

ORACLE®

Oracle Digital Assistant

The Complete Training

Introduction to the System.CommonResponse 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.

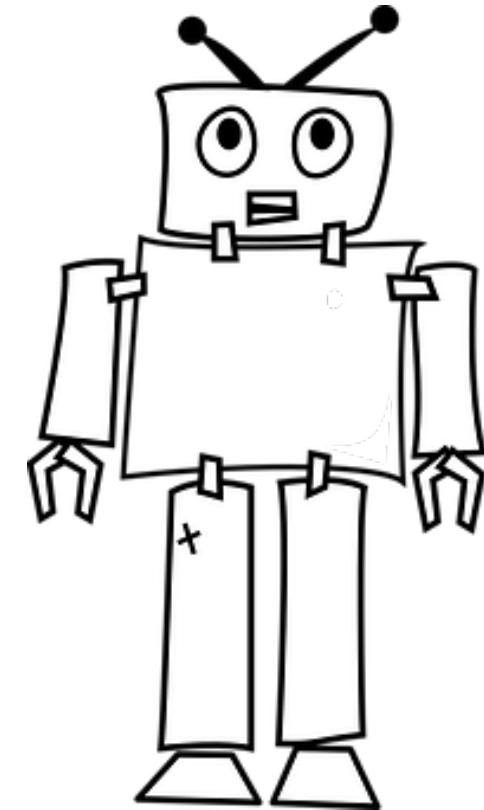
Topic agenda

- 1 ➤ Building good conversational UI
- 2 ➤ Building an input text component
- 3 ➤ Displaying value and action lists
- 4 ➤ Creating a card layout
- 5 ➤ Displaying attachments
- 6 ➤ Choosing a location
- 7 ➤ Local & global actions
- 8 ➤ Composite responses

Topic agenda

- 1 ➤ Building good conversational UI
- 2 ➤ Building an input text component
- 3 ➤ Displaying value and action lists
- 4 ➤ Creating a card layout
- 5 ➤ Displaying attachments
- 6 ➤ Choosing a location
- 7 ➤ Local & global actions
- 8 ➤ Composite responses

There is **no excuse for bad user interface design** when building chatbots.



Building compelling chatbot user interfaces

- Guide and assist users in making a choice or providing input
- Display a UI that is pleasing to the eye
 - Lists, Card Layouts, Images, Buttons or a combination of them
- Optimize bot responses for the messaging channel that is used



About Oracle

Oracle provides essential elements for companies to pioneer innovations and drive new business models.

For example, applications based on artificial intelligence (AI) can suggest the next best actions, automate answers, and provide personalized service.

Get our latest facts sheet ...

<http://www.oracle.com/us/corporate/oracle-fact-sheet-079219.pdf>

... or contact an expert.

GRANT RONALD
Great Britain (UK)

FRANK NIMPHIUS
Germany (DE)

DON MCINES
United States (US)

Select Select Select

A screenshot of a mobile application interface. It displays three contact cards in a grid. Each card contains a small profile picture of a man, the name, and the location. Below each card is a blue rectangular button labeled "Select".

Common response component

- The 'Clark Kent' among the system components
 - Can build simple and complex bot UI
 - Support for composite bag entities and iterators
 - Renders text, list, cards, location and attachment UI
- Aligns with Conversational Message Model (CMM)
- For many use cases, avoids the need for custom components



About Oracle

Oracle provides essential elements for companies to pioneer innovations and drive new business models.

For example, applications based on artificial intelligence (AI) can suggest the next best actions, automate answers, and provide personalized service.

Get our latest facts sheet ...

<http://www.oracle.com/us/corporate/oracle-fact-sheet-079219.pdf>

... or contact an expert.

The interface displays three user profiles in a grid:

- GRANT RONALD**
Great Britain (UK)
Select
- FRANK NIMPHIUS**
Germany (DE)
Select
- DON MCINES**
United States (US)
Select

Topic agenda

- 1 ➤ Building good conversational UI
- 2 ➤ Building an input text component
- 3 ➤ Displaying value and action lists
- 4 ➤ Creating a card layout
- 5 ➤ Displaying attachments
- 6 ➤ Choosing a location
- 7 ➤ Local & global actions
- 8 ➤ Composite responses

Creating a text response using the component templates

The screenshot shows the Oracle对话设计器 (Dialog Designer) interface. In the top left, there's a code editor window with some metadata code. Below it is a navigation bar with tabs: '+ Components' (highlighted in green), '?', and a search bar 'Select a Component Type'. The main workspace contains four categories: 'Control' (represented by a tree icon), 'Language' (represented by a speech bubble icon), 'Security' (represented by a lock icon), and 'User Interface' (represented by a hand icon). A 'Variables' section is also present. On the right, a modal window titled 'Component Template' is open, showing the 'User Interface' tab selected. Inside, a list of component templates is shown, with 'Common response - text' highlighted with a red rectangle. To the right of the list is the component template configuration code:

