

Robin Osborne



Debugging BotFramework locally using ngrok

No doubt you're already having lovely long conversations with your bot via Skype (or Facebook Messenger, or even SMS!) built using the botframework, and by using the Bot Emulator you can run your bot locally and debug it.

However, once it's deployed and is being called via the Bot Connector framework, instead of directly, things get a bit tougher.

If you haven't managed to override the – rather nasty – default exception handling that swallows exceptions and spews out reams of useless stack trace, then you may not have much idea what's going on with your deployed bot, since all you get back is “Sorry, my bot code is having a problem.”

When you encounter a strange problem whilst conversing with your bot in Skype, going through the process of adding loads of logging and redeploying, trying again, checking logs – just to see the journey your bot code is going through – isn't the most efficient.

If only you could debug the code on your development PC just as easily as you could before the bot was deployed, locally in Visual Studio...

In this article I'm going to show you how to debug your bot code from Skype though to your local PC's Visual Studio instance, thanks to the amazing ngrok!

Ngrok

Ngrok is a small executable that creates secure tunnels to localhost. What does that actually mean? It means that ngrok can create a *public* HTTPS (And HTTP. And SSH) endpoint for any port on your local PC.

WOAH.

SO. CLEVER.

Go and download ngrok, then come back here. I'll wait. It's only a small zip.

...

Got it? Awesome, now extract the executable somewhere handy, perhaps a directory that's already in your PATH.

Done that? Ok, let's get started!

Using ngrok

There are a plethora of configuration options for ngrok, but we're going to concentrate on the ones that allow localhost to be exposed externally and map the incoming request back to Visual Studio.

First up – Visual Studio

Open Visual Studio, get your bot code loaded and hit F5. You'll be presented with a screen like this:



rposbo.botframework.demobot

Describe your bot here and your terms of use etc.

Visit [Bot Framework](#) to register your bot. When you register it, remember to set your bot's endpoint to

`https://your_bots_hostname/api/messages`

That's the port we're interested in. Write it down. Or just remember it. Whatever works for you

Secondly – ngrok

Now open up a command prompt of your choice and (unless you extracted ngrok somewhere that's already in your PATH or you added it to your PATH) navigate to the location of your ngrok executable.

All you need to do now is enter this:

```
ngrok.exe http 3979 -host-header="localhost:3979"
```

This is exposing the port 3979 over HTTP and mapping incoming connections with the host header "localhost:3979" so that IIS Express picks them up; without this `-host-header` param the incoming connections will just fail.

Hit [Enter], and be patient. It's doing EXTREME MAGIC in the background, which naturally takes a few seconds. After a short break you should end up with something like this:

```
ngrok by @inconshreveable

Session Status      online
Version             2.1.14
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding           http://9544146f.ngrok.io -> localhost:3979
Forwarding           https://9544146f.ngrok.io -> localhost:3979

Connections         ttl      opn      rt1      rt5      p50      p90
                   0        0        0.00     0.00     0.00     0.00
```

The foundations have been laid! Pay attention to that “Web Interface” URL, since we’ll use that later on!

Third – botframework dev portal

Now we need to register our bot in the portal so we can get the App ID and App Key/Secret/Password. Head on over to <https://dev.botframework.com/bots/new>, log in with your Microsoft Live (or whatever they’re calling it this month) account, and you’ll see something like this:

Bot profile



Icon

[Upload custom icon](#)

30K max, png only

* Name ?

ngrokdemobot

* Required field

* Bot handle ?

ngrokdemobot

* Description ?

ngrokdemobot

Configuration

Messaging endpoint ?

https://9544146f.ngrok.io/api/messages

* Microsoft App ID ?

Microsoft App ID from the Microsoft App registration portal

Create Microsoft App ID and password

Under “Configuration” you’ll see “Messaging Endpoint”: paste your HTTPS ngrok URL in here, adding “/api/messages” at the end.

