

ORACLE

Entity Event Handlers

Program agenda

- 1 Introduction
- 2 Entity event handler
- 3 Example - composite bag vs event handler
- 4 Creating an event handler
- 5 Deployment
- 6 Under the hood

Program agenda

- 1 **Introduction**
- 2 Entity event handler
- 3 Example - composite bag vs event handler
- 4 Creating an event handler
- 5 Deployment
- 6 Under the hood

Introduction

Problem statement

Composite bag entities are powerful

Bag items rely heavily on Apache FreeMarker expressions

- prompts, errorMessage, promptForValue, validationRule

Common problems

- Apache FreeMarker expressions quickly become complex
 - Error prone
 - Hard to debug
- No backend services access
- Cross referencing other bag items is cumbersome
 - No 'this.<bag_item>' style handle
 - Requires visual flow designer variable name

Error Message 

```
${system.entityToResolve.value.userInput!'this'}
```

Validation Rules

+ Validation Rule

Expression 

```
${(pizza.value.DeliveryTime.hrs?number < 10)?then('true','false')}
```

”

Everything has its limits!
Apache FreeMarker expressions
cannot replace programming.

Program agenda

- 1 Introduction
- 2 **Entity event handler**
- 3 Example - composite bag vs event handler
- 4 Creating an event handler
- 5 Deployment
- 6 Under the hood

Entity event handler

Event-driven approach to resolving Composite Bag Entity (CBE)

- Single event handler registered with entity
- Entity resolution event functions called for entity and bag items
- Developers implement functions for events they want to handle

Entity Event Handler (EEH)

- Special type of custom component
- Supported by Oracle Bots Node SDK
- JavaScript programming

Requires Resolve Composite Bag , Resolve Entities components

- Components detect registered EEH and call functions when resolving the composite bag entity

The screenshot displays the configuration page for the 'PizzaBag' entity. At the top, the entity name 'PizzaBag' is shown with a trash icon. Below this, the components 'PizzaDough' and 'PizzaSize' are listed. The main configuration area is divided into sections: 'General Information' and 'Configuration'. Under 'General Information', the 'Name' field is set to 'PizzaBag' and the 'Description' field is empty. Under 'Configuration', the 'Type' is set to 'Composite Bag'. A separate 'Event Handler' section shows a dropdown menu with 'pizzaeeh' selected, and a list of available event handlers including 'oda.PizzaEEH', 'pizzaeeh', and another 'pizzaeeh'.

Entity event handler

Benefits

Handle complex validation rules and logic with ease

- Easy to document and maintain

Can programmatically set any bag item value

- E.g. to set default values or values from a scan

Can access backend services during entity resolution

- No dependencies to dialog flow variables

Can display custom dialogs and Messages

- e.g. for users to confirm or approve an action

Support for local debugging



”

Apache FreeMarker expressions remain a valid option.

Program agenda

- 1 Introduction
- 2 Entity event handler
- 3 **Example - composite bag vs event handler**
- 4 Creating an event handler
- 5 Deployment
- 6 Under the hood

Example

Composite bag

- Let's use an example of a business rule validation within a composite bag
- In this case we use a FreeMarker expression to implement the rule
- At runtime the Resolve Entities or Resolve Composite Bag components can execute the expression and apply the validation

What if we need some more advanced validation logic? For that we can use EEH

Type

Composite Bag

Event Handler

Bag Items

Name	Type	Entity Name	Sequence Number
PizzaSize	ENTITY	PizzaSize	1
PizzaTopping	ENTITY	PizzaTopping	2
PizzaDough	ENTITY	PizzaDough	3
DeliveryTime	ENTITY	TIME	4

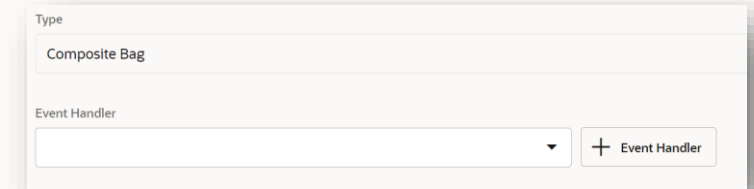
Validation Rules

Expression	Error Message
<code>\${(pizza.value.DeliveryTime.hrs?number < 10)?then('true','false')}</code>	Sorry, we only deliver up to 9:30pm

