
ORACLE®

Oracle Digital Assistant

The Complete Training

Design Practice: Out-of-Order Message Handling

Image courtesy of pixabay.com

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. Users are unpredictable
2. Default implementation
3. Customizing the behavior
4. Display a choice to user
5. Using composite bag entities

Topic agenda

1. Users are unpredictable
2. Default implementation
3. Customizing the behavior
4. Display a choice to user
5. Using composite bag entities

Chatbots do not control the messenger client and therefore **can not prevent users** from scrolling the chat history and **selecting an old action item**.

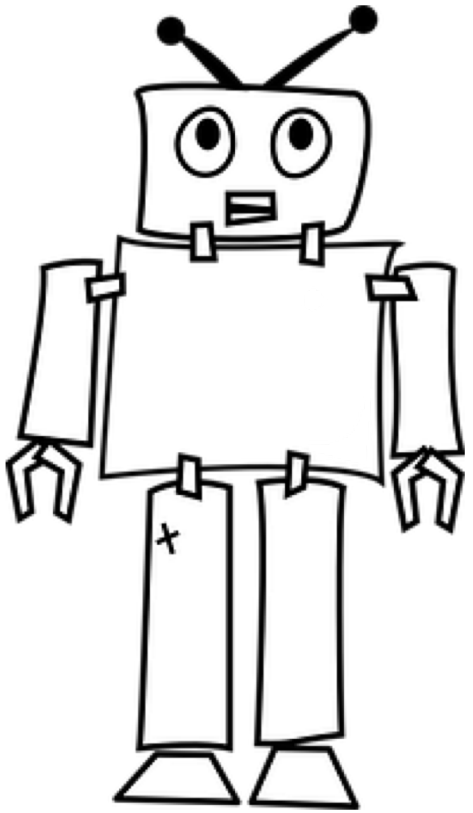
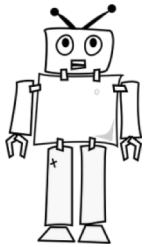
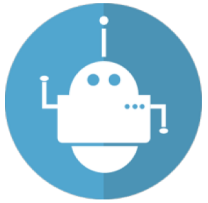


Image courtesy of pixabay.com



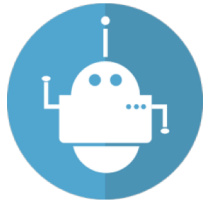
This is what happened to me the other day.

Selects destination airport from list



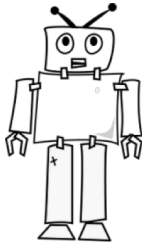
Confirms selection, asks for name on ticket

Enters name to put on ticket



Informs that destination is entitled For free gift voucher. Asks user to choose one.

