

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.

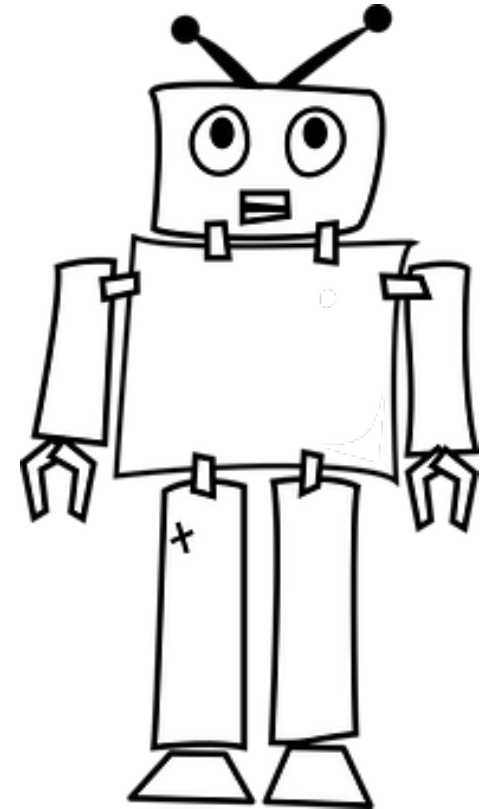
Topic agenda

- 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

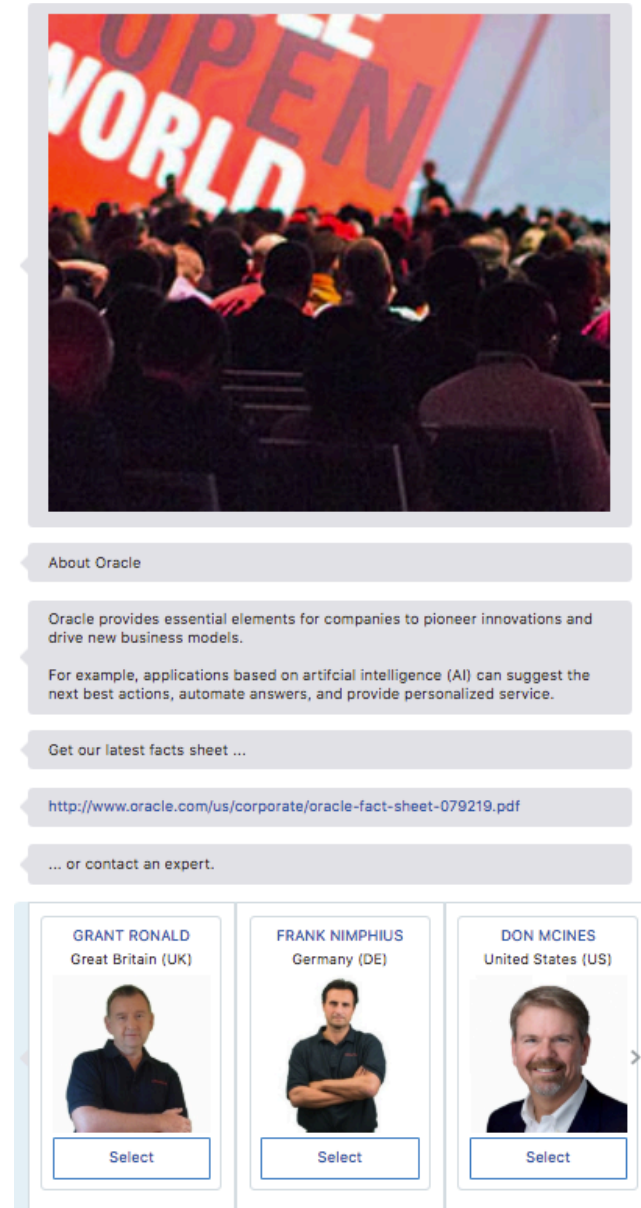
- 1 Building good conversational UI
- 2 Building an input text component
- 3 Displaying value and action lists
- 4 Creating a card layout
- 5 Displaying attachments
- 6 Choosing a location
- 7 Local & global actions
- 8 Composite responses

