ORACLE

Dialog design considerations

Session Agenda

1 Implement a modular design

2 How to end flows

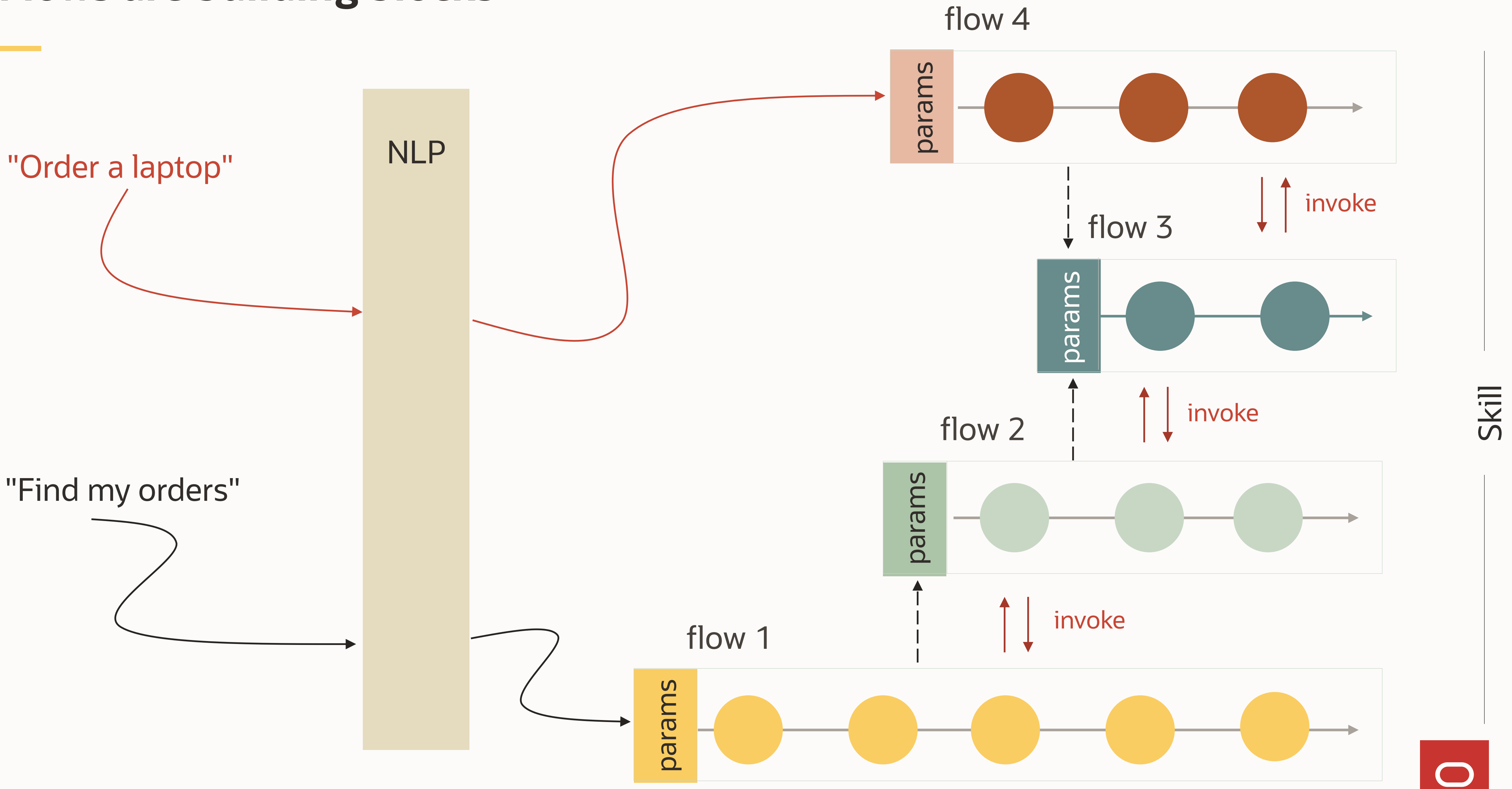
3 Use a naming convention

4 Handle errors

5 Practice design by validation

Session Agenda

Flows are building blocks



How many flows do you need?

> Don't worry about the number of flows

Intent flows

- Mapped to one or many intents
- Control flow for the task to accomplish

Services flows

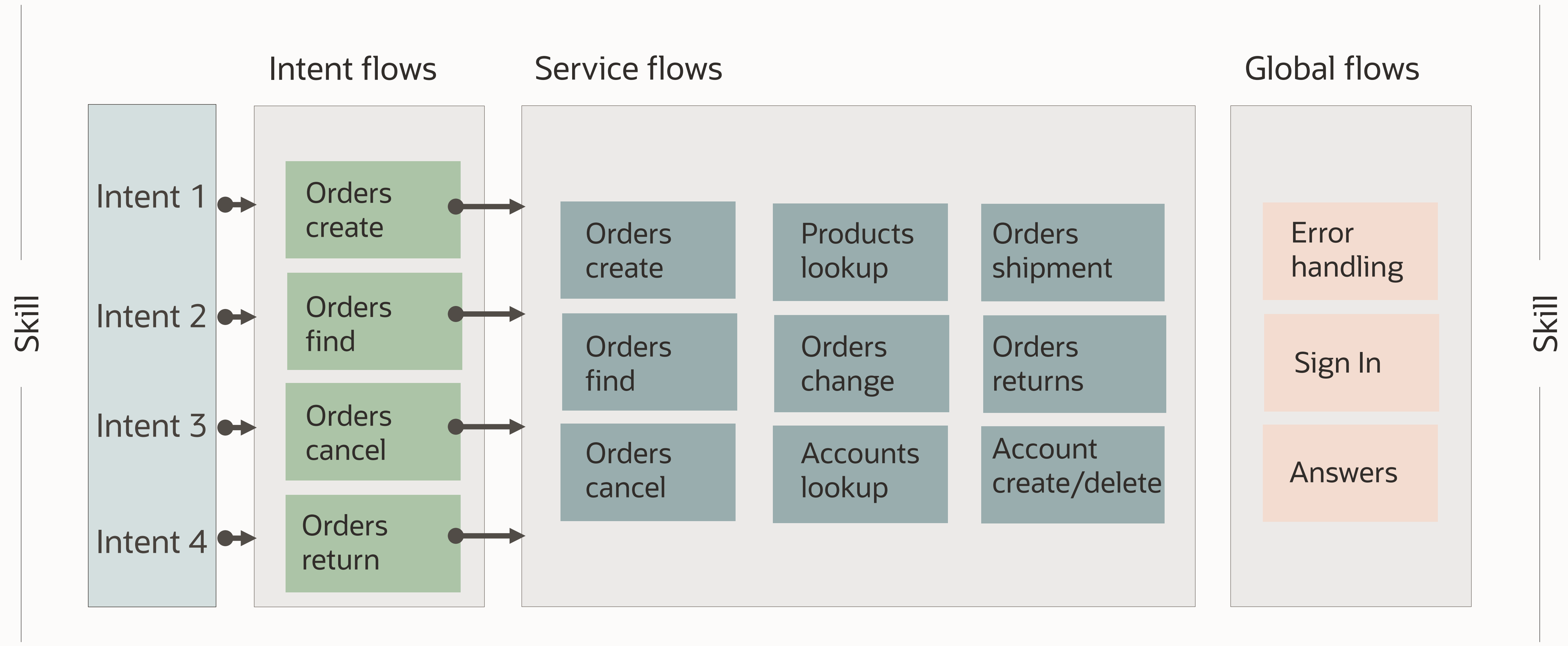
- Provides reusable functionality
- Not aware of the user task

Global flows

- Flows mapping to built-in events

Image courtesy of pixabay.com

Modular dialog flow architecture



How to size flows

Don't think about sizing. Just keep flows small and simple

- Easy to read and understand

If a flow becomes large, review it for what part could be moved into its own service flow

Consider using composite bag entities to interact with users for a given piece of information

- A single state that could have multi-turn conversations
- Allows you to skip prompts if users provide information out-of-order

Build flows that focus on a single conversational task

- Easy for copy writers to find the right messages and prompts
- A change request by an UX designer does not impact other functionality

What type of messages to print in a flow?

Modular development carries the risk of too many prompts and messages

- Typically, excessive display of prompts and messages only becomes apparent after the flows have been orchestrated into a conversation

A suggestion based on our concept of intent flows and service flows

- Intent flows print contextual messages
    - "I placed that order for you"
    - "To cancel this order, I need you to sign-in. Are you ready?"

- Service flows print functional messages
    - "I seem to be having trouble locating the order number you requested an update for. Did I understand the order number 'ORCL1234567' correctly?"
    - "I found several products that match your query. Please help me clarify by selecting one of the choices or providing a new product name."

Session Agenda

1 Implement a modular design

2 **How to end flows**

3 Use a naming convention

4 Handle errors

5 Practice design by validation

”

what goes up must come down

Calling a flow from another creates a call hierarchy	Every flow eventually comes to an end	Values may need to be passed to calling flow upon flow return

How to end a flow

End flow states return control to the parent flow

Two types of ending flows: implicit, explicit

Implicit end flows don't need to be added to a flow

- Only return control to parent

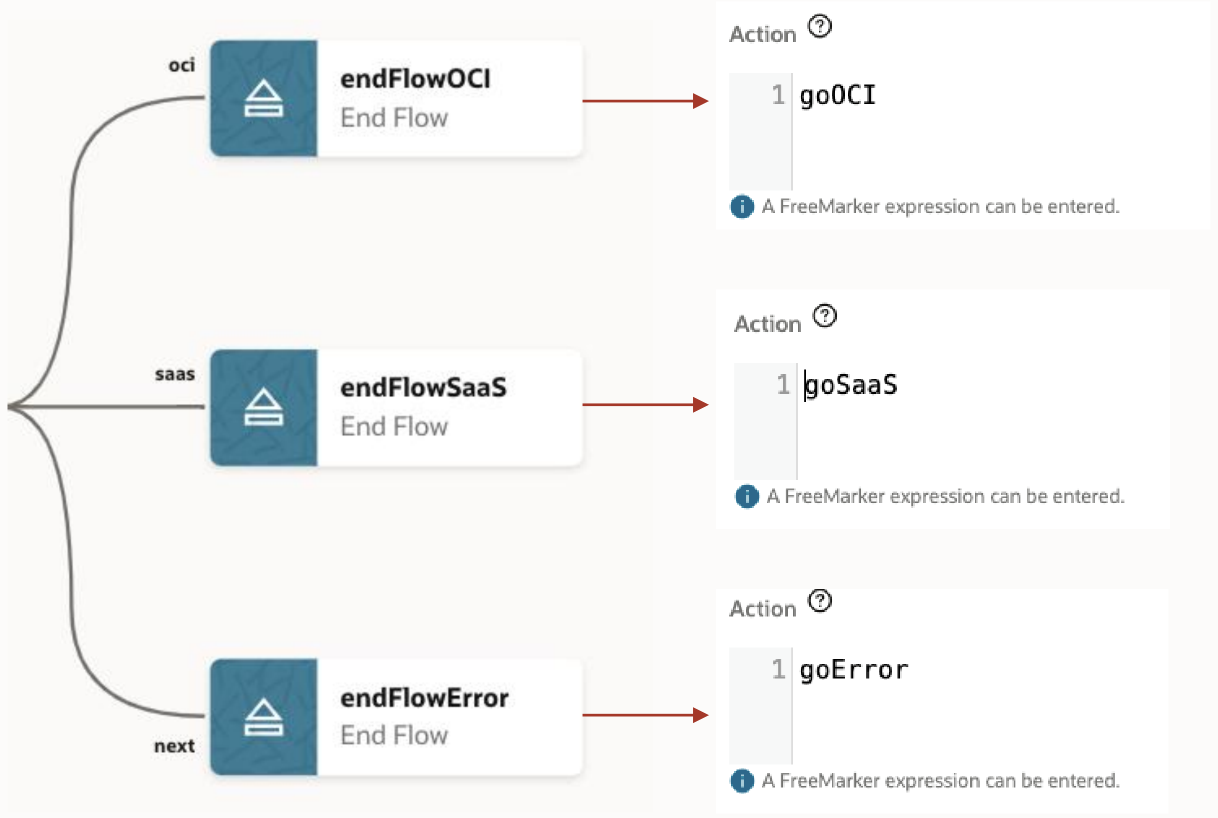
Explicit end flow are added as a component states

- May return values using output parameters
- Return an action string to the parent flow that could be used to decide the next navigation

Explicit flows make it easier to understand what a flow does after it has passed the last component state

Parent navigation based on action string returned from a child flow

Child Flow



```
goOCI
```

A FreeMarker expression can be entered.

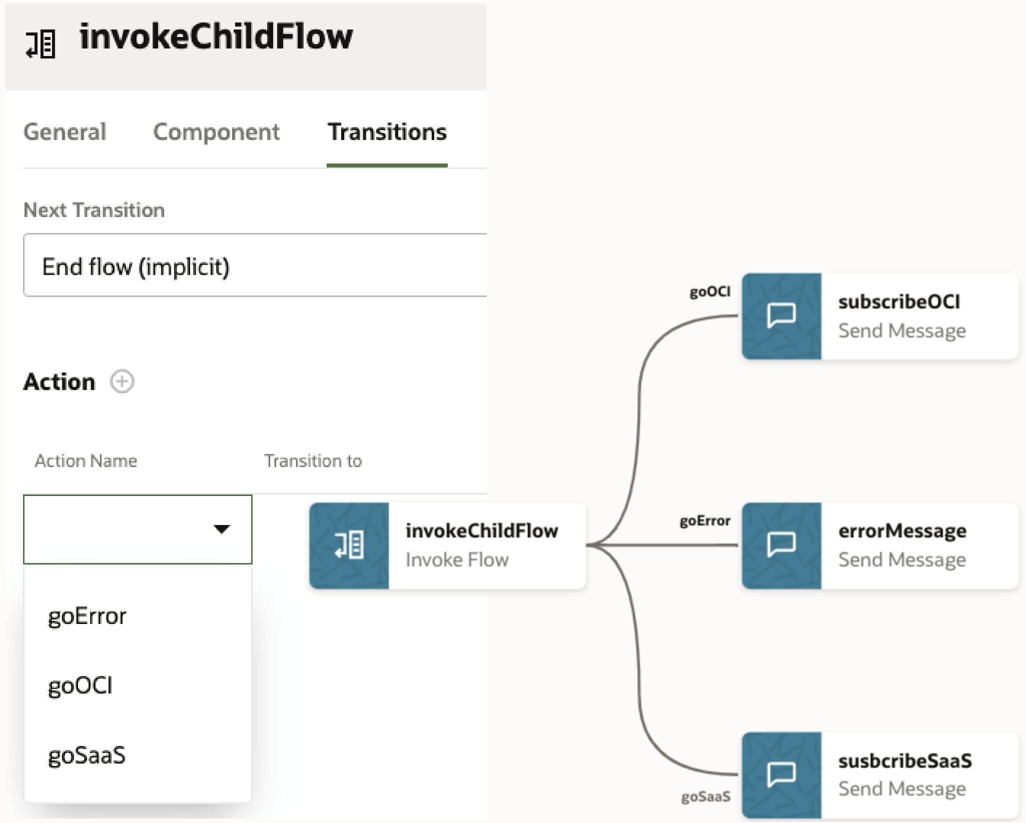
```
goSaaS
```

A FreeMarker expression can be entered.

```
goError
```

A FreeMarker expression can be entered.

Parent Flow



End flow (implicit)

Understanding end flow keepTurn behaviors

Controls transition when the flow ends.

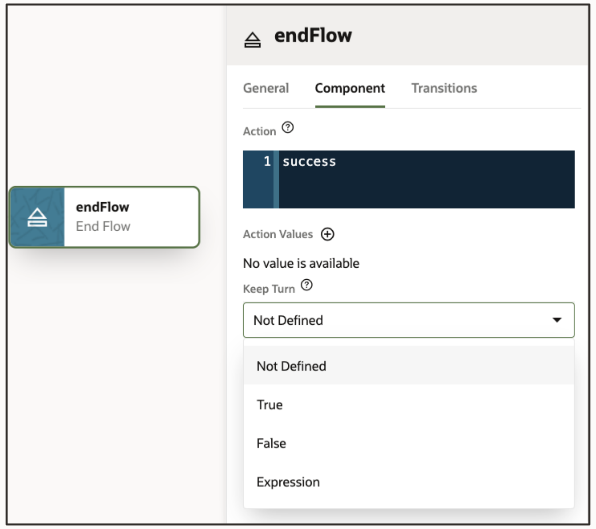
- True: after the flow ends, the next state is executed directly.
- False: after flow ends, the skill waits for a user input before the next state executes.

Value ignored and **ALWAYS** set to "*true*" regardless of the value you define:

- Current running flow is a **child flow**.
- Current running flow is a **root Flow**, and the last state before the "end flow" captured user input.

Default value set to "**Not Defined**"

- Recommended not to change this value


endFlow

End Flow

endFlow

General | Component | Transitions

Action

```
success
```

Action Values

No value is available

Keep Turn

Not Defined

Session Agenda

1 Implement a modular design
2 How to end flows
3 **Use a naming convention**
4 Handle errors
5 Practice design by validation

Naming convention

> Having a good naming convention is priceless

Use a naming convention to indicate context, meaning and a possible relationship between artifacts (e.g., intents and flows)

Helps copy writers and conversation designers to locate strings and conversation states

Allows developers to find the area within the flow design that is misbehaving at runtime

Allows to filter e.g., intent names more efficiently at design time and runtime

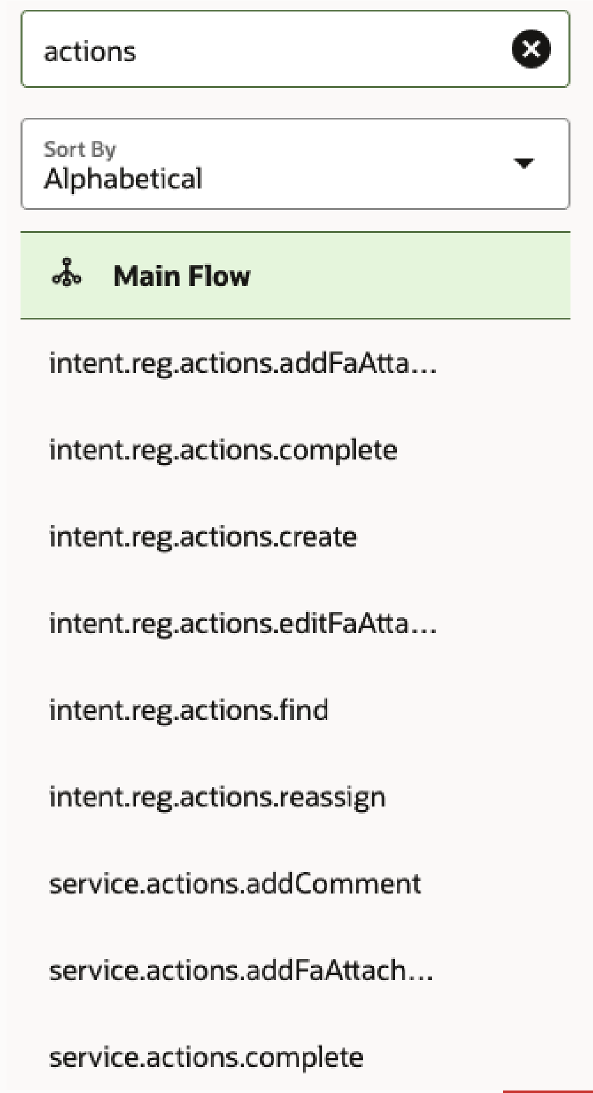
Example naming convention

Intents

- Regular intent
  - reg.orders.create, orders.reg.create, reg.profile.find, reg.profile.find.mine
- Answer intent
  - ans.oci.freetier.targetAudience, oci.ans.freetier.targetAudience

Flows

- For intent flows use e.g., "intent.<name of intent>"
  - intent.reg.orders.create
- For service flow use e.g., "service.<name>"
  - service.orders.create
- For global flows use e.g., "global.<name>"



intent.reg.actions.addFaAtta...
intent.reg.actions.complete
intent.reg.actions.create
intent.reg.actions.editFaAtta...
intent.reg.actions.find
intent.reg.actions.reassign
service.actions.addComment
service.actions.addFaAttach...
service.actions.complete

Example naming convention

Entities

- cbe.<name>, list.<name>, regex.<name>
- optional: metadata.list.<name>, metadata.regex.<name>, …

Entity event handler

- Use name of CBE in name
    - CreateOrder_EEH, OrderSummary_EEH

Resource bundle keys

- Add location of use to key name
    - <flow name>.<state name>.<key name>
    - <entity name>.<prompt | errorMessage |disambiguationMessage>
    - <entity name>.<bag item name>.<prompt | errorMessage |disambiguationMessage>
    - eeh.<entity event handler name>.<key name>

Session Agenda

1 Implement a modular design

2 How to end flows

3 Use a naming convention

4 **Handle errors**

5 Practice design by validation

Error handling

Always implement an error handling strategy

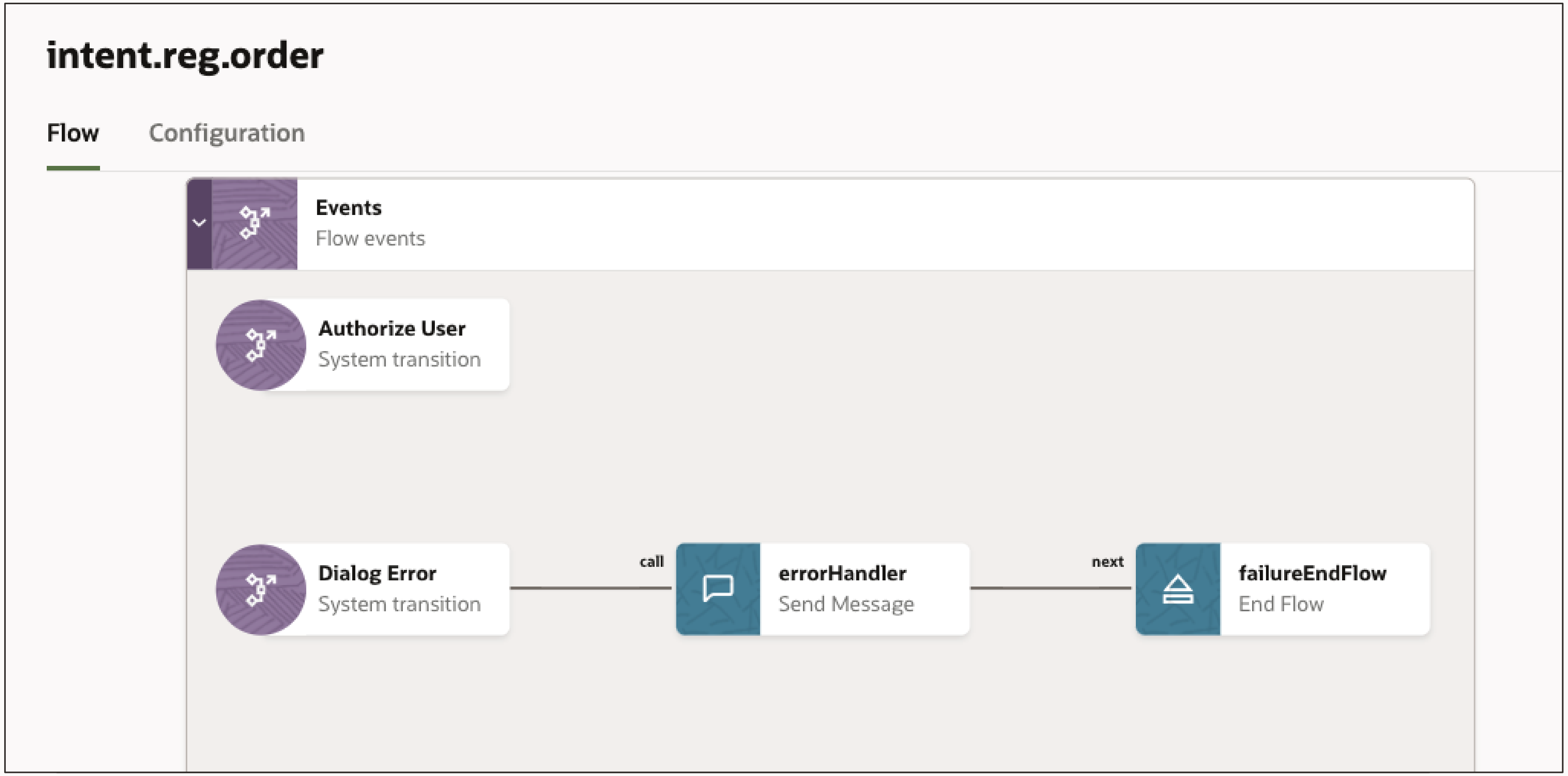
- Print a customized message
- Log error
- Notify admins
- Initialize/Reset variables

Error handler at the flow level

Global error handler shared across all flows

Error handler

Flow specific


intent.reg.order

Flow Configuration

Events
Flow events

Authorize User
System transition

Dialog Error
System transition

call

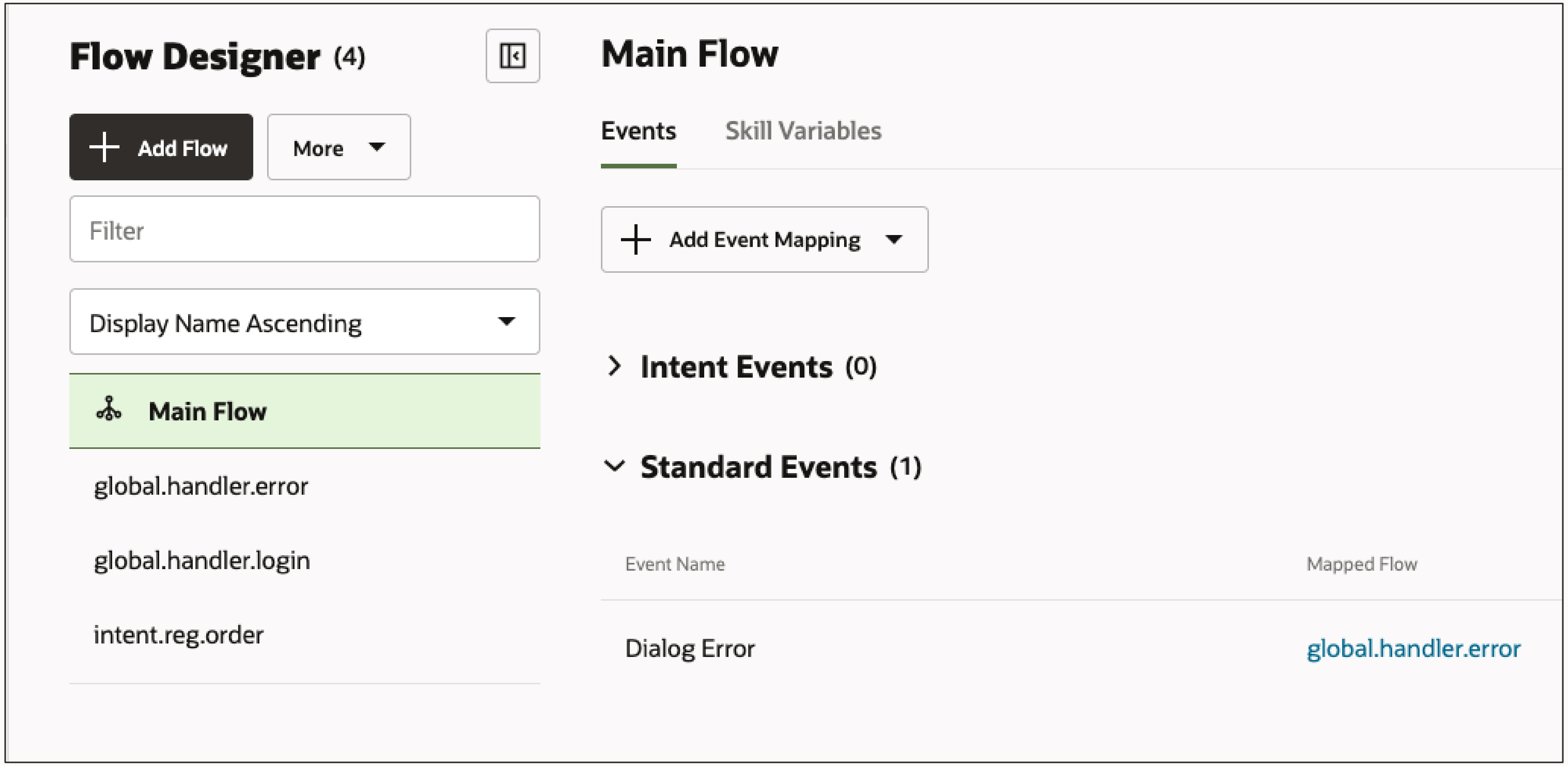
errorHandler
Send Message

next

failureEndFlow
End Flow

Error handler

Global handler



Flow Designer (4)

Add Flow | More

Filter

Display Name Ascending

- Main Flow
- global.handler.error
- global.handler.login
- intent.reg.order

Main Flow

Events | Skill Variables

Add Event Mapping

Intent Events (0)

Standard Events (1)

Event Name	Mapped Flow
Dialog Error	global.handler.error

Session Agenda

1. Implement a modular design
2. How to end flows
3. Use a naming convention
4. Handle errors
5. **Practice design by validation**

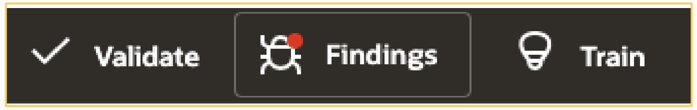
Design by validation

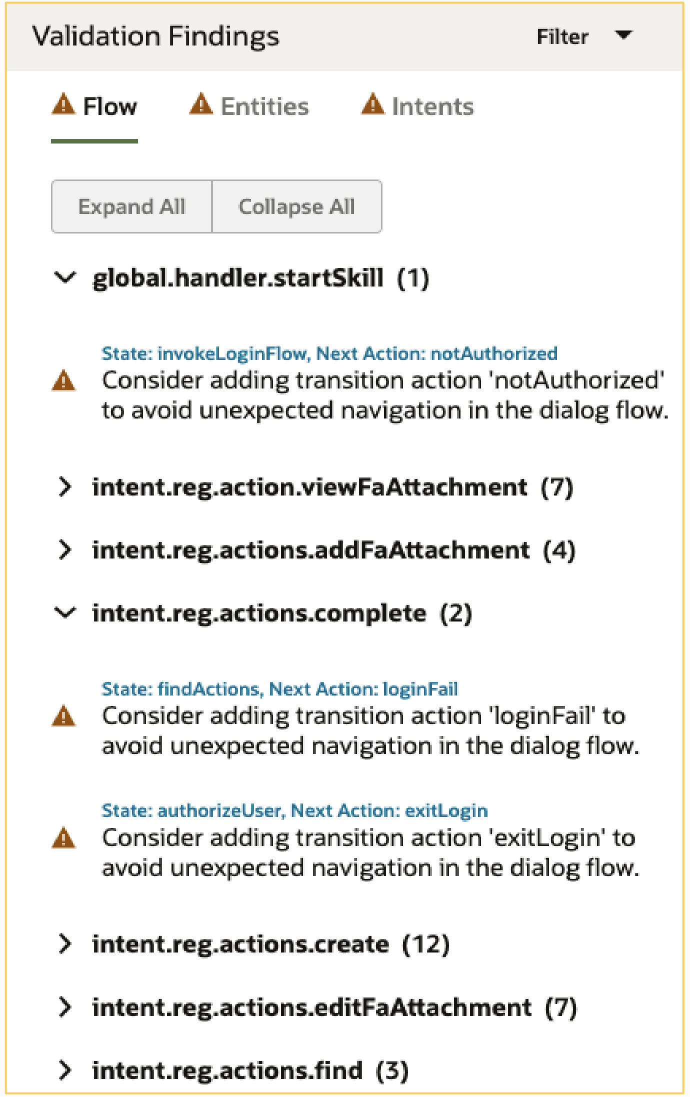
To err is human - Alexander Pope

See what you missed implementing in your intents, entities and flows by using the validate button

- Best practices built-in

Validator shows errors as well as warnings and tips





global.handler.startSkill (1)

State: invokeLoginFlow, Next Action: notAuthorized
Consider adding transition action 'notAuthorized' to avoid unexpected navigation in the dialog flow.

intent.reg.action.viewFaAttachment (7)

intent.reg.actions.addFaAttachment (4)

intent.reg.actions.complete (2)

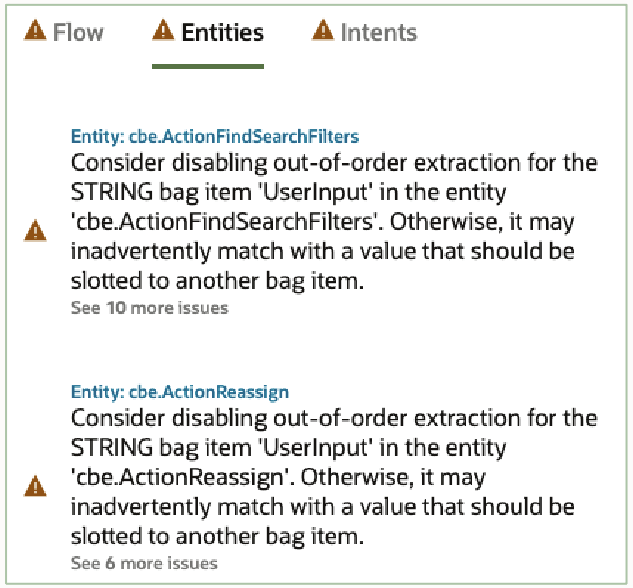
State: findActions, Next Action: loginFail
Consider adding transition action 'loginFail' to avoid unexpected navigation in the dialog flow.

State: authorizeUser, Next Action: exitLogin
Consider adding transition action 'exitLogin' to avoid unexpected navigation in the dialog flow.

intent.reg.actions.create (12)

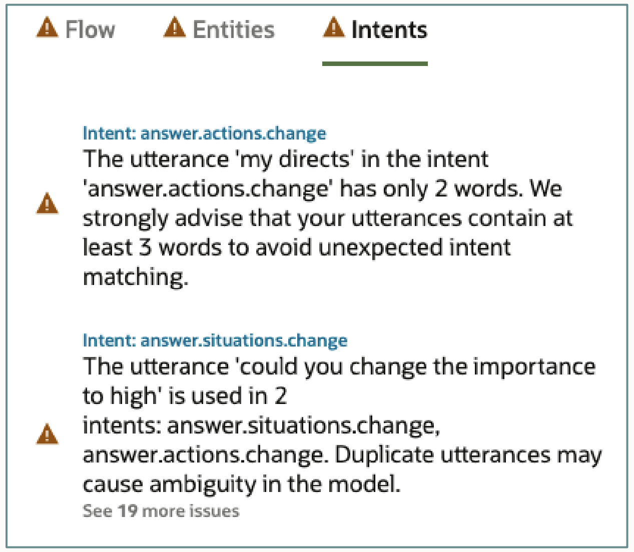
intent.reg.actions.editFaAttachment (7)

intent.reg.actions.find (3)



Entity: cbe.ActionFindSearchFilters
Consider disabling out-of-order extraction for the STRING bag item 'UserInput' in the entity 'cbe.ActionFindSearchFilters'. Otherwise, it may inadvertently match with a value that should be slotted to another bag item.
See 10 more issues

Entity: cbe.ActionReassign
Consider disabling out-of-order extraction for the STRING bag item 'UserInput' in the entity 'cbe.ActionReassign'. Otherwise, it may inadvertently match with a value that should be slotted to another bag item.
See 6 more issues



Intent: answer.actions.change
The utterance 'my directs' in the intent 'answer.actions.change' has only 2 words. We strongly advise that your utterances contain at least 3 words to avoid unexpected intent matching.

Intent: answer.situations.change
The utterance 'could you change the importance to high' is used in 2 intents: answer.situations.change, answer.actions.change. Duplicate utterances may cause ambiguity in the model.
See 19 more issues

ORACLE