
ORACLE®

Oracle Intelligent Bots Advanced Training

Common Response 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. Conversation Message Model
2. Common Response Component
3. Examples
4. Composite Bag Entity Support

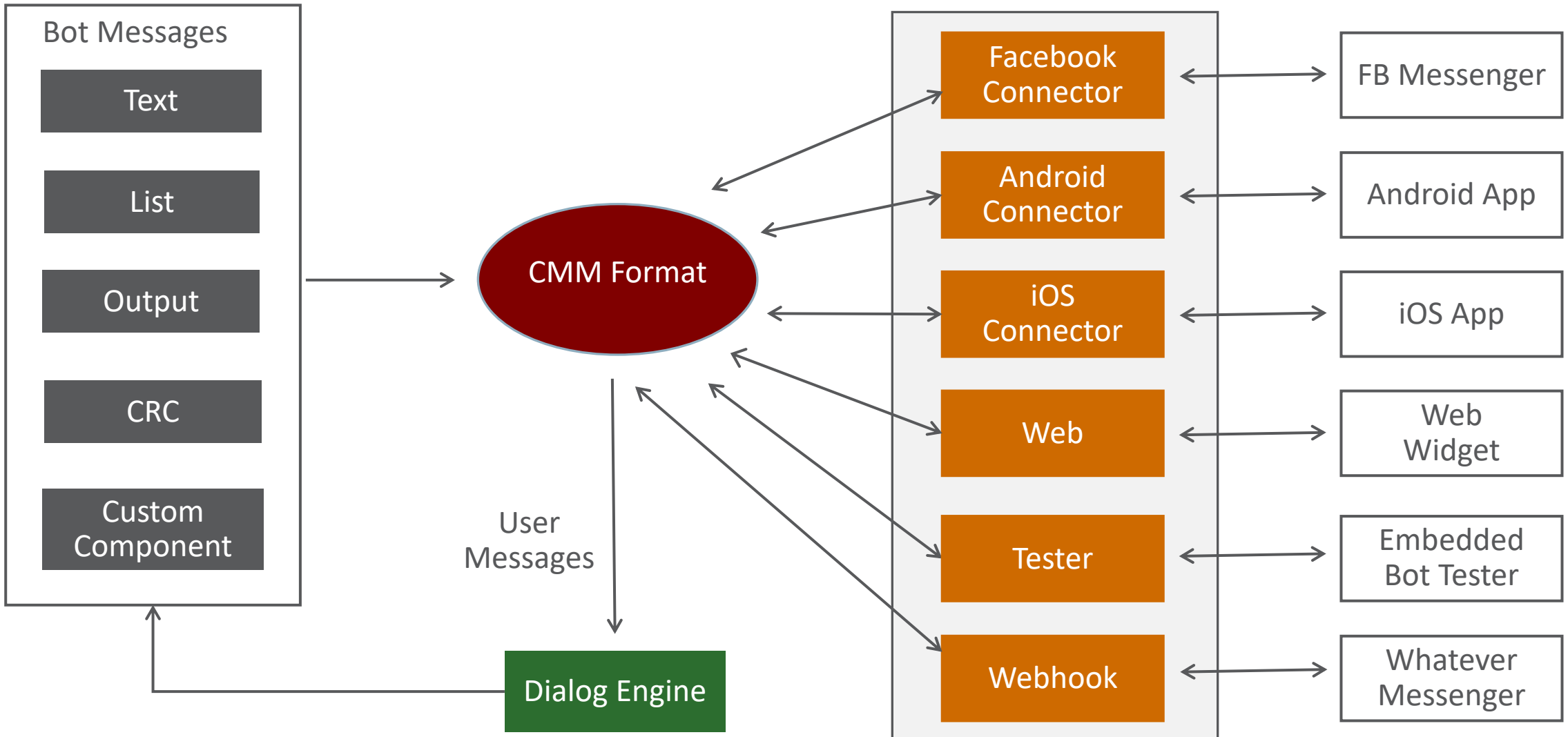
Conversation Message Model

All messengers are different in the layouts and the message structure they support.

The Conversation Message Model uses a **channel agnostic message structure** that abstracts component developers from channel specific code.

Image courtesy of Farbentek at FreeDigitalPhotos.net

Intelligent Bots CMM Architecture



Common Response Component

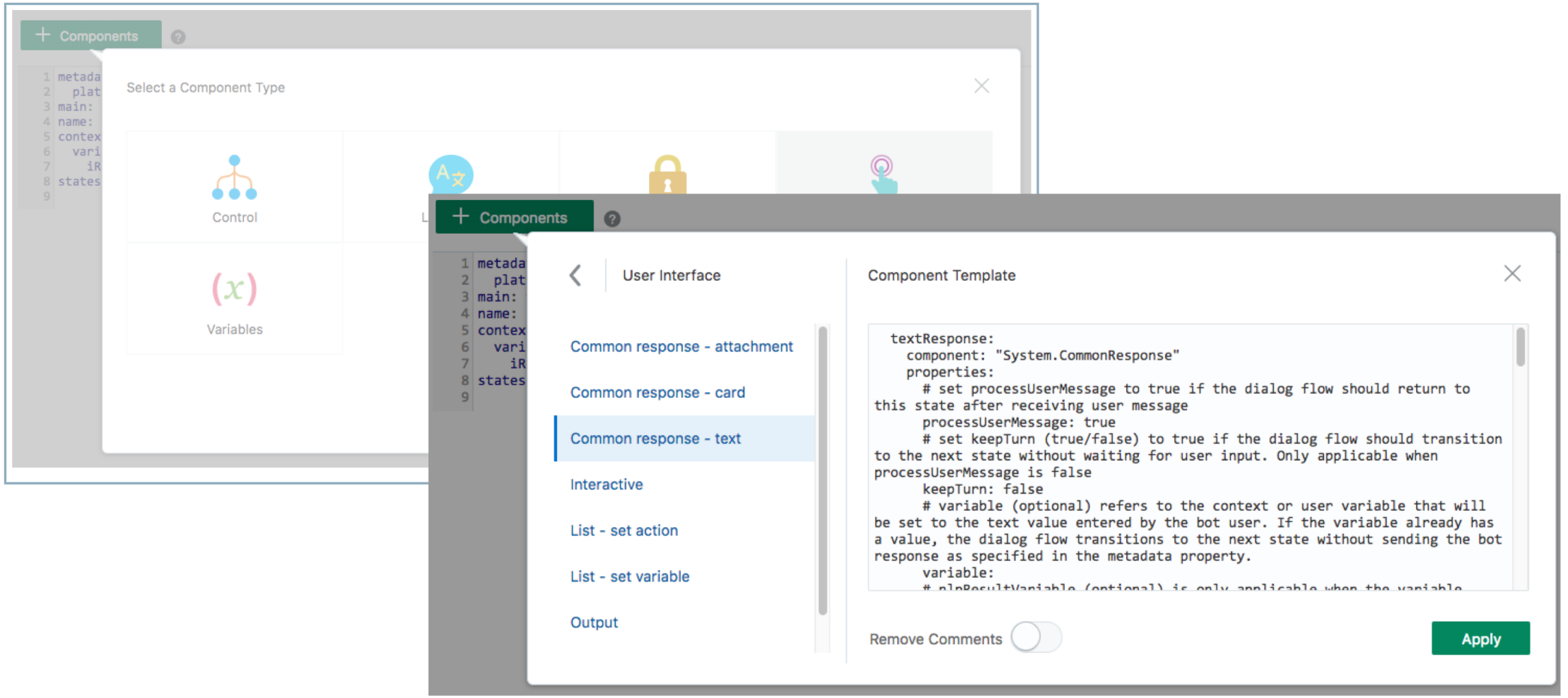
Introduction

The CR component is a **declarative design-time for CMM**.

