

ORACLE®

Oracle Digital Assistant

The Complete Training

Design Practice: Error Handling Strategies

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

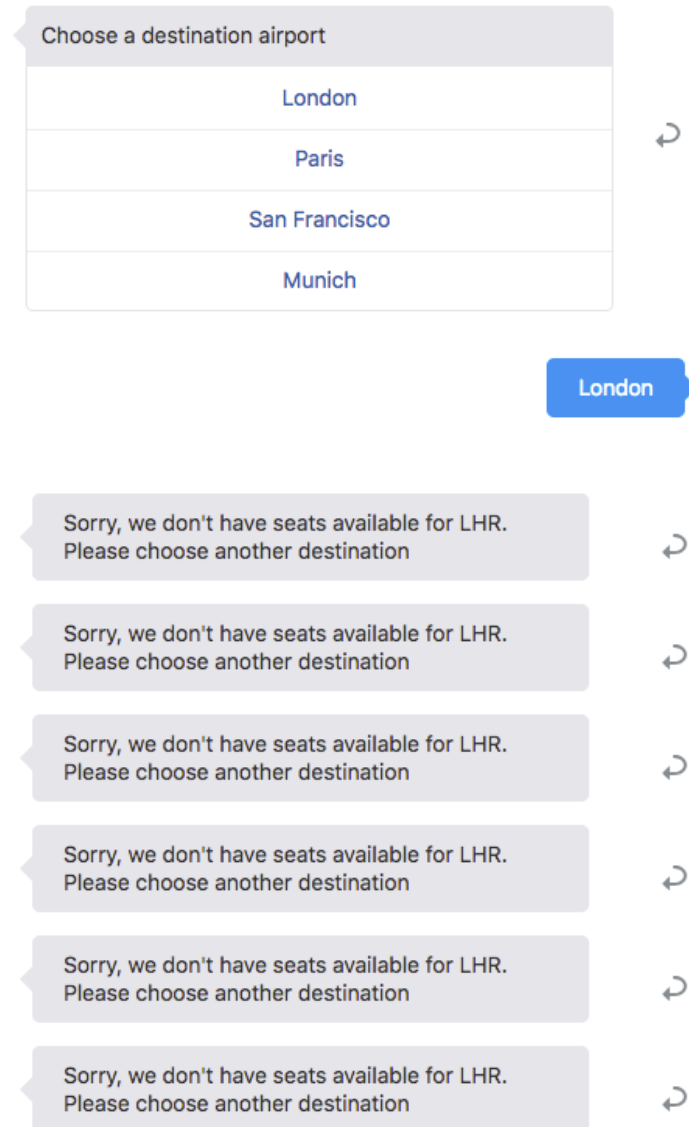
- 1 Runtime Errors
- 2 Design Time Errors

Topic agenda

- 1 Runtime Errors
- 2 Design Time Errors

Runtime errors

- Usually occur in skills
- Component exceptions
 - Misconfiguration (missing required properties)
 - Syntax errors in Apache FreeMarker expressions
 - Custom component runtime exceptions
 - Missing dependencies, parsing exceptions etc.
- Navigation errors
 - Session is in infinite loop
 - Navigation goes to an unexpected state



