On C++, Javascript and WebSockets

Alex Fabijanic


alex@pocoproject.org


@0x00FA


aleks-f

Contents

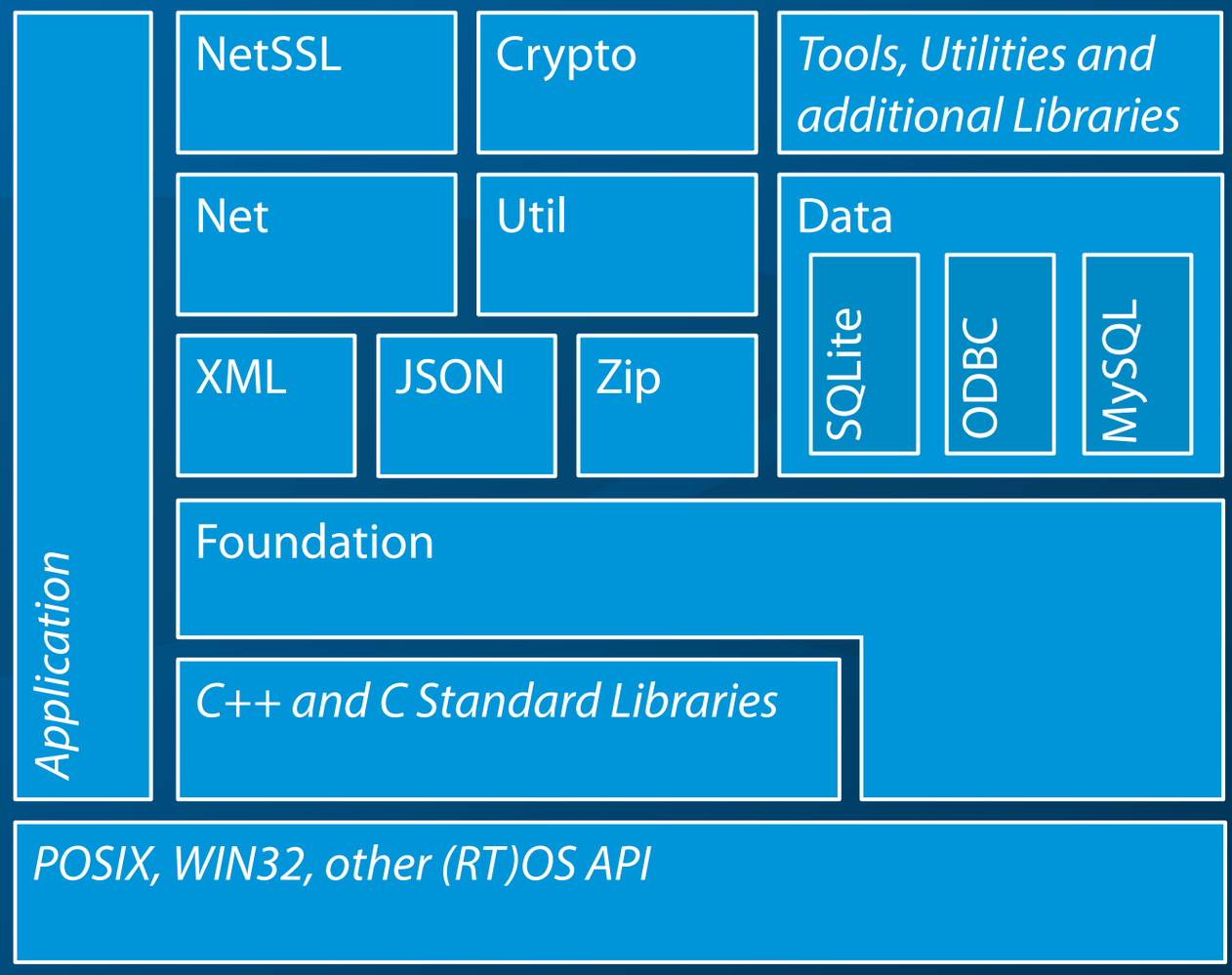
- POCO
- Remoting
- OSP
- Javascript integration
- WebSockets
- macchina.io

POCO C++ Libraries

- A collection of C++ class libraries, conceptually similar to the Java Class Library, the .NET Framework or Apple's Cocoa.
- Focused on solutions to frequently-encountered practical problems.
- Focused on 'internet-age' network-centric applications.
- Written in 100% ANSI/ISO Standard C++.
- Based on and complementing the C++ Standard Library/STL.
- Highly portable and available on many different platforms.
- Open Source, licensed under the Boost Software License.

