Tel: +27 11 343 2000 | **Fax**: +27 11 442 5908 | **Email**: info@prism.co.za
Address: President Place, Johannesburg, South Africa

Guide to Using Thrift

2018-10-26

Document number:	PR-D2-0984 Rev 1.0.1
Release date:	2018-10-26
Prepared by:	TrevorD
Copyright:	© 2018 Prism Payment Technologies
Synopsis:	This document is a guide to setting up an environment to develop clients for Thrift services, with examples for Prism's UVS/STSVS API.

Contents

1 Introduction

"Thrift is an interface definition language and binary communication protocol that is used to define and create services for numerous languages. It is used as a remote procedure call (RPC) framework" – Wikipedia:Apache Thrift

This document is a guide to setting up an environment to develop clients for Thrift services.

1.1 Thrift in Prism products

Prism's Utility Vending System (UVS) is programmable via a Thrift API. The document PR-D2-0975 "STS Vending Subsystem API" describes the API in detail. Prism also supplies a Thrift IDL file for the services provided by the STSVS.

1.2 Overview of client development with Thrift

The exact process to set up an environment to develop Thrift clients depends on your development language, but the general process is as follows:

- Download and Install Thrift (section 2). This will give you access to the Thrift sources (including the runtime libraries for your language) and the Thrift compiler.
- Download and install additional libraries required by the Thrift runtime (language-dependent).
- Generate client stub code using the Thrift compiler. The client stub is a marshalling layer that makes calling the remote Thrift service as easy as invoking a local function/method. The generated code references functions in the Thrift runtime libraries.
- Write a client application that incorporates or references the client stub code to invoke functions of the Thrift service.
- Build your client application, linking against the Thrift runtime libraries (and additional libraries, if required).

1.3 Conventions in this document

- Our **working directory** is `C:\User\`. All source code and libraries are located under this directory. You should select an appropriate directory for your development environment.
- We install third party libraries (such as Thrift) under `C:\User\ThirdParty`.

1.4 Tools & References

- Some software is distributed as compressed archives (e.g. `.zip` or `.tar.gz`). You will need an archiving tool such as PeaZip (from http://www.peazip.org/) to extract these archives.

2 Install Thrift

2.1 Install Thrift sources

- Download Thrift from https://thrift.apache.org/. We have used Thrift v0.9.3, released 2015-10-06.
- Use an archive tool (e.g. PeaZip) to extract `thrift-0.9.3.tar.gz` to `C:\User\ThirdParty\thrift-0.9.3\` (so that the `CHANGES` file is in that folder).

2.2 Download or build Thrift compiler

- Download the matching version of the Thrift compiler (`thrift-0.9.3.exe`) or build the compiler (section 3).
- Save (or build) the executable file as `C:\User\ThirdParty\thrift-0.9.3\compiler\thrift.exe`.

2.3 Generate client stubs from IDL

Check your Thrift installation and compiler by building a sample IDL (`.thrift`) file:

- In a Command Prompt:

```
cd C:\User\ThirdParty\thrift-0.9.3\tutorial
..\compiler\thrift.exe --gen java tutorial.thrift
```

- The generated files should be in `C:\User\ThirdParty\tutorial\gen-java\`.

You can also generate client & server stubs for other languages:

Language	Compiler option	Output folder
Java	`--gen java`	`gen-java\`
C#	`--gen csharp`	
Python	`--gen py`	
JavaScript	`--gen js` `--gen js:node`	
C++	`--gen cpp`	`gen-cpp\`
Other	Consult the Thrift documentation at https://thrift.apache.org/docs/.	

3 Build the Thrift compiler

We recommend that you download the pre-built compiler (Win32 EXE) from Apache. Refer to section 2.2.

4 Thrift client in C++

Overview:

- Install Thrift (section 2).
- Download, install, and build Boost (section 4.1). Boost is a third party library used by the Thrift runtime libraries.
- Generate the client stubs for C++ (section 4.2).
- Write a client application (4.3) that incorporates the stub code and links against the Thrift libraries and Boost.

You must use the same C++ toolchain (compiler & linker) to build Boost, the Thrift runtime libraries, the client stubs, and your application.

