Get Started with Intel® Integrated Performance Primitives Cryptography

Contents

Get Started with Intel® Integrated Performance Primitives Cryptography for Intel® oneAPI Base Toolkit

1

Intel® Integrated Performance Primitives (Intel® IPP) Cryptography is a software library that provides a broad range of secure and efficient cryptographic algorithm implementations.

The library is delivered as a part of Intel® oneAPI Base Toolkit. You may install specific library version as well. This get started guide assumes you have installed Intel IPP Cryptography library as a part of the toolkit.

Prerequisites (Windows* OS)

Set Environment Variables

After installing Intel IPP Cryptography, set the *PATH*, *LIB*, and *INCLUDE* environment variables by running the script appropriate to your target platform architecture. The scripts are available in *<install dir>\ippcp\bin*.

By default, the *<install dir>* is *C:\Program files (x86)\Intel\oneapi*. See Intel IPP high-level directories structure.

Configure Your IDE Environment to Link with Intel IPP Cryptography

To configure your Microsoft* Visual Studio* development system for linking with the Intel IPP Cryptography library, follow the steps below. Though some versions of the Visual Studio* IDE may vary slightly in the menu items mentioned below, the fundamental configuring steps are applicable to all these versions.

1. In Solution Explorer, right-click your project and click **Properties**.
2. Select **Configuration Properties** > **VC++ Directories** and set the following from the **Select directories for** drop down menu:
   - **Include Files** menu item, and then type in the directory for the Intel IPP Cryptography include files (default is *<install_dir>\ippcp\include*)
   - **Library Files** menu item, and then type in the directory for the Intel IPP Cryptography library files (default is *<install_dir>\ippcp\lib\<arch>*)
   - **Executable Files** menu item, and then type in the directory for the Intel IPP Cryptography executable files (default is *<install_dir>\redist\<arch>\ippcp*)

Build and Run Your First Intel® IPP Cryptography Application (Windows* OS)

