Threading Design Decisions in AutoCAD Web & Mobile

using Autodesk::AutoCAD::Mobile::Engineering::MaxRaskin;

Core C++ @ TLV, August 16th 2018

Agenda

Agenda

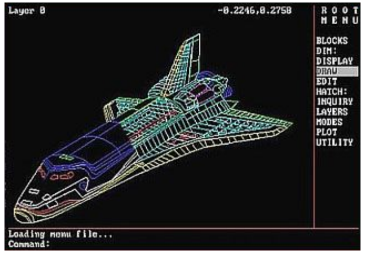
1. Introduction to AutoCAD
2. Cross Thread Communications
3. Generating Cross Platform APIs
4. Dividends!

Introduction to AutoCAD

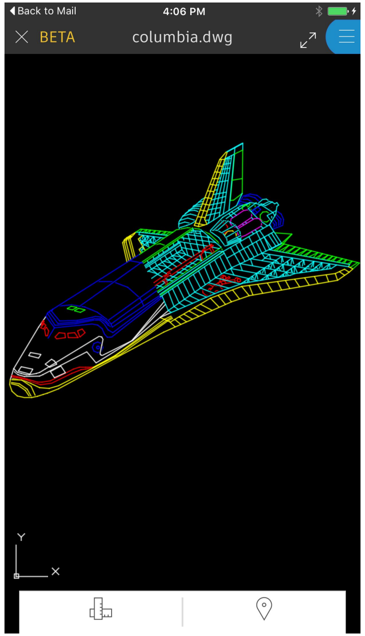
It is...

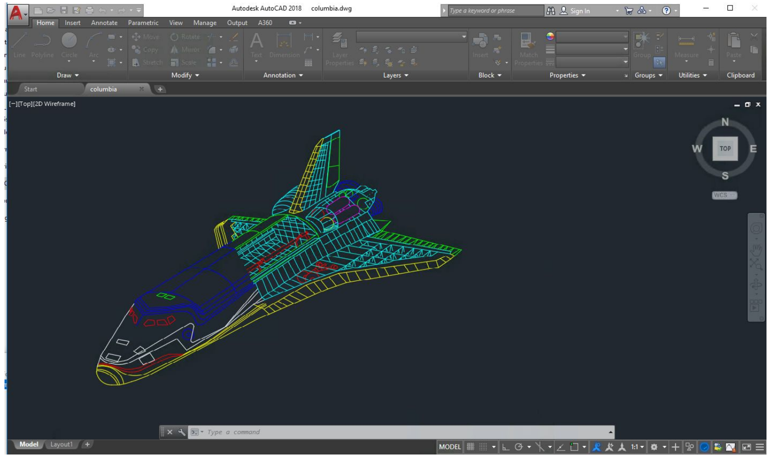
- Autodesk’s flagship product.
- A multipurpose CAD (Computer-Aided-Design) software used for:
  - Architecture
  - Construction
  - Electrical
  - Mechanical
- Comprised of battle proven code continuously worked on since ‘82
- We target *C++ 14* on **all** platforms!

