

ORACLE®

Oracle Digital Assistant

The Complete Training

Custom Component Development with Mobile Hub

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.

Topic agenda

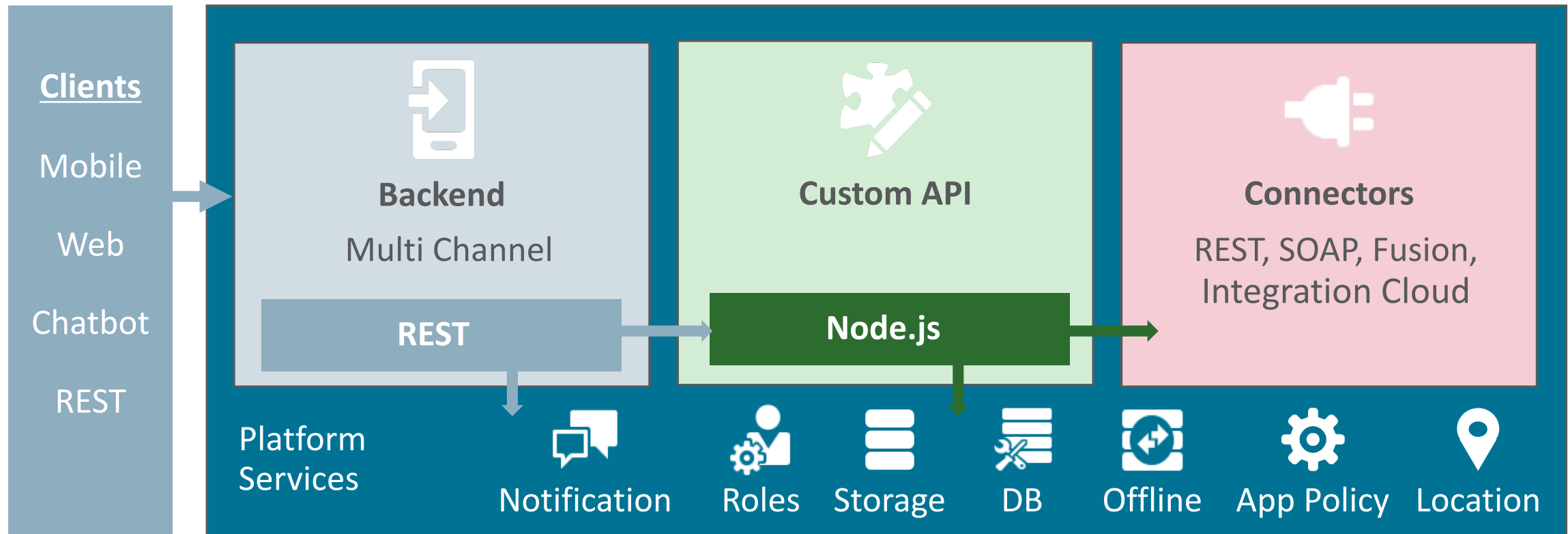
- 1 ➤ Mobile Hub introduction
- 2 ➤ Custom component services in Mobile Hub
- 3 ➤ Building custom components in Mobile Hub
- 4 ➤ Backend integration
- 5 ➤ Local development and debugging

Topic agenda

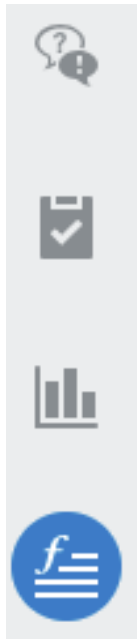
- 1 Mobile Hub introduction
- 2 Custom component services in Mobile Hub
- 3 Building custom components in Mobile Hub
- 4 Backend integration
- 5 Local development and debugging

Oracle Mobile Hub

Multi channel backend



Custom component service deployment options



Oracle Digital Assistant Skill

A 'Create Service' dialog box with a dark header and a close button. It contains a form with the following fields: a required 'Name' field with the placeholder 'Service name', and a 'Description' field with the placeholder 'Optional short description for this service'. Below these fields are three radio buttons: 'Embedded Container' (selected), 'Oracle Mobile Cloud', and 'External'. At the bottom, there is a 'Package File' section with a question mark icon, a dashed box containing a download icon and the text 'Upload a component package file (.tgz file created by npm pack) or drag it here.', and a green 'Create' button.

Oracle Mobile Hub

Skill bot Local Container

3rd Party Node Containers

Mobile Hub benefits

- Multi channel backend service
 - API sharing between web, mobile and bot applications
 - Secure API access
 - Payload shaping
 - Platform services: storage, analytics, database, location, push etc.
- API and API Implementation versioning
- Declarative REST, SOAP and Fusion Apps connectors
- Single point of administration and maintenance
- Diagnostics

