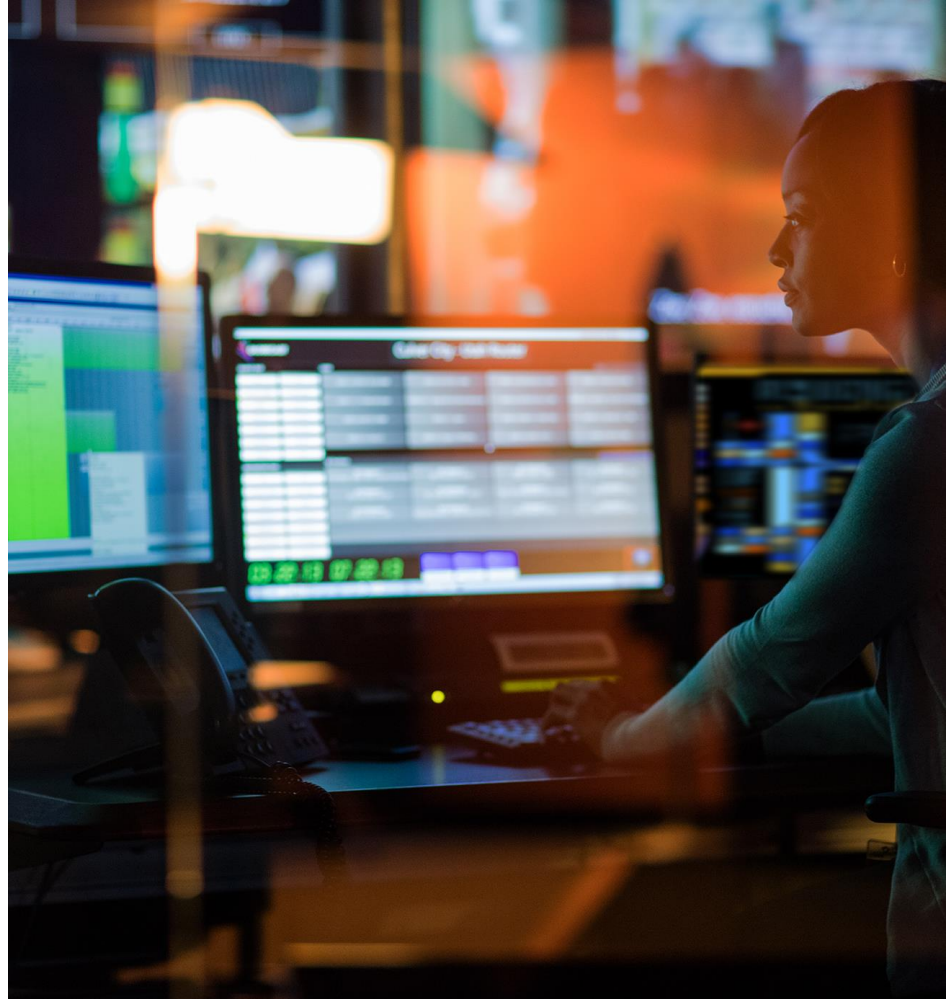


1. An Intel® SGX-enabled app is built with trusted and untrusted parts.
2. Best practice: the trusted partition should contain just the minimum amount of content required to protect your secrets.

What Intel® SGX Can't Protect You From

- Buffer-overflow attacks
- Side-channel attacks
- Untrusted inputs into the enclave

There is no substitute for good, secure programming practices



Intel® SGX Design Imperatives



Protect sensitive data



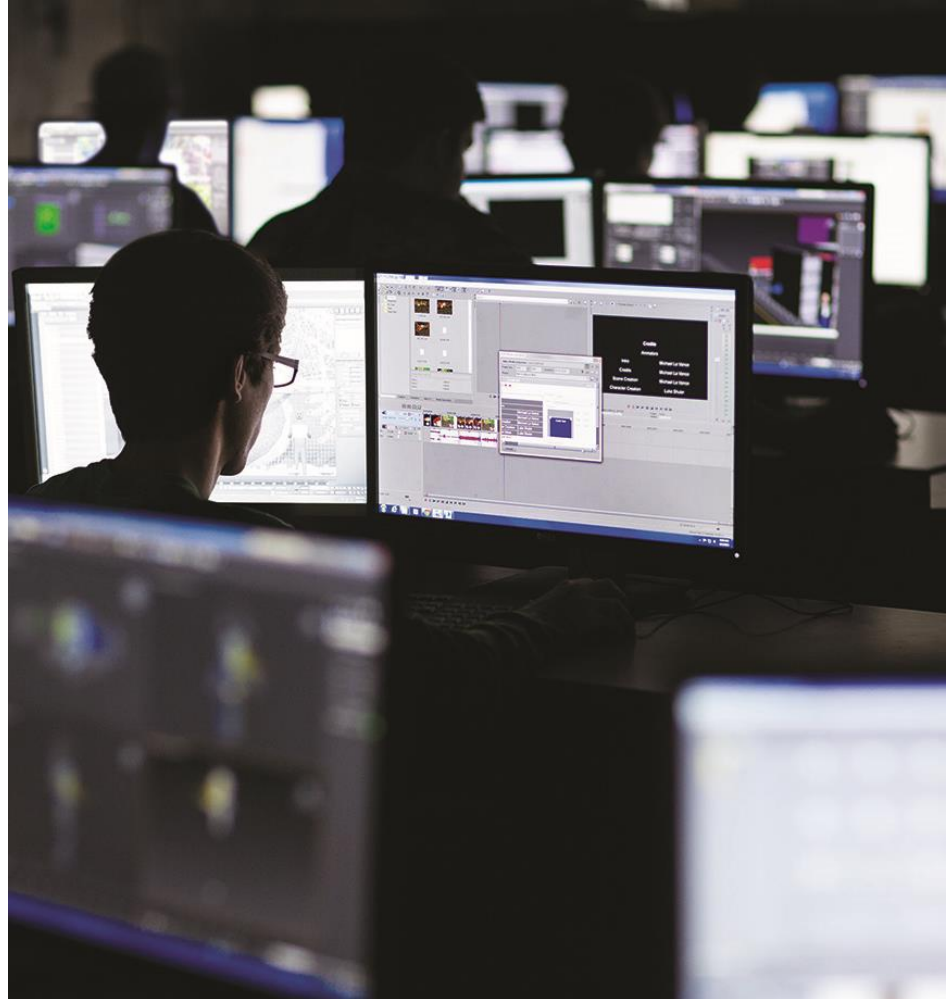
Do not interfere with legitimate operations



Develop trusted applications using familiar tools and processes

What Do You Need to Develop and Run Enclaves?

- A platform with an Intel® SGX-enabled CPU and BIOS
- Intel SGX platform software (PSW)
- The Intel SGX software-development kit (SDK)
- Microsoft Visual Studio 2015 Professional* or Eclipse™
 - Supports Intel SGX debugger



What Can You Do with Enclaves?