The CR component allows you to define **arbitrary complex composite bot responses.**

Image courtesy of Farbentek at FreeDigitalPhotos.net

How-to Add the CR Component to a Conversation



MVC Pattern with CR Component

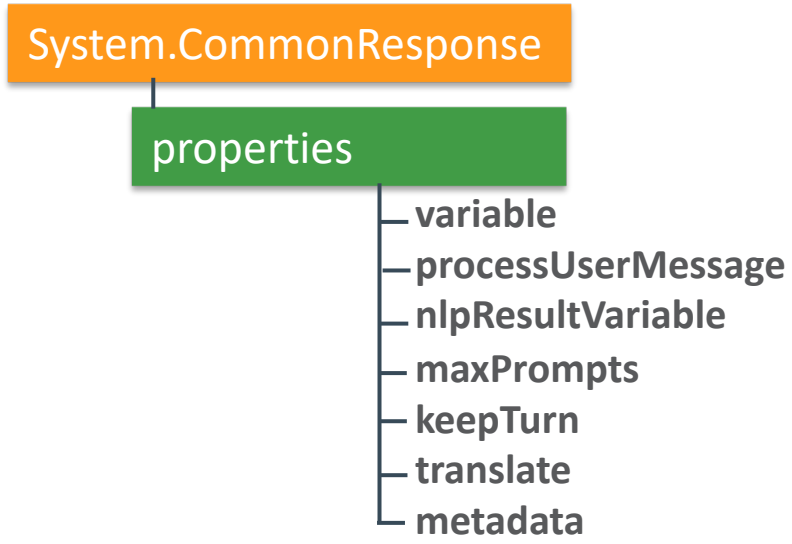
Common Response Component

Component Structure

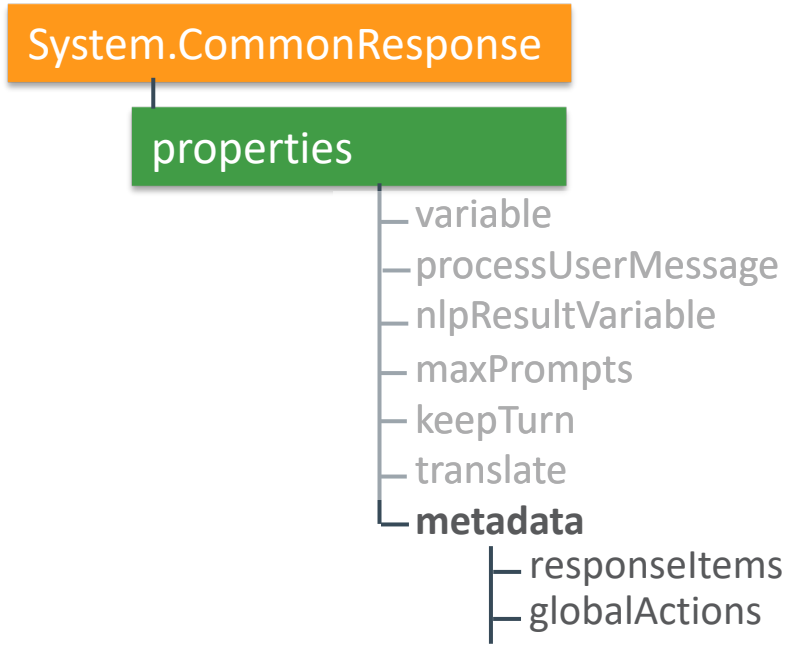
Component Properties

Like any other system component the **CR component exposes properties** for bot designers **to define its behavior.**

Component Properties

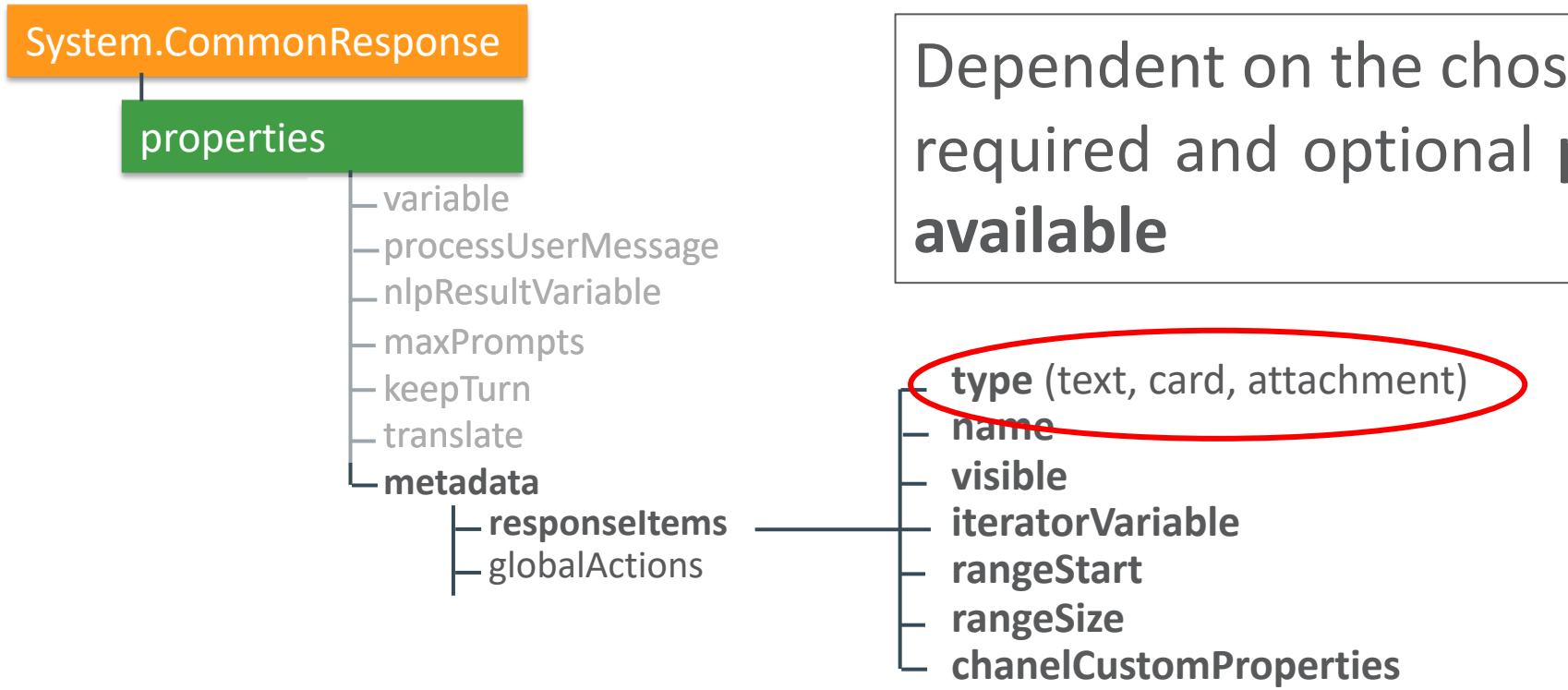


Component Metadata Property



The **metadata property is mandatory** and renders the component response. It's the design time for the CMM model.

Component Metadata Property



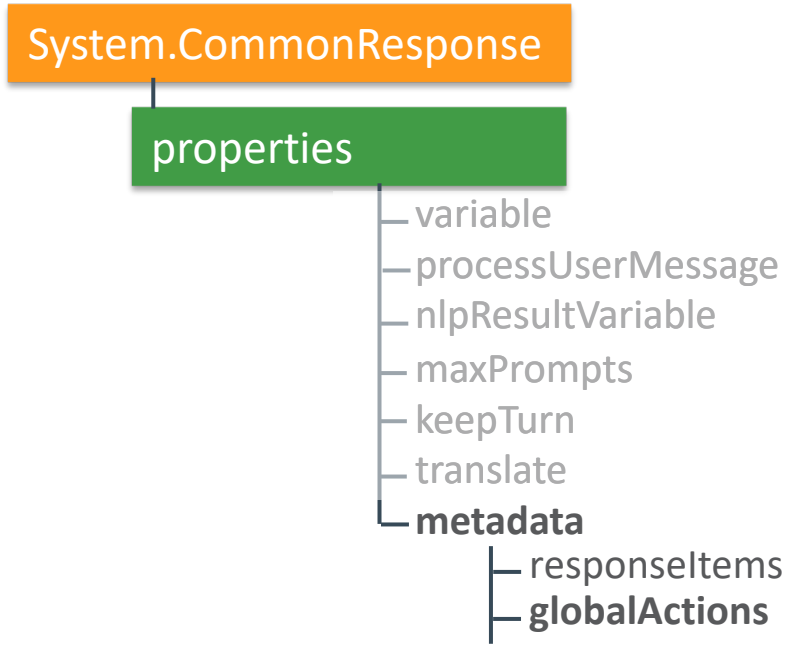
Dependent on the chosen type, **additional** required and optional **properties become available**

"visible" Property

- type (text, card, attachment)
- name
- **visible**
- iteratorVariable
- rangeStart
- rangeSize
- chanelCustomProperties