The code example below represents a short application to help you get started with Intel IPP Cryptography:

```
#include "ippcp.h"
#include <stdio.h>

#define   ippCPUID_MMX            0x00000001   /* Intel® Architecture MMX™
technology                                        */
#define   ippCPUID_SSE            0x00000002   /* Intel® Streaming SIMD
Extensions                                        */
#define   ippCPUID_SSE2           0x00000004   /* Intel® Streaming SIMD Extensions
2                                              */
#define   ippCPUID_SSE3           0x00000008   /* Intel® Streaming SIMD Extensions
3                                              */
```

```
#define   ippCPUID_SSSE3           0x00000010   /* Supplemental Streaming SIMD Extensions
3                                          */
#define   ippCPUID_MOVBE           0x00000020   /* The processor supports the MOVBE
instruction                                  */
#define   ippCPUID_SSE41           0x00000040   /* Intel® Streaming SIMD Extensions
4.1                                          */
#define   ippCPUID_SSE42           0x00000080   /* Intel® Streaming SIMD Extensions
4.2                                          */
#define   ippCPUID_AVX             0x00000100   /* Intel® Advanced Vector Extensions (Intel®
AVX)                                   */
#define   ippAVX_ENABLEDBYOS       0x00000200   /* The operating system supports Intel
AVX                                         */
#define   ippCPUID_AES             0x00000400   /* Intel® AES New Instructions (Intel® AES-
NI)                                       */
#define   ippCPUID_CLMUL           0x00000800   /* PCLMULQDQ
instruction                                                  */
#define   ippCPUID_ABR             0x00001000   /*
Reserved                                                                */
#define   ippCPUID_RDRAND          0x00002000   /* Read Random Number
instructions                                                 */
#define   ippCPUID_F16C            0x00004000   /* Float16
instructions                                                   */
#define   ippCPUID_AVX2            0x00008000   /* Intel® Advanced Vector Extensions
2                                           */
#define   ippCPUID_ADCOX           0x00010000   /* ADCX and ADOX
instructions                                              */
#define   ippCPUID_RDSEED          0x00020000   /* The RDSEED
instruction                                                 */
#define   ippCPUID_PREFETCHW       0x00040000   /* The PREFETCHW
instruction                                               */
#define   ippCPUID_SHA             0x00080000   /* Intel® SHA
Extensions                                                 */
#define   ippCPUID_AVX512F         0x00100000   /* Intel® Advanced Vector Extensions 512 (Intel®
AVX-512) Foundation instructions */
#define   ippCPUID_AVX512CD        0x00200000   /* Intel AVX-512 Conflict Detection
instructions                                  */
#define   ippCPUID_AVX512ER        0x00400000   /* Intel AVX-512 Exponential & Reciprocal
instructions                                */
#define   ippCPUID_AVX512PF        0x00800000   /* Intel AVX-512 Prefetch
instructions                                             */
#define   ippCPUID_AVX512BW        0x01000000   /* Intel AVX-512 Byte & Word
instructions                                            */
#define   ippCPUID_AVX512DQ        0x02000000   /* Intel AVX-512 DWord & QWord
instructions                                           */
#define   ippCPUID_AVX512VL        0x04000000   /* Intel AVX-512 Vector Length
extensions                                             */
#define   ippCPUID_AVX512VBMI      0x08000000   /* Intel AVX-512 Vector Bit Manipulation
instructions                                  */
#define   ippCPUID_MPX             0x10000000   /* Intel® Memory Protection
Extensions                                              */
#define   ippCPUID_AVX512_4FMADDPS 0x20000000   /* Intel AVX-512 DL floating-point single
precision                                   */
#define   ippCPUID_AVX512_4VNNIW   0x40000000   /* Intel AVX-512 DL enhanced word variable
precision                                  */
#define   ippCPUID_KNC             0x80000000   /* Intel® Xeon Phi™
Coprocessor                                              */

int main(int argc, char* argv[])
```

```
{
   const IppLibraryVersion *lib;
   IppStatus status;
   Ipp64u mask, emask;

   /* Get Intel IPP Cryptography library version info */
   lib = ippcpGetLibVersion();
   printf("%s %s\n", lib->Name, lib->Version);

   /* Get CPU features and features enabled with selected library level */
   status = ippcpGetCpuFeatures( &mask );
   if( ippStsNoErr == status ) {
      emask = ippcpGetEnabledCpuFeatures();
      printf("Features supported by CPU\tby Intel® Integrated Performance Primitives Cryptography
\n");
      printf("-----------------------------------------\n");
      printf("  ippCPUID_MMX        = ");
      printf("%c\t%c\t",( mask & ippCPUID_MMX ) ? 'Y':'N',( emask & ippCPUID_MMX ) ? 'Y':'N');
      printf("Intel® Architecture MMX technology supported\n");
      printf("  ippCPUID_SSE        = ");
      printf("%c\t%c\t",( mask & ippCPUID_SSE ) ? 'Y':'N',( emask & ippCPUID_SSE ) ? 'Y':'N');
      printf("Intel® Streaming SIMD Extensions\n");
