WindowsBuild

For notes on Installation see this topic

Building the SP consists of downloading and (usually) building the dependencies and then building the Shibboleth components.

This process is entirely performed from the command line using the `nmake` utility.

- Environment
  - Directory layout
  - Environmental variables
  - Obeying the Component rules
- Downloaded External Dependencies
  - Apache
  - FastCgi
- Built External Dependencies
  - Preconditions
  - NMAKE targets
- Building Shibboleth
  - Preconditions
  - NMAKE targets
- Windows Subsystem for Linux (WSL)

Environment

All building is done from within a Visual C command line. The 64 bit builds for the dependencies should be done from a 64 bit environment and likewise the 32 bit.

Both architectures (32/64) for the Shibboleth components are built at the same time and this can be done from either a 32 bit or 64 bit command line..

Directory layout

Everything is built under a common directory referred to as the root directory. The shibboleth products (xmltooling, OpenSAML and the SP) have fixed names (cpp-xmtooling, cpp-opensaml, cpp-sp), and the build is managed from the cpp-msbuild subdirectory. The names of the dependency directories can be controlled by the environmental variables.

Environmental variables

General Environment			
Name	**Default**	**Used to build...**	**Description**
ROOT_DIR	<Mandatory>	everything	The location of the build
SED	sed	ZLIB	The location of the sed command
PERL	perl	OpenSSL	The location of the perl command
Zlib ENVIRONMENT			
Name	**Default**	**Used to build...**	**Description**
ZLIB_DIR	zlib-1.2.11	ZLIB, Shib DLLs, Installer	The name of the directory with the xlib sources
ZLIB_IMPLIB		ZLIB	The name of the zlin library (hence `zlib1.lib`)
ZLIB_MM_VERSION	1.2.11	Installers	The version given to the zlib MergeModule
ZLIB_FILE_VERSION	1_2_11	Installlers	The garnish added to the DLL(hence `zlib1_2_11.dll`). Changing this requires changes to `versions.props`
Log4Shib ENVIRONMENT			
Name	**Default**	**Used to build...**	**Description**
LOG4SHIB_DIR	cpp-log4shib	Log4Shib, Shib DLLs, Installers	The name of the directory with the log4shib sources.
LOG4SHIB_FILE_VERSION	1_0	Installers	The garnish added to the DLL (hence `log4shib1_0D.dll`). Changing this requires changes to `versions.props`

LOG4SHIB_MM_VERSION		Installers	The version given to the Log4Shib MergeModule
OpenSLL ENVIRONMENT			
Name	**Default**	**Used to build...**	**Description**
OPENSSL_DIR	openssl-1.1.0g	OpenSSL All Shib DLLs Installers	The name of the directory with the OpenSSL sources.
OPENSSL_FILE_VERSION	1_1	Installers	The garnish added to the OpenSSL DLLs (hence `libcrypto-1_1.dll`). Changing this requires changes to `versions.props`
OPENSSL_MM_VERSION		Installers	The version given to the OpenSSL MergeModule
Curl ENVIRONMENT			
LIBCURL_DIR	libcurl-7.57.0	Curl, Shib DLLs Installers	The name of the directory with the Curl sources
LIBCURL_VERSION	7.57.0	Installers	The Version of Libcurl (used for paths and Merge Module Version)
Xerces ENVIRONMENT			
XERCES_DIR	xerces-c-3.2.0	Xerces, Shib DLLs Installers	The name of the directory with the Santuario sources
XERCES_MM_VERSION	3.2.0	Installers	The version of Xerces (used for Merge module version)
XERCES_FILE_VERSION	3_2	Installers	The garnish on the DLL name. Hence `xerces-3_2.dll`. Changing this requires changes to `versions.props`
XmlSecurity (Santuario) ENVIRONMENT			
XSEC_DIR	xml-security-cpp	XmlSecurity all Shib DLLs Installers	The name of the directory with the Santuario sources
XSEC_MM_VERSION	2.0.0	Installers	The version of XML Security (used for Merge module version)
XSEC_FILE_VERSION	2_0	Installers	The garnish on the DLL name. Hence `xsec_2_0d.dll.` Changing this requires changes to `versions.props`
Apache ENVIRONMENT			
Name	**Default**	**Used to build...**	**Description**
~~APACHE_13_ROOT~~		~~not used~~	~~Location of Apache 1.3 build tree as downloaded from ApacheLounge~~
~~APACHE_20_ROOT~~		~~mod_shib20.so~~	~~Location of 32 bit Apache 2.0 build tree as downloaded from ApacheLounge~~
APACHE_22_ROOT		mod_shib22.so	Location of 32 bit Apache 2.2 build tree as downloaded from ApacheLounge
APACHE_22_ROOT64		mod_shib22.so	Location of 64 bit Apache 2.2 build tree as downloaded from ApacheLounge
APACHE_24_ROOT		mod_shib24.so	Location of 32 bit Apache 2.4 build tree as downloaded from ApacheLounge
APACHE_24_ROOT64		mod_shib24.so	Location of 64 bit Apache 2.4 build tree as downloaded from ApacheLounge

