

ORACLE®

# Oracle Digital Assistant

## The Complete Training

### Advanced Common Response Component Topics

# 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 ➤ Component fundamentals
- 2 ➤ Component properties
- 3 ➤ Response type specific properties
- 4 ➤ Component actions
- 5 ➤ Calling skills



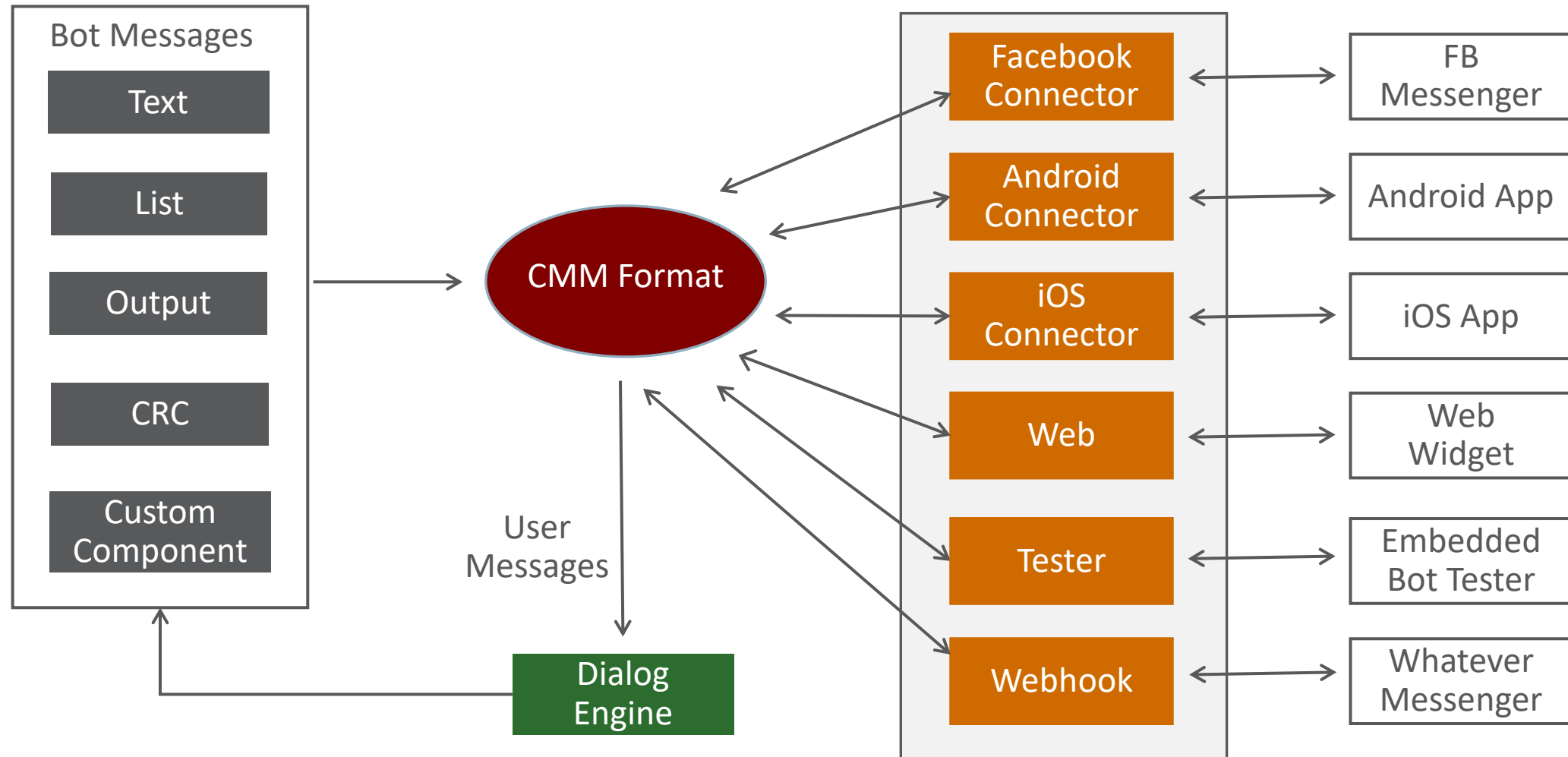
# Topic agenda

- 1 ➤ Component fundamentals
- 2 ➤ Component properties
- 3 ➤ Response type specific properties
- 4 ➤ Component actions
- 5 ➤ Calling skills