4.1 Install Boost

- Download Boost from http://www.boost.org/. We have used Boost 1.59.0, released 2015-08-13.
- Use an archive tool (e.g. PeaZip) to extract `boost_1_59_0.zip` to `C:\User\ThirdParty\boost_1_59_0\` (so that the `INSTALL` file is in that folder).
- Thrift v0.9.3 makes use of the Boost.Thread library, for which it is necessary to build Boost. Some older versions of Thrift only use the Boost headers. You can skip the build step, and complete it later if your application fails to link.

4.1.1 Build Boost

- In a command prompt that is configured for your C++ compiler environment:

```
cd C:\User\ThirdParty\boost_1_59_0\
bootstrap.bat
.\b2.exe --with-thread --with-date_time
```

  - *We use Microsoft Visual C++; to open a command prompt use Windows Start → All Programs → Microsoft Visual Studio 2012 → Visual Studio Tools → Developer Command Prompt for VS2012.*
  - *You can omit the '--with' parameters to build all Boost libraries.*

- At the end of the build the `b2.exe` application will output a list of directories; you should note these for later use.

```
The Boost C++ Libraries were successfully built!

The following directory should be added to compiler include paths:
    C:\User\ThirdParty\boost_1_59_0

The following directory should be added to linker library paths:
    C:\User\ThirdParty\boost_1_59_0\stage\lib
```

- For more information consult the Boost "Getting Started on Windows" document at http://www.boost.org/doc/libs/1_59_0/more/getting_started/windows.html.

4.2 Generate client stubs

- Make a folder `C:\User\demo\StsvsCpp`.
- Copy `stsvs.thrift` (from the STSVS API distribution) into `C:\User\demo\StsvsCpp`.
- In a Command Prompt:

```
cd C:\User\demo\StsvsCpp
C:\User\ThirdParty\thrift-0.9.3\compiler\thrift.exe --gen cpp stsvs.thrift
```

- The generated files should be in `C:\User\demo\StsvsCpp\gen-cpp\`.

4.3 Sample STSVS client in MS Visual C++

We developed this sample application using Microsoft Visual Studio 2012 (Visual C++).

The sample connects to an STSVS service (IP address 10.11.12.22, port 9100), calls the `ping()` function to test connectivity, then calls the `vendCredit()` function to generate an STS credit token.

We present the C++ source code for the client application, followed by detailed instructions to set up the project.

4.3.1 Source code

```
// Project & system headers
#include "stdafx.h"
#include <stdio.h>

// Thrift client stubs for STSVS
#include <Stsvs.h>

// Thrift transports & protocols (from the Thrift runtime library)
#include <thrift/transport/TSocket.h>
#include <thrift/transport/TBufferTransports.h>
#include <thrift/protocol/TBinaryProtocol.h>
#include <thrift/protocol/TJSONProtocol.h>

// For convenience pull all definitions into the global namespace
using namespace apache::thrift;
using namespace apache::thrift::protocol;
using namespace apache::thrift::transport;
using namespace PrismUvs;
```

```
int _tmain(int argc, _TCHAR* argv[])
{
  // Create an STSVS client that uses TBinaryProtocol over a TFramedTransport
  // to a TSocket (TCP/IP) endpoint.
  boost::shared_ptr<TSocket> socket(new TSocket("10.11.12.22", 9100));
  boost::shared_ptr<TTransport> transport(new TFramedTransport(socket));
  boost::shared_ptr<TProtocol> protocol(new TBinaryProtocol(transport));
  StsvsClient client(protocol);

  // Open the transport and ping the server
  transport->open();
    printf("client.ping()\n");
  std::string str;
  client.ping(str, 0, "Hello");
    printf("Done: %s\n", str.c_str());

  // Vend an STS credit token, and dump the result as a JSON array
    printf("client.vendCredit()\n");
  PrismUvs::VendResult vr;
  client.vendCredit(vr, "", "600727150000000085000002079050100 11", 100.0, "ZAR", 0, 1,
0, -1);
    boost::shared_ptr<TMemoryBuffer> mb(new TMemoryBuffer());
    boost::shared_ptr<TJSONProtocol> jp(new TJSONProtocol(mb));
    vr.write(jp.get());
    printf("Done: %s\n", mb->getBufferAsString().c_str());

  // We're done; clean up
  transport->close();
  return 0;
}
```

4.3.2 Visual Studio project

Perform the following steps in Microsoft Visual Studio 2012.

