

ORACLE®

Oracle Digital Assistant

The Complete Training

Composite Bag

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 ➤ Entities and why we need them
- 2 ➤ Composite bag basics
- 3 ➤ Composite bag error handling
- 4 ➤ Working with entity values
- 5 ➤ Slotting entities out of order

Topic agenda

- 1 ➤ Entities and why we need them
- 2 ➤ Composite bag basics
- 3 ➤ Composite bag error handling
- 4 ➤ Working with entity values
- 5 ➤ Slotting entities out of order

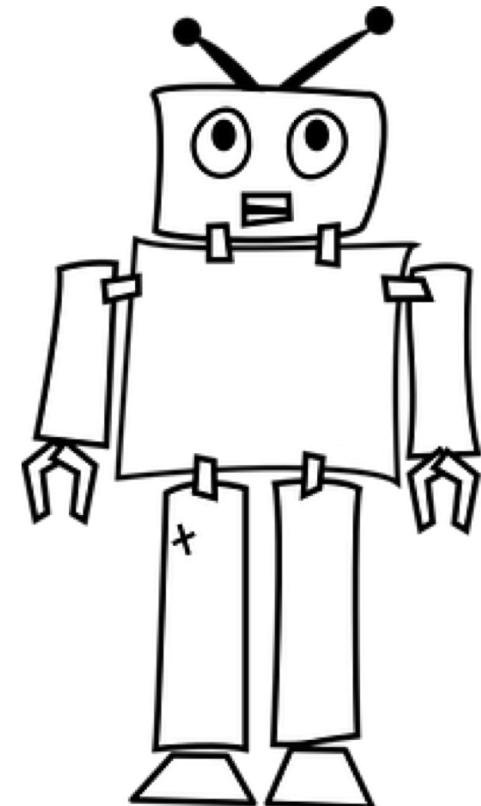
Entities and why we need them – a recap

- Important variable elements related to an intent
 - Date, bank account, amount to transfer, pizza size, pizza topping
- Entity slotting
 - Process of filling those variable elements

```
startBalances:  
  component: "System.SetVariable"  
  properties:  
    variable: "accountType"  
    value: "${iResult.value.entityMatches['AccountType'][0]}"  
  transitions: {}  
  
askBalancesAccountType:  
  component: "System.List"  
  properties:  
    options: "${accountType.type.enumValues}"  
    prompt: "For which account do you want your balance?"  
    variable: "accountType"  
  transitions: {}
```

```
askBalancesAccountType:  
  component: "System.List"  
  properties:  
    options: "${accountType.type.enumValues}"  
    nlpResultVariable: "iResult"  
    prompt: "For which account do you want your balance?"  
    variable: "accountType"  
  transitions: {}
```

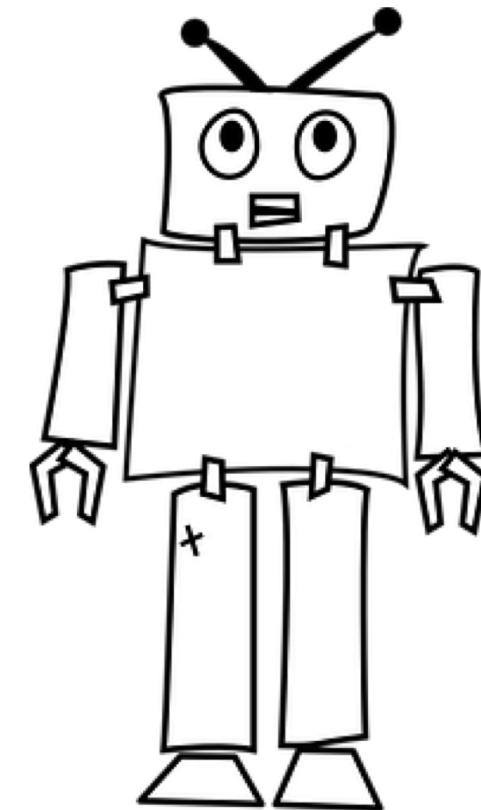
Can anyone see the **problem** with
this approach?



```

resolvesize:
  component: "System.SetVariable"
  properties:
    variable: "size"
    value: "${iResult.value.entityMatches['PizzaSize'][0]}"
  transitions: {}
resolvecrust:
  component: "System.SetVariable"
  properties:
    variable: "crust"
    value: "${iResult.value.entityMatches['PizzaCrust'][0]}"
  transitions: {}
resolvetype:
  component: "System.SetVariable"
  properties:
    variable: "type"
    value: "${iResult.value.entityMatches['PizzaType'][0]}"
  transitions: {}
askage:
  component: "System.Output"
  properties:
    text: "How old are you?"
  transitions: {}
checkage:
  component: "AgeChecker"
  properties:
    minAge: 18
  transitions:
    actions:
      allow: "crust"
      block: "underage"
crust:
  component: "System.List"
  properties:
    options: "Thick,Thin,Stuffed,Pan"
    prompt: "What crust do you want for your Pizza?"
    variable: "crust"
  transitions: {}
size:
  component: "System.List"
  properties:
    options: "${size.type.enumValues}"
    prompt: "What size Pizza do you want?"
    variable: "size"
  transitions: {}
type:
  component: "System.Text"
  properties:
    prompt: "What Type of Pizza do you want?"
    variable: "type"
  transitions: {}

```



The challenge of “real world” entity slotting

- Many values required for an intent
- Error handling for each entity value
- Different prompts should the user error
- Out of order information
- Validation
- Allow multiple values or not?
- Slot entity if specifically said, otherwise, default it

Topic agenda

- 1 ➤ Entities and why we need them
- 2 ➤ Composite bag basics
- 3 ➤ Composite bag error handling
- 4 ➤ Working with entity values
- 5 ➤ Slotting entities out of order

Composite bag entity

- Models a business domain object
 - Pizza order, holiday request, expense
- Each composite bag is composed of one to many items
 - Custom entities
 - Built-in entities
 - String, location and attachment
- All contained entities get resolved automatically in a single dialog flow state
 - System.ResolveEntities
 - Automatically displays enumerated values as a list and provides pagination
 - System.CommonResponse

Ordering a pizza with composite bag entity

variables:

order: "PizzaOrder" ←
iResult: "nlpresult"

Configure

states:

intent:

component: "System.Intent" ← "A cheese pizza please."

properties:

variable: "iResult"

orderState:

component: "System.ResolveEntities"

properties:

variable: "order"

nlpResultVariable: "iResult"

[...]