Example

Entity event handler

- In the composite bag entity we can add an Event Handler
- We provide a name for the service and for the handler
- This creates the framework with metadata and handlers on top of which we can select the target bag item and the desired event




Type

Composite Bag

Event Handler

+ Event Handler



Create Event Handler

Service Name

eeh.pizza

Handler Name

eeh.validate

Create



Edit Event Handler Code

+ Add Event

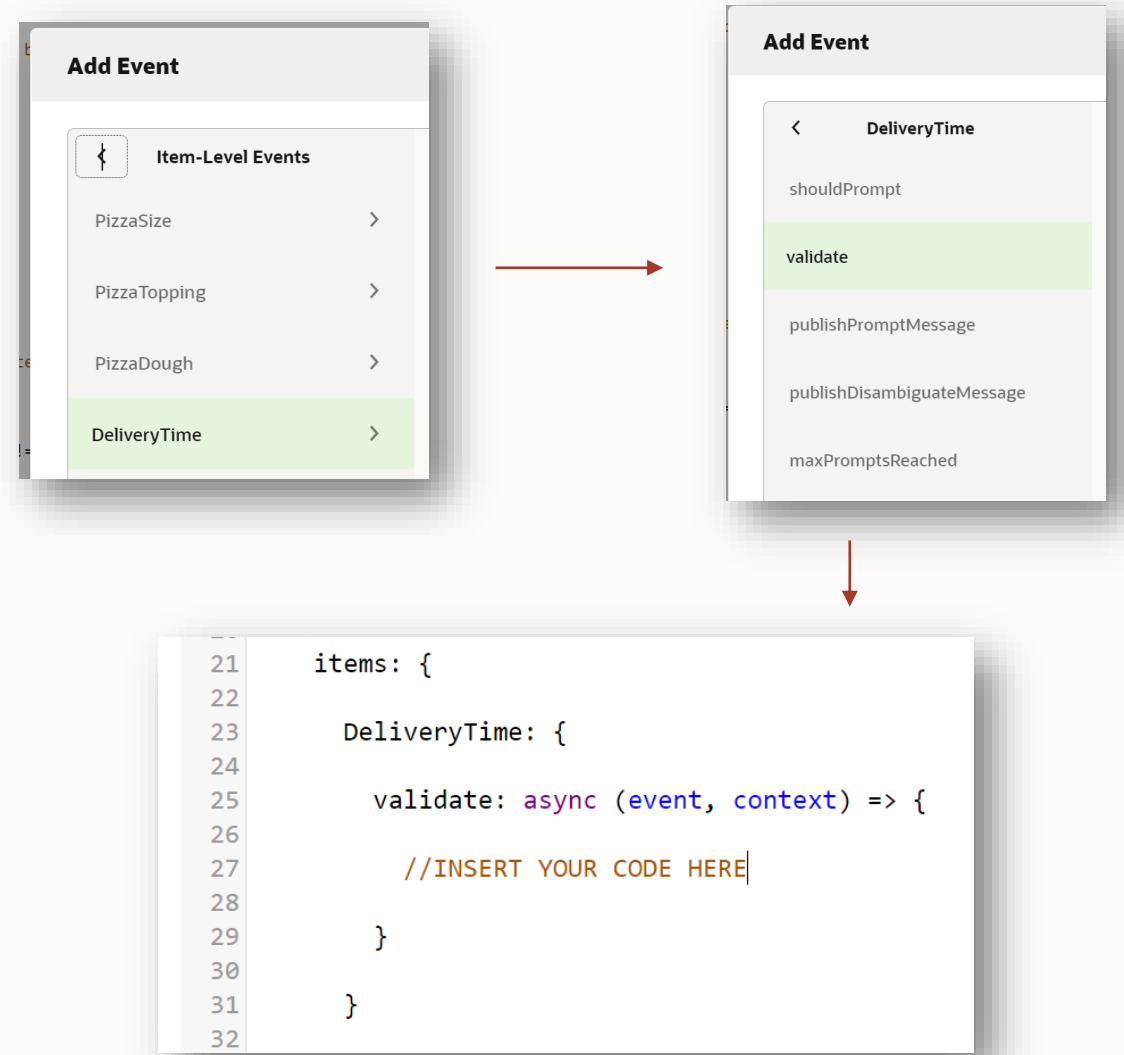
```
1 'use strict';
2
3 // node fetch API can be used to make REST calls, see https://www.npmjs.com/package/node-fetch
4 const fetch = require("node-fetch");
5
6 module.exports = {
7   metadata: {
8     name: 'eeh.validate',
9     eventHandlerType: 'ResolveEntities',
10    supportedActions: [] // string array of transition actions that might be set by the event handler
11  },
12  handlers: {
13    entity: {
14      publishMessage: async (event, context) => {
15        updatedItemsMessage(context);
16        outOfOrderItemsMessage(context);
17        context.addCandidateMessages();
18      }
19    }
20  }
21 };
```

