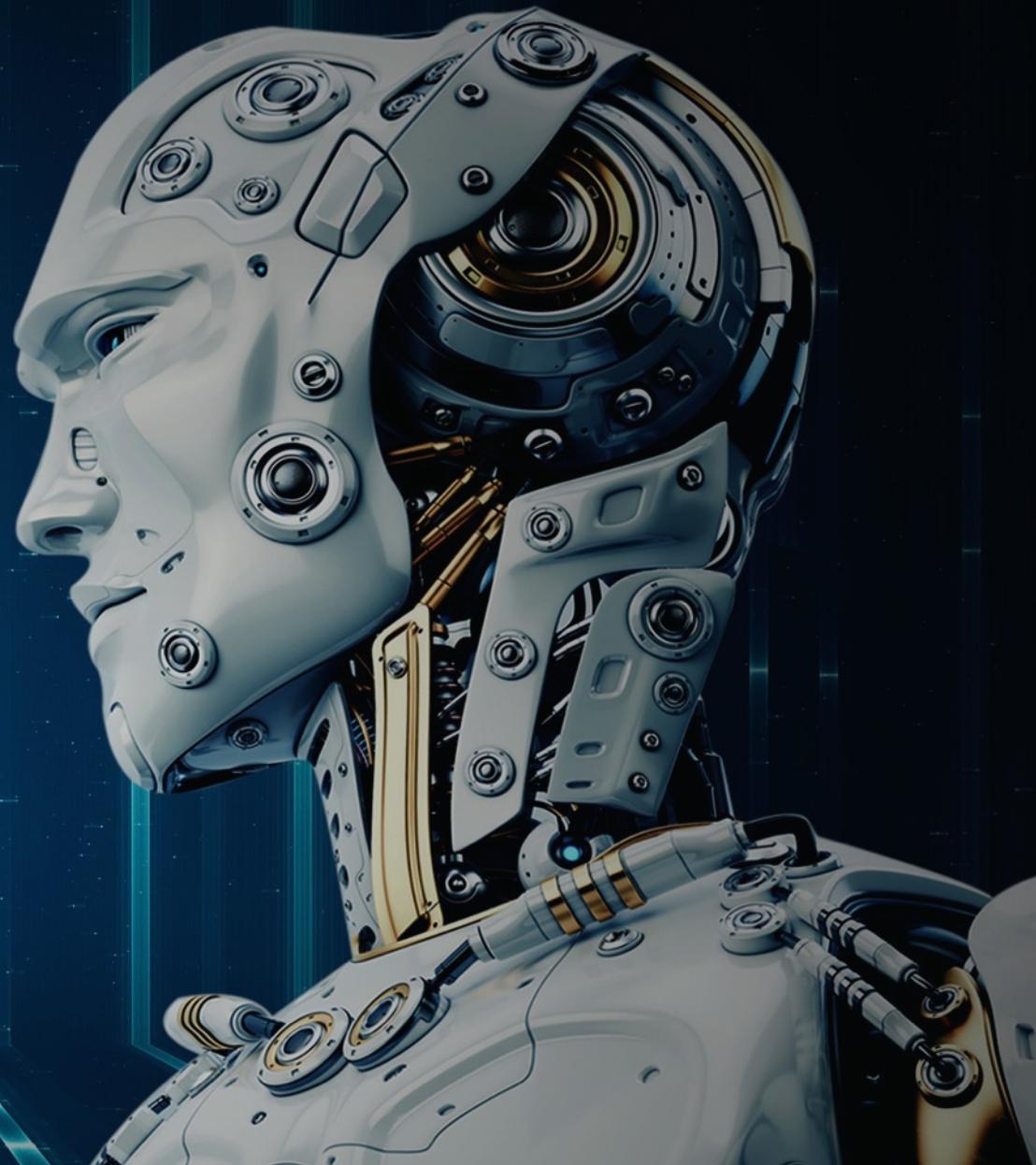


ORACLE®

Oracle Intelligent Bots Advanced Training

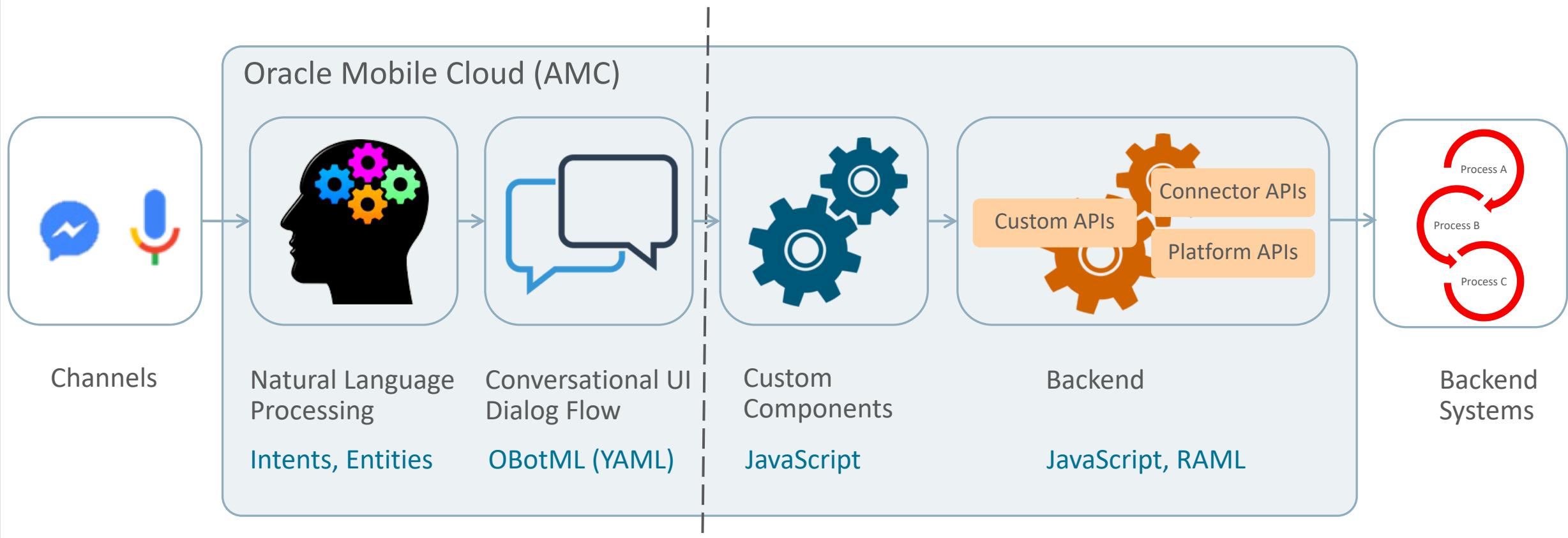
Custom Component Local Development &
Debugging



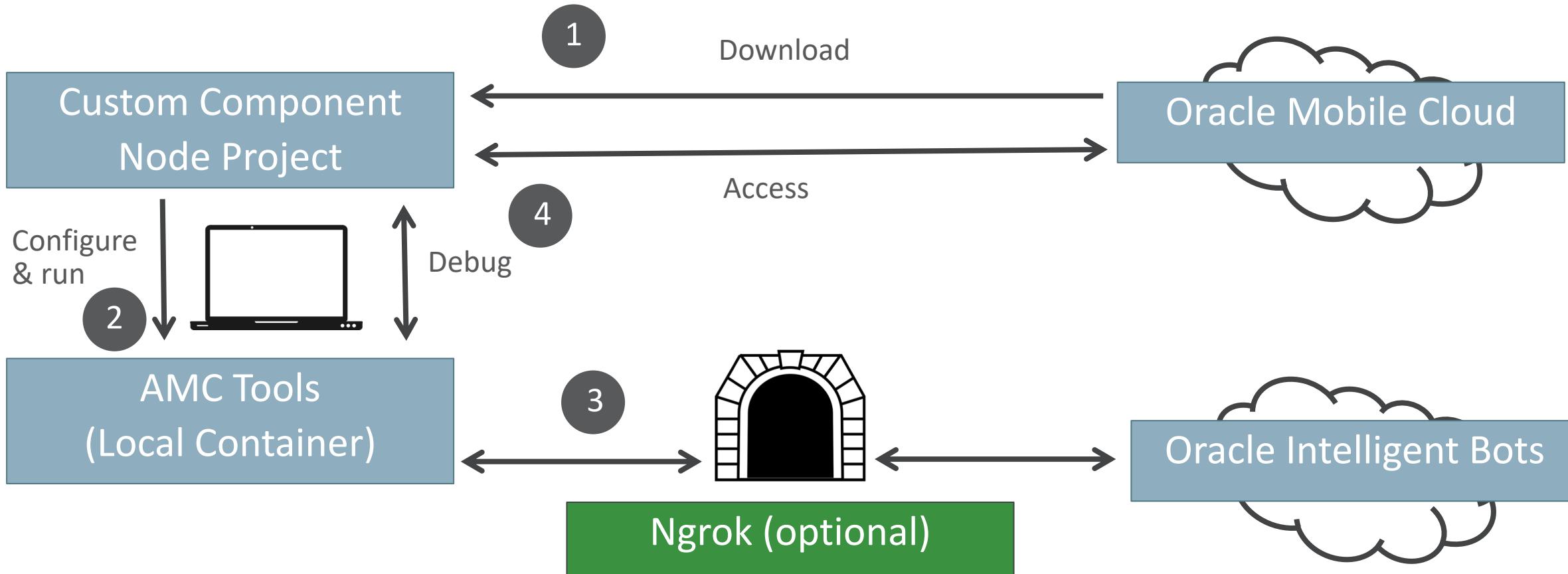
Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Custom Components In Oracle AMC



AMC / Bot Local Development & Debugging Architecture



Topic Agenda

- 1 ➤ Install Custom Code Test Tools
- 2 ➤ Setup for Custom Component Project
- 3 ➤ Run Custom Component in Local Container from Oracle Intelligent Bots
- 4 ➤ Debug Custom Component in Local Container

Local Development & Debugging

Install Custom Code Test Tools

Oracle Custom Code Test Tools

- Set of tools to locally test Custom API
 - Local node container provides AMC integration
- Tools
 - npm module that includes command-line tools
 - Starting a local custom code container
 - Run tests
 - Deploying an API to AMC
 - OracleMobileAPI to proxy AMC API calls from local node container to AMC
 - Needs to be uploaded as custom API to AMC

Custom Code Test Tools

- Download from OTN
 - Custom Code Test Tools
- Need to accept license agreement
- Choose most recent version
- Unzip download on local machine

The screenshot shows the 'Downloads' section of the Oracle Mobile Cloud Enterprise website. At the top, there are navigation links: What's New, Downloads, Articles & Whitepapers, Community, and Learn More. The 'Downloads' link is highlighted.