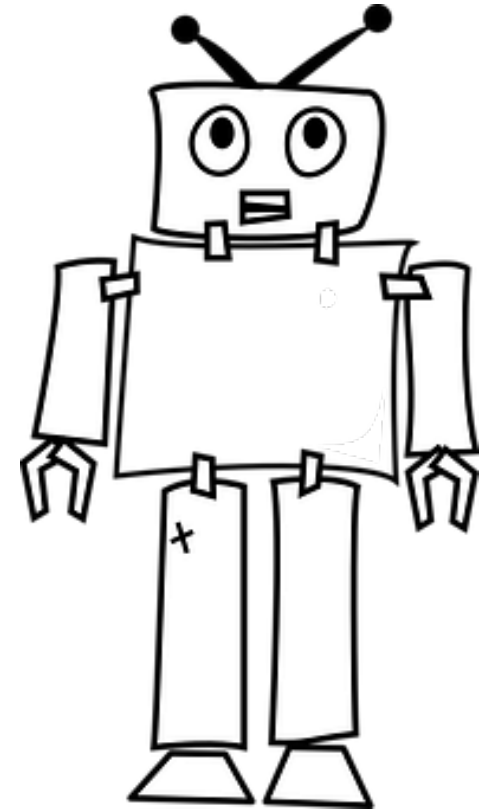
# About the conversation message model (CMM)

- Messengers are different
  - in the User Interface they can render for a Bot Response
  - in the format of the incoming and outgoing message payloads
- Oracle Digital Assistant uses a channel agnostic messaging format
  - Conversation Message Model (CMM)
  - Used by all UI components
  - Supported by the Custom Component SDK
  - CMM payload is automatically translated to payload format of target channel

# Oracle digital assistant CMM architecture



The **common response component** is the lid of a pot called the **conversation message model**



# About System.CommonResponse component

- Declarative design time for CMM skill responses
- Can be used for building any kind of user interface
- Most important component to know about
- Dialog flow builder provides quick-start component templates
  - Provided for text, card, attachment responses and use of composite bag entities
  - Not all component properties are part of the component templates



# Topic agenda

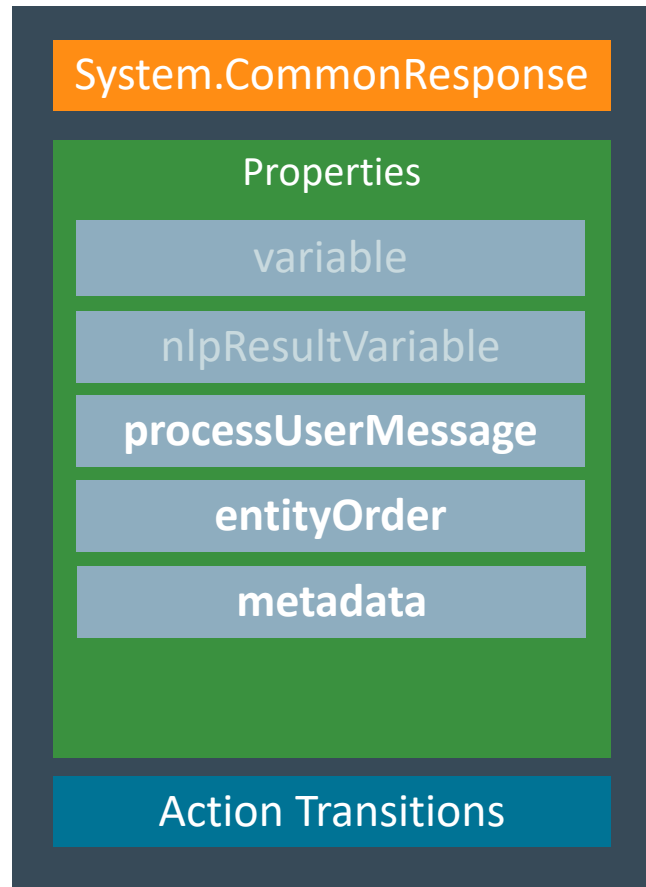
- 1 Component fundamentals
- 2 Component properties**
- 3 Response type specific properties
- 4 Component actions
- 5 Calling skills

# Entity slotting and component rendering



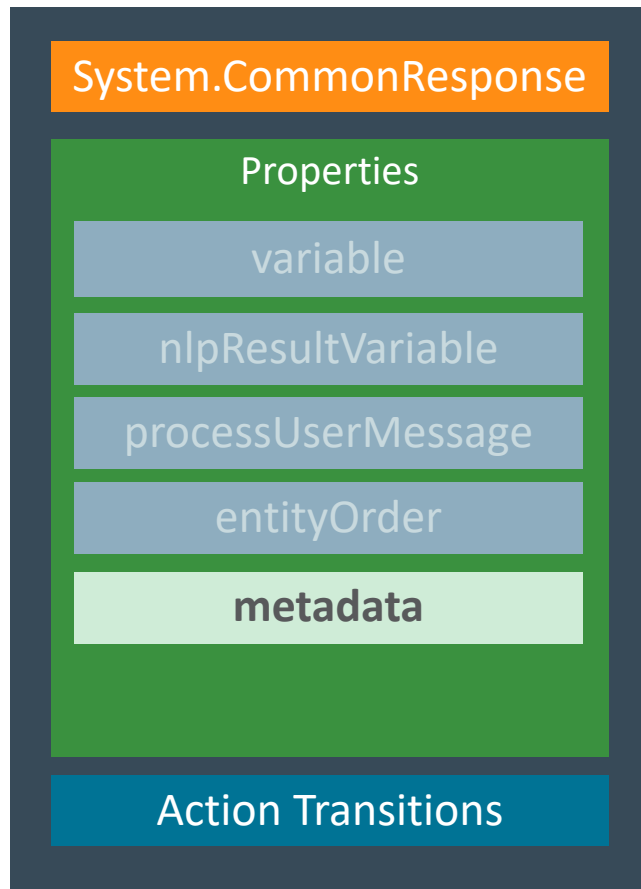
- If context variable referenced from *variable* property has value, then component is not rendered
- Referenced context variable is updated through user input in CR component or postback action
- If set, the *nlpResultVariable* property must reference a context variable of type "nlpresult".
  - For variables of type entity that are referenced from the *variable* property, attempts to extract values through NLP
  - If value is found in NLP then variable property is getting updated and component rendering is skipped