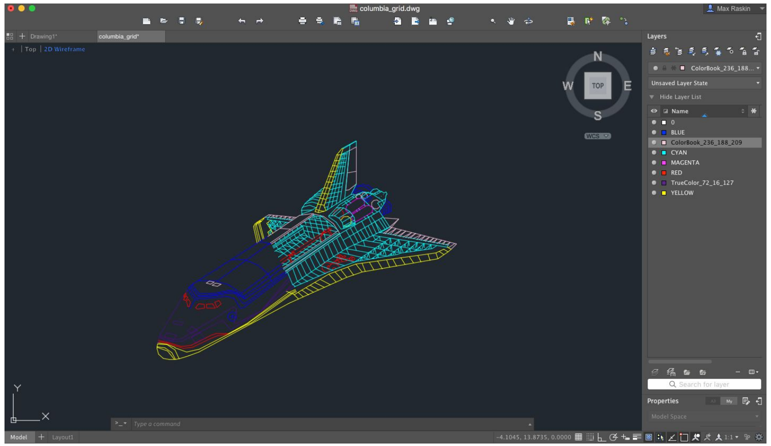
Runs on multiple platforms

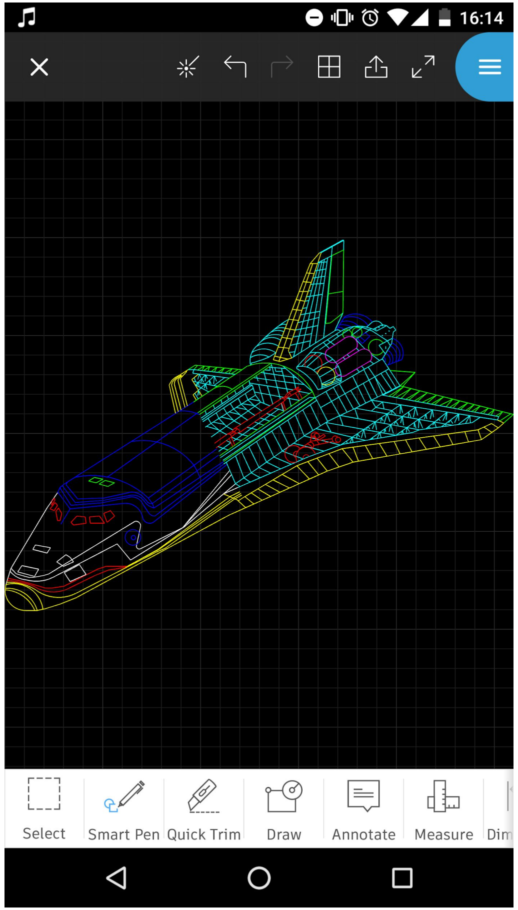


```
Loading menu file...
Command:
```

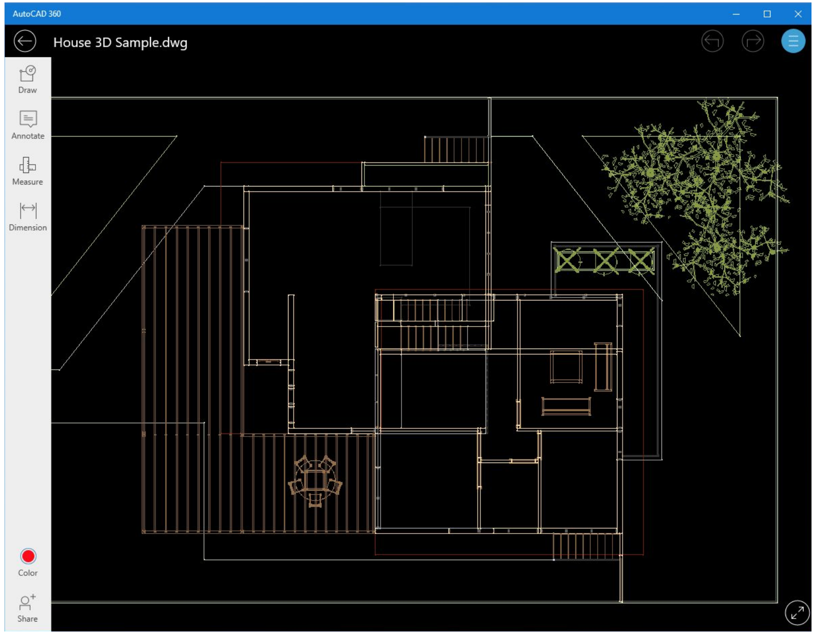


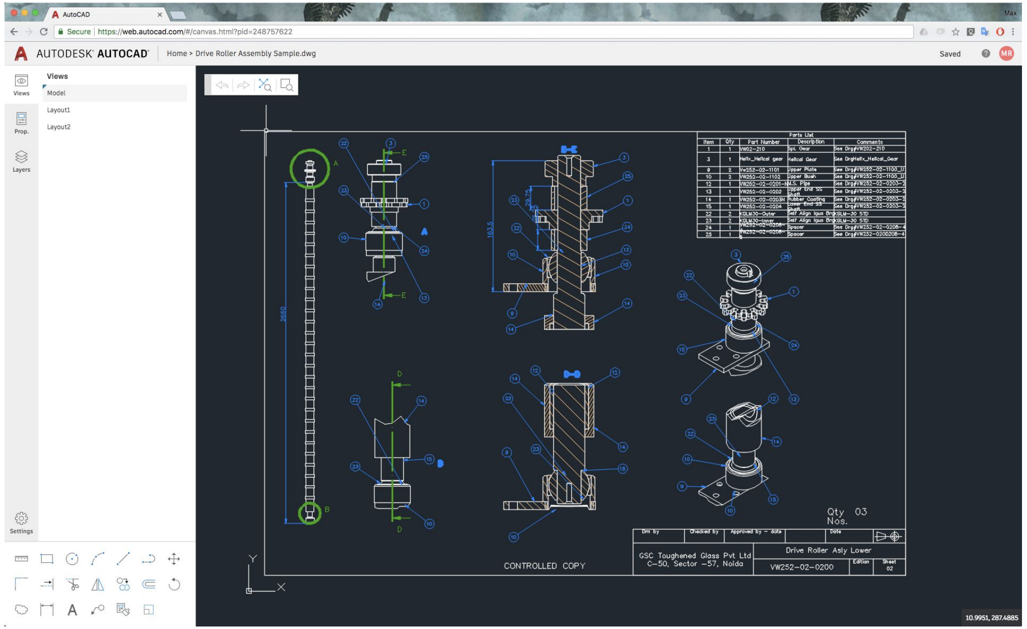






Runs on multiple platforms




AutoCAD
Secure | https://web.autocad.com/#/canvas.html?pid=248757622
AUTODESK AUTOCAD
Home > Drive Roller Assembly Sample.dwg
Saved
Views
Model
Layout1
Layout2
Prop.
Layers
Settings
CONTROLLED COPY
Qty 03 Nos.