Oracle Mobile Cloud Enterprise Downloads

This page contains all the downloads for Oracle Mobile Cloud, Enterprise (OMCe).

OTN License Agreement

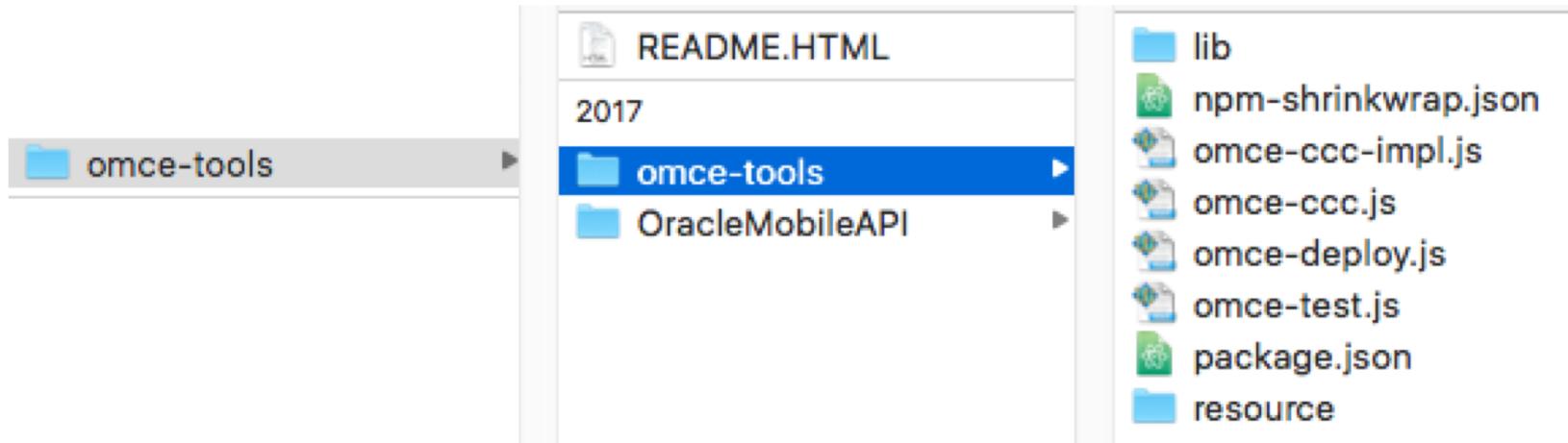
You must accept the [OTN License Agreement](#) to download this software.

Accept License Agreement | Decline License Agreement

TOOLS

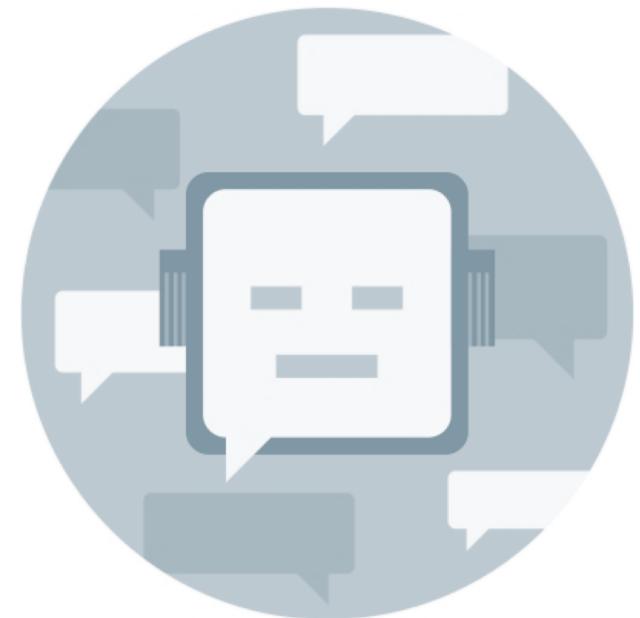
Tool	Download	Description
Custom Code Test Tools v18.1.1.0 01/10/2018	 omce-tools-v18.1.1.0	Custom code offline deployment and debugging.
Admin tools v18.2.5.0 05/08/2018	 omce-admin-tools-v18.2.5.0.zip	Administrator tools, including diagnostics and patching.
Admin tools v18.2.1.0 04/06/2018	 omce-admin-tools-v18.2.1.0.zip	Administrator tools, including diagnostics and patching.

Unzipped Custom Code Test Tools



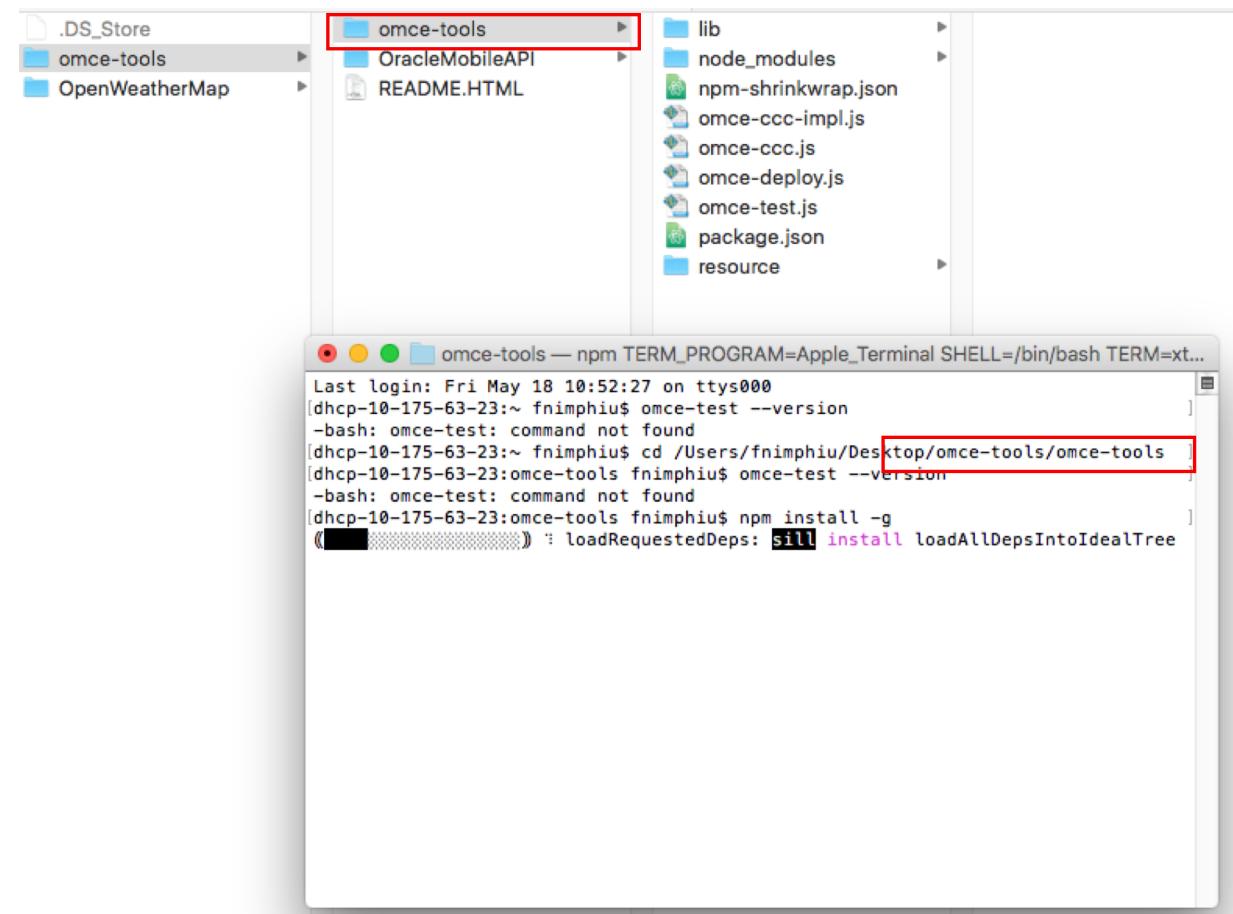
- **omce-tools**
 - Contains the actual command line
 - Starts the local node container
- **OracleMobileAPI**
 - AMC custom API acting as a proxy

You install the OracleMobileAPI only once per AMC instance. The custom code test tool also only needs to be installed only once for a device.



