

ORACLE®

# Oracle Digital Assistant

## The Complete Training

**Integrating human agents into an existing digital assistant conversation**

# 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 ➤ Overview of agent integration
- 2 ➤ Use case in action
- 3 ➤ How it works
- 4 ➤ Context transfer using custom properties
- 5 ➤ Using custom properties for queue routing

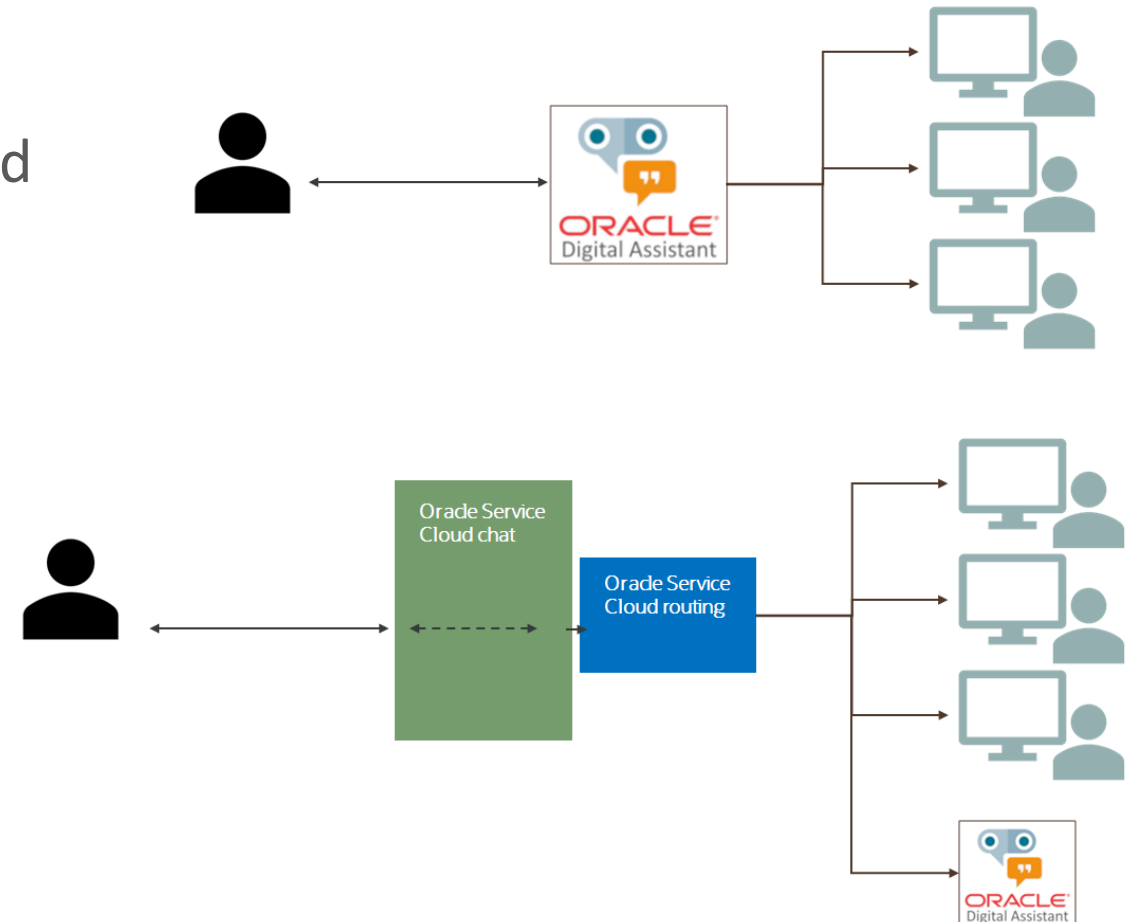
# Topic agenda

- 1 Overview of agent integration
- 2 Use case in action
- 3 How it works
- 4 Context transfer using custom properties
- 5 Using custom properties for queue routing

# Skills

## Human agent integration

- Integrate ODA with call center
  - Digital assistant can help with call center load
  - Agents focus
  - Introduce agent as and when it makes sense
    - Escalation, high value call, complex question
- Architecture options
  - Digital assistant fronts call center
  - Digital assistant as an agent



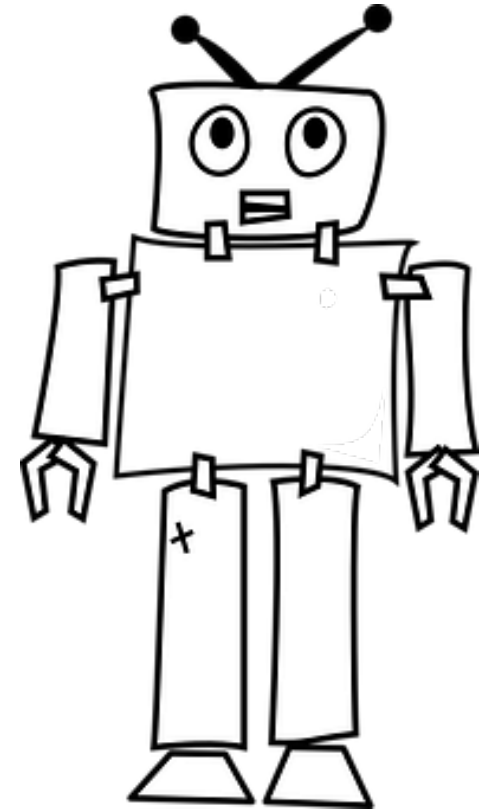
## More about option 1 (escalation from DA chat)

- Agents only get calls when escalated so there will always be a conversation hand off.
- It's best used when clients want to have a standalone digital assistant that can also escalate to humans, if needed.
- This approach uses the publicly-available Service Cloud API.
- Not all features of the Service Cloud chat client are available in the API (e.g., typing indicators).

## More about option 2 (ODA as an agent)

- Best used when a client has a “pre-chat form” and an established live chat implementation.
- No disruption to call center or agent workflow. Agents get calls using the same interface experience as before.
- All features of the Service Cloud chat client are available for the users and agents.
- This approach is only available to ODA (no other chat providers will have this level of integration).
- Clients must have Service Cloud 19c or later.

This session focuses on integrating human agents into an existing digital assistant conversation

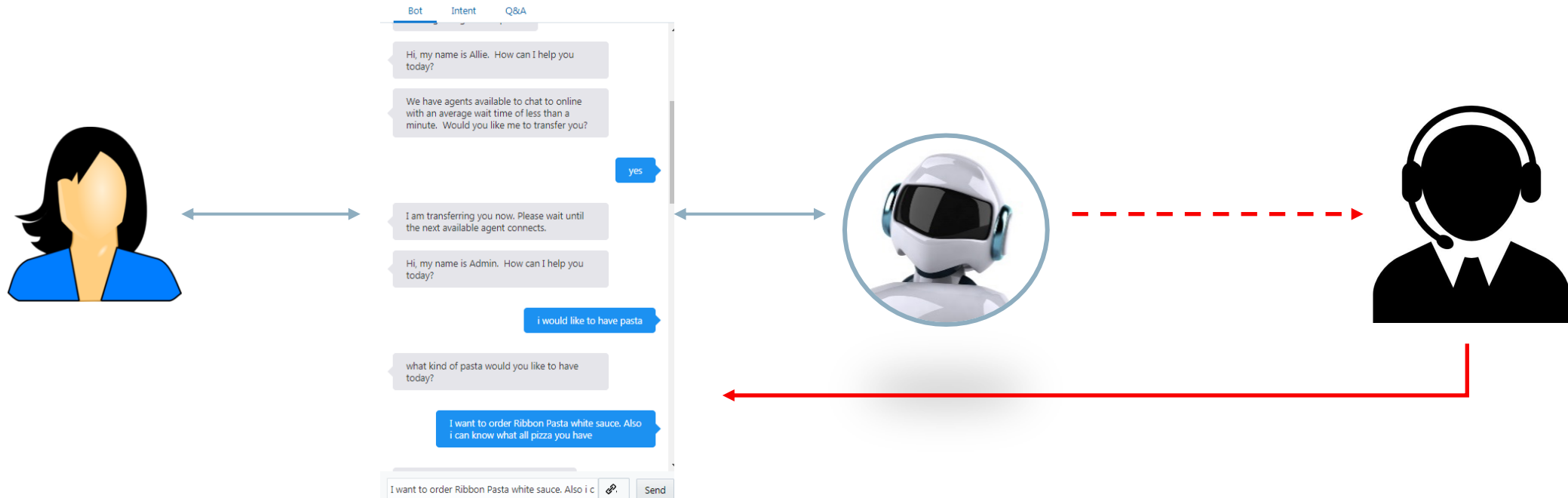


# Overview of agent integration

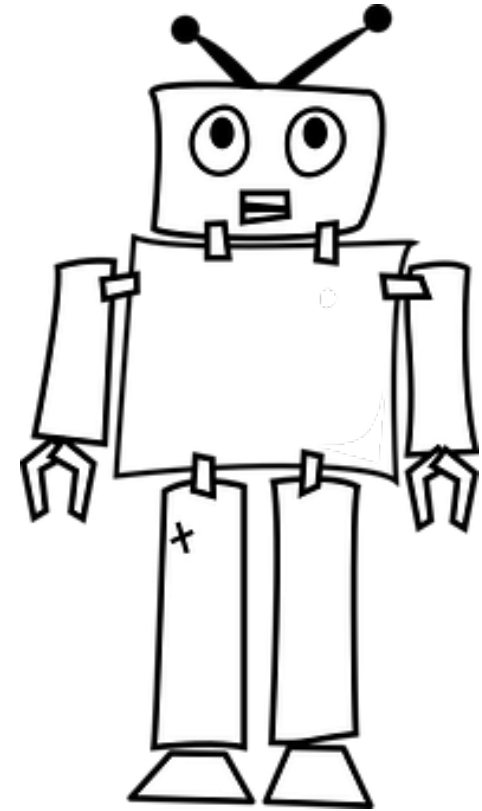
- Digital Assistant often used as call deflection mechanism for call center
  - But sometime you may still need to speak to a human
    - High value engagements
    - Some business interaction might require human involvement
    - Human handles use cases the bot is not set up to handle
- The bot doesn't always get it right
  - Enable users to get unstuck
  - User gets frustrated because bot isn't helping
  - Give access to human support if bot is struggling to deal with human conversation

# Overview of agent integration

- When this happens
  - The digital assistant could transfer the current chat conversation to a human agent.
  - User should continue to stay in the same channel (web, messenger, app)



Oracle Digital Assistant provides  
built-in agent integration with  
**Oracle Service Cloud**



# Actors

- Bot user
  - Customer using Oracle Digital Assistant
  - User can be on any channel supported by Oracle Digital Assistant



- Skill
  - Built using Oracle Digital Assistant



- Agent system
  - Receive and respond to the chat from their existing customer service application platforms like Oracle Service Cloud (RightNow), Zendesk, Genesys etc.