The build takes configuration from environmental variable set up in the file `cpp-msbuild/dependencies/config.bat.` The complete list of settings can be found from either build system with the target 'environment',

Obeying the Component rules

The component rules require that if the name of a DLL is changed then the component GUID is changed. In terms of the Shibboleth build this means that if any of the environmental variables ending in `_FILE_VERSION` change then you need to edit the Versions.cpp file. This is because this environmental variable is used to control the name of the DLL during the build of the dependencies and during the build of the installer.

The `versions.props` file has code to ensure that the installers will not build if a file version is changed without a change to the GUID for instance. The code contains comments to explain what to do

```
<PropertyGroup Label="CurlGuidsBad" Condition="'$(LIBCURL_FILE_VERSION)' != '7_58'">
  <CurlFileVersion>BAD_LIBCURL_FILE_VERSION</CurlFileVersion>
  <!-- Changing LIBCURL_FILE_VERSION version requires changing the GUIDs since it is garnish on the name -->
</PropertyGroup>
```

In this case, if LIBCURL was updated to version 7.59 then it would be relevant to change `LIBCURL_FILE_VERSION` to `7_59`. This in turn qould require changing the code section above **and** the 4 GUIDs associate with libcurl (`<LibCurlGuid32>, <LibCurlGuid64>, <LibCurlGuid32d>, <LibCurlGuid64d>`). If you don't understand why read up on the file version and component rules. Or just do it.

A GUID generator is shipped as part of Visual Studio. The ones you use need to be in 'registry format'

Downloaded External Dependencies

Apache

Apache should be downloaded (and verified) from Apache Lounge.

- 32 ***and*** 64bit Apache 2.2
- 32 ***and*** 64bit Apache 2.4

The location of each download should be set up as properties as per the above.

FastCgi

FastCGI is not packaged as part of the V3 SP.

Built External Dependencies

The following external dependencies

- zlib
- log4shib
- xerces
- libcUrl
- OpenSSL
- XMLSecurity

are rebuilt under the control of the makefile `dependency.make` in the subdirectory `dependencies` of the **`cpp-msbuild`** project. This make file will build the dependencies specified for the architecture associated with the Visual studio build environment from which it is invoked. For example

```
d:\Program Files (x86)\Microsoft Visual Studio\2017\Professional\VC\Auxiliary\Build\vcvars32.bat
```

or

```
d:\Program Files (x86)\Microsoft Visual Studio\2017\Professional\VC\Auxiliary\Build\vcvars64.bat
```

> ⚠ **One architecture at a time!**
>
> Several of the dependencies are built into common area and then installed into architecture specific directories. It is therefore **not safe** to build both architectures in parallel.