Check for Resolved Entities

What size of pizza do
you want?

And what kind of
crust?

Composite Bag:
PizzaOrder

Entity: PizzaType

Entity: PizzaSize

Entity: PizzaCrust

PizzaType

PizzaSize

PizzaCrust

Skills • GR_Pizza_Composite_Bag DRAFT • 1.0 ▾[Instant Apps](#)[Validate](#)[Train](#)

[+ Entity](#) [More ▾](#)

Filter [🔍](#)

Sort By [Created Ascending](#)

- [PizzaBag](#) [X](#)
- [PizzaDough](#) [X](#)
- [PizzaSize](#) [X](#)
- [PizzaTopping](#) [X](#)
- [ADDRESS](#) [X](#)
- [CURRENCY](#) [X](#)
- [DATE](#) [X](#)
- [DURATION](#) [X](#)
- [EMAIL](#) [X](#)
- [NUMBER](#) [X](#)
- [PHONE_NUMBER](#) [X](#)

Description

Name *
PizzaBag

Description

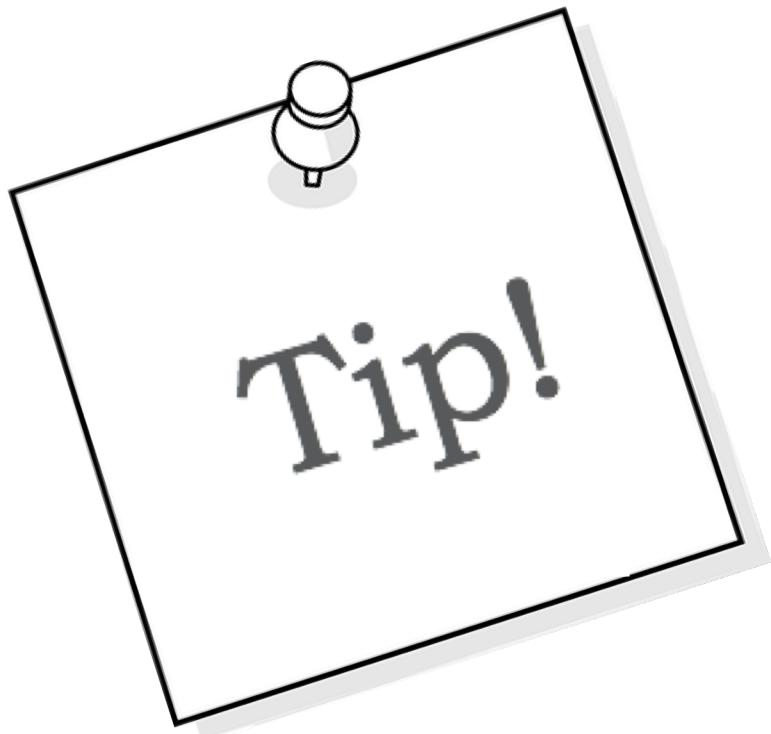
Configuration

Type [?](#)
Composite Bag

Bag Items

[+ Bag Item](#)

Name	Type	Entity Name
PizzaSize	ENTITY	PizzaSize
PizzaTopping	ENTITY	PizzaTopping
DeliveryTime	ENTITY	TIME



Remember to train if
you add an item to
composite bag

```
1 #metadata: information about the flow
2 # platformVersion: the version of the bots platform that this flow was written to work with
3 metadata:
4   platformVersion: 1.0
5 main: true
6 name: GR_Pizza_Composite_Bag
7 #context: Define the variables which will be used throughout the dialog flow here.
8 context:
9   variables:
10     iResult: "nlpresult"
11     pizza: "PizzaBag"
12
13 states:
14
15 intent:
16   component: "System.Intent"
17   properties:
18     variable: "iResult"
19     optionsPrompt: "Do you want to"
20   transitions:
21     actions:
22       OrderPizza: "startOrderPizza"
23       WelcomePizza: "startWelcome"
24       unresolvedIntent: "startUnresolved"
25
26 resolveEntities:
27   component: "System.ResolveEntities"
28   properties:
29     variable: "pizza"
30     nlpResultVariable: "iResult"
31     maxPrompts: 3
32     cancelPolicy: "immediate"
33     entityOrder:
34   transitions:
35     actions:
36       cancel: "maxError"
37       next: "showPizzaOrder"
```

Ordering a pizza with composite bag entity

ok lets get that order sorted

What kind of pizza would you like

- Meaty
- Veggie
- Hot and Spicy
- American Hot

I want a large pizza

Meaty

When would you like us to deliver your pizza?

anytime you want

Ok, we need a valid time for delivery. When would you like us to deliver your pizza?

now?