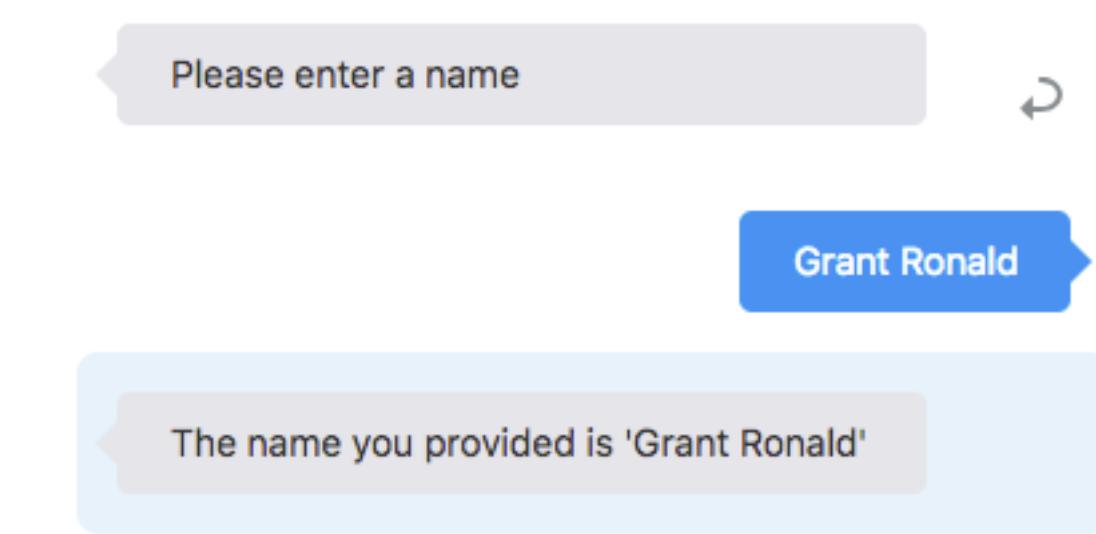
```
textResponse:  
  component: "System.CommonResponse"  
  properties:  
    # set processUserMessage to true if the dialog flow should  
    # return to this state after receiving user message  
    processUserMessage: true  
    # set keepTurn (true/false) to true if the dialog flow should  
    # transition to the next state without waiting for user input. Only  
    # applicable when processUserMessage is false  
    keepTurn: false  
    # variable (optional) refers to the context or user variable  
    # that will be set to the text value entered by the bot user. If the  
    # variable already has a value, the dialog flow transitions to the next  
    # state without sending the bot response as specified in the metadata  
    # property
```

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

Displaying text input prompts

```
getName:  
  component: "System.CommonResponse"  
  properties:  
    processUserMessage: true  
    keepTurn: false  
    variable: "person"  
    nlpResultVariable:  
  metadata:  
    responseItems:  
      - type: "text"  
        text: "Please enter a name"  
transitions:  
  next: "printName"
```

```
printName:  
  component: "System.Output"  
  properties:  
    text: "The name you provided is '${person.value}'"  
    keepTurn: false  
transitions:  
  return: "done"
```



Topic agenda

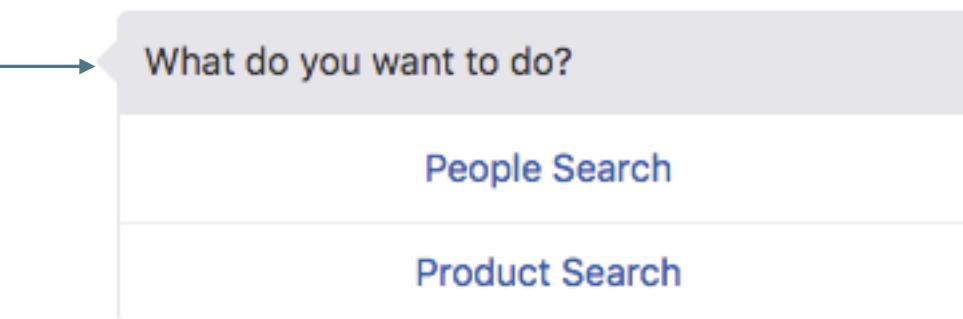
- 1 ➤ Building good conversational UI
- 2 ➤ Building an input text component
- 3 ➤ Displaying value and action lists
- 4 ➤ Creating a card layout
- 5 ➤ Displaying attachments
- 6 ➤ Choosing a location
- 7 ➤ Local & global actions
- 8 ➤ Composite responses

About list-of-values

- Value list
 - Displays a single-select list of values
 - Updates one or many context variables



- Action lists
 - Displays a list of actions
 - Commonly used to build select menus
 - Selecting a list item triggers a transition action
 - Action strings can be freely chosen



- Hybrid list
 - Combines value and action lists

Building list-of-values

The screenshot shows the Oracle Dialog Designer interface. On the left, there's a code editor window with some metadata code. In the center, a modal dialog titled "Select a Component Type" is open, displaying four categories: Control, Language, Security, and User Interface. Below this, another modal dialog titled "Component Template" is open, specifically for the "User Interface" category. This dialog lists several component types: Common response - attachment, Common response - card, Common response - composite bag, **Common response - text** (which is highlighted with a red rectangle), Interactive, List - set action, and List - set variable. To the right of the list is the component template code:

```
textResponse:  
  component: "System.CommonResponse"  
  properties:  
    # set processUserMessage to true if the dialog flow should  
    # return to this state after receiving user message  
    processUserMessage: true  
    # set keepTurn (true/false) to true if the dialog flow should  
    # transition to the next state without waiting for user input. Only  
    # applicable when processUserMessage is false  
    keepTurn: false  
    # variable (optional) refers to the context or user variable  
    # that will be set to the text value entered by the bot user. If the  
    # variable already has a value, the dialog flow transitions to the next  
    # state without sending the bot response as specified in the metadata  
    # property
```

At the bottom of the "Component Template" dialog, there are buttons for "Insert After", "Remove Comments" (with a toggle switch), and a green "Apply" button.