# Properties specific to System.CommonResponse



- CR component supports all properties common to system components
- Specific properties
  - Setting *processUserMessage* to false turns component into a read-only view
  - *entityOrder* property used with composite bag entities to override default order in which bag items are resolved
  - The *metadata* property defines the component response to display in the messenger

# Specific System.CommonResponse component properties



- **responseItem**

- type (text, card, attachment)
- visible

- Optional

- Conditionally renders response

- Apache FreeMarker expression returning true or false
- onInvalidUserInput
- channels
  - include (comma separated list of channels)
  - exclude (comma separated list of channels)
- entitiesToResolve
  - include
  - exclude

# Conditional visibility of items

```
14 selectAirport:
15   component: "System.CommonResponse"
16   properties:
17     ...
18   metadata:
19     responseItems:
20     - type: "text"
21       ...
22     actions:
23     - label: "${enumValue}"
24       type: "postback"
25       visible:
26       expression: "${system.invalidUserInput?boolean}"
27       payload:
28         variables:
29           airports: "${enumValue}"
30           iteratorVariable: "airports.type.enumValues"
31     globalActions:
32     - label: "Cancel"
33       type: "postback"
34       visible:
35       # expression: "${system.invalidUserInput?boolean}"
36       onInvalidUserInput: true
37       payload:
38         action: "exitComponentLoop"
39     - label: "Help"
40       type: "postback"
41       visible:
42       #expression: "${system.invalidUserInput?boolean}"
43       onInvalidUserInput: true
44       payload:
45         action: "help"
```

I like to book a flight

Please provide a code for the destination airport

huh?

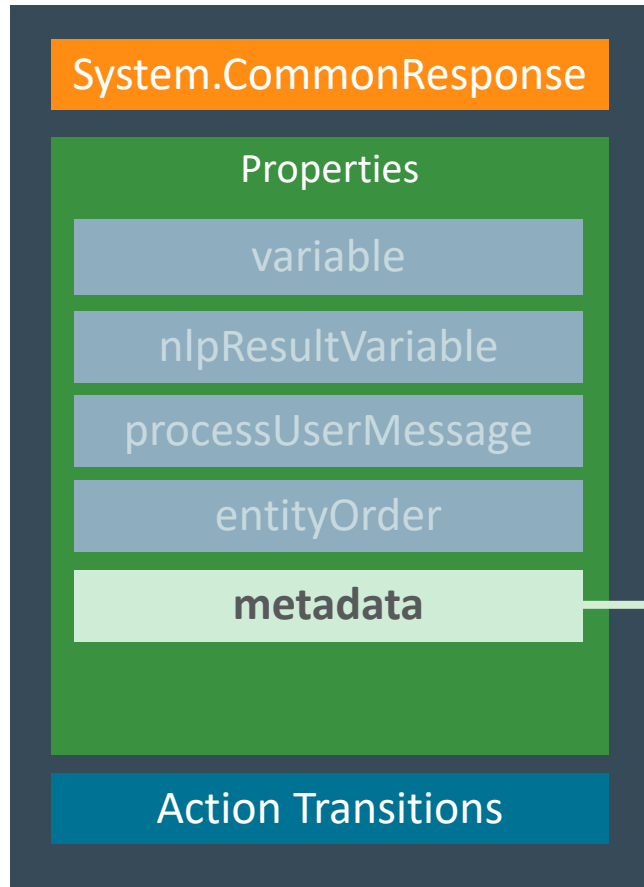
"huh?" is not a valid entry. Please select a departing airport.