Description

Name *
PizzaBag

Description

Configuration

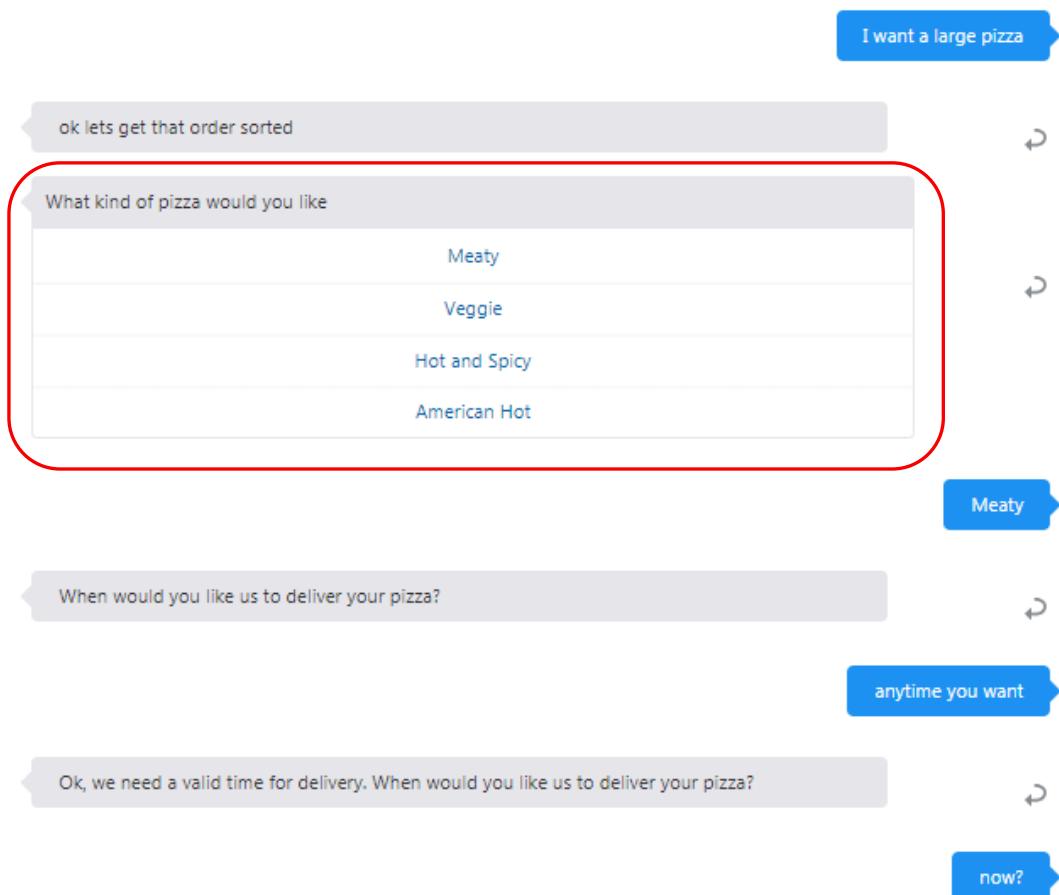
Type ?
Composite Bag

Bag Items

+ Bag Item

Name	Type	Entity Name
PizzaSize	ENTITY	PizzaSize
PizzaTopping	ENTITY	PizzaTopping
DeliveryTime	ENTITY	TIME

Composite bag prompts



?

Multiple Values

Fuzzy Match

Disambiguation Resolution

?

Prompt for Disambiguation

?

Disambiguation Prompt

Sorry you can only select one pizza size

Entity Extraction

?

Out of Order Extraction

?

Extract With

PizzaSize

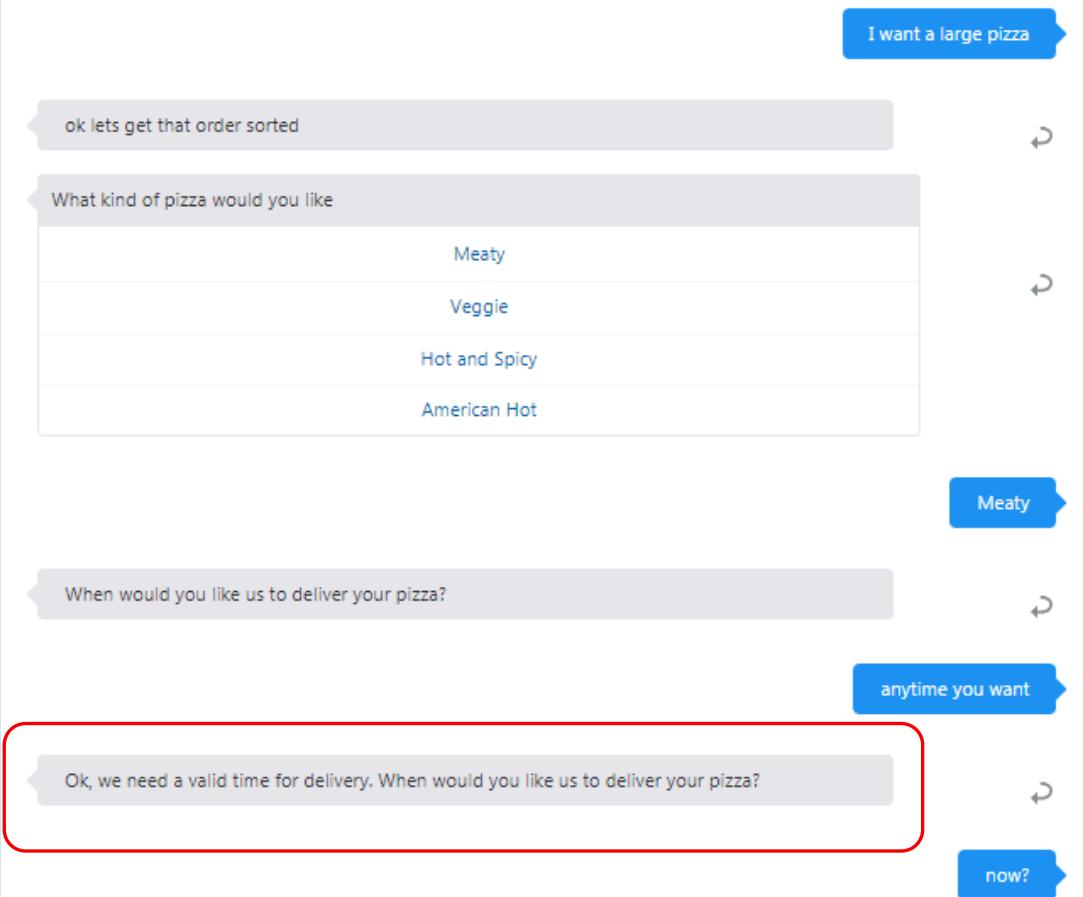
?