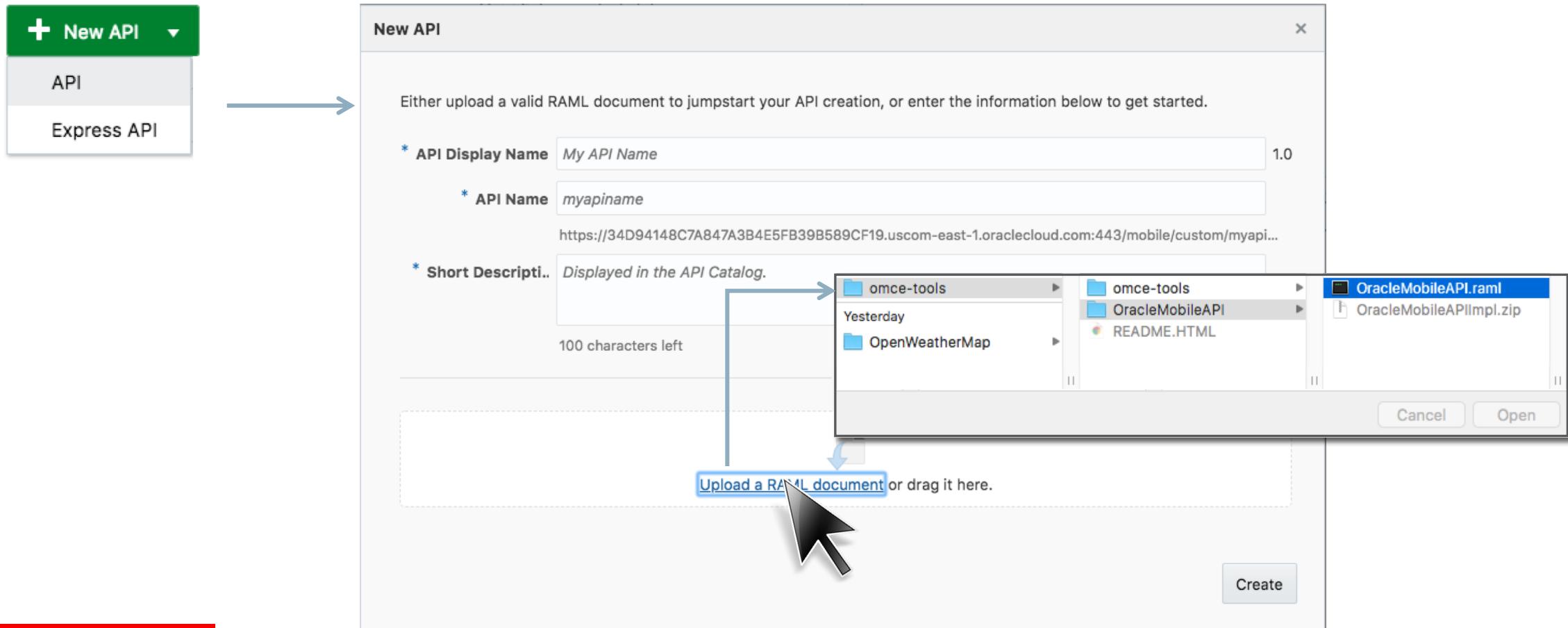
AMC Tools Installation

- Ensure you have Node package manager (npm) installed
- In terminal window, switch to omce-tools directory
- Issue *npm install -g*
 - Installation takes a minute or two
 - If installation fails try with "sudo" command
- Run *omce-test -- version* to verify installation success



Uploading the OracleMobileAPI Proxy to AMC

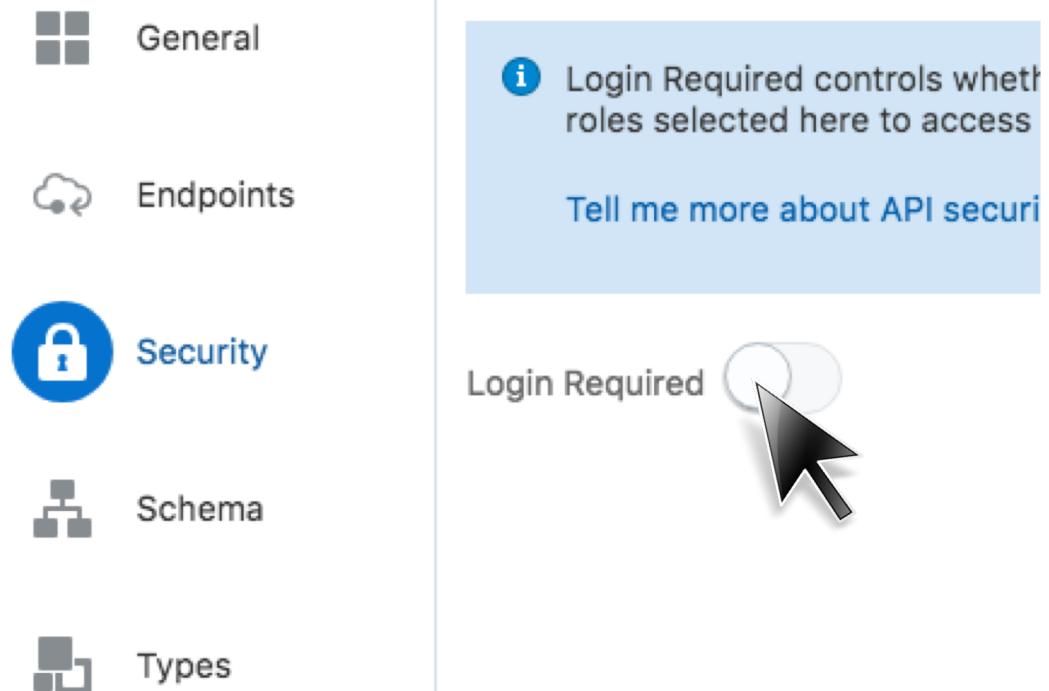
Create Custom API from RAML File



Uploading the OracleMobileAPI Proxy to AMC

Disabling Named User Authentication

- Disable named user account authentication by unsetting the **Login Required** toggle
- Save your change
- Still, anonymous authentication will be required



Uploading the OracleMobileAPI Proxy to AMC

Upload Proxy Implementation Code



Types



Traits



Documentation



Implementation



API Catalog

Tell me what's expected in my implementation archive

JavaScript Scaffold



Upload an implementation archive or drag it here.



omce-tools	▶
OracleMobileAPI	▶
README.HTML	

OracleMobileAPI.raml

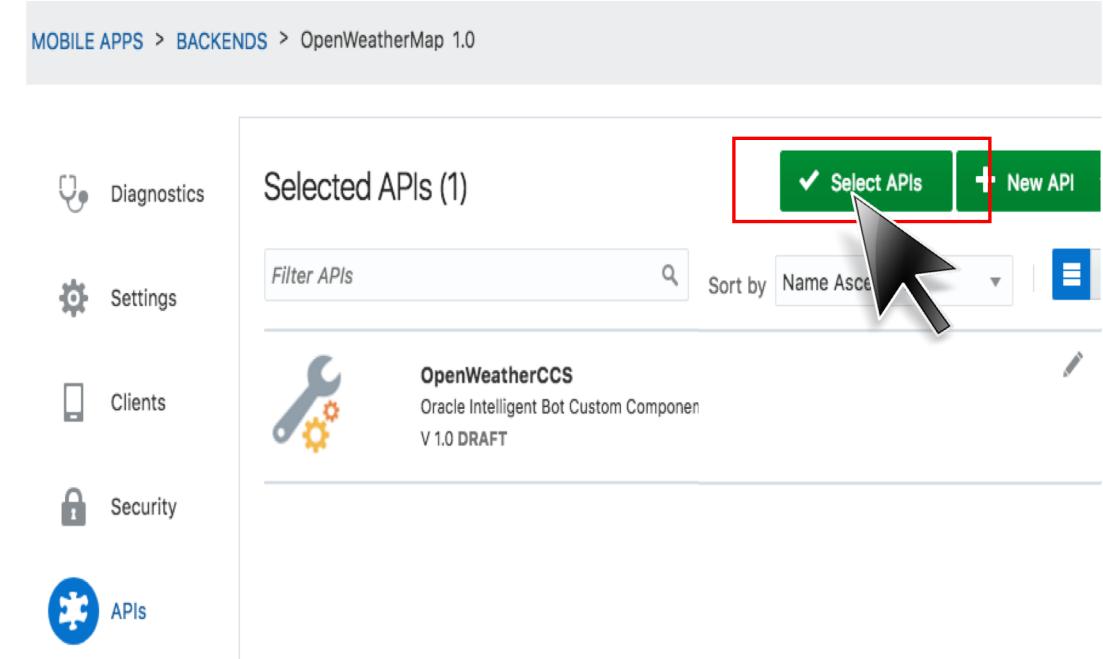
OracleMobileAPIImpl.zip

Local Development & Debugging

Setup for Custom Component Project

Adding the Oracle Mobile API to Your Backend

- Go to bot the mobile backend that exposes the bot custom component service
 - Open the mobile backend in the AMC dashboard
- Select the "APIs" menu item
- Select "Select APIs" to add the Oracle Mobile API to the mobile backend



[**< OpenWeatherMap 1.0**](#) [**> API Catalog**](#)

Oracle



Sort by

Name Ascending



OpenWeatherCCS
Oracle Intelligent Bo...

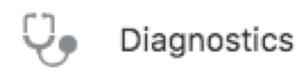
V 1.0 DRAFT



OracleMobileAPI
Custom API Tooling

V 1.0 DRAFT





Diagnostics



Settings



Clients



Security



APIs

Selected APIs (2)

Select APIs

▾

Filter APIs



Sort by

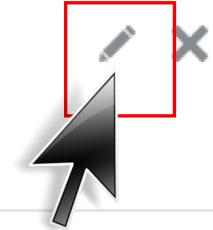
Name Ascending



OpenWeatherCCS

Oracle Intelligent Bot Custom Component

V 1.0 DRAFT



OracleMobileAPI

Custom API Tooling

V 1.0 DRAFT

Download Custom Component Service Implementation Scaffold

Traits

Documentation

Implementation

Download a new JavaScript scaffold at any time to include changes.

JavaScript Scaffold

Set as Default Publish Move to Trash Download

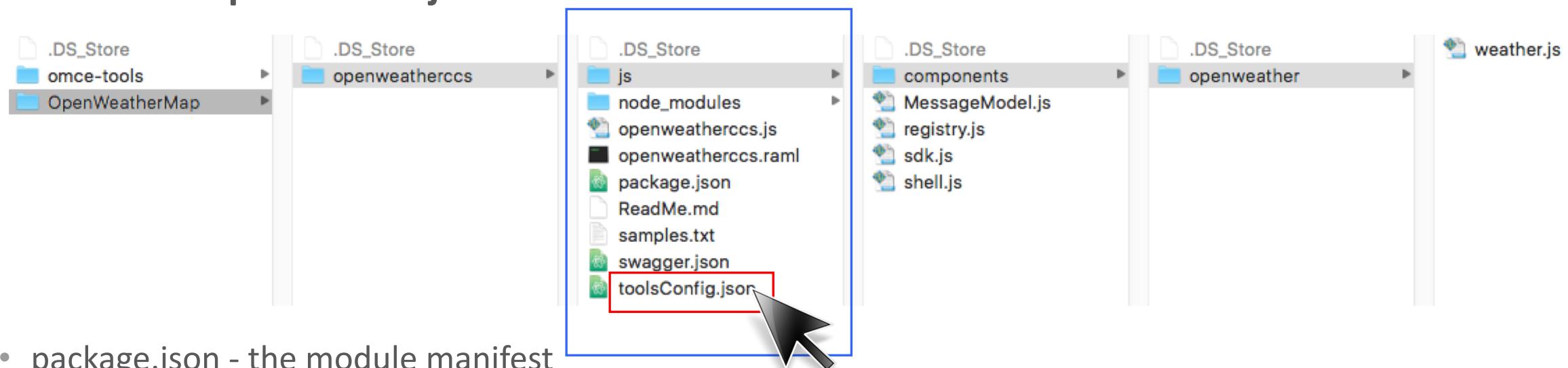
Status	Default	Name
		openweathermapccs

Mock

► Dependencies for openweathermapccs 1.0.0

OpenWeatherMap Custom Component Project

Custom Component Project



- package.json - the module manifest
- <api name>.js - your starter implementation
- <api name>.raml - the API definition in RAML format
- swagger.json - the API definition in Swagger format
- toolsConfig.json - metadata needed by the tools