MUC
LHR
CDG
SFO

Cancel Help



# Specific System.CommonResponse component properties

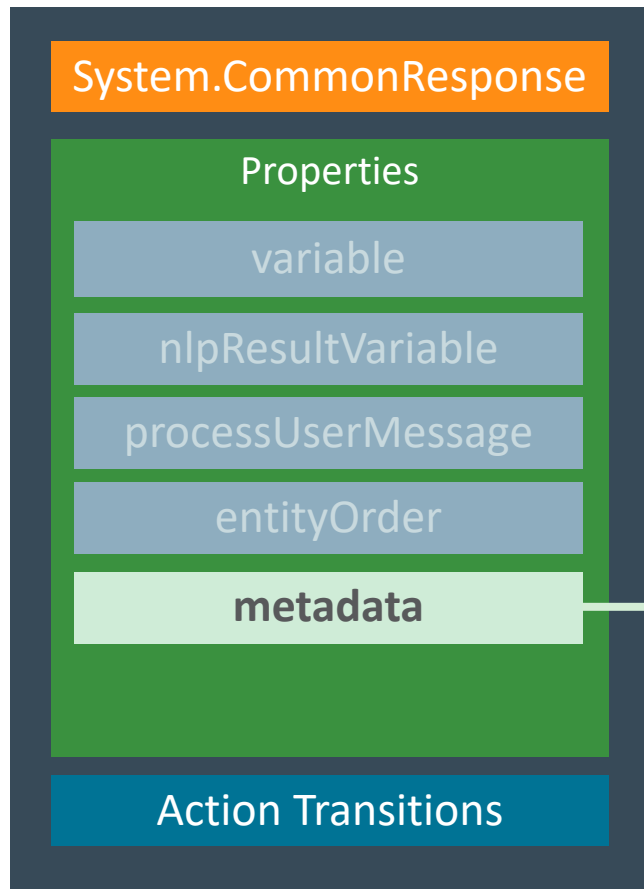


- **responseItem**

- type (text, card, attachment)
- visible
- iteratorVariable, rangeStart, rangeSize
- channelCustomProperties
- **actions**
  - type (postback, share, call, url, location)
  - Label, imageUrl
  - iteratorVariable
  - channelCustomProperties

- Properties added to the response "as is"
  - Set channel-specific functionality that cannot be set with component properties
- Can be used as a UI hints when building custom messengers
- Example
  - channelCustomProperties:
    - channel: "facebook"
  - properties:
    - top\_element\_style: "large"

# Specific System.CommonResponse component properties



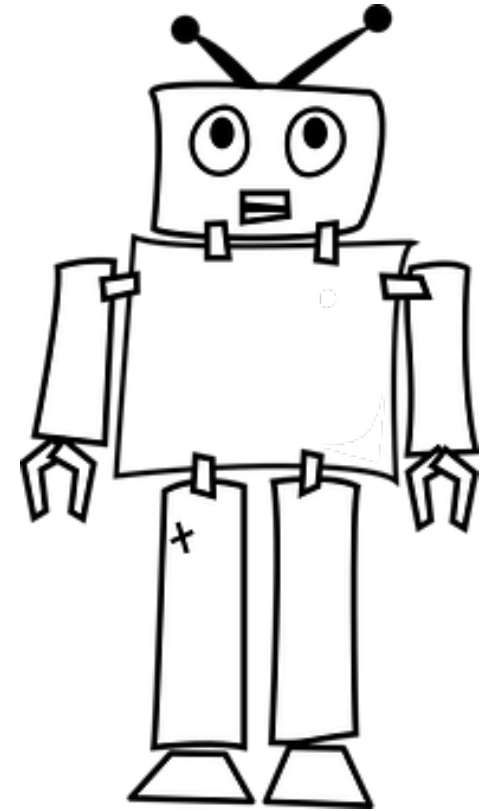
- **responseItem**

- type (text, card, attachment)
- visible
- iteratorVariable, rangeStart, rangeSize
- channelCustomProperties
- **actions**
  - type (postback, share, call, url, location)
  - Label, imageUrl
  - iteratorVariable
  - channelCustomProperties
  - payload

- Required for action types 'url', 'postback' and 'call'
- Allowed properties
  - *action* (optional) - custom string mapped in the actions transition
  - *variables* (optional) - one or many key-value pairs of context, system and user variable names and values
  - *url* (optional) website to open for URL type
  - *phoneNumber* (optional) used with call type

You can define **global actions** that are **displayed outside the response items** below the component.