Prompt for Value

Prompts

Prompt	Sequence Number
What kind of pizza would you like	1
Which one of our pizzas would you like to try?	2

Composite bag prompts

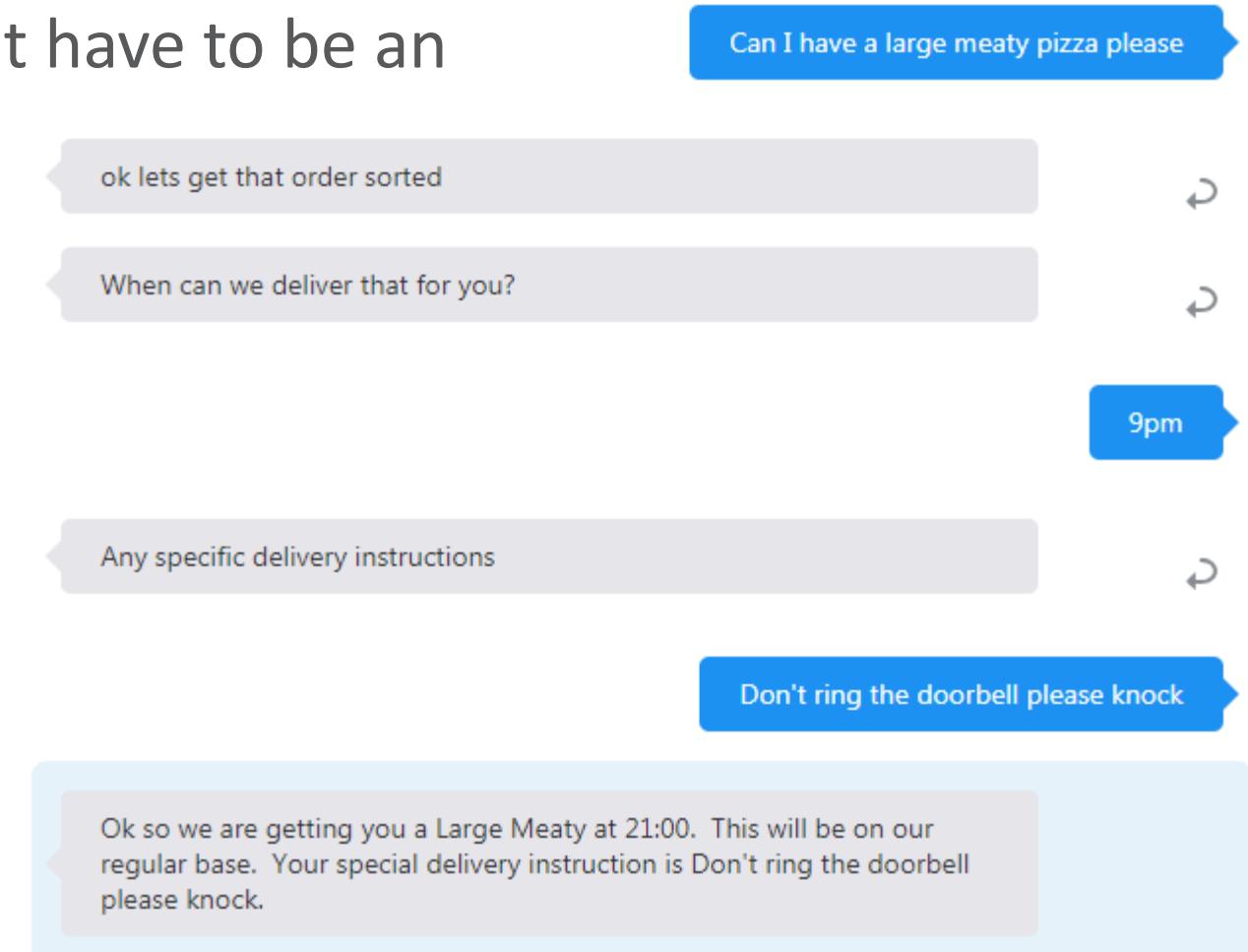


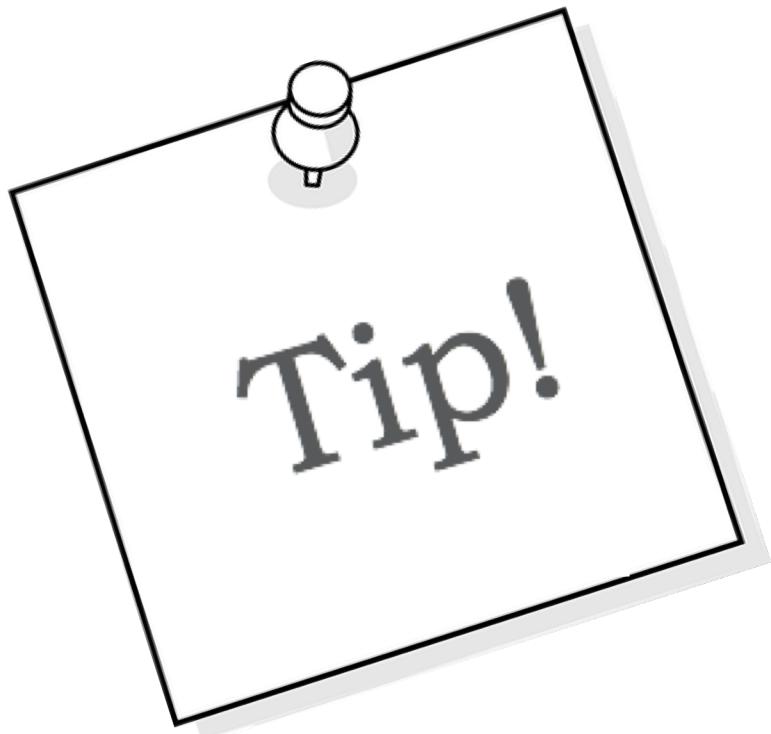
The screenshot shows the configuration of a composite bag prompt:

- Error Message:** "Ok, we need a valid time for delivery." (highlighted with a red box)
- Multiple Values:** Off
- Fuzzy Match:** Off
- Disambiguation Resolution:**
 - Prompt for Disambiguation: On
 - Disambiguation Prompt: (empty input field)
- Entity Extraction:**
 - Out of Order Extraction: On
 - Extract With: PizzaSize
 - Prompt for Value: (empty input field)
- Prompts:**
 - + Prompt: "When would you like us to deliver your pizza?" (highlighted with a red box)
 - Sequence Number: 1