Action lists

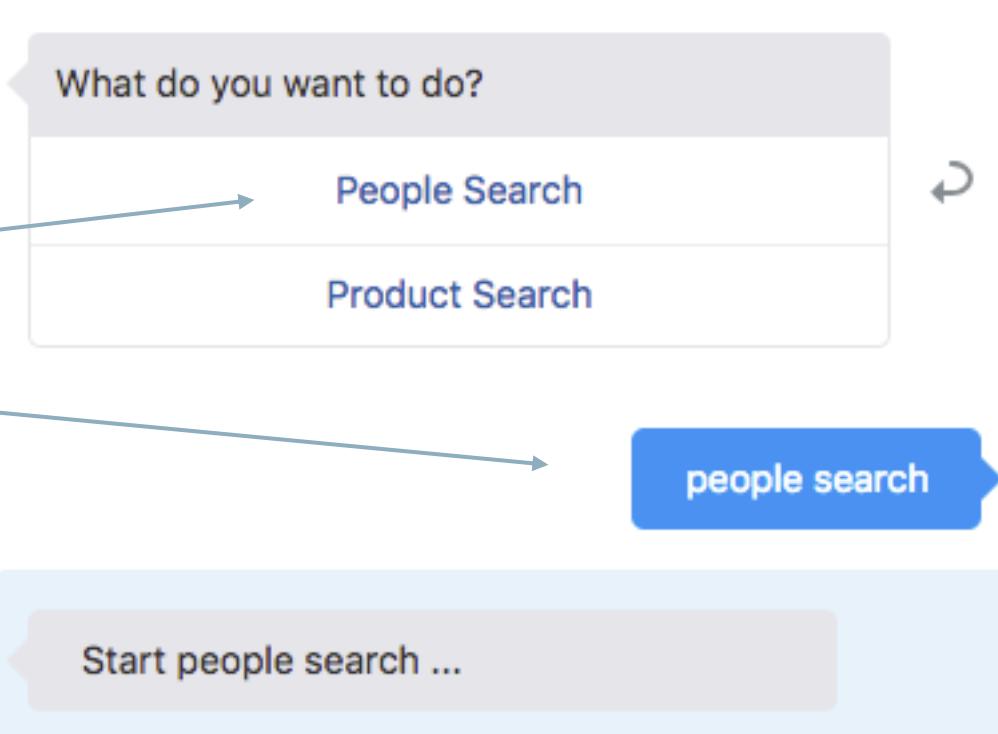
```
displayMenu:  
  component: "System.CommonResponse"  
  properties:  
    processUserMessage: true  
    keepTurn: false
```

metadata:

```
  responseItems:  
    - type: "text"  
      text: "What do you want to do?"  
      actions:  
        - label: "People Search"  
          type: "postback"  
          keyword: "people, people search"  
          payload:  
            action: "peopleSearch"  
        - label: "Product Search"  
          type: "postback"  
          keyword: "product, product search"  
          payload:  
            action: "productSearch"
```

transitions:

```
  actions:  
    peopleSearch: "searchPeople"  
    productSearch: "searchProduct"
```



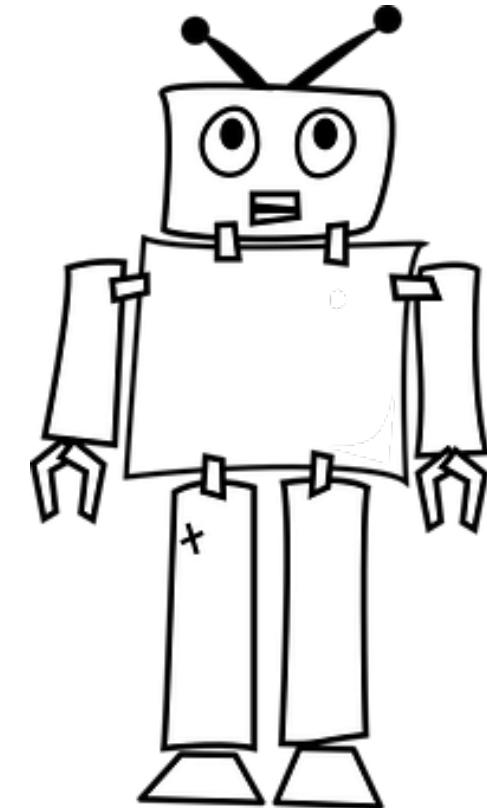
Static list-of-values

```
searchPeople:  
  component: "System.CommonResponse"  
  properties:  
    processUserMessage: true  
    keepTurn: false  
    variable:  
    nlpResultVariable:  
      metadata:  
        responseItems:  
          - type: "text"  
            text: "Please select"  
            actions:  
              - label: "Grant Ronald"  
                type: "postback"  
                keyword: "Grant, Grant Ronald"  
                payload:  
                  variables:  
                    person: "Grant Ronald"  
                    location: "Great Britain (UK)"  
              - label: "Frank Nimphius"  
                type: "postback"  
                keyword: "Frank, Frank Nimphius"
```

People Search



Use the component **variable** and
nlpResultVariable properties to
implement entity slotting and
entity validation