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



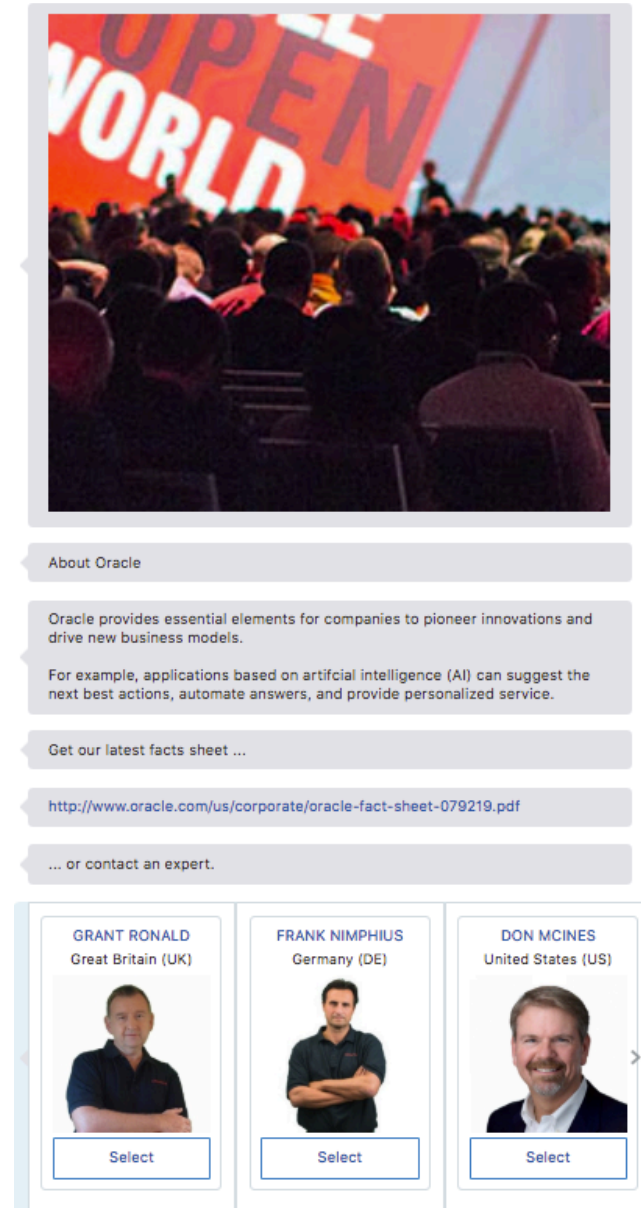
Building compelling chatbot user interfaces

- 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



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



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 illustrates the process of creating a text response using component templates in Oracle APEX. It shows two overlapping windows.

Left Window: Components Palette

- Title:** + Components
- Header:** Select a Component Type
- Grid:**
 - Control:** Represented by a tree icon.
 - Language:** Represented by a speech bubble icon with 'A' and 'x'.
 - Security:** Represented by a padlock icon.
 - User Interface:** Represented by a hand icon pointing at a screen. This category is highlighted with a grey background.
 - Variables:** Represented by a green '(x)' icon.