4.3.2.1 Create Thrift client application "StsvsCpp"

- Create a new Win32 Console Application called "StsvsCpp", located in `C:\User\demo\`.
  - File → New → Project
    - Installed → Templates → Visual C++ → Win32, Win32 Console Application:
      - Name: `StsvsCpp`
      - Location: `C:\User\demo\`
    - In the Application Wizard, Application Settings page:
      - [select] Console application
      - [Uncheck "Precompiled Header"]
      - [Uncheck "Security Development Lifecycle checks"].
- Add the Thrift runtime libraries (project 'libthrift') to the solution.
  - File → Add → Existing Project, select `C:\User\ThirdParty\thrift-0.9.3\lib\cpp\libthrift.vcxproj`.
- Make the 'StsvsCpp' project dependent on 'libthrift'.

- Project → Project Dependencies … ; on the Dependencies tab select Project: `StsvsCpp` and Depends on: [check] `libthrift`.

4.3.2.2 Set up and build libthrift

- Add Boost include path (`C:\User\ThirdParty\boost_1_59_0`) to 'libthrift', and correct the project's tool chain (to build with VS2012 v110) (**Configuration: All Configurations**).
  - In Solution Explorer select the 'libthrift' project then: Project → Properties
    - Configuration: All Configurations; Configuration Properties → VC++ Directories. In the "Include Directories" setting replace $(BOOST_ROOT) with `C:\User\ThirdParty\boost_1_59_0` (the compiler include path output by Boost after the `b2.exe` build).
      - Alternatively set the environment variable `BOOST_ROOT=C:\User\ThirdParty\boost_1_59_0` and restart Visual Studio.
    - Configuration: All Configurations; Configuration Properties → General. Set the Platform Toolset to `Visual Studio 2012 (v110)`. *This must match the toolset for project 'StsvsCpp', and that used to build Boost.*
- Fix miscellaneous errors in the 'libthrift' project:
  - Exclude transport/TSSLSocket.cpp from the build (**Configuration: All Configurations**).
    - To exclude a file from the build: In Solution Explorer expand the project, then navigate to the file; right-click on the file and select Properties; choose Configuration: All Configurations; on the Configuration Properties → General page set 'Excluded From Build' to `Yes`.
  - Remove Thrift.cpp from the project.
    - To remove a file from the project: In Solution Explorer expand the project, then navigate to the file; right-click on the file and select 'Exclude from Project'.
  - Add the following source files to the project (source files are located under `C:\User\ThirdParty\thrift-0.9.3\lib\cpp\src\thrift\`):
    - `TOutput.cpp`
    - `async/TConcurrentClientSyncInfo.cpp` (into folder `async`)
    - `protocol/TProtocol.cpp` (into folder `protocol`)
    - To add an existing file to the project: In Solution Explorer right-click on the project (or destination folder within the project) and select Add → Existing Item. Browse for the file.
- Build the 'libthrift' project.
  - In Solution Explorer right-click the project and select Project Only → Build Only libthrift.

4.3.2.3 Set up and build StsvsCpp

- Add the following include paths to project 'StsvsCpp' (**Configuration: All Configurations**):
  - `C:\User\ThirdParty\boost_1_59_0` (as reported by the Boost build)
  - `C:\User\ThirdParty\thrift-0.9.3\lib\cpp\src`
  - `..\gen-cpp`

- Add the following library paths to project 'StsvsCpp' (**Configuration: All Configurations**):
  - `C:\User\ThirdParty\boost_1_59_0\stage\lib` (as reported by the Boost build)
  - `$(OutDir)` (literally that)
  - The "Library Directories" setting is in project Properties → Configuration Properties → VC++ Directories.
- Add 'libthrift.lib' as an additional linker dependency of project 'StsvsCpp' (**Configuration: All Configurations**).
  - Add `libthrift.lib` to the 'Additional Dependencies' list under project Properties → Configuration Properties → Linker → Input.
- Add the following source files to the project (under folder 'Source Files'):
  - `gen-cpp\Stsvs.cpp`
  - `gen-cpp\stsvs_constants.cpp`
  - `gen-cpp\stsvs_types.cpp`
- Copy the source code (section 4.3.1) into the file `StsvsCpp.cpp` (which was generated for you by Visual Studio).
- Build the solution.
  - If you get linker errors relating to a 'lib_boost' library, you may need to build Boost (section 4.1.1).

5 Amendment History

Version	Description	Person	Date
1.0	Basic document with C++ client description.	TD	2015-11-24
1.0.1	Updated template	TD	2018-10-26