In Facebook Messenger those are rendered as **quick replies**

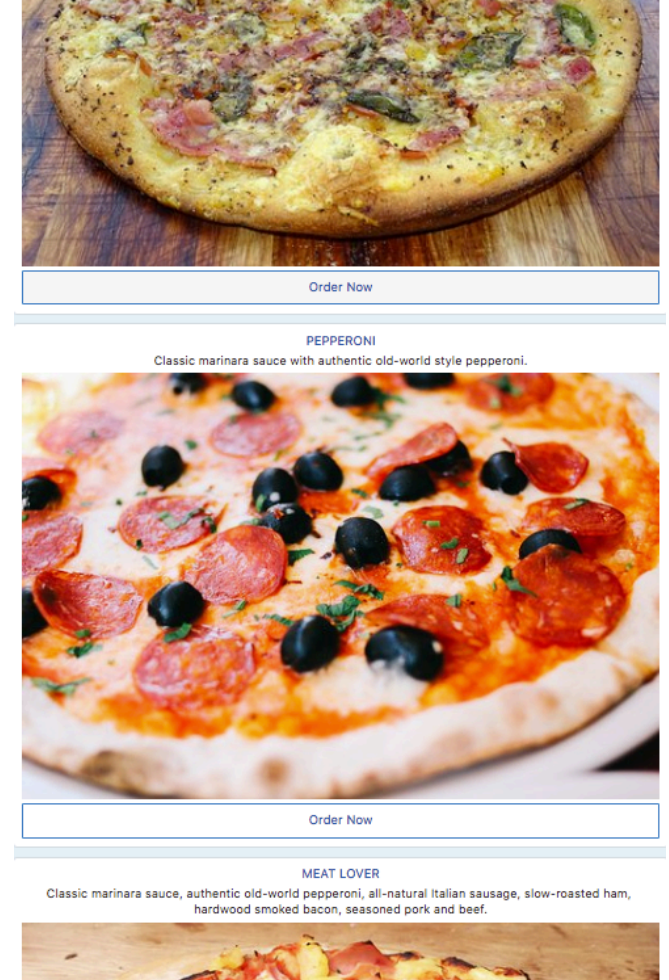


# Topic agenda

- 1 ➤ Component fundamentals
- 2 ➤ Component properties
- 3 ➤ Response type specific properties**
- 4 ➤ Component actions
- 5 ➤ Calling skills

# Cards response specific properties

- Cards (required)
  - Defines list of card responses
- cardLayout (optional)
  - Vertical (default), horizontal
- Supports page ranging to overcome channel specific card limits





# Card item specific properties (1 of 2)

- title (required)
  - title of the card, displayed as first line on the card
- description (optional)
  - Description of the card (displayed as second line)
- imageUrl (optional)
  - URL of the image to be displayed on card
- cardUrl (optional)
  - URL of site displayed opened when tapping the card
- actions (optional)
  - List of actions rendered with each card



## Card item specific properties (2 of 2)

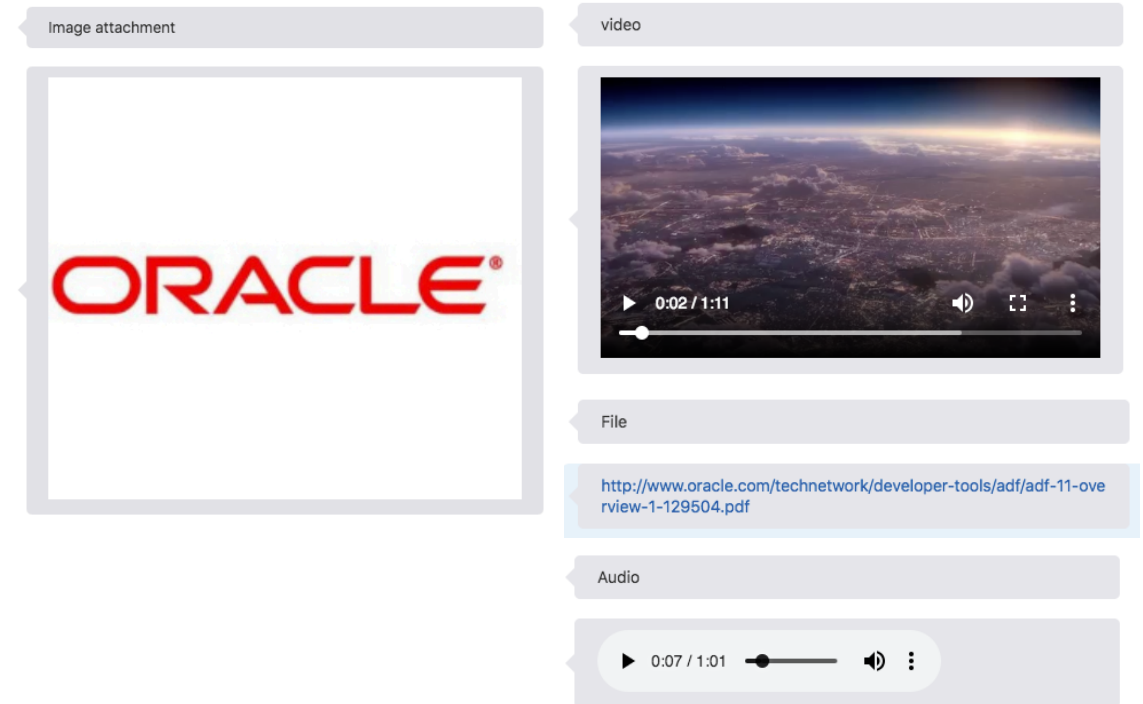
- visible (optional)
- iteratorVariable (optional)
  - Stamps cards by iterating over items stored in an array
- rangeStart / rangeSize (optional)
  - Used with iteratorVariable set
  - Stamp a subset of cards
- channelCustomProperties (optional)
  - Channel-specific properties that cannot be controlled with the standard properties

# Text response item specific properties

- text (required)
  - The message text
  - Supports Apache FreeMarker expressions
- separateBubbles (optional)
  - Boolean property
  - Applicable when the iteratorVariable property is set
    - If set to true, each text item is sent as separate message
    - If set to false, one text message is sent with each text item starting at a new line
- actions (optional)
  - List of items that are rendered with the text response. Used e.g. to render select lists