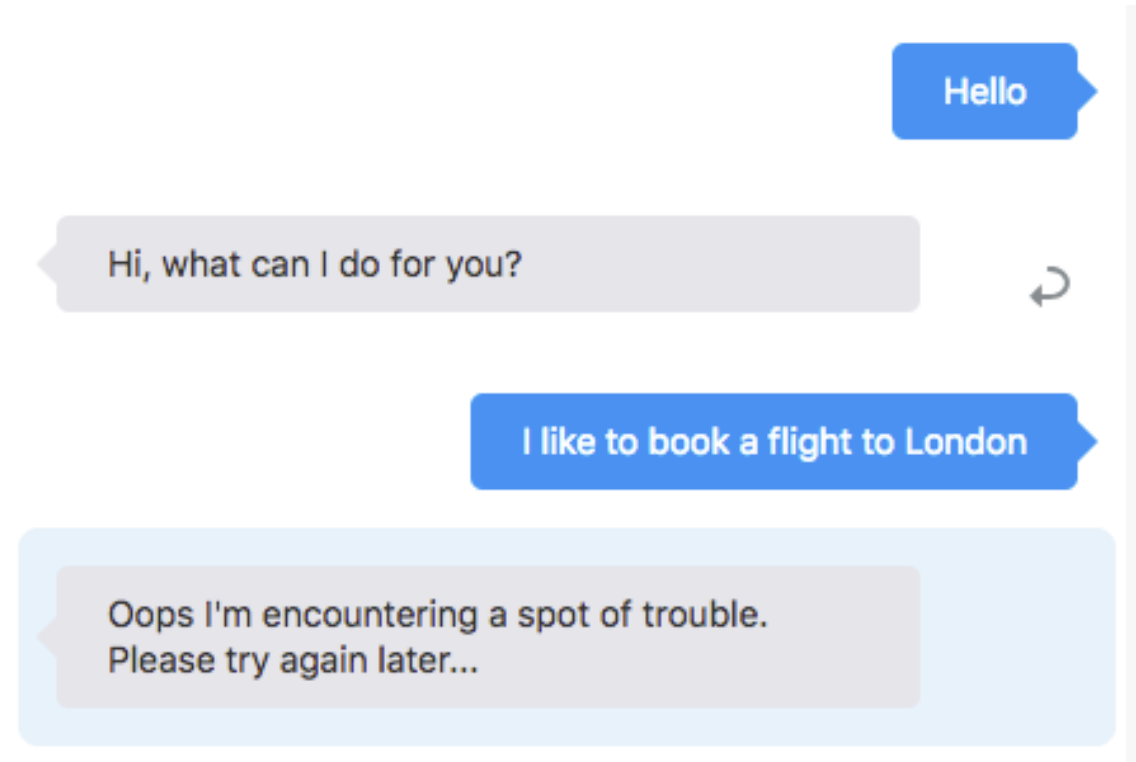
Runtime

Component exceptions

Component exceptions

```
start:
  component: "System.Output"
  properties:
    text: "Hi, what can I do for you?"
    keepTurn: false
  transitions:
    next: "getUserIntent"

getUserIntent:
  component: "System.Intent"
  properties:
    variable: 
  transitions:
    actions:
      Booking: "bookFlight"
      ChangeReservation: "changeReservation"
      unresolvedIntent: "handleUnresolved"
```



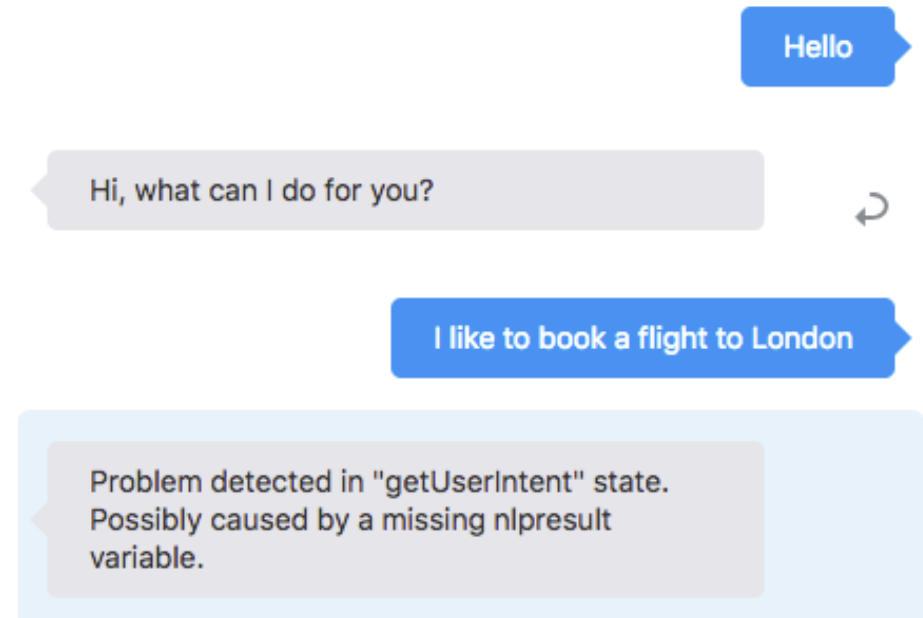
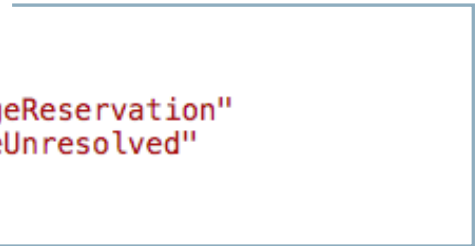
About component exceptions

- Component exceptions are errors in a system or custom component
 - Missing required properties
 - Runtime exceptions caused by the component logic
- Component errors can be handled by bot designers
 - Using the error transition defined on a dialog flow state
 - Globally, by setting an error transition as a defaultTransition
- If no error transition is defined then the system error handler is used