1

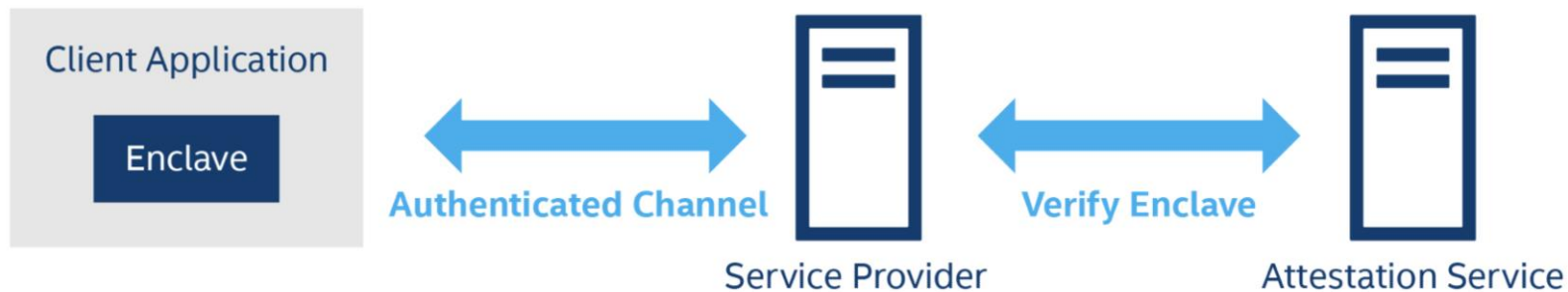
Access secrets

2

Attest services

3

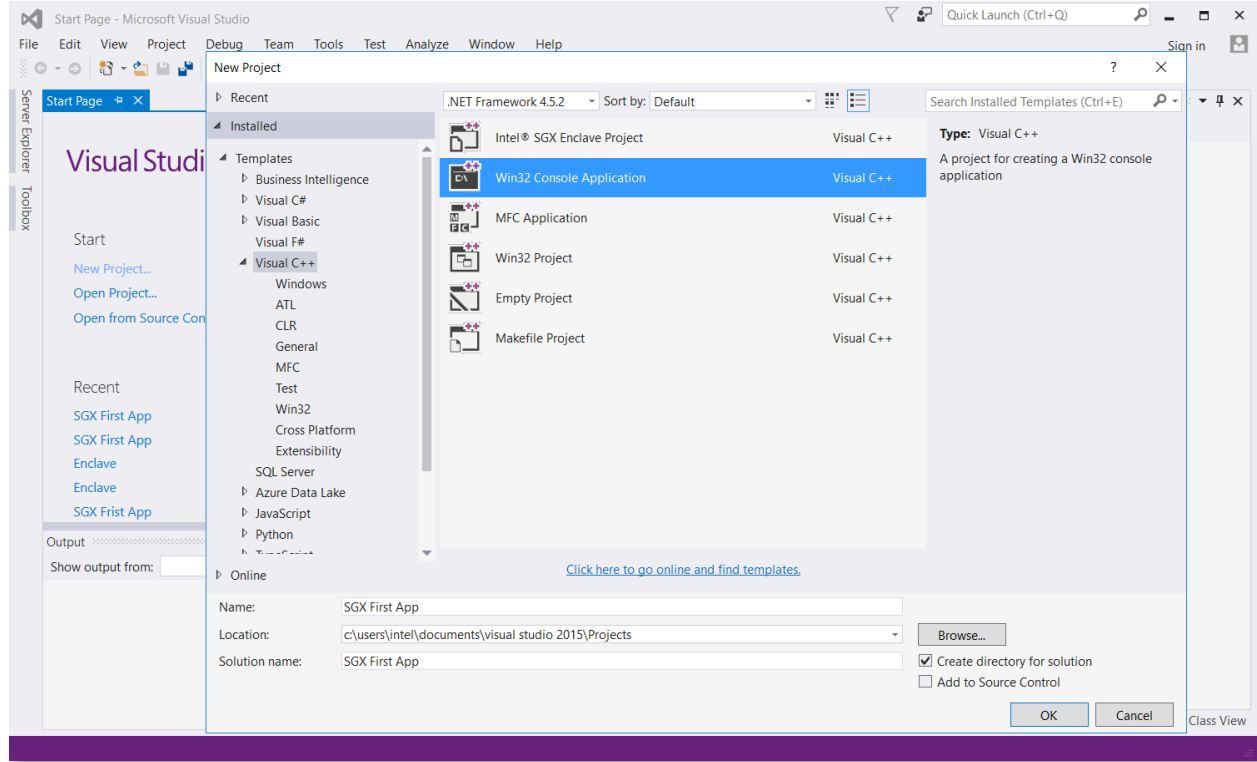
Seal secrets



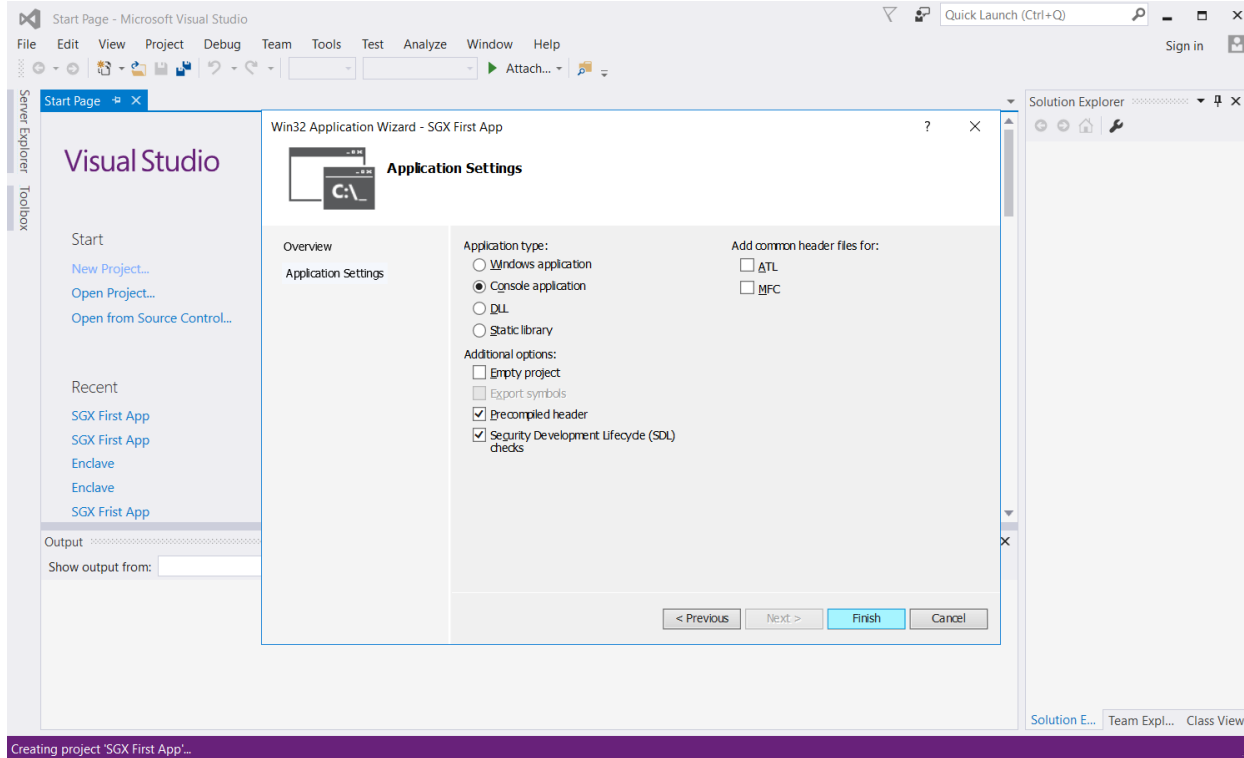
Creating a “SGX First App” Intel® SGX Application

