

ORACLE®

Oracle Digital Assistant

The Complete Training

The System.ResolveEntities Component

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.

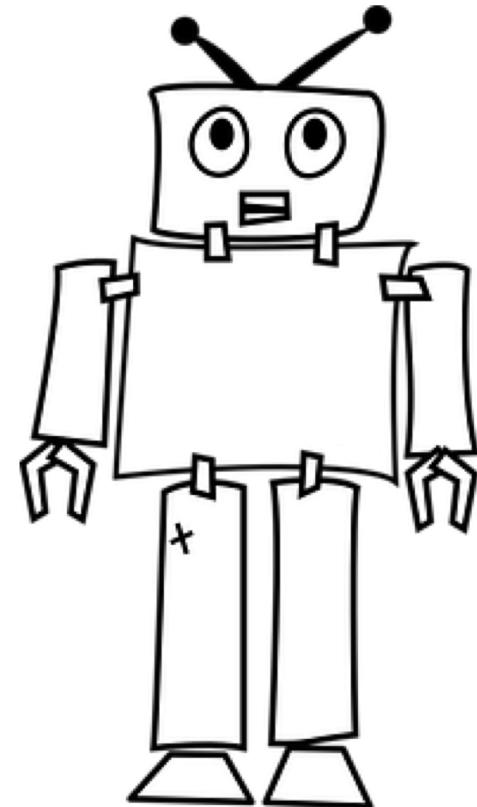
Topic Agenda

- 1 ➤ Entity derived conversations
- 2 ➤ Component overview
- 3 ➤ Use with composite bag entities

Topic Agenda

- 1 ➤ Entity derived conversations
- 2 ➤ Component overview
- 3 ➤ Use with composite bag entities

Dialog flow is the conversation script
that is followed by a skill in a user
interaction. However, the **best dialog**
flow is no dialog flow.



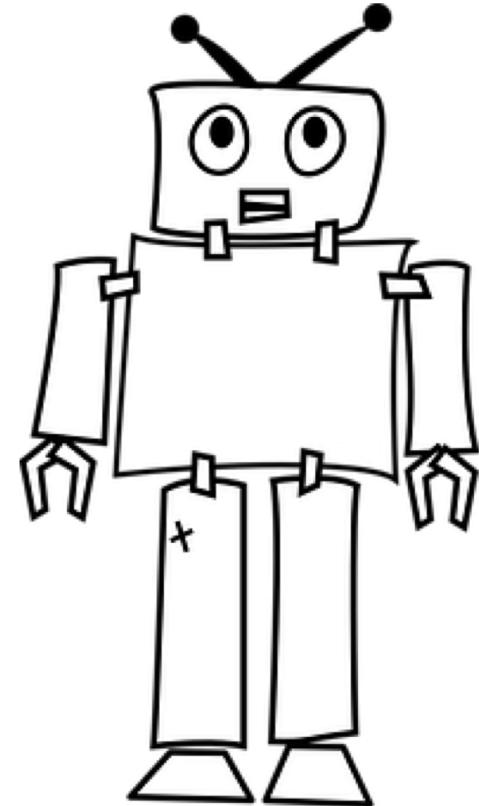
What wrong with dialog flows

- Users are not good at giving a single answer to a question
 - In human-to-human interaction it is natural to overload answers with information
 - Bot: "what pizza type do you like?"
 - User: "a large salami with extra cheese"
- 'story telling' vs. 'data driven'
 - Natural conversation design is chatty
 - Skills only need data input to complete a task
- Violates of the DRY principle (don't repeat yourself)
 - Bot response configured on component
 - Prompt, error message, validation, range size etc.
 - No reuse of settings if configuration is on the component

Entity driven bot conversations

- Reduce the amount of dialog flow steps to write at design time
- Dynamically generate UI at runtime
 - Bot UI rendered based on entity type
 - Simple entities have a single user prompt
 - Composite bag entities may prompt users multiple times
 - All configurations and behaviors are defined on the entity
 - Prompts, error message, validation rules, range size
 - Entity extraction, out-of-order message handling (composite bag entity only)
- Oracle Digital Assistant promotes entity derived conversations
- Require use of `System.ResolveEntities` and `System.CommonResponse` components