Error handling on a individual states

- Handles error for a single state
- Uses "error" transition defined on state

```
getUserIntent:  
  component: "System.Intent"  
  properties:  
    variable:  
  transitions:  
    error: "handleIntentError"  
    actions:  
      Booking: "bookFlight"  
      ChangeReservation: "changeReservation"  
      unresolvedIntent: "handleUnresolved"  
  
handleIntentError:  
  component: "System.Output"  
  properties:  
    text: "Problem detected in \"getUserIntent\" state.  
          Possibly caused by a missing nlresult variable."  
  transitions:  
    return: "done"
```



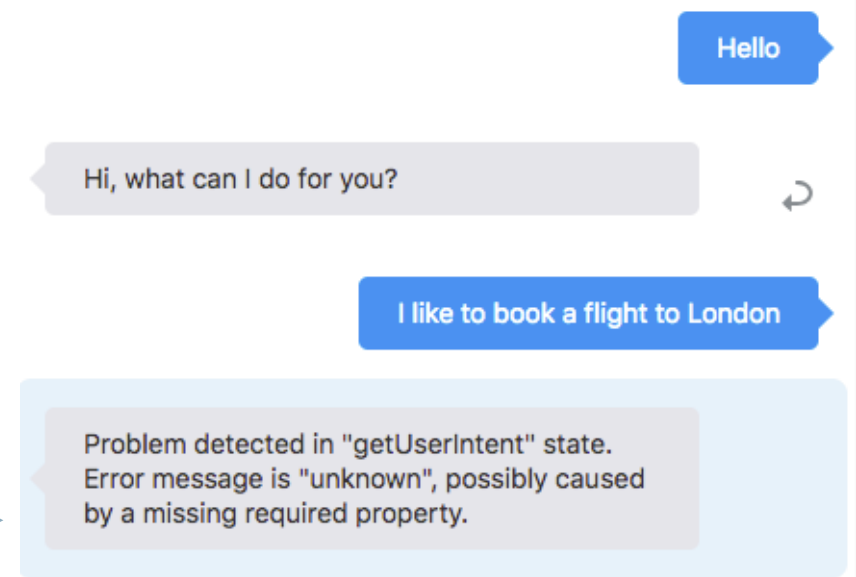
Global custom error handler

- Uses error transition defined in defaultTransitions section
- Handles errors for all states without an error transition

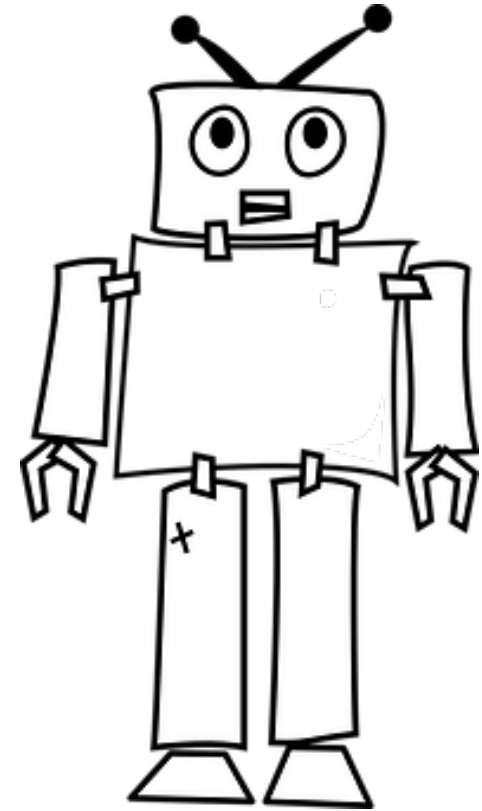
```
defaultTransitions:
  error: "globalCustomErrorHandler"

states:
  start:
    component: "System.Output"
    properties:
      text: "Hi, what can I do for you?"
      keepTurn: false
    transitions:
      next: "getUserIntent"
  getUserIntent:
    component: "System.Intent"
    properties:
      variable:
    transitions:
      actions:
        Booking: "bookFlight"
        ChangeReservation: "changeReservation"
        unresolvedIntent: "handleUnresolved"

globalCustomErrorHandler:
  component: "System.Output"
  properties:
    text: "Problem detected in \"${system.errorState}\" state.
          Error message is \"${system.errorMessage! 'unknown'}\",
          possibly caused by a missing required property.'"
  transitions:
    return: "done"
```



