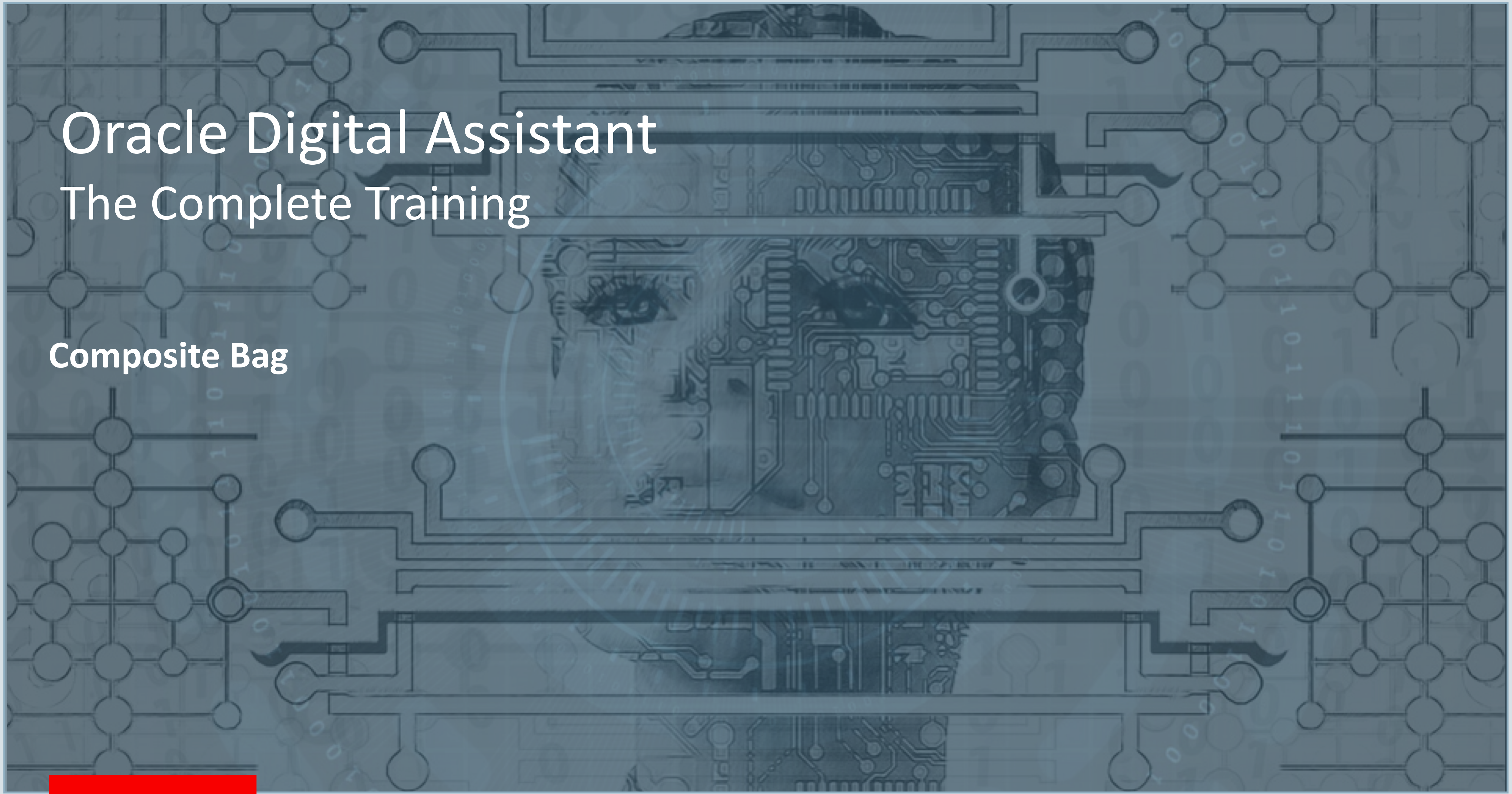
ORACLE®



# Oracle Digital Assistant

## The Complete Training

**Composite Bag**

ORACLE

## Safe Harbor Statement

## 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

## 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?

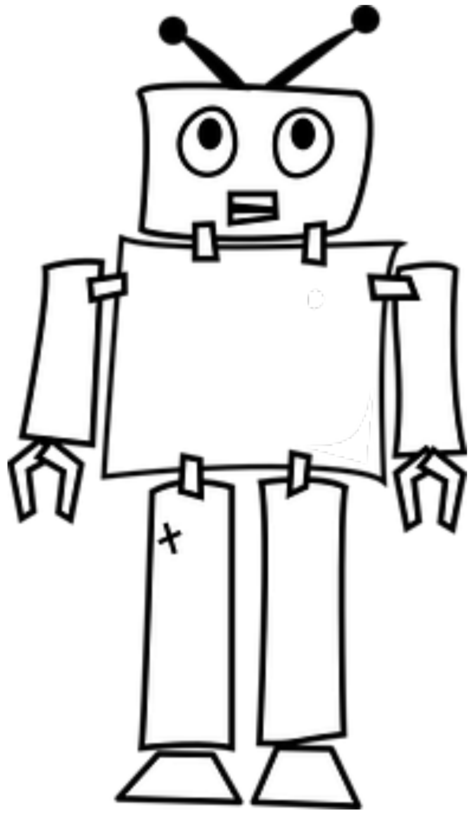
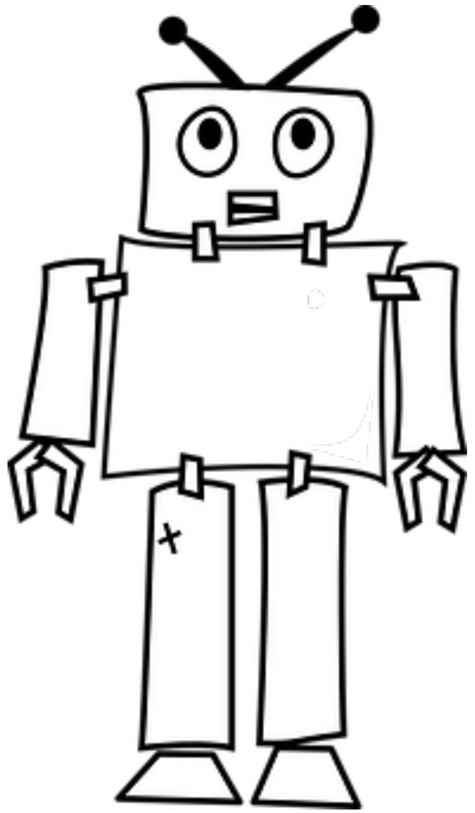


Image courtesy of pixabay.com

```
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

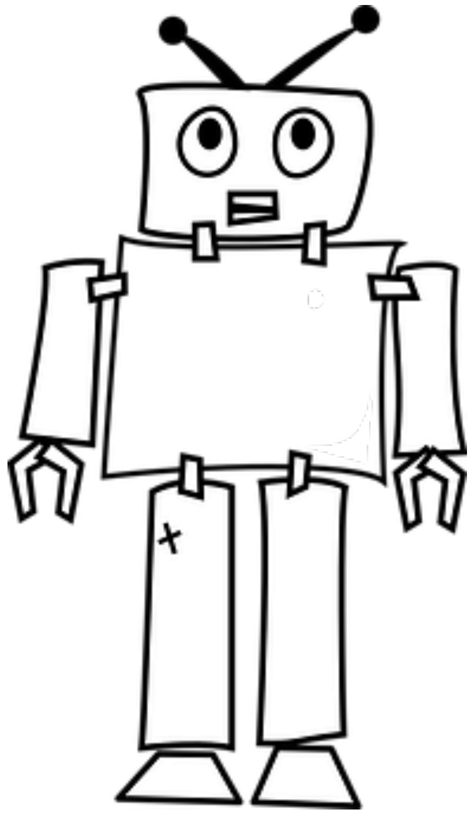
- Many values required for an intent
- Error handling for each entity value
- Different prompts should the user error
- Out of order information
- Validation
- Allow multiple values or not?
- Slot entity if specifically said, otherwise, default it

## Topic agenda

## 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 <u>automatically</u> in a single dialog flow state
  - System.ResolveEntities
    - Automatically displays enumerated values as a list and provides pagination
  - System.CommonResponse

**Dynamic entities too can be used** as the type of a composite bag item

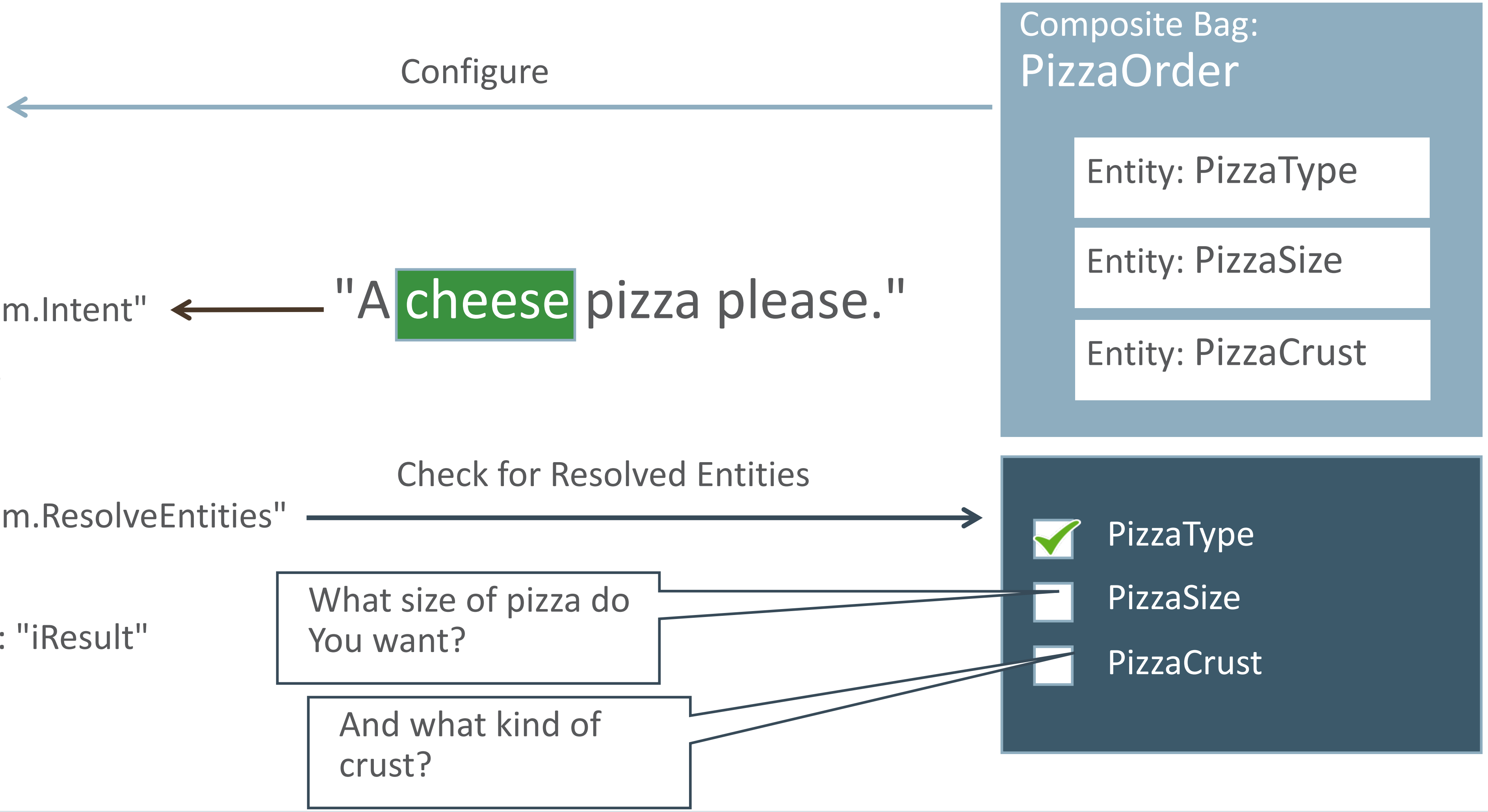


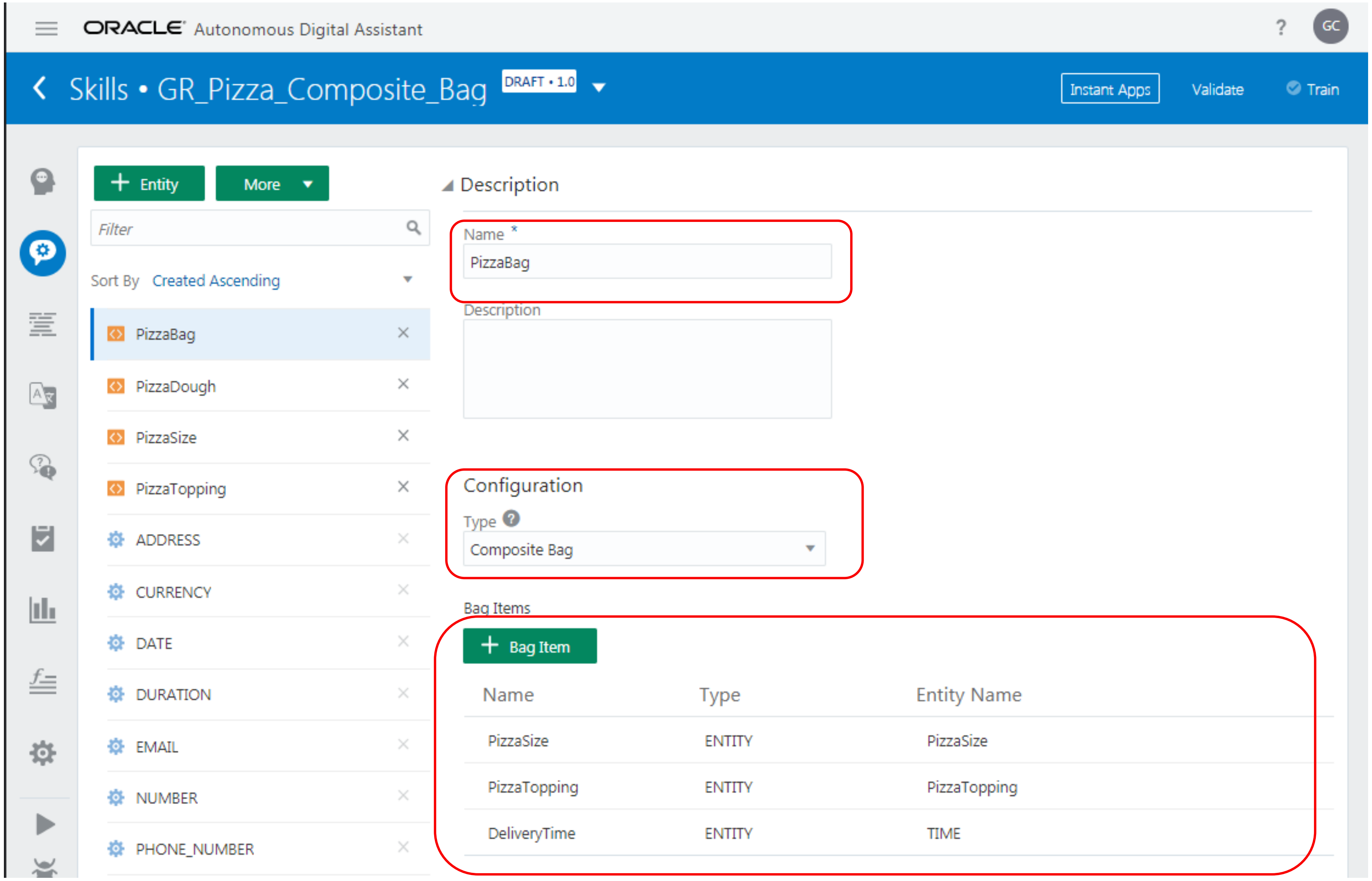
# Ordering a pizza with composite bag entity