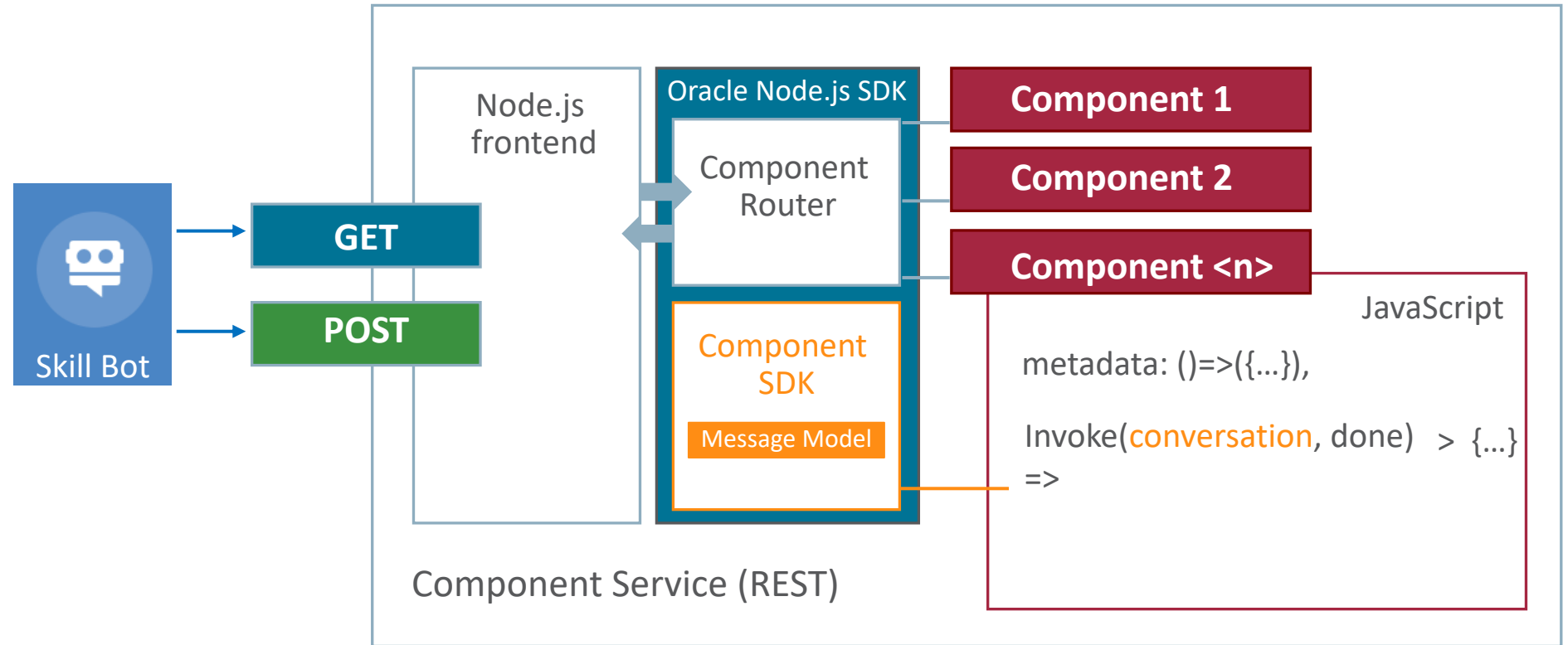
Topic agenda

- 1 Mobile Hub introduction
- 2 Custom component services in Mobile Hub**
- 3 Building custom components in Mobile Hub
- 4 Backend integration
- 5 Local development and debugging

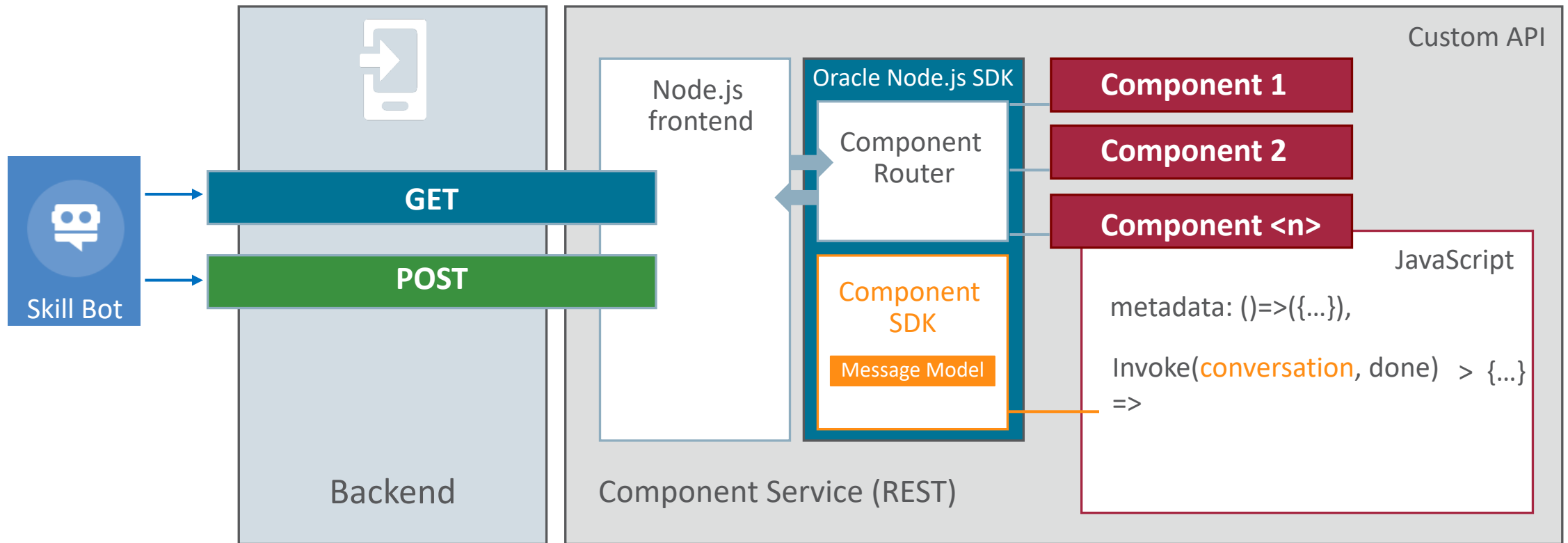
Custom component service development in Mobile Hub

- Custom component service built as Custom API
 - Component Service exposed through a Backend
 - Access to Mobile Hub services and SDK
 - Leverages Mobile Hub connector framework (REST, SOAP, Fusion, ICS)
- Component service API and Implementation versioned in Mobile Hub
 - Node programming
- Logs and diagnostic information saved in backend analytics

Custom component service architecture



Custom component service architecture in Mobile Hub



Topic agenda

- 1 Mobile Hub introduction
- 2 Custom component services in Mobile Hub
- 3 Building custom components in Mobile Hub**
- 4 Backend integration
- 5 Local development and debugging

Installing Oracle Node.js SDK

- Global installation provides the command line to create custom components
- Requires Node and Node Package Manager (NPM) to be installed
- Open a terminal window and type