Demonstration

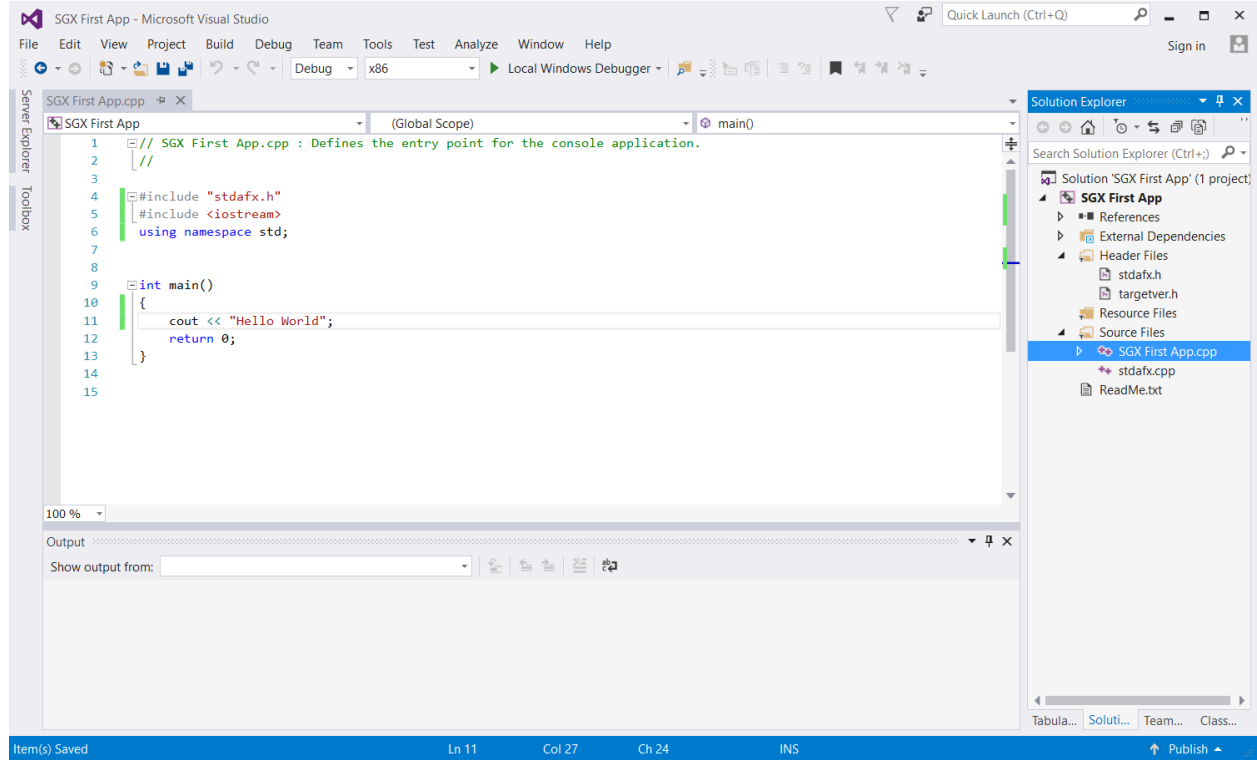
Create a Console-Application Project



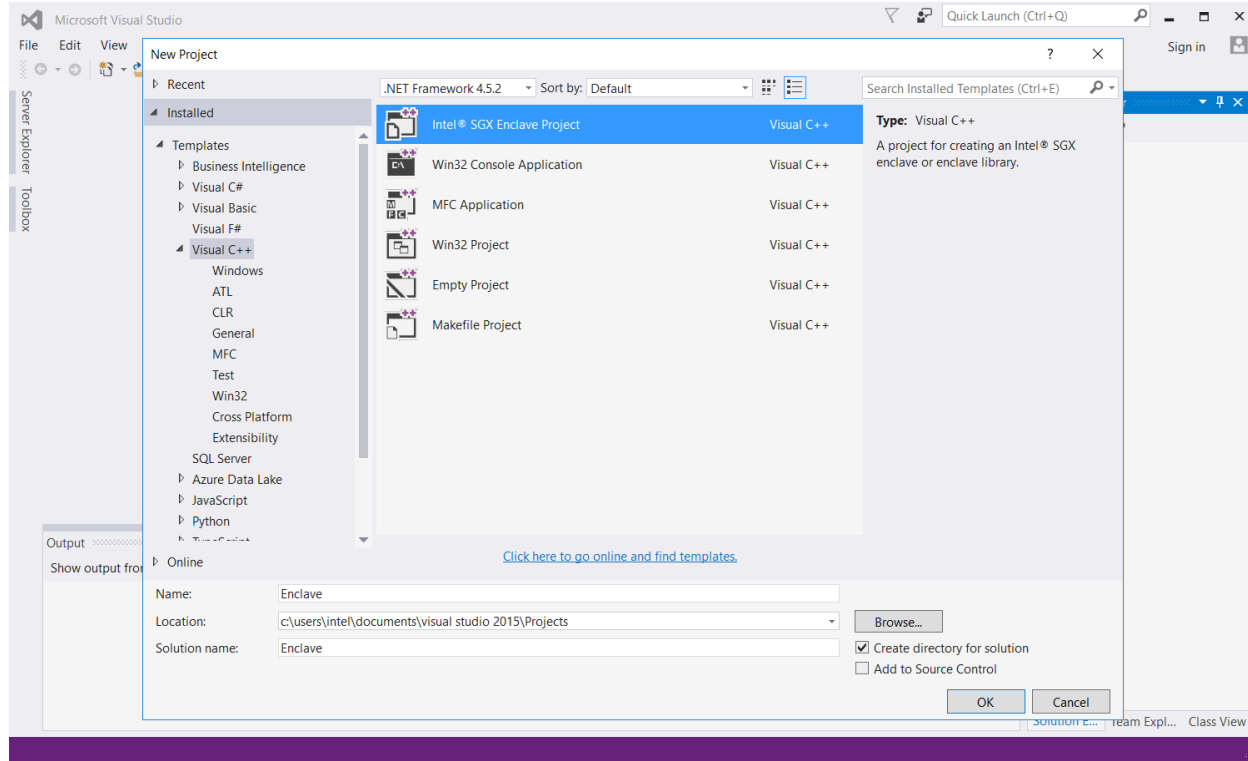
Create a Console-Application Project



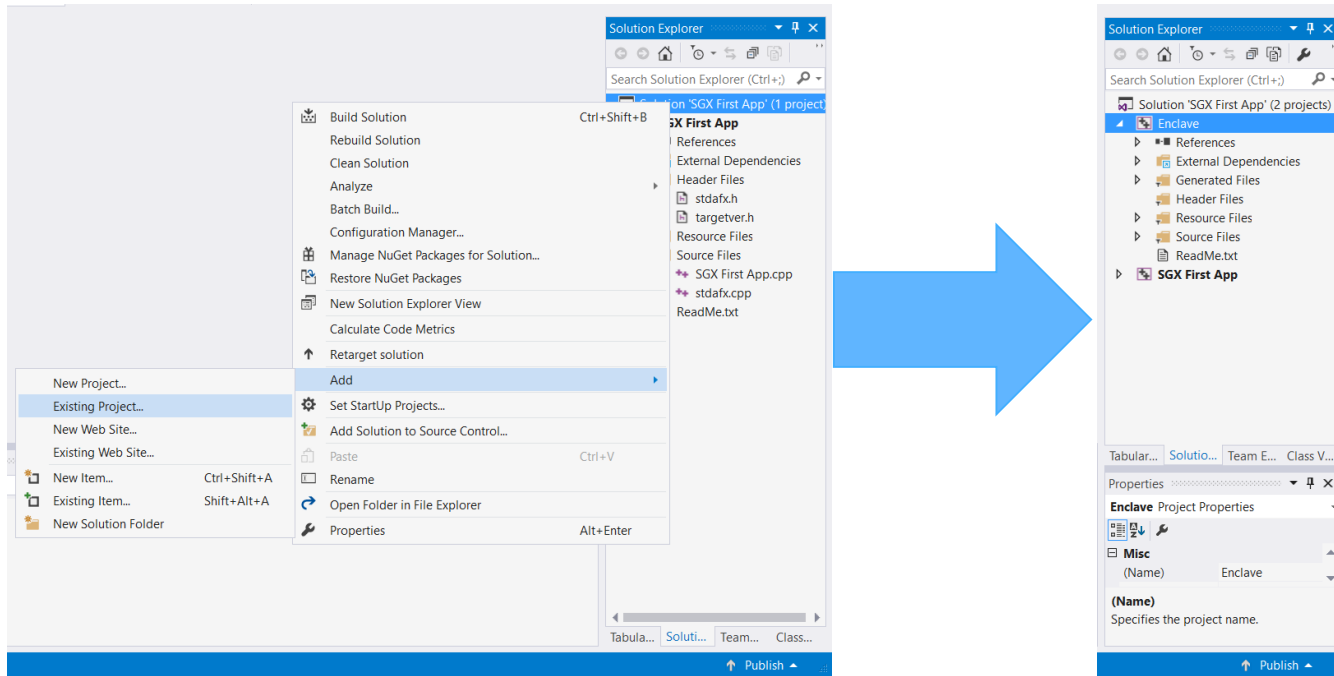
Create a Console-Application Project



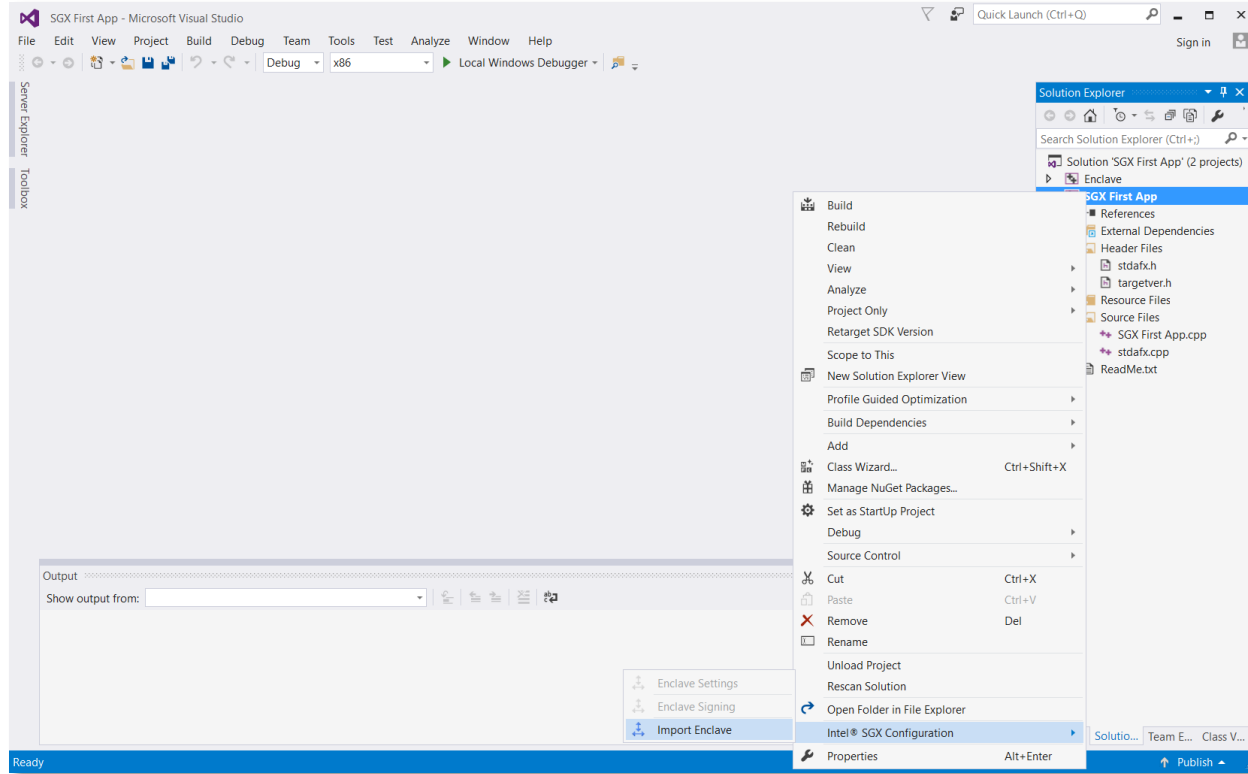
Create an Enclave-Application Project



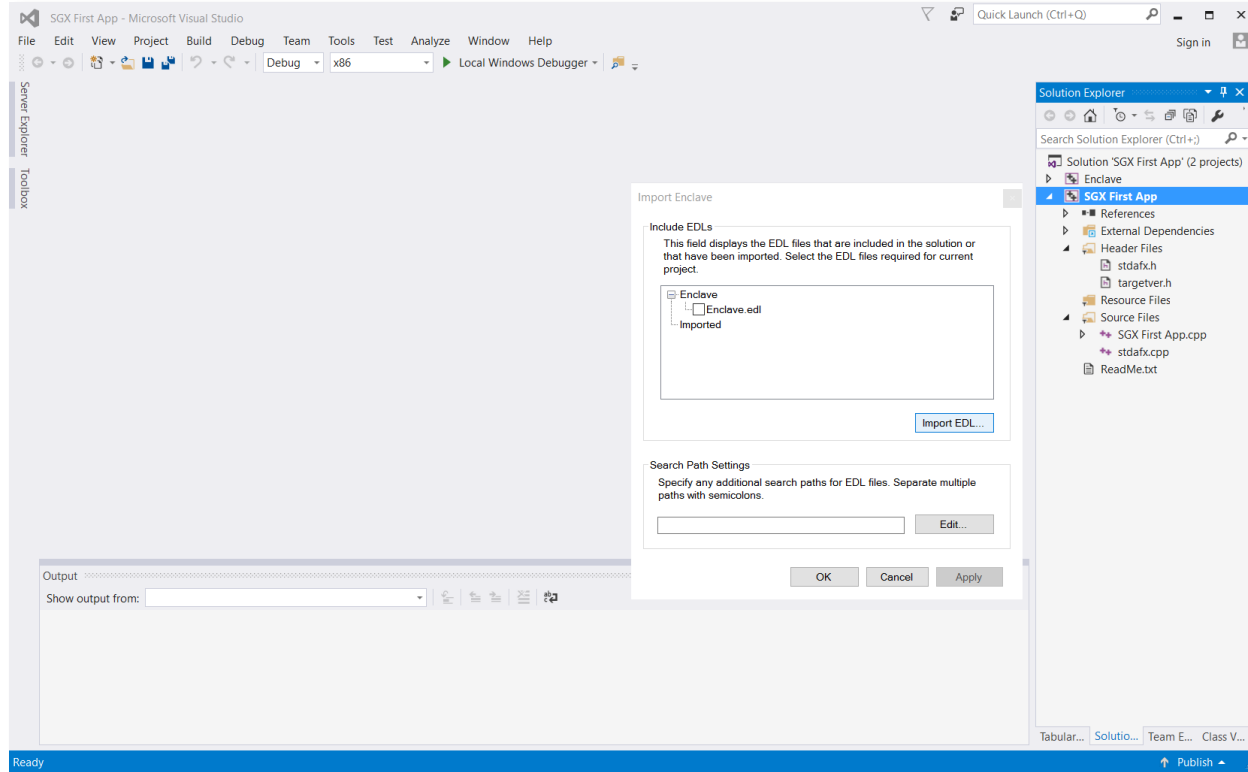
Add an Enclave Project to the Console-Application Project