Then click the “Create Microsoft AppID and password” to open a new page where you specify the app name and are given an **App Id** . Now

click “Generate a password to continue” – take these two values and paste them in your bot’s `web.config`:

```
<appSettings>
  <!-- update these with your BotId, Microsoft App Id and your M
  <add key="BotId" value="PUT YOUR BOT HANDLE IN HERE" />
  <add key="MicrosoftAppId" value="PUT YOUR APP ID IN HERE" />
  <add key="MicrosoftAppPassword" value="PUT YOUR PASSWORD IN HERI
</appSettings>
```



Save this file – which will cuase your debug session to stop – and rerun your bot; don’t worry about ngrok as the URL will stay the same unless you restart ngrok itself. Which you should avoid doing, since this is the URL you’re registering with the portal.

Complete the rest of the form in the portal and hit “Register”.

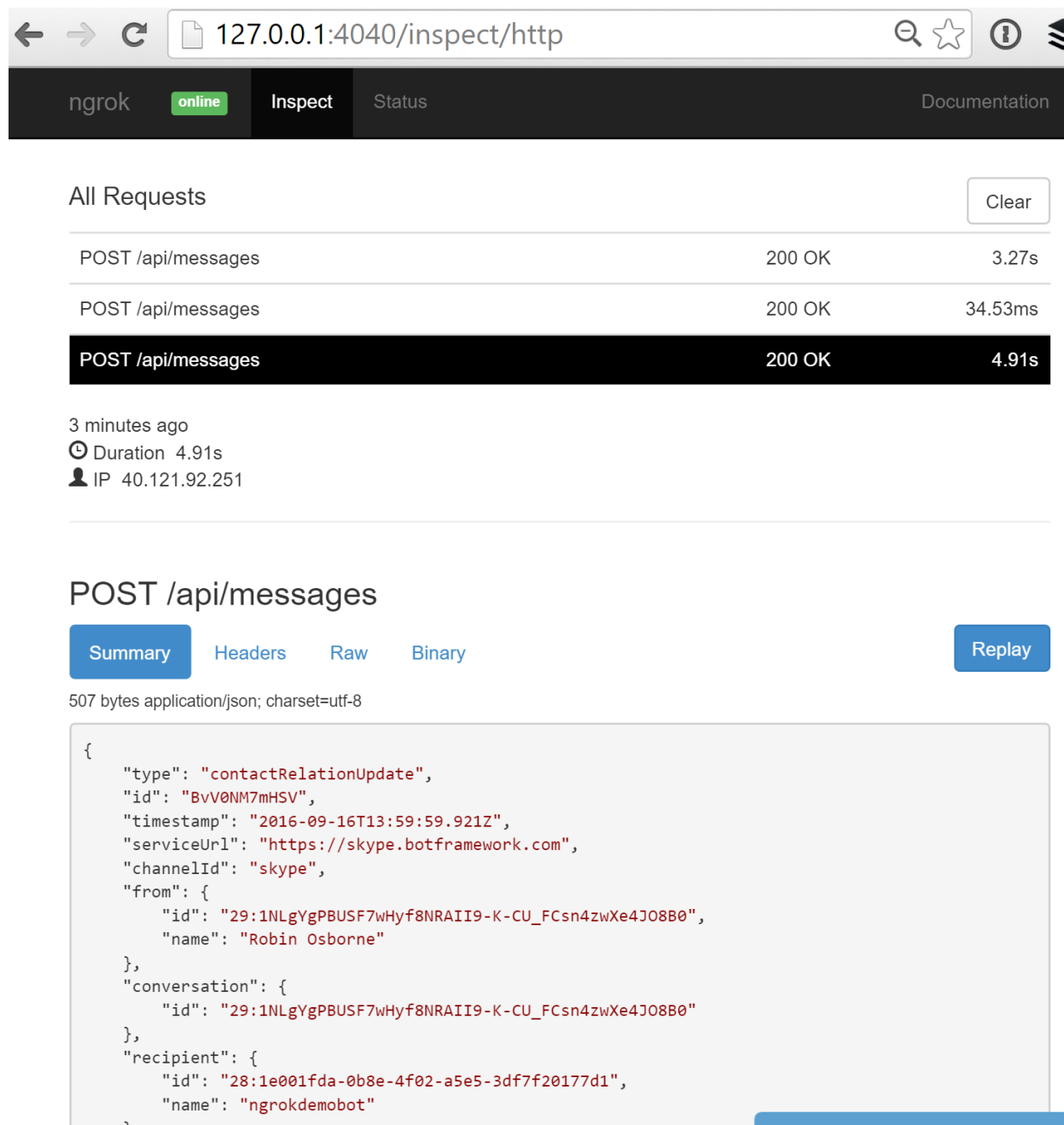
You should now see the bot dashboard where you can enable or disable the various integrations; Skype should already be enabled by default.