MAC / Linux

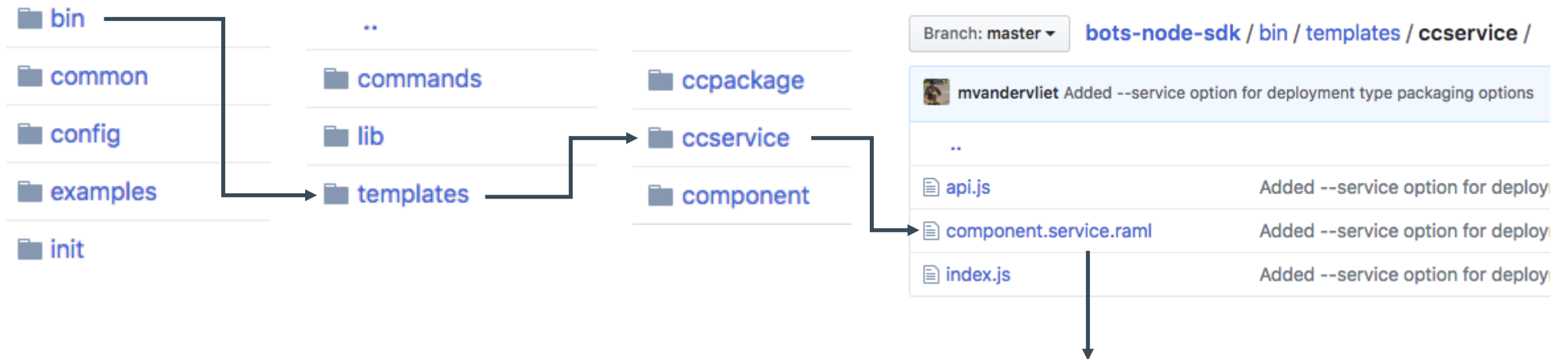
```
sudo npm install -g @oracle/bots-node-sdk
```

Windows

```
npm install -g @oracle/bots-node-sdk
```


Downloading the custom API template

- Oracle Mobile Hub Custom API starter template is available in Oracle Bots Node.js SDK on GitHub
 - <https://github.com/oracle/bots-node-sdk>
 - Defines GET and POST methods required for Bot custom component services



<https://github.com/oracle/bots-node-sdk/blob/master/bin/templates/ccservice/component.service.raml>

Creating a custom component service API

The image illustrates the process of creating a custom component service API in Oracle API Designer. It consists of three main panels:

- New API Dialog:** Shows the 'New API' button and the 'API' selection. The 'API Display Name' is 'helloworldCCS', the 'API Name' is 'helloworldCCS', and the 'Short Description' is 'hello world custom component s'. A 'Create' button is at the bottom.
- API Configuration Panel:** Shows the 'General' tab for 'helloworldCCS 1.0'. It includes fields for 'API Display Name', 'API Name', and 'Default Media Type'. A 'Save' button is highlighted.
- API Editor Panel:** Shows the 'helloworldCCS.raml' file. The first four lines are highlighted in a red box:

```
1 #%RAML 0.8
2 title: helloworldCCS
3 version: 1.0
4 baseUri: /mobile/custom/helloworldCCS
```

 A 'Save' button is highlighted in the top right corner.

Annotations and arrows indicate the workflow:

- 'Create new API' points to the 'New API' dialog.
- 'Open API' points to the 'API' selection in the 'New API' dialog.
- 'Copy & paste RAML (keep title, version, baseUri)' points to the first four lines of the RAML file.

Custom component service endpoints

DEVELOPMENT > APIS > helloworldCCS 1.0 Save Test

General

Endpoints

Security

Schema

Types

Traits

+ New Resource

Compact Mode ☐

+ X

/ components

Components context root

/components

Display Name

Resource Type

G

Methods >

+ X

/components/ {component}

Component invocation

/components/{component}

Display Name

Resource Type

P

Methods >

Disable login requirement

DEVELOPMENT > APIS > helloworldCCS 1.0

Save

Test



General



Endpoints



Security



Schema



Types



Login Required controls whether credentials are required to test this API's endpoints. When enabled, credentials are required. Additionally, Enterprise mobile users must have at least one of the roles selected here to access the API. When disabled, credentials are not required. These properties aren't saved in the associated RAML document, so you won't see them in the Source view.



[Tell me more about API security](#)

