

ORACLE

# Entities Best Practices Using Platform Features

---

# Program Agenda

---

- 1 Best practices with composite bag entity
- 2 Best practices with other features

# Program Agenda

---

- 1 **Best practices with composite bag entity**
- 2 Best practices with other features

# Best practices with composite bag entity

What is a composite bag?

Composite bag entities represent domain objects

- person, order, product, booking, request etc.

Composed of bag items (attributes)

- Built-in entities
- Custom entities
- Dynamic entities
- String, location, attachment

All bag items get resolved by a single Visual Flow Designer component

- Resolve Composite Bag
- Resolve Entities

The screenshot displays a software interface for managing entities. On the left, a sidebar titled 'Entities' shows a list of entity types: ADDRESS, CURRENCY, DATE, DURATION, EMAIL, NUMBER, PERSON, and PHONE\_NUMBER. The 'ADDRESS' type is currently selected. A 'Create Entity' dialog box is open, showing the 'General Information' section with fields for 'Name' (containing 'cbe.Person') and 'Description'. The 'Configuration' section shows the 'Type' dropdown set to 'Composite Bag'. A 'Create' button is visible at the bottom right of the dialog. Below the dialog, a 'Bag Items' table is shown, listing attributes and their types for the selected entity.

Name	Type
FirstName	STRING
LastName	STRING
MiddleName	STRING
DateOfBirth	ENTITY

## Best practices with composite bag entity

---

### ” Always use composite bag entities

Composite bag is the most versatile and powerful entity available.

It simplifies the dialog design because it can resolve multiple entities together with less dialog states.

More easily allows entity slotting where there is validation and rule dependencies between entities

Wrap system entities to be able to set properties for them

# Best practices with composite bag entity

## Multiple entity extraction

- A pizza order intent may have size, type, address, date, time and even more associated entities
- Composite bags allow to extract all together
- Only needed a Resolve Composite Bag component in the dialog flow.

Bag Items			
+ Bag Item			
Name	Type	Entity Name	Sequence Number
pizzaSize	ENTITY	PizzaSize	1
pizzaTopping	ENTITY	PizzaTopping	2
deliveryTime	ENTITY	TIME	3
address	ENTITY	ADDRESS	4

I would like to order a Large Salami pizza, for 7PM delivered at 100 Amsterdam Street

Ok, let's get that order sorted.

Your large Salami pizza will be delivered at 19:00 at this address: 100 Amsterdam Street

The screenshot shows a chat interface with three messages. The first is a user message in a light bubble: "I would like to order a Large Salami pizza, for 7PM delivered at 100 Amsterdam Street". The second is a system message in a dark bubble: "Ok, let's get that order sorted.". The third is another system message in a dark bubble: "Your large Salami pizza will be delivered at 19:00 at this address: 100 Amsterdam Street". Each message has a small circular icon on the right side.

# Best practices with composite bag entity

## Validation

- We can validate the input against the entity type (email, URL, phone number etc.)
- Against a custom list of values, regular expression or a custom rule
- Validation error messages can be defined

What is the size of the pizza?

small  
medium  
large

XL

Sorry, 'XL' is not a valid size of pizza. Please choose from the below sizes.

small  
medium  
large

This screenshot shows a chat interface where a user has selected 'XL' for a pizza size. The system has responded with an error message: "Sorry, 'XL' is not a valid size of pizza. Please choose from the below sizes." The error message is displayed in a dark grey bubble, and the original selection of 'XL' is shown in a white bubble above it. The list of valid sizes (small, medium, large) is shown below the error message.

When can we deliver that for you (e.g., 4pm)?

11 PM

We can only deliver before 10PM!

This screenshot shows a chat interface where a user has selected '11 PM' for a delivery time. The system has responded with an error message: "We can only deliver before 10PM!". The error message is displayed in a dark grey bubble, and the original selection of '11 PM' is shown in a white bubble above it.