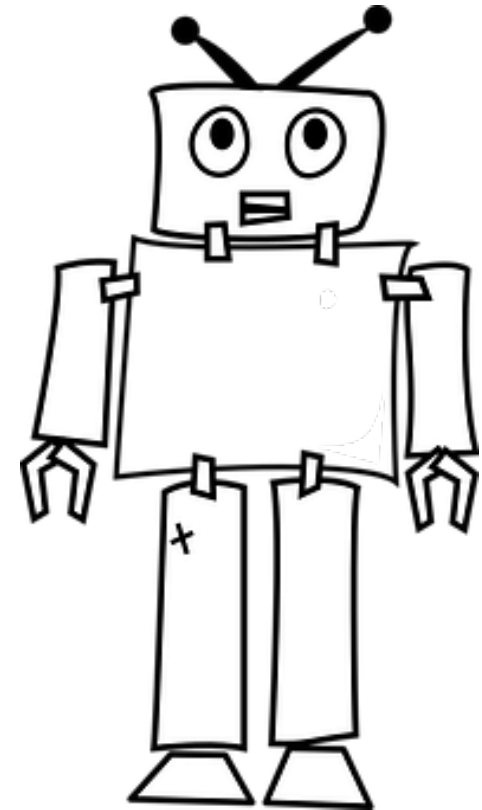
# Attachment response item specific properties

- attachmentType (required)
  - image
  - audio
  - video
  - file
- attachmentUrl (required)
  - Attachment download URL
- Attachments may or may not render inside messenger



The **number of actions** and cards **that can be displayed** for a bot response is **channel dependent**.

Use pagination when not all card or action items can be rendered

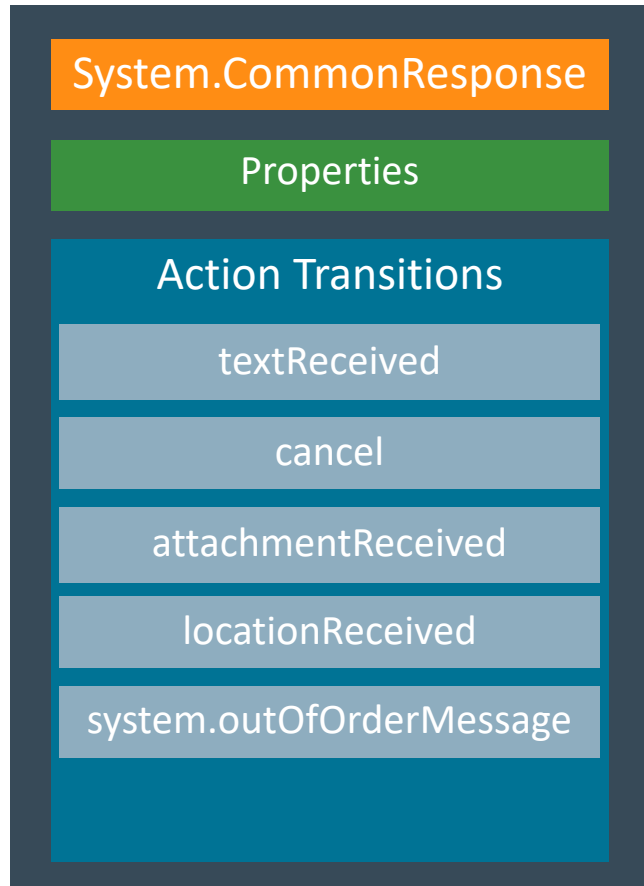




# Topic agenda

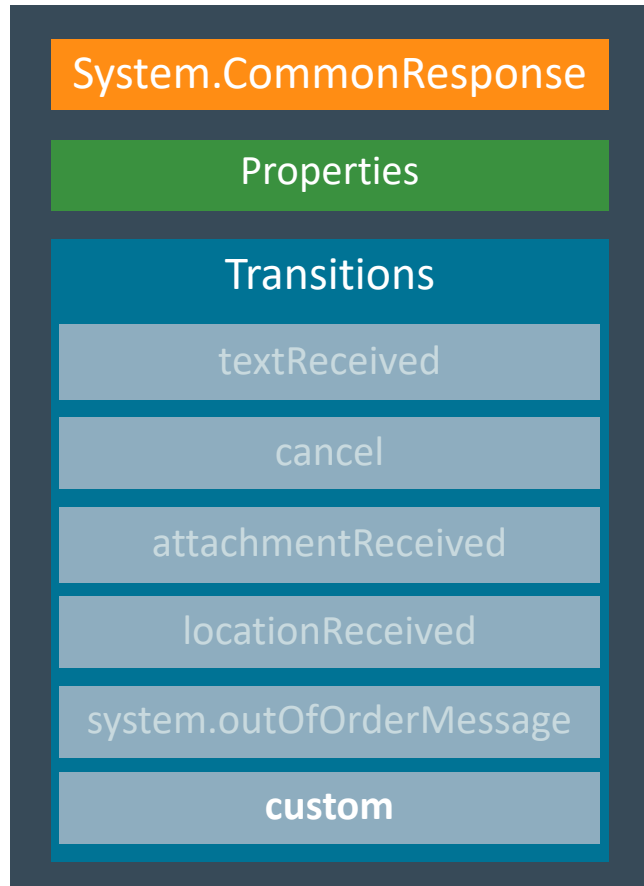
- 1 ➤ Component fundamentals
- 2 ➤ Component properties
- 3 ➤ Response type specific properties
- 4 ➤ Component actions**
- 5 ➤ Calling skills