Preconditions

- Download sources as appropriate.
- Edit and execute the `dependencies\config.bat` file. This established the environment as detailed above. It is parameterized to allow for different directory layouts on different machines, while keeping the rest of the configuration common.
- The OpenSSL build configure script requires changes to control the names of the output DLLS.
  - To add the standard 'D' postfix to the Debug DLLs
  - To uniquify the DLL names by subversion number (to protect again feature creep in subversions and thus "DLL hell" with multiple OpenSL distributions)

  We do this by defining new targets in the `Configurations/10-main.conf` file
  - Go into the subdirectory `Configurations`
  - Edit the file `10-main.conf`

- Change the setting VC-WIN64A rmaking `multilib` be "`_<subVersion>-x64`"

Example for Version 1.1.1

```
"VC-WIN64A" => {
    inherit_from     => [ "VC-WIN64-common", asm("x86_64_asm"),
                          sub { $disabled{shared} ? () : "x86_64_uplink" } ],
    AS               => sub { vc_win64a_info()->{AS} },
    ASFLAGS          => sub { vc_win64a_info()->{ASFLAGS} },
    asoutflag        => sub { vc_win64a_info()->{asoutflag} },
    asflags          => sub { vc_win64a_info()->{asflags} },
    sys_id           => "WIN64A",
    bn_asm_src       => sub { return undef unless @_;
                              my $r=join(" ",@_); $r=~s|asm/x86_64-gcc|bn_asm|; $r; },
    perlasm_scheme   => "auto",
    multilib         => "_1-x64",
},
```

- Duplicate the whole setting to a new target "`VC-WIN64AD`" , this time add D to the `multilib`

```
"VC-WIN64AD" => {
    inherit_from     => [ "VC-WIN64-common", asm("x86_64_asm"),
                          sub { $disabled{shared} ? () : "x86_64_uplink" } ],
    AS               => sub { vc_win64a_info()->{AS} },
    ASFLAGS          => sub { vc_win64a_info()->{ASFLAGS} },
    asoutflag        => sub { vc_win64a_info()->{asoutflag} },
    asflags          => sub { vc_win64a_info()->{asflags} },
    sys_id           => "WIN64A",
    bn_asm_src       => sub { return undef unless @_;
                              my $r=join(" ",@_); $r=~s|asm/x86_64-gcc|bn_asm|; $r; },
    perlasm_scheme   => "auto",
    multilib         => "_1D-x64",
},
```

- For the 32 bit build you will need to **add** `multilib` This time the name only gets the sub (3rd digit) version (and the `D` for debug - for a new target `VC-WIN32D`.)

```
"VC-WIN32" => {
    inherit_from     => [ "VC-noCE-common", asm("x86_asm"),
                          sub { $disabled{shared} ? () : "uplink_common" } ],
    CFLAGS           => add("/WX"),
    AS               => sub { vc_win32_info()->{AS} },
    ASFLAGS          => sub { vc_win32_info()->{ASFLAGS} },
    asoutflag        => sub { vc_win32_info()->{asoutflag} },
    asflags          => sub { vc_win32_info()->{asflags} },
    sys_id           => "WIN32",
    bn_ops           => add("BN_LLONG"),
    perlasm_scheme   => sub { vc_win32_info()->{perlasm_scheme} },
    # "WOW" stands for "Windows on Windows", and "VC-WOW" engages
    # some installation path heuristics in windows-makefile.tmpl...
    build_scheme     => add("VC-WOW", { separator => undef }),
    multilib         => "_1",
},
"VC-WIN32D" => {
    inherit_from     => [ "VC-noCE-common", asm("x86_asm"),
                          sub { $disabled{shared} ? () : "uplink_common" } ],
    CFLAGS           => add("/WX"),
    AS               => sub { vc_win32_info()->{AS} },
    ASFLAGS          => sub { vc_win32_info()->{ASFLAGS} },
    asoutflag        => sub { vc_win32_info()->{asoutflag} },
    asflags          => sub { vc_win32_info()->{asflags} },
    sys_id           => "WIN32",
    bn_ops           => add("BN_LLONG"),
    perlasm_scheme   => sub { vc_win32_info()->{perlasm_scheme} },
    # "WOW" stands for "Windows on Windows", and "VC-WOW" engages
    # some installation path heuristics in windows-makefile.tmpl...
    build_scheme     => add("VC-WOW", { separator => undef }),
    multilib         => "_1D",
},
```