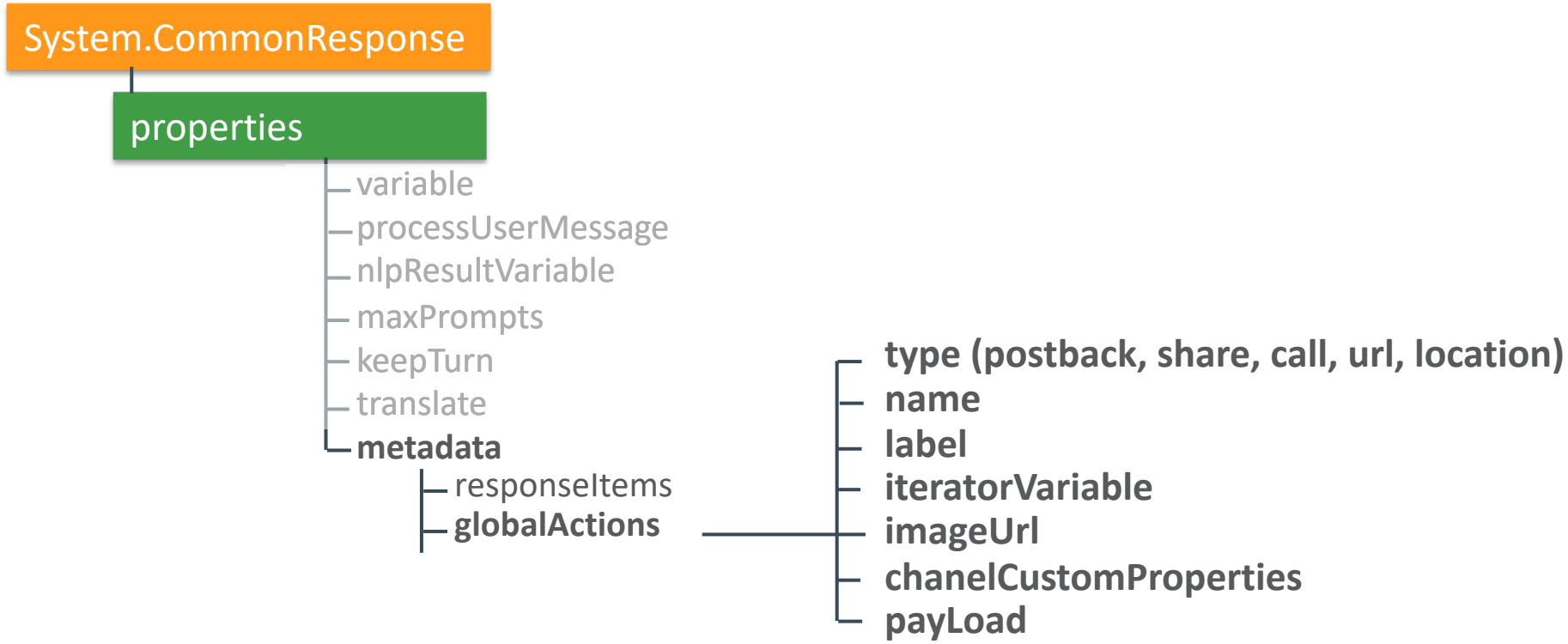
- Optional
- Conditionally renders response item
- Requires nested property values
  - expression
    - Apache Freemarker expressions returning true or false
  - onInvalidUserInput
  - channels
    - include (comma separated list of channels)
    - exclude (comma separated list of channels)
  - entitiesToResolve
    - include
    - exclude

Component Metadata Property

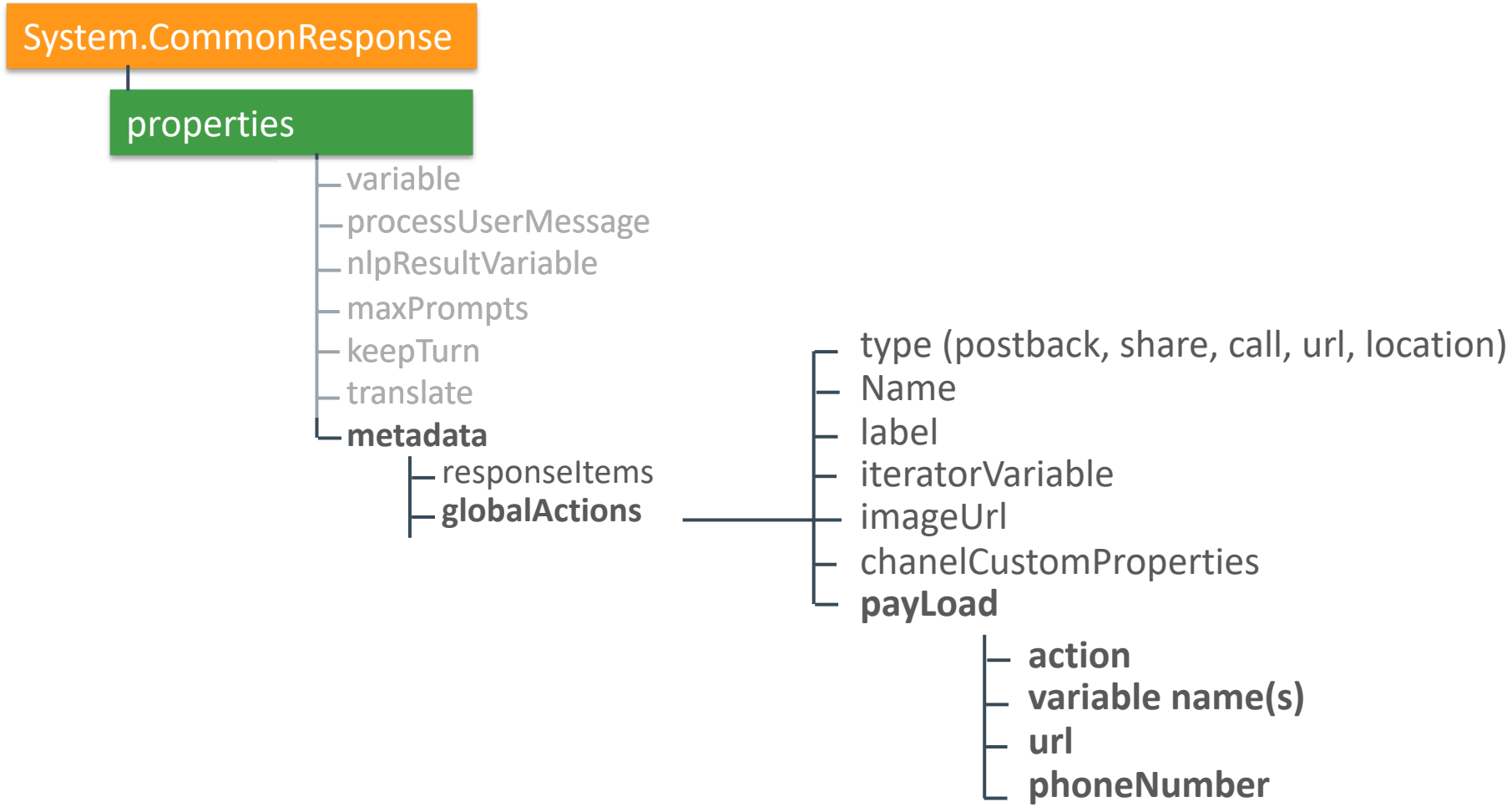


List of actions **not related to a specific response item**. Typically displayed as buttons at the bottom of the chat window, e.g. Quick Replies in Facebook Messenger