      printf("  ippCPUID_SSE2       = ");
      printf("%c\t%c\t",( mask & ippCPUID_SSE2 ) ? 'Y':'N',( emask & ippCPUID_SSE2 ) ? 'Y':'N');
      printf("Intel® Streaming SIMD Extensions 2\n");
      printf("  ippCPUID_SSE3       = ");
      printf("%c\t%c\t",( mask & ippCPUID_SSE3 ) ? 'Y':'N',( emask & ippCPUID_SSE3 ) ? 'Y':'N');
      printf("Intel® Streaming SIMD Extensions 3\n");
      printf("  ippCPUID_SSSE3      = ");
      printf("%c\t%c\t",( mask & ippCPUID_SSSE3 ) ? 'Y':'N',( emask & ippCPUID_SSSE3 ) ?
'Y':'N');
      printf("Supplemental Streaming SIMD Extensions 3\n");
      printf("  ippCPUID_MOVBE      = ");
      printf("%c\t%c\t",( mask & ippCPUID_MOVBE ) ? 'Y':'N',( emask & ippCPUID_MOVBE ) ?
'Y':'N');
      printf("The processor supports MOVBE instruction\n");
      printf("  ippCPUID_SSE41      = ");
      printf("%c\t%c\t",( mask & ippCPUID_SSE41 ) ? 'Y':'N',( emask & ippCPUID_SSE41 ) ?
'Y':'N');
      printf("Intel® Streaming SIMD Extensions 4.1\n");
      printf("  ippCPUID_SSE42      = ");
      printf("%c\t%c\t",( mask & ippCPUID_SSE42 ) ? 'Y':'N',( emask & ippCPUID_SSE42 ) ?
'Y':'N');
      printf("Intel® Streaming SIMD Extensions 4.2\n");
      printf("  ippCPUID_AVX        = ");
      printf("%c\t%c\t",( mask & ippCPUID_AVX ) ? 'Y':'N',( emask & ippCPUID_AVX ) ? 'Y':'N');
      printf("Intel® Advanced Vector Extensions (Intel® AVX) instruction set\n");
      printf("  ippAVX_ENABLEDBYOS  = ");
      printf("%c\t%c\t",( mask & ippAVX_ENABLEDBYOS ) ? 'Y':'N',( emask & ippAVX_ENABLEDBYOS ) ?
'Y':'N');
      printf("The operating system supports Intel® AVX\n");
      printf("  ippCPUID_AES        = ");
      printf("%c\t%c\t",( mask & ippCPUID_AES ) ? 'Y':'N',( emask & ippCPUID_AES ) ? 'Y':'N');
      printf("Intel® AES instruction\n");
      printf("  ippCPUID_SHA        = ");
      printf("%c\t%c\t",( mask & ippCPUID_SHA ) ? 'Y':'N',( emask & ippCPUID_SHA ) ? 'Y':'N');
      printf("Intel® SHA new instructions\n");
      printf("  ippCPUID_CLMUL      = ");
```

```
      printf("%c\t%c\t",( mask & ippCPUID_CLMUL ) ? 'Y':'N',( emask & ippCPUID_CLMUL ) ?
'Y':'N');
      printf("PCLMULQDQ instruction\n");
      printf("  ippCPUID_RDRAND     = ");
      printf("%c\t%c\t",( mask & ippCPUID_RDRAND ) ? 'Y':'N',( emask & ippCPUID_RDRAND ) ?
'Y':'N');
      printf("Read Random Number instructions\n");
      printf("  ippCPUID_F16C       = ");
      printf("%c\t%c\t",( mask & ippCPUID_F16C ) ? 'Y':'N',( emask & ippCPUID_F16C ) ? 'Y':'N');
      printf("Float16 instructions\n");
      printf("  ippCPUID_AVX2       = ");
      printf("%c\t%c\t",( mask & ippCPUID_AVX2 ) ? 'Y':'N',( emask & ippCPUID_AVX2 ) ? 'Y':'N');
      printf("Intel® Advanced Vector Extensions 2 instruction set\n");
      printf("  ippCPUID_AVX512F    = ");
      printf("%c\t%c\t",( mask & ippCPUID_AVX512F ) ? 'Y':'N',( emask & ippCPUID_AVX512F ) ?
'Y':'N');
      printf("Intel® Advanced Vector Extensions 512 Foundation instruction set\n");
      printf("  ippCPUID_AVX512CD   = ");
      printf("%c\t%c\t",( mask & ippCPUID_AVX512CD ) ? 'Y':'N',( emask & ippCPUID_AVX512CD ) ?
'Y':'N');
      printf("Intel® Advanced Vector Extensions 512 Conflict Detection instruction set\n");
      printf("  ippCPUID_AVX512ER   = ");
      printf("%c\t%c\t",( mask & ippCPUID_AVX512ER ) ? 'Y':'N',( emask & ippCPUID_AVX512ER ) ?
'Y':'N');
      printf("Intel® Advanced Vector Extensions 512 Exponential & Reciprocal instruction set\n");
      printf("  ippCPUID_ADCOX      = ");
      printf("%c\t%c\t",( mask & ippCPUID_ADCOX ) ? 'Y':'N',( emask & ippCPUID_ADCOX ) ?
'Y':'N');
      printf("ADCX and ADOX instructions\n");
      printf("  ippCPUID_RDSEED     = ");
      printf("%c\t%c\t",( mask & ippCPUID_RDSEED ) ? 'Y':'N',( emask & ippCPUID_RDSEED ) ?
'Y':'N');
      printf("The RDSEED instruction\n");
      printf("  ippCPUID_PREFETCHW  = ");
      printf("%c\t%c\t",( mask & ippCPUID_PREFETCHW ) ? 'Y':'N',( emask & ippCPUID_PREFETCHW ) ?
'Y':'N');
      printf("The PREFETCHW instruction\n");
      printf("  ippCPUID_KNC        = ");
      printf("%c\t%c\t",( mask & ippCPUID_KNC ) ? 'Y':'N',( emask & ippCPUID_KNC ) ? 'Y':'N');
      printf("Intel® Xeon Phi™ Coprocessor instruction set\n");
   }
   return 0;
}
```

This application consists of two sections:

1. Get the library layer name and version.
2. Show the hardware optimizations used by the selected library layer and supported by CPU.

On Windows* OS, Intel IPP Cryptography applications are significantly easier to build with Microsoft* Visual Studio*. To build the code example above, follow the steps:

1. Start Microsoft* Visual Studio* and create an empty C++ project.
2. Add a new c file and paste the code into it.
3. Set the include directories and the linking model.
4. Compile and run the application.

Training and Documentation

Document	Description
Intel® IPP Cryptography Developer Guide	Provides detailed guidance on Intel IPP Cryptography library configuration, development environment, and linkage modes.
Intel® IPP Cryptography Developer Reference	Contains detailed descriptions of the Intel IPP Cryptography functions.
Intel® Integrated Performance Primitives	Intel® IPP product page. See this page for support and online documentation.

Notices and Disclaimers

Product and Performance Information
Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex. Notice revision #20201201