Login Required

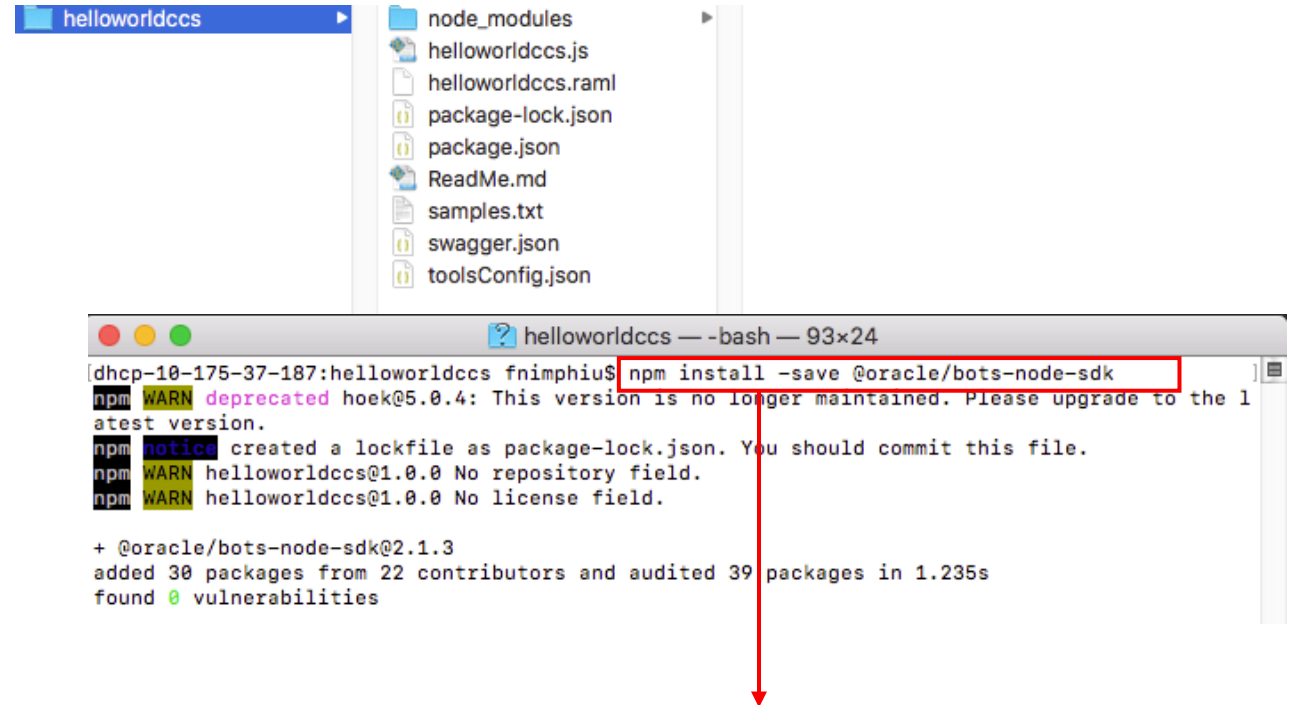


Downloading the custom API scaffold

The screenshot shows a web interface for API implementation. On the left is a sidebar with four items: 'Types' (with a square icon), 'Traits' (with a circle icon), 'Documentation' (with a document icon), and 'Implementation' (with a code icon and highlighted in blue). The main area has a heading 'You don't have any API implementa' and a subheading 'Download a JavaScript scaffold of your API to help you get started, or implementation is ready to go.' Below this is a link 'Tell me what's expected in my implementation archiv' and a green button labeled 'JavaScript Scaffold' with a download icon, which is highlighted by a red rectangular box.

Setting up the local development environment

- Unzip the downloaded scaffold
- Open command line and navigate into custom API root folder
 - Folder that contains package.json
- Install Oracle Bots Node.js SDK locally
 - Provides custom component SDK
 - Handles component request routing



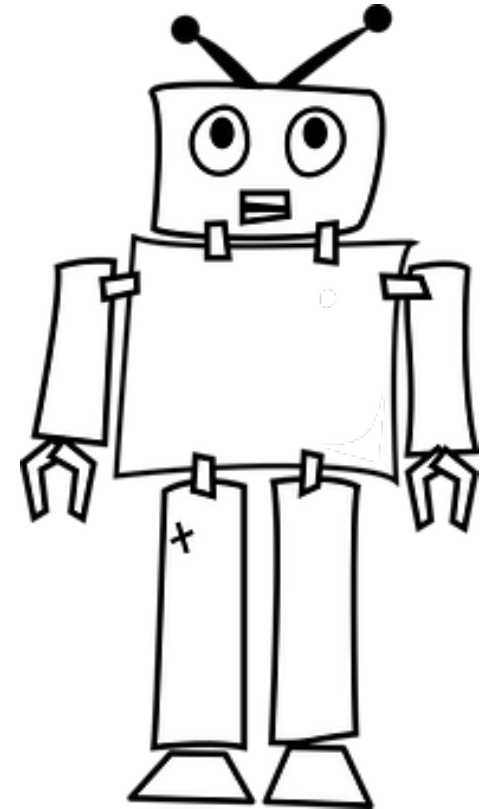
The screenshot shows a file explorer window for a directory named 'helloworldccs'. The files listed are: node_modules, helloworldccs.js, helloworldccs.raml, package-lock.json, package.json, ReadMe.md, samples.txt, swagger.json, and toolsConfig.json. Below the file explorer is a terminal window titled 'helloworldccs — -bash — 93x24'. The terminal shows the command 'npm install -save @oracle/bots-node-sdk' being executed. The output includes several warnings and a notice about creating a lockfile. A red box highlights the command, and a red arrow points from it to a code block below.

```
dhcp-10-175-37-187:helloworldccs fnimphiu$ npm install -save @oracle/bots-node-sdk
npm WARN deprecated hoek@5.0.4: This version is no longer maintained. Please upgrade to the latest version.
npm NOTICE created a lockfile as package-lock.json. You should commit this file.
npm WARN helloworldccs@1.0.0 No repository field.
npm WARN helloworldccs@1.0.0 No license field.

+ @oracle/bots-node-sdk@2.1.3
added 30 packages from 22 contributors and audited 39 packages in 1.235s
found 0 vulnerabilities
```

```
npm install @oracle/bots-node-sdk
```

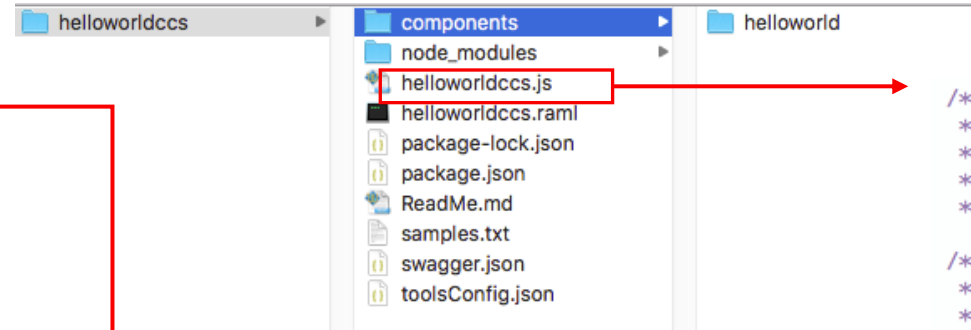
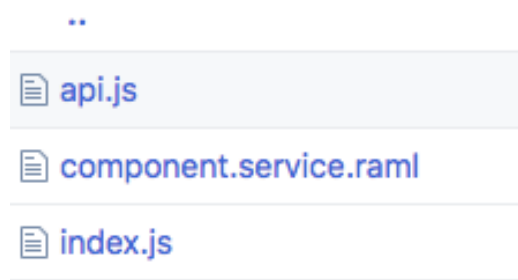
For better code organization, **create a "components" folder** and then a folder for each custom component you build therein