```
variables:
  order: "PizzaOrder"
  iResult: "nlpresult"

states:
  intent:
    component: "System.Intent"
    properties:
      variable: "iResult"

  orderState:
    component: "System.ResolveEntities"
    properties:
      variable: "order"
      nlpResultVariable: "iResult"
      [...]
```





Description

Name *
PizzaBag

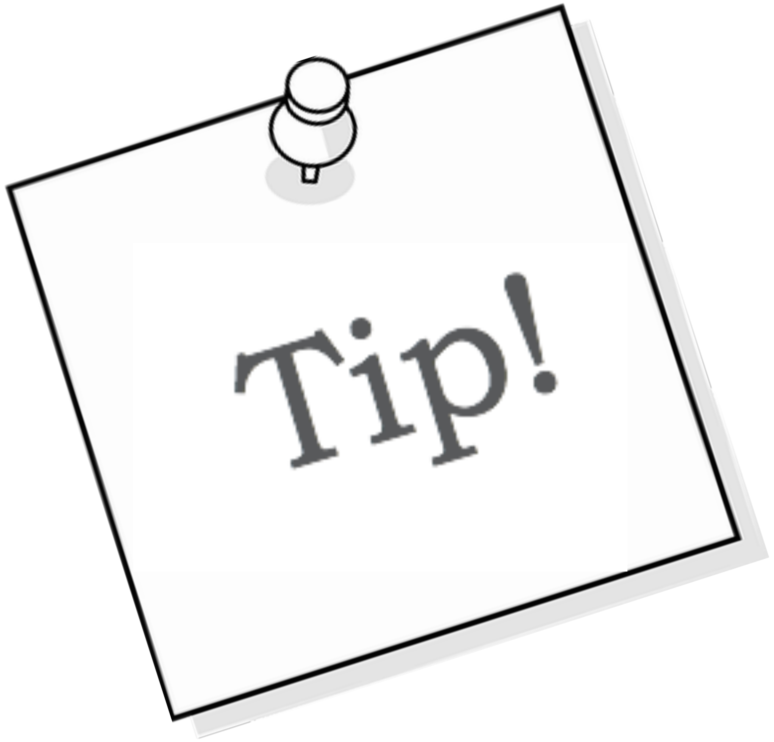
Description

Configuration

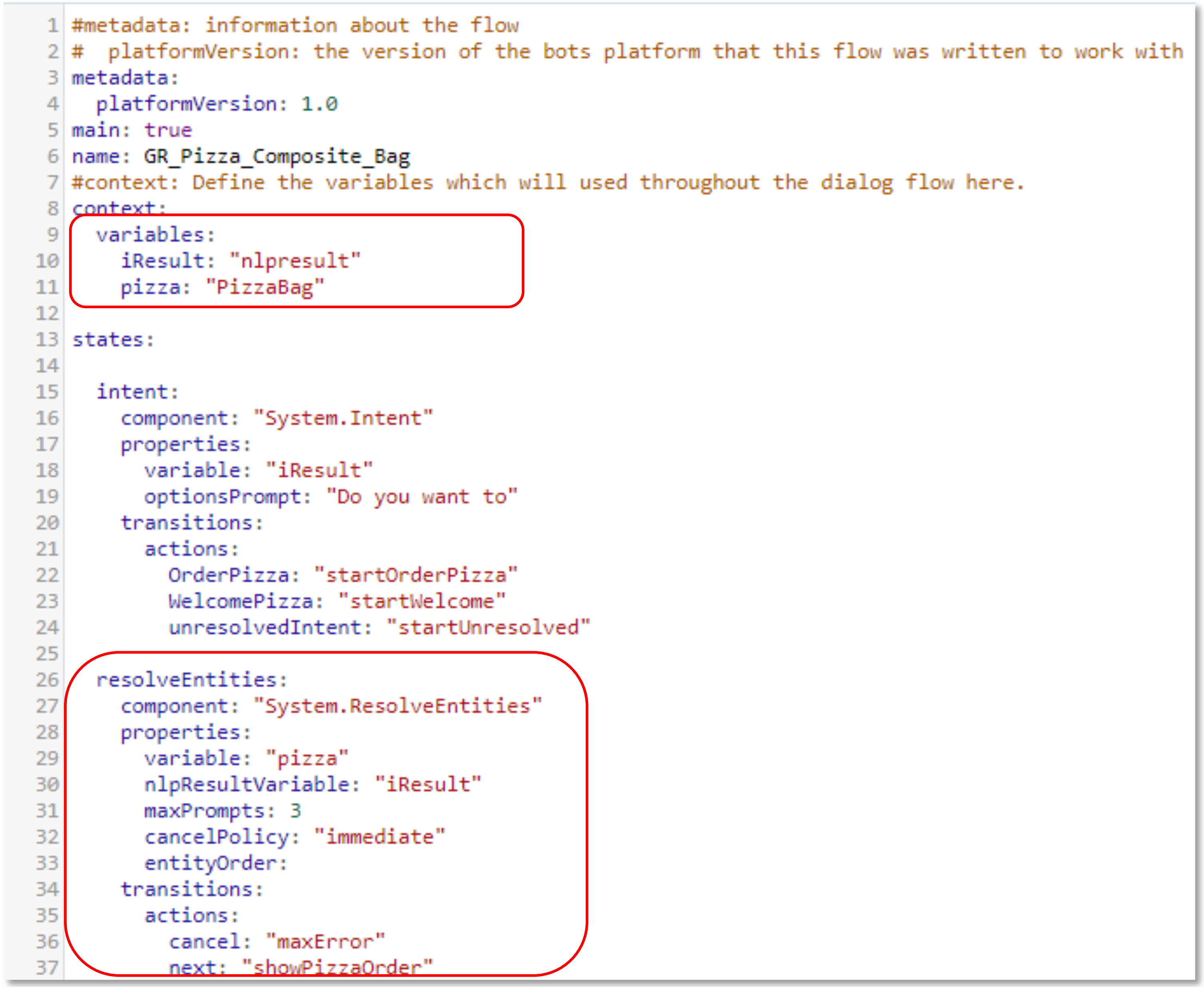
Type: Composite Bag

Bag Items

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



Remember to train if you add an item to composite bag



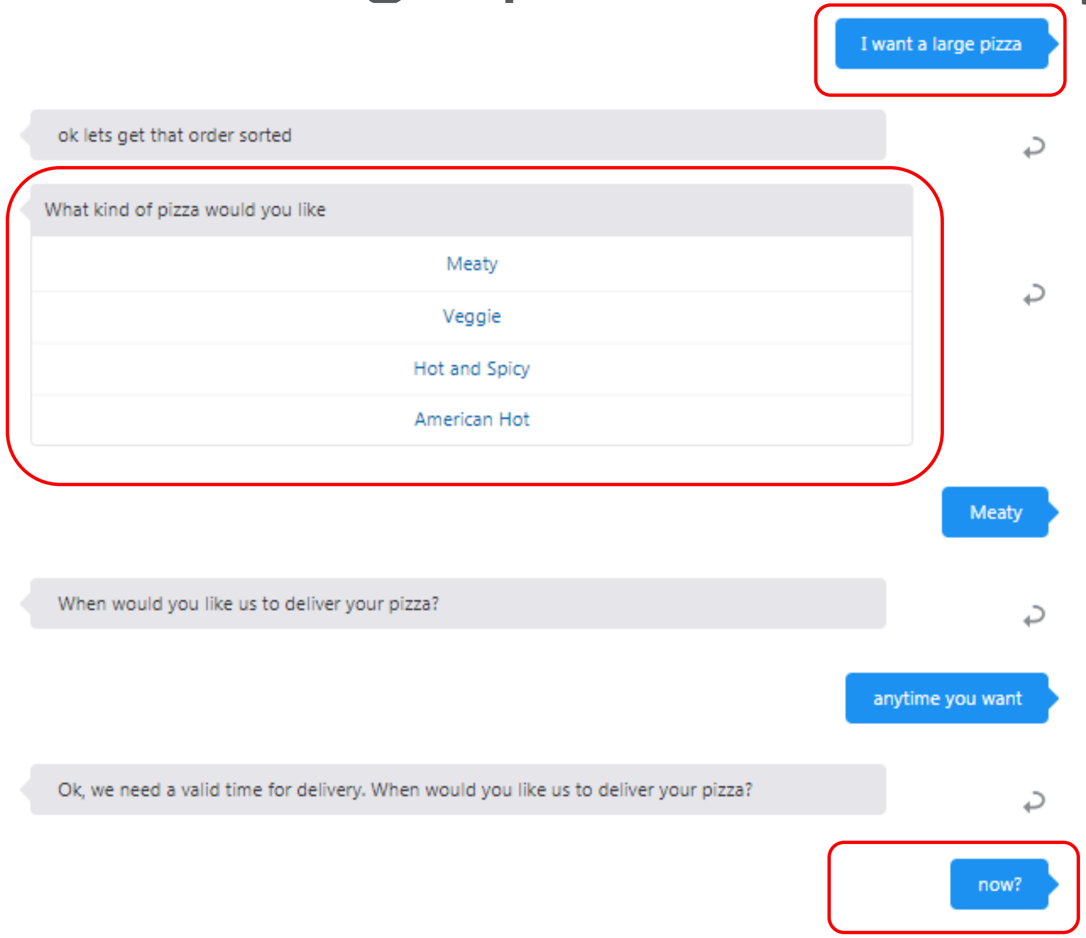
```
#metadata: information about the flow
#  platformVersion: the version of the bots platform that this flow was written to work with
metadata:
  platformVersion: 1.0
main: true
name: GR_Pizza_Composite_Bag
#context: Define the variables which will used throughout the dialog flow here.
context:
  variables:
    iResult: "nlpresult"
    pizza: "PizzaBag"

states:

  intent:
    component: "System.Intent"
    properties:
      variable: "iResult"
      optionsPrompt: "Do you want to"
    transitions:
      actions:
        OrderPizza: "startOrderPizza"
        WelcomePizza: "startWelcome"
        unresolvedIntent: "startUnresolved"