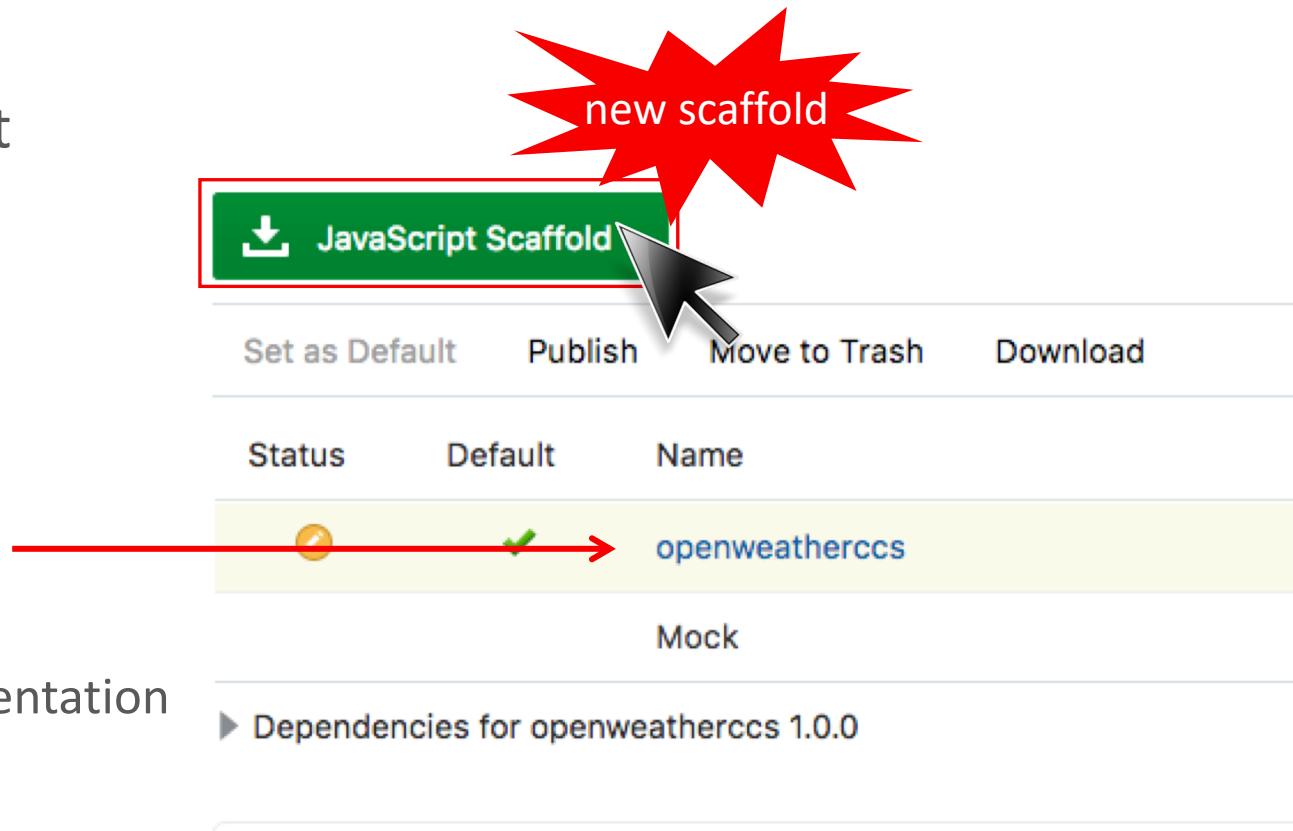


**toolsConfig.json format
is different for migrated
Custom API projects**



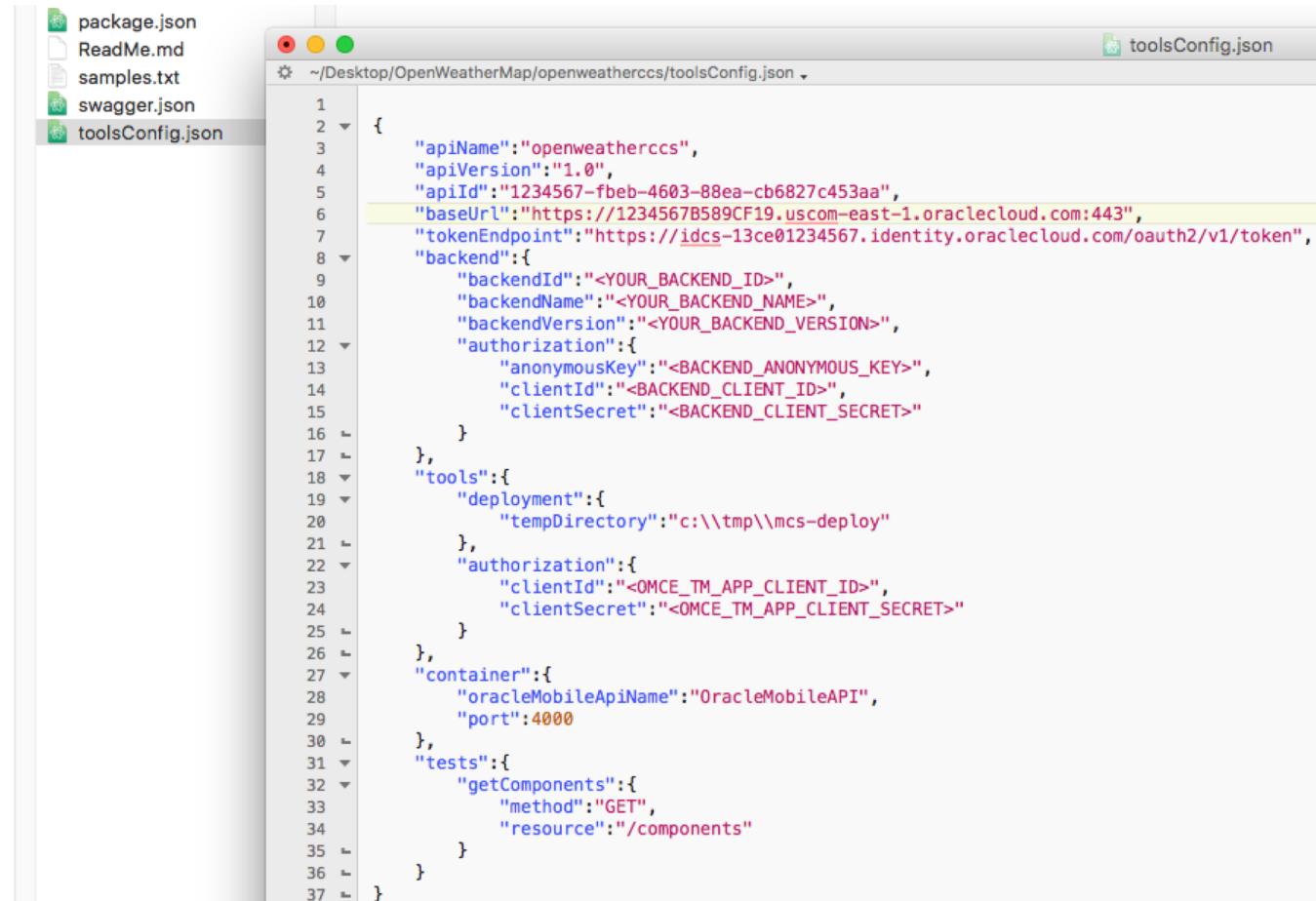
Handling Migrated Custom API Projects

- toolsConfig.json has a different format in Oracle MCS and Oracle AMC
 - Format may change in the future too
- To update the toolsConfig.json file
 - Go to custom API in AMC
 - Download and unzip API implementation
 - Download new scaffold
 - Copy toolsConfig.json to unzipped implementation
 - Re-upload implementation
 - But not before setting it up for debugging



Configuring the toolsConfig.json File

- Configure toolsConfig.json with settings from Mobile Backend
 - baseURL
 - backendId
 - backendName
 - backendVersion
 - anonymousKey
 - clientId
 - clientSecret



```
1  {
2     "apiName": "openweathermapccs",
3     "apiVersion": "1.0",
4     "apiId": "1234567-fbeb-4603-88ea-cb6827c453aa",
5     "baseUrl": "https://12345678589CF19.uscom-east-1.oraclecloud.com:443",
6     "tokenEndpoint": "https://idcs-13ce01234567.identity.oraclecloud.com/oauth2/v1/token",
7     "backend": {
8         "backendId": "<YOUR_BACKEND_ID>",
9         "backendName": "<YOUR_BACKEND_NAME>",
10        "backendVersion": "<YOUR_BACKEND_VERSION>",
11        "authorization": {
12            "anonymousKey": "<BACKEND_ANONYMOUS_KEY>",
13            "clientId": "<BACKEND_CLIENT_ID>",
14            "clientSecret": "<BACKEND_CLIENT_SECRET>"
15        }
16    },
17    "tools": {
18        "deployment": {
19            "tempDirectory": "c:\\tmp\\mcs-deploy"
20        },
21        "authorization": {
22            "clientId": "<OMCE_TM_APP_CLIENT_ID>",
23            "clientSecret": "<OMCE_TM_APP_CLIENT_SECRET>"
24        }
25    },
26    "container": {
27        "oracleMobileApiName": "OracleMobileAPI",
28        "port": 4000
29    },
30    "tests": {
31        "getComponents": {
32            "method": "GET",
33            "resource": "/components"
34        }
35    }
36}
37}
```