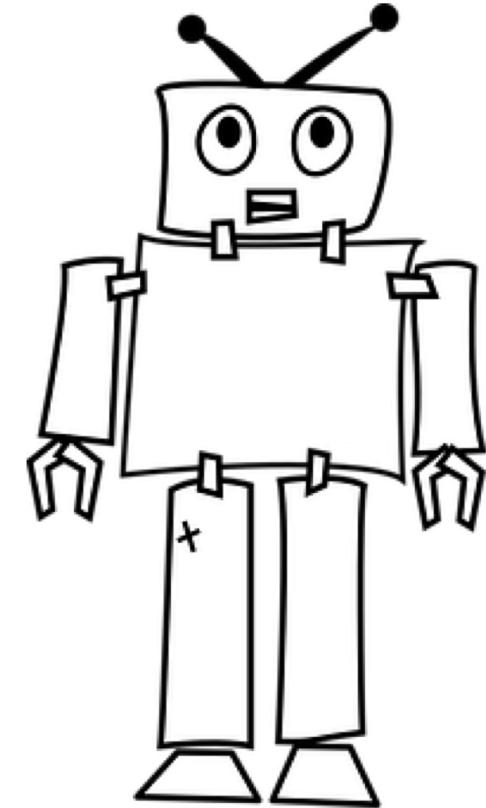
Entity driven conversation delegates common component configurations to the entity level, which is a much better model for reuse.



Topic Agenda

- 1 ➤ Entity derived conversations
- 2 ➤ Component overview
- 3 ➤ Use with composite bag entities

You can use the System.ResolveEntities component with system, custom and **composite bag entities**. It generates input fields (prompts) and value lists.



Building System.ResolveEntities from component template

The screenshot shows the Oracle Dialog Editor interface. At the top, there's a toolbar with a green button labeled '+ Components' and a question mark icon. Below it is a sidebar with a code snippet and a 'Select a Component Type' dropdown. The main area displays five categories: Control (Control tree icon), Language (Speech bubble icon), Security (Padlock icon), User Interface (Hand icon), and Variables (Variables icon). A modal window titled 'Component Template' is open over the 'User Interface' category. The modal has tabs for 'Interactive', 'List - set action', 'List - set variable', 'Output', 'Text', and 'Webview'. The 'Output' tab is selected, and the 'Resolve entities' option is highlighted with a red box. To the right of the tabs is a detailed description of the component template:

```
resolveEntities:  
  component: "System.ResolveEntities"  
  properties:  
    # variable (required) refers to the composite entity context  
    # variable that will be populated by this component. If all child  
    # entities of the composite entity variable already have a value, then  
    # the dialog flow transitions to the next state and no message to the  
    # user is sent.  
    variable:  
      # nlpResultVariable (optional) refers to the nlpresult variable  
      # that can be used to resolve (part of) the composite entity variable.  
      # If the nlpResultVariable value contains an entity match of the same  
      # type as one of the child entities of the composite entity variable,  
      # then this child entity value will be set inside the variable value.  
      # If all child entities are populated by the entity matches in the  
      # nlpResultVariable, the dialog flow will transition to the next state.
```

At the bottom of the modal are buttons for 'Insert After', 'handleMaxPromptsExc...', 'Remove Comments' (with a toggle switch), and 'Apply'.

System.ResolveEntities component with custom entity

Entity Manager interface showing configuration for a custom entity named "Airports".