POCO C++ Libraries

- Started 2004
- ~300.000 LOC
- 1000+ classes
- on GitHub since 2012
  1000+ stars
  400+ forks
  30-50 clones/day
- ~100 contributors
- Boost License
- http://pocoproject.org



Remoting

- C++ implementation, similar to .NET Remoting or Java RMI
- Code generator parses annotated C++ header files and generates code (serialization/deserialization, method dispatching, helpers)
- Supports different transports (binary TCP, SOAP, JSON-RPC)
- Used for automatic C++ - to - JavaScript bridging

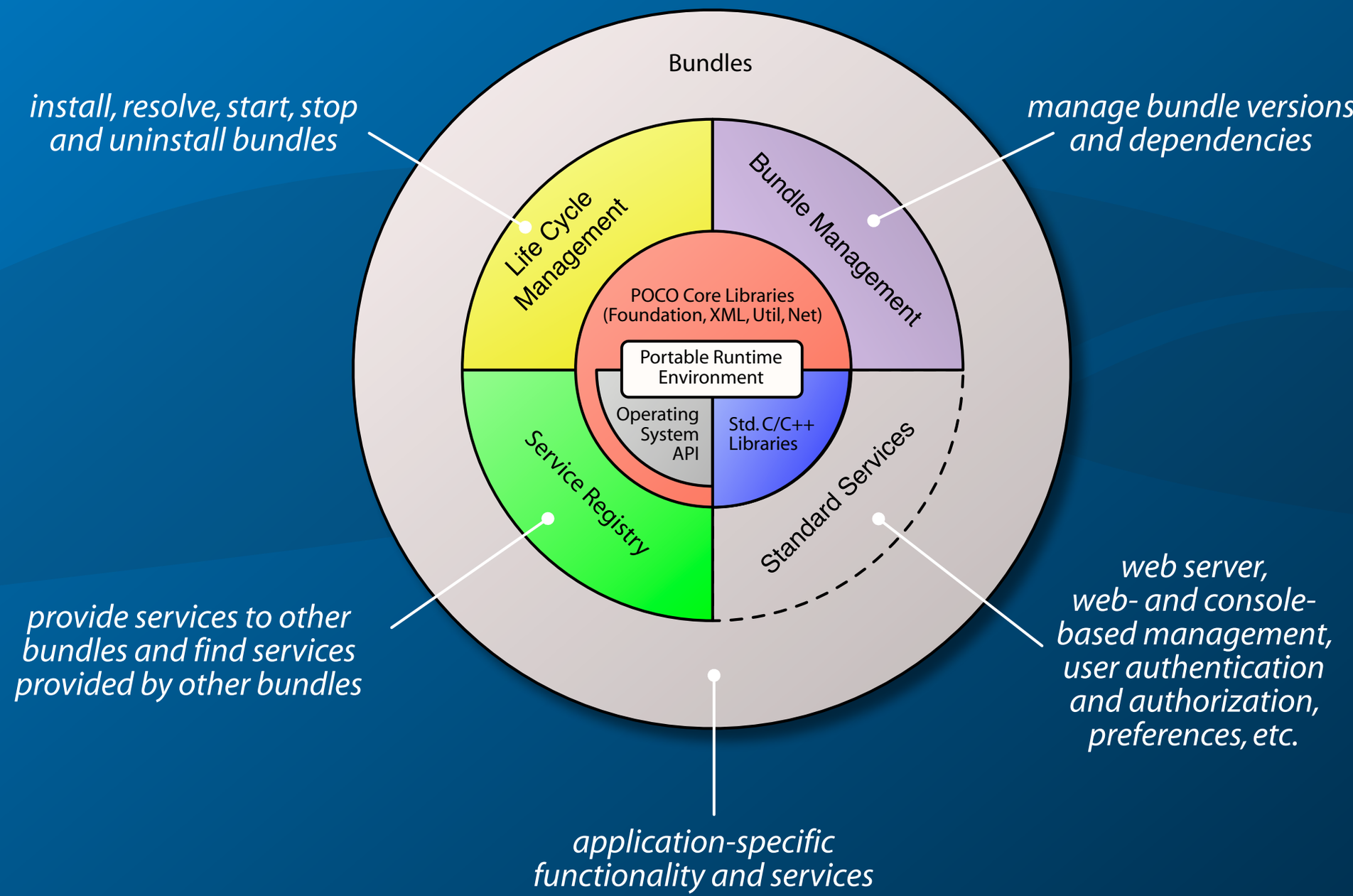
V8

- Google's JavaScript Engine
- Used by Chrome/Chromium and node.js
- C++ library, relatively easy to integrate and extend
- Compiles JavaScript to native code (x86, ARM, MIPS)
- Great performance
- BSD License