GSC Toughened Glass Pvt Ltd C-50, Sector -57, Noida
Drive Roller Asly Lower
VW252-02-0200
10.9951, 287.4885

Cross Thread Communications

∞ AutoCAD’s life span

- The roots of AutoCAD are in MS-DOS (Input Polling).
- Most of it’s life AutoCAD spent on Windows (Msg Loop).
- It is **Single threaded** (!!!)

```
while(true) {
    auto ret = GetMessage(&msg, nullptr, 0, 0);
    if (ret > 0)
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    } //...
}
```

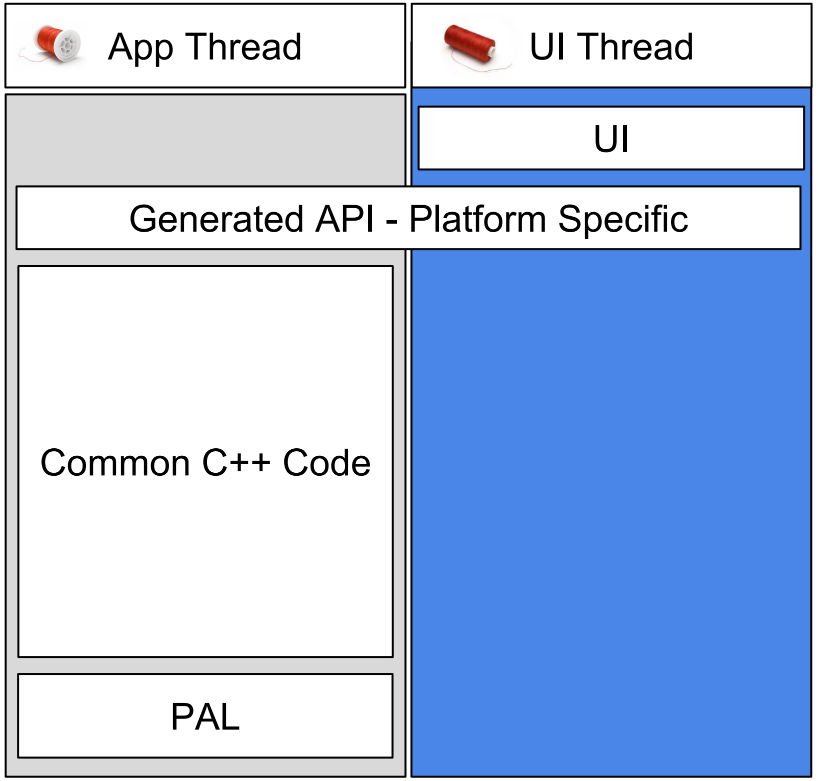
But Alas, On Web & Mobie...