About data arrays

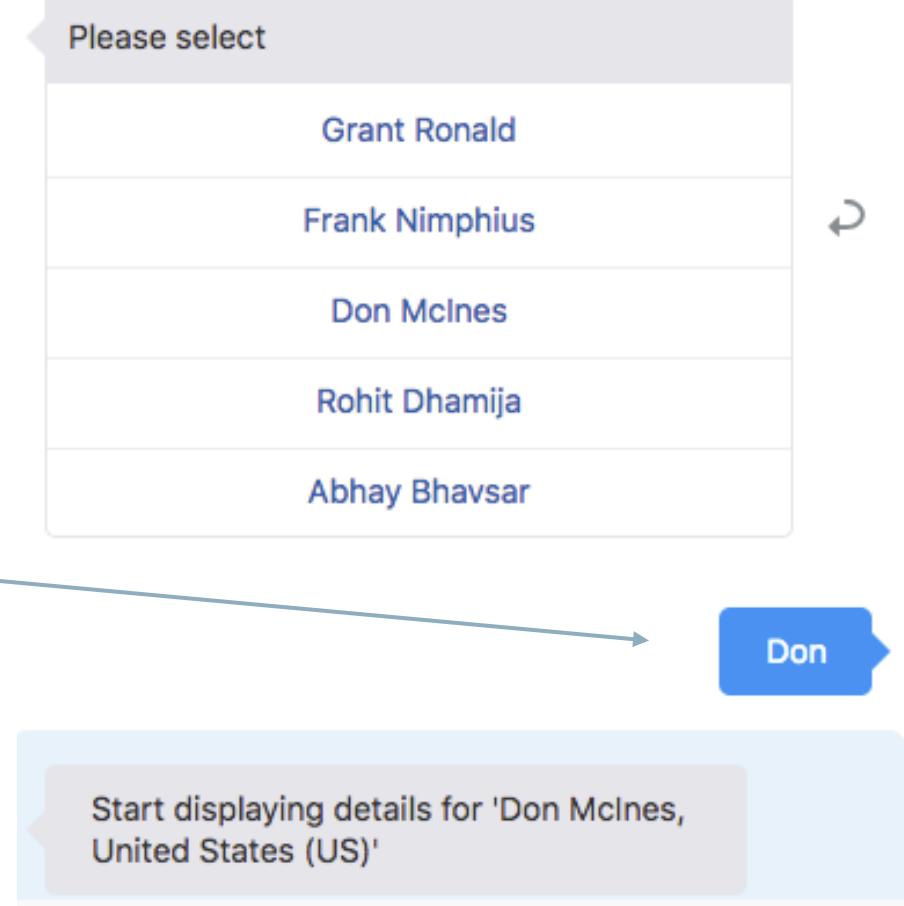
- Oracle Digital Assistant does not provide a map or array type for context variables
- Arrays are defined in context variables of type "string"
 - Created using Apache FreeMarker expressions in System.SetVariable
 - Created using custom components that write to the variable

```
variables:  
  personArray: "string"  
  
setPeople:  
  component: "System.SetVariable"  
  properties:  
    variable: "personArray"  
    value:  
      - name: "Grant Ronald"  
        location: "Great Britain (UK)"  
        image: "https://people.oracle.com/apex/oracle/images/scaled/grant.ronald@oracle.com"  
        mail: "grant.ronald@oracle.com"  
      - name: "Frank Nimphius"  
        location: "Germany (DE)"  
        image: "https://people.oracle.com/apex/oracle/images/scaled/frank.nimphius@oracle.com"  
        mail: "frank.nimphius@oracle.com"  
      - name: "Don McInnes"  
        location: "United States (US)"  
        image: "https://people.oracle.com/apex/oracle/images/scaled/don.mcinnnes@oracle.com"  
        mail: "don.mcinnnes@oracle.com"  
      - name: "Rohit Dhamija"  
        location: "India (IN)"  
        image: "https://people.oracle.com/apex/oracle/images/scaled/rohit.dhamija@oracle.com"  
        mail: "rohit.dhamija@oracle.com"  
      - name: "Abhay Bhavsar"  
        location: "India (IN)"  
        image: "https://people.oracle.com/apex/oracle/images/scaled/abhay.bhavsar@oracle.com"  
        mail: "abhay.bhavsar@oracle.com"  
transitions:  
  next: "displayMenu"
```

Dynamic list-of-values

People Search

```
searchPeople:  
  component: "System.CommonResponse"  
  properties:  
    processUserMessage: true  
    keepTurn: false  
    variable:  
      nlpResultVariable:  
        metadata:  
          responseItems:  
            - type: "text"  
              text: "Please select"  
              actions:  
                - label: "${personArray.name}"  
                  type: "postback"  
                  keyword: "${personArray.name?replace(' ', ',')}"  
                  payload:  
                    variables:  
                      person: "${personArray.name}"  
                      location: "${personArray.location}"  
        iteratorVariable: "personArray"  
  transitions:  
    next: "printPersonDetails"
```



Topic agenda

- 1 ➤ Building good conversational UI
- 2 ➤ Building an input text component
- 3 ➤ Displaying value and action lists
- 4 ➤ Creating a card layout
- 5 ➤ Displaying attachments
- 6 ➤ Choosing a location
- 7 ➤ Local & global actions
- 8 ➤ Composite responses

Building card layouts using the component templates

The screenshot shows the Oracle Service Cloud interface for building card layouts using component templates. The top navigation bar has a green button labeled '+ Components' and a question mark icon. Below it, a sidebar displays code snippets for metadata files, including sections for #metadata, #platform, and main: name: context: variables: data: states:.