Right Window: Component Template Editor

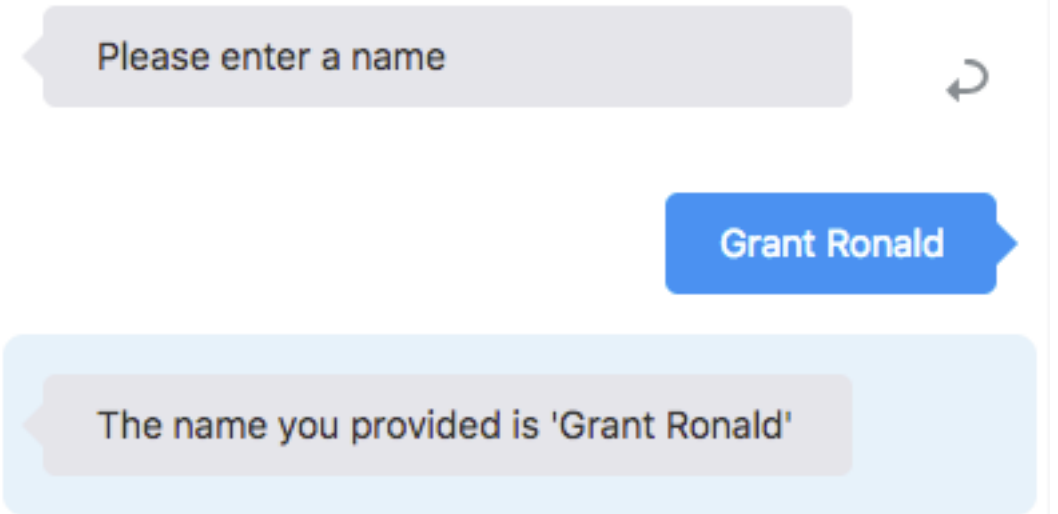
- Title:** User Interface
- Left List:**
 - Common response - attachment
 - Common response - card
 - Common response - composite bag
 - Common response - text** (highlighted with a red box)
 - Interactive
 - List - set action
 - List - set variable
- Right Panel: Component Template**

```
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
```
- Bottom:** Insert After, Remove Comments (toggle), and an Apply button.

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

Please select

Grant Ronald

Frank Nimphius

Don McInes

What do you want to do?

People Search

Product Search

Building list-of-values

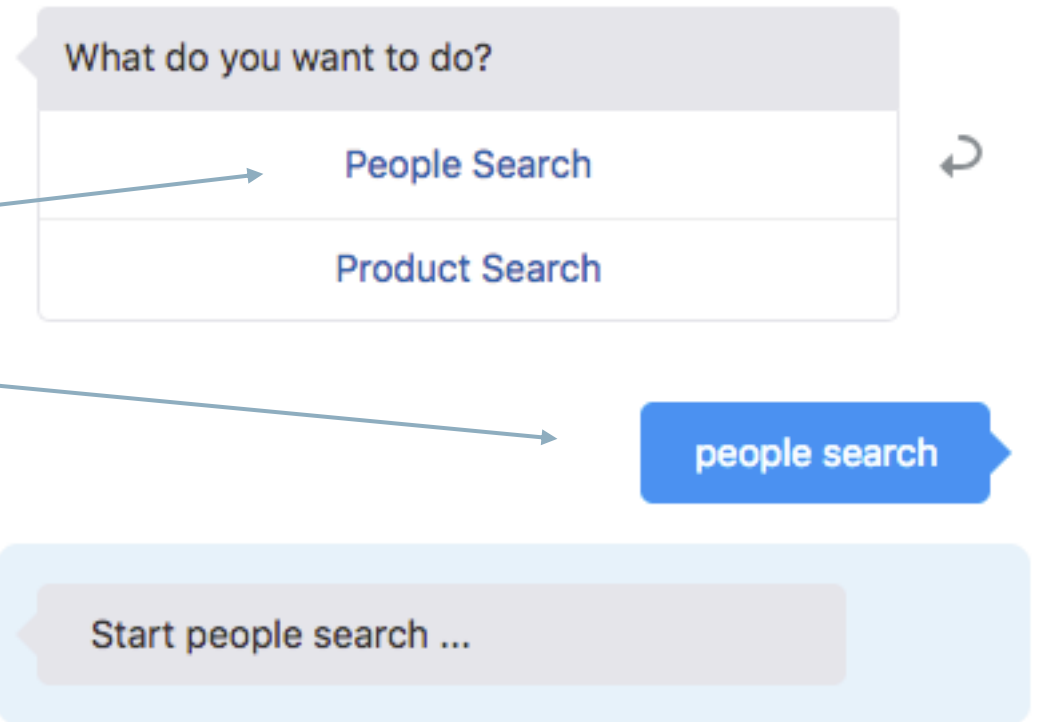
The screenshot shows the Oracle APEX Components palette on the left, with the 'User Interface' component type selected. The 'Common response - text' option is highlighted in the list. The 'Component Template' editor on the right shows the following JSON template:

```
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 editor, there are checkboxes for 'Insert After' and 'Remove Comments' (which is checked), and an '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

Please select

Grant Ronald

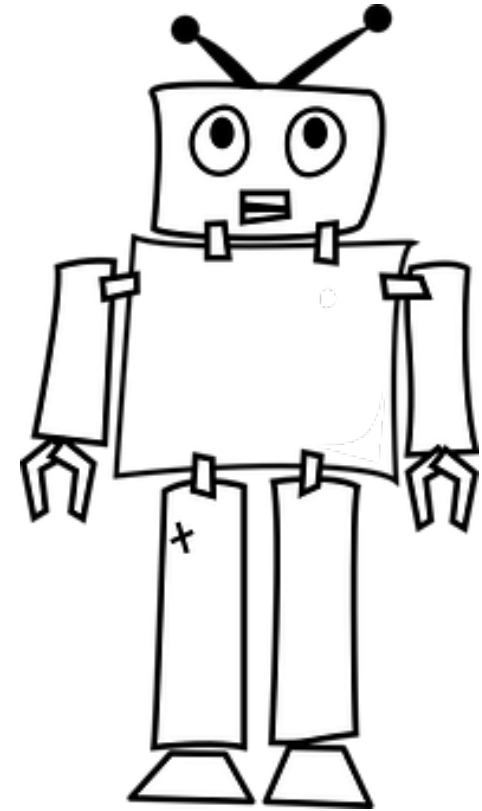
Frank Nimphius

Don McInnes

Rohit Dhamija

Abhay Bhavsar

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.mcinnes@oracle.com"  
        mail: "don.mcinnes@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