NMAKE targets

clean

Cleans all build environments

all

Builds all targets

per-dependency targets

Each of the 6 dependencies (**openssl**, **log4shib**, **zlib**, **xerces**, **curl** and **xmlsec**) has four targets associated

- **nmake /f dependency.make <component>** builds both the release and debug installation of the appropriate architecture for **<component>** and any of its dependencies.
- **nmake /f dependency.make <component>-clean** cleans the build environment (both architectures) for **<component>**
- **nmake /f dependency.make <component>-debug** performs the debug build and installation of the appropriate architecture **<component>** and any of its dependencies.
- **nmake /f dependency.make <component>-release** (if not there) performs the release build and installation of the appropriate architecture for **<component>** and any of its dependencies.

In addition

- **openssl-build** builds the debug installation of the appropriate architecture, but does **NOT** rebuild the configuration or do a distclean. Appropriate when working on OpenSSL development (as if)

Building Shibboleth

A centralized makefile `cpp-msbuild/build.make` allows all the Shibboleth components to be build (cpp-xmltooling, cpp-saml, cpp-sp)

In contrast to the dependency build this nmake files makes both architectures (X64 and ia386)

The final outcome of a build is usually the two installers.

Preconditions

- Download or build the depenencies as per above
- Set the Environmental variables described above.
- The Shibboleth only "external" environment is defined in the `$(BuildRoot)cpp-msbuild/versions.props` file. Create this and edit it appropriately based on this template

```
<Project DefaultTargets="Build" ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild
/2003">
    <PropertyGroup Label="PathMacros">
        <BuildRoot>h:\perforce\VS2017\</BuildRoot>
        <Apache22Root>$(BuildRoot)apache\VC10\Apache2</Apache22Root>
        <Apache22Root64>$(BuildRoot)apache\VC10\Apache22-64</Apache22Root64>
        <Apache24Root>$(BuildRoot)apache\VC15\Apache24-32\Apache24</Apache24Root>
        <Apache24Root64>$(BuildRoot)apache\VC15\Apache24-64\Apache24</Apache24Root64>
    </PropertyGroup>
</Project>
```

- Check the `Versions.props` files for any required changes. These will include
    - changes to GUIDs as `FILE_VERSIONS` changes
    - Boost versions
    - and so forth.
- As noted above, the installers will fail to build if the FILE_VERSION has been changed and appropriate changes have not been made in this file.

NMAKE targets

- **clean**
- **exe32** - build 32 bit executables (all DLLs and exe files)
- **exe64** - build 64 bit executables (all DLLs and exe files)
- **all** - build both installers

Rebuilding individual projects is usually better achieved from within Visual Studio (which also takes configuration from `$(BuildRoot)buildpath.props` and `$(BuildRoot)cpp-msbuild//versions.props`.

Ensure that the environment is setup before starting Visual Studio

```
h:\Perforce\VS2017>cpp-msbuild\dependencies\config.bat
h:\Perforce\VS2017>cpp-sp\Projects\vc15\Shibboleth.sln
```

Windows Subsystem for Linux (WSL)

The SP has been built using WSL (largely to test the Linux build environment without a Linux machine), but this is not supported. The Linux build instructions should be followed

If for whatever reason you need to use WSL the following helps in the run up to the `configure` stage.

```
autoreconf -fvi
autoconf
```