ONE DOES NOT SIMPLY
Slow Process isn't responding.
Do you want to close it?
WAIT
OK
BLOCK THE UI THREAD
imgflip.com

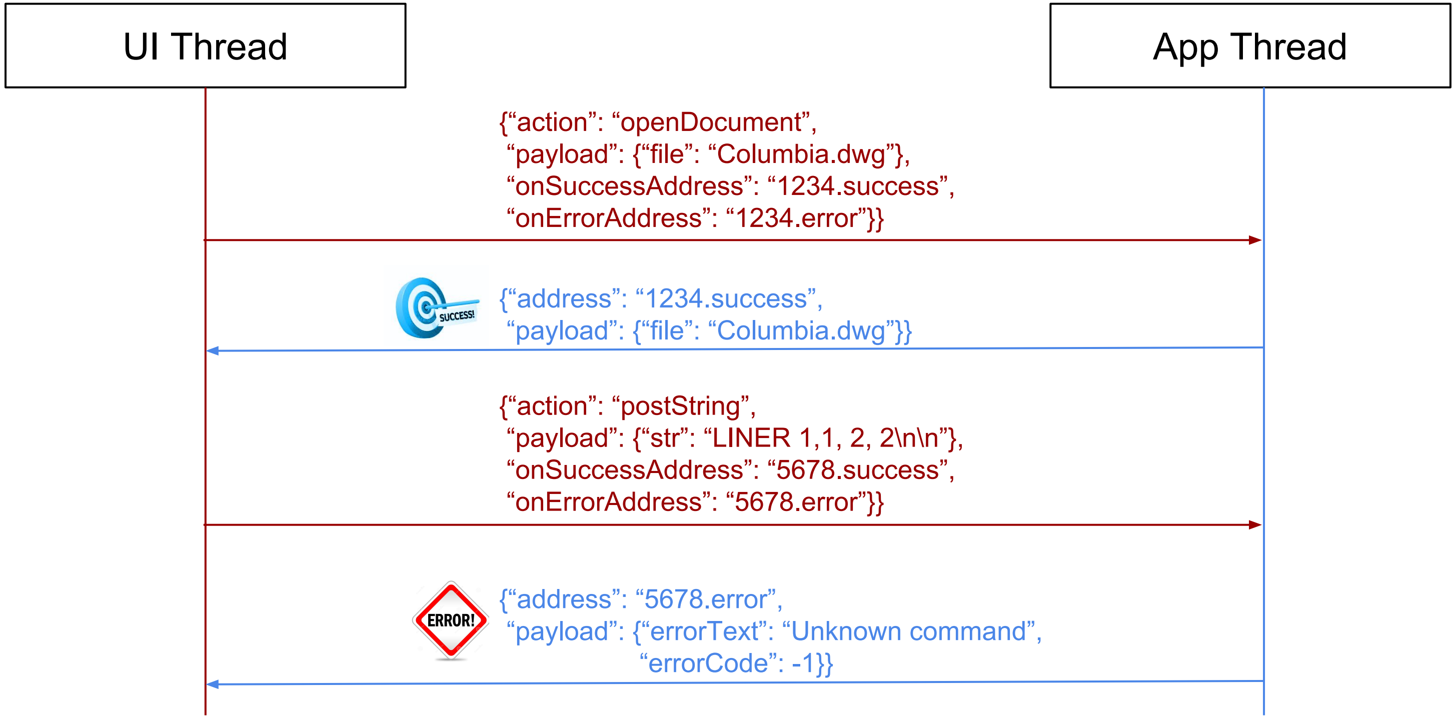
Dual Threading

- *UI Thread* - each platform's OS/browser creates it by default.
- *App Thread* - this is an std::thread/Web Worker we create.
- Communications between threads is achieved via a "*Messaging API*".