Composite bag string

- A composite bag entity item doesn't have to be an entity
 - String, location, attachment





Entities are resolved in the **order** in which they appear in the composite bag – you can change the order at design time

resolveEntities resolves composite bag in dialog flow

```
resolveEntities:  
    component: "System.ResolveEntities"  
    properties:  
        variable: "pizza"  
        nlpResultVariable: "iResult"  
        maxPrompts: 3  
        cancelPolicy: "immediate"  
        headerText: "This message appears for each entity"  
    transitions:  
        actions:  
            cancel: "maxError"  
            next: "setPizzaDough"
```

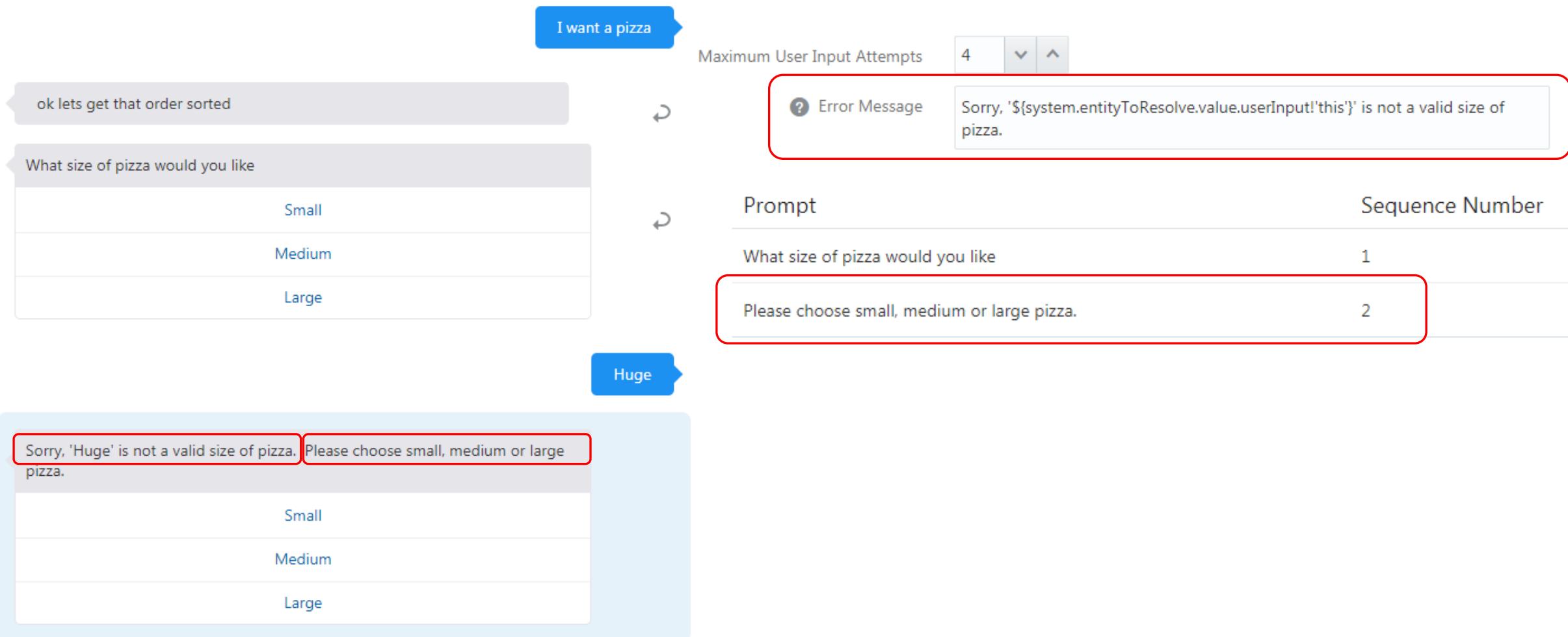
Topic agenda

- 1 ➤ Entities and why we need them
- 2 ➤ Composite bag basics
- 3 ➤ Composite bag error handling
- 4 ➤ Working with entity values
- 5 ➤ Slotting entities out of order

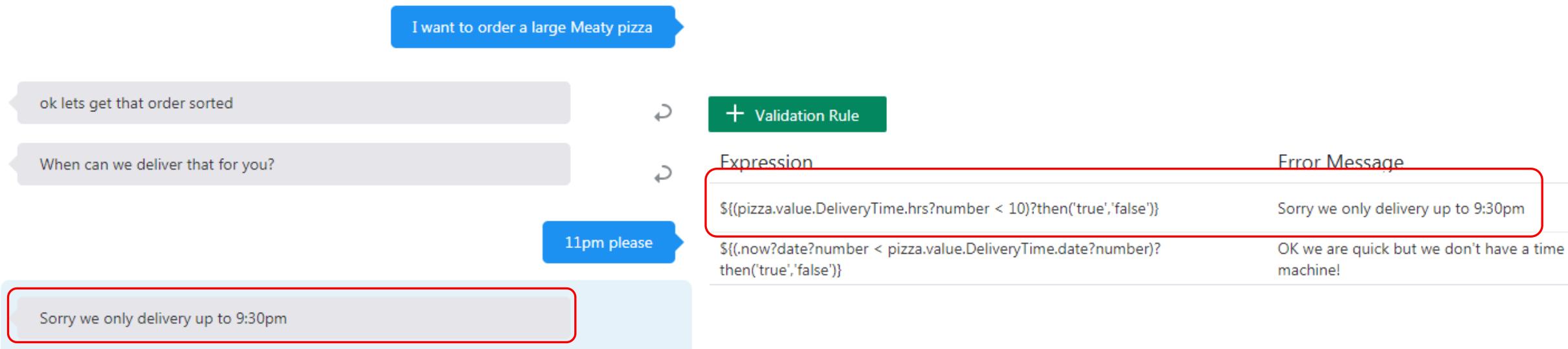
Composite bag validation and error handling

- You can define error messages for invalid input
 - This can also include ApacheFreemarker expressions
- You can define validation rules to enforce business rules
 - Also can include ApacheFreemarker expressions
- You can define maximum attempts for valid input
 - Error retries in composite bag overrides maxPrompts in dialog flow

Composite bag validation and error handling



Composite bag validation and error handling



Composite bag validation and error handling

- Define the maximum number of retries in dialog flow
- Override within each entity
- Define if failure is
 - Immediate
 - On last entity only (backwards compatibility)

```
resolveEntities:  
  component: "System.ResolveEntities"  
  properties:  
    variable: "pizza"  
    nlpResultVariable: "iResult"  
    maxPrompts: 3  
    cancelPolicy: "immediate"  
  transitions:  
    actions:  
      cancel: "maxError"  
      next: "showPizzaOrder"
```

The screenshot shows the configuration for a PizzaSize entity in the Oracle Dialog Flow interface. The entity is defined with the following settings:

- Name:** PizzaSize
- Type:** Entity
- Entity Name:** PizzaSize
- Description:** (Empty field)
- Enumeration Range Size:** (Empty field)
- Maximum User Input Attempts:** Set to 4.