Open Service Platform (OSP)

- OSGi inspired C++ implementation
- Dynamic module system based on bundles
  (zip files containing metadata, shared libs, etc.)
- Dependency and lifecycle management
- Services and service registry
- Web Server

Open Service Platform (OSP)



OSP & friends

Platform

JavaScript

V8 JavaScript engine and C++ bindings/bridging

Remoting

serialization, remote methods and events, IPC

Open Service Platform

dynamic module system
service registry
web application server
user authentication/authorization

POCO C++ LIBRARIES

platform abstraction, multithreading, XML and JSON processing, filesystem access, stream, datagram and multicast sockets, HTTP server and client, SSL/TLS, etc.

Combining POCO C++ Libraries and V8

- JavaScript is single-threaded and garbage-collected
- POCO is multithreaded
- Make C++ object available to JavaScript
  easy for static objects, just provide Wrapper
- Allow JavaScript code to create C++ objects
  easy if you don't care about memory/resource leaks
- Register a callback function called by GC when object is deleted
  allows you to properly delete underlying C++ object
- However, V8 does not do callbacks when script ends
  wrapped C++ objects won't be deleted, leaks resulting
- Need to track every C++ object a script creates and clean up afterwards :-(

Remoting

```
// TimeService.h

// expose whole class (public members)
//@ remote
class TimeService
{
public:
    TimeService();
    ~TimeService();

    std::string currentTime() const;
};
```

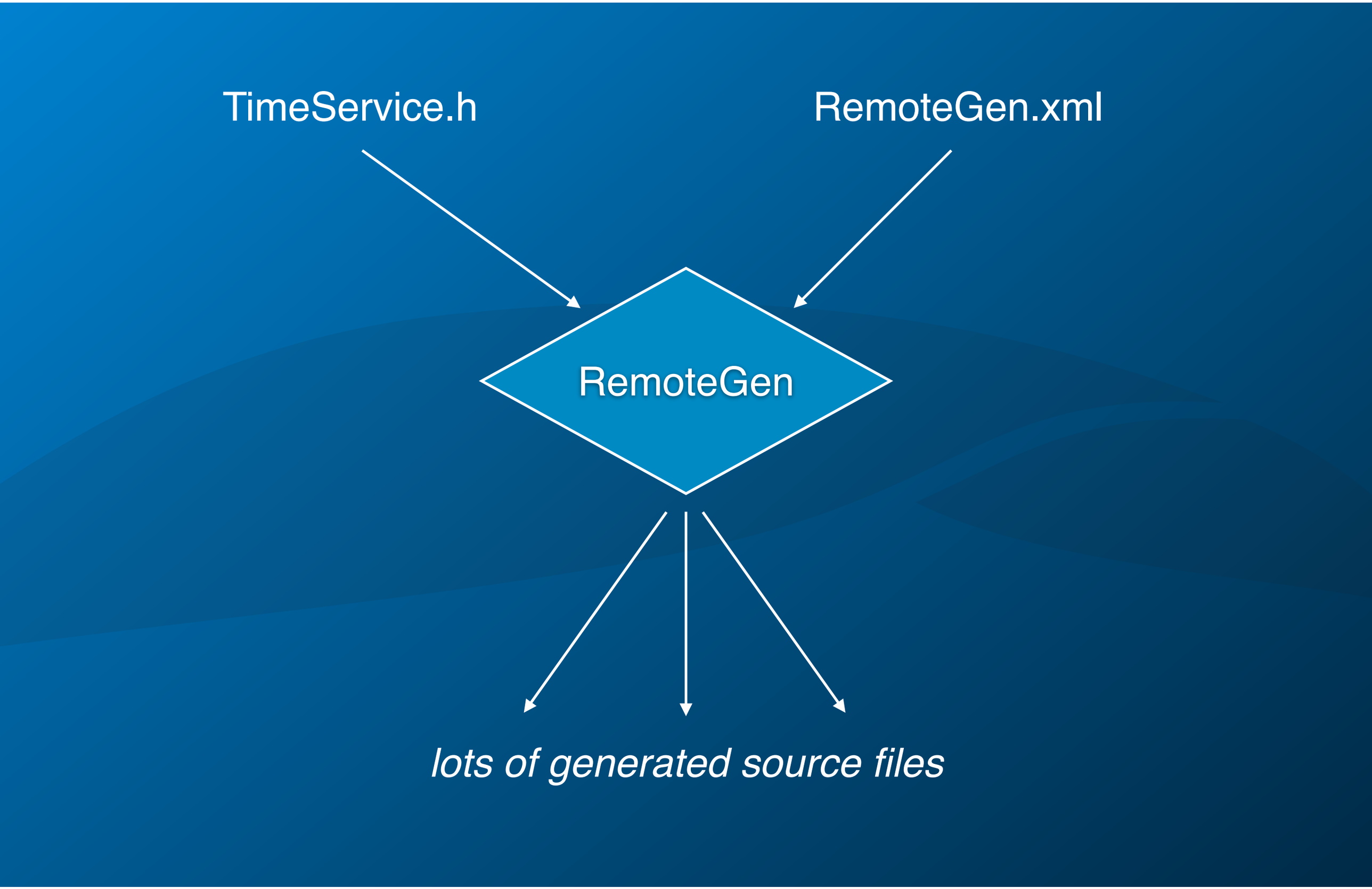
```
// OR


// expose individual functions
class TimeService
{
public:
    TimeService();
    ~TimeService();

    //@ remote
    std::string currentTime() const;
};
```