Editing the custom API

<https://github.com/oracle/bots-node-sdk/blob/master/bin/templates/ccservice/api.js>

Copy Content from api.js



Copy Content

```
/**
 * The ExpressJS namespace.
 * @external ExpressApplicationObject
 * @see {@link http://expressjs.com/3x/api.html#app}
 */

/**
 * Mobile Cloud custom code service entry point.
 * @param {external:ExpressApplicationObject}
 * service
 */
module.exports = function (service) {

  const OracleBot = require('@oracle/bots-node-sdk');
  OracleBot.init(service);

  // implement custom component api
  OracleBot.Middleware.customComponent(service, {
    baseUrl: '/mobile/custom/helloworldCCS/components',
    cwd: __dirname,
    register: [
      './components'
    ]
  });
};
```

Edit baseUrl, cwd, register properties

Custom component service code explained

```
module.exports = function (service) {  
  
  const OracleBot = require('@oracle/bots-node-sdk');  
  OracleBot.init(service);  
  
  // implement custom component api  
  OracleBot.Middleware.customComponent(service, {  
    baseUrl: '/mobile/custom/helloworldCCS/components',  
    cwd: __dirname,  
    register: [  
      './components'  
    ]  
  });  
  
};
```

—————→ Node module definition

—————→ Load command for Bot Node.js SDK

—————→ Request to custom component router

—————→ REST URI to invoke custom component (should match your API URI)

—————→ Relative folder custom components are Searched in

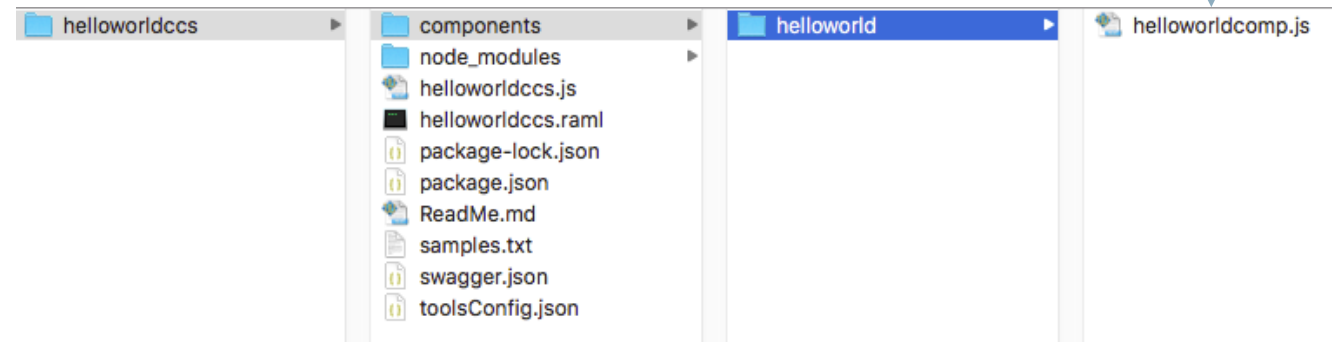
Creating a custom component

```
[fnimphiu-orcl:helloworldccs fnimphiu$ bots-node-sdk init component --name helloworldcomp components/helloworld
```

```
Writing file: /Users/fnimphiu/Downloads/helloworldccs/components/helloworld/helloworldcomp.js
```

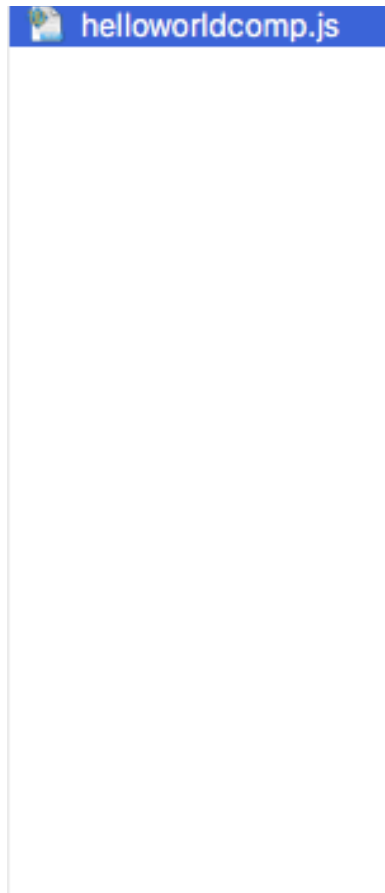
```
-----  
Added Custom Component: 'helloworldcomp'  
-----
```

```
fnimphiu-orcl:helloworldccs fnimphiu$
```



```
bots-node-sdk init component --name helloworldcomp components/helloworld
```


Generated custom component file & code



```
'use strict';

module.exports = {
  metadata: () => ({
    name: 'helloworldcomp',
    properties: {
      human: { required: true, type: 'string' },
    },
    supportedActions: ['weekday', 'weekend']
  }),
  invoke: (conversation, done) => {
    // perform conversation tasks.
    const { human } = conversation.properties();
    // determine date
    const now = new Date();
    const dayOfWeek = now.toLocaleDateString('en-US', { weekday: 'long' });
    const isWeekend = [0, 6].indexOf(now.getDay()) > -1;
    // reply
    conversation
      .reply(`Greetings ${human}`)
      .reply(`Today is ${now.toLocaleDateString()}, a ${dayOfWeek}`)
      .transition(isWeekend ? 'weekend' : 'weekday');

    done();
  }
};
```

Annotations for the code:

- Component invocation name (points to `name: 'helloworldcomp'`)
- Component properties (points to `properties: { ... }`)
- Action transitions (points to `supportedActions: ['weekday', 'weekend']`)
- Function invoked at runtime (points to `invoke: (conversation, done) => { ... }`)
- Callback that must be called at the end (points to `done();`)

Deploying the custom component service to Mobile Hub

- compress project root folder to a *zip-file*
- Upload *zip-file* as custom API implementation
- Use embedded tester in Oracle Mobile Hub to test GET method
- Expose custom API on Mobile Hub backend

The screenshot displays the Oracle Mobile Hub interface. On the left is a sidebar with navigation icons and labels: General, Endpoints, Security, Schema, Types, Traits, Documentation, and Implementation (which is highlighted with a blue circle and icon). The main content area is titled 'Download a new JavaScript scaffold at any time to include changes you make to the API design.' Below this is a green button labeled 'JavaScript Scaffold'. A table lists API implementations with columns: Status, Default, Name, Version, and Uploaded. The first row shows a status of 'In Progress' (yellow circle with a slash), 'Default' (green checkmark), 'Name' 'helloworldccs', 'Version' '1.1.0', and 'Uploaded' 'Fri, 2/15/2019 14:05'. Below the table is a section for 'Dependencies for helloworldccs 1.1.0' showing a 'Mock' with version 'N/A'. At the bottom, there is a dashed box with a download icon and the text 'Upload an implementation archive or'.

Status	Default	Name	Version	Uploaded
In Progress	Yes	helloworldccs	1.1.0	Fri, 2/15/2019 14:05
		Mock	N/A	

'Deployment' to Oracle Digital Assistant

Component registration in Oracle Digital Assistant skill

Oracle Mobile Hub

DEVELOPMENT > BACKENDS > SayHelloBackend 1.0

Diagnostics

Settings

Clients

Security

APIs

Access Keys ?

OAuth Consumer

Refresh | Revoke

Client ID

8dcb27eea1994f7da8a4d27981252df8

Client Secret

Show

Environment URLs ?

Base URL

https://006B186491194B64A833A511C6F8A566

HTTP Basic



Refresh

Backend ID

97fa003a-8ed3-4fd1-b853-a3358baac132

Anonymous Key

Show

Skill

Create Service

* Name

MobileHubService

Description

Optional short description for this service

☐ Embedded Container ☒ Oracle Mobile Cloud ☐ External

* Backend ID

97fa003a-8ed3-4fd1-b853-a3358baac132

* Metadata URL

https://006B186491194B64A833A511C6F8A566.mobile.ocp.oraclecloud.com:443/mobile/c

☒ Use anonymous access

* Anonymous Key

MDA2QjE4NjQ5MTE5NEI2NEE4MzNBNTExQzZGOEE1NjZfTW9iaWxlQW5vbntb3VzX0FQU

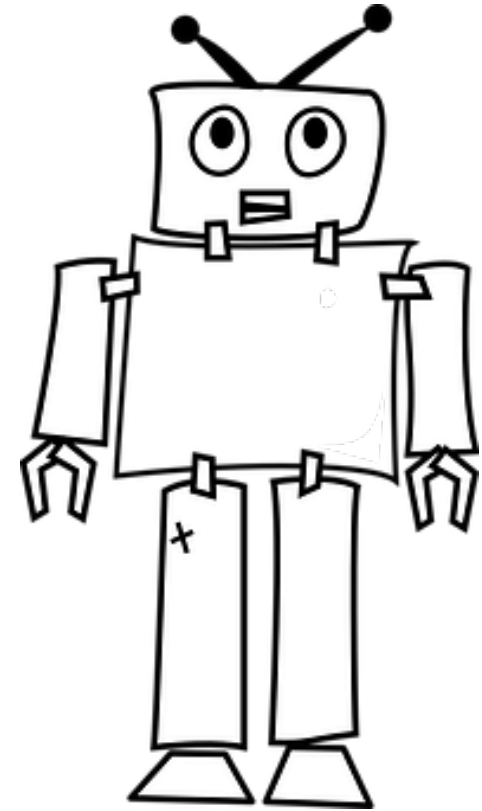
Optional HTTP Headers ?