Topic agenda

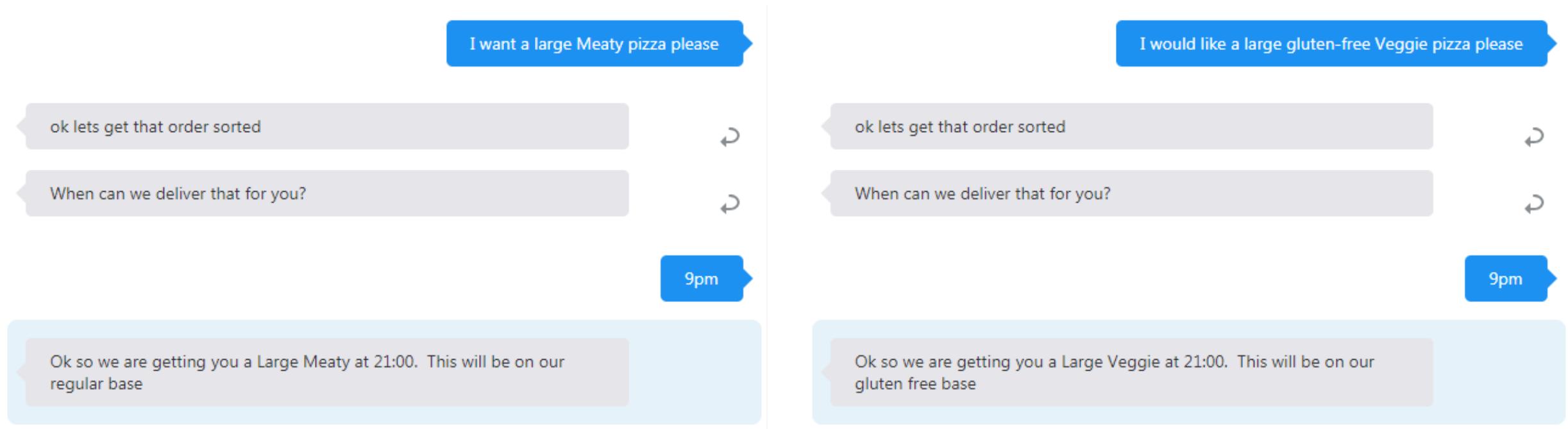
- 1 ➤ Entities and why we need them
- 2 ➤ Composite bag basics
- 3 ➤ Composite bag error handling
- 4 ➤ Working with entity values
- 5 ➤ Slotting entities out of order

Defaulting entity values

- Composite bag will slot any entity values in initial sentence
- Then will prompt for other entity values
- What if you want to capture a entity value if mentioned, but not specifically prompt for it
 - Pizza dough
 - Assume regular unless someone specifically asks for gluten-free



Defaulting entity values



Defaulting entity values

- Set prompt to false, then populate default in the dialog flow after resovleEntities

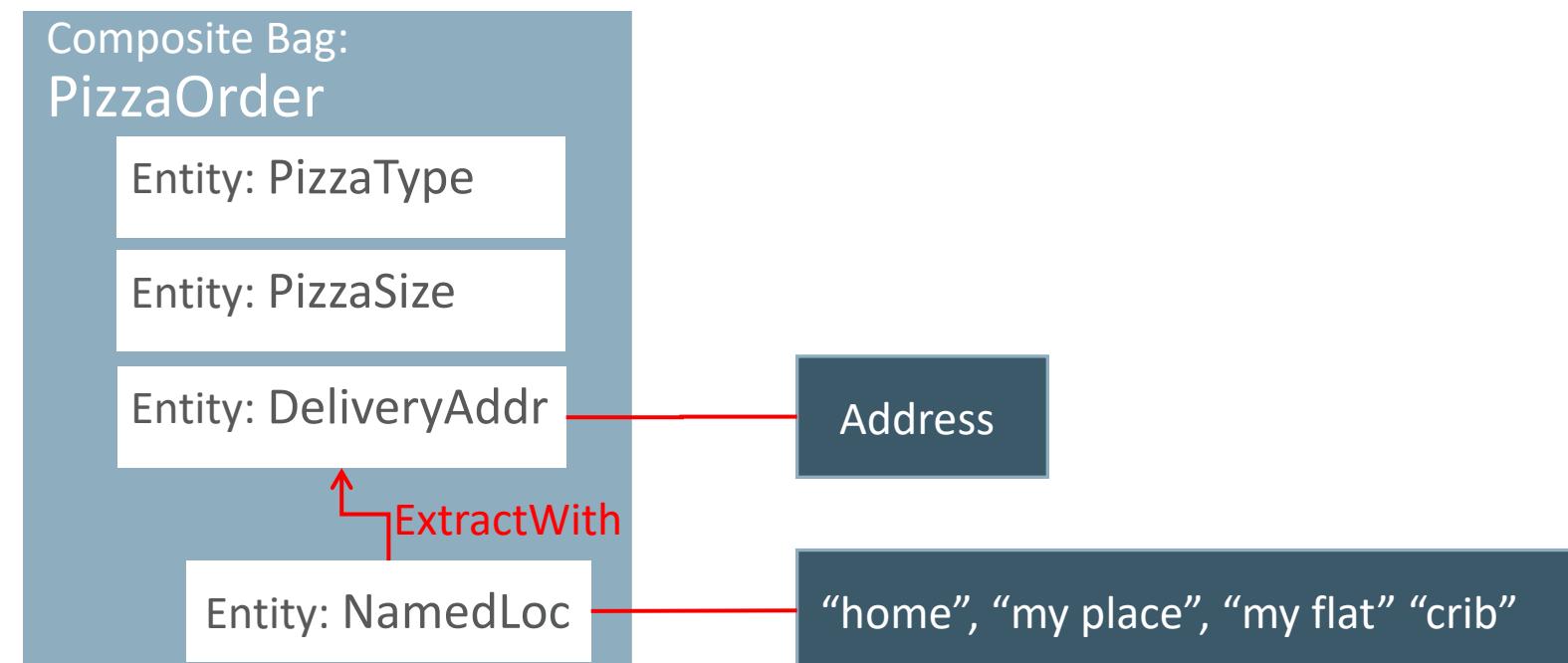
Entity Extraction



```
setPizzaDough:  
  component: "System.SetVariable"  
  properties:  
    variable: "pizza.PizzaDough"  
    # value set for the variable.  
    value: "${pizza.value.PizzaDough?has_content?then(pizza.value.PizzaDough,'regular')}"
```