The main area is titled 'Select a Component Type' and shows four categories: Control (Control icon), Language (Language icon), Security (Security icon), and User Interface (User Interface icon). A modal window is open for the 'User Interface' category, specifically for 'Common response - card'. This modal has a back arrow, a title 'User Interface', and a list of options: Agent conversation, Agent initiation, Common response - attachment, **Common response - card** (which is highlighted with a red border), Common response - composite bag, Common response - text, and Interactive. To the right of the list is a 'Component Template' pane containing the following JSON code:

```
cardResponse:  
  component: "System.CommonResponse"  
  properties:  
    # set processUserMessage to true if the dialog flow should  
    return to this state after receiving the user's message.  
    processUserMessage: true  
    # autoNumberPostBackActions (optional) allows you to override  
    the global autoNumberPostBackActions variable. If set to true, the  
    labels of the buttons for postback actions are prefixed with a  
    sequence number. Entering this sequence number will execute the  
    button postback payload as if the button was tapped. This is useful  
    for channels like SMS that do not support buttons.  
    autoNumberPostBackActions:  
      # translate property allows you to override the global  
      AutoTranslate variable. If set to true, then both the bot's and the
```

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

Card layout definition

```
searchPeople:
```

```
  component: "System.CommonResponse"
```

```
properties:
```

```
  processUserMessage: true
```

```
  autoNumberPostbackActions:
```

```
metadata:
```

```
  responseItems:
```

```
- type: "cards"
```

```
  cardLayout: "horizontal"
```

```
  cards:
```

```
- title: "${personArray.name?upper_case}"
```

```
  description: "${personArray.location}"
```

```
  imageUrl: "${personArray.image}"
```

```
  iteratorVariable: "personArray"
```

```
  rangeStart:
```

```
  rangeSize:
```

```
  actions:
```

```
- label: "Select"
```

```
  type: "postback"
```

```
  keyword: "${personArray.name?replace(' ', ',')}"
```

```
  payload:
```

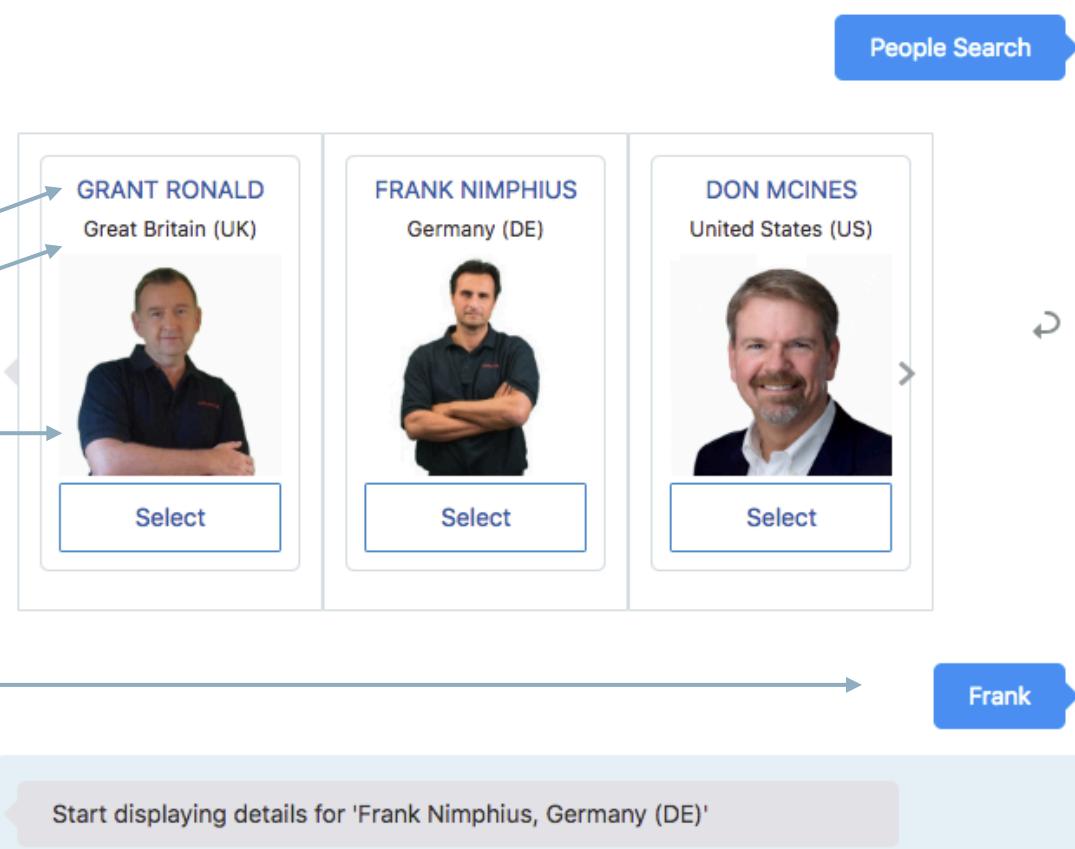
```
    variables:
```

```
      person: "${personArray.name}"
```

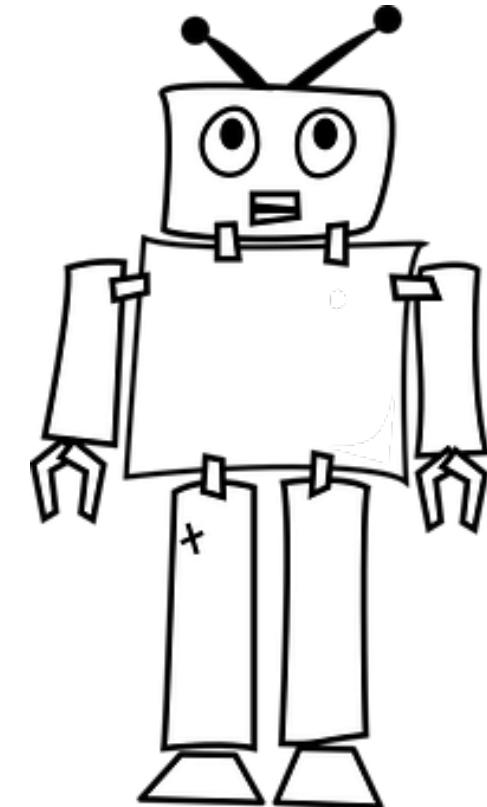
```
      location: "${personArray.location}"
```

```
transitions:
```

```
  next: "printPersonDetails"
```