Components exceptions show during testing. **Components exceptions** that appear in production **often indicate a problem with testing.**



Error handling strategies

- Avoid errors propagating to the system error handler
 - Bad user experience
- Avoid errors in the error handler
 - Be conservative when handling errors
- Use a single location for your error handling
 - Move to bottom of dialog flow so its easy to find
 - Doesn't clutter dialog flow states with error handling routines
 - Use the global error handler with System.Switch component
 - Use local error handling by exception

Using a single location for all error handling

- `${system.errorState}` holds name of dialog flow state that caused an error
- `System.Switch` compares the name in the error state with state names that bot developer wants to perform local error handling for
- If `System.Switch` does not find a matching state value for its error state, then `NONE` is followed
 - `NONE` directs to a generic (global) error handler

```
defaultTransitions:  
  error: "globalErrorHandler"
```



```
globalErrorHandler:  
  component: "System.Switch"  
  properties:  
    source: "${system.errorState}"  
  values:  
    - "getOrderStatus"  
    - "displayOrderStatus"  
    - "createOrder"  
  transitions:  
    actions:  
      getOrderStatus: "handleOrderStatusError"  
      displayOrderStatus: "handleOrderStatusError"  
      createOrder: "handleOrderStatusError"  
      NONE: "UnhandledErrorToHumanAgent"
```

Runtime

Navigation errors

Infinite Loop

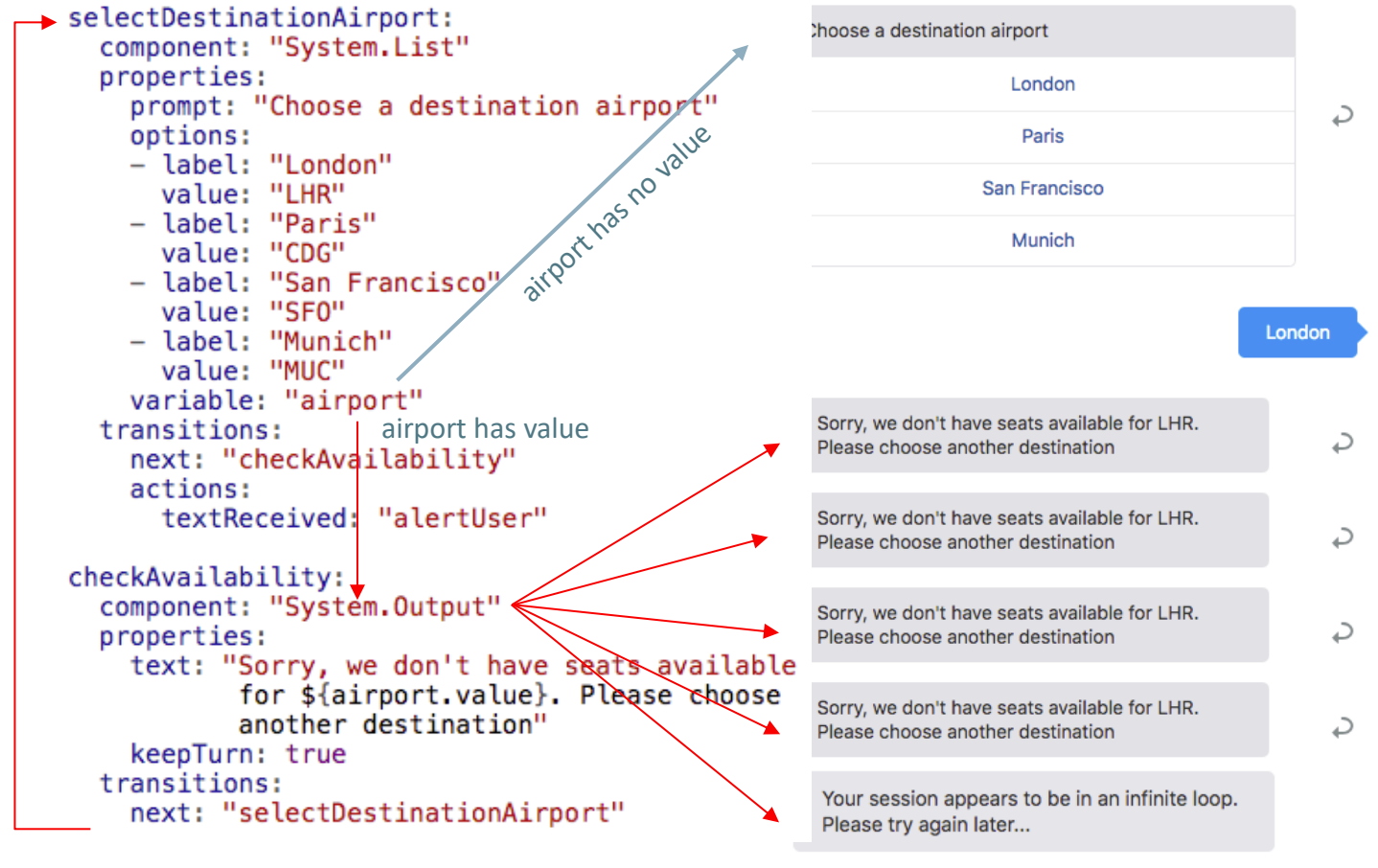
- Typically shows during design-time and testing
 - Reoccurring navigation without user input
 - Usually indicates
 - "keepTurn = true" or similar issue
 - Forgetting to reset variable states
 - Custom component problem
- Bot throws error after 100 responses sent without user interaction
 - Not configurable
 - Prevents bot from hanging itself