RemoteGen.xml

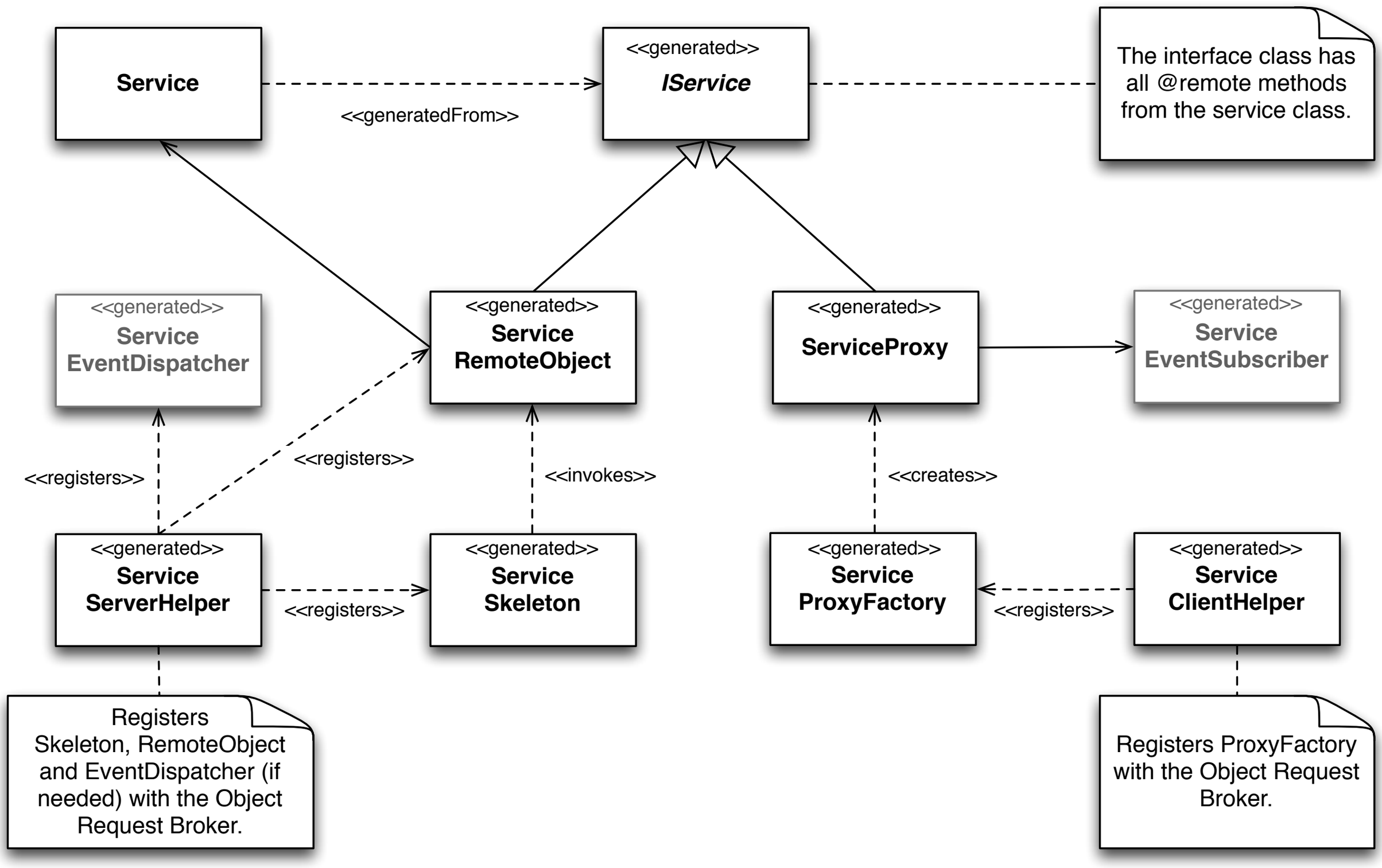
```
<AppConfig>
    <RemoteGen>
        <files>
            <include>
                ${POCO_BASE}/RemotingNG/include/Poco/RemotingNG/RemoteObject.h
                ${POCO_BASE}/RemotingNG/include/Poco/RemotingNG/Proxy.h
                ${POCO_BASE}/RemotingNG/include/Poco/RemotingNG/Skeleton.h
                include/TimeService.h
            </include>
        </files>
        <output>
            <mode>server</mode>
            <include>include</include>
            <src>src</src>
            <namespace>Sample</namespace>
            <copyright>Copyright (c) 2012</copyright>
        </output>
        <compiler>
            <exec>cl</exec>
            <options>
                /I "${POCO_BASE}/RemotingNG\Foundation\include"
                /I "${POCO_BASE}/RemotingNG\RemotingNG\include"
                /nologo
                /C
                /P
                /TP
            </options>
        </compiler>
    </RemoteGen>
</AppConfig>
```