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

# Ordering a pizza with composite bag entity



I want a large pizza

ok lets get that order sorted

What kind of pizza would you like

- Meaty
- Veggie
- Hot and Spicy
- American Hot

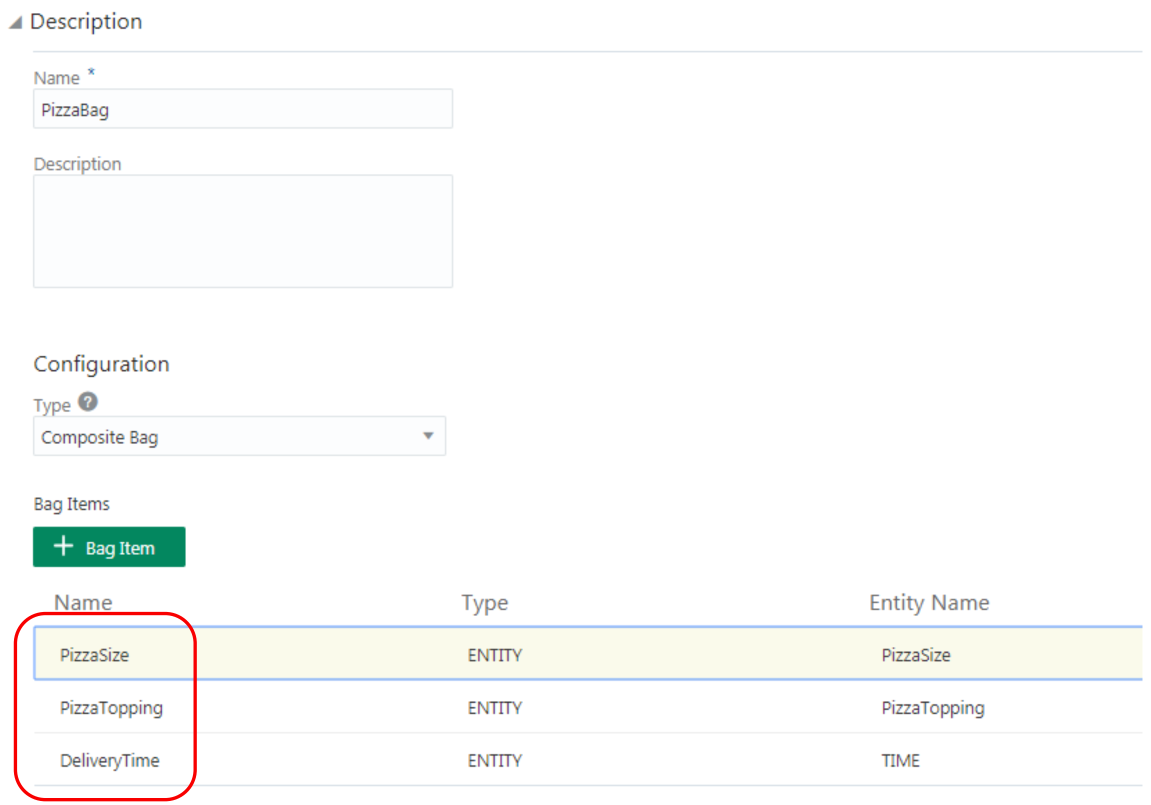
Meaty

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

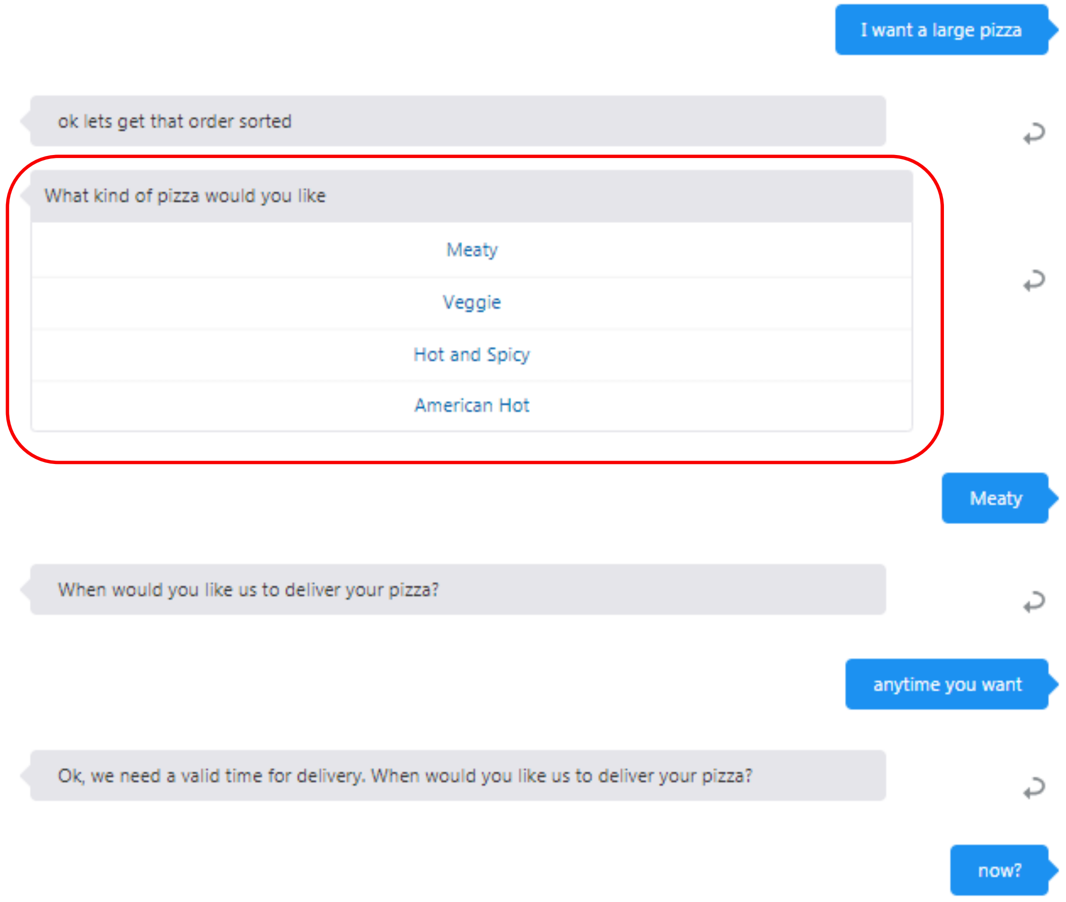
Type

Composite Bag

Bag Items

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

# Composite bag prompts



I want a large pizza

ok lets get that order sorted

What kind of pizza would you like

Meaty

Veggie

Hot and Spicy

American Hot

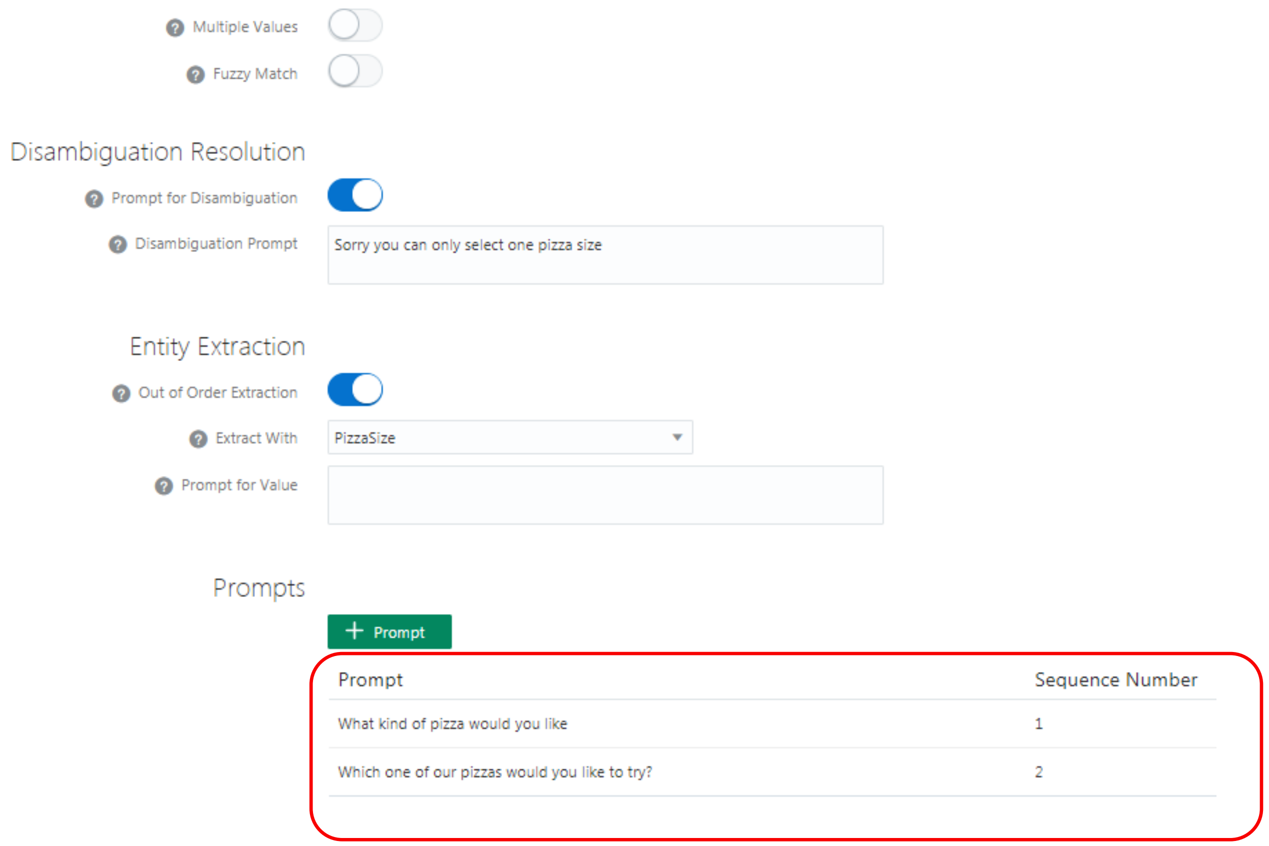
Meaty

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?