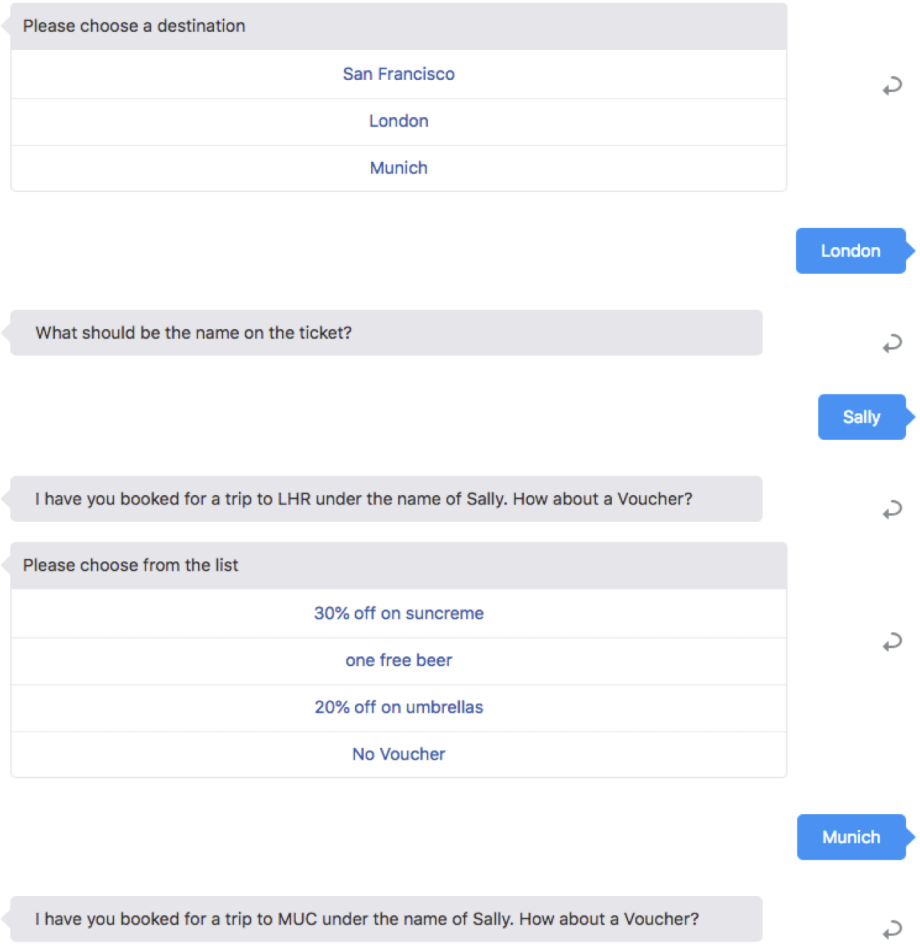
User scrolls chat history. Selects different airport.



Do you see the problem?

Out-of-order message handling

- Posting of old action message is referred to as "out-of-order message"
- Out-of-order messages need to be handled by your design
  - Default implementation exists in Oracle Digital Assistant
- Use cases to consider
  - Allow and follow out-of-order messages
  - Suppress out of order messages
  - Ask user what to do



Topic agenda

1. Users are unpredictable
2. Default implementation
3. Customizing the behavior
4. Display a choice to user
5. Using composite bag entities

Default implementation

- Variables that are referenced in the out-of-order message action will be updated
- Dialog flow navigation continues with the state referenced in the out-of-order-message action
  - Variables are not reset, which means that subsequent states that reference variables that have a value set are skipped

How out-of-order messages are detected

- Dialog flow engine keeps track of conversation state
  - Action items (e.g. buttons) are "tagged" with state token
- For incoming messages, if an invalid state token is detected
  - **${system.actualState}** variable holds name of dialog flow state associated with incoming message
  - **${system.expectedState}** holds name of the current dialog flow state
  - Triggers **system.outOfOrderMessage** action transition

Out-of-order message handling
works for custom components too.
No need to code for it.

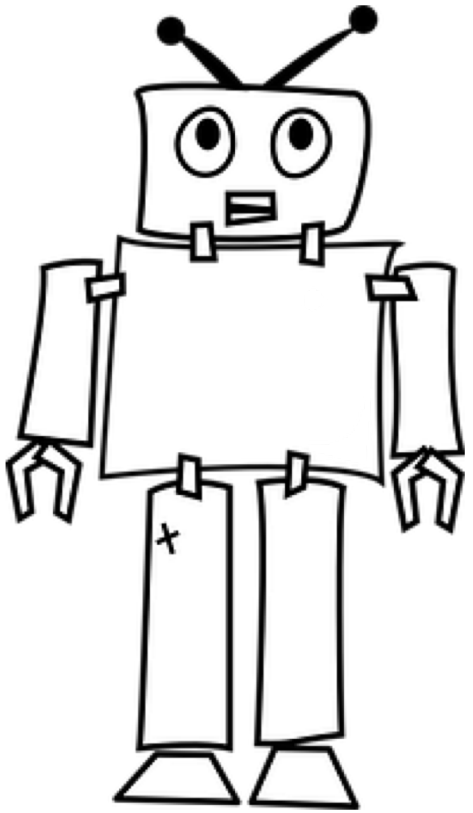


Image courtesy of pixabay.com

Topic agenda

1. Users are unpredictable
2. Default implementation
3. Customizing the behavior
4. Display a choice to user
5. Using composite bag entities