```
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"
```

People Search

Please select

Grant Ronald

Frank Nimphius

Don McInes

Rohit Dhamija

Abhay Bhavsar

Don

Start displaying details for 'Don McInes, United States (US)'

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 APEX Components dialog. On the left, a list of components is visible, including Control, Language, Security, User Interface, and Variables. The User Interface component is selected. On the right, the 'Component Template' dialog is open, showing a list of templates. The 'Common response - card' template is highlighted. The template details show a JSON structure for a card response, including properties for processUserMessage, autoNumberPostbackActions, and translate.

Components

Select a Component Type

- Control
- Language
- Security
- User Interface
- Variables

User Interface

- Agent conversation
- Agent initiation
- Common response - attachment
- Common response - card**
- Common response - composite bag
- Common response - text
- Interactive

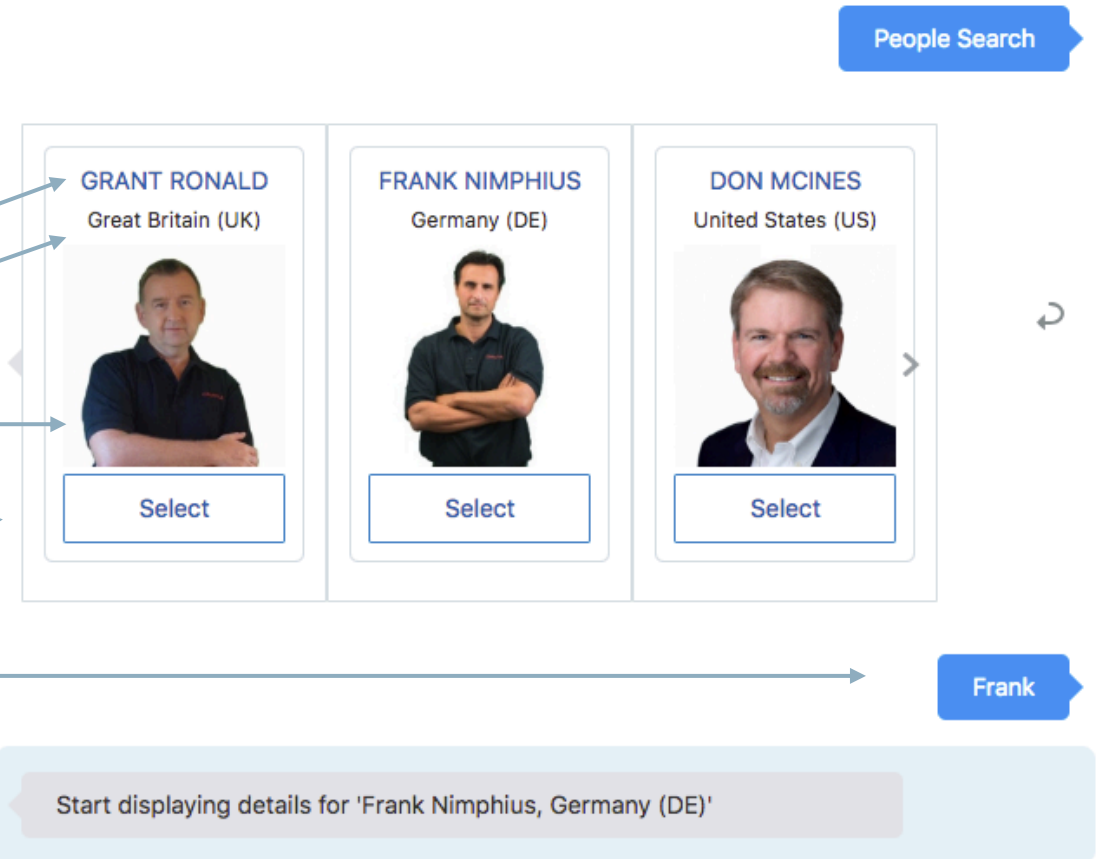
Component Template