Other compilers

```
<compiler id="gcc">
   <exec>g++</exec>
   <options>
       -I${POCO_BASE}/RemotingNG/Foundation/include
       -I${POCO_BASE}/RemotingNG/include
       -I./include
       -E
       -C
       -o%.i
    </options>
</compiler>
<compiler id="clang">
    <exec>clang++</exec>
    <options>
       -I${POCO_BASE}/Foundation/include
       -I${POCO_BASE}/RemotingNG/include
       -I./include
       -E
       -C
       -xc++
       -o%.i
    </options>
</compiler>
```


TimeService.h
RemoteGen.xml
RemoteGen
lots of generated source files

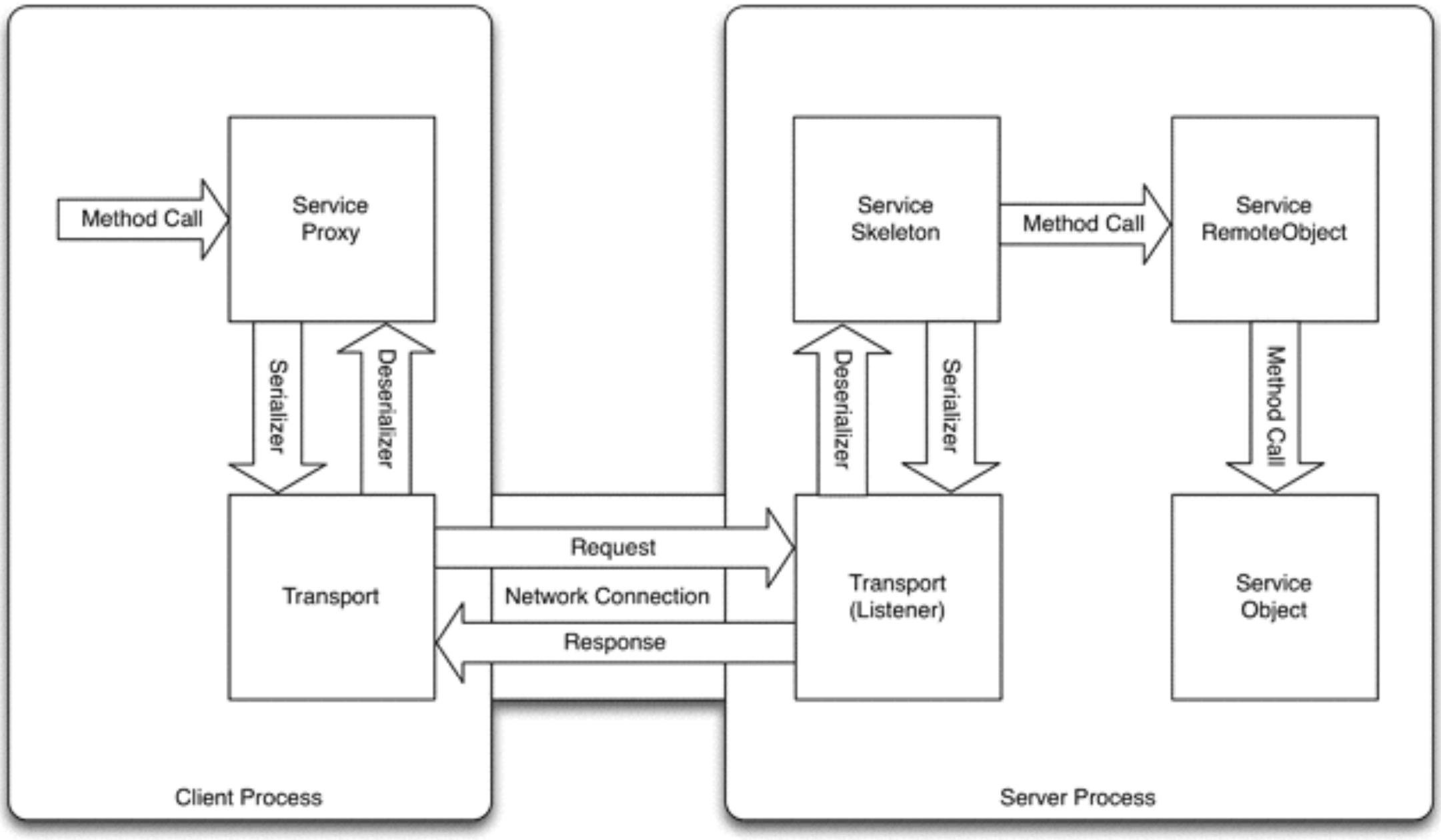
Generated Classes



Parent for Proxy and RemoteObject

```
class ITimeService: public virtual Poco::RefCountedObject
{
public:
    typedef Poco::AutoPtr<ITimeService> Ptr;

    ITimeService();
    virtual ~ITimeService();

    virtual std::string currentTime() const = 0;

    static const Poco::RemotingNG::Identifiable::TypeId& remoting__typeId();
};
```

How a Remote Method Call Works



Usage Example

```
std::string uri(“http://server:8080/soap/TimeService/TheTimeService");

ITimeService::Ptr pTimeService = TimeServiceClientHelper::find(uri);

std::string time = pTimeService->currentTime();
```