Create

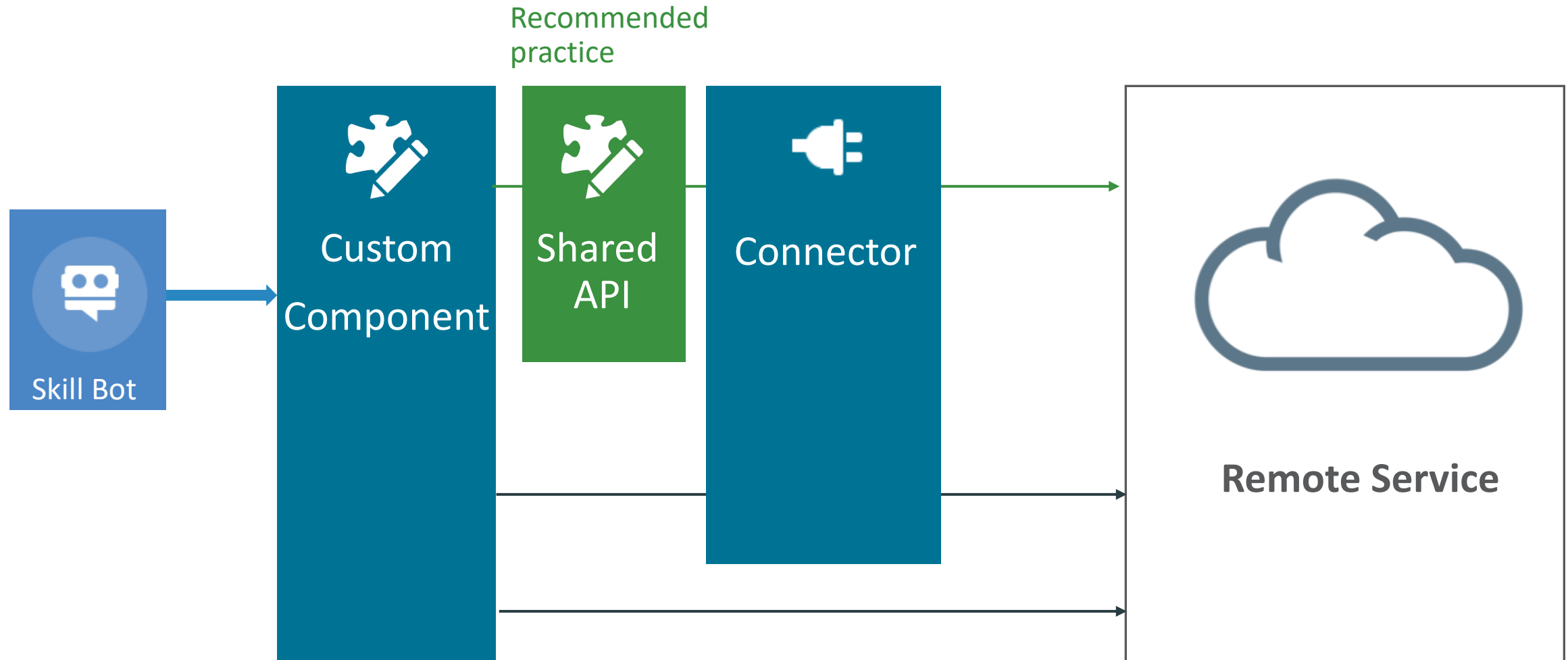
Topic Agenda

- 1 Mobile Hub introduction
- 2 Custom component services in Mobile Hub
- 3 Building custom components in Mobile Hub
- 4 Backend integration**
- 5 Local development and debugging

In addition to multi-channel support,
backend integration is a good argument
for using Oracle Mobile Hub to create
custom components in

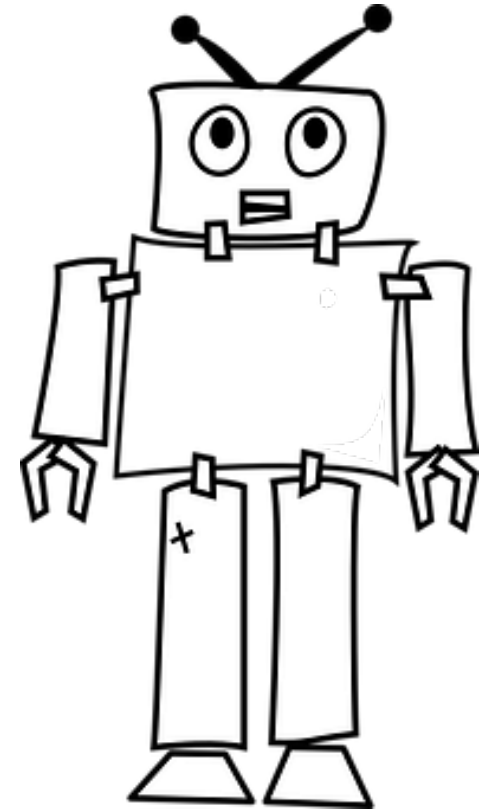


Mobile Hub backend integration options



You can access the **Oracle Mobile Hub SDK**
through the Custom Component SDK to
access Mobile Hub custom APIs, platform
APIs and Connectors

```
conversation.oracleMobile.<function>
```



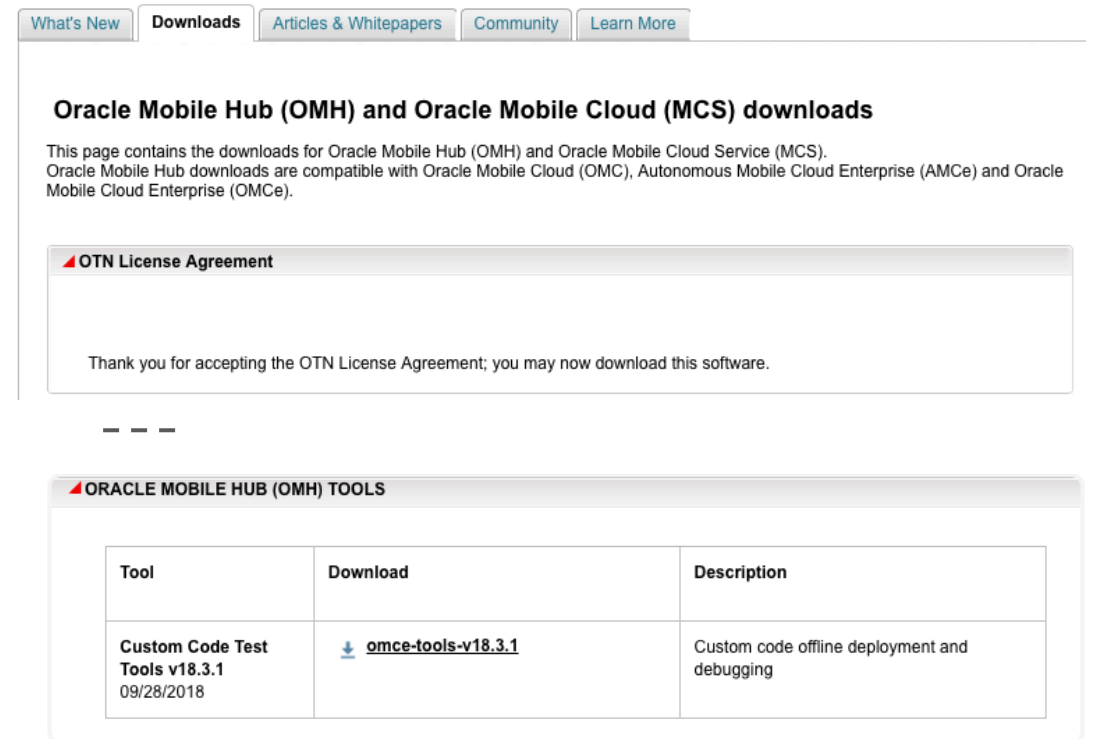
Topic agenda

- 1 Mobile hub introduction
- 2 Custom component services in Mobile Hub
- 3 Building custom components in Mobile Hub
- 4 Backend integration
- 5 Local development and debugging