Your session appears to be in an infinite loop. Please try again later...

Example

- User selects airport
 - Airport saved in "airport" variable
- Bot detects no seats available
 - Bot redirects user to select another airport
- List component is not rendered because airport variable is not empty
 - Developer forgot to reset variable



The problem displayed in the conversation tester

Testing Skill

Reset Conversation Intent/Q&A JSON

Download JSON Enter a key or value to search in JSON

Hi

States

Choose a destination airport

London
Paris
San Francisco
Munich

London

Sorry, we don't have seats available for LHR. Please choose another destination

Sorry, we don't have seats available for LHR. Please choose another destination

Sorry, we don't have seats available for LHR. Please choose another destination

Sorry, we don't have seats available for LHR. Please choose another destination

Sorry, we don't have seats available for LHR. Please choose another destination

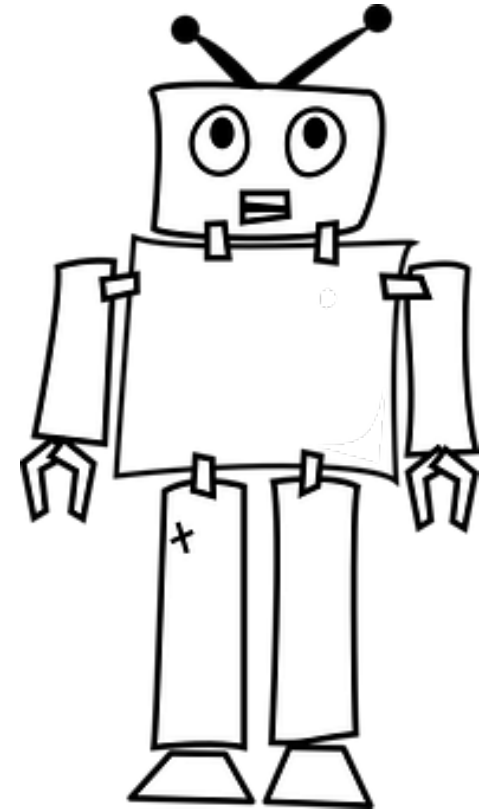
InfiniteLoop Skill

```
graph TD; A((selectDestinationAirport)) --> B((checkAvailability)); B --> C((System.MaxStatesExceededHandler));
```