Allowing related terms for an entity value

- User may answer in a way which is different from the expected entity value
- DeliveryAddress (primary) NamedLocation (secondary)
 - “What is the delivery address for your pizza” – “home delivery please”



Allowing related terms for an entity value

- Create NamedLocation entity add to bag
 - Don't specifically prompt for secondary entity (Prompt for Value false)
 - Extract with Delivery Address
 - Only prompt for primary if secondary has no content

The image displays two separate configuration screens for entity types.

DeliveryAddress Entity Configuration:

- * Name: DeliveryAddress
- Type: Entity
- Entity Name: ADDRESS
- Prompt for Value: \${!pizza.value.NamedLocation?has_content}

NamedLocation Entity Configuration:

- * Name: NamedLocation
- Type: Entity
- Entity Name: NamedLocation
- Extract With: DeliveryAddress
- Prompt for Value: false

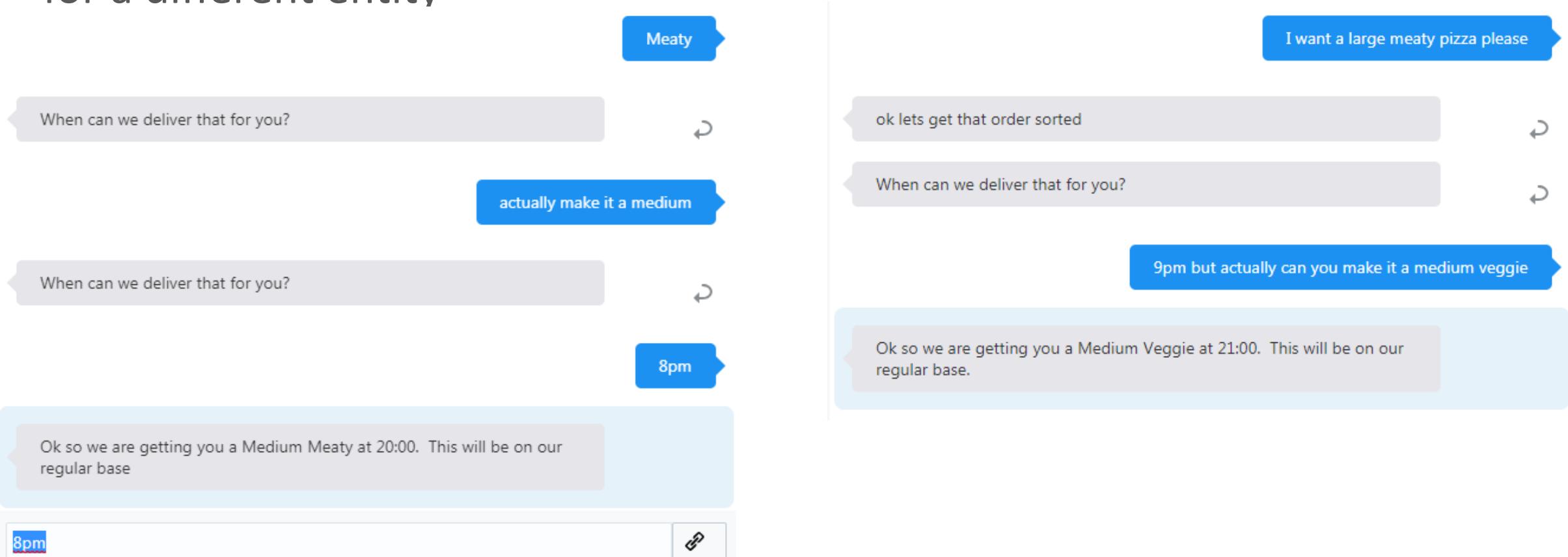
Both sections have red boxes highlighting the "Prompt for Value" fields.

Topic agenda

- 1 ➤ Entities and why we need them
- 2 ➤ Composite bag basics
- 3 ➤ Composite bag error handling
- 4 ➤ Working with entity values
- 5 ➤ Slotting entities out of order

Slotting entities out of order

- Sometimes the user might supply a new entity value whilst awaiting a value for a different entity



Slotting entities out of order

- If Out of Order Extraction is set, composite bag will resolve if it finds any entities of that type in any user input within resolveEntities
 - Would not work for string (as every input could be a string)

The screenshot illustrates a conversational interface for ordering a pizza. On the left, there's a configuration panel titled "Entity Extraction" with three settings:

- "Out of Order Extraction" is turned on (indicated by a blue toggle switch).
- "Extract With" is set to "PizzaTopping".
- "Prompt for Value" is present but empty.

On the right, a conversation log shows the following messages:

- User message: "I want a large meaty pizza please"
- Bot response: "ok lets get that order sorted"
- User message: "When can we deliver that for you?"
- Bot response: "9pm but actually can you make it a medium veggie"
- Bot response: "Ok so we are getting you a Medium Veggie at 21:00. This will be on our regular base."