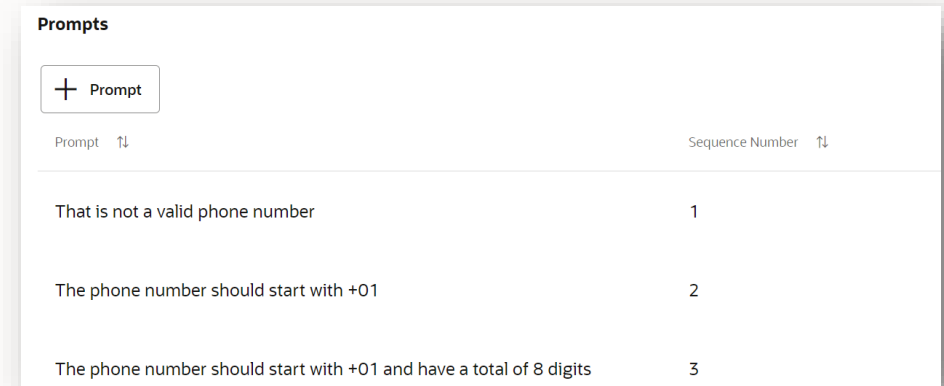
Validation Rules	
Expression	Error Message
<code>\${(pizza.value.deliveryTime.hrs?number &lt; 10)?then('true','false')}</code>	We can only deliver before 10PM!



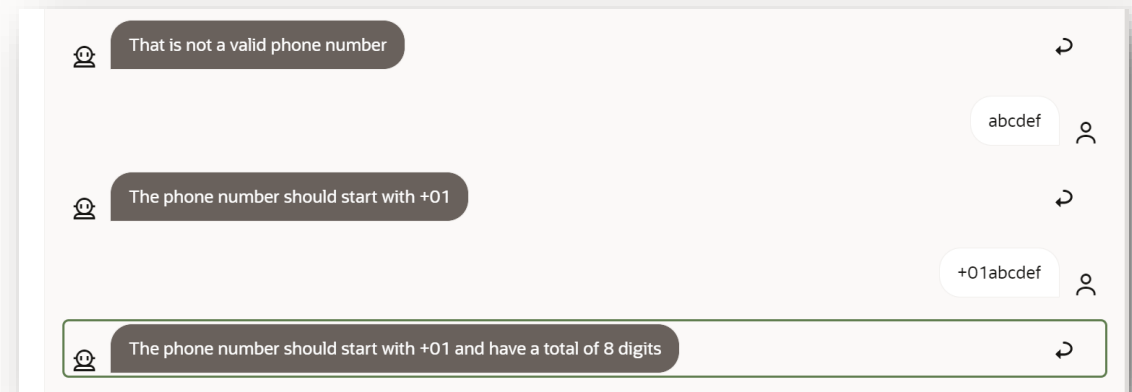
# Best practices with composite bag entity

## Error handling

- For failed validations, we can have prompt variation
- It is best practice to slowly disclose more information with each prompt
- Good designed prompts can help the user while maintaining a conversational tone



Prompt	Sequence Number
That is not a valid phone number	1
The phone number should start with +01	2
The phone number should start with +01 and have a total of 8 digits	3



The chat interface shows a sequence of prompts and user responses:

- Bot: "That is not a valid phone number" (with a sad face icon)
- User: "abcdef" (with a happy face icon)
- Bot: "The phone number should start with +01" (with a sad face icon)
- User: "+01abcdef" (with a happy face icon)
- Bot: "The phone number should start with +01 and have a total of 8 digits" (with a sad face icon)



# Best practices with composite bag entity

- Maximum prompts
  - We can define how many times we prompt the user for the input avoiding user lock
  - After which we should define a repair path with a transition to a different dialog state

Maximum User Input Attempts ?

3 ▼ ▲

The diagram illustrates a chatbot's response to invalid phone number inputs. It consists of three main stages of user prompts and corresponding user inputs:

- Stage 1:** The chatbot prompts "That is not a valid phone number". The user inputs "abcdef".
- Stage 2:** The chatbot prompts "The phone number should start with +01". The user inputs "+01abcdef".
- Stage 3:** The chatbot prompts "The phone number should start with +01 and have a total of 8 digits". The user inputs "+01abcdegh".