Variables

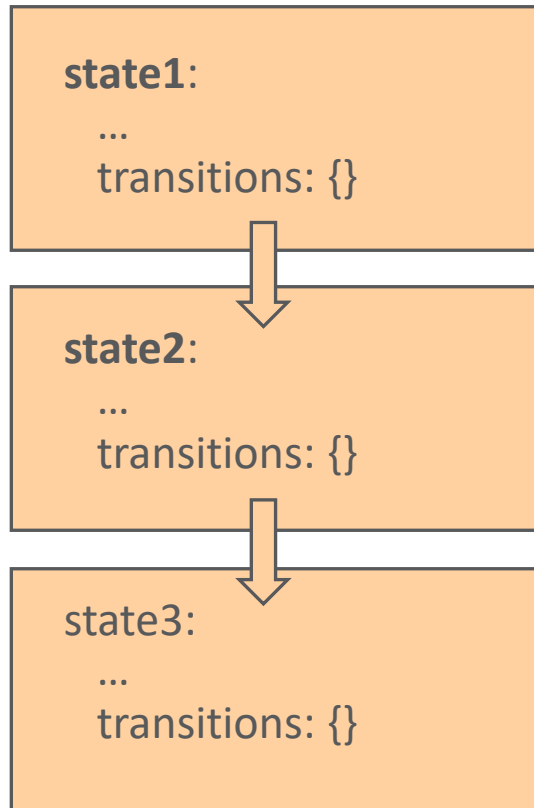
```
{  "currentDialogState": "System.MaxStatesExceededHandler",  "executedStates": [    {      "stateName": "selectDestinationAirport",      "botName": "InfiniteLoop",      "botId": "8B58C99E-05B8-41D0-9438-CD41CA17C3A3"    },    {      "stateName": "checkAvailability",      "botName": "InfiniteLoop",      "botId": "8B58C99E-05B8-41D0-9438-CD41CA17C3A3"    },    {      "stateName": "selectDestinationAirport",      "botName": "InfiniteLoop",      "botId": "8B58C99E-05B8-41D0-9438-CD41CA17C3A3"    },    {      "stateName": "checkAvailability",      "botName": "InfiniteLoop",      "botId": "8B58C99E-05B8-41D0-9438-CD41CA17C3A3"    },    {      "stateName": "selectDestinationAirport",      "botName": "InfiniteLoop",      "botId": "8B58C99E-05B8-41D0-9438-CD41CA17C3A3"    },    {      "stateName": "checkAvailability",      "botName": "InfiniteLoop",      "botId": "8B58C99E-05B8-41D0-9438-CD41CA17C3A3"    }  ]}
```

Empty transitions often are a recipe for errors and can **lead to unexpected navigation**



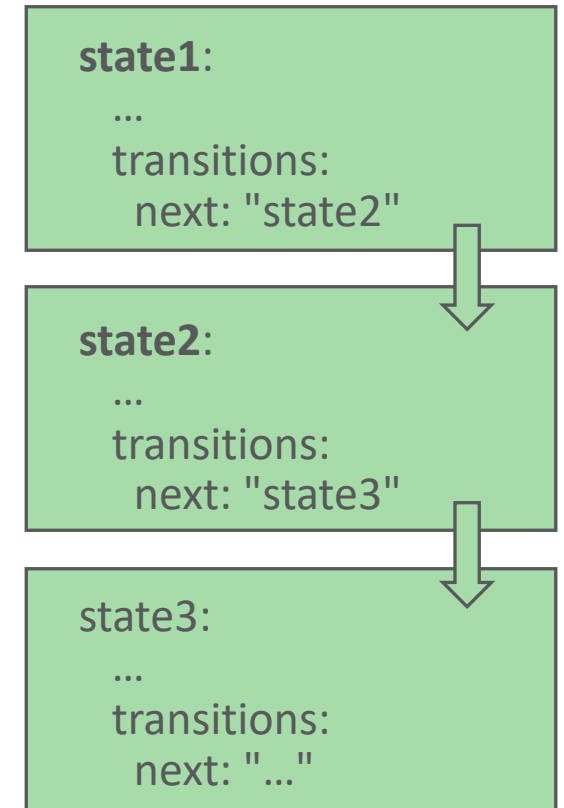
Empty navigation vs. next transition

Empty navigation



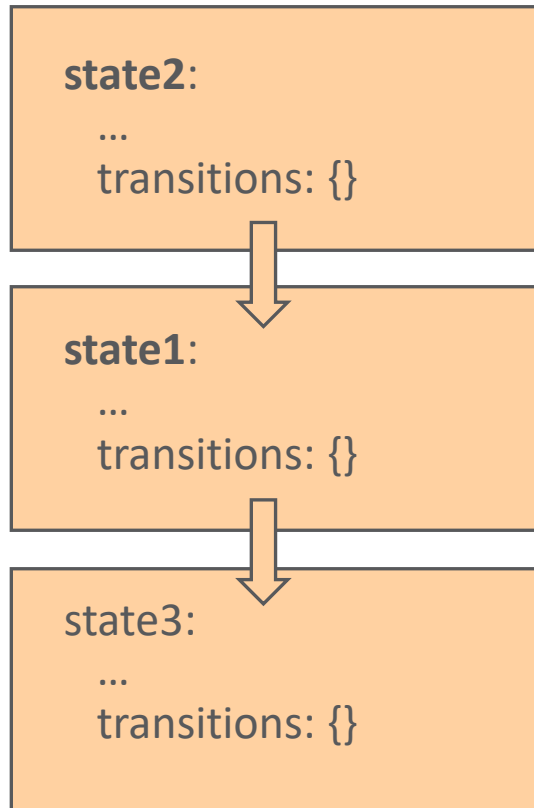
What's the difference?