The **search order** for out-of-order message handlers is **current** dialog flow state, then **global default transition** and finally **the default** implementation

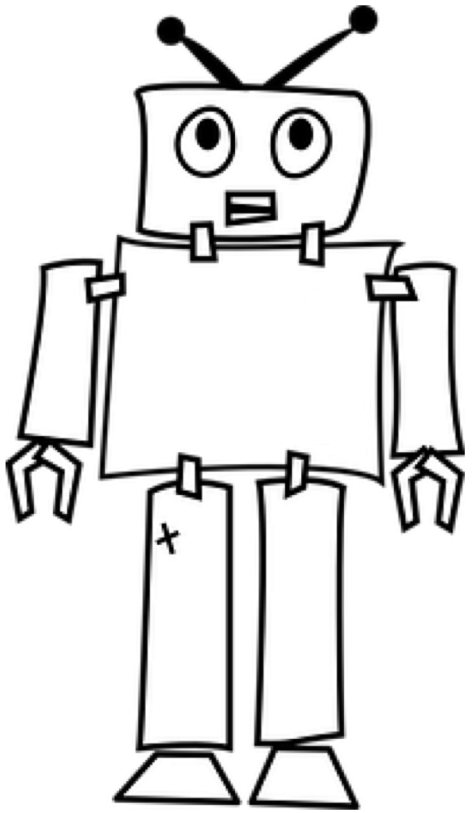
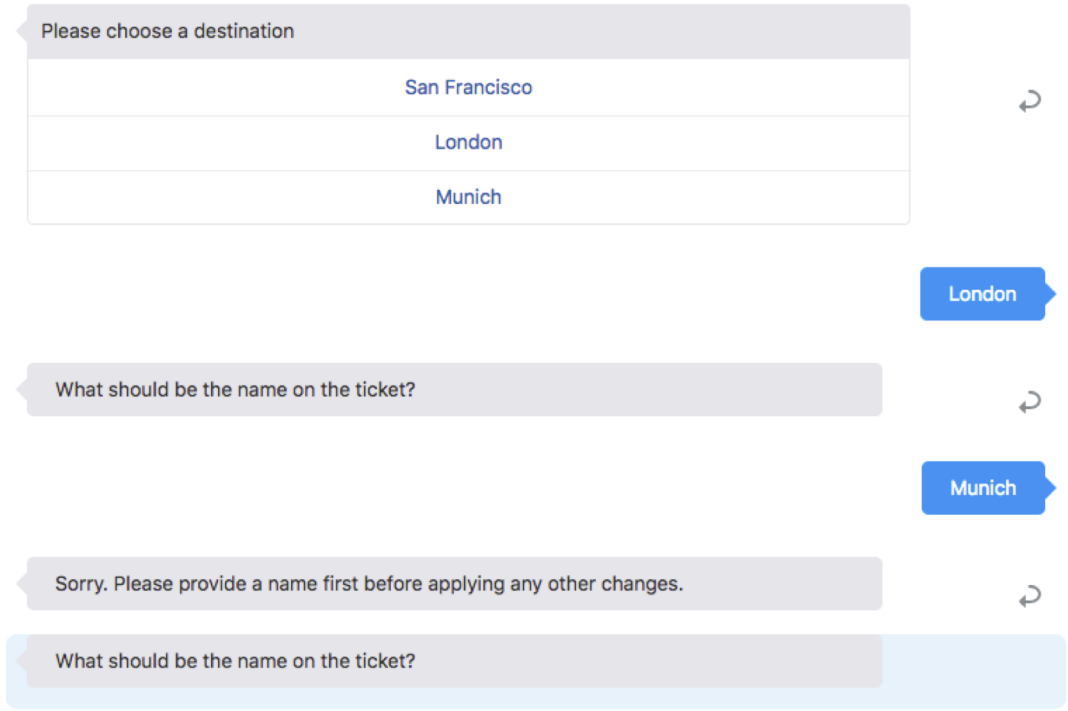