Multiple Values: off

Fuzzy Match: off

Disambiguation Resolution

Prompt for Disambiguation: on

Disambiguation Prompt: Sorry you can only select one pizza size

Entity Extraction

Out of Order Extraction: on

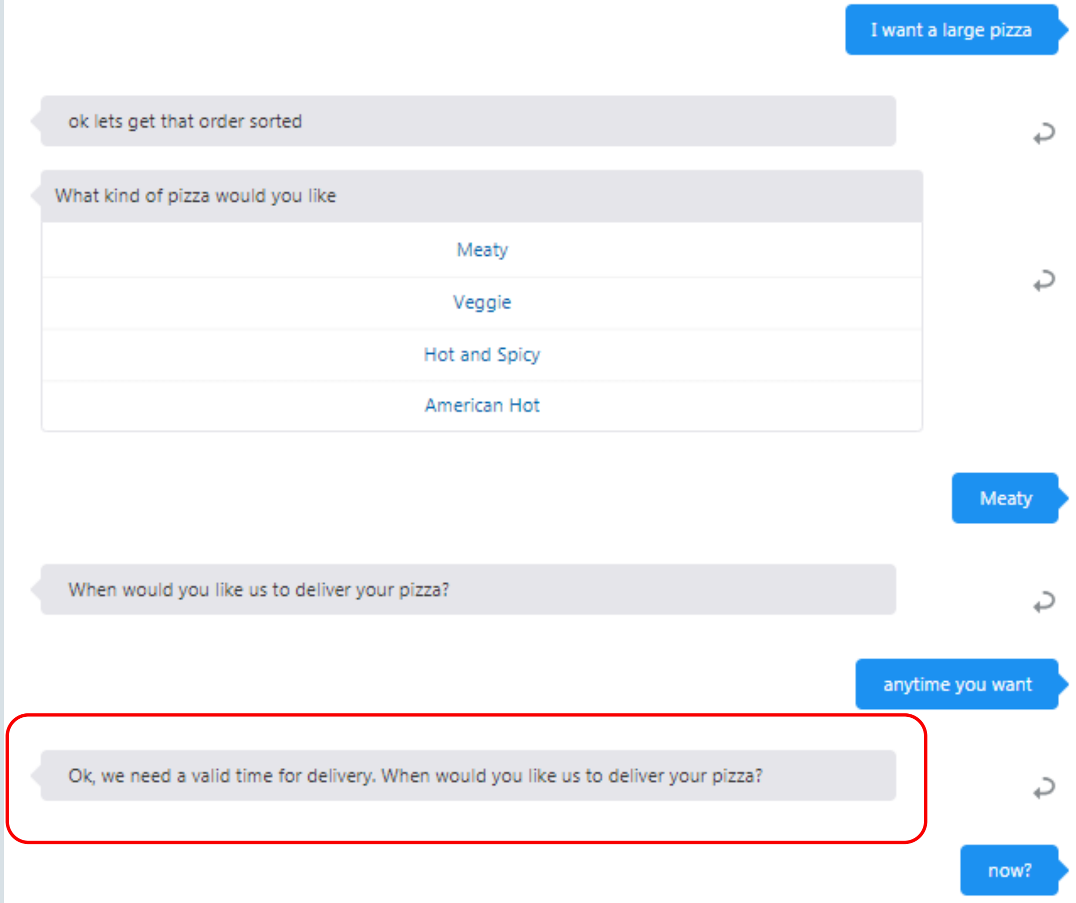
Extract With: PizzaSize

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



I want a large pizza

ok lets get that order sorted

What kind of pizza would you like

Meaty

Veggie

Hot and Spicy

American Hot

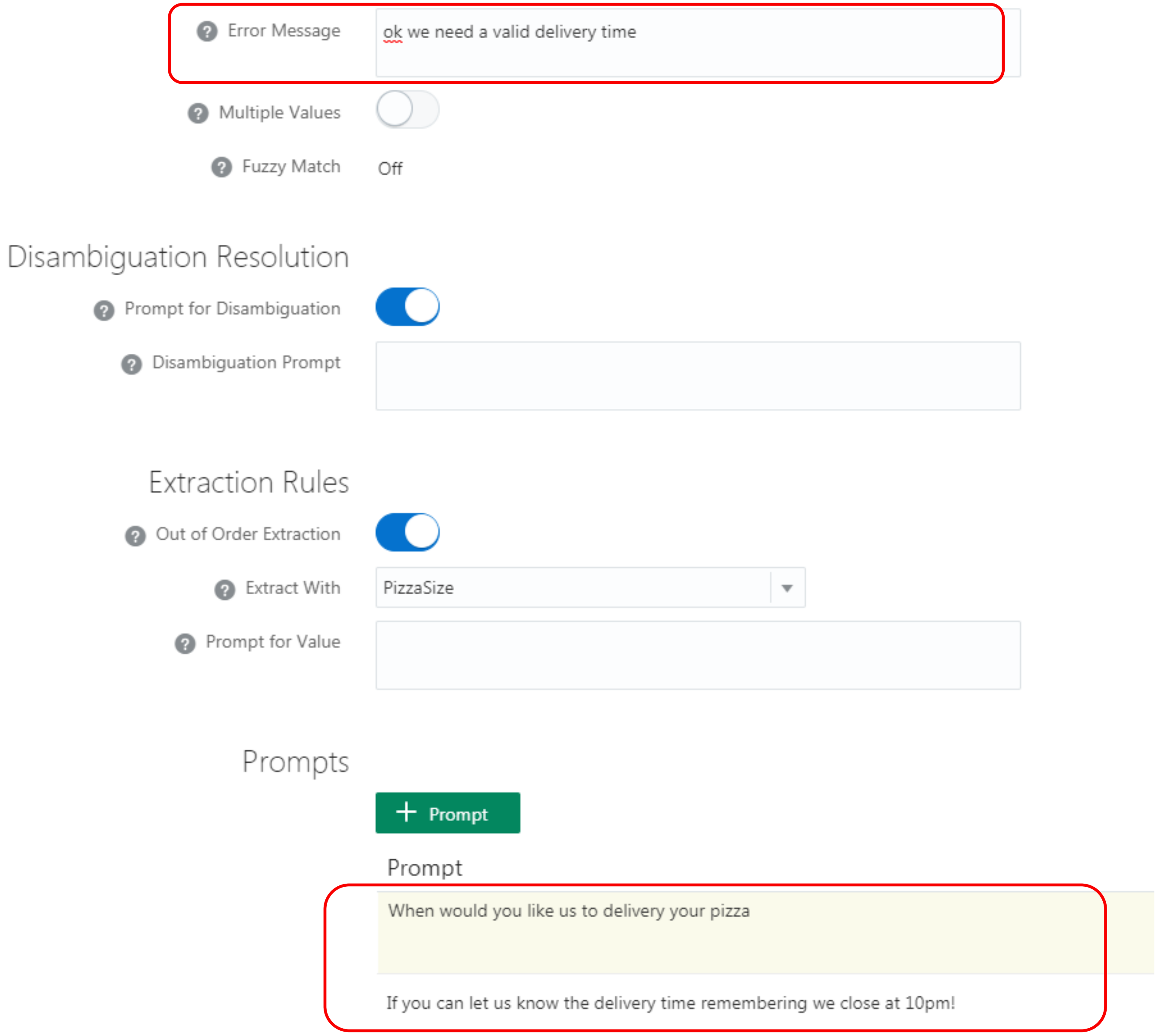
Meaty

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?



Error Message: ok we need a valid delivery time

Multiple Values: Off

Fuzzy Match: Off

Disambiguation Resolution

Prompt for Disambiguation: On

Disambiguation Prompt:

Extraction Rules

Out of Order Extraction: On

Extract With: PizzaSize

Prompt for Value:

Prompts

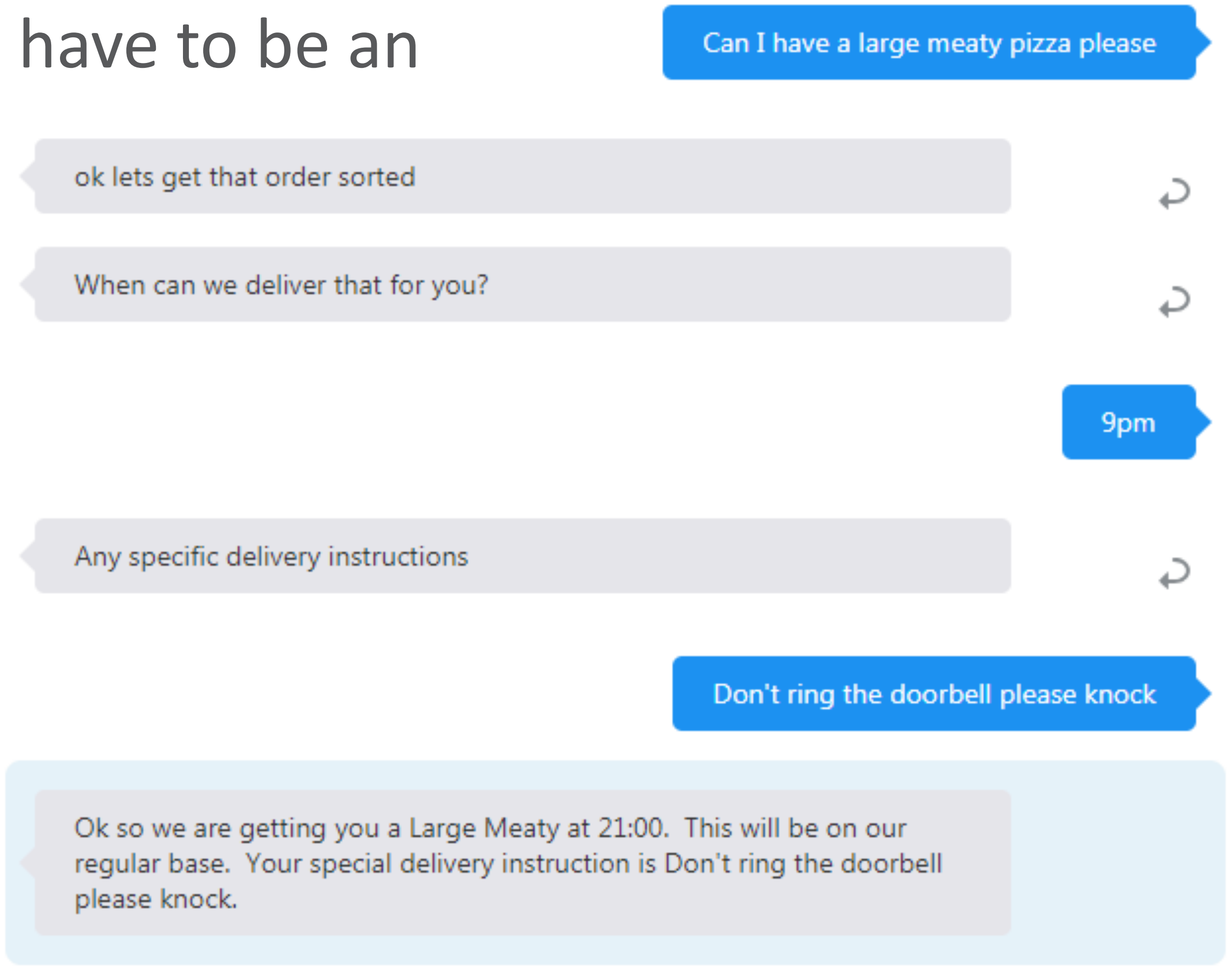
Prompt

When would you like us to delivery your pizza

If you can let us know the delivery time remembering we close at 10pm!

# Composite bag string

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



Can I have a large meaty pizza please

ok lets get that order sorted

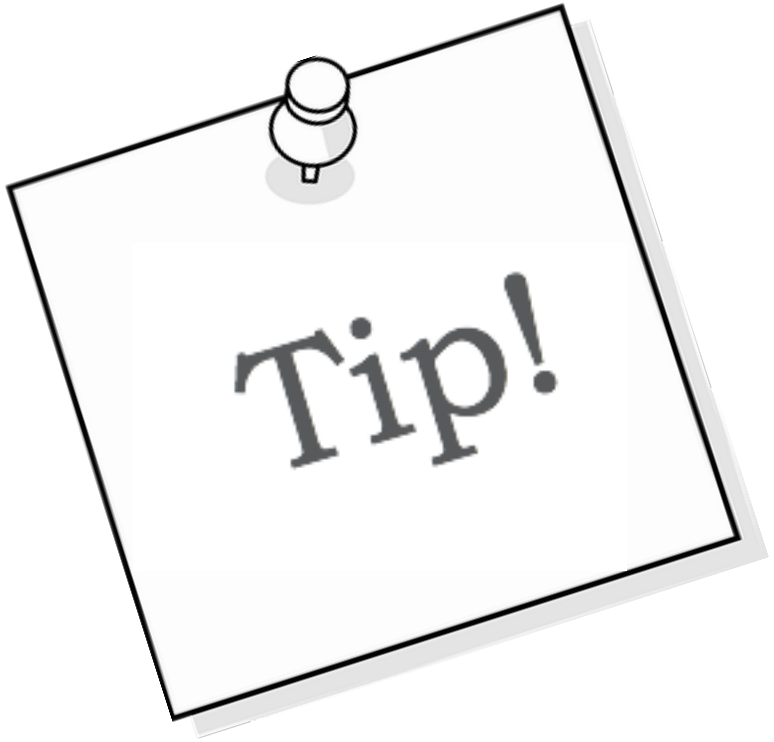
When can we deliver that for you?