High Level Architecture



The Messaging API Protocol



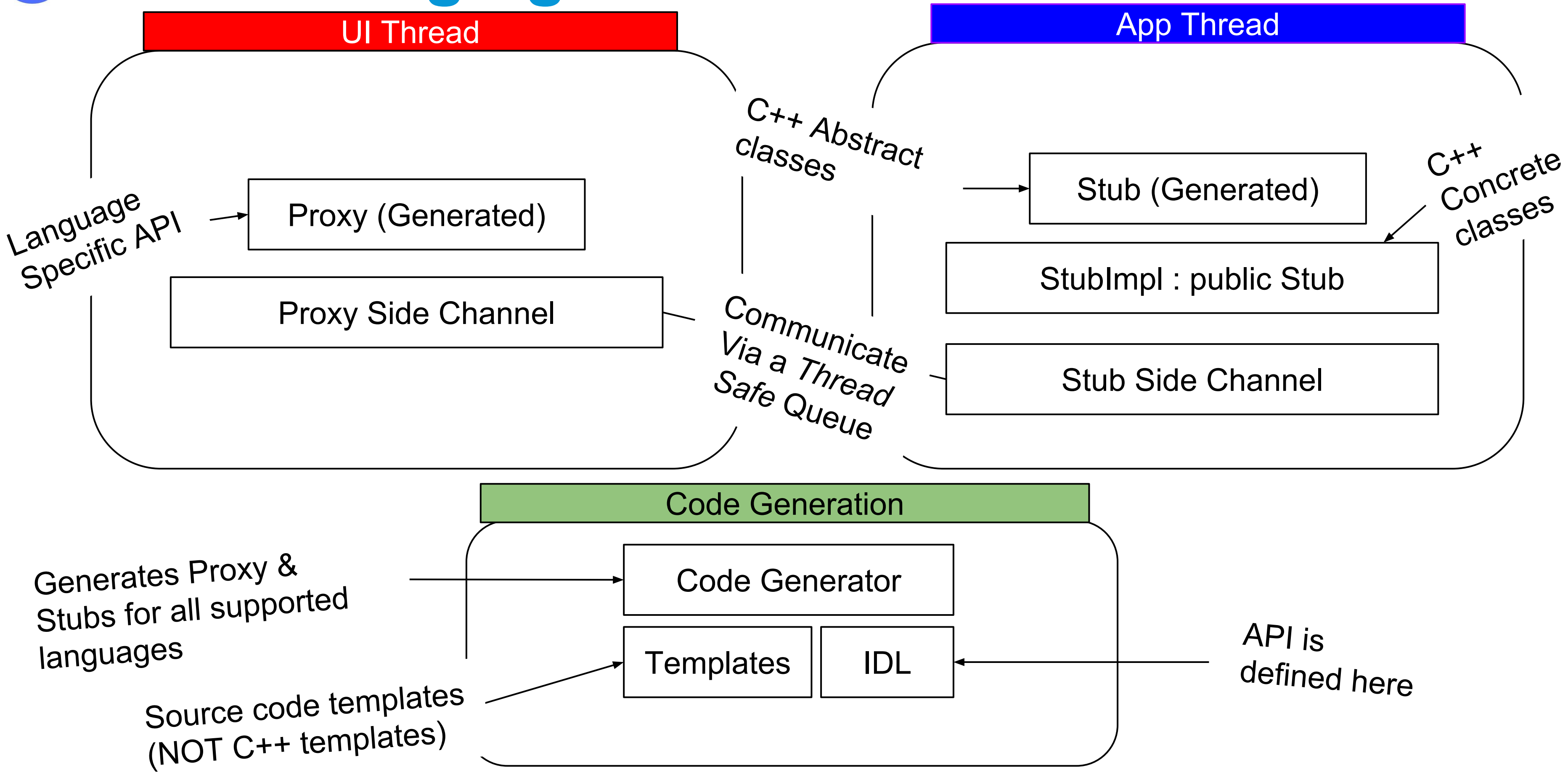
```
{“action”: “openDocument”,
 “payload”: {“file”: “Columbia.dwg”},
 “onSuccessAddress”: “1234.success”,
 “onErrorAddress”: “1234.error”}}
```

```
{“address”: “1234.success”,
 “payload”: {“file”: “Columbia.dwg”}}
```

```
{“action”: “postString”,
 “payload”: {“str”: “LINER 1,1, 2, 2\n\n”},
 “onSuccessAddress”: “5678.success”,
 “onErrorAddress”: “5678.error”}}
```

```
{“address”: “5678.error”,
 “payload”: {“errorText”: “Unknown command”,
                 “errorCode”: -1}}
```