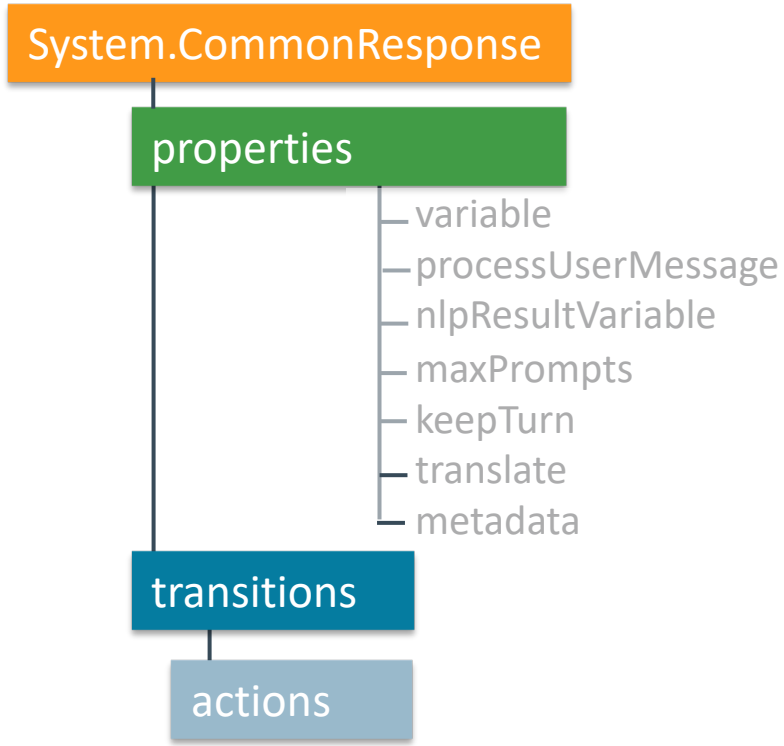
Component Metadata Property



Component Metadata Property

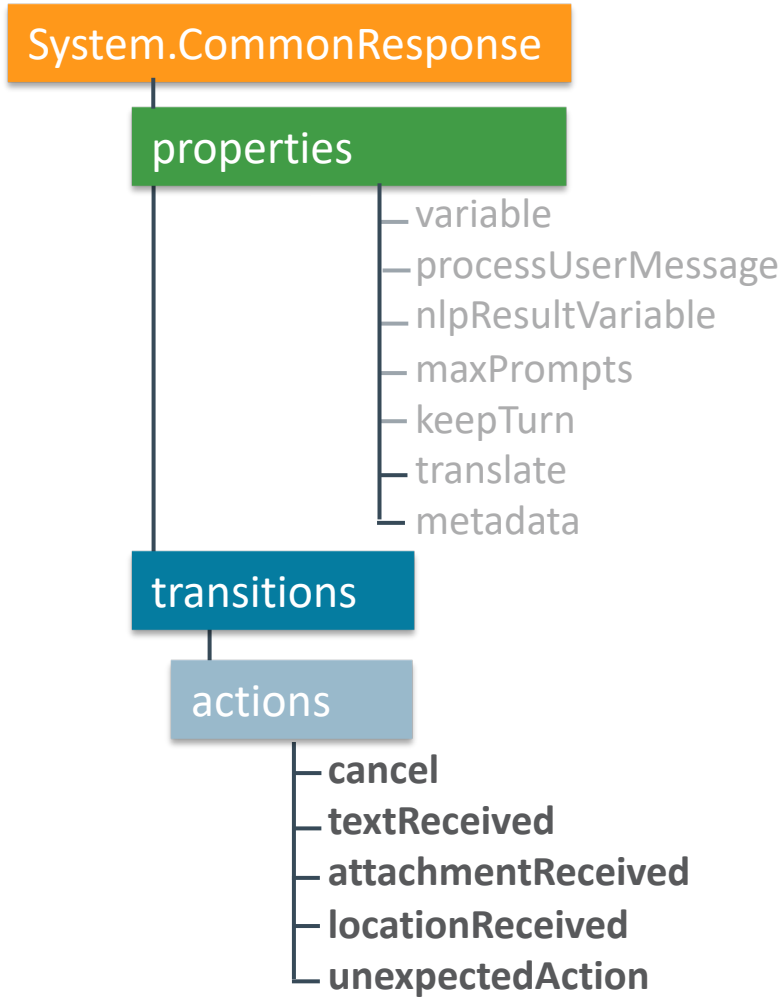


Component Transition Actions



Like other system components the **CR component exposes transition actions** for bot designers **to define conditional navigation to a next state** in the conversation.