```
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
```

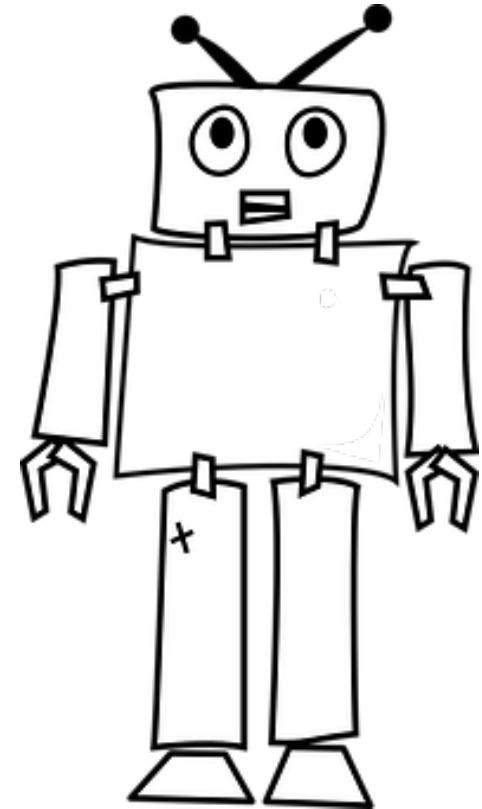
Insert After: displayMenu Remove Comments: ☒ Apply

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 APEX Components dialog. On the left, a list of components is visible, including Control, Language, Security, User Interface, and Variables. The User Interface component is selected. On the right, the 'Component Template' dialog is open, showing a list of templates under the 'User Interface' category. The 'Common response - attachment' template is highlighted with a red box. The template details show the following JSON structure:

```
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.
```

At the bottom of the dialog, there is a dropdown menu for 'Insert After' set to 'searchPeople', a 'Remove Comments' toggle switch, and an 'Apply' button.

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

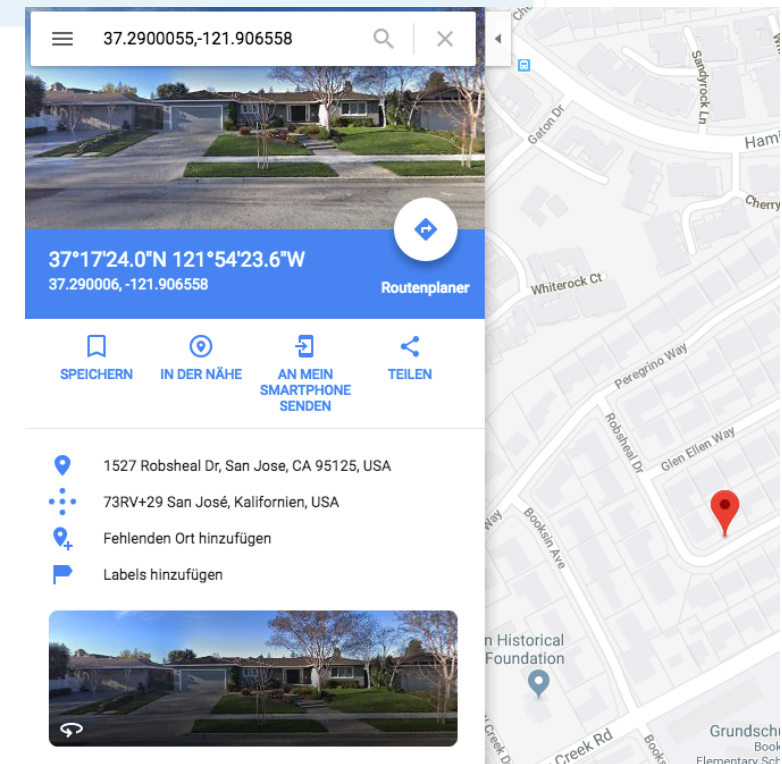
Choosing a location