Local Development & Debugging

Run Custom Component in Local Container from Oracle Intelligent Bots

Starting the Local Container for Testing

Command:

omce-ccc <path to>/toolsConfig.json

```
...p/openweathermapccs — node • node /usr/local/bin/omce-ccc ~/Desktop/OpenWeatherMap/openweathermapccs/toolsConfig.json
[dhcp-10-175-63-23:openweathermapccs fnimphiu$ omce-ccc /Users/fnimphiu/Desktop/OpenWeatherMap/openweathermapccs/toolsConfig.json
 Ping OracleMobileAPI to verify that OracleMobileAPI-uri and authorization are correct.
OracleMobileAPI ping succeeded!
shell.js create console logger
The Node server is listening at port 4000
|
```

Query Components Hosted by Custom Component Service

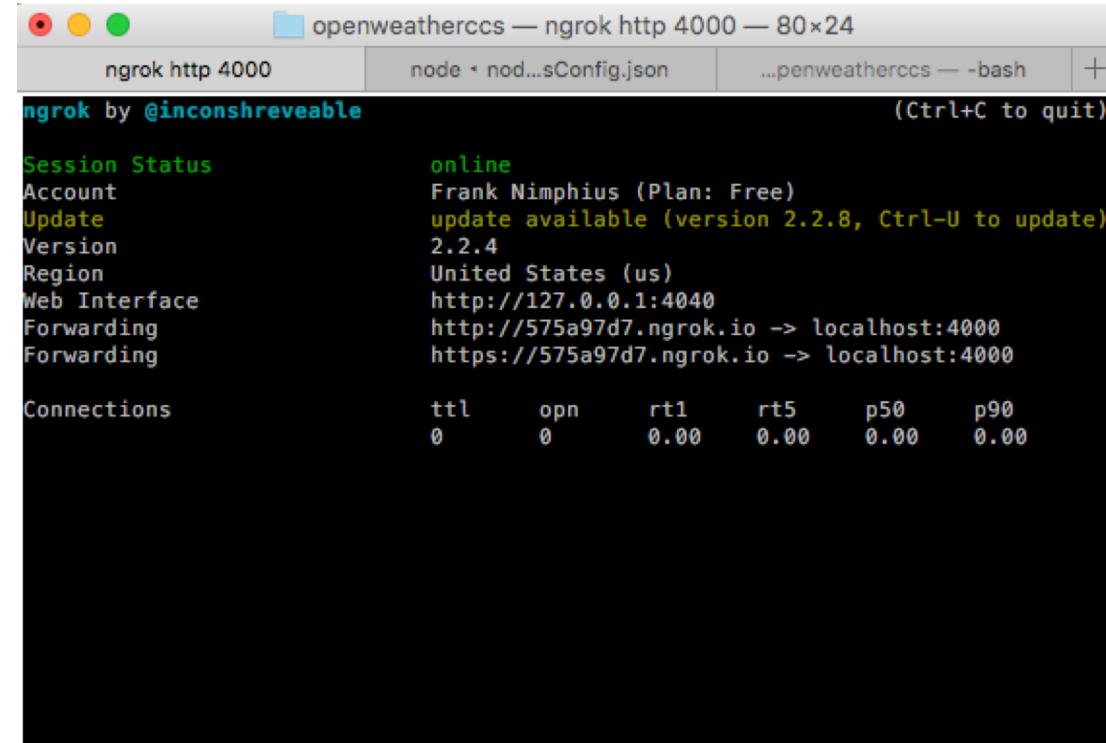
Command:

omce-test <path to>/toolsConfig.json getComponents

```
~/Desktop/OpenWeatherMap/openweathermapccs — bash ...p/openweathermapccs — node + node /usr/l...  
[fnimphiu-mac:openweathermapccs fnimphiu$ omce-test /Users/fnimphiu/Desktop/OpenWeatherMap/openweathermapccs/toolsConfig.json getComponents  
Executing test getComponents. Method: GET, URL: http://localhost:4000/mobile/custom/openweathermapccs/components  
Response body:  
{"version":"1.1","components":[{"name":"fn.weathercc","properties":{"city":{"type":"string","required":true}, "country":{"type":"string"} }]}  
Test getComponents completed successfully with status 200  
fnimphiu-mac:openweathermapccs fnimphiu$
```

Using ngrok

- Exposes local services to public internet
 - Secure tunnel
- Setup
 - Install ngrok from <https://ngrok.com/>
 - In a terminal window, type *ngrok http 4000* and press the enter key
- (Optional) Inspect traffic in browser
 - <http://localhost:4040>



A screenshot of a terminal window titled "openweathermapccs — ngrok http 4000 — 80x24". The window shows the output of the "ngrok" command. It displays session status information, including the account owner (Frank Nimphius), plan (Free), and update availability (version 2.2.8). It also shows the region (United States, us), web interface URL (http://127.0.0.1:4040), and two forwarding entries: http://575a97d7.ngrok.io and https://575a97d7.ngrok.io, both mapped to localhost:4000. Below this, a "Connections" table provides performance metrics: ttl (0), opn (0), rt1 (0.00), rt5 (0.00), p50 (0.00), and p90 (0.00).

Session Status	online					
Account	Frank Nimphius (Plan: Free)					
Update	update available (version 2.2.8, Ctrl-U to update)					
Version	2.2.4					
Region	United States (us)					
Web Interface	http://127.0.0.1:4040					
Forwarding	http://575a97d7.ngrok.io -> localhost:4000					
Forwarding	https://575a97d7.ngrok.io -> localhost:4000					
Connections	ttl	opn	rt1	rt5	p50	p90
	0	0	0.00	0.00	0.00	0.00

Running NGROK From Behind a Proxy

- Get the external IP address of your proxy
 - <http://www.whatismyproxy.com/>
- Issue an export statement in a terminal window
 - `export https_proxy=http://<external ip>:80`
 - `export http_proxy=http://<external ip>:80`
- Launch ngrok
 - `./ngrok http <port>`

Configure Custom Component in Oracle Intelligent Bots

Using the locally deployed custom component

The screenshot shows the Oracle Intelligent Bots Service configuration interface. On the left is a vertical toolbar with icons for Home, Services, Data, Analytics, Components, and Help. The main area has a green header bar with a '+ Service' button. Below it is a search bar labeled 'Filter' with a magnifying glass icon. A dropdown menu shows 'OpenWeatherMap' selected, with a sub-item 'fn.weathercc'. To the right, there are fields for 'Name' (set to 'OpenWeatherMap'), 'Description' (set to 'Optional short description for this service.'), and 'Version' (set to '1.1'). Below these are two radio buttons: 'Mobile Cloud' (selected) and 'Other'. Underneath are fields for 'Backend ID' (set to '10f4bcda-fbc0-4e1a-9107-8ac36bd69c0b') and 'Metadata URL' (set to 'https://13d2443e.ngrok.io/mobile/custom/openweatherccs/components'). A red box highlights the 'Metadata URL' field. There is also a checked checkbox for 'Use anonymous access' and a field for 'Anonymous Key' containing a series of dots. At the bottom, there is a section for 'Optional HTTP Headers'.



If your component does anything useful already, you can test it in the bot.

Local Development & Debugging

Starting the Local Container for Debugging

Starting the Local Container for Debugging

Command:

omce-ccc <path to>/toolsConfig.json --debug [--verbose]

--debug

Provides a URL you can use to open a debug session in Google Chrome.

--verbose

If you have errors or warnings, will show examples of the missing property in addition to a description of the property

Starting the AMC Local Container in Debug Mode

```
dhcp-10-175-22-71:/ fnimphiu$ omce-ccc /Users/fnimphiu/Desktop/OpenWeatherMap/openweathermapccs/toolsConfig.json --debug
Debugger listening on ws://127.0.0.1:9229/07cc0cf1-465e-4944-9ea2-750ea17cc30b
For help see https://nodejs.org/en/docs/inspector

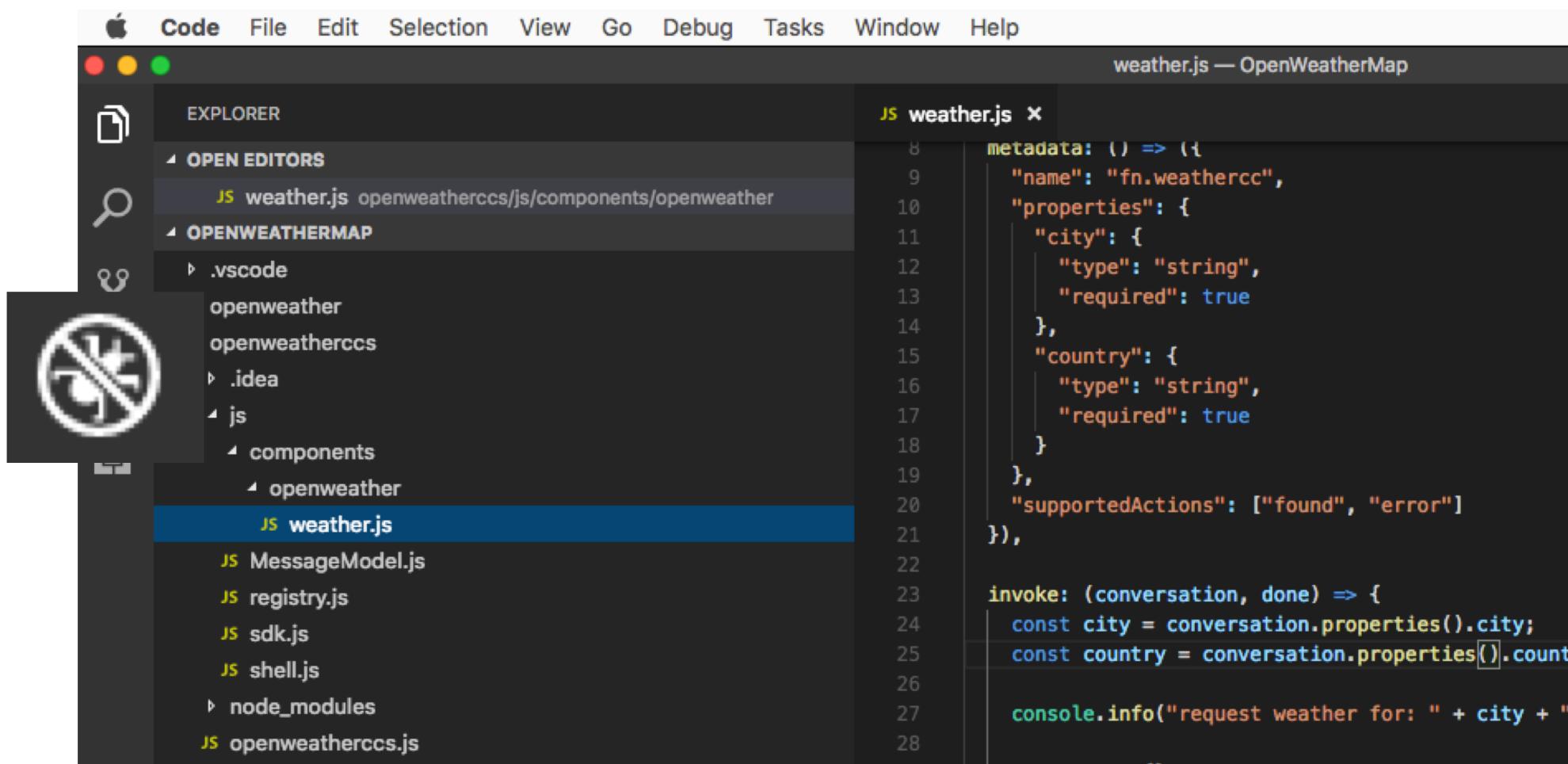
Warning: Configuration property "proxy" is undefined

To display help and examples associated with warnings, use the --verbose option
Ping OracleMobileAPI to verify that OracleMobileAPI-uri and authorization are correct.
OracleMobileAPI ping succeeded!
shell.js create console logger
The Node server is listening at port 4000
```