The Messaging API Architecture



Code Deep Dive

Level 0: App (Swift)

UI Thread

```
let app = AcadAppProxy()
app.openDocument(withPath: “Columbia.dwg”,
                    success: { (path) in
                              print(“success!”
                           },
                    fail: { (errorText, errorCode) in
                           print(“failed!”)
                         }
                   )
```

Level -1: Generated Proxy (ObjC)

UI Thread

```
-(void)openDocumentWithPath:(NSString*)path
                        success:(void(^)void)success
                            error:(void(^)(int, NSString*))error {
      NSString* address = nil;
      __block Continuation* continuation = [Continuation new];
      continuation.success = ^(NSString* payload) { [Channel unregisterContinuation:address];
                                                     success(deserializeJson(payload));}
      continuation.error = ^(NSString* payload) { [Channel unregisterContinuation:address];
                                                   success(deserializeJson(payload)); }

      NSMutableDictionary* jsonObject = [NSMutableDictionary new];
      jsonObject[@"path"] = path;
      NSString* jsonStr = serializeToJson(jsonObject);
      address = [Channel registerContinuation: continuation];

      Message* msg = [Message new];
      msg.action = @"openDocument";
      msg.address = address;
      msg.payload = jsonStr;

      [Channel postMessage: msg];
}
```

Level -2: Message Queue (C++)

UI Thread

```
void Stub::Channel::postMessage(const Message& msg) {
    MessageQueue::instance().push(msg);
}

void MessageQueue::push(const Message& msg) {
    std::lock_guard<std::mutex> lock(m_mutex);
    m_postQueue->push_front(msg);
    m_condition.notify_one();
}
```

Level -3: Message Loop C++

App Thread

```
// Meanwhile in the message loop...
while (true) {
   // …
  const auto& msg =  MessageQueue::instance().pop();
  Stub::Channel::instance().dispatchMessage(msg);
   // ...
}
```

Queue Congestion Problem

- Queue congestion due to high frequency message - e.g Pan (hold and drag) gesture.
  - This made canvas navigation to lag behind.

Our Solution - Message Coalescing

```
template <class TFilter>
Message coalesce(std::deque<Message>& queue, TFilter& coalescingFilter)
{
        auto iter = std::remove_if(queue.begin(), queue.end(), std::ref(coalescingFilter));

        if (iter != queue.end()) {
            // Coalesce by discarding all messages which matched the filter
            queue.erase(iter, queue.end());
        }

        auto messageToHandle = queue.back();
        queue.pop_back();

        return messageToHandle;
}
```

Level -3: Pop Message (C++)

App Thread

```
Message MessageQueue::pop(const Message& msg) {
    std::unique_lock<std::mutex> lock(m_mutex);
    m_condition.wait(lock, [this] { return !m_postQueue->empty(); });
    return coalesce(*m_postQueue);
}
```

Level -3: Dispatch Message (C++)

App Thread

```
void Stub::Channel::dispatchMessage(const Message& msg) {
        auto stubImpl = stubImplsMap.find(msg.action);
        auto result = stubImpl->invoke(msg.action, msg.payload);
        return Proxy::Channel::instance().postMessage(msg.address,
                                    result.ok()/*success?*/, result.serialize());
}
```