Automatic JavaScript Wrappers for C++ Objects

```
// Sensor.h

//@ remote
class Sensor: public Device
{
public:
    Poco::BasicEvent<const double> valueChanged;

    virtual double value() const = 0;

    virtual bool ready() const = 0;
};
```

```
exports.findTempSensor = function()
{
  var tempRefs = serviceRegistry.find('physicalQuantity == "temperature"');
  if (tempRefs.length > 0)
     return tempRefs[0].instance();
  else
     return null;
};

var tempSensor = findTempSensor();
```

```
tempSensor.on(‘valueChanged', function(ev) {
    var temp = ev.data;
    // ...
});

if (tempSensor.ready())
{
    var temp = tempSensor.value();
    // ...
}
```

Web Event Service

WebEvent at a Glance

- *Publish-subscribe* pattern
- Notifications through RFC 6455 *WebSocket* or *OS Event*
- Transparent in-process and over-the-net asynchronous communication
- Transparent *server <=> client* or *client <=> client* communication
- Implements PING/PONG for heartbeat check at application level

WebEvent Protocol

```
SUBSCRIBE <subjectList> WebEvent/1.0

UNSUBSCRIBE <subjectList>|* WebEvent/1.0

NOTIFY <subjectName> WebEvent/1.0\r\n
<data>
```

Subscriptions are hierarchical, e.g.:

`org.poco.events` will receive "child" subjects:

`org.poco.events.someEvent`

`org.poco.events.anotherEvent`

WebEvent Code - Server Side

```
ServiceRef::Ptr pWebEventSvcRef =
_pContext>registry().findByName(WebEventService::SERVICE_NAME);

WebEventService::Ptr pWebEventService =
pWebEventServiceRef->castedInstance<WebEventService>();

Poco::BasicEvent<WebNotificationEvent>& notification =
pWebEventService->subjectNotified(“org.poco.demo");

notification += Poco::delegate(this, &MyClass::onNotify);

//…

void MyClass::onNotify(const WebNotificationEvent& ev)
{
    std::cout << ev.second << std::flush;
}
```

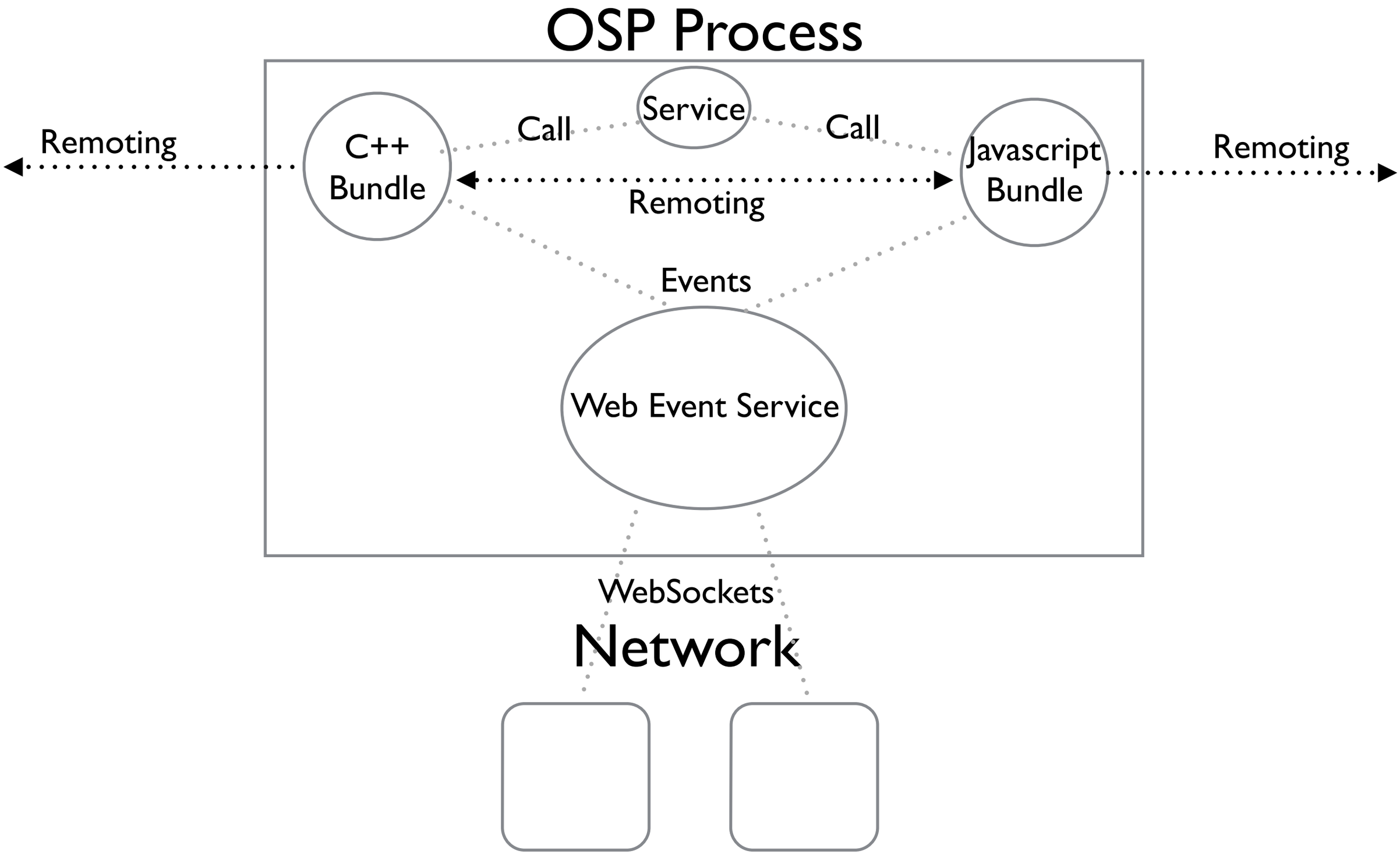
WebEvent Code - Client Side (in process)

```
ServiceRef::Ptr pWebEventSvcRef =
_pContext>registry().findByName(WebEventService::SERVICE_NAME);

WebEventService::Ptr pWebEventService =
pWebEventSvcRef->castedInstance<WebEventService>();

pWebEventService->notify(“org.poco.demo”, “some data”);
```