Image courtesy of pixabay.com

Defining System.outOfOrderMessage on a state

```
addName:
  component: "System.Text"
  properties:
    prompt: "What should be the name on the ticket?"
    variable: "name"
  transitions:
   next: "checkVoucher"
   actions:
     system.outOfOrderMessage: "provideNameFirst"

provideNameFirst:
  component: "System.Output"
  properties:
    text: "Sorry. Please provide a name first before applyin
    keepTurn: true
  transitions:
    next: "resetNameVar"
```



Define system.outOfOrderMessage as a default transition

```
context:
  variables:
    iResult: "nlpresult"
    location: "string"
    name: "string"
    voucher: "string"

defaultTransitions:
  actions:
    system.outOfOrderMessage: "${system.actualState}"

states:
  displayLocations:
    component: "System.CommonResponse"
```

- Default transitions define global action handlers
  - *system.outOfOrderMessage* action handles out-of-order messages
- Setting *${system.actualState}* as a value implements the default implementation behavior to the global handler

Defining a custom Out-Of-Order message handler

```
defaultTransitions:
  actions:
    system.outOfOrderMessage: "startOutOfOrderMessageHandling"
```

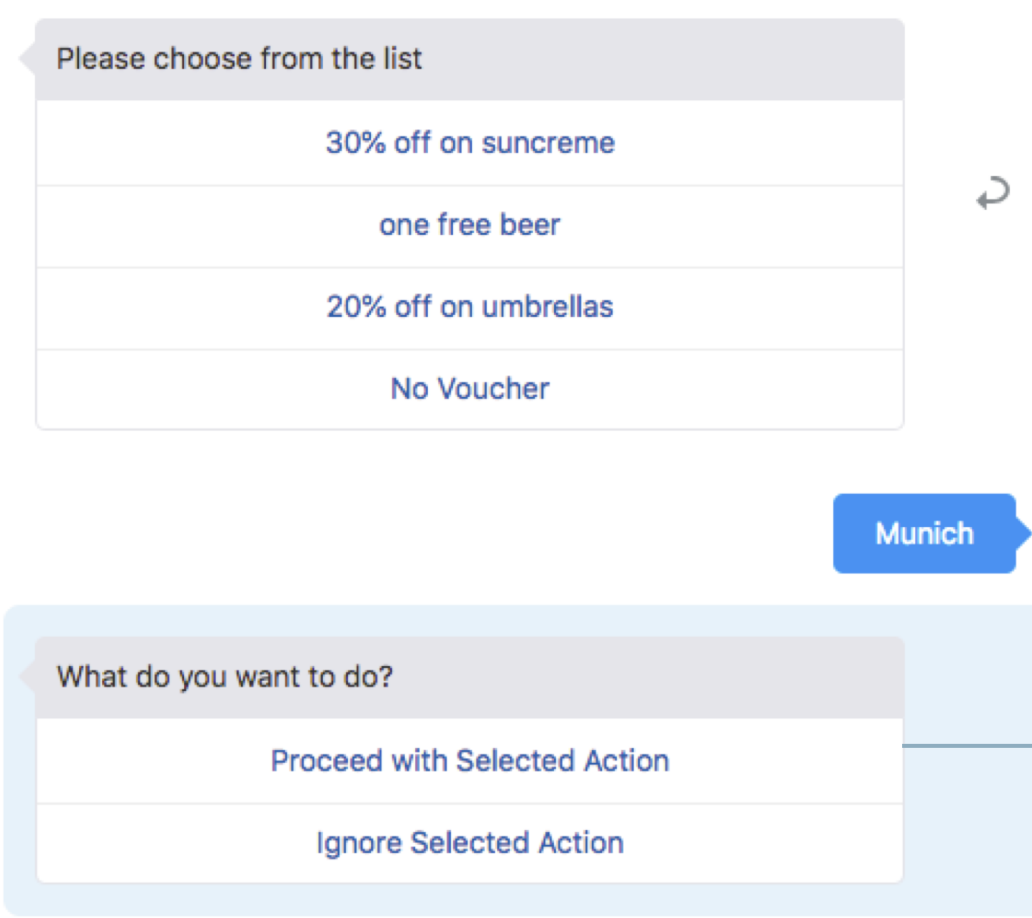
```
startOutOfOrderMessageHandling:
  component: "System.Switch"
  properties:
    source: "${system.actualState}"
    values:
    - "displayLocations"
    - "getVoucher"
  transitions:
    actions:
      displayLocations: "saveOutOfOrderValue"
      getVoucher: "invalidOperation"
      NONE: "${system.actualState}"
```