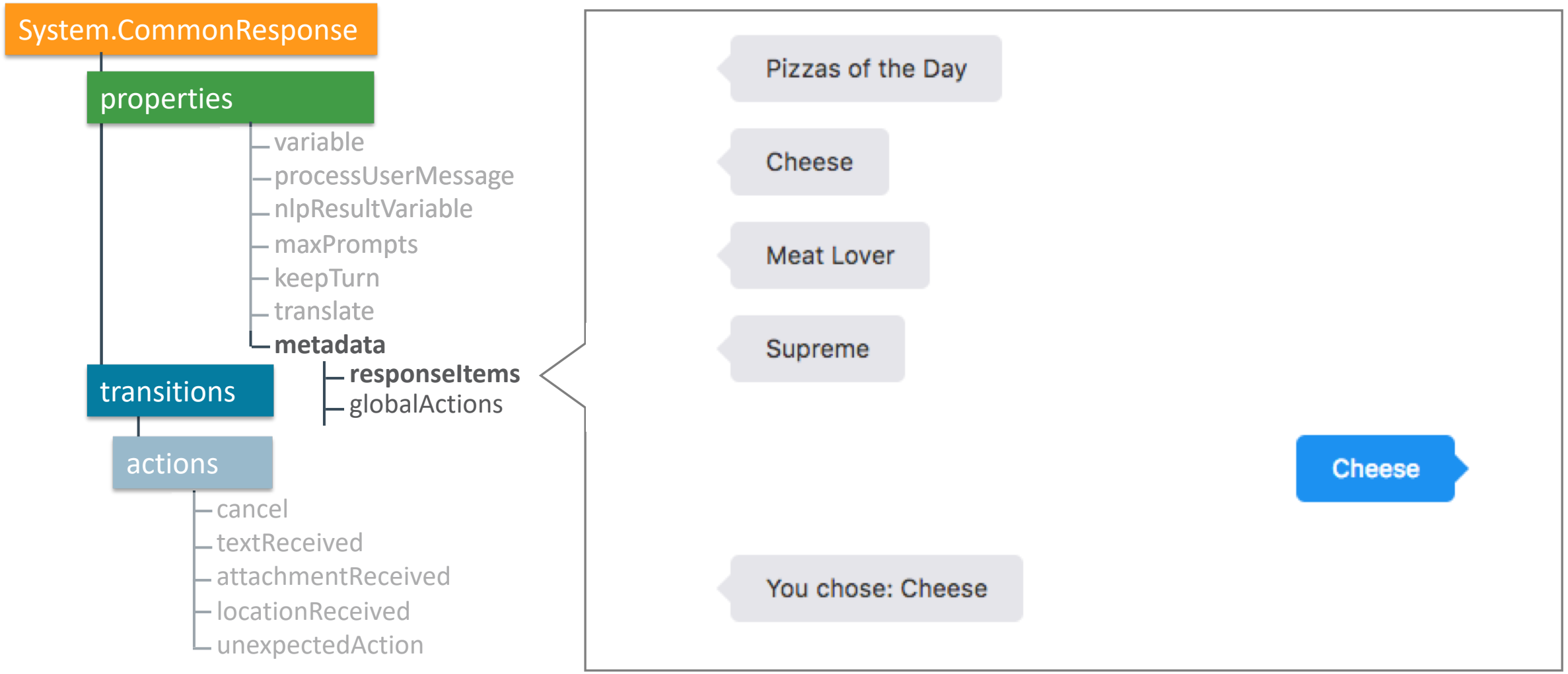
Component Transition Actions



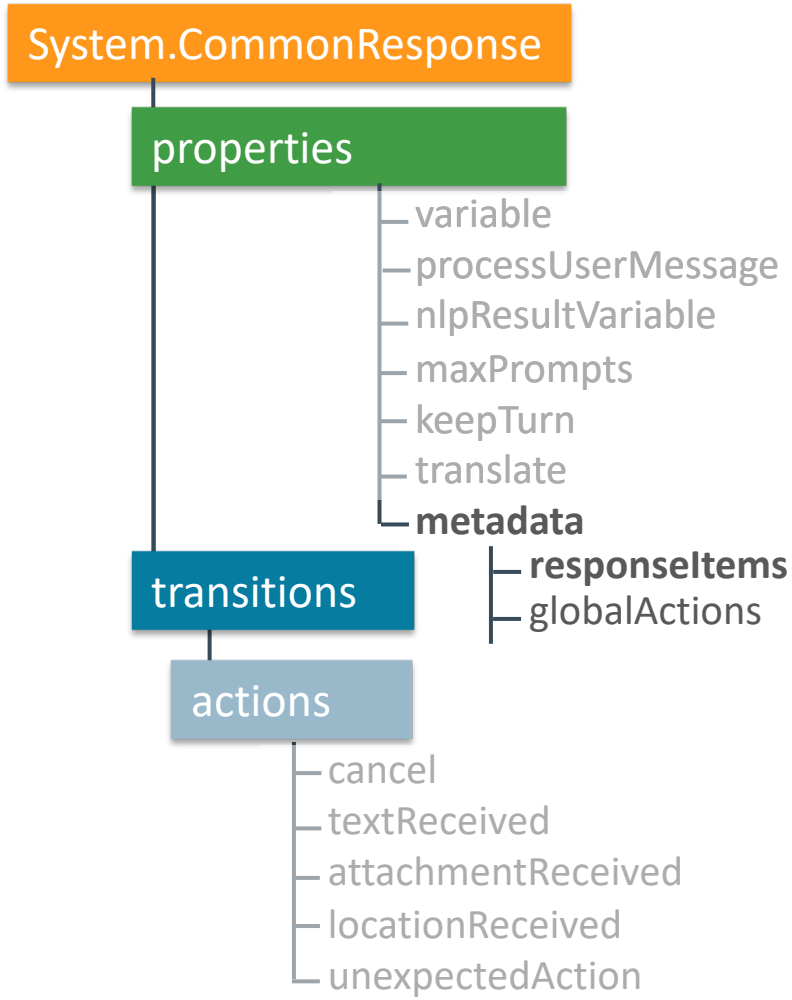
Common Response Component

Text Example

CRC Text Response Type

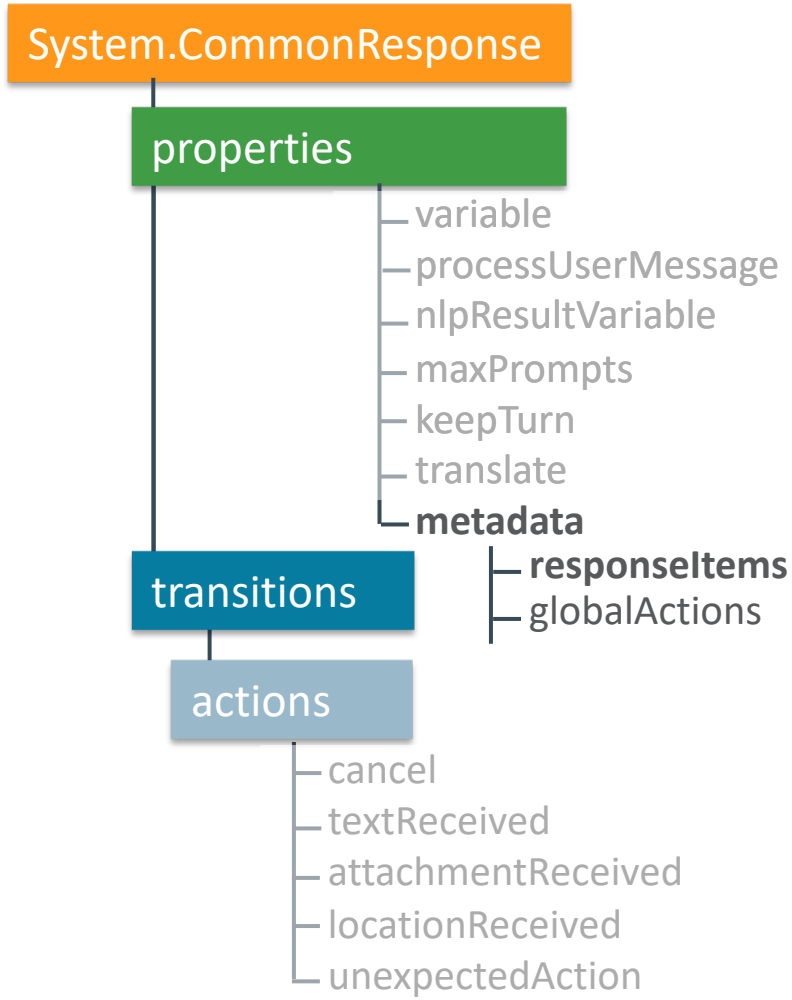


CRC Text Response Type – Static Data



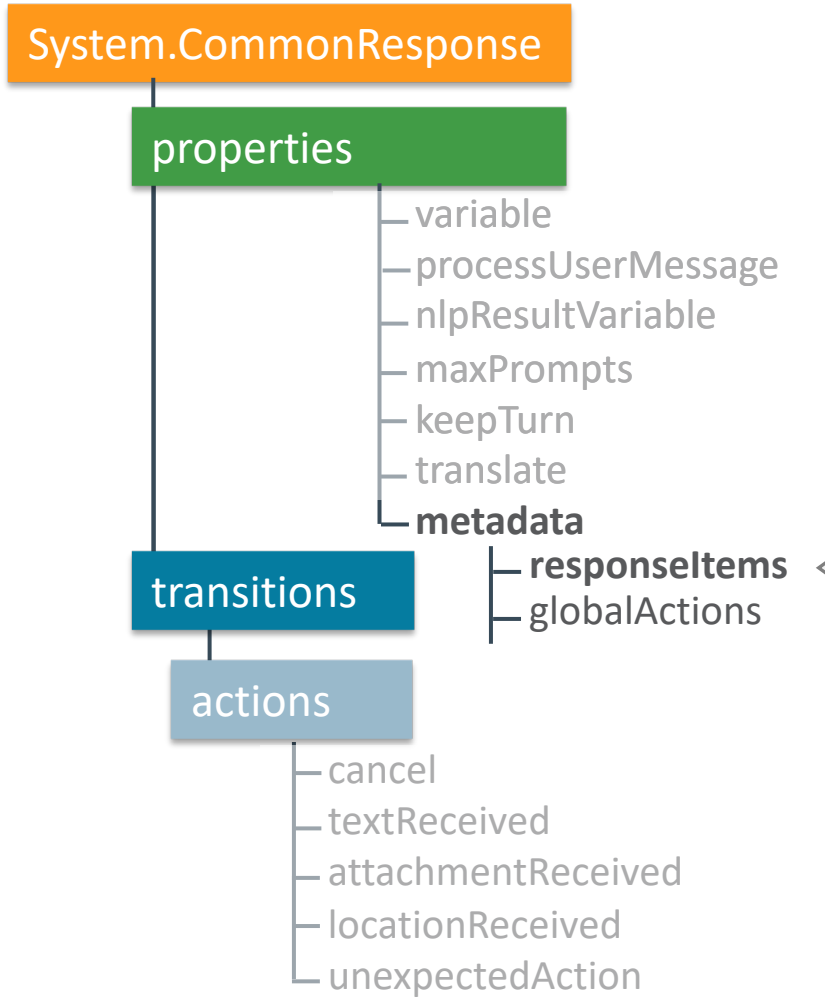

```
pizzasOfTheDayState:
  component: "System.CommonResponse"
  properties:
    processUserMessage: true
    variable: "selectedPizza"
metadata:
    responseItems:
    - type: "text"
      text: "Pizzas of the Day"
    - type: "text"
      text: "Cheese"
    - type: "text"
      text: "Meat Lover"
    - type: "text"
      text: "Supreme"
  transitions: {}
```

CRC Text Response Type – Data Iterator




```
displayPizzas:
  component: "System.CommonResponse"
  properties:
    processUserMessage: true
    variable: "selectedPizza"
    metadata:
      responseItems:
      - type: "text"
        text: "Pizzas of the Day"
      - type: "text"
        text: "${pizzas.name}"
        iteratorVariable: "pizzas"
  transitions: {}
```

CRC Text Response Type – Iterate Data in Entity



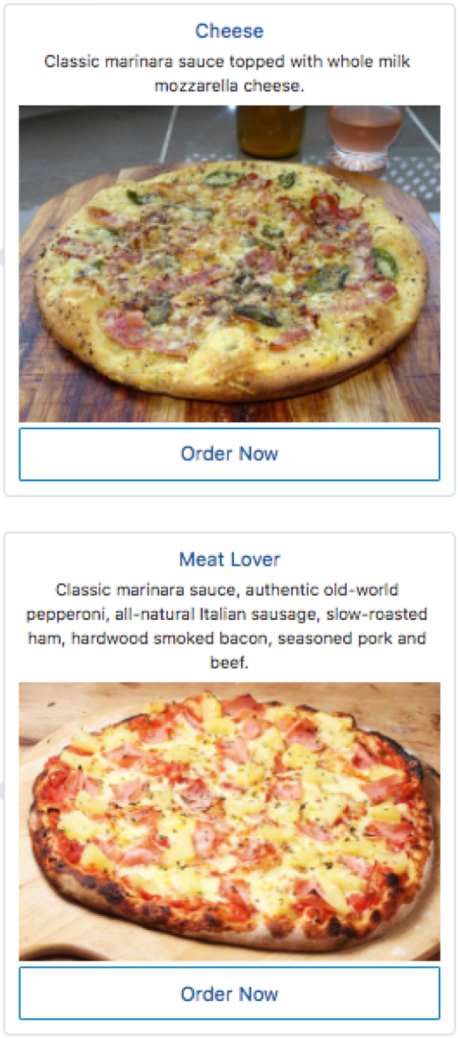

```
variables:
 pizzas: "PIZZA"
...
displayPizzas:
   component: "System.CommonResponse"
   properties:
    processUserMessage: true
    variable: "selectedPizza"
    metadata:
     responseItems:
     - type: "text"
       text: "Pizzas of the Day"
     - type: "text"
       text: "${enumValue}"
       iteratorVariable: "pizzas.type.enumValues"
   transitions: {}
```

Common Response Component

Cards Example

CRC Cards Response Type - Attributes

- cards
  - Holds a list of cards
  - A card can be defined in BotML or stamped from a data array
- cardLayout
  - vertical
  - horizontal
- actions
  - Actions displayed below list of cards
  - postback, share, call, url, location