Configuration:

- Type: Value list
- Value:
 - LAX
 - SFO
 - LHR
 - MUC
 - CDG

Enumeration Range: 3

Prompts:

- + Prompt

Prompt: Please provide an airport code

```
context:  
variables:  
airports: "Airports"  
iResult: "nlpresult"  
  
states:  
  
showAirports:  
component: "System.ResolveEntities"  
properties:  
variable: "airports"  
nlpResultVariable: "iResult"  
maxPrompts: 1  
cancelPolicy: "immediate"  
transitionAfterMatch: "true"  
autoNumberPostbackActions: false  
headerText:  
footerText:  
showMoreLabel: "Show More"  
translate:  
transitions:  
actions:  
match: "handleEntityMatch"  
cancel: "handleFailedValidInput"
```

UI screenshot showing the results of the System.ResolveEntities component.

Please provide an airport code

LAX
SFO
LHR

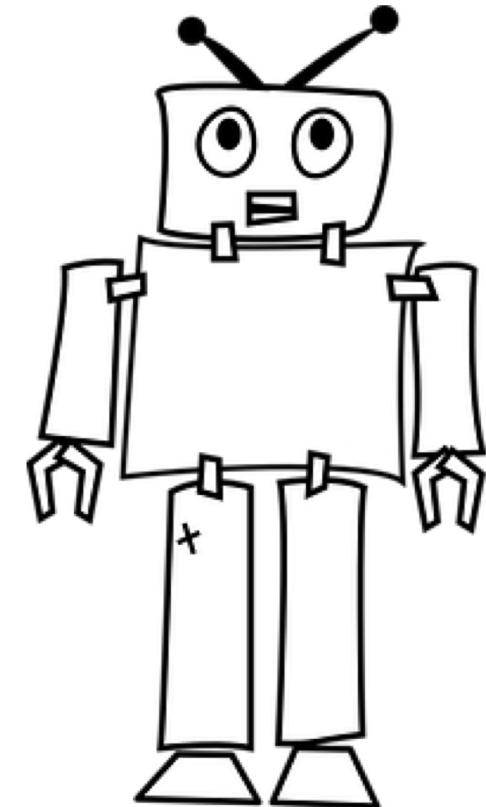
Show More

Message

Topic Agenda

- 1 ➤ Entity derived conversations
- 2 ➤ Component overview
- 3 ➤ Use with composite bag entities

You need to train **the skill bot model** before using `System.ResolveEntities` with composite bag entities.



System.ResolveEntities with composite bag entity

The screenshot shows the Oracle Service Cloud interface for managing entities. On the left, a sidebar lists categories like Entity, More, Filter, Sort By (Created Ascending), and various entity types (Travel, Airports, CabinClass, ADDRESS, CURRENCY, DATE). The main area displays a configuration for an entity named "Travel". The "Configuration" section shows it is a "Composite Bag" type. Below this, a table lists "Bag Items":

Name	Type	Entity Name
DestinationAirport	ENTITY	Airports
DepartureDate	ENTITY	DATE
Cabin	ENTITY	CabinClass

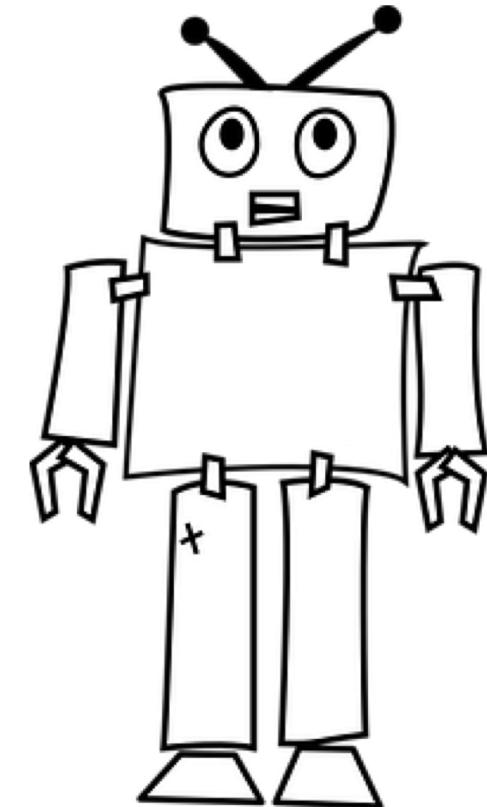
On the right, a large block of JSON configuration code is shown, with several sections highlighted in red:

```
context:  
variables:  
  booking: "Travel"  
  iResult: "nlpresult"  
  
states:  
showAirports:  
  component: "System.ResolveEntities"  
properties:  
  variable: "booking"  
  nlpResultVariable: "iResult"  
  maxPrompts: 1  
  cancelPolicy: "immediate"  
  transitionAfterMatch: "true"  
  autoNumberPostbackActions: false  
headerText:  
footerText:  
showMoreLabel: "Show More"  
translate:  
transitions:  
actions:  
  match: "handleEntityMatch"  
  cancel: "handleFailedValidInput"
```