Level -3: Generated Stub (C++)

App Thread

```
//  Generated code:
class AcadAppApiStub : public ApiObject {
public:
    virtual Result openDocument(const string& path) = 0;
    Result invoke(const string& action, const string& payload)    {
            if (funcName == “openDocument”s) {
                auto jsonObject = deserialize(payload);
                return openDocument(jsonObject[“path”]);
            }
            return Result::failed(“Bad api”);
        }
}
```

Level -3: Stub Impl (C++)

App Thread

```
// Hand written code:
class AcadAppApi : public AcadAppApiStub {
public:
        void openDocument(const std::string& path) {
                auto result = openDoc(str);
                JsonObject payload;
                payload["path"] = path;
                return Result{result.code, result.msg, payload};
        }
}
```

Level -3: Proxy Post Message (C++)

App Thread

```
Result Proxy::Channel::postMessage
(const string& address, bool success, const string& payload)
 {
    Message msg;
    msg.address = address + (success) ? “.success” : “.fail”;
    msg.payload = payload;
    PAL::runOnUiThread( [msg]
                               {Proxy::Channel::dispatchMessage(msg); });
}
```

Level -2: Proxy Dispatch Msg (C++)

UI Thread

```
void Proxy::Channel::dispatchMessage(const string& msg) {
    const auto& addressParts = split(“.”, msg.address);
    if (addressParts[1] == “fail”s)
        continuations[addressParts[0]].success(msg.payload);
    else
        continuations[addressParts[0]].fail(msg.payload);
}
```

Generating
Cross Platform APIs

IDL and Templates

- Common to all target languages:
  - We define our API using an IDL - Interface Definition Language.
  - Our IDL is C#
- Per target language:
  - Templates for boilerplate code.
  - Templates are Microsoft's T4 template engine.

IDL Example

```
namespace AsyncApi
{
    interface App
    {
        void openDocument(string path);
    }
}
```