Local Development & Debugging

Debugging with Microsoft Visual Studio Code

Debugging With Microsoft Visual Studio Code IDE

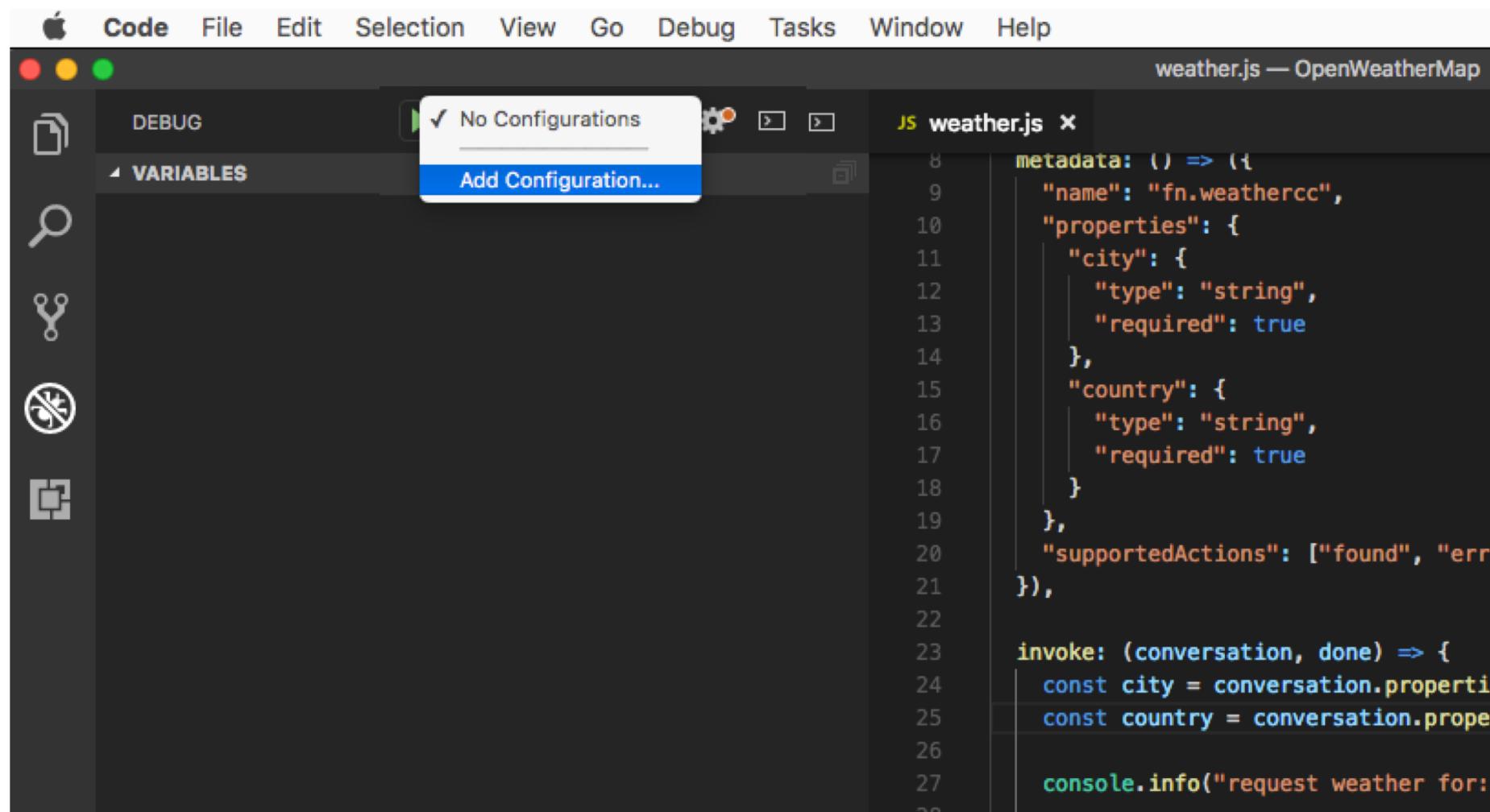


The screenshot shows the Microsoft Visual Studio Code interface. The top menu bar includes Code, File, Edit, Selection, View, Go, Debug, Tasks, Window, and Help. The title bar indicates the file "weather.js — OpenWeatherMap". The left sidebar (Explorer) displays a tree view of files and folders. The "OPEN EDITORS" section shows "weather.js openweathermap/js/components/openweather". The "OPENWEATHERMAP" section shows ".vscode", "openweather", "openweathermapccs", ".idea", "js", "components", "openweather", and "weather.js", with "weather.js" being the active editor. Other files listed include "MessageModel.js", "registry.js", "sdk.js", "shell.js", "node_modules", and "openweathermapccs.js". The main editor pane shows the code for "weather.js". The code defines a schema for a weather object, including properties for city and country, and supported actions found and error. It also includes an invoke function that logs a message to the console.

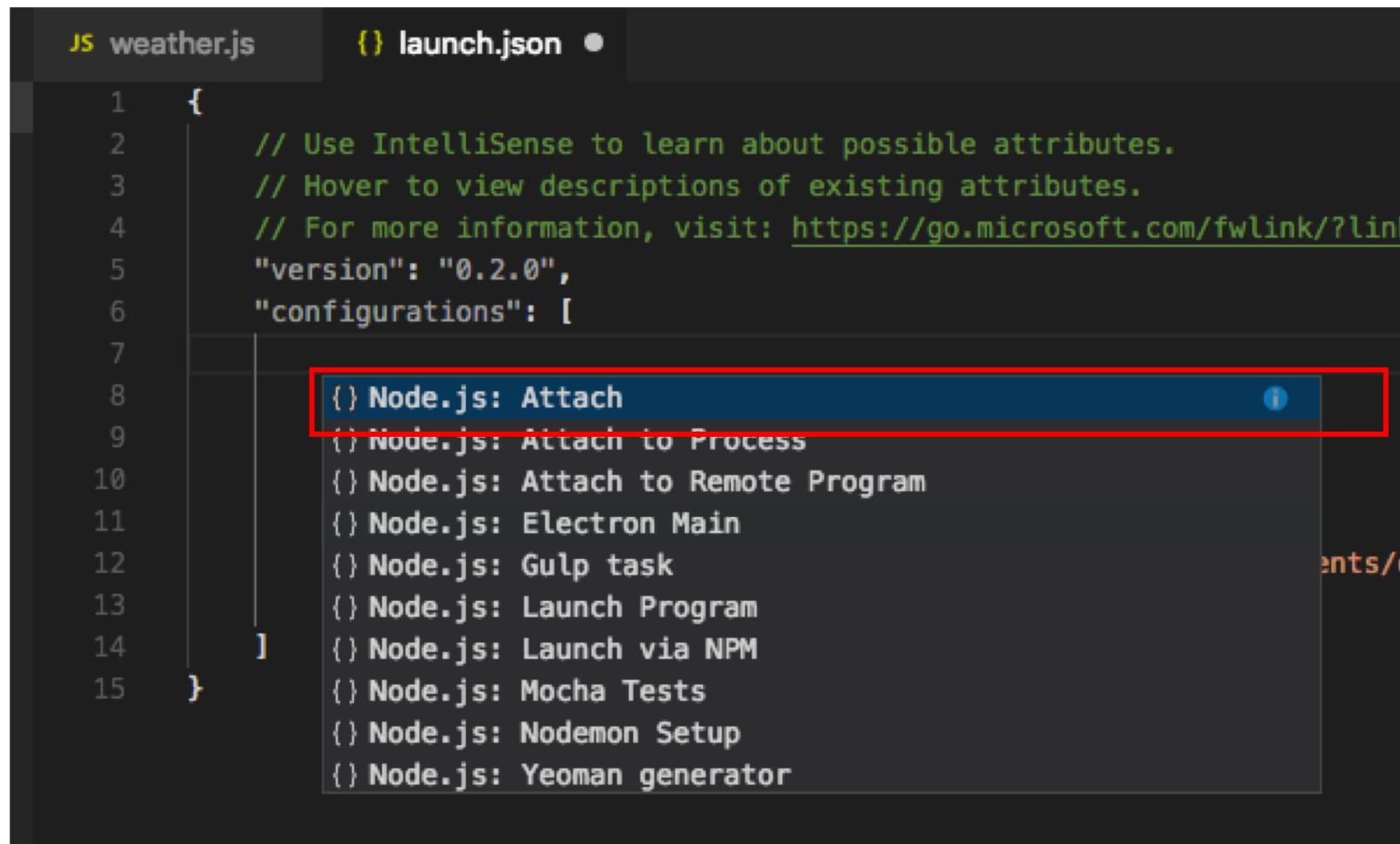
```
metadata: () => ({
  "name": "fn.weathercc",
  "properties": {
    "city": {
      "type": "string",
      "required": true
    },
    "country": {
      "type": "string",
      "required": true
    }
  },
  "supportedActions": ["found", "error"]
}),
invoke: (conversation, done) => {
  const city = conversation.properties().city;
  const country = conversation.properties().country;

  console.info("request weather for: " + city + ", " + country);
}
```