# System.CommonResponse component action transitions



- `textReceived`
  - Set when user enters free text or sends an emoji
- `cancel`
  - When data input attempts exceed `maxPrompts` value
- `attachmentReceived`
  - When user sends an image, audio, video or file attachment
- `locationReceived`
  - Followed when user sends geo location
- `System.outOfOrderMessage`
  - Out-of-order message handling

# System.CommonResponse component action transitions



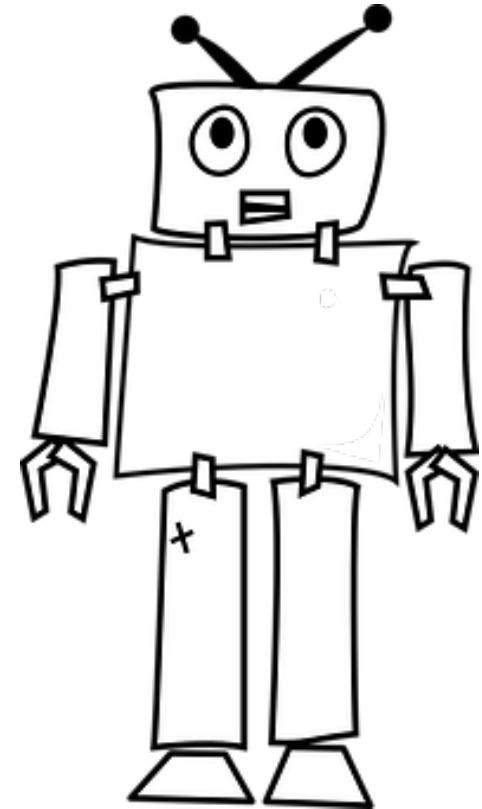
- Custom actions set as part of the action payload

```
actions:  
  - label: "Postback action"  
    type: "postback"  
    payload:  
      action: "myCustomAction"  
    variables:  
      orderId: "${orders.id}"
```

- Action name is mapped in actions transition on state

```
transitions:  
  actions:  
    myCustomAction : "stateHandlingTheAction"
```

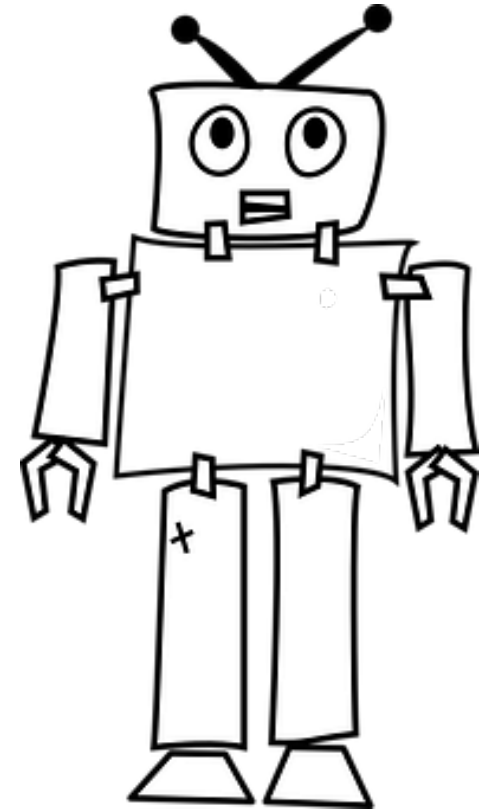
Always define a "next" transition for  
a Common Response component  
state because **not every component  
interaction results in an action  
transition**



# Topic agenda

- 1 ➤ Component fundamentals
- 2 ➤ Component properties
- 3 ➤ Response type specific properties
- 4 ➤ Component actions
- 5 ➤ Calling skills

You can use the Common Response component to trigger navigation to other skills in Oracle Digital Assistant