T4 Templates Snippet for ObjC

Platform Specific
Code Fragments

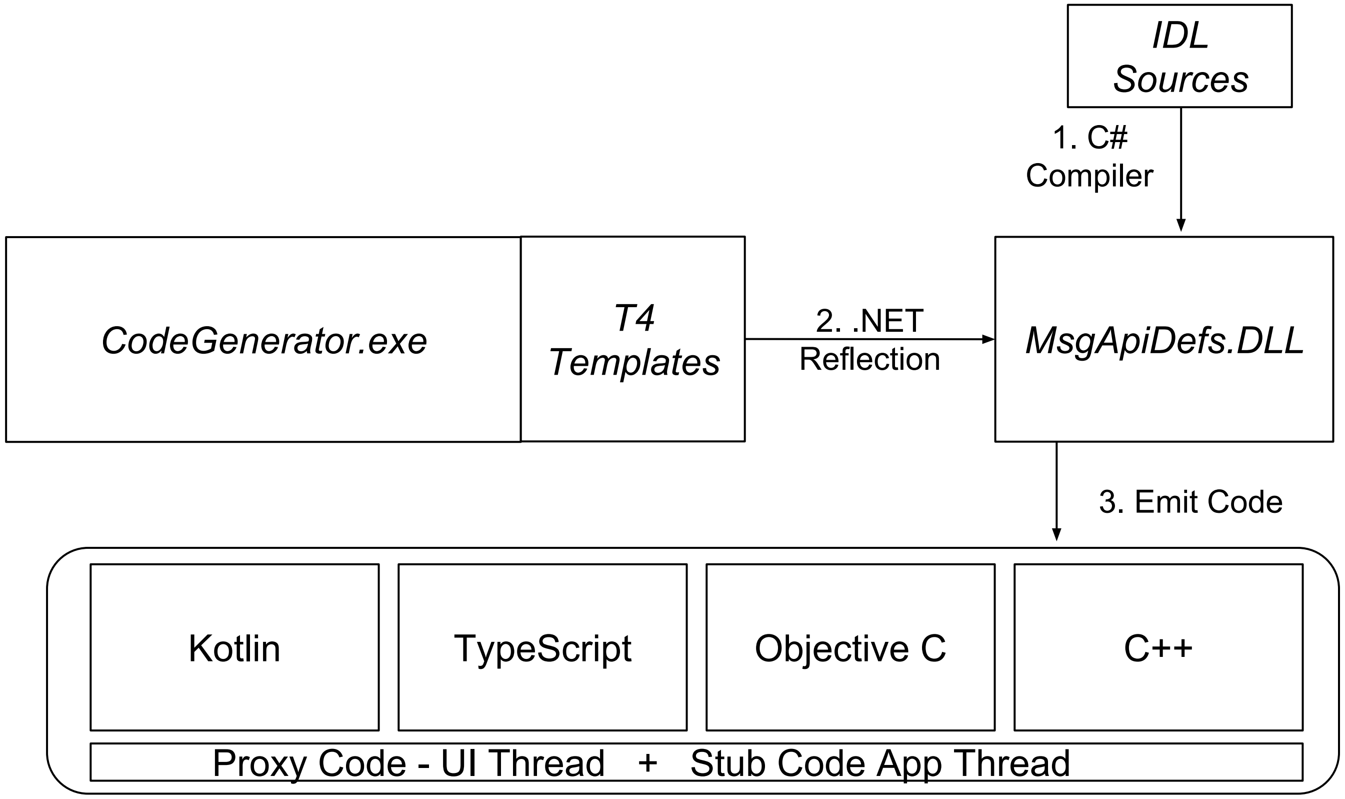
C# Code That
Uses Reflection

```
// ...
@protocol <#=StringHelpers.PROTOCOL_NAMESPACE_PREFIX + m_type.Name#>Protocol

foreach (var method in this.m_type.GetMethods())
{
// …
- (void) <#=methodName#>With<#=parameters#> <#=successCallback#> <#=errorCallback#>;

// …
}
```

Generation Pipeline



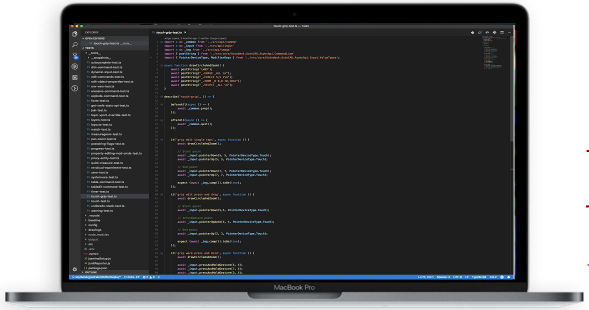
Dividends

E2E Cross Platform

- The architecture, being client-server with a Messaging API, allows for switching threads to processes.
- Consider:

Proxy - E2E Tests in Typescript in Jest

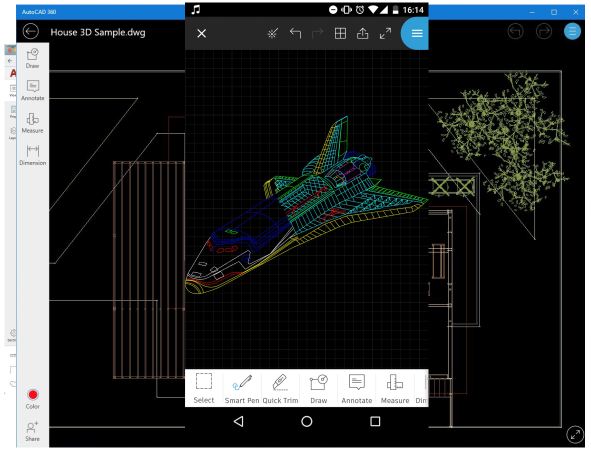
Stub - ANY PLATFORM!



Deploy app

Run tests

Get results



Questions?

AUTODESK

We’re Hiring!

https://my.tomigo.com/p/KAip3

Thanks for bearing with me!

AUTODESK®