- Switch component checks the actualState variable value
  - Allows you to handle out-of-order messages specifically for a state (states may require different handling)
  - Allows defining a custom default implementation

Topic agenda

1. Users are unpredictable
2. Default implementation
3. Customizing the behavior
4. Display a choice to user
5. Using composite bag entities

Conditionally display a choice




```
outOfOrderMessageHandler:
  component: "System.List"
  properties:
    prompt: "What do you want to do?"
    options:
    - label: "Proceed with Selected Action"
      value: "continueOutOfOrderFlow"
    - label: "Ignore Selected Action"
      value: "continueCurrentFlow"
  transitions:
    actions:
      continueOutOfOrderFlow: "continueOutOfOrderFlow"
      continueCurrentFlow: "${system.expectedState}"
```

```
continueOutOfOrderFlow:
  component: "System.Switch"
  properties:
    source: "${system.actualState}"
    values:
    - "displayLocations"
  transitions:
    actions:
      displayLocations: "prepareNavigationToDisplayLocation"
      NONE: "${system.actualState}"
```

Out-of-order message handling is an integral part of your overall **skill bot design**.

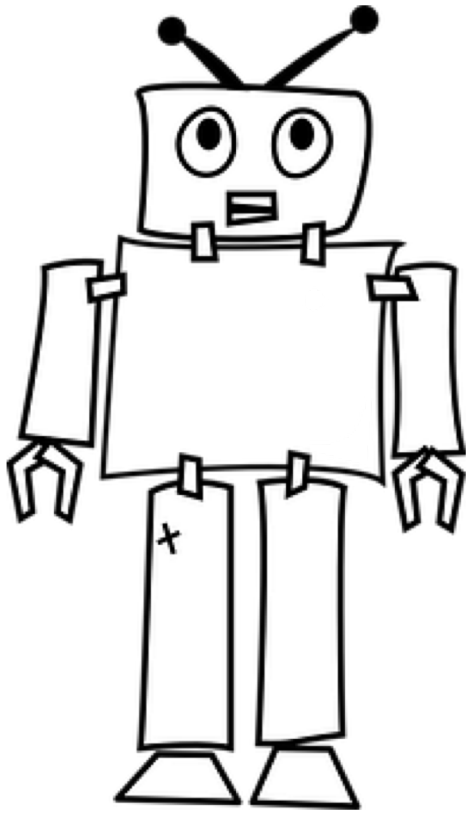


Image courtesy of pixabay.com

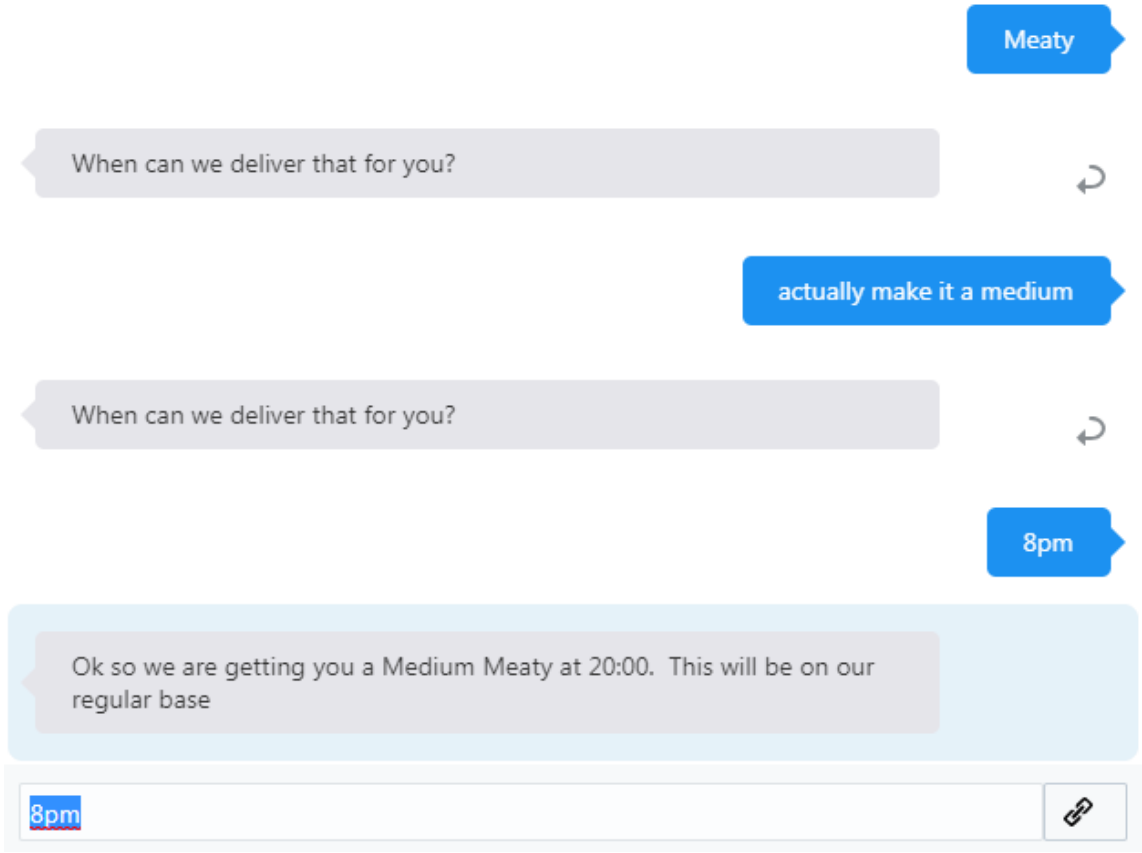
Topic agenda

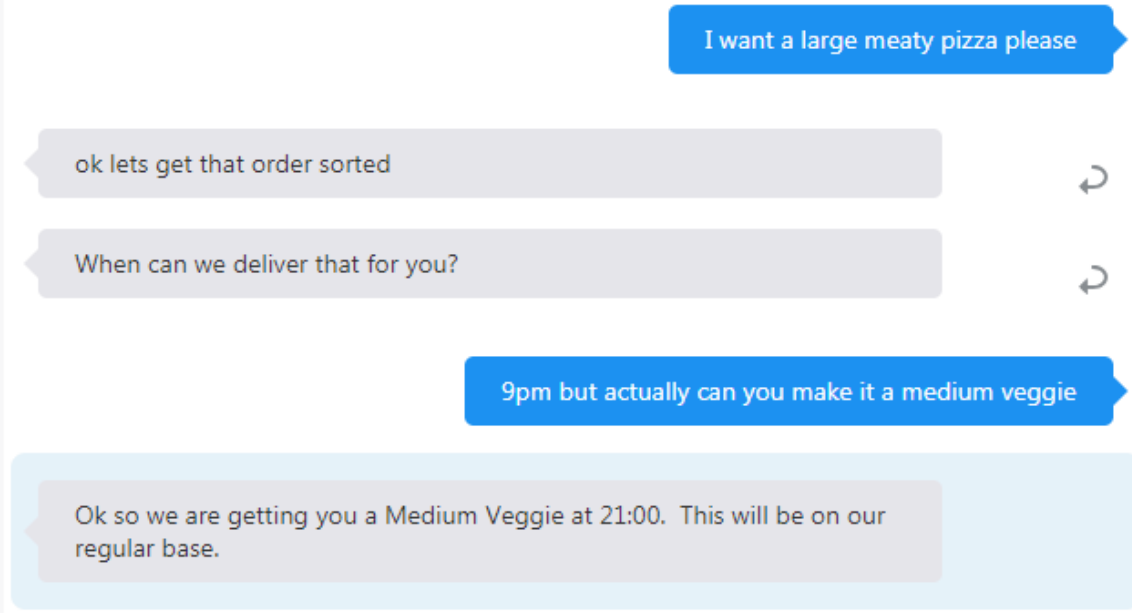
1. Users are unpredictable
2. Default implementation
3. Customizing the behavior
4. Display a choice to user
5. Using composite bag entities

About composite bag entities

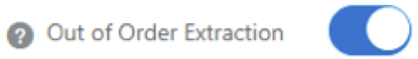
- Models a business domain object
  - Pizza order, holiday request, expense
- Each composite bag is composed of one to many items
  - Custom entities
  - Built-in entities
  - String, location and attachment