Next transition



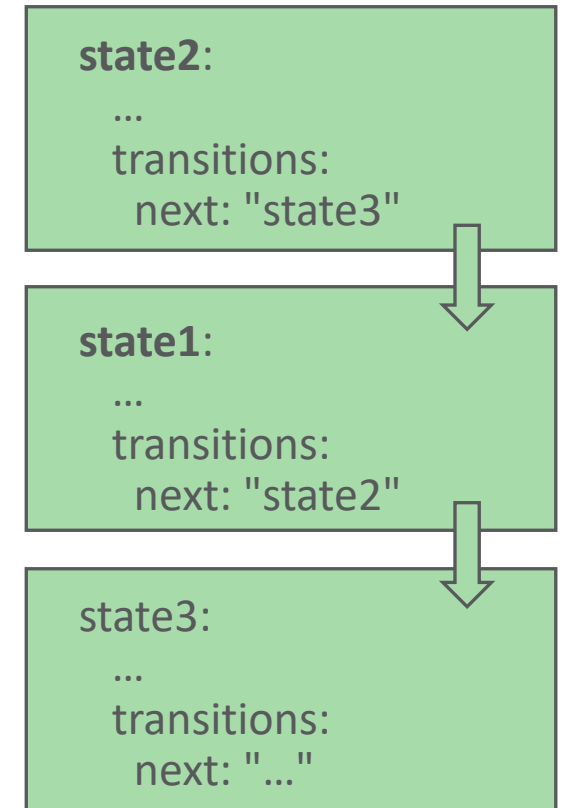
Empty navigation vs. next transition

Empty navigation



What's the difference **now**?

Next transition



Always define a "next" transition

- Prevent unexpected dialog flows
- Define "next" transition on every state
 - Even if you don't need it
 - Requires discipline
- Define "next " as a global transition
 - Use for development and testing only
 - Catches all empty navigation
 - Conversation tester displays executed states
 - Makes it easy to find state with missing next transition

```
defaultTransitions:  
  error: "globalCustomErrorHandler"  
  next: "alertEmptyTransition"  
  
states:  
  start:  
    component: "System.Output"  
    properties:  
      text: "Hi, what can I do for you?"  
      keepTurn: true  
  
  getUserIntent:  
    component: "System.Intent"
```



Testing Skill

Reset

Conversation Intent/Q&A JSON

Download JSON

Enter a key or value to search

Hello

Hi, what can I do for you?

Detected empty transition. Check embedded tester for details

```
{  
  "text": "Detected empty transition. Check embedded  
  tester for details",  
  "type": "text",  
  "channelExtensions": {  
    "debugInfo": {  
      "variables": {  
        "profile.updatedOn": "2019-02-19T12:41:21.  
        "profile.timezoneOffset": -3600000,  
        "system.config.system.intentConfidenceWinM  
0,  
        "profile.lastName": "Chatbot",  
        "system.config.system.startState": "",  
        "iResult": null,  
        "profile.firstName": "Admin",  
        "system.config.system.helpState": "",  
        "system.config.system.welcomeState": "",  
        "system.config.system.intentConfidenceThre  
0,  
        "profile.locale": "en_US"  
      },  
      "currentDialogState": "alertEmptyTransition"  
    },  
    "executedStates": [  
      {  
        "stateName": "start",  
        "botName": "ComponentErrorHandler",  
        "botId": "B16251C2-AF1B-445C-BD27-DF22DB  
      },  
      {  
        "stateName": "alertEmptyTransition",  
        "botName": "ComponentErrorHandler",  
        "botId": "B16251C2-AF1B-445C-BD27-DF22DB  
    ]  
  }  
}
```

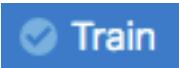
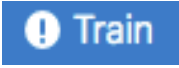

Topic agenda

- 1 Runtime Errors
- 2 Design Time Errors

Design time errors

- Forgetting to train the model
- Missing required component properties
- Invalid code indenting
 - Two blanks per hierarchy
- Hidden characters after copying and & pasting content
- Missing or duplicated quotes
- Invalid expression syntax
- Setting a semicolon(';') at the end of an expression

How-to find the trouble maker(s)

- Ensure model is trained if needed
 -  
- Does the intent tester work?
 - If so, then chances are that a System.Intent action references a state that doesn't exist or misses the nlresult variable reference
- Exclude formatting errors
 - Use validation button
 - Use debugger
- If you can't see an error on the indicated state, how about the state below?