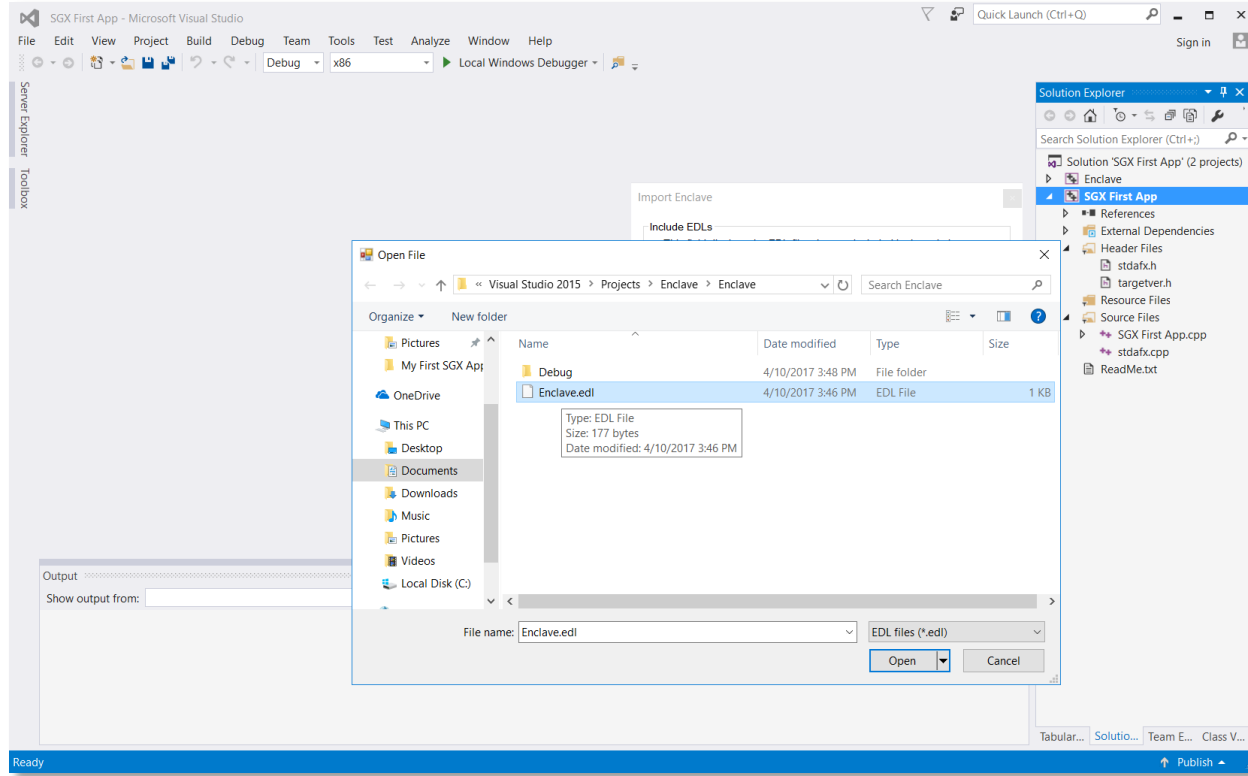
Import an Enclave into a Console Application



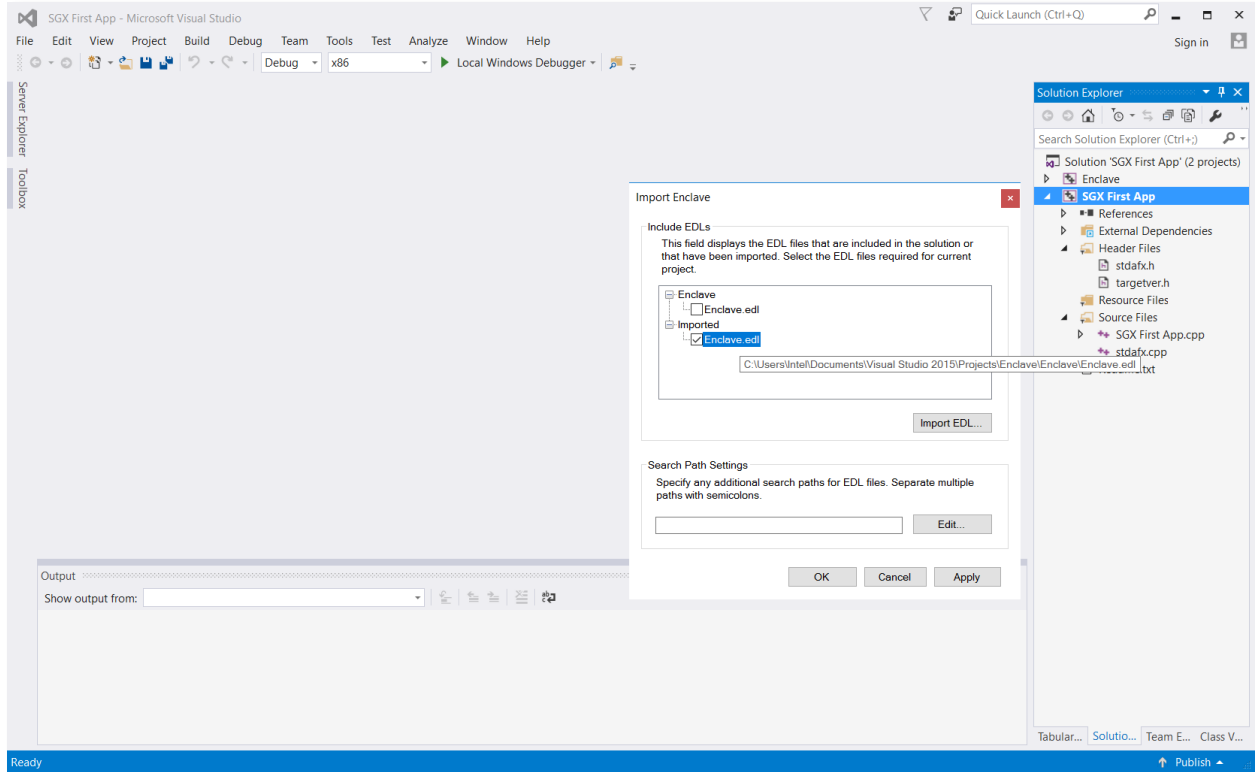
Import an Enclave into a Console Application



Import an Enclave into a Console Application



Import an Enclave into a Console Application



Technical Discussion: EDL

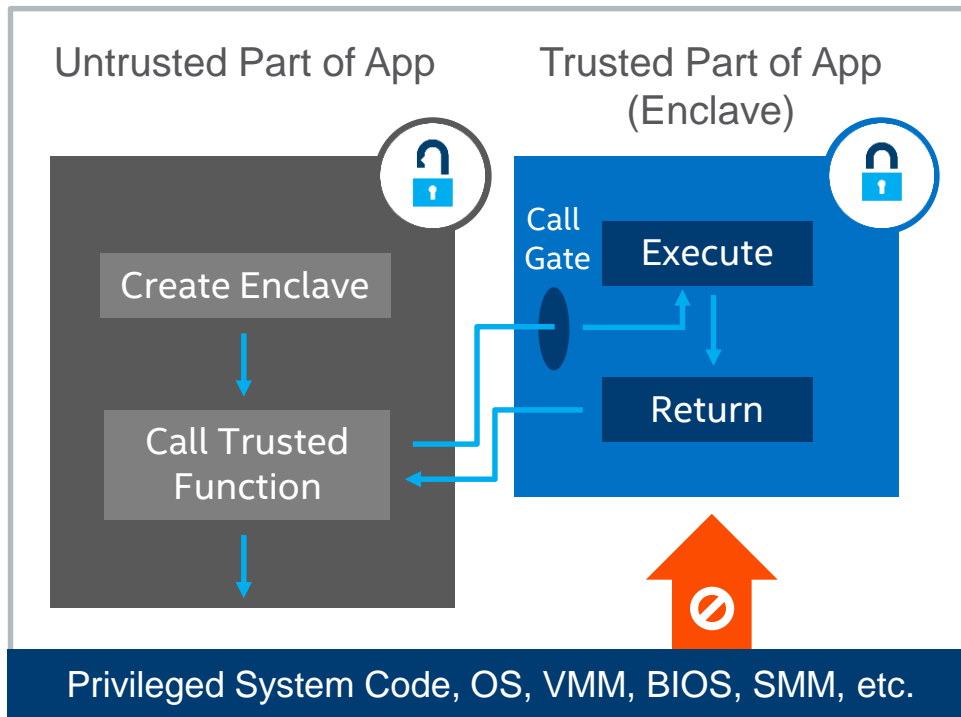
A Brief Introduction to EDL

EDL stands for *Enclave Definition Language*.