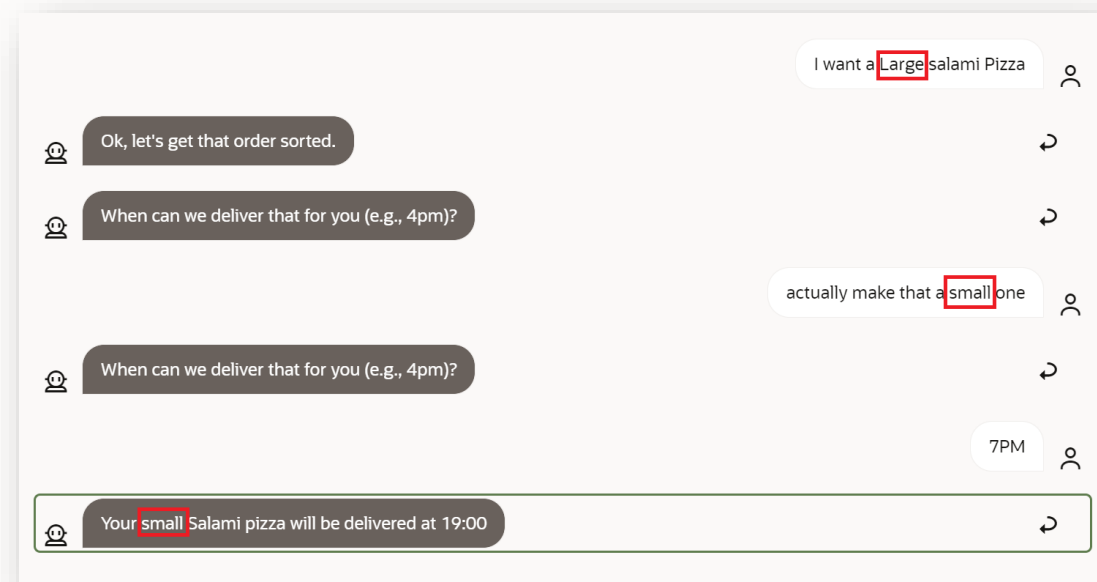
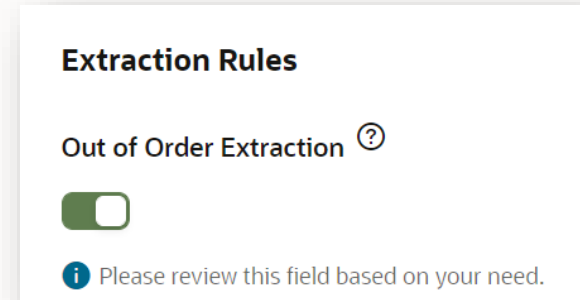
After the third stage, the chatbot provides a final prompt: "Let me connect you to an Agent!". Each prompt and input is shown in a dark grey bubble on the left and a light grey bubble on the right, with a circular arrow icon indicating the flow.



# Best practices with composite bag entity

## Out-of-order extraction

- Can be set at item level
- Values are slotted even when user is not prompted for it, or is prompted for a different entity value
- This allows us to solve a very common pattern in natural conversation



# Best practices with composite bag entity

## When to use entity event handlers

- Composite bags are powerful but may fall short when we need to:
  - Implement backend integration
    - Stock check during entity resolution (instead of doing that at the end)
  - Implement more complex business or validation logic
    - E.g. Very dependent on Apache FreeMarker expressions
  - Reference other composite bags
- Entity Event Handlers are the answer for this as they allow a programmatic approach to resolve the composite bag entity.

The screenshot shows the 'Entities (17)' management interface. On the left, a list of entities includes 'cbePhone', 'cbePizza' (highlighted), 'PizzaSize', 'PizzaTopping', 'ADDRESS', 'CURRENCY', 'DATE', 'DURATION', 'EMAIL', 'LOCATION', 'NUMBER', and 'PERSON'. The 'cbePizza' entity is selected, and its configuration is shown on the right. The 'General Information' section includes 'Name' (cbePizza) and 'Description'. The 'Configuration' section shows 'Type' as 'Composite Bag'. The 'Event Handler' field is highlighted with a red box and contains a dropdown menu and a '+ Event Handler' button. Below this, the 'Bag Items' section shows a table with columns for Name, Type, Entity Name, and Sequence Number.