Example

Entity event handler

- Then we choose the desired Bag Item
 - DeliveryTime
- And we select the type of event
 - validate
- This will add the framework where we can implement the required logic

Whenever an event handler is implemented, the Resolve Entities or Resolve Composite Bag components know, at runtime, that they should execute that code. This is out of the box, no need to explicitly configure this.



Program agenda

- 1 Introduction
- 2 Entity event handler
- 3 Example - composite bag vs event handler
- 4 **Creating an event handler**
- 5 Deployment
- 6 Under the hood

Creating an event handler

Development choices

Built-in code editor

Easy to use

Context dialog shows event functions that can be added for bag item or entity

No packaging and deployment

- No option to add external Node modules

Code is auto-saved

- OCI login session expiry may be a problem

No debugger

External JavaScript IDE

Advanced option

Oracle Bots Node SDK needed

- Generates Entity Event Handler skeleton

Component needs to be packaged, deployed and registered with composite bag entity

Local debugging possible

Local code project can be source controlled

Creating an event handler

Built-in editor

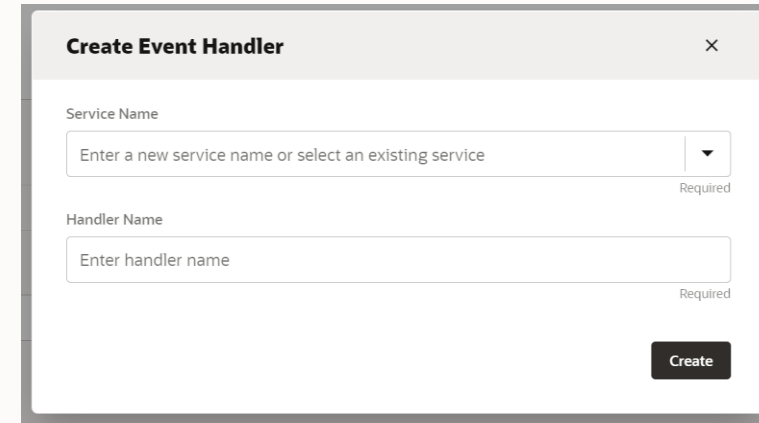
Launched from composite bag entity

- Service name becomes package name
- Handler name become component name

Context dialogs

- Allow to select bag item for which to implement an event function
- Code comments explain functions
 - Arguments
 - What a function is for

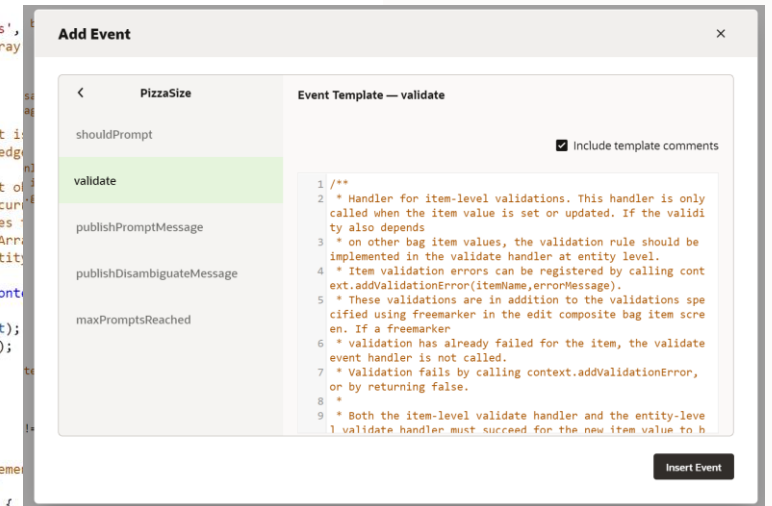
Closing the code editor deploys or re-deploys the component package



The 'Create Event Handler' dialog features two input fields. The first is 'Service Name' with a dropdown arrow and a 'Required' label. The second is 'Handler Name' with a 'Required' label. A 'Create' button is located at the bottom right.