```
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"
```

Please provide your location longitude/latitude information

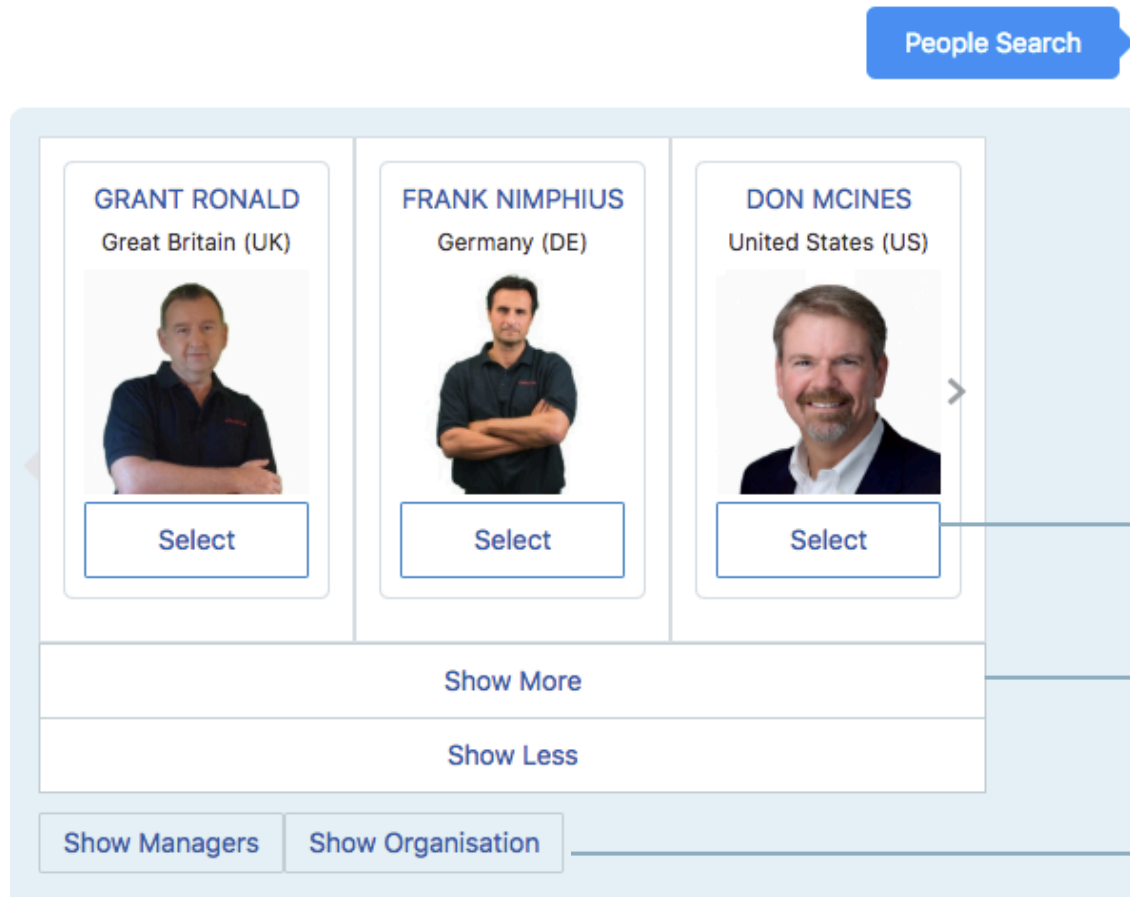
Lookup your location



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

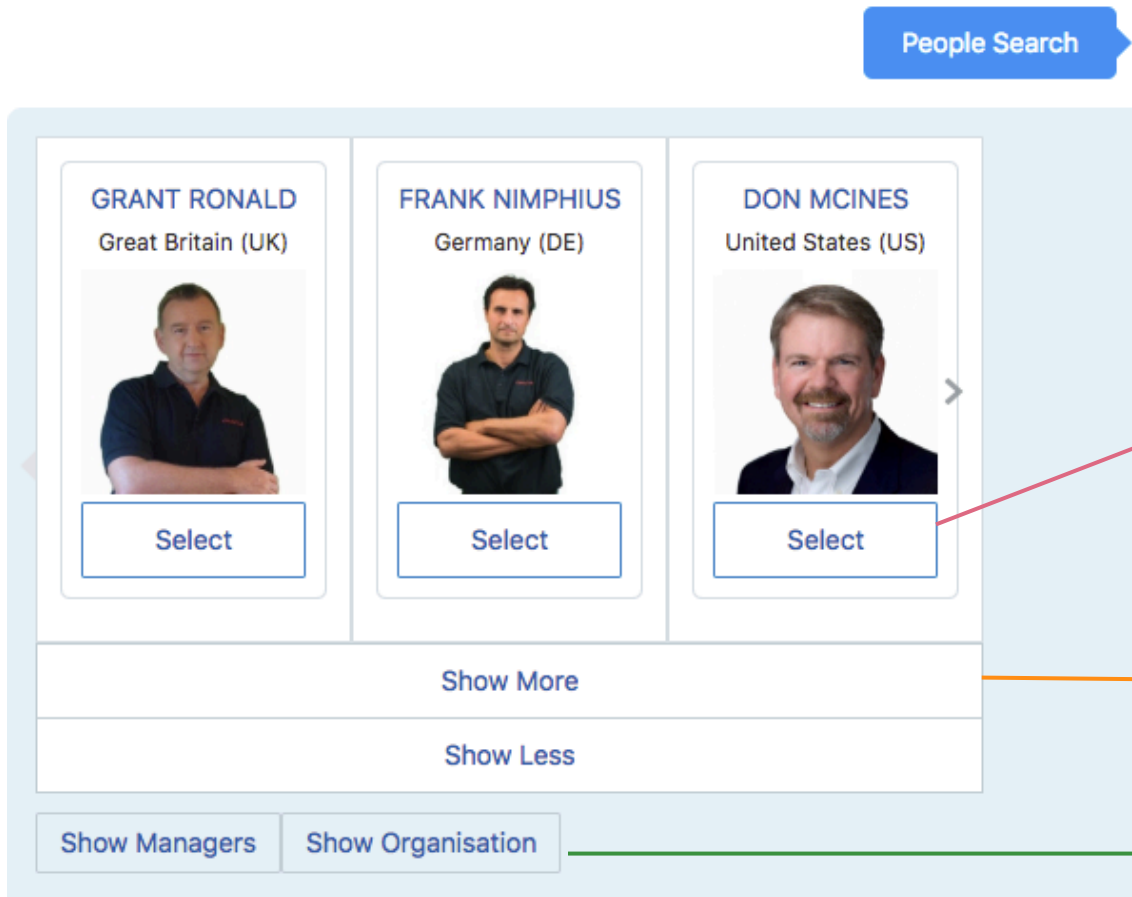


Action items local to a card

Actions local to a response type

Actions global for a component

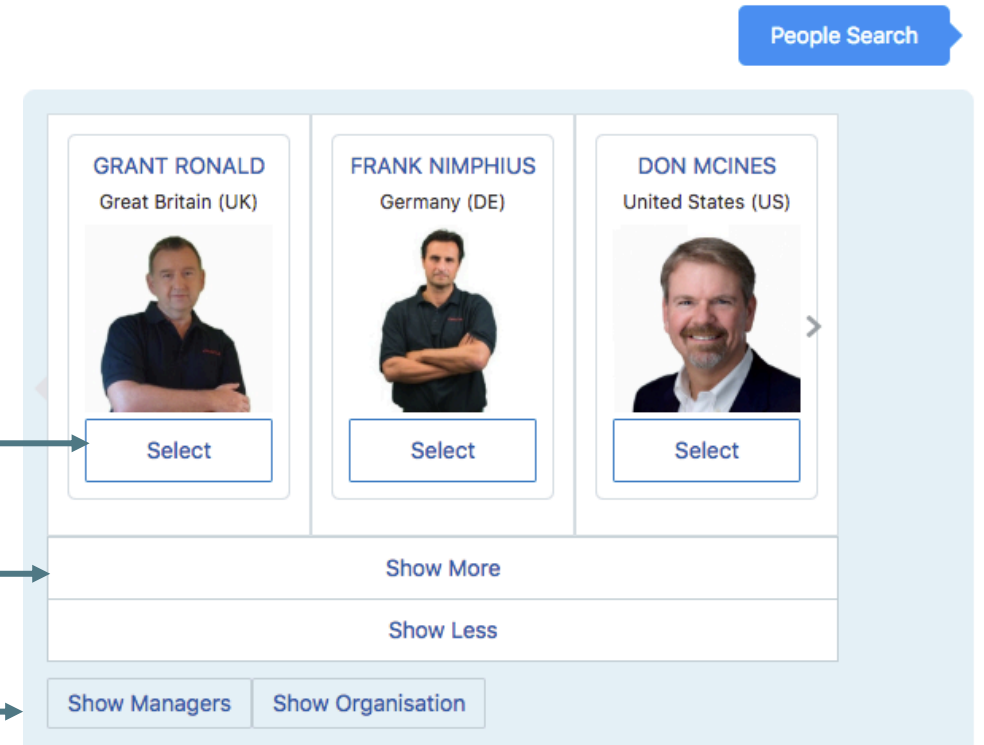
Action types