Messengers are limited in the number of cards that can be viewed at one time. Use the **rangeStart** and **rangeSize** properties to implement page ranging



Topic agenda

- 1 ➤ Building good conversational UI
- 2 ➤ Building an input text component
- 3 ➤ Displaying value and action lists
- 4 ➤ Creating a card layout
- 5 ➤ Displaying attachments
- 6 ➤ Choosing a location
- 7 ➤ Local & global actions
- 8 ➤ Composite responses

Creating attachments using the component templates

The screenshot shows the Oracle Bot Service interface. On the left, there's a code editor window with some metadata code. In the center, a modal window titled "Select a Component Type" is open, showing categories: Control, Language, Security, and User Interface. Below this, another modal window titled "Component Template" is open under the "User Interface" category. This window lists several options: Agent conversation, Agent initiation, Common response - attachment (which is highlighted with a red box), Common response - card, Common response - composite bag, Common response - text, and Interactive. To the right of these options is a detailed description of the "Common response - attachment" template, which includes code snippets and explanatory text about attachment responses. At the bottom of the "Component Template" window are buttons for "Insert After", "Remove Comments", and "Apply".

```
attachmentResponse:  
  component: "System.CommonResponse"  
  properties:  
    # set processUserMessage to true if the dialog flow should  
    # return to this state after receiving the user's message.  
    processUserMessage: false  
    # set keepTurn to true if the dialog flow should transition to  
    # the next state without waiting for the user input. Only applicable  
    # when processUserMessage is false.  
    keepTurn: false  
    # metadata property specifies the structure of the bot response  
    # message(s) that are sent to the user. You can define text, card, and  
    # attachment message types, add actions to text and card messages, and  
    # specify global actions that typically appear at the bottom of the  
    # chat window.  
    metadata:
```

Attachment

- Displays content
 - Audio, video, image, file
 - No streaming of binaries
- Content rendering depends on messenger
 - No guarantee that videos e.g. are displayed in place
- Attachments cannot have action items

```
showImageAsAttachment:  
  component: "System.CommonResponse"  
  properties:  
    processUserMessage: false  
    keepTurn: false  
  metadata:  
    responseItems:  
      - type: "attachment"  
        attachmentType: "image"  
        attachmentUrl: "${imageUrl.value}"  
  transitions:  
    return: "done"
```

Thank you for your order, your Large PEPPERONI pizza with Tomatoes will be delivered in 30 minutes at home!