Debugging With Microsoft Visual Studio Code IDE



Debugging With Microsoft Visual Studio Code IDE



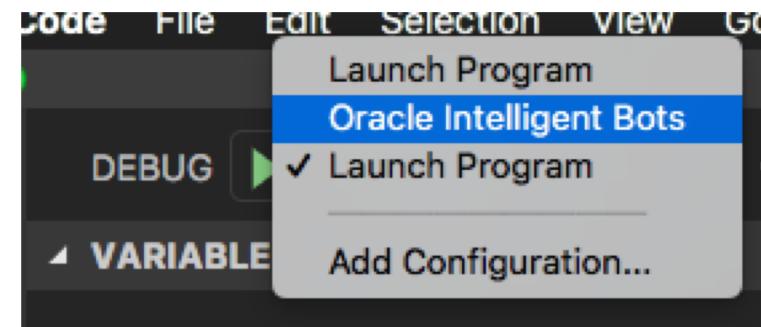
The screenshot shows the Microsoft Visual Studio Code interface with two tabs open: 'weather.js' and 'launch.json'. The 'weather.js' tab contains a simple Node.js script. The 'launch.json' tab shows a configuration object with a 'version' key and a 'configurations' array. A red box highlights the first item in the 'configurations' array, which is the 'Node.js: Attach' option. Other options listed include 'Attach to Process', 'Attach to Remote Program', 'Electron Main', 'Gulp task', 'Launch Program', 'Launch via NPM', 'Mocha Tests', 'Nodemon Setup', and 'Yeoman generator'.

```
1  {
2      // Use IntelliSense to learn about possible attributes.
3      // Hover to view descriptions of existing attributes.
4      // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830302
5      "version": "0.2.0",
6      "configurations": [
7          {
8              // Node.js: Attach
9              // Node.js: Attach to Process
10             // Node.js: Attach to Remote Program
11             // Node.js: Electron Main
12             // Node.js: Gulp task
13             // Node.js: Launch Program
14             // Node.js: Launch via NPM
15             // Node.js: Mocha Tests
16             // Node.js: Nodemon Setup
17             // Node.js: Yeoman generator
18         }
19     ]
20 }
```

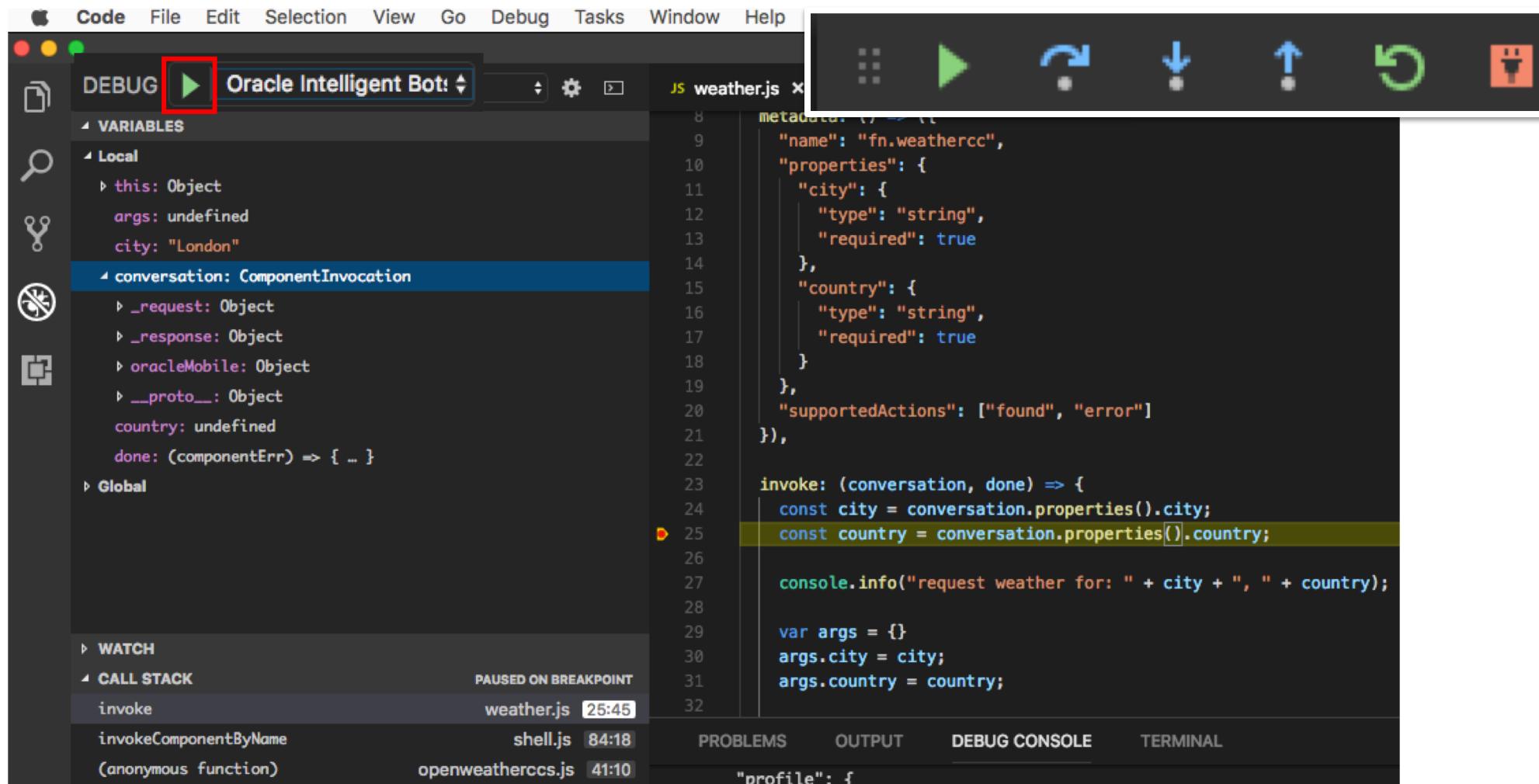
You can assign a custom name to the configuration



```
{  
  "type": "node",  
  "request": "attach",  
  "name": "Oracle Intelligent Bots",  
  "port": 9229  
},
```



Debugging With Microsoft Visual Studio Code IDE



The screenshot shows the Microsoft Visual Studio Code interface with the following details:

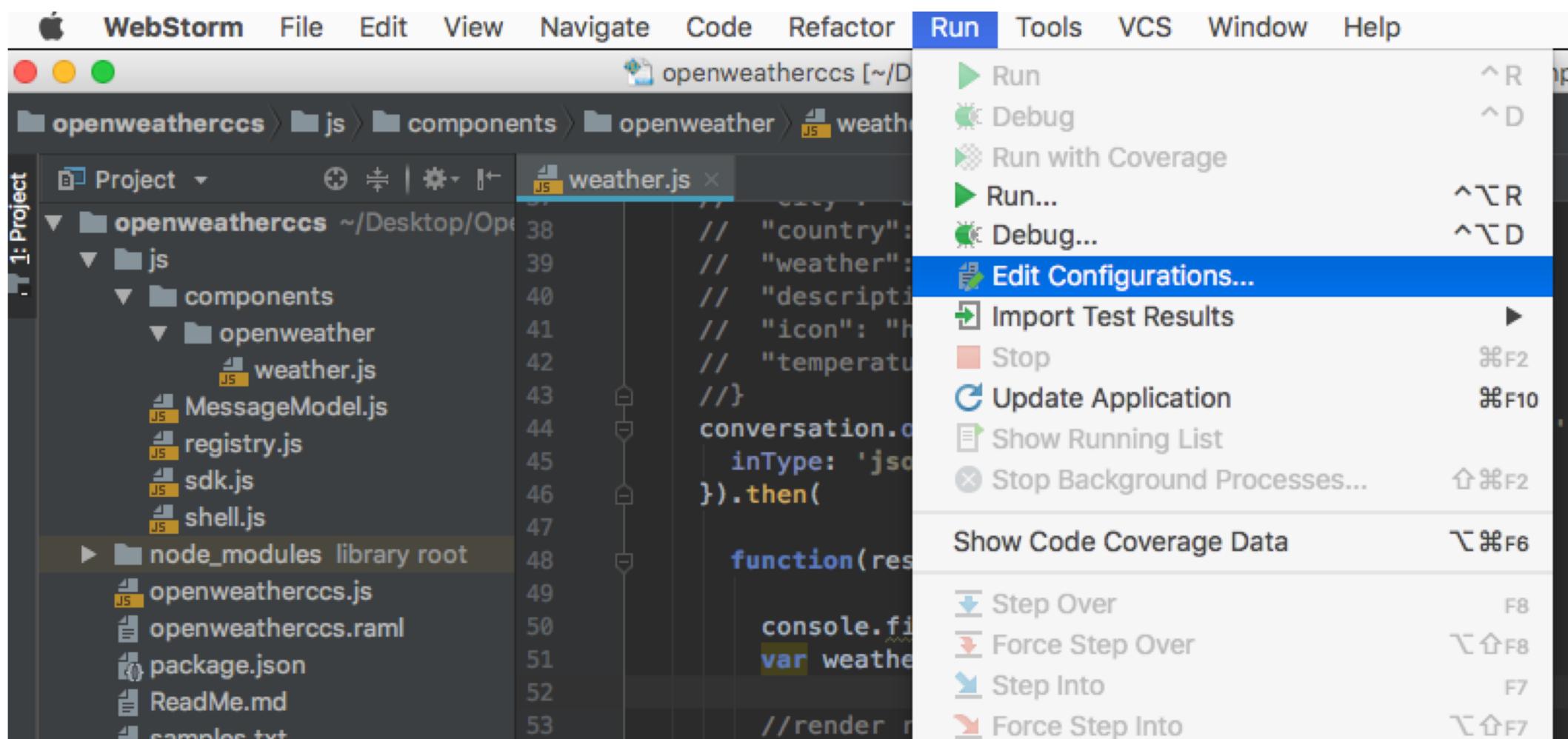
- Top Bar:** Code, File, Edit, Selection, View, Go, Debug, Tasks, Window, Help.
- Title Bar:** DEBUG (highlighted with a red box), Oracle Intelligent Bot: weather.js.
- Left Sidebar:** Variables, Watch, Call Stack. The Call Stack section shows "PAUSED ON BREAKPOINT" with entries: invoke (weather.js: 25:45), invokeComponentByName (shell.js: 84:18), (anonymous function) (openweathermapccs.js: 41:10).
- Center Area:** A code editor window titled "weather.js" showing the following code:

```
8 metadata: {
9   "name": "fn.weathercc",
10  "properties": {
11    "city": {
12      "type": "string",
13      "required": true
14    },
15    "country": {
16      "type": "string",
17      "required": true
18    }
19  },
20  "supportedActions": ["found", "error"]
21},
22
23 invoke: (conversation, done) => {
24   const city = conversation.properties().city;
25   const country = conversation.properties().country;
26
27   console.info("request weather for: " + city + ", " + country);
28
29   var args = {};
30   args.city = city;
31   args.country = country;
32}
```
- Right Side:** A toolbar with icons for step over, step into, step out, run, pause, and stop.