```
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}"
              - 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

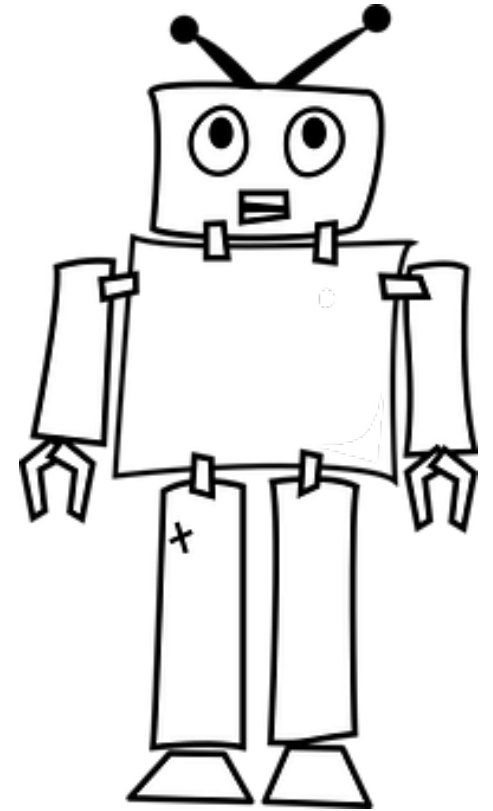
- Action items local to a card
 - Buttons associated with list items or cards
 - Remain visible when component goes out of scope
- Actions local to a response type
 - Buttons associated with component
 - Remains visible when component goes out of scope
- Actions global for a component
 - E.g. quick replies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



Select

FRANK NIMPHIUS
Germany (DE)



Select

DON MCINES
United States (US)



Select

Integrated Cloud

Applications & Platform Services

ORACLE®



Oracle Digital Assistant Hands-On

TBD