- All contained entities get resolved automatically in a single dialog flow state
  - System.ResolveEntities
  - System.CommonResponse

Composite bag entity example





Out-of-order message handling

- Out-of-order messages happening in the context of resolving a composite bag entities can be resolved automatically
- Each bag item in a composite bag entity has an "Out of Order Extraction" toggle 

  - Enable toggle to allow out of order extraction
  - Disable to only resolve bag item values when prompted

Composite bag entities resolve out-of-order messages for **all actions** that belong to the composite bag entity, while the focus is **on the state in which the entity resolves**.

Any other out-of-order messages are handled by the dialog flow.

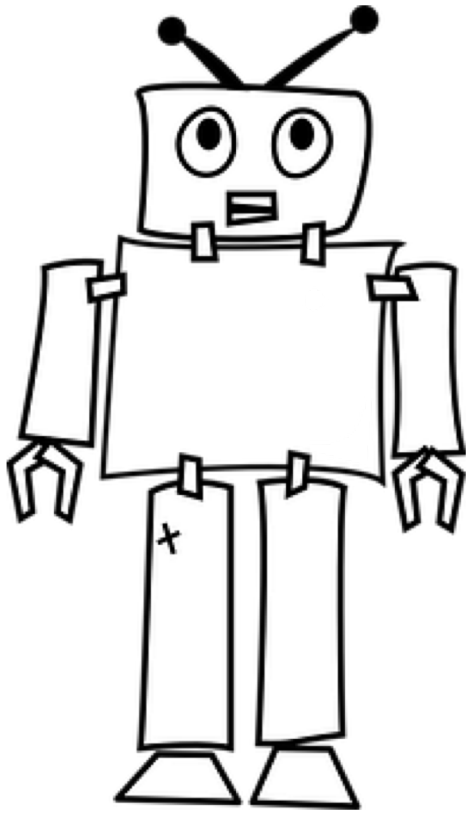


Image courtesy of pixabay.com

Integrated Cloud

Applications & Platform Services


ORACLE®

Oracle Digital Assistant Hands-On

TBD