Custom component debugging with Oracle Mobile Hub

- Install Oracle Custom Code Test Tool
 - Download from OTN
 - Follow instructions in readme
- Configure backend with code test tool reference
 - Code test tool proxy installed as custom API
- Have local copy of component service custom API
 - Configure toolsConfig.json with backend access
- Start Local Debugging
 - Code test tool command

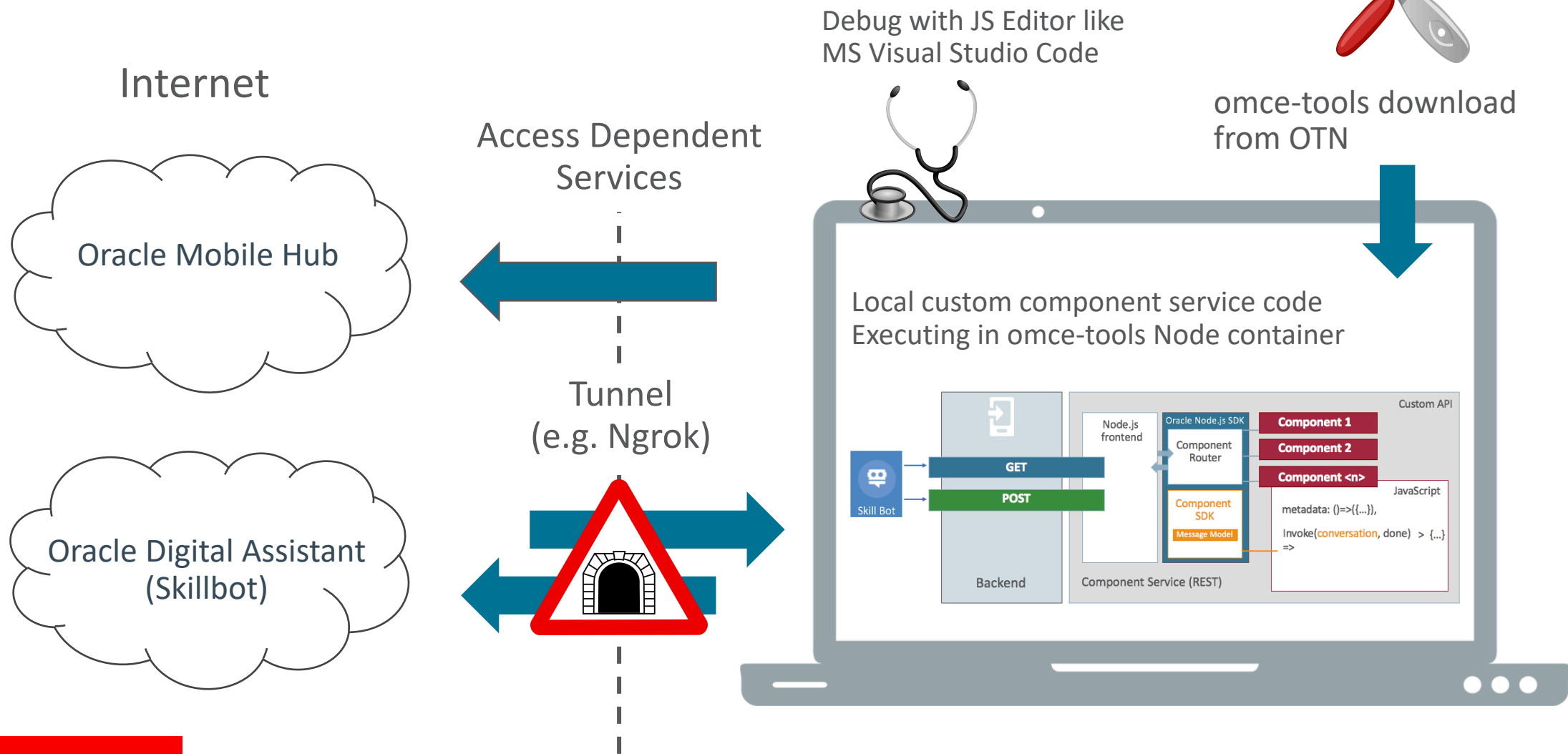
<https://www.oracle.com/technetwork/topics/cloud/downloads/mobile-cloud-service-3636470.html>



The screenshot shows the Oracle Mobile Hub (OMH) and Oracle Mobile Cloud (MCS) downloads page. The page has a navigation bar with links: What's New, Downloads, Articles & Whitepapers, Community, and Learn More. The main heading is "Oracle Mobile Hub (OMH) and Oracle Mobile Cloud (MCS) downloads". Below the heading is a paragraph stating: "This page contains the downloads for Oracle Mobile Hub (OMH) and Oracle Mobile Cloud Service (MCS). Oracle Mobile Hub downloads are compatible with Oracle Mobile Cloud (OMC), Autonomous Mobile Cloud Enterprise (AMCe) and Oracle Mobile Cloud Enterprise (OMCe)." Below this is a section titled "OTN License Agreement" with a message: "Thank you for accepting the OTN License Agreement; you may now download this software." Below this is a section titled "ORACLE MOBILE HUB (OMH) TOOLS" containing a table with the following data:

Tool	Download	Description
Custom Code Test Tools v18.3.1 09/28/2018	omce-tools-v18.3.1	Custom code offline deployment and debugging

Mobile Hub debugging architecture



Integrated Cloud

Applications & Platform Services

ORACLE®



Oracle Digital Assistant Hands-On

TBD