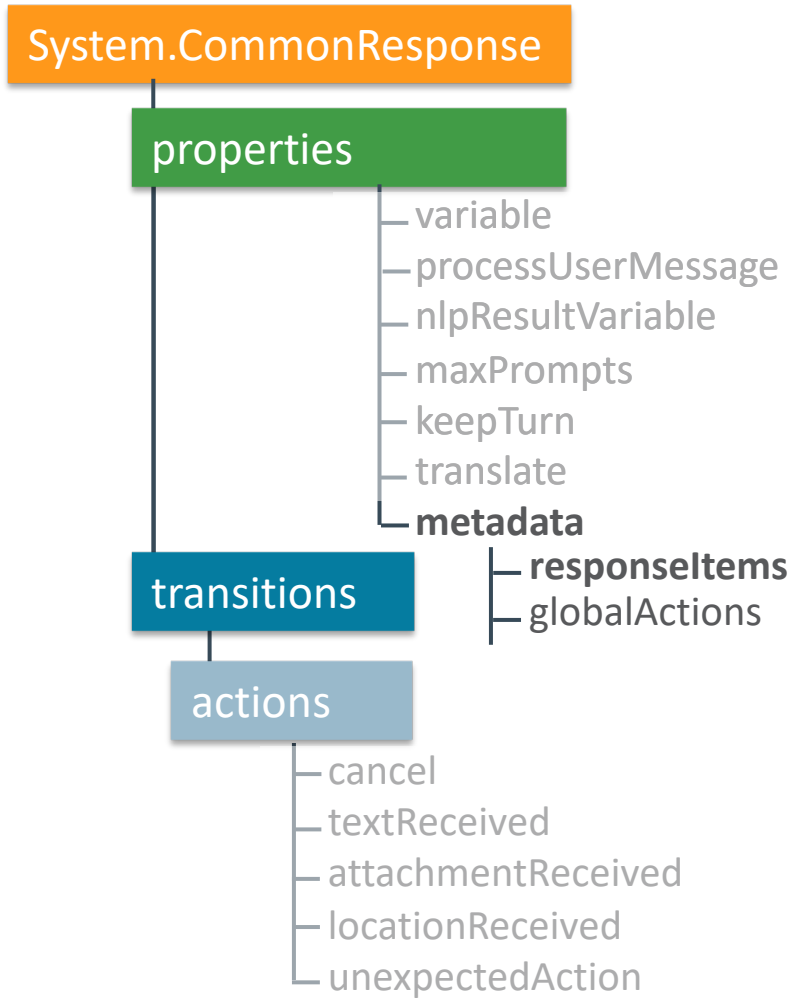
CRC Cards Card - Attributes

- name
  - Used in BotML to identify cards
- title
  - Required attribute that defines the first text line in the card
- description
  - Optional second line of the card
- imageUrl
  - Allows an image to be rendered below the second text line
- cardURL
  - Optional link to a website that will be opened when a user clicks on the link

CRC Cards Card - Attributes

- actions
  - Optional list of actions associated with a card
  - postback, share, call, url, location
- iteratorVariable
  - Stamps cards based on content in a data array
- rangeStart / rangeSize
  - Pagination when "iteratorVariable" property set
- channelCustomProperties

CRC Card Response Type




```
displayMenu:
  component: "System.CommonResponse"
  properties:
    metadata:
      responseItems:
      - type: "cards"
        cardLayout: "vertical"
        cards:
        - title: "${pizzas.name}"
          description: "${pizzas.description}"
          imageUrl: "${pizzas.image}"
            actions:
            - ...
        iteratorVariable: "pizzas"
    processUserMessage: true
  transitions:
    ...
```

Common Response Component

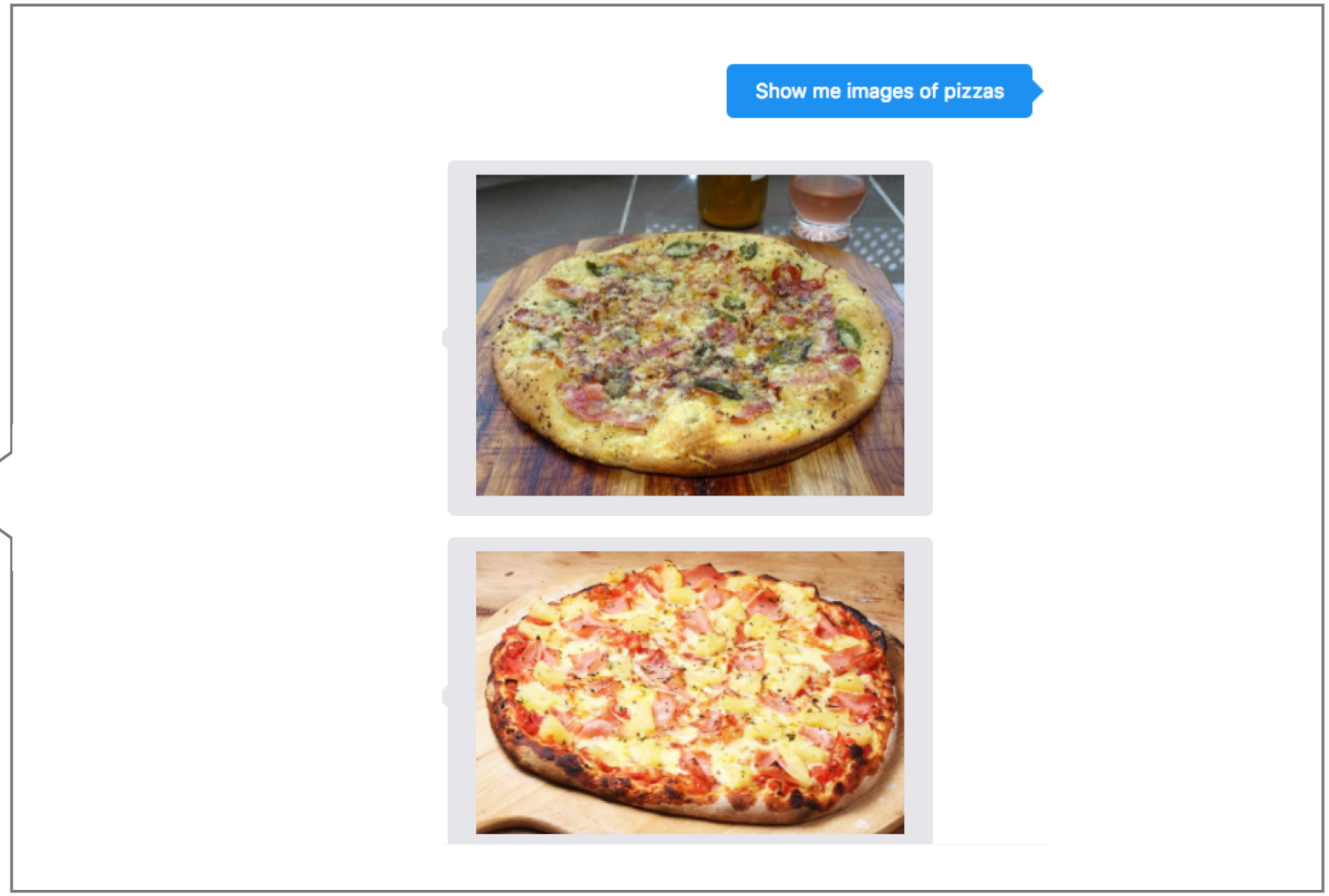
Attachment Example

CRC Attachment Response Type - Attributes

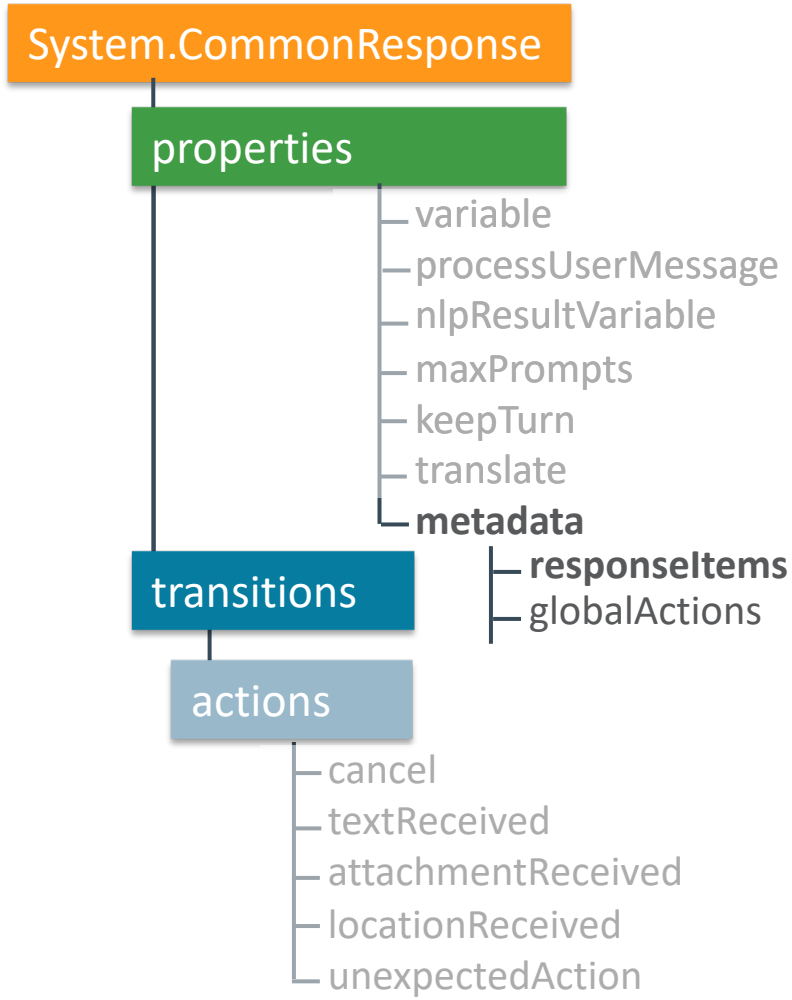
- attachmentType
  - image, audio,video, file
- attachmentUrl
  - The URL to download the attachment

CRC Attachment Response Type

- System.CommonResponse
  - properties
    - variable
    - processUserMessage
    - nlpResultVariable
    - maxPrompts
    - keepTurn
    - translate
    - **metadata**
      - **responseItems**
      - globalActions
  - transitions
  - actions
    - cancel
    - textReceived
    - attachmentReceived
    - locationReceived
    - unexpectedAction