It defines the interface between the trusted and untrusted portions of your applications.

```
01  enclave {
02      //Include files
03
04      //Import other edl files
05
06      //Data structure declarations to be used as parameters of the
07      //function prototypes in edl
08
09      trusted {
10          //Include header files if any
11          //Will be includedd in enclave_t.h
12
13          //Trusted function prototypes
14
15      };
16
17      untrusted {
18          //Include header files if any
19          //Will be included in enclave_u.hhead
20
21          //Untrusted function prototypes
22
23      };
24  };
```

Trusted and Untrusted Application Code

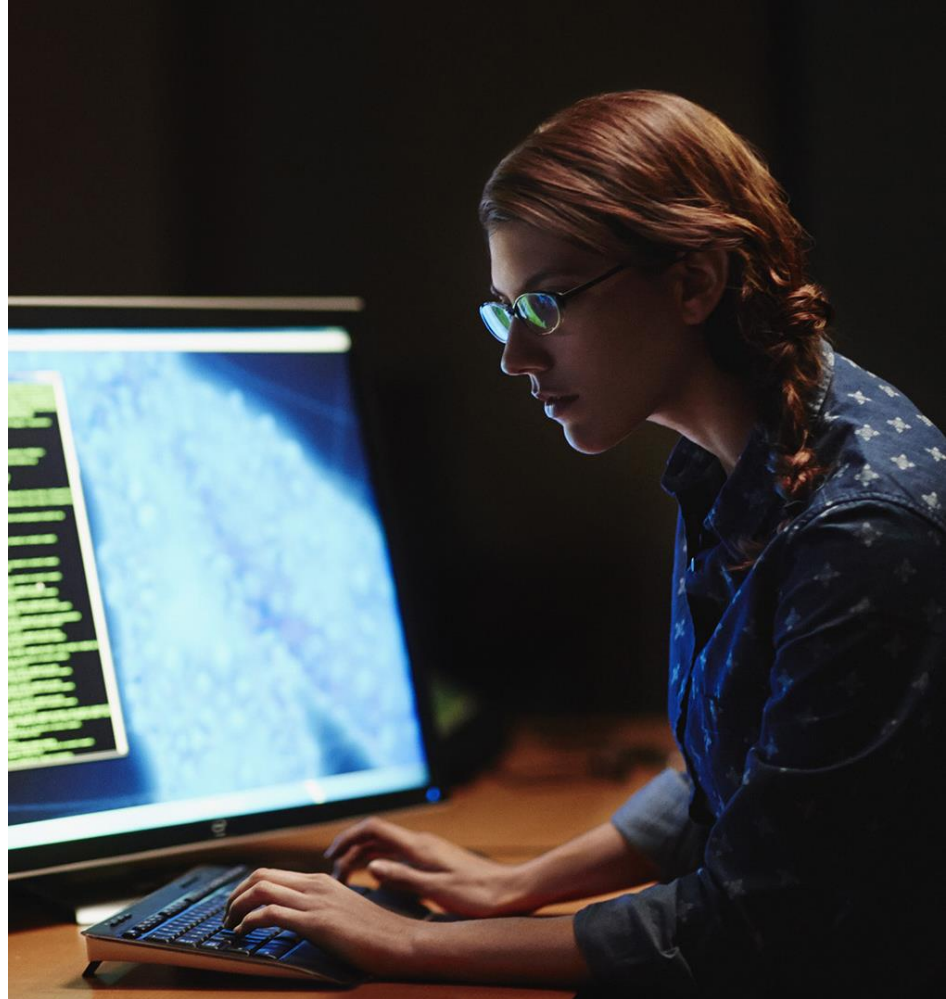


1. An Intel® SGX-enabled app is built with trusted and untrusted parts.
2. The app runs and creates an enclave, which is placed in trusted memory.
3. A trusted function is called, code running inside the enclave sees the data as clear text, and external access to the data is denied.
4. The function returns, and the enclave data remains in trusted memory.

Edge Code Auto-generation

The Intel® SGX SDK auto-generates edge routines defined by EDL through use of the Edger8r* tool:

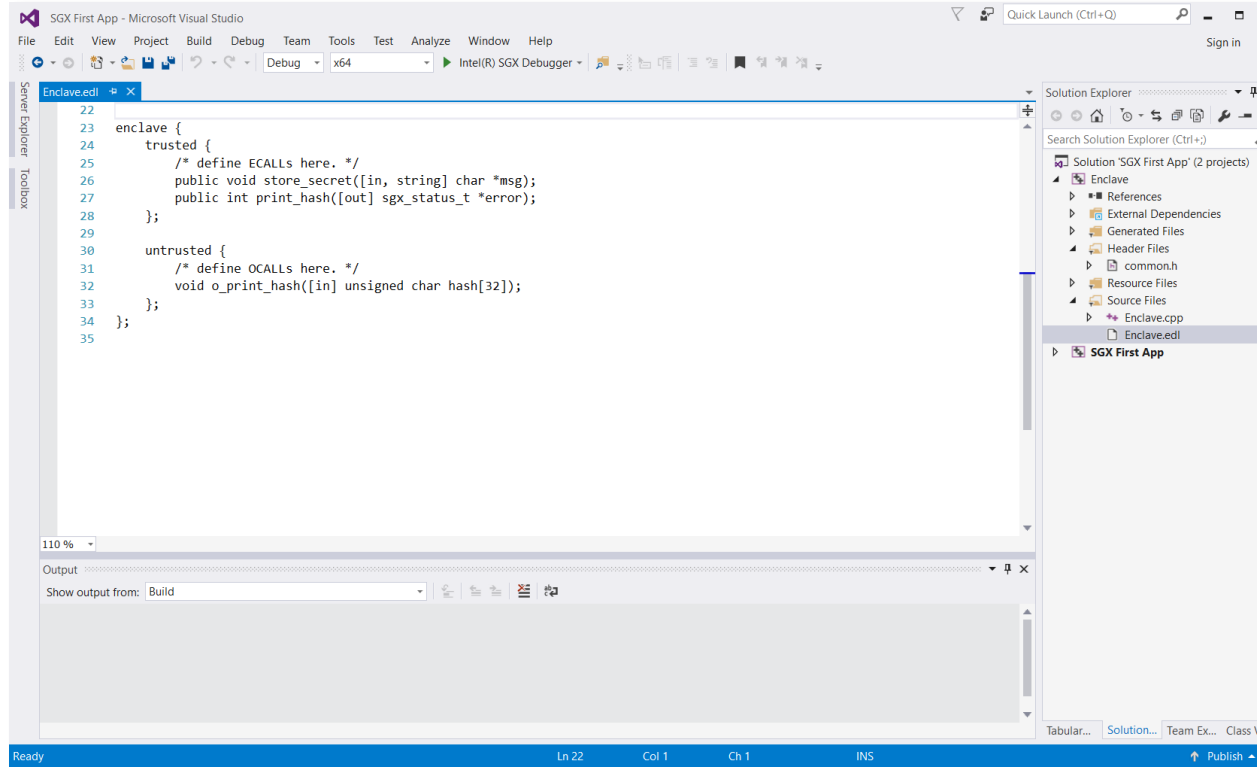
- Intel SGX uses only plain C, so wrapper functions are necessary.
- It auto-generates the code the first time you build a project.
- It produces edge routines for accessing enclaves.
- Edge routines are specified by the developer in the EDL file.