# Navigating to other skill

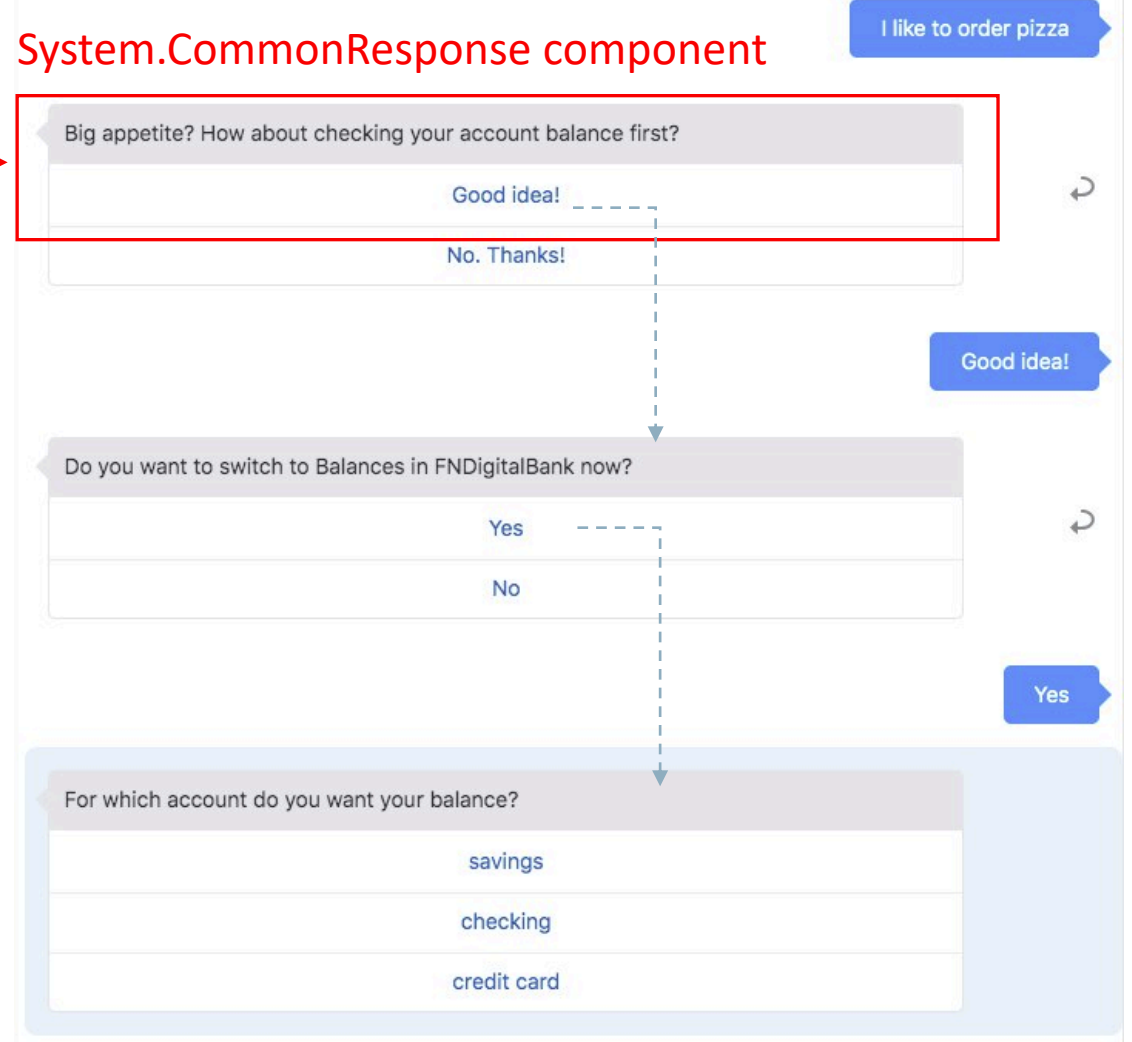
- Use a list or card layout
  - Set the postback > payload > action string to "system.textReceived"
  - Set postback > payload > variables variable name to "system.text"
  - Set variable value to message that digital assistant routes to a specific skill
- Use explicit routing
  - Improves routing confidence

```
callSkill:
  component: "System.CommonResponse"
  properties:
    processUserMessage: true
  metadata:
    responseItems:
      - type: "text"
        text: "prompt text"
        actions:
          - label: "label string"
            type: "postback"
            payload:
              action: "system.textReceived"
              variables:
                system.text: "ask <skill name> message_here"
```

# Calling a skill from System.CommonResponse component

```
OrderPizza:
  component: "System.CommonResponse"
  properties:
    processUserMessage: true
  metadata:
    responseItems:
      - type: "text"
        text: "Big appetite? How about checking your account bal
        actions:
          - label: "Good idea!"
            type: "postback"
            payload:
              action: "system.textReceived"
              variables:
                system.text: "Ask FNDigitalBank how much money do
                label: "No. Thanks!"
                type: "postback"
                keyword:
                payload:
                  action: "order"
  transitions:
    next: "StartOrderPizza"
    actions:
      order: "StartOrderPizza"
```

## System.CommonResponse component





# Avoid dependencies!

- Skill invocation names may be changed in digital assistant
- Make invocation message configurable using skill parameters
  - Skill parameters with a "da." prefix are exposed in digital assistant

**Create Parameter**

\* Name: da.invocationMessage

\* Display Name: Invocation Message

Type: String

\* Value: ask <invocation name> <message>

Description: Replace <invocation name> with name of skill in digital assistant and <message> with an utterance you know will resolve an intent in the skill

Create

actions:

- label: "label string"

type: "postback"

payload:

action: "system.textReceived"

variables:

system.text: "\${system.config.da.invocationMessage}"

# Best Practices

- **Avoid inter-skill dependencies**

Skills should be developed as modules that don't know of each other and that make no assumption about the names of variables or other skills in the same digital assistant. To avoid inter-skill dependencies, use custom skill parameters that use the "da." prefix.

- **Use explicit routing when calling skills from a skill**

Explicit routing provides better results! So when implementing calls to other skills in a skill, ensure you use explicit routing.

# Integrated Cloud

## Applications & Platform Services

ORACLE®