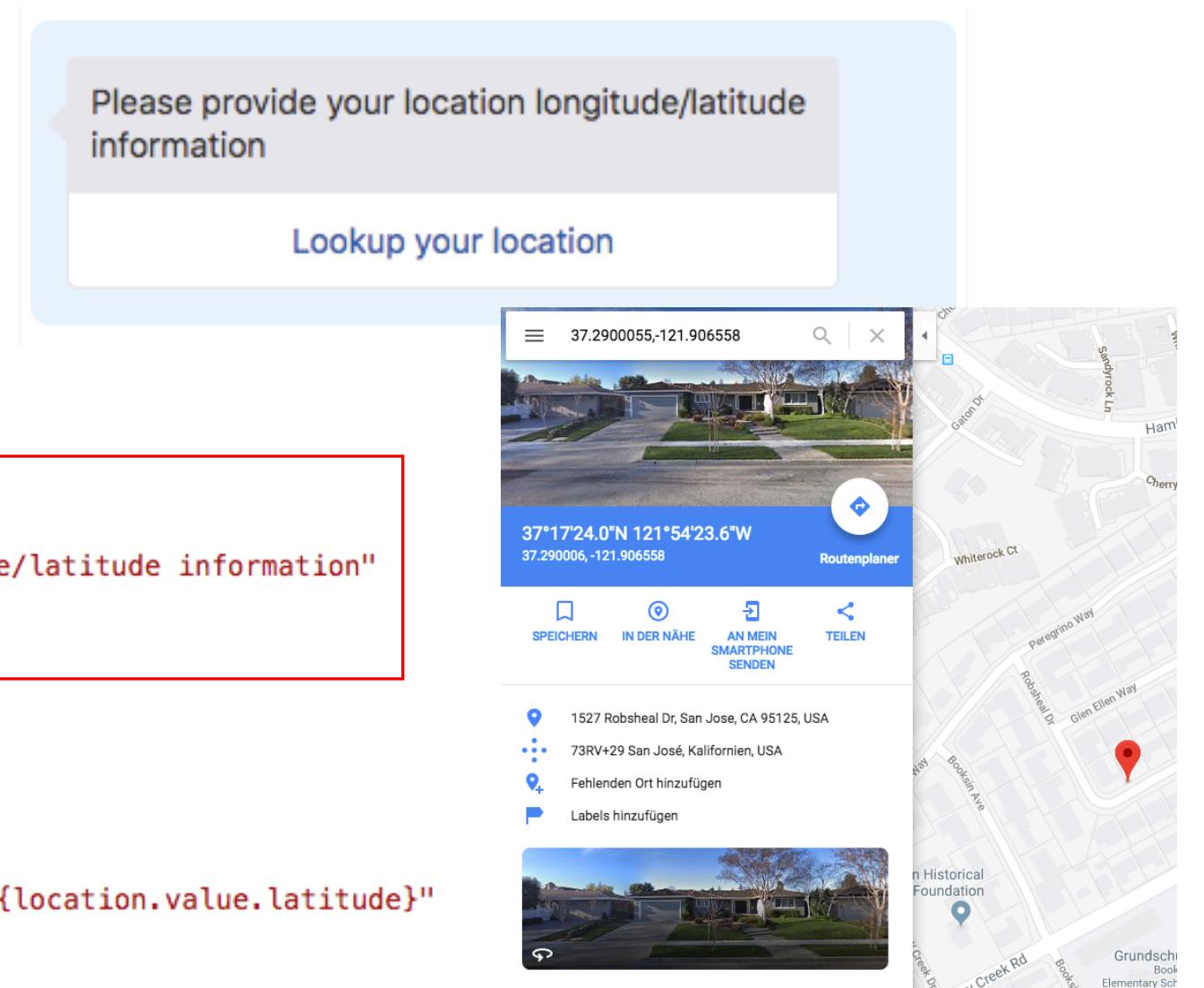


Topic agenda

- 1 ➤ Building good conversational UI
- 2 ➤ Building an input text component
- 3 ➤ Displaying value and action lists
- 4 ➤ Creating a card layout
- 5 ➤ Displaying attachments
- 6 ➤ Choosing a location
- 7 ➤ Local & global actions
- 8 ➤ Composite responses

Choosing a location

```
chooseLocation:  
  component: "System.CommonResponse"  
  properties:  
    processUserMessage: true  
    keepTurn: false  
    variable: "location"  
    nlpResultVariable:  
      metadata:  
        responseItems:  
          - type: "text"  
            text: "Please provide your location longitude/latitude information"  
          actions:  
            - label: "Lookup your location"  
              type: "location"  
  transitions:  
    next: "printLocation"  
  
printLocation:  
  component: "System.Output"  
  properties:  
    text: "long: ${location.value.longitude} lat: ${location.value.latitude}"  
  transitions:  
    return: "done"
```



Topic agenda

- 1 ➤ Building good conversational UI
- 2 ➤ Building an input text component
- 3 ➤ Displaying value and action lists
- 4 ➤ Creating a card layout
- 5 ➤ Displaying attachments
- 6 ➤ Choosing a location
- 7 ➤ Local & global actions
- 8 ➤ Composite responses

Action types

People Search

GRANT RONALD Great Britain (UK) Select	FRANK NIMPHIUS Germany (DE) Select	DON MCINES United States (US) Select
...		
Show More		
Show Less		
Show Managers Show Organisation		

Action items local to a card

Actions local to a response type

Actions global for a component

Action types

People Search

GRANT RONALD
Great Britain (UK)
Select

FRANK NIMPHIUS
Germany (DE)
Select

DON MCINES
United States (US)
Select

Show More

Show Less

Show Managers Show Organisation

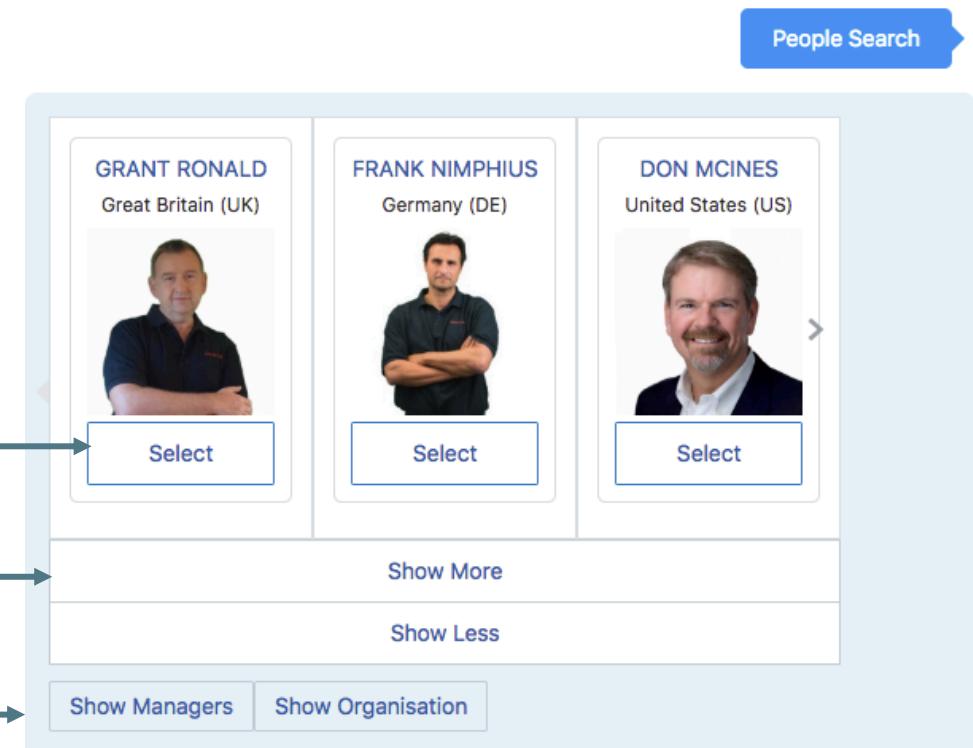
```
searchPeople:  
  component: "System.CommonResponse"  
  properties:  
    processUserMessage: true  
    autoNumberPostbackActions:  
    metadata:  
      responseItems:  
        - type: "cards"  
          cardLayout: "horizontal"  
          cards:  
            - title: "${personArray.name?upper_case}"  
              description: "${personArray.location}"  
              imageUrl: "${personArray.image}"  
              iteratorVariable: "personArray"  
              rangeStart:  
              rangeSize:  
            actions:  
              - label: "Select"  
                type: "postback"  
                keyword: "${personArray.name?replace(' ', ',')}"  
                payload:  
                  variables:  
                    person: "${personArray.name}"  
                    location: "${personArray.location}"
```

```
actions:  
  - label: "Show More"  
    type: "postback"  
    keyword: "show more, more"  
    payload:  
      action: "showMore"  
  - label: "Show Less"  
    type: "postback"  
    keyword: "show less, less"  
    payload:  
      action: "showMore"
```

```
globalActions:  
  - label: "Show Managers"  
    type: "postback"  
    keyword: "manager"  
    payload:  
      action: "showManagers"  
  - label: "Show Organisation"  
    type: "postback"  
    keyword: "organisation"  
    payload:  
      action: "showOrg"  
transitions:
```