CRC Attachment Response Type



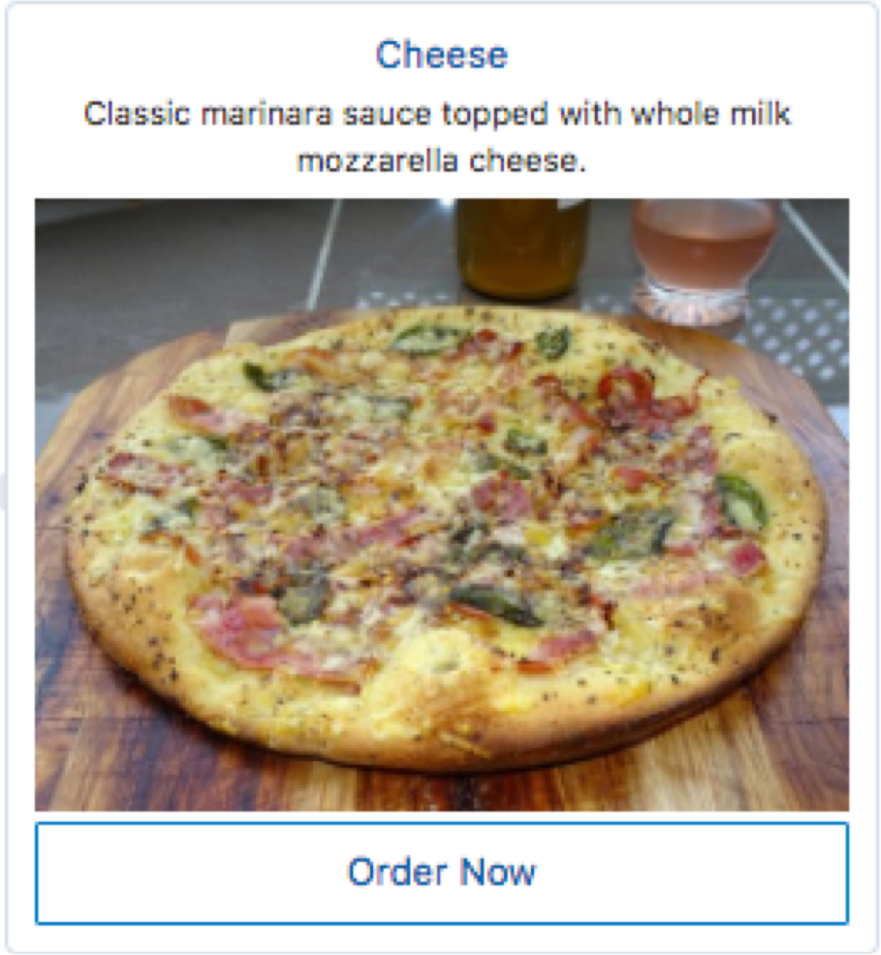

```
showAttachment:
  component: "System.CommonResponse"
  properties:
    metadata:
      responseItems:
      - type: "attachment"
        attachmentType: "image"
        attachmentUrl: "${pizzas.image}"
        iteratorVariable: "pizzas"
    transitions:
      ...
```

Common Response Component

Actions

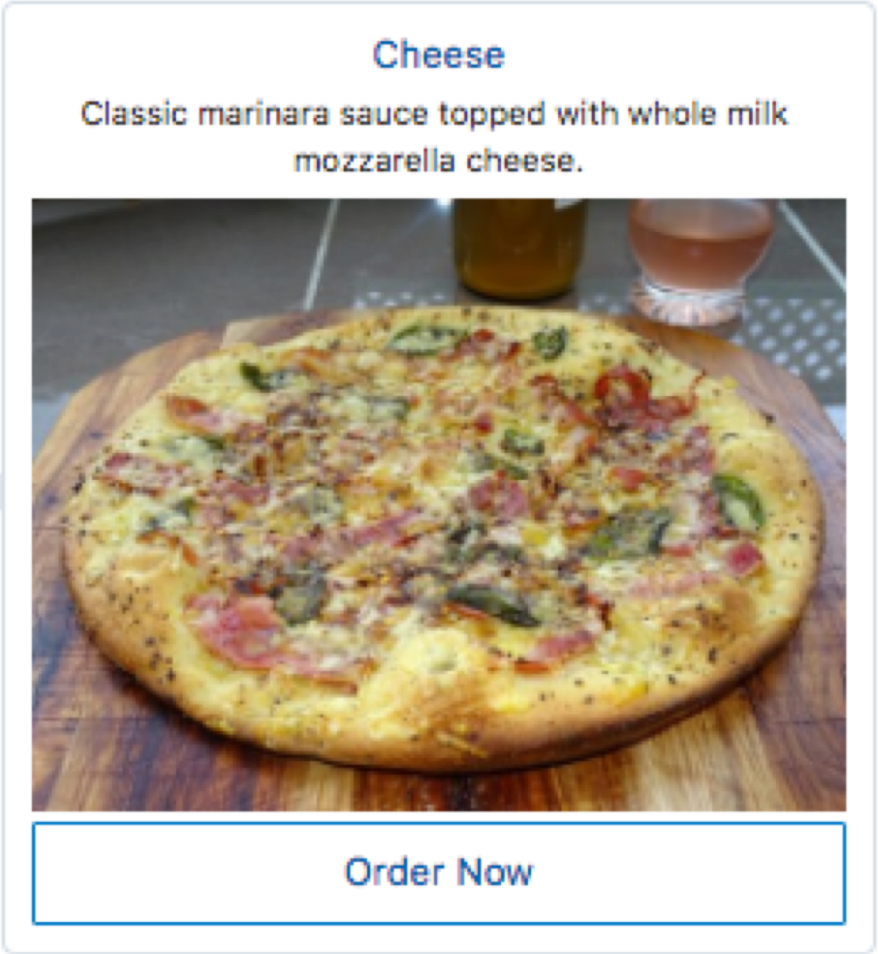
Actions

- Optionally, response items may have actions associated to it
- Action metadata
  - name
  - type
    - postback, share, call, url, location
  - label
  - iteratorVariable
  - imageUrl
  - channelCustomProperties
  - payload



Actions - BotML Example

```
displayMenu:
 component: "System.CommonResponse"
  properties:
   metadata:
    responseItems:
    - type: "cards"
     cardLayout: "vertical"
     cards:
     - title: "${pizzas.name}"
      description: "${pizzas.description}"
      imageUrl: "${pizzas.image}"
      actions:
       - label: "Order Now"
         type: "postback"
         payload:
           action: "order"
          variables:
            orderedPizza: "${pizzas.name}"
    iteratorVariable: "pizzas"
   processUserMessage: true
```



