YOUR VOICE HAS NO LIMIT

Compiling VSDK samples with Conan

Compiling VSDK samples with Conan

To simplify its C++ development, Vivoka works with the Conan package manager. You can find its official documentation here.

Install Conan

Conan is made with Python 3, so:

1. To install Python 3 you can either use your package manager (Linux) or download it (Windows). You may need to add paths to the environment depending on how you install it.
2. Then, use `[sudo] pip install conan`. If you have any trouble, please follow the official tutorial.
3. Make sure its version is `>= 1.40.3` by running `conan --version`. If it's not, please run `[sudo] pip install -U conan`.

If you ever get an error like `[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: certificate has expired`, that means your version is outdated!

Configure Conan

We have a custom Conan server at Vivoka to deliver our SDK and its dependencies. To configure it you need to register the remote server with the following commands:

```
$ conan profile new default --detect
$ conan remote add vivoka-customer https://conan-customer.vivoka.com/artifactory/api/conan/vivoka-customer
$ conan config set general.revisions_enabled=1
```

Please double check that your PDF reader didn't remove any characters like spaces or hyphens during copy. Also if that last step triggers errors please have a look at this.

Now let's add your user and password (those are the ones you used to access your VDK download):

```
$ conan user <username> -r vivoka-customer -p
Please enter your password: <password>
```

You can set the password in the `CONAN_PASSWORD_VIVOKA_CUSTOMER` environment variable if you want to avoid inputting it regularly.

Linux

Most likely Conan will emit a warning about the version of your `libstdc++`, so run this command:

```
$ conan profile update settings.compiler.libcxx=libstdc++11 default
```

Windows

VDK 3 supports MSVC 15 and 16. To be sure that Conan detected a compatible version, print it:

```
$ conan profile get settings.compiler.version default
```

CMake

This is our build system, version `>= 3.13` is needed. You can either get it via package manager, installer, or use Conan.

1. Package Manager

This is mainly for Unix distributions as Windows has no official package manager:

```
# Debian/Ubuntu Distributions
$ sudo apt install cmake
# Fedora Distribution
$ sudo dnf install cmake
```

2. Installer

1. Download from here ;
2. (Windows): Extract in `C:\CPP\cmake` ;
3. (Windows): Add `C:\CPP\cmake\bin` to your `PATH`.

3. Conan

This method is the less obtrusive way but will not add the binaries to your `PATH` by default!

Edit your default conan profile (located in `~/.conan/profiles` or `%userprofile%\.conan\profiles`) and under `[build_requires]`, add `cmake/[~3.21]`:

```
[build_requires]
cmake/[~3.21]
[env]
```

Compile samples

Go to the root of the sample and execute these commands:

```
# First install the dependencies, copy DLLs, etc
$ conan install . -if build -b missing
# Build the program
$ conan build . -bf build
```

Execute samples

Linux:

```
$ . ./build/activate_run.sh # this temporarily exports LD_LIBRARY_PATH for you
$ ./build/bin/<executable>
$ . ./build/deactivate_run.sh # unexport LD_LIBRARY_PATH
```

Windows (cmd.exe):

```
$ .\build\activate_run.bat
$ .\build\bin\<executable>.exe
$ .\build\deactivate_run.bat
```

Windows (PowerShell):

```
$ .\build\activate_run.ps1
$ .\build\bin\<executable>.exe
$ .\build\deactivate_run.ps1
```