9pm

Any specific delivery instructions

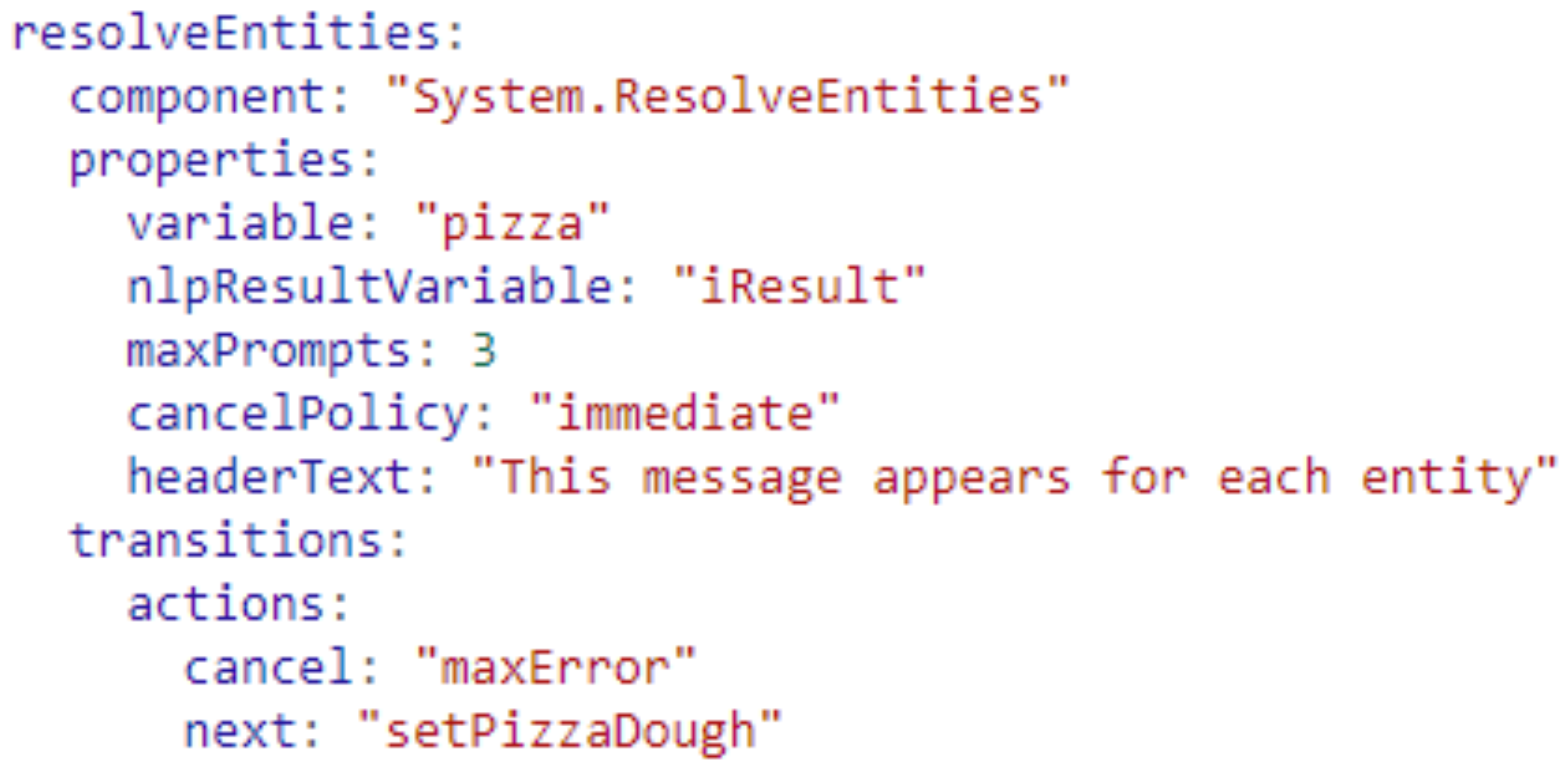
Don't ring the doorbell please knock

Ok so we are getting you a Large Meaty at 21:00. This will be on our regular base. Your special delivery instruction is Don't ring the doorbell please knock.



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



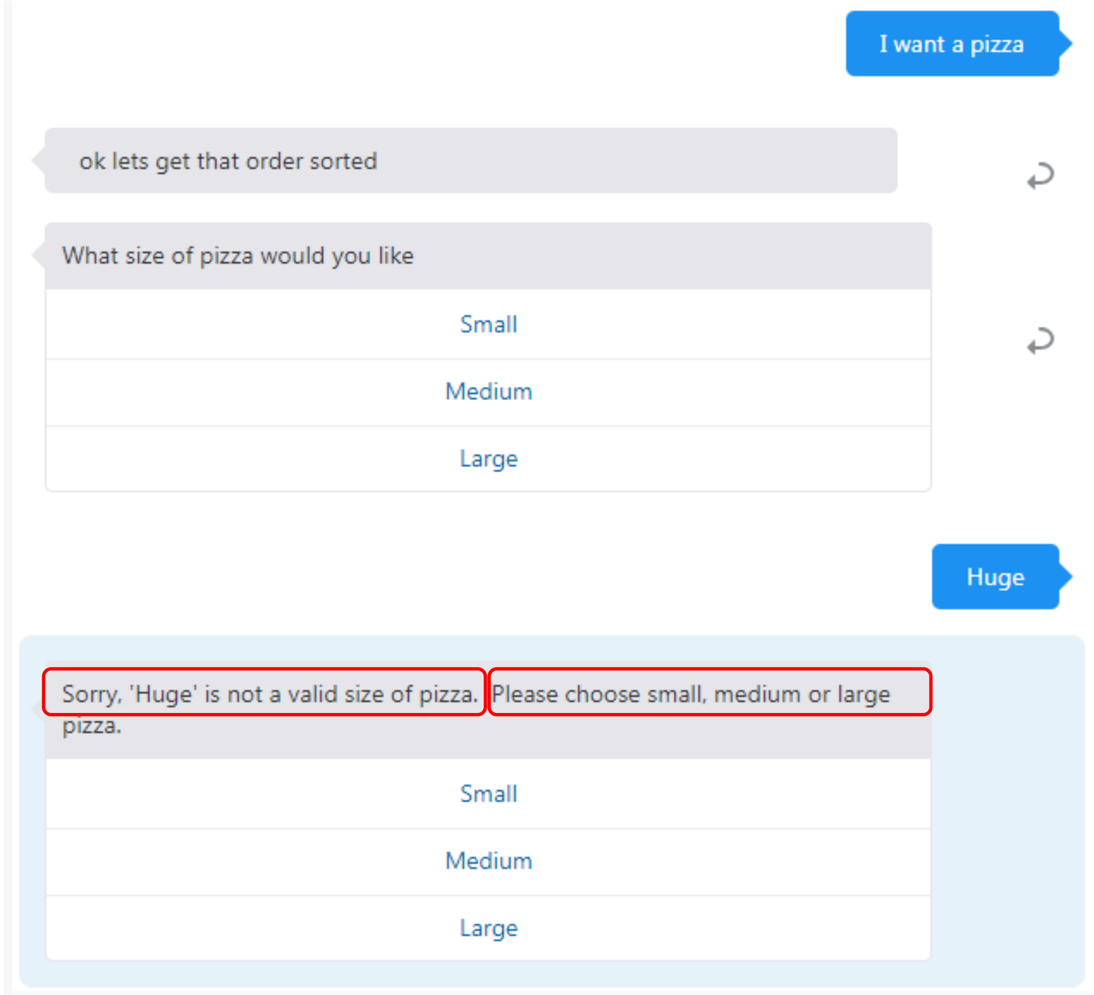
```
resolveEntities:
  component: "System.ResolveEntities"
  properties:
    variable: "pizza"
    nlpResultVariable: "iResult"
    maxPrompts: 3
    cancelPolicy: "immediate"
    headerText: "This message appears for each entity"
  transitions:
    actions:
      cancel: "maxError"
      next: "setPizzaDough"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

## Topic agenda

## Composite bag validation and error handling

- You can define error messages for invalid input
  - This can also include Apache FreeMarker expressions
- You can define validation rules to enforce business rules
  - Also can include Apache FreeMarker 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



I want a pizza

ok lets get that order sorted

What size of pizza would you like

Small

Medium

Large

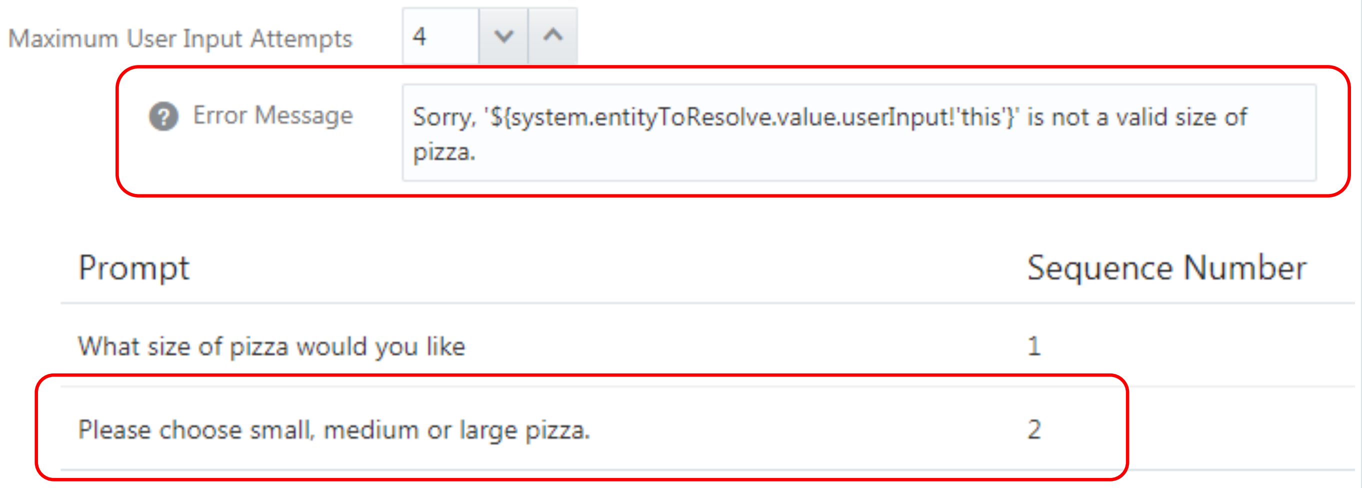
Huge

Sorry, 'Huge' is not a valid size of pizza. Please choose small, medium or large pizza.

Small

Medium

Large

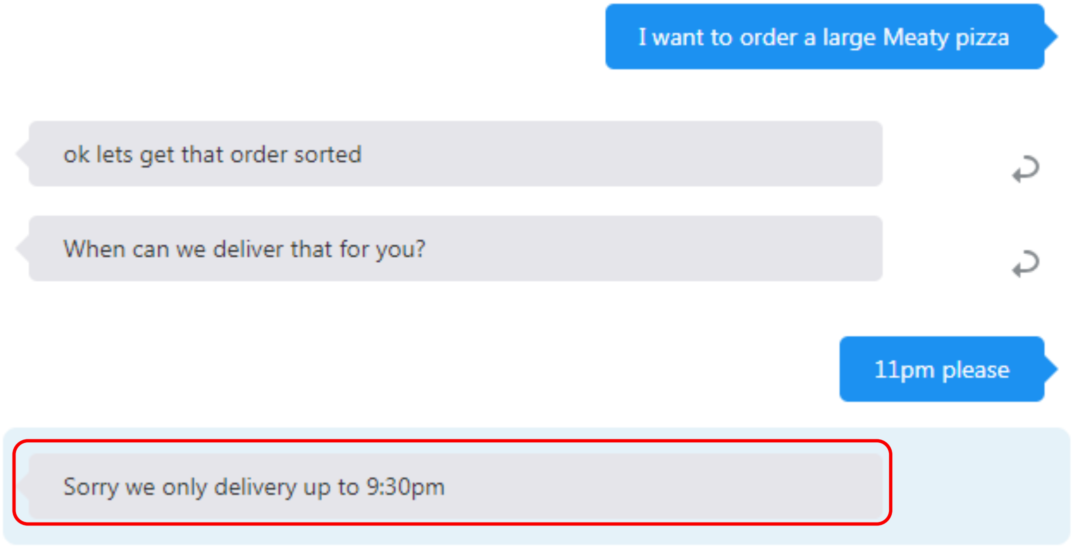


Maximum User Input Attempts: 4

Error Message: Sorry, '${system.entityToResolve.value.userInput!'this'}' is not a valid size of pizza.

| Prompt | Sequence Number |
|---|---|
| What size of pizza would you like | 1 |
| Please choose small, medium or large pizza. | 2 |

# Composite bag validation and error handling



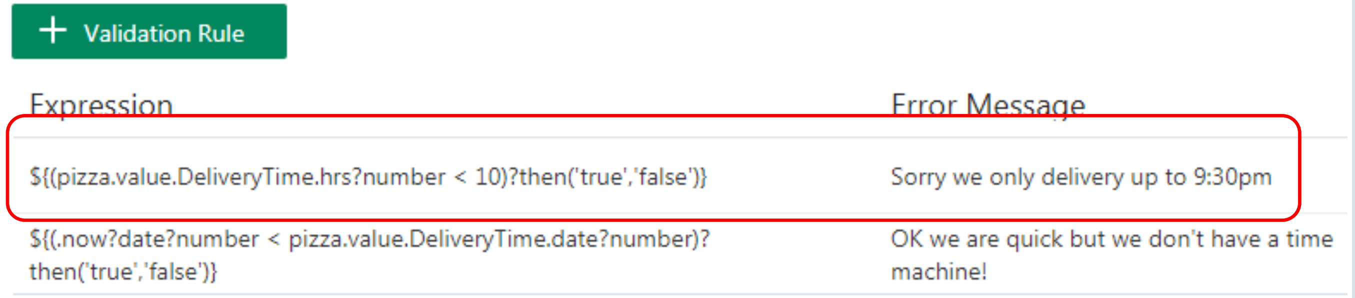
I want to order a large Meaty pizza

ok lets get that order sorted

When can we deliver that for you?

11pm please

Sorry we only delivery up to 9:30pm

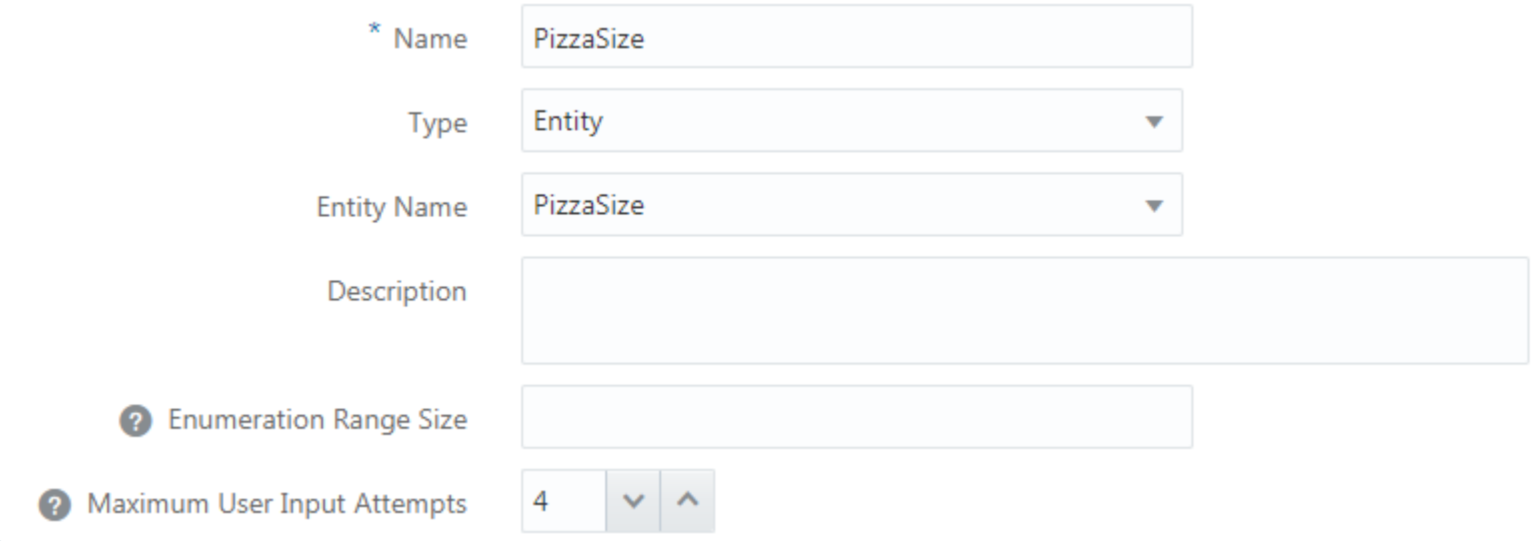


| Expression | Error Message |
|---|---|
| ${(pizza.value.DeliveryTime.hrs?number < 10)?then('true','false')} | Sorry we only delivery up to 9:30pm |
| ${(.now?date?number < pizza.value.DeliveryTime.date?number)?then('true','false')} | OK we are quick but we don't have a time machine! |

## 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"
```