The word "medium" in the user message and the word "veggie" in the bot response are highlighted with red boxes, demonstrating how the system identifies and extracts entities even if they appear out of order or as part of a larger phrase.

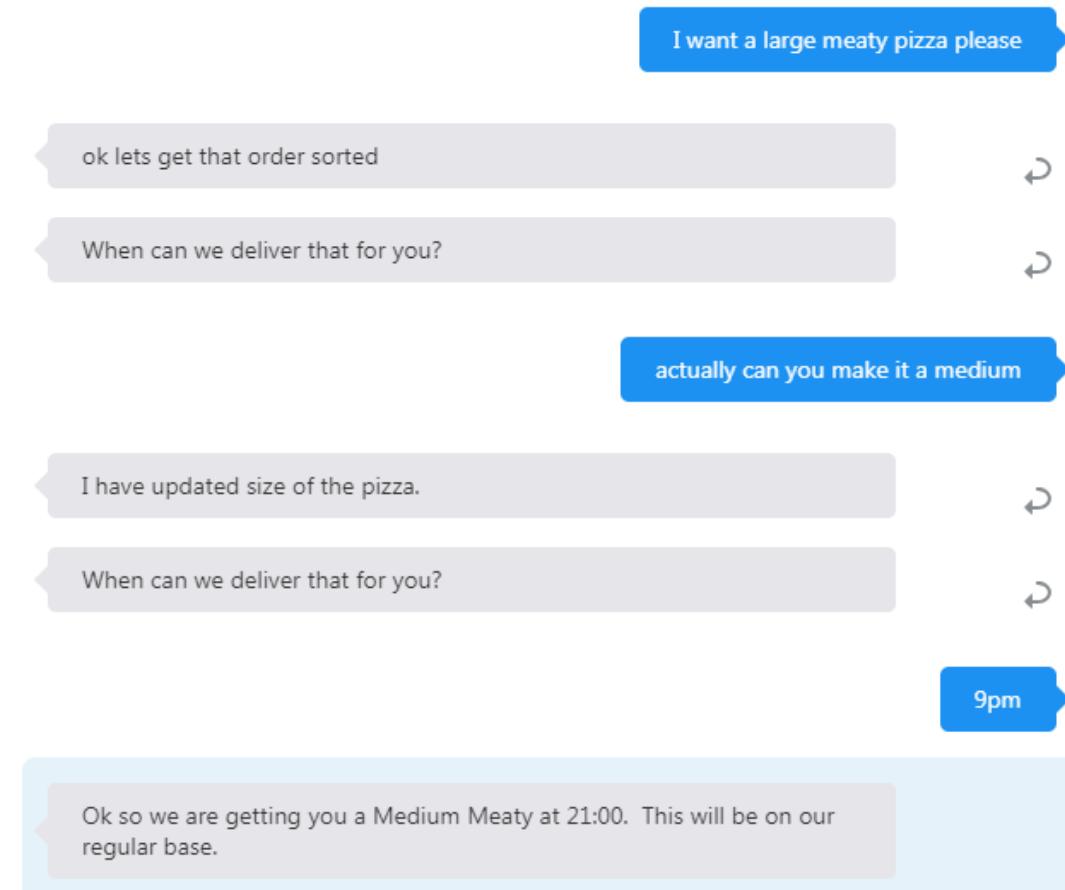
Slotting entities out of order

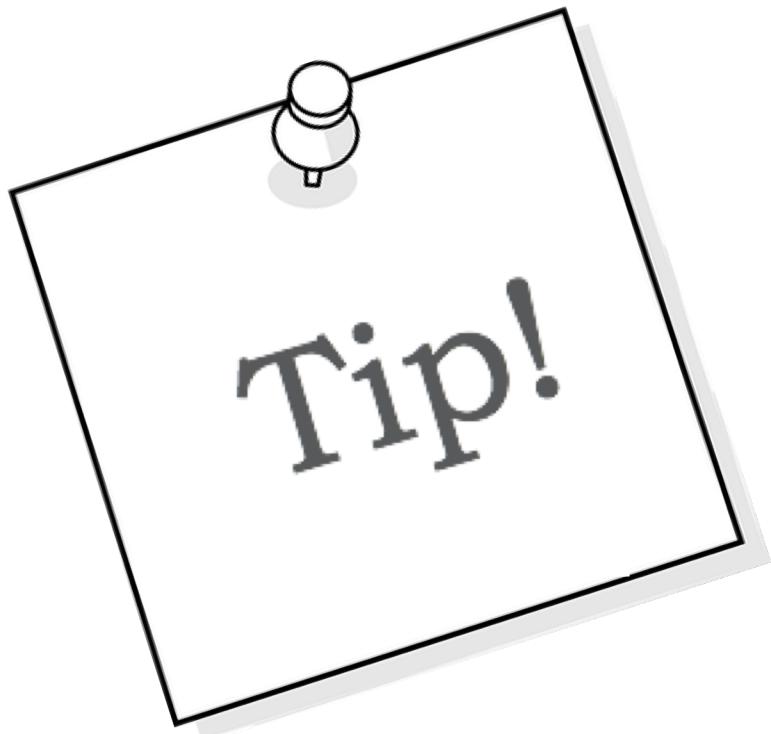
- Confirming to the user that the entity has changed

```
resolveEntities:  
    component: "System.ResolveEntities"  
    properties:  
        variable: "pizza"  
        nlpResultVariable: "iResult"  
        maxPrompts: 3  
        cancelPolicy: "immediate"  
        headerText: "<#list system.entityToResolve.value.updatedEntities>I have updated <#items as ent>${ent.description}<#sep> and </#items>."  
    transitions:  
        actions:  
            cancel: "maxError"  
            next: "setPizzaDough"
```

Slotting entities out of order

```
headerText: "<#list system.entityToResolve.value.updatedEntities>I have updated <#items as ent>${ent.description}<#sep> and </#items>. </#list>"
```





Going forward,
composite bag is your
primary “go to” for
entity resolution



Oracle Digital Assistant Hands-On

TBD

Integrated Cloud Applications & Platform Services

ORACLE®