How-to find the trouble maker(s)

Use the **JSON conversation log** in the embedded tester to find exceptions and their cause



Using the embedded conversation tester

Testing Skill

Reset Conversation Intent/Q&A View JSON

I like to order pizza

How old are you?

25

Oops I'm encountering a spot of trouble. Please try again later...

States

FNPIzza Skill

resolvesize

resolvecrust

resolvetype

askage

crust

System.DefaultErrorHandler

Variables

system

View JSON

Enter a key or value to search in JSO

```
"profile.timezoneOffset": -3600000,
"size": null,
"system.config.system.intentConfidenceWinMargin":
0.1,
"system.config.system.errorUnexpectedErrorPrompt":
"Oops I'm encountering a spot of trouble. Please try again
later...",
"system.config.system.startState": "",
"system.config.system.errorExpiredSessionPrompt":
"Your session has expired. Please start again.",
"profile.firstName": "frank.nimphius@oracle.com",
"system.postbackActions": {},
"system.config.system.welcomeState": "",
"system.config.system.errorMaxStatesExceededPrompt": "Your
session appears to be in an infinite loop."
},
"errorState": "crust",
"errorMessage": "The component System.Lists was not
found.",
"currentDialogState": "System.DefaultErrorHandler",
"executedStates": [
{
"stateName": "crust",
"botName": "FNPIzza",
"botId": "058BB32E-B67D-451A-A4A1-F73903F8A70F"
},
{
"stateName": "System.DefaultErrorHandler",
"botName": "FNPIzza",
"botId": "058BB32E-B67D-451A-A4A1-F73903F8A70F"
}
]
}
```

Introspect variables

The screenshot shows a chatbot conversation with three questions and their corresponding user inputs:

- Question: "How old are you?" Input: 25
- Question: "What crust do you want for your Pizza?" Input: Thick
- Question: "What size Pizza do you want?" Input: Medium

The Variables inspector on the right displays the following data:

```
profile.updatedOn: 2019-01-22T17:34:48.659Z
profile.lastName: Unspecified
profile.locale: en_US
crust: thick
profile.timezoneOffset: -3600000
size: Medium
profile.firstName: frank.nimphius@oracle.com
userInput
└─ system
   └─ config
      └─ postbackActions
         ├── small
         ├── large
         ├── personal
         └─ medium
            └─ postback
               label: Medium
               type: postback
```

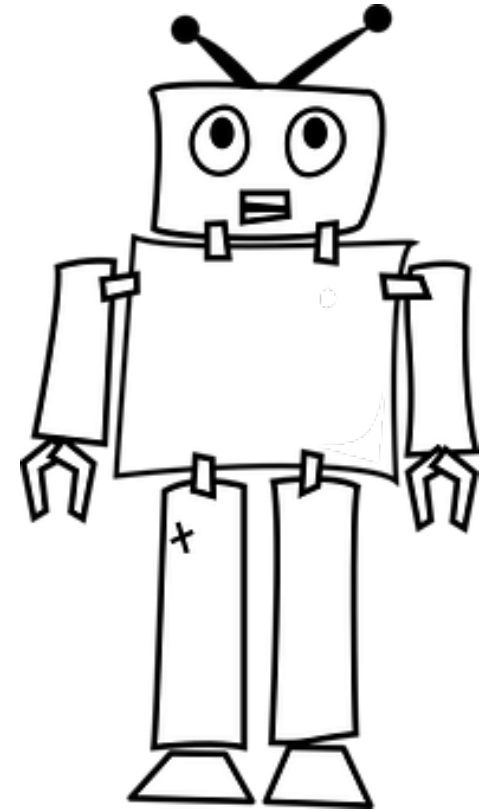
Arrows indicate the mapping from user input to variables: "25" to `profile.updatedOn`, "Thick" to `crust`, and "Medium" to `size`.

Variables inspector shows

- Context variables and their values
- System variables
 - Profile settings (set by messenger)
 - User name, Locale, Time offset
 - Skillbot parameters (config)
 - Actions settings and postback payloads for menu lists
 - System properties set when using
 - composite bag entities
 - Common response component
 - Resolve entities component

The only source of knowledge
is **experience**.

- Albert Einstein



Integrated Cloud

Applications & Platform Services

ORACLE®



Oracle Digital Assistant Hands-On

TBD