| | |
|---|---|
| * Name | PizzaSize |
| Type | Entity |
| Entity Name | PizzaSize |
| Description | |
| Enumeration Range Size | |
| Maximum User Input Attempts | 4 |

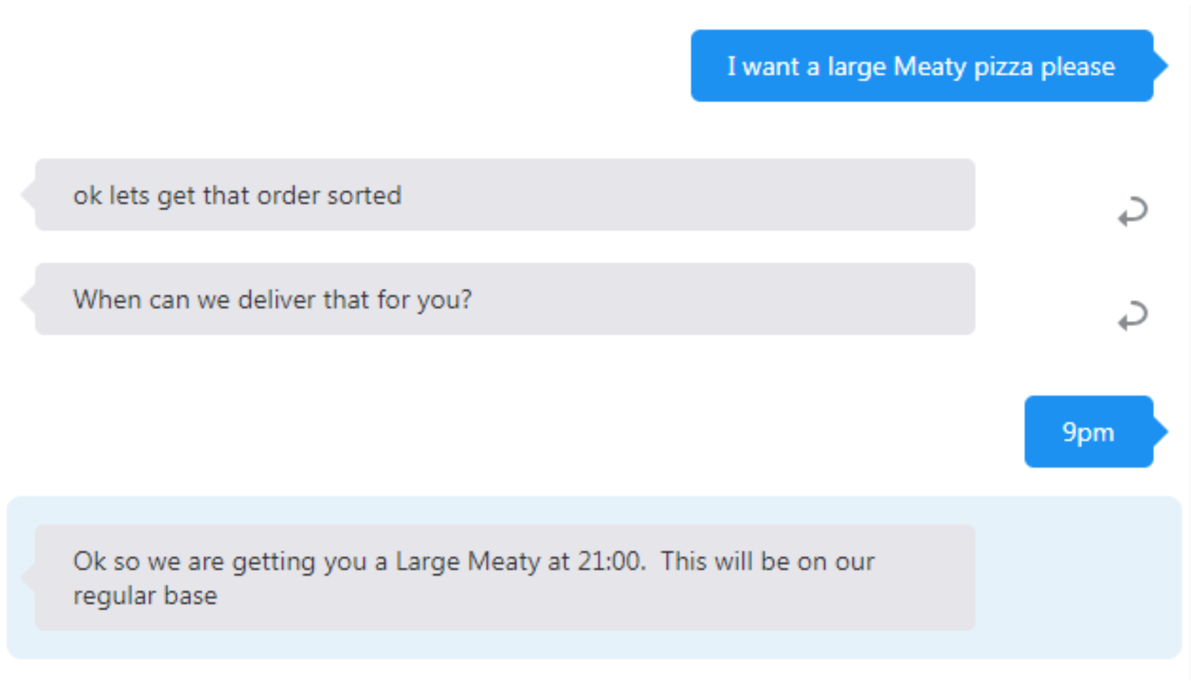
## Topic agenda

## 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

✓ PizzaType
✓ PizzaSize
✓ PizzaCrust
☐ PizzaDough

# Defaulting entity values



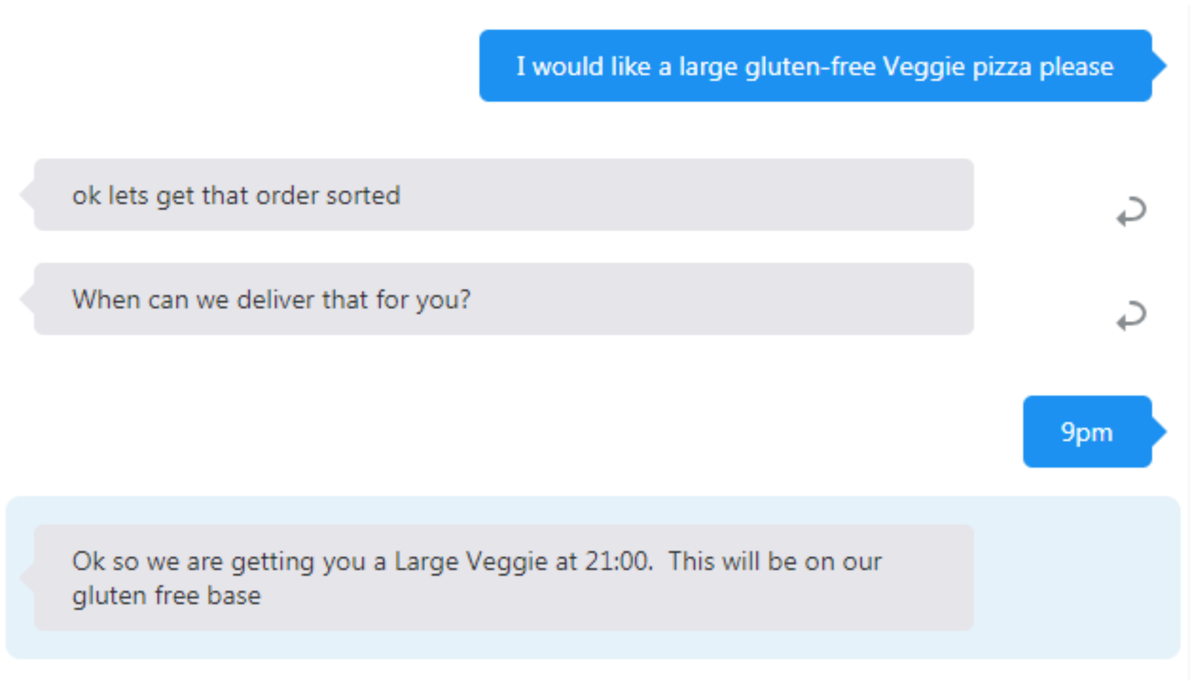
I want a large Meaty pizza please

ok lets get that order sorted

When can we deliver that for you?

9pm

Ok so we are getting you a Large Meaty at 21:00. This will be on our regular base



I would like a large gluten-free Veggie pizza please

ok lets get that order sorted

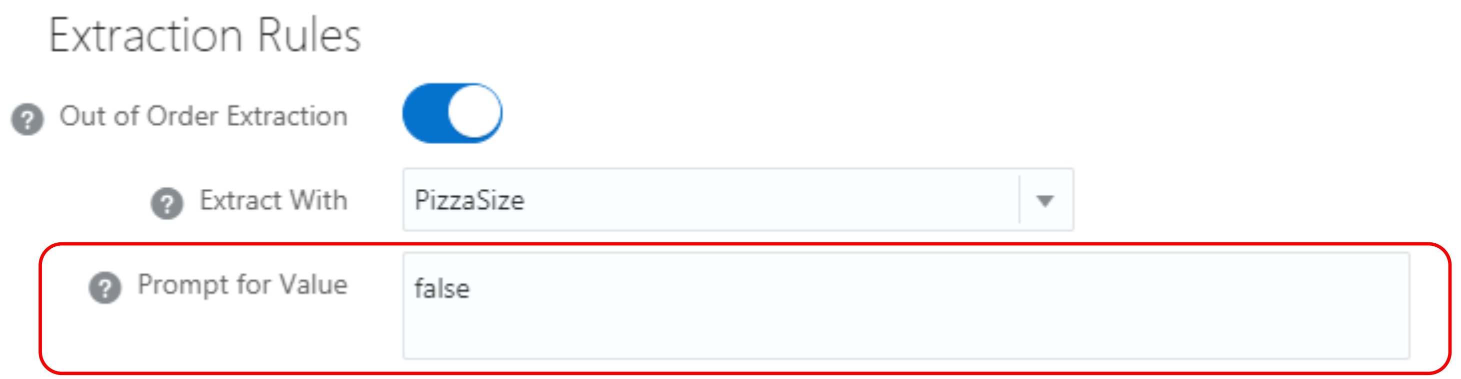
When can we deliver that for you?