Name	Type	Entity Name	Sequence Number
pizzaSize	ENTITY	PizzaSize	1
pizzaTopping	ENTITY	PizzaTopping	2
deliveryTime	ENTITY	TIME	3

# Best practices with composite bag entity

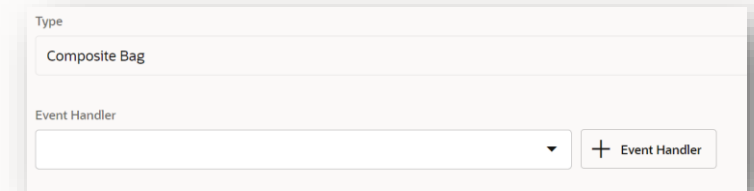
## Entity event handlers

### Complex validation

- With event handlers we no longer are limited to the Apache free marker expressions
- It also gives us the ability to cross reference other bag items

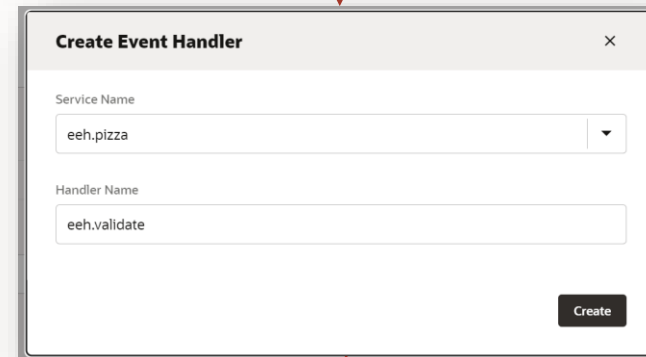
### Backend Integration

- We can do backend checks during resolution, instead of having to do it in the next dialog step



Type  
Composite Bag

Event Handler  
[Dropdown] + Event Handler



Create Event Handler

Service Name  
eeh.pizza

Handler Name  
eeh.validate

Create



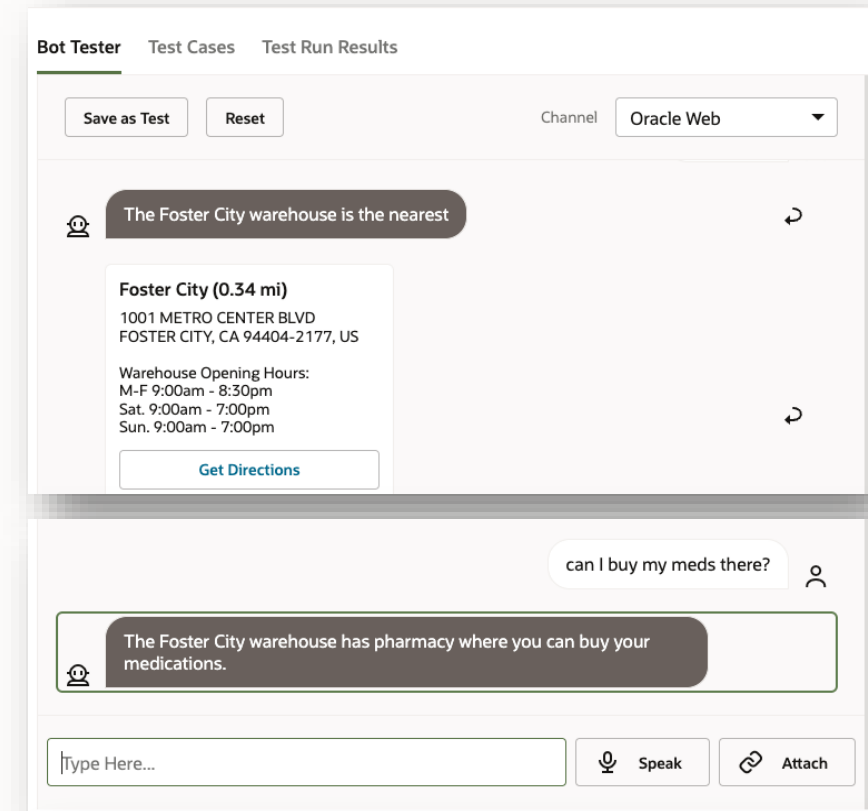
```
1 'use strict';
2
3 // node fetch API can be used to make REST calls, see https://www.npmjs.com/package/node-fetch
4 const fetch = require("node-fetch");
5
6 module.exports = {
7   metadata: {
8     name: 'eeh.validate',
9     eventHandlerType: 'ResolveEntities',
10    supportedActions: [] // string 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 };
```