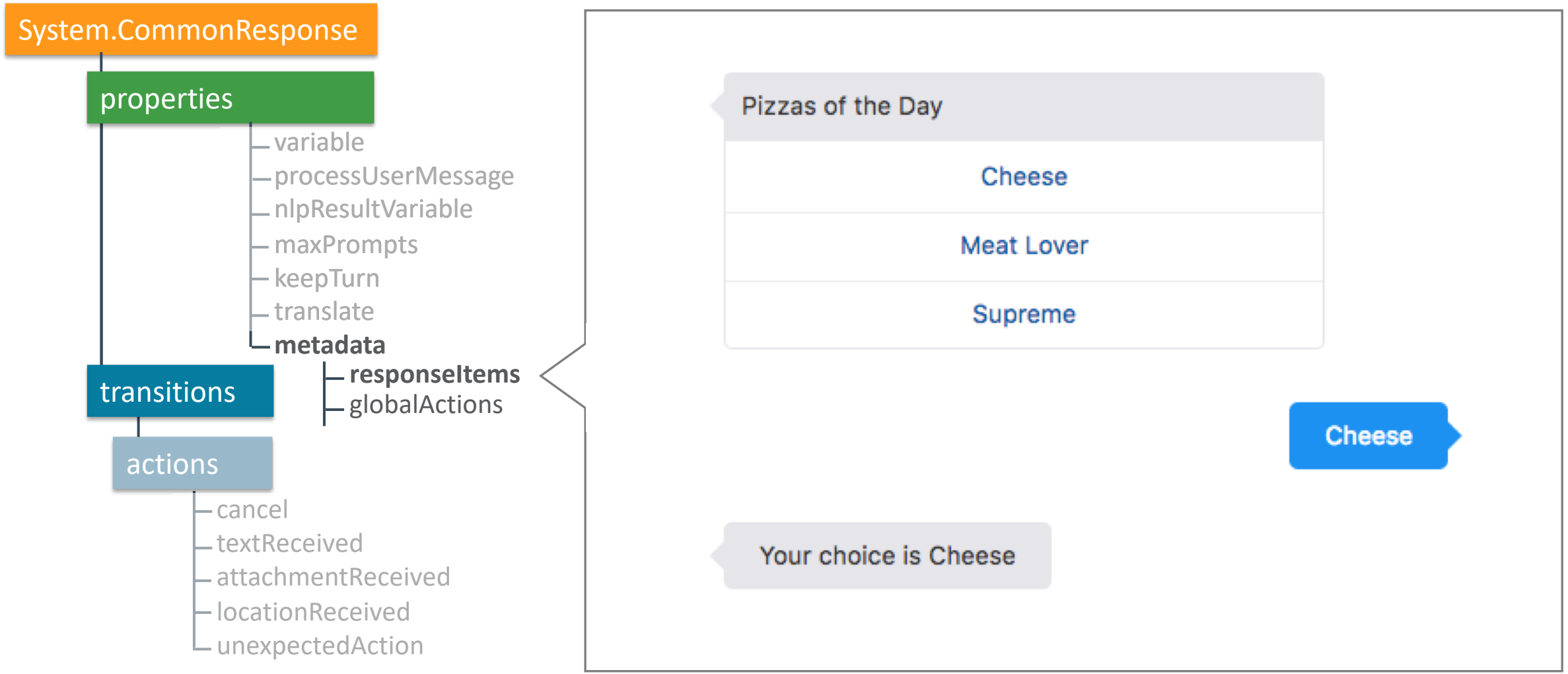
Action Payload

- Only required for "postback", "url" and "call" types
- Properties
  - action
    - Defines the transition action the dialog engine sets when a user selects this action
  - variable_name(s)
    - If the action type is postback, then variable names and values can be defined
    - When action is selected, then the variables referenced in the payload are updated
  - url
    - Used with action type "url" to define the URL to open when the action is selected
  - phoneNumber
    - Used with action type "call" to hold the number to call

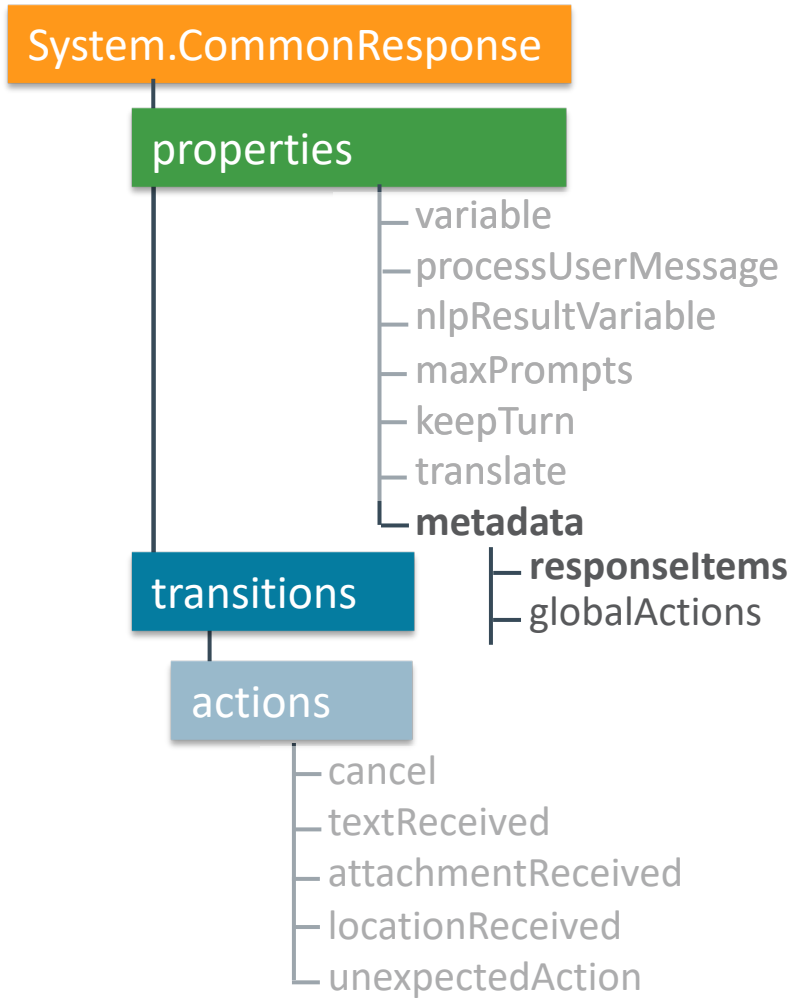
Common Response Component

Postback Actions Example

Actions Example – Displaying A Lists



Actions Example – How to Display Lists




```
displayMenu:
  component: "System.CommonResponse"
  properties:
   processUserMessage: true
   metadata:
    responseItems:
    - type: "text"
      text: "Pizzas of the Day"
      actions:
    - label: "${pizzas.name}"
      type: "postback"
      payload:
        action: "printChoice"
        variables:
          pizzaSelect: "${pizzas.name}"
      iteratorVariable: "pizzas"
  transitions: …
```

Component Response Component

Composite Bag Entities Support

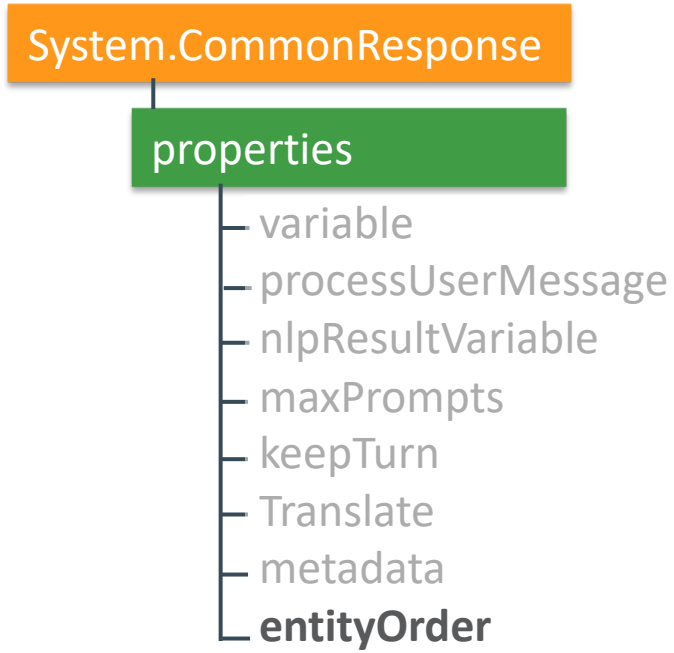
CR Component Support for Composite Bag Entities

- Provides same functionality as System.ResolveEntities
- Allows conditional display of prompts
  - Uses entity defined prompts for custom entities
  - Can use custom prompt for system entities
- Allows definition of global actions

System.entityToResolve

- System variable that tracks the current entity that is rendered
- ${system.entityToResolve.value.**prompt**}
  - Accesses the prompt to display from the entity
- ${system.entityToResolve.value.**enumValues**}
  - Value iterator for entities of "value list" type

Specifying The Entity Order



- Composite Bag Entities have entities defined in a certain order
- CR component "entityOrder" allows the bot designer to change the order in which the component asks the user for missing entity values
- entityOrder accepts a comma separated string of entities
  - Missing entities in this configuration are displayed in the order they are defined in the composite bag entity after the entities listed in this property

CR Component Rendering Composite Bag Entity

```
context:
  variables:
    pizza: "PizzaBag"
    iResult: "nlpresult"
---------------------------------------------------------
orderPizza:
  component: "System.CommonResponse"
  properties:
    variable: "pizza"
    nlpResultVariable: "iResult"
    processUserMessage: true
    entityOrder: "PizzaSize,Toppings,Crust,DATE"
    metadata:
      responseItems:
      - type: "text"
        text: "${system.entityToResolve.value.prompt}"
        actions:
        - label: "${enumValue}"
          type: "postback"
          iteratorVariable: "system.entityToResolve.value.enumValues"
          payload:
            variables:
              pizza: "${enumValue}"
  transitions:
    done
```

Accessing Content of the Composite Bag Entity Variable

- CR component updates variable of type composite bag entity
  - Values are saved to entityName attributes
- To access the value of an entity use: ${pizza.value.<entityName>}

```
printOrder:
  component: "System.Output"
  properties:
    text: "Your  ${pizza.value.PizzaSize} size ${pizza.value.PizzaType} with a ${pizza.value.PizzaCrust}
           crust and ${pizza.value.CheeseType} is on it's way."
  transitions:
    return: "done"
```

Customized Prompts for System Entities

```
orderPizza:
  component: "System.CommonResponse"
  properties:
    variable: "pizza"
    ...
    metadata:
      responseItems:
      - type: "text"
        text: "${system.entityToResolve.value.prompt}"
        visible:
          entitiesToResolve:
            exclude: "EMAIL"
        actions:
        - ...
      - type: "text"
        text: "Please provide your mail address"
        visible:
          entitiesToResolve:
            include: "EMAIL"
  transitions:
    next: done
```

Advanced Bot Training Hands-On

Lab 3b: Create a CR Component Card Layout

Lab 4: Add Pagination Support

Integrated Cloud

Applications & Platform Services


ORACLE®