Completing the “SGX First App” Enclave Application

Demonstration

Add a Trusted Call to an Enclave Project



Add a Trusted Call to an Enclave Project

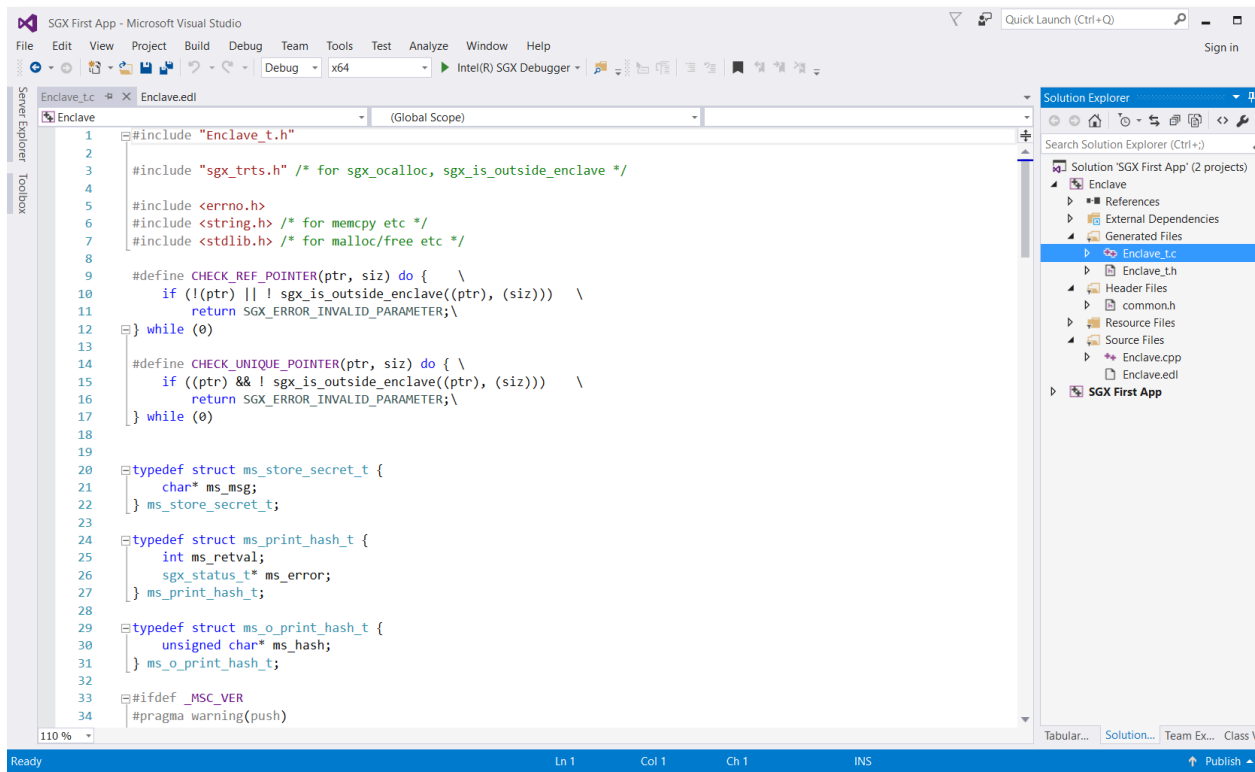
The screenshot shows the Microsoft Visual Studio IDE with the following components:

- Code Editor:** Displays the `Enclave.edl` file with the following code:

```
22
23  enclave {
24      trusted {
25          /* define ECALLS here. */
26          public void store_secret([in, string] char *msg);
27          public int print_hash([out] sgx_status_t *error);
28      };
29
30      untrusted {
31          /* define OCALLS here. */
32          void o_print_hash([in] unsigned char hash[32]);
33      };
34  };
35
```
- Solution Explorer:** Shows the project structure for "SGX First App" (2 projects). The "Enclave" project is selected, showing files like `Enclave.cpp` and `Enclave.edl`.
- Context Menu:** A right-click context menu is open over the `Enclave.edl` file, with the "Compile" option (Ctrl+F7) highlighted.
- Output Window:** Shows the build output for the "Enclave" project, indicating a successful build:

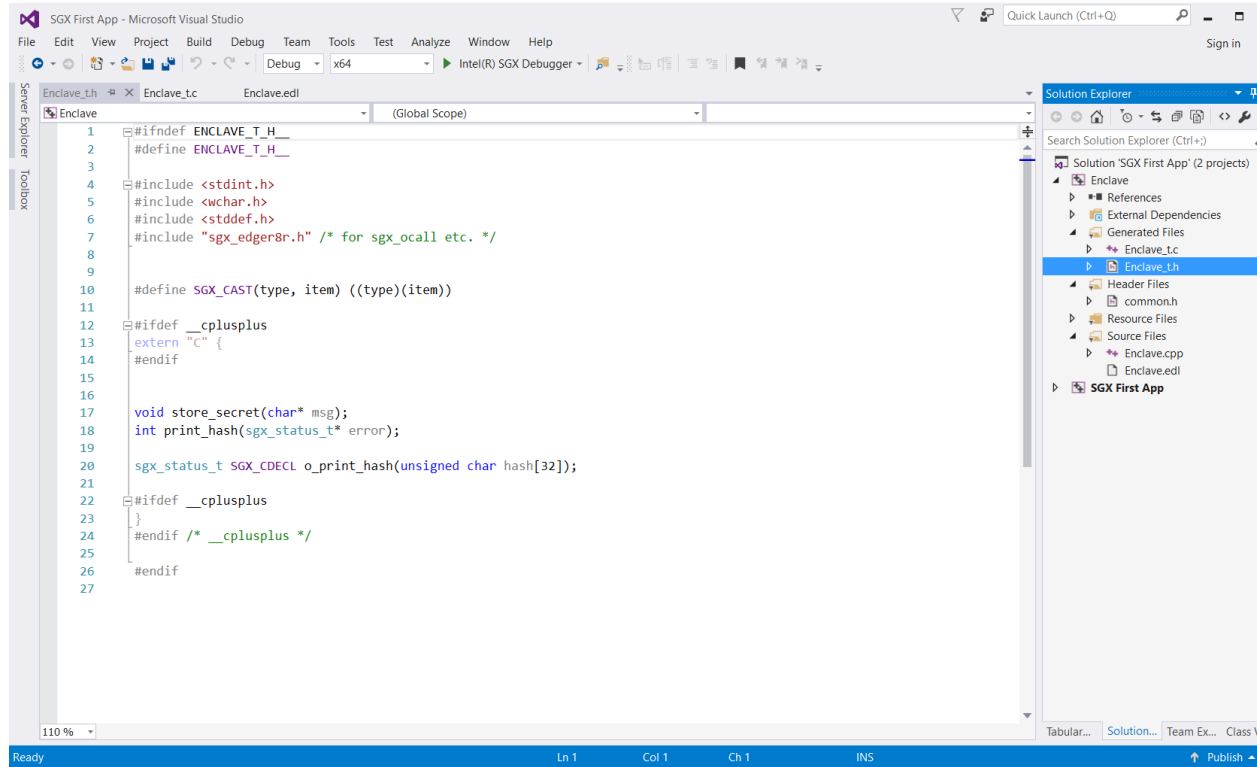
```
1>----- Build started: Project: Enclave, Configuration: Debug x64 -----
1> Creating proxy/bridge routines
----- Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped -----
```
- Status Bar:** Displays "Build succeeded" and a "Publish" button.

Add a Trusted Call to an Enclave Project



```
1 #include "Enclave_t.h"
2
3 #include "sgx_trts.h" /* for sgx_ocalloc, sgx_is_outside_enclave */
4
5 #include <errno.h>
6 #include <string.h> /* for memcpy etc */
7 #include <stdlib.h> /* for malloc/free etc */
8
9 #define CHECK_REF_POINTER(ptr, siz) do { \
10     if (!(ptr) || ! sgx_is_outside_enclave((ptr), (siz))) \
11         return SGX_ERROR_INVALID_PARAMETER;\
12 } while (0)
13
14 #define CHECK_UNIQUE_POINTER(ptr, siz) do { \
15     if ((ptr) && ! sgx_is_outside_enclave((ptr), (siz))) \
16         return SGX_ERROR_INVALID_PARAMETER;\
17 } while (0)
18
19
20 typedef struct ms_store_secret_t {
21     char* ms_msg;
22 } ms_store_secret_t;
23
24 typedef struct ms_print_hash_t {
25     int ms_retval;
26     sgx_status_t* ms_error;
27 } ms_print_hash_t;
28
29 typedef struct ms_o_print_hash_t {
30     unsigned char* ms_hash;
31 } ms_o_print_hash_t;
32
33 #ifdef _MSC_VER
34 #pragma warning(push)
```

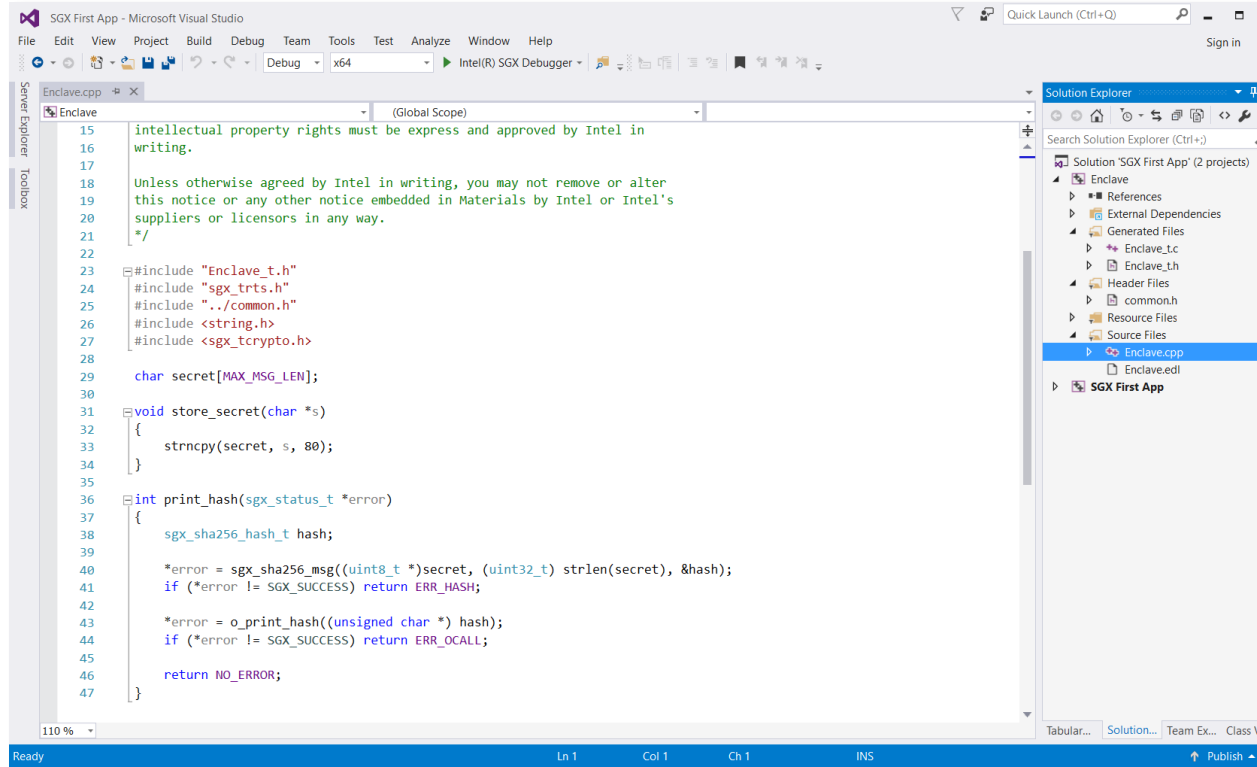
Add a Trusted Call to an Enclave Project



The screenshot shows the Microsoft Visual Studio IDE with the 'Enclave.edl' file open in the editor. The code defines a trusted call function and includes necessary headers. The Solution Explorer on the right shows the project structure, including the 'Enclave' project and its sub-projects 'Enclave_tc' and 'Enclave_th'.