# Best practices with composite bag entity

## Entity event handlers

### Context awareness

- With entity event handlers we have the flexibility to handle more complex scenarios
- We can store context and use it to set following actions



# Program Agenda

---

- 1 Best practices with composite bag entity
- 2 **Best practices with other features**

# Best practices with other features

## Date properties

- Date ambiguity
  - *“I want to book a flight for Monday”*
  - *“I want to create an expense for a meal I had Monday”*

We can easily define the Ambiguity Resolution Rule for Date Entity as *Past*, *Future* or *Nearest*.

- Date locale is another very typical parameter that we can set
  - DD/MM/YY or MM/DD/YY

**Ambiguity Resolution Rule**

Consider End User Locale <sup>?</sup>

Default Date Format

DD/MM/YY

Resolve Date as

Tense

Past

in intents

**Utterance Tester**

Quick Test [Go to Test Cases](#) [Run Test for All](#)

Language

Auto

Utterance

I want to create expenses from Monday

**Results**

[View JSON](#)

Confidence Threshold

Utterance

I want to create expenses from Monday

Detected Entities

[Label View](#) [List View](#)

I want to create expenses from **Monday**

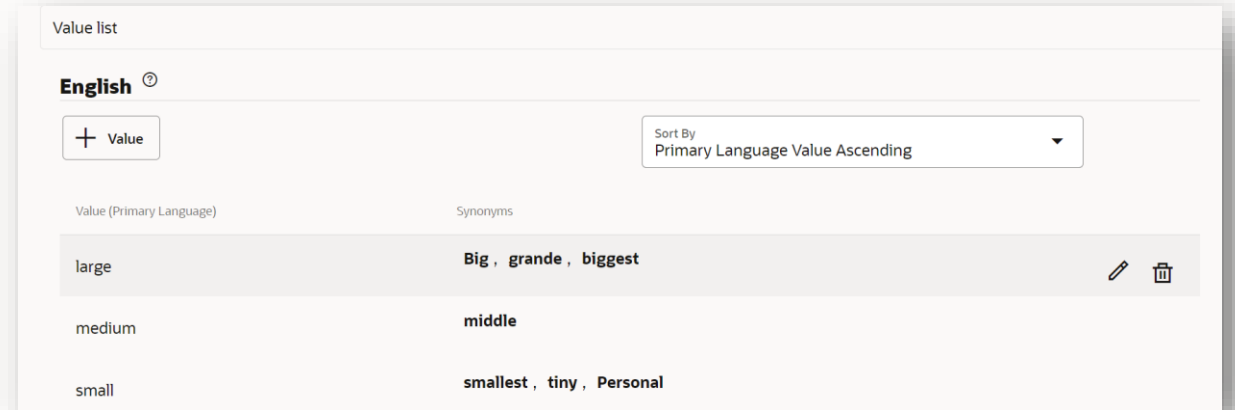
DATE

Entity	Value
DATE	Mon Apr 04 2022 00:00:00 GM...

