**Introduction**

**Design Goals**

The NDN Common Client Libraries (CCL) provide a common application programming interface (API) across several languages for building applications that communicate using Named Data Networking (NDN), with these goals:

- Encourage development and experimentation with NDN for a larger audience of developers
- Multiple platforms and languages: C++ (std or boost), Python (version 2 and 3), JavaScript (browser and Node.js) and Java
- Consistent, stable API across all languages
- Minimal dependencies or assumptions about threading/memory management for easier integration with applications
- Provide install packages so applications can deploy easily
- Track updates to message protocols and the TLV wire format
- Incorporate advances from NDN research projects as library modules to speed adoption by applications (Sync, etc.)

**Example Code**

Following are examples of the CCL API in several programming languages. The sample creates a Face, calls expressInterest, then loops calling processEvents. The onData callback displays the data packet’s Name and content. We also handle the onTimeout callback.

**JavaScript (ndn.js for Node.js)**

```javascript
var Face = require('./').Face;
var Name = require('./').Name;

var onData = function(interest, data) {
    console.log("Name: "+ data.getName().toUri());
    console.log("Content: "+ data.getContent().toBuffer().toString());
};

var onTimeout = function(interest) {
    console.log("Interest timed out: "+ interest.getName().toUri());
};

var face = new Face({host: 'memoria.ndn.ucla.edu'});
face.onData(onData);
face.onTimeout(onTimeout);
```

// Don’t need a processEvents loop. JavaScript is already asynchronous.

**Java (jnd)**

```java
public void onData(Name interest, Data data) {
    ++callbackCount;
    System.out.println("Name: " + data.getName().toUri());
    System.out.println("Content: " + data.getContent().toBuffer().toString());
}

public void onTimeout(Name interest) {
    ++callbackCount;
    System.out.println("T/O: " + interest.getName().toUri());
}
```

**C++ (.cpp)**

```c++
#include <iostream>
#include <ndn-cpp/face.hpp>
using namespace ndn;

class Counter {
public:
    int callbackCount;
    Counter() { callbackCount = 0; }
    void onData(const ptr::shared_ptr<const Interest>& interest, const ptr::shared_ptr<Data>& data) {
        ++callbackCount;
        std::cout << "Name: " << data->getName().toUri() << std::endl;
        std::cout << "Content: " << data->getContent() << std::endl;
    }
    void onTimeout(const ptr::shared_ptr<const Interest>& interest) {
        ++callbackCount;
        std::cout << "T/O: " << interest->getName().toUri() << std::endl;
    }
};

int main(int argc, char** argv)
{
    Face face("memoria.ndn.ucla.edu");
    Counter counter;
    Name name("/ndn/edu/ucla/remap/ndn-js-test/howdy.txt");
    face.expressInterest(name, bind(counter.onData, &counter, _1), bind(counter.onTimeout, &counter, _1));
    while (counter.callbackCount < 1) {
        face.processEvents();
        usleep(10000);
    }
    return 0;
}
```

**Python (PyNDN)**

```python
import time, trollius as asyncio
from pyndn import ThreadsafeFace, Name

class Counter(object):
    def __init__(self):
        self.callbackCount = 0
    def onData(self, interest, data):
        self.callbackCount += 1
    def onTimeout(self, interest):
        self.callbackCount += 1

face = asyncio.Task(self.callbackCount)
```

**C# (ndn-dot-net) in development**

```csharp
using System.Diagnostics;
using System.Threading;

public static void Main(string[] args) {
    try {
        Face face = new Face("memoria.ndn.ucla.edu");
        Counter counter = new Counter();
        Name name = new Name("/ndn/edu/ucla/remap/ndn-js-test/howdy.txt");
        face.expressInterest(name, counter, counter);
        while (counter.callbackCount < 1) {
            face.processEvents();
            System.Threading.Thread.Sleep(5);
        }
    } catch (Exception e) {}}
```

**NDN Common Client Libraries API**

Jeff Thompson, Jeff Burke
Center for Research in Engineering, Media and Performance (REMAP)
University of California, Los Angeles (UCLA)