Finally – debug!

Let’s set a breakpoint in your `MessagesController` in the `Post` method. Now go back to your bot portal and click the “Add to Skype” button in the Skype channel; this should take you to a page with an “Add to Contacts” button – hit that, log in with your Skype details, and eventually you’ll be able to open up Skype and start a conversation with your locally hosted bot.

In Skype say “hi” to your bot; in the ngrok window you should see some traffic being logged. If you look at the previous ngrok screenshot you’ll see a URL for a “web interface” – I said to remember it! – so head over

to that URL and you'll even be able to inspect the structure of the requests coming in!



The screenshot shows the ngrok web interface. At the top, the address bar displays `127.0.0.1:4040/inspect/http`. Below the address bar, there's a navigation bar with `ngrok`, `online`, `Inspect`, `Status`, and `Documentation`. The main content area is titled `All Requests` and includes a `Clear` button. A table lists three requests:

Method	URL	Status	Duration
POST	/api/messages	200 OK	3.27s
POST	/api/messages	200 OK	34.53ms
POST	/api/messages	200 OK	4.91s

Below the table, it shows `3 minutes ago`, `Duration 4.91s`, and `IP 40.121.92.251`. The selected request details are shown below, with tabs for `Summary`, `Headers`, `Raw`, and `Binary`. The `Summary` tab is active, showing the request body as a JSON object:

```
{
  "type": "contactRelationUpdate",
  "id": "BvV0NM7mHSV",
  "timestamp": "2016-09-16T13:59:59.921Z",
  "serviceUrl": "https://skype.botframework.com",
  "channelId": "skype",
  "from": {
    "id": "29:1NLgYgPBUSF7wHyf8NRAII9-K-CU_FCsn4zwXe4JO8B0",
    "name": "Robin Osborne"
  },
  "conversation": {
    "id": "29:1NLgYgPBUSF7wHyf8NRAII9-K-CU_FCsn4zwXe4JO8B0"
  },
  "recipient": {
    "id": "28:1e001fda-0b8e-4f02-a5e5-3df7f20177d1",
    "name": "ngrokdemobot"
  }
}
```

Absolutely magical. This is a fantastically powerful and simple tool for debugging and capturing traffic coming in to whatever process is being exposed on your local PC.

Eventually your breakpoint should be hit:

So long as you keep your ngrok process running, you should keep the same url; you can keep stopping and starting Visual Studio, however, as you make you changes and try debugging again.

Summary

In this article we've learned about using ngrok to expose your local PC over HTTPS with a public URL, register this with the botframework portal, and debug a Skype chat running your bot code locally.

Hope you found this useful!

Until next time...



September 19, 2016



Robin Osborne



bot, botframework, debug, ngrok

← *Summing CSV data with Powershell*

Create your first QnA bot using botframework's QnA Maker →

3 thoughts on “Debugging BotFramework locally using ngrok”