Three input fields are displayed, each with a blue arrow pointing from the configuration code to its respective section:

- An input field for "Please provide an airport code" containing "LAX", "SFO", and "LHR". A "Show More" button is present.
- An input field for "Please provide a departure date" containing "January 23rd 2019". A "Show More" button is present.
- An input field for "Please select a cabin class" containing "Economy", "Economy Plus", and "Business". A "Show More" button is present.

Okay. We need to talk. What if you want to **perform additional validation** or just need to **invoke a custom component** in response to a matched entity?



'transitionAfterMatch' property

- If set to "true", component transitions to dialog flow state upon entity match
 - String "true", not the boolean true
 - Bot designers can call custom component or just acknowledge the value match
- 'match' action transition called for each entity match

What kind of pizza would you like to order?

CHEESE BASIC
PEPPERONI
MEAT LOVER
SUPREME
PREMIUM GARDEN VEGGIE
ULTIMATE CHEESE LOVER
HAWAIIAN CHICKEN
BACON SPINACH ALFREDO

PEPPERONI

Confirming entity match in composite bag:

Bag item: Type
Entity name: PizzaType
Entity value: PEPPERONI

What size do you want?

Large
Medium
Small
X-Large

Small

Confirming entity match in composite bag:

Bag item: Size
Entity name: PizzaSize
Entity value: Small

Navigating on entity match

```
orderPizza:  
  component: "System.ResolveEntities"  
  properties:  
    variable: "pizza"  
    nlpResultVariable: "iResult"  
    maxPrompts: 2  
    transitionAfterMatch: "true"  
    showMoreLabel: "More"  
    translate:  
  transitions:  
    next: "confirmation"  
    actions:  
      match: "handleAfterMatch"  
      cancel: "cancelOrder"  
  
handleAfterMatch:  
  component: "System.Output"  
  properties:  
    text: |-  
      Confirming entity match in composite bag:  
  
      Bag item: ${system.entityToResolve.value.resolvingField}  
      Entity name: ${system.entityToResolve.value.allMatches[0].entityName}  
      Entity value: ${pizza.value[system.entityToResolve.value.resolvingField]}  
  
  keepTurn: true  
  transitions:  
    #resume orderPizza  
    next: "orderPizza"
```

What kind of pizza would you like to order?

CHEESE BASIC
PEPPERONI
MEAT LOVER
SUPREME
PREMIUM GARDEN VEGGIE
ULTIMATE CHEESE LOVER
HAWAIIAN CHICKEN
BACON SPINACH ALFREDO

PEPPERONI

Confirming entity match in composite bag:

Bag item: Type
Entity name: PizzaType
Entity value: PEPPERONI

What size do you want?