About actions types

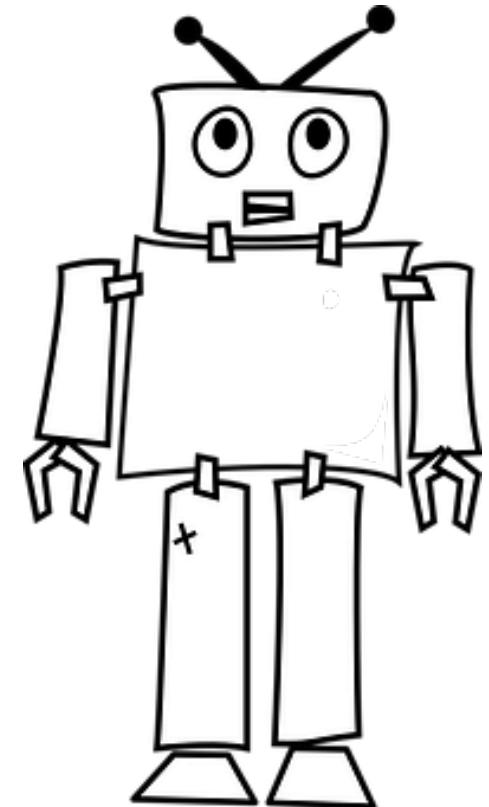
- Action items local to a card
 - Buttons associated with list items or cards
 - Remain visible when component goes out of scope
- Actions local to a response type
 - Buttons associated with component
 - Remains visible when component goes out of scope
- Actions global for a component
 - E.g. quick replies on Facebook



Topic agenda

- 1 ➤ Building good conversational UI
- 2 ➤ Building an input text component
- 3 ➤ Displaying value and action lists
- 4 ➤ Creating a card layout
- 5 ➤ Displaying attachments
- 6 ➤ Choosing a location
- 7 ➤ Local & global actions
- 8 ➤ Composite responses

**Using the Common Response component,
you can combine multiple response types
to build arbitrarily complex bot responses
with ease**



```

helloOracle:
  component: "System.CommonResponse"
  properties:
    processUserMessage: true
    autoNumberPostbackActions:
    metadata:
      responseItems:
        - type: "attachment"
          attachmentType: "image"
          attachmentUrl: "https://www.oracle.com/us/assets/cb15-small-events-2868339.jpg?16"
        - type: "text"
          text: "About Oracle"
        - type: "text"
          text: |-  

            Oracle provides essential elements for companies to pioneer innovations and drive  

            new business models.  

            For example, applications based on artificial intelligence (AI) can suggest the ne...
        - type: "text"
          text: "Get our latest facts sheet ... "
        - type: "attachment"
          attachmentType: "file"
          attachmentUrl: "http://www.oracle.com/us/corporate/oracle-fact-sheet-079219.pdf"
        - type: "text"
          text: "... or contact an expert."
        - type: "cards"
          cardLayout: "horizontal"
          cards:
            - title: "${personArray.name?upper_case}"
              description: "${personArray.location}"
              imageUrl: "${personArray.image}"
              iteratorVariable: "personArray"
              rangeStart:
              rangeSize:
              actions:
                - label: "Select"
                  type: "postback"
                  keyword: "${personArray.name?replace(' ','','')}"
                  payload:
                    action: "showPersonDetails"
                    variables:
                      personIndex: "${personArray?index}"
  transitions:
    actions:
      showPersonDetails: "printPersonDetails"
      textReceived: "done"

```

About Oracle

Oracle provides essential elements for companies to pioneer innovations and drive new business models.

For example, applications based on artificial intelligence (AI) can suggest the next best actions, automate answers, and provide personalized service.

Get our latest facts sheet ...

<http://www.oracle.com/us/corporate/oracle-fact-sheet-079219.pdf>

... or contact an expert.

GRANT RONALD Great Britain (UK) 	FRANK NIMPHIUS Germany (DE) 	DON MCINES United States (US)
Select	Select	Select

Integrated Cloud Applications & Platform Services

ORACLE®



Oracle Digital Assistant Hands-On

TBD