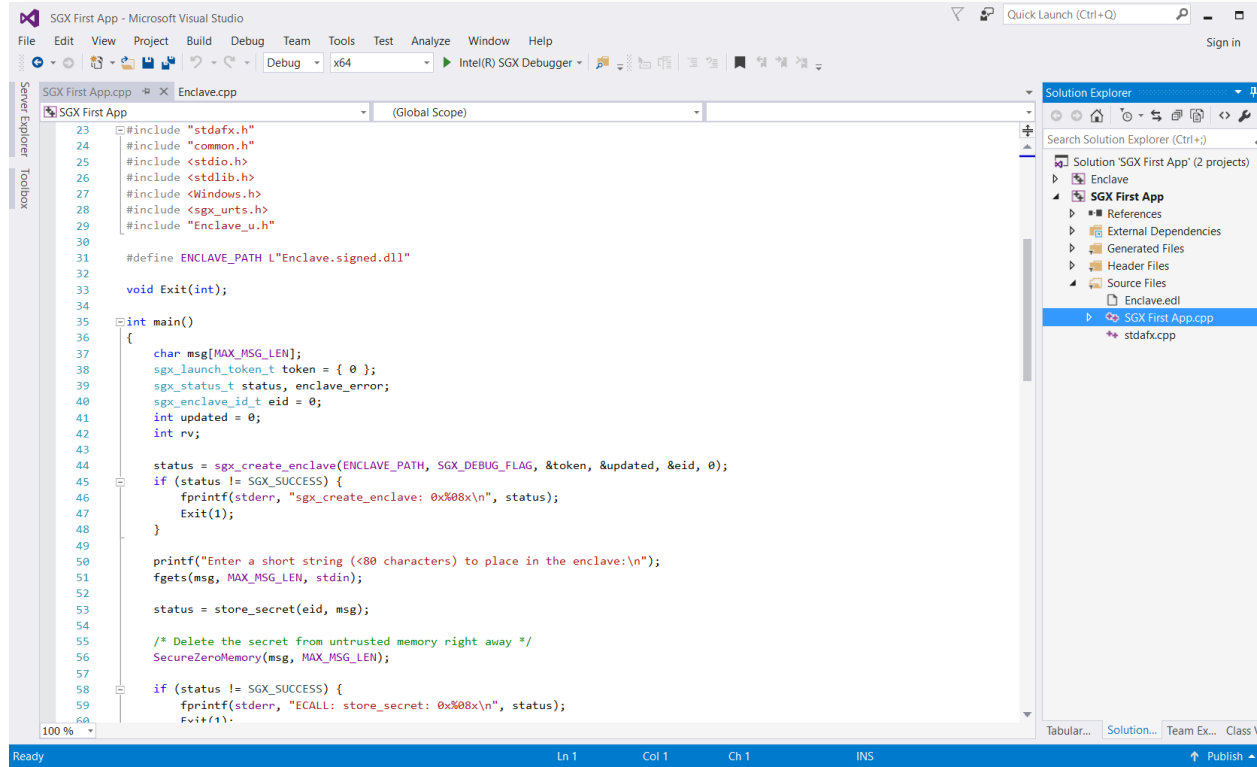
```
1  #ifndef ENCLAVE_T_H_
2  #define ENCLAVE_T_H_
3
4  #include <stdint.h>
5  #include <wchar.h>
6  #include <stddef.h>
7  #include "sgx_edger8r.h" /* for sgx_ocall etc. */
8
9
10 #define SGX_CAST(type, item) ((type)item)
11
12 #ifdef __cplusplus
13 extern "C" {
14 #endif
15
16
17 void store_secret(char* msg);
18 int print_hash(sgx_status_t* error);
19
20 sgx_status_t SGX_CDECL o_print_hash(unsigned char hash[32]);
21
22 #ifdef __cplusplus
23 }
24 #endif /* __cplusplus */
25
26 #endif
```

Implement Enclave Functions



```
15 intellectual property rights must be express and approved by Intel in
16 writing.
17
18 Unless otherwise agreed by Intel in writing, you may not remove or alter
19 this notice or any other notice embedded in Materials by Intel or Intel's
20 suppliers or licensors in any way.
21 */
22
23 #include "Enclave_t.h"
24 #include "sgx_trts.h"
25 #include "../common.h"
26 #include <string.h>
27 #include <sgx_tcrypto.h>
28
29 char secret[MAX_MSG_LEN];
30
31 void store_secret(char *s)
32 {
33     strncpy(secret, s, 80);
34 }
35
36 int print_hash(sgx_status_t *error)
37 {
38     sgx_sha256_hash_t hash;
39
40     *error = sgx_sha256_msg((uint8_t *)secret, (uint32_t) strlen(secret), &hash);
41     if (*error != SGX_SUCCESS) return ERR_HASH;
42
43     *error = o_print_hash((unsigned char *) hash);
44     if (*error != SGX_SUCCESS) return ERR_OCALL;
45
46     return NO_ERROR;
47 }
```

Implement the Enclave Application

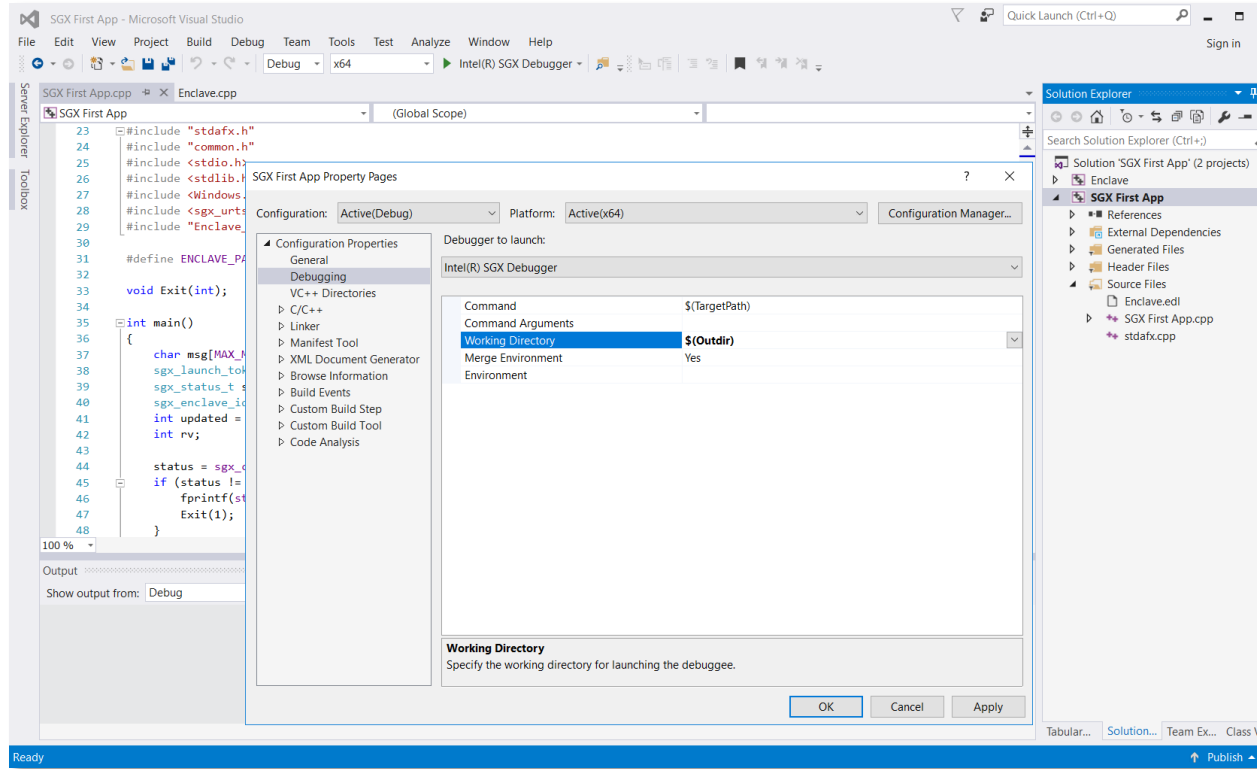


The screenshot displays the Microsoft Visual Studio IDE with the following components:

- Menu Bar:** File, Edit, View, Project, Build, Debug, Team, Tools, Test, Analyze, Window, Help.
- Toolbar:** Includes icons for file operations, build, and debugging. The current configuration is Debug mode on a x64 architecture using the Intel(R) SGX Debugger.
- Code Editor:** Shows the source file `Enclave.cpp` with the following code:

```
23 #include "stdafx.h"
24 #include "common.h"
25 #include <stdio.h>
26 #include <stdlib.h>
27 #include <Windows.h>
28 #include <sgx_urts.h>
29 #include "Enclave_u.h"
30
31 #define ENCLAVE_PATH L"Enclave.signed.dll"
32
33 void Exit(int);
34
35 int main()
36 {
37     char msg[MAX_MSG_LEN];
38     sgx_launch_token_t token = { 0 };
39     sgx_status_t status, enclave_error;
40     sgx_enclave_id_t eid = 0;
41     int updated = 0;
42     int rv;
43
44     status = sgx_create_enclave(ENCLAVE_PATH, SGX_DEBUG_FLAG, &token, &updated, &eid, 0);
45     if (status != SGX_SUCCESS) {
46         fprintf(stderr, "sgx_create_enclave: 0x%08x\n", status);
47         Exit(1);
48     }
49
50     printf("Enter a short string (<80 characters) to place in the enclave:\n");
51     fgets(msg, MAX_MSG_LEN, stdin);
52
53     status = store_secret(eid, msg);
54
55     /* Delete the secret from untrusted memory right away */
56     SecureZeroMemory(msg, MAX_MSG_LEN);
57
58     if (status != SGX_SUCCESS) {
59         fprintf(stderr, "ECALL: store_secret: 0x%08x\n", status);
60         Exit(1);
61     }
62 }
```
- Solution Explorer:** Shows the project structure for "SGX First App" (2 projects). The "SGX First App" project contains:
 - References
 - External Dependencies
 - Generated Files
 - Header Files
 - Source Files
 - Enclave.edl
 - SGX First App.cpp
 - stdafx.cpp
- Status Bar:** Shows "Ready" and column/line information (Ln 1, Col 1, Ch 1, INS).

Implement the Enclave Application



Implement the Enclave Application

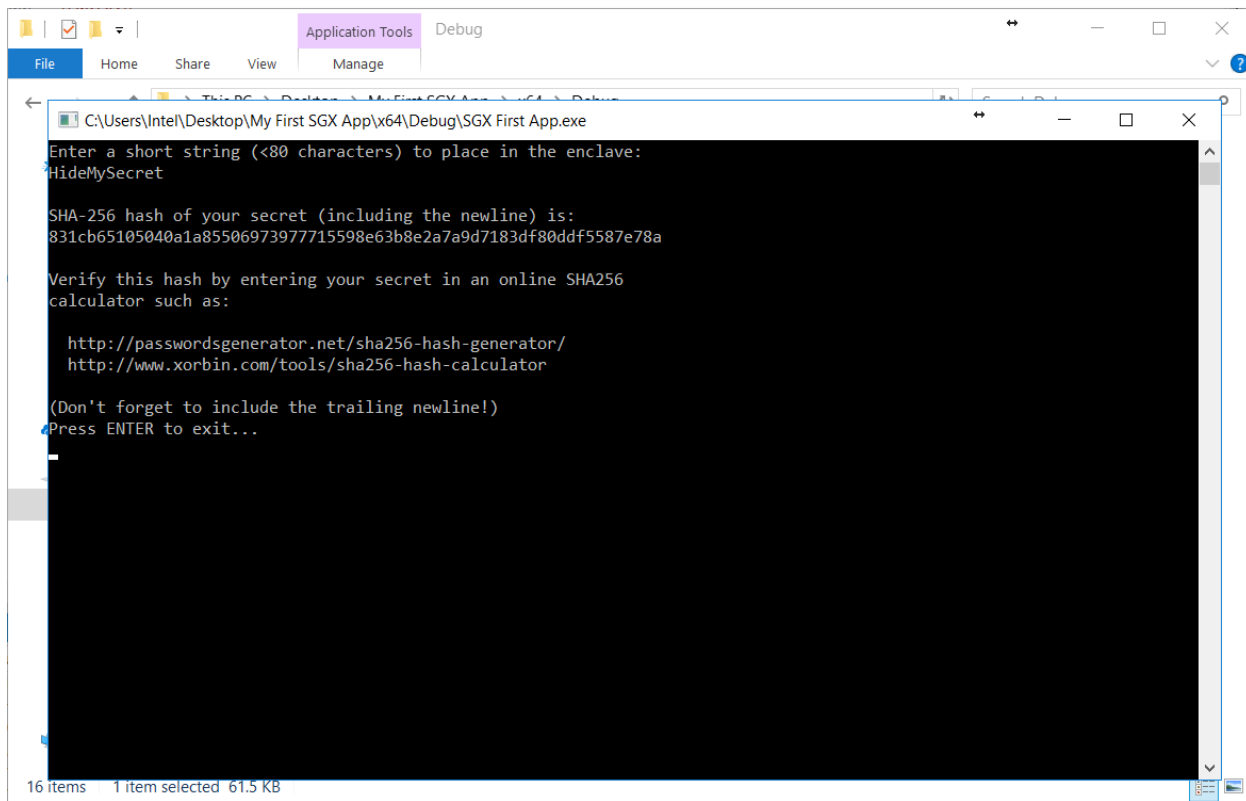
The screenshot shows the Microsoft Visual Studio IDE with the following components:

- Code Editor:** Displays the source code for `stdafx.h` with lines 23 through 48 visible.
- File Explorer:** Shows the file system structure for the project's `Debug` directory. The files listed are:

Name	Date modified	Type	Size
SGX First App.tlog	4/10/2017 4:27 PM	File folder	
Enclave.dll	4/10/2017 4:27 PM	Application extens...	160 KB
Enclave.exp	4/10/2017 4:27 PM	Exports Library File	1 KB
Enclave.lib	4/10/2017 4:27 PM	Object File Library	2 KB
Enclave.pdb	4/10/2017 4:27 PM	Program Debug D...	484 KB
Enclave.signed.dll	4/10/2017 4:27 PM	Application extens...	160 KB
Enclave_u.obj	4/10/2017 4:27 PM	Object File	8 KB
SGX First App.Build.CppClean.log	4/10/2017 4:27 PM	Text Document	2 KB
SGX First App.exe	4/10/2017 4:27 PM	Application	62 KB
SGX First App.ilk	4/10/2017 4:27 PM	Incremental Linker...	352 KB
SGX First App.log	4/10/2017 4:27 PM	Text Document	1 KB
SGX First App.obj	4/10/2017 4:27 PM	Object File	37 KB
SGX First App.pdb	4/10/2017 4:27 PM	Program Debug D...	628 KB
stdafx.obj	4/10/2017 4:27 PM	Object File	5 KB
vc140.idb	4/10/2017 4:27 PM	VC++ Minimum R...	579 KB
vc140.pdb	4/10/2017 4:27 PM	Program Debug D...	180 KB
- Solution Explorer:** Shows the project structure for 'SGX First App' (2 projects), including folders for References, External Dependencies, Generated Files, Header Files, Source Files, and Enclave.edl. The source files `SGX First App.cpp` and `stdafx.cpp` are listed.
- Output Window:** Shows the build process output:

```
1> SGX First App. 16 items 1 item selected 615 KB
----- Rebuild All: 2 succeeded, 0 failed, 0 skipped -----
```
- Status Bar:** Displays 'Rebuild All succeeded' and a 'Publish' button.

Implement the Enclave Application



The screenshot shows a Windows File Explorer window titled "Application Tools Debug" with a sub-window titled "C:\Users\Intel\Desktop\My First SGX App\x64\Debug\SGX First App.exe". The terminal window displays the following text:

```
Enter a short string (<80 characters) to place in the enclave:  
HideMySecret  
  
SHA-256 hash of your secret (including the newline) is:  
831cb65105040a1a85506973977715598e63b8e2a7a9d7183df80ddf5587e78a  
  
Verify this hash by entering your secret in an online SHA256  
calculator such as:  
  
  http://passwordsgenerator.net/sha256-hash-generator/  
  http://www.xorbin.com/tools/sha256-hash-calculator  
  
(Don't forget to include the trailing newline!)  
Press ENTER to exit...  
_
```

At the bottom of the File Explorer window, it shows "16 items 1 item selected 61.5 KB".

Conclusion

Wrap-up

- Hardware-based protection for sensitive data
- SDK auto-generates many necessary wrapper functions
- Build trusted apps with familiar tools
- Visit the links at the end of the webinar to learn more and see other examples



Additional Resources

Intel® SGX sample code and tutorials: <https://software.intel.com/en-us/sgx/code-samples>

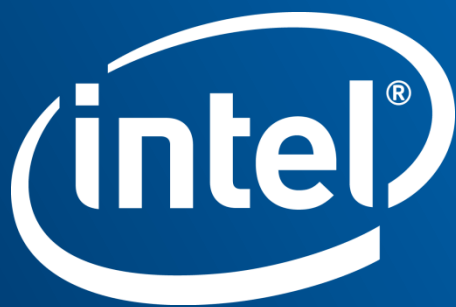
Intel SGX forum: <https://software.intel.com/en-us/forums/intel-software-guard-extensions-intel-sgx>

Intel SGX for Linux*: <https://01.org/intel-software-guard-extensions>

Intel SGX at the Intel® Developer Zone: <https://software.intel.com/en-us/sgx>

Q&A





Legal Notices and Disclaimers

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com](https://www.intel.com).

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel, the Intel logo, and Intel Core are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

© 2017 Intel Corporation.