Edit Event Handler Code

```
+ Add Event
1 'use strict';
2
3 // node fetch API can be used to make REST calls, see https://www.npmjs.com/package/node-fetch
4 const fetch = require("node-fetch");
5
6 module.exports = {
7   metadata: {
8     name: 'pizzaehh',
9     eventHandlerType: 'ResolveEntities',
10    supportedActions: [] // string array
11  },
12  handlers: {
13    entity: {
14      /**
15       * Generic fallback handler that is
16       * Used here to provide acknowledgment
17       *
18       * @param {object} event - event of
19       * - currentItem: name of item currently
20       * - promptCount: number of times
21       * - disambiguationValues: JSONArr
22       * @param {object} context - entity
23       */
24      publishMessage: async (event, context) => {
25        updatedItemsMessage(context);
26        outOfOrderItemsMessage(context);
27        context.addCandidateMessages();
28      }
29    }
30  }
31 };
32
33 /**
34  * Helper function to show acknowledgment
35  */
36 function updatedItemsMessage(context) {
37   if (context.getItemsUpdated().length > 0) {
38     let message = "I have updated"+context.getItemsUpdated().map((item, i) => (i!==0 ?
39     context.addMessage(message);
40   }
41 }
42
```



The 'Add Event' dialog shows a list of event types on the left, with 'validate' selected. The right pane shows the 'Event Template — validate' with a checkbox for 'Include template comments' and a text area containing the handler code. An 'Insert Event' button is at the bottom right.



Creating an event handler

External IDE

Any JavaScript IDE

Node and Node Package Manager (NPM) need to be installed

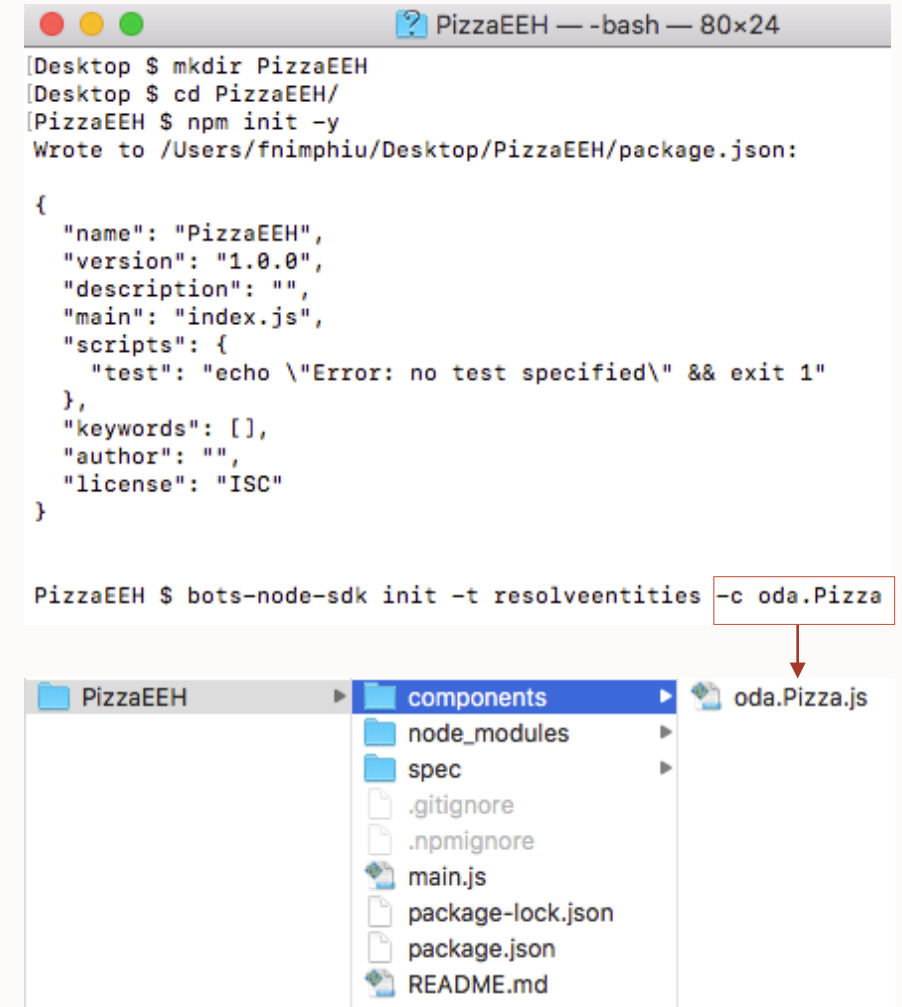
- Generates Entity Event Handler project

Install Oracle Bots Node SDK globally

- <https://github.com/oracle/bots-node-sdk>
- `npm install -g @oracle/bots-node-sdk`

Commands to create new project

- `cd <project folder>`
- `npm init -y`
- `bots-node-sdk init -t resolveentities -c <component name>`



The image shows a terminal window and a file explorer. The terminal window, titled "PizzaEEH — -bash — 80x24", shows the following commands and output:

```
[Desktop $ mkdir PizzaEEH
[Desktop $ cd PizzaEEH/
[PizzaEEH $ npm init -y
Wrote to /Users/fnimphiu/Desktop/PizzaEEH/package.json:

{
  "name": "PizzaEEH",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

PizzaEEH $ bots-node-sdk init -t resolveentities -c oda.Pizza
```

The file explorer shows the directory structure of the PizzaEEH project. The "components" folder is expanded, showing the following files and folders:

- node_modules
- spec
- .gitignore
- .npmignore
- main.js
- package-lock.json
- package.json
- README.md

The file "oda.Pizza.js" is highlighted in the file explorer, and a red arrow points from the terminal command to this file.

Program agenda

- 1 Introduction
- 2 Entity event handler
- 3 Example - composite bag vs event handler
- 4 Creating an event handler
- 5 **Deployment**
- 6 Under the hood

Deployment

Packaging

Only required when using external IDE

- Built-in editor automatically deploys changes

Navigate to EEH project root folder

- In terminal window or command line
- Ensure **npm install** is called
 - Ensure Node dependencies to be installed

Issue **bots-node-sdk pack** command

Creates "<project name>.tgz" file

- Deployment file

```
PizzaEEH — -bash — 80x34
-----
PizzaEEH $ bots-node-sdk pack
-----
Preparing artifact from: PizzaEEH...
-----
> PizzaEEH@1.0.0 prepack /Users/.../Documents/CodeSample
> npm run bots-node-sdk -- --pack --dry-run

> PizzaEEH@1.0.0 bots-node-sdk /Users/.../Documents/Code
H
> bots-node-sdk "--pack" "--dry-run"

npm notice
npm notice 📦 PizzaEEH@1.0.0
npm notice === Tarball Contents ===
npm notice 60B main.js
npm notice 4.5kB components/oda.PizzaEEH.js
npm notice 539B package.json
npm notice 2.4kB README.md
npm notice === Tarball Details ===
npm notice name: PizzaEEH
npm notice version: 1.0.0
npm notice filename: PizzaEEH-1.0.0.tgz
npm notice package size: 3.0 kB
npm notice unpacked size: 7.5 kB
npm notice shasum: 2bd96e45f9055d8e644b998517dd522c787
npm notice integrity: sha512-AfqRP+97dckIs[...]KL8ADwJq+u
npm notice total files: 4
npm notice
PizzaEEH-1.0.0.tgz
-----
Component package 'PizzaEEH-1.0.0.tgz' created successfully!
```

Deployment

Deployment options

- Local container
 - Drag&drop tgz file
- Remote server or cloud
 - Also good to use for local EEH debugging

Use registration field on CCB

- Choice displays component services with EEH
- One EEH can be selected per CBE

Create Service

Name: oda.PizzaEEH

Description: Optional short description for this service

Service Type: Embedded Container Oracle Mobile Cloud External Oracle Function

Package File: **Drag and Drop** (Select or drop a component package file (.tgz file created by running bots -node -sdk pack or npm pack) to upload.)

Selected file: oda.PizzaEEH.tgz
ready to be processed.

Enable Component Logging:

Components

Custom Webview

Services (1)

+ Add Service

Filter

oda.PizzaEEH

oda.PizzaEEH

Service Enabled:

Name: oda.PizzaEEH

Description: Optional short description for this service

Status: Ready

Platform Version: 2.0

Deployment Metadata

```
{
  "fnComponentVersion": "1",
  "botsSdkVersion": "2.0.1",
  "fnFunction": "b8cf0059-4eed-4140-8092-e731c437b9ce",
  "nodeVersion": "14.17.0",
  "fnApplication": "3ix7e3hm7kvvzapumh0fm7axftqjyamtlgsraqmf0jvvt03c"
}
```

Package File: oda.PizzaEEH-1.0.0.tgz

Configuration

Type: Composite Bag

Event Handler: oda.PizzaEEH

oda.PizzaEEH

oda.PizzaEEH

oda.PizzaEEH

Program agenda

- 1 Introduction
- 2 Entity event handler
- 3 Example - composite bag vs event handler
- 4 Creating an event handler
- 5 Deployment
- 6 **Under the hood**

Under the hood

Entity event handler component anatomy

```
module.exports = {  
  metadata: () => ({  
    name: 'oda.Pizza',  
    eventHandlerType: 'ResolveEntities'  
  })),  
  
  handlers: () => ({  
    'SomeEntity': { ← Name of composite bag entity  
  
    entity: { ... }, ← Entity level event handler  
    items: {  
      SomeBagItemName: { ... }, ← Bag item to create event handlers for  
      ...  
    },  
    custom: { ... } ← Custom events  
  })  
};
```

Under the hood

Event types

Item level events

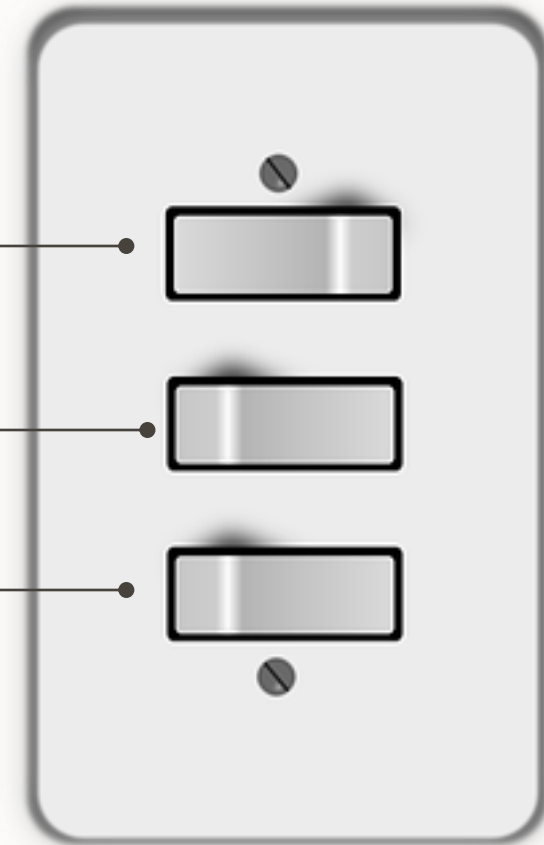
- Events execute for registered bag items
- Receive an **event** and **context** argument

Entity level events

- Called after item events
- Receive an **event** and **context** argument
- resolved event called at end of CBE processing

Custom events

- Buttons and list items may call custom events
- Receive an **event** and **context** argument
- Allow EEH developers to write generic code



Under the hood

EEH Events

Entity Level

- validate
- publishMessage
- maxPromptsReached
- resolved
- attachmentReceived
- locationReceived

Custom

- postback event with custom payload

Item Level

- shouldPrompt
- publishPromptMessage
- validate
- publishDisambiguateMessage
- maxPromptsReached

ORACLE