WebEvent Code - Client Side (over network)

```
ws = new WebEventServer(null, onOpen, null, onMessage, null, true);

function onOpen(response) {
    ws.subscribe("org.poco.demo");
}

function onMessage(response) {
    var subject = response.data.split(" ")[1];
    if (subject.lastIndexOf(uid) != -1) return;// ignore echo
    var data = response.data.split("\r\n")[1];
    // …
}

function send(event, data)
{
    ws.send(event + uid, data);
}
```

Communication Channels



Demo

macchina.io

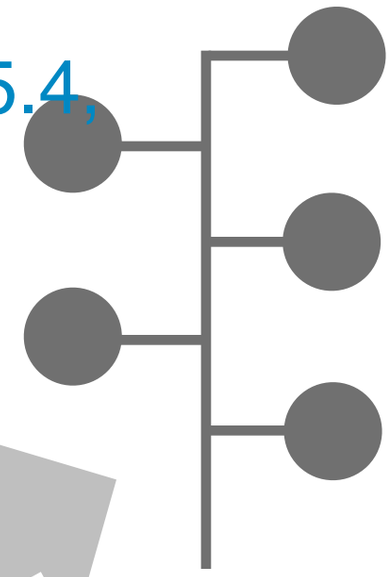
A modular open source toolkit for building embedded IoT applications that connect sensors, devices and cloud services.

Cloud Services

AirVantage, Bluemix, Tinamous, Xively, etc.

Devices/Sensor Networks

CoAP, IEEE 802.15.4, Modbus, USB, Bluetooth, RS-232

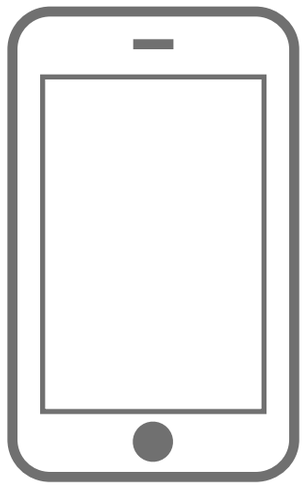


HTTP(S)
MQTT

IoT Gateway

device apps
local “business logic”
web services
web visualization
database
discoverability

Mobile/Web Clients



Remote Access
my-devices.net

IoT Components

Services

WebEvent, SMS, Twitter, …

WebUI

Login, Launcher, Bundles, Playground, etc.

Devices and Sensors

Sensors, Serial Port, GNSS/GPS, Accelerometer, I/O, …

Protocols

MQTT, COAP, Modbus, …

Platform

JavaScript

V8 JavaScript engine and C++ bindings/bridging

Remoting

serialization, remote methods and events, IPC

Open Service Platform

dynamic module system
service registry
web application server
user authentication/authorization

POCO C++ LIBRARIES

platform abstraction, multithreading, XML and JSON processing, filesystem access, stream, datagram and multicast sockets, HTTP server and client, SSL/TLS, etc.

- open source (Apache 2.0 License)
- built in C++ for best performance and efficiency
  (JavaScript for parts of web interface)
- modular and extensible
- mature, proven codebase:
  POCO C++ Libraries, Google V8, Eclipse Paho, SQLite
  AngularJS, jQuery, OpenLayers, Ace (text editor),
  + Applied Informatics OSP and Remoting frameworks
- C++ - to - JavaScript bridge
- Raspberry Pi, Beaglebone, Edison, RED, MangOH, etc.
- prototype on Linux or OS X host, easily deploy to device
- web interface with JavaScript editor

Pro Users and Device Manufacturers

- add device specific APIs
- make devices programmable in JavaScript for partners or end users
- device specific app store (sell additional software features)
- additional frameworks (UPnP, Remoting SOAP and JSON-RPC)
- customizable web user interface
- improved user authentication and authorization
- signed bundles
- pro support, more info http://www.appinf.com

Conclusion

- OSP + Remoting + WebEvent = easy data exchange
- Quickly connect multiple languages and environments
- Internet age ready
- Good performance for most tasks
- Production tested code

alex@pocoproject.org @0x00FA aleks-f

Q&A