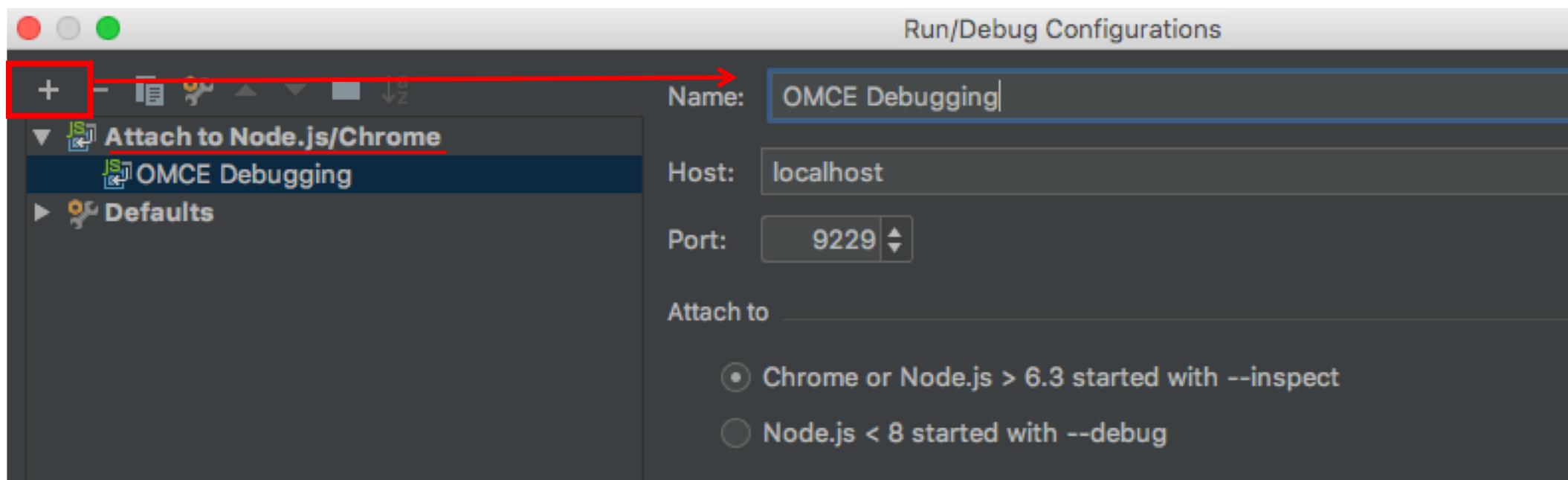
Local Development & Debugging

Debugging with IntelliJ Webstorm

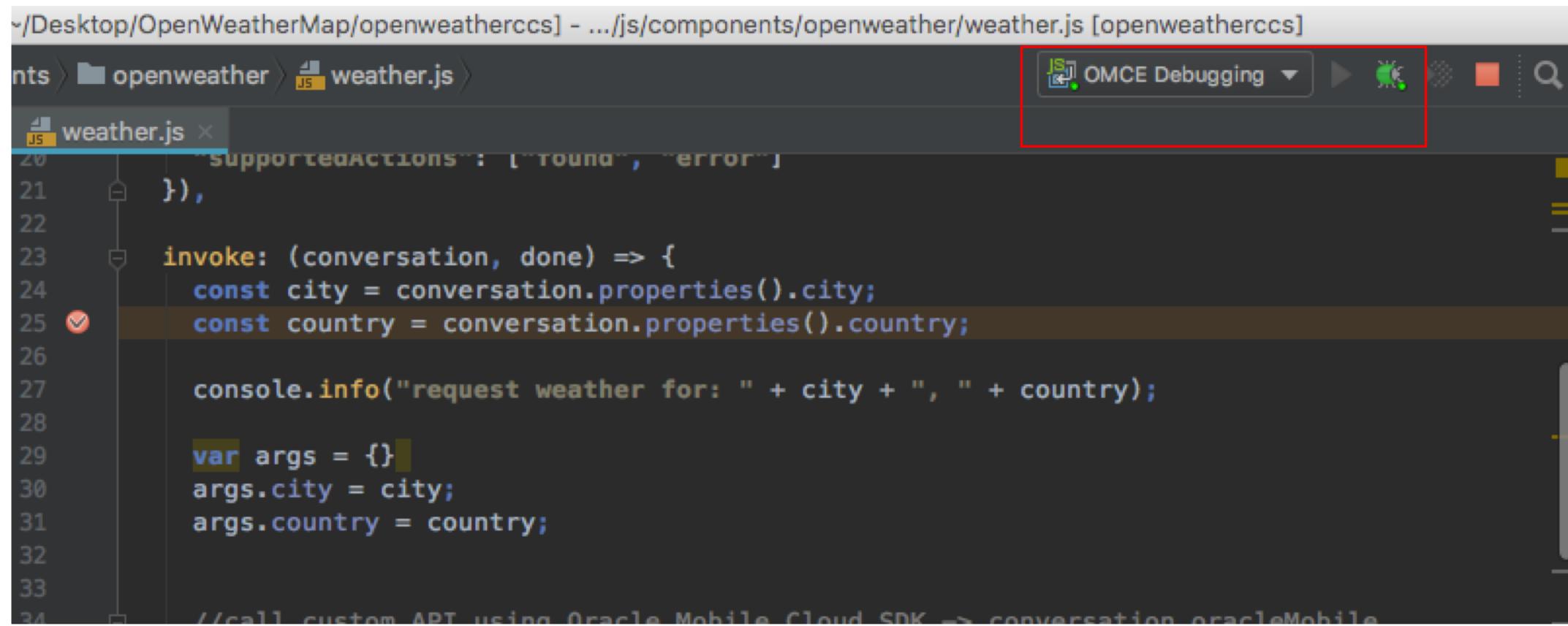
Debugging With IntelliJ Webstorm IDE



Debugging With IntelliJ Webstorm IDE



Debugging With IntelliJ Webstorm IDE



The screenshot shows the IntelliJ Webstorm IDE interface. The title bar indicates the current project is 'openweathermapccs' and the file being edited is 'weather.js'. The code editor displays the following JavaScript code:

```
~ /Desktop/OpenWeatherMap/openweathermapccs] - .../js/components/openweathermap/weather.js [openweathermapccs]
nts > openweathermap > weather.js >
JS weather.js ×
20     supportedActions: ['round', 'error'],
21
22
23     invoke: (conversation, done) => {
24         const city = conversation.properties().city;
25         const country = conversation.properties().country;
26
27         console.info("request weather for: " + city + ", " + country);
28
29         var args = {};
30         args.city = city;
31         args.country = country;
32
33
34         //call custom API using Oracle Mobile Cloud SDK -> conversation.oracleMobile
```

A red box highlights the top right corner of the IDE window, specifically the toolbar area which includes the 'OMCE Debugging' dropdown and other standard debugging and navigation icons.

Debugging With IntelliJ Webstorm IDE

The screenshot shows the IntelliJ Webstorm IDE interface during a debugging session. The code editor displays a file named `weather.js` with the following content:

```
    "SUPPORT_ACTIONS": ["round", "error"],  
},  
  invoke: (conversation, done) => {  
  conversation: ComponentInvocation {_request: Object, _response: Object, _error: Object, _keepTurn: true, _context: Object, _oracleMobile: OracleMobile, _times: 1},  
  const city = conversation.properties().city; city: "London"  
  const country = conversation.properties().country; country: undefined  
  console.info("request weather for: " + city + ", " + country); console.info(args);  
  var args = {};  
  args.city = city; city: "London"  
  args.country = country; country: undefined  
  //call custom API using Oracle Mobile Cloud SDK -> conversation.oracleMobile.  
  // returns:  
  // {  
  module.exports = invoke();  
}
```

The line `const country = conversation.properties().country; country: undefined` is highlighted with a red box, indicating it is the current line of execution.

In the bottom right corner, the `Variables` tool window is open, showing the local scope variables:

- `city = "London"`
- `conversation = ComponentInvocation {_request: Object, _response: Object, _error: Object, _keepTurn: true, _context: Object, _oracleMobile: OracleMobile, _times: 1}`
- `oracleMobile = Object {}`
- `_request = Object {botId: "36134306-5350-4E08-B2A5-4D9FEF8E9DF1", platformVersion: "1.1", context: Object, action: undefined, keepTurn: true, transactionId: undefined}`
- `_response = Object {platformVersion: "1.1", context: Object, action: undefined, keepTurn: true, transactionId: undefined}`
- `_proto_ = Object {}`
- `country = undefined`
- `this = undefined`
- `Functions`
- `Global = global`

A red box highlights the `country = undefined` entry in the `Variables` list, corresponding to the variable in the code editor.

Discussion

Lets assume a **custom component service project** hosts **3 custom components**. How many **toolsConfig.json** files do you need?

And why?





Advanced Bot Training Hands-On

Lab 5a: Setup Local Debugging

ORACLE®