9pm

Ok so we are getting you a Large Veggie at 21:00. This will be on our gluten free base

## Defaulting entity values

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


Extraction Rules

Out of Order Extraction: on

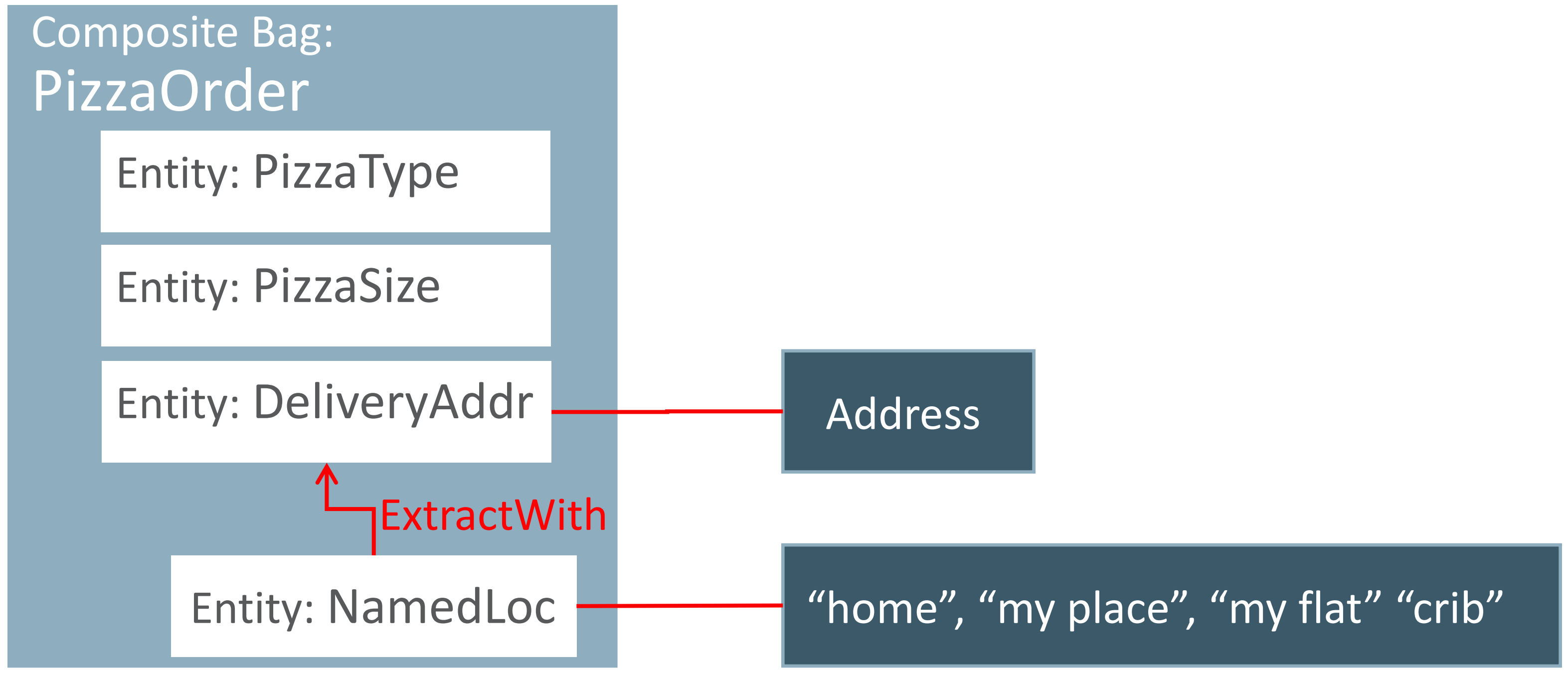
Extract With: PizzaSize

Prompt for Value: false

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

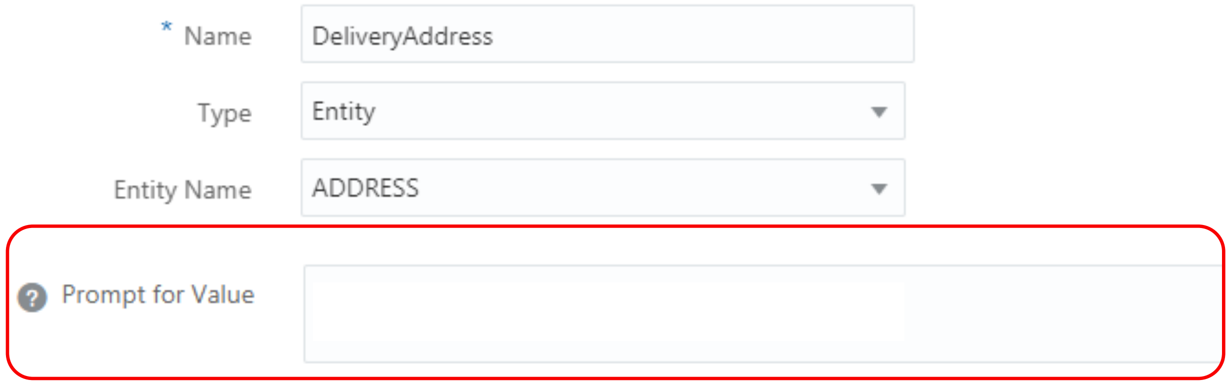
## Allowing related terms for an entity value

- 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”

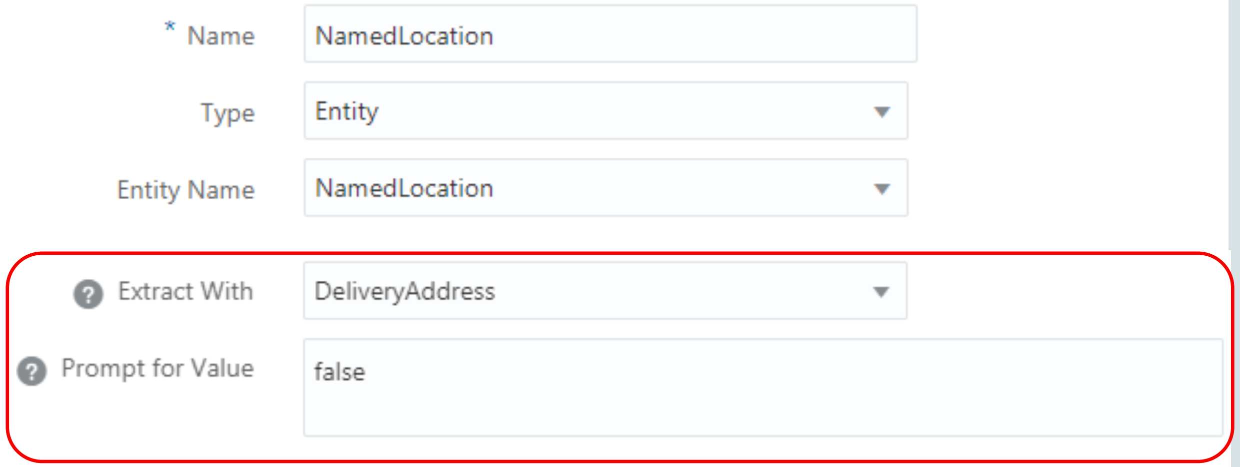


# Allowing related terms for an entity value

- 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



| | |
|---|---|
| * Name | DeliveryAddress |
| Type | Entity |
| Entity Name | ADDRESS |
| Prompt for Value | |

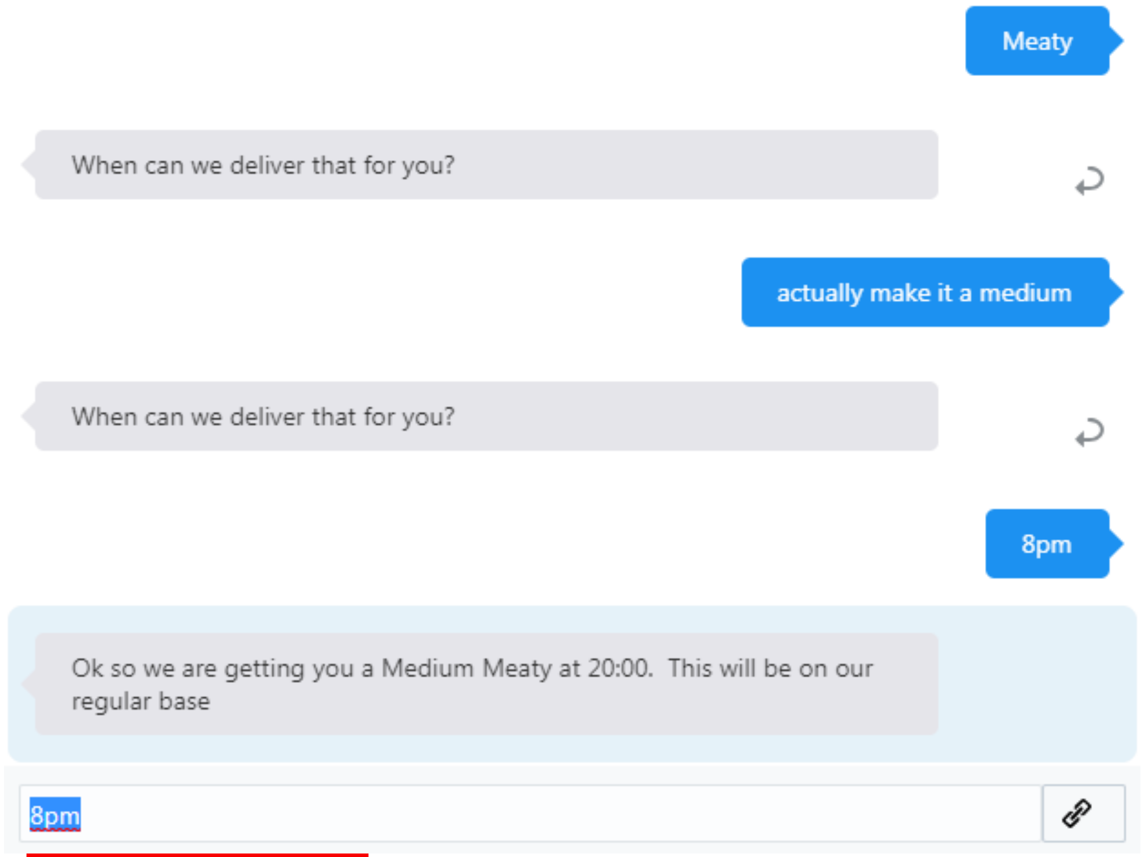


| | |
|---|---|
| * Name | NamedLocation |
| Type | Entity |
| Entity Name | NamedLocation |
| Extract With | DeliveryAddress |
| Prompt for Value | false |

# Topic agenda

## Slotting entities out of order

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



Meaty

When can we deliver that for you?

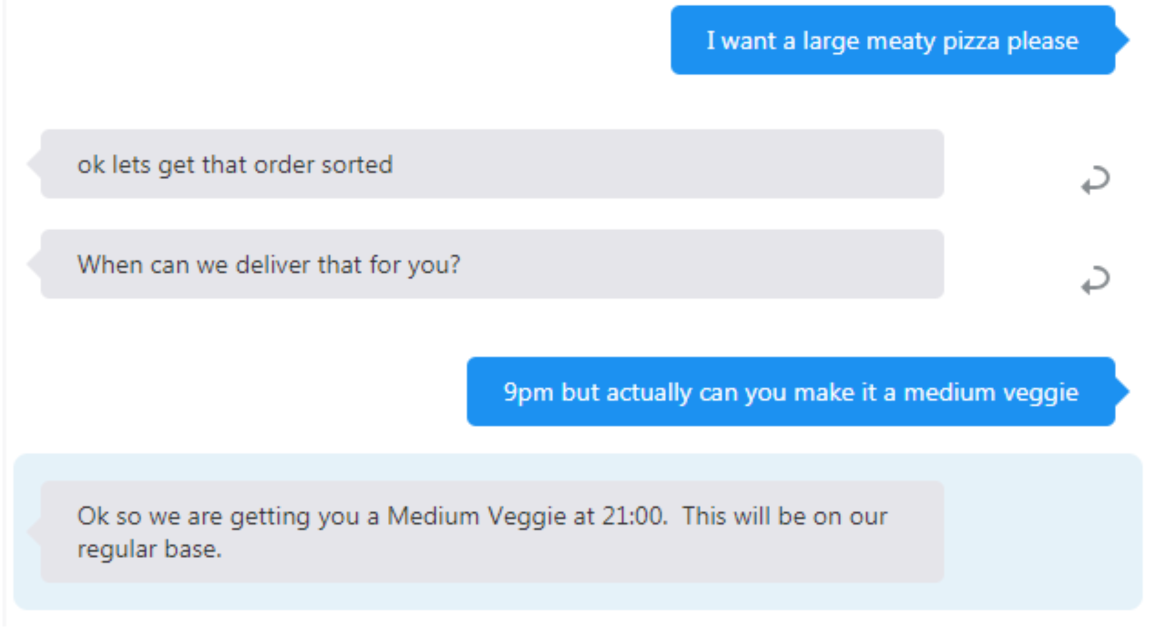
actually make it a medium

When can we deliver that for you?

8pm

Ok so we are getting you a Medium Meaty at 20:00. This will be on our regular base

8pm



I want a large meaty pizza please

ok lets get that order sorted

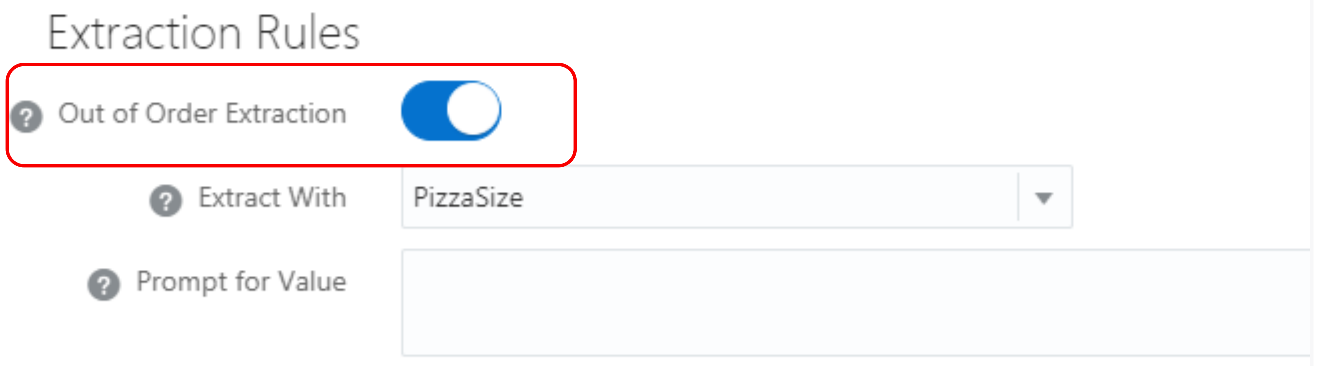
When can we deliver that for you?

9pm but actually can you make it a medium veggie

Ok so we are getting you a Medium Veggie at 21:00. This will be on our regular base.

# 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)

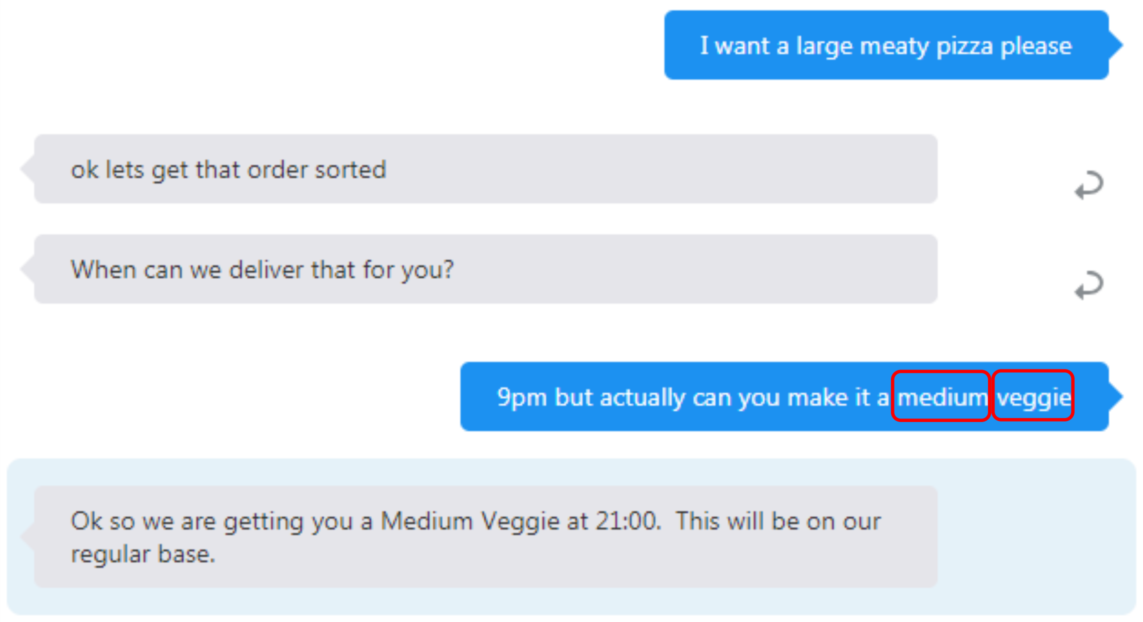


Extraction Rules

Out of Order Extraction [✓ on]

Extract With: PizzaSize

Prompt for Value:



I want a large meaty pizza please

ok lets get that order sorted

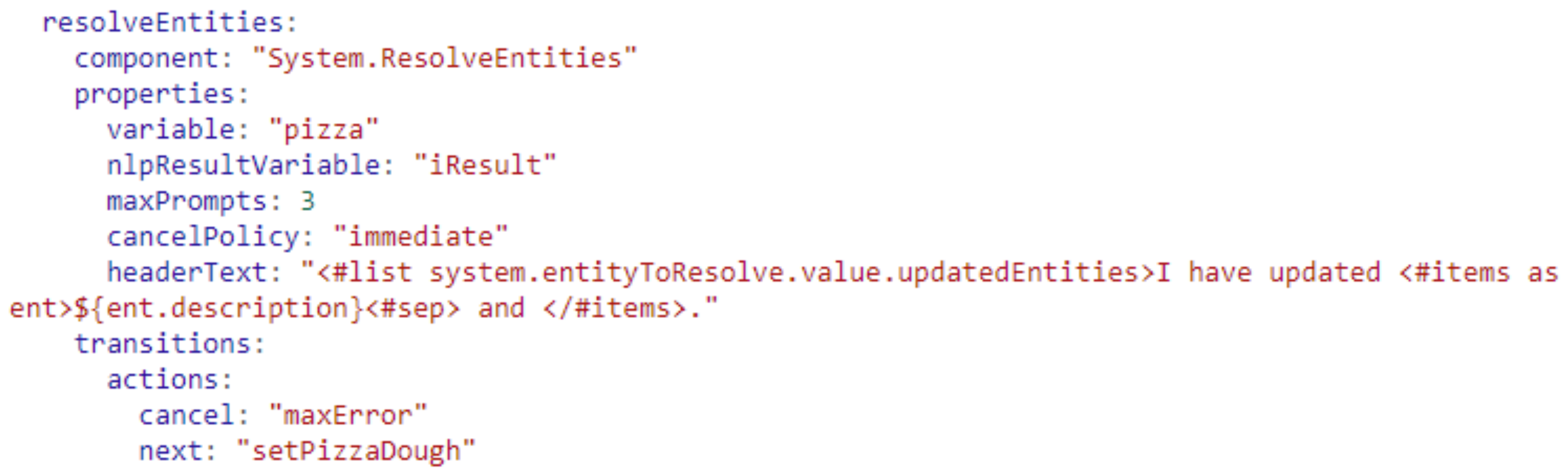
When can we deliver that for you?

9pm but actually can you make it a medium veggie

Ok so we are getting you a Medium Veggie at 21:00. This will be on our regular base.

## Slotting entities out of order

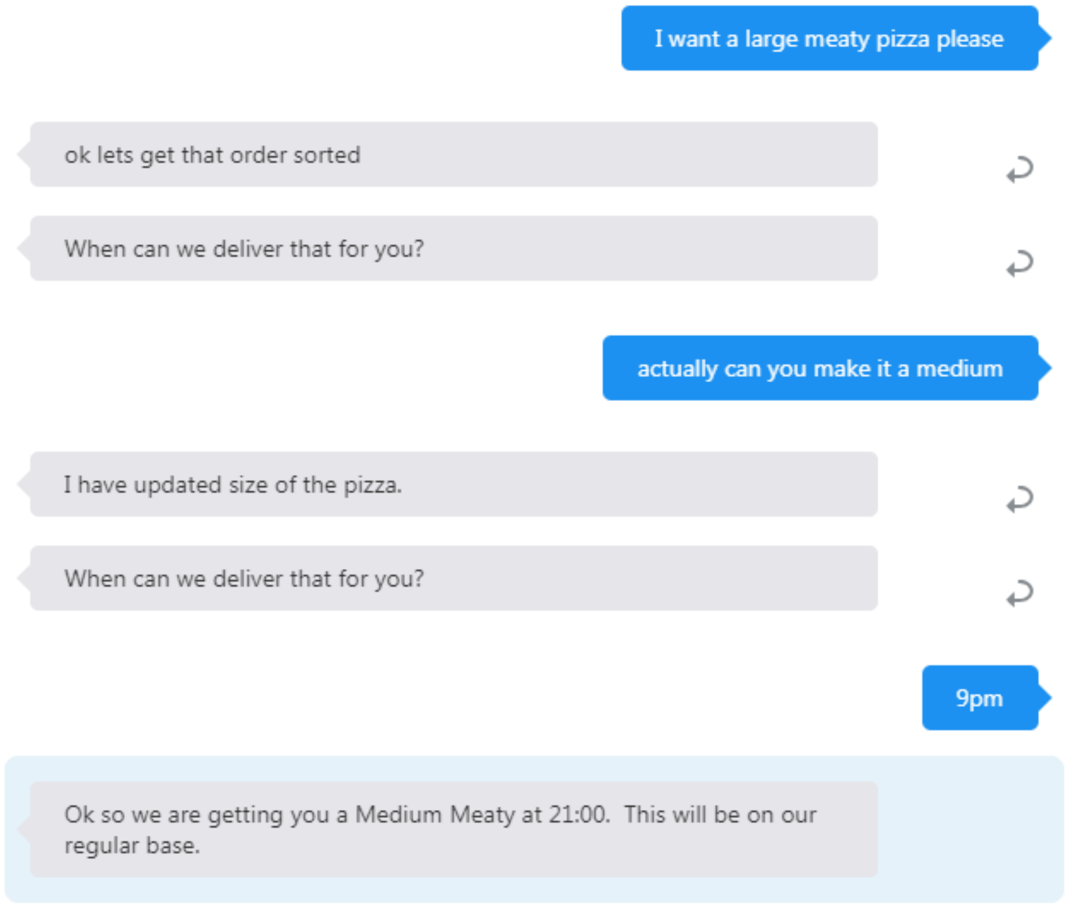
- Confirming to the user that the entity has changed



```
  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>"
```



I want a large meaty pizza please

ok lets get that order sorted

When can we deliver that for you?

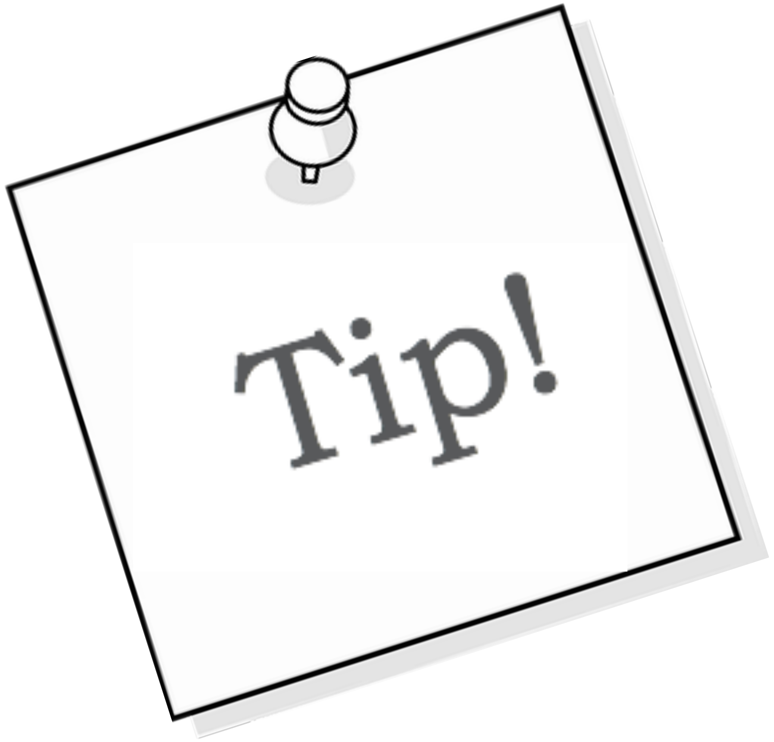
actually can you make it a medium

I have updated size of the pizza.

When can we deliver that for you?

9pm

Ok so we are getting you a Medium Meaty at 21:00. This will be on our regular base.



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

## Oracle Digital Assistant  Hands-On

TBD

# Integrated Cloud

Applications & Platform Services