Large
Medium
Small
X-Large

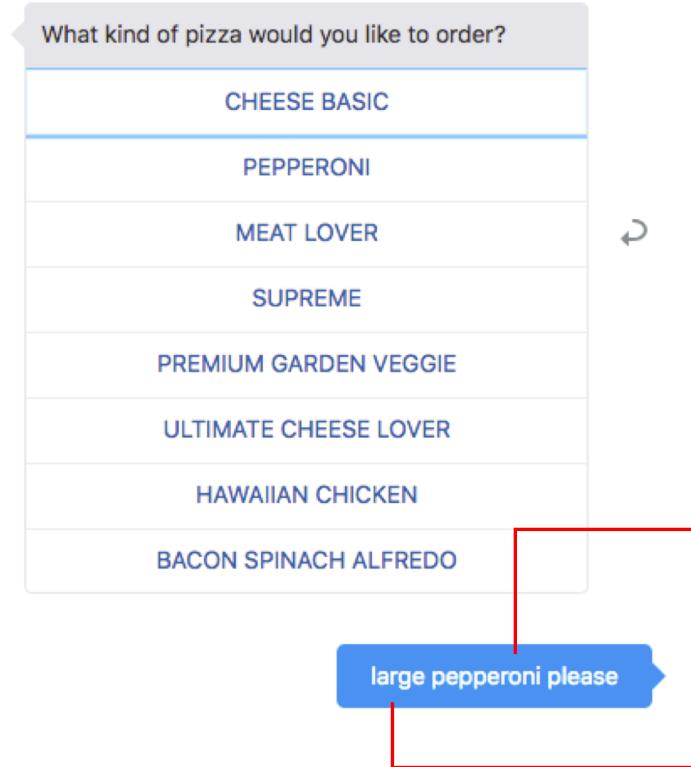
Small

Confirming entity match in composite bag:

Bag item: Size
Entity name: PizzaSize
Entity value: Small

Behavior when multiple entities are getting resolved

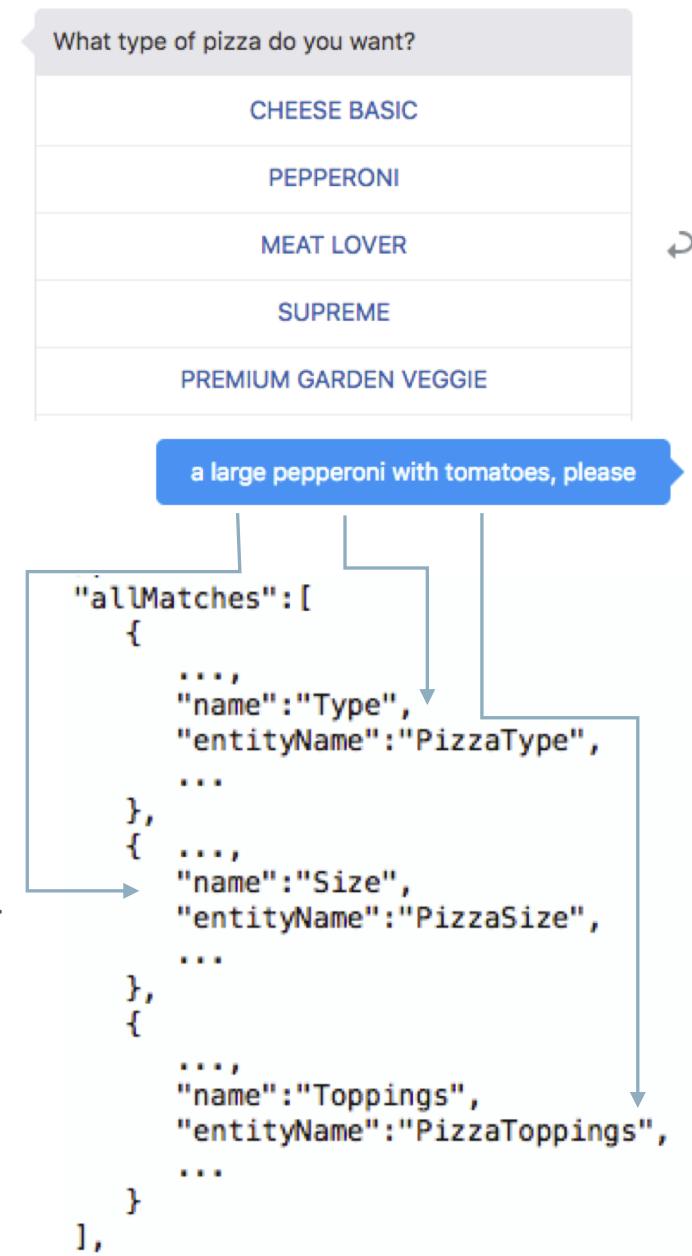
- User input may lead to multiple entity matches
 - Out-of-order extraction
- 'match' transition is called only once
- Access matched bag items
 - \${system.entityToResolve.value.allMatches[n].entityName}
 - \${system.entityToResolve.value.allMatches[n].name}



```
system
  security.configuredAuthenticationServices: upgr
entityToResolve
  nextRangeStart: 0
  updatedEntities
  needShowMoreButton: false
outOfOrderMatches
  rangeStartVar:
  transitioningAfterMatch: false
  validationErrors
allMatches
  0
  1
  resolvingField: Toppings
  userInput: large pepperoni please
  skippedItems
  disambiguationValues
```

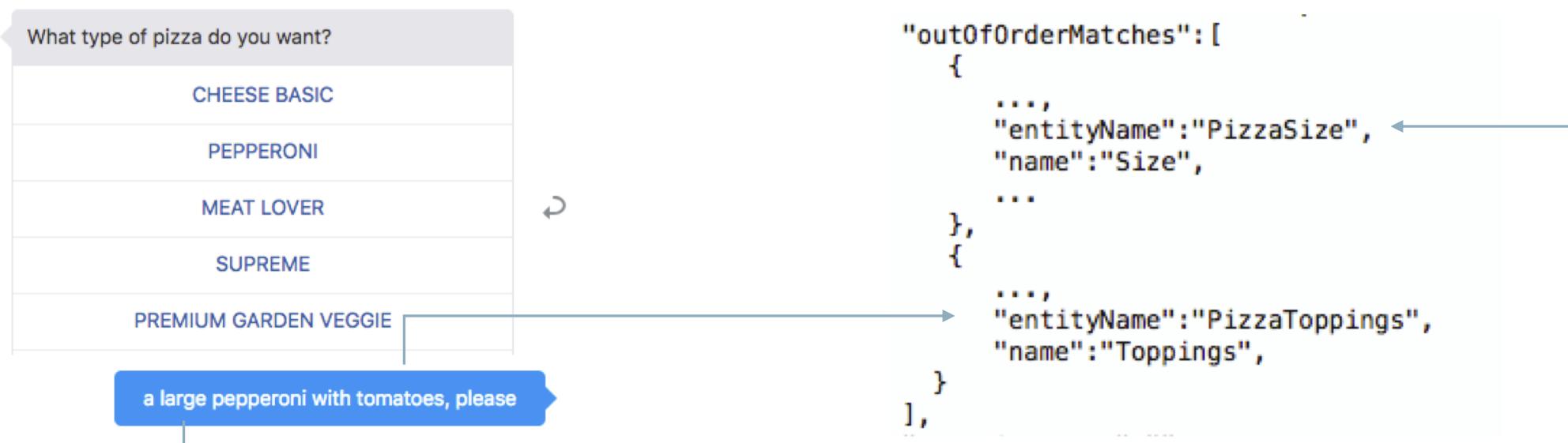
Accessing matched entities

- All updated entities from a user input
 - E.g. User provides more information than prompted for
 - Bot: "what pizza type do you like?"
 - User: "a large salami with tomatoes"
 - Updated entities: PizzaSize, PizzaType, PizzaToppings
- Expression to access matched entities
 - \${system.entityToResolve.value.allMatches?size}
 - \${system.entityToResolve.value.allMatches[n].entityName}
 - \${system.entityToResolve.value.allMatches[n].name}



Accessing out-of-order entity matches

- Updated entities from a user input for which there was no prompt
 - \${system.entityToResolve.value.outOfOrderMatches[n].entityName}
 - \${system.entityToResolve.value.outOfOrderMatches[n].name}
 - \${system.entityToResolve.value.outOfOrderMatches?has_content?then(...,...)}



Integrated Cloud Applications & Platform Services

ORACLE®