# Best practices with other features

## Value list properties

- Always use synonyms with value lists
- It increases accuracy
- Can be used to build entity based robust action menus
  - Entity validation
  - Prompts

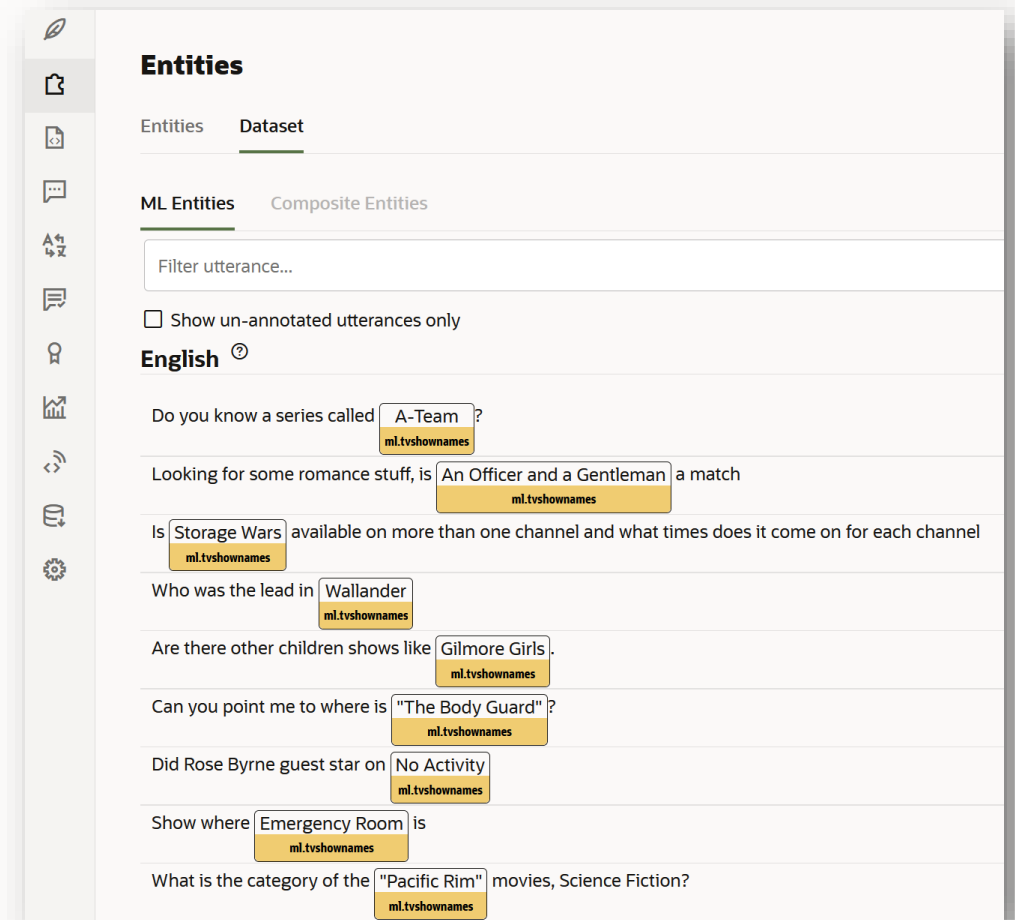




# Best practices with other features

## Machine learning entity

- ML entities extract known and unknown values from an infinite list of options
- Use this entity when you have a use case for which you cannot know all the possible values



The screenshot displays the 'Entities' interface in the Microsoft Bot Framework console. It shows a list of utterances with ML entities extracted and labeled. The interface includes a sidebar with navigation icons, a top navigation bar with 'Entities' and 'Dataset' tabs, and a main content area with a search filter and a list of utterances. The utterances are in English, and the entities are labeled with 'ml.tvshownames'.

**Entities**

Entities Dataset

ML Entities Composite Entities

Filter utterance...

Show un-annotated utterances only

**English** ⓘ

Do you know a series called **A-Team** ?  
ml.tvshownames

Looking for some romance stuff, is **An Officer and a Gentleman** a match  
ml.tvshownames

Is **Storage Wars** available on more than one channel and what times does it come on for each channel  
ml.tvshownames

Who was the lead in **Wallander**  
ml.tvshownames

Are there other children shows like **Gilmore Girls** .  
ml.tvshownames

Can you point me to where is **"The Body Guard"** ?  
ml.tvshownames

Did Rose Byrne guest star on **No Activity**  
ml.tvshownames

Show where **Emergency Room** is  
ml.tvshownames

What is the category of the **"Pacific Rim"** movies, Science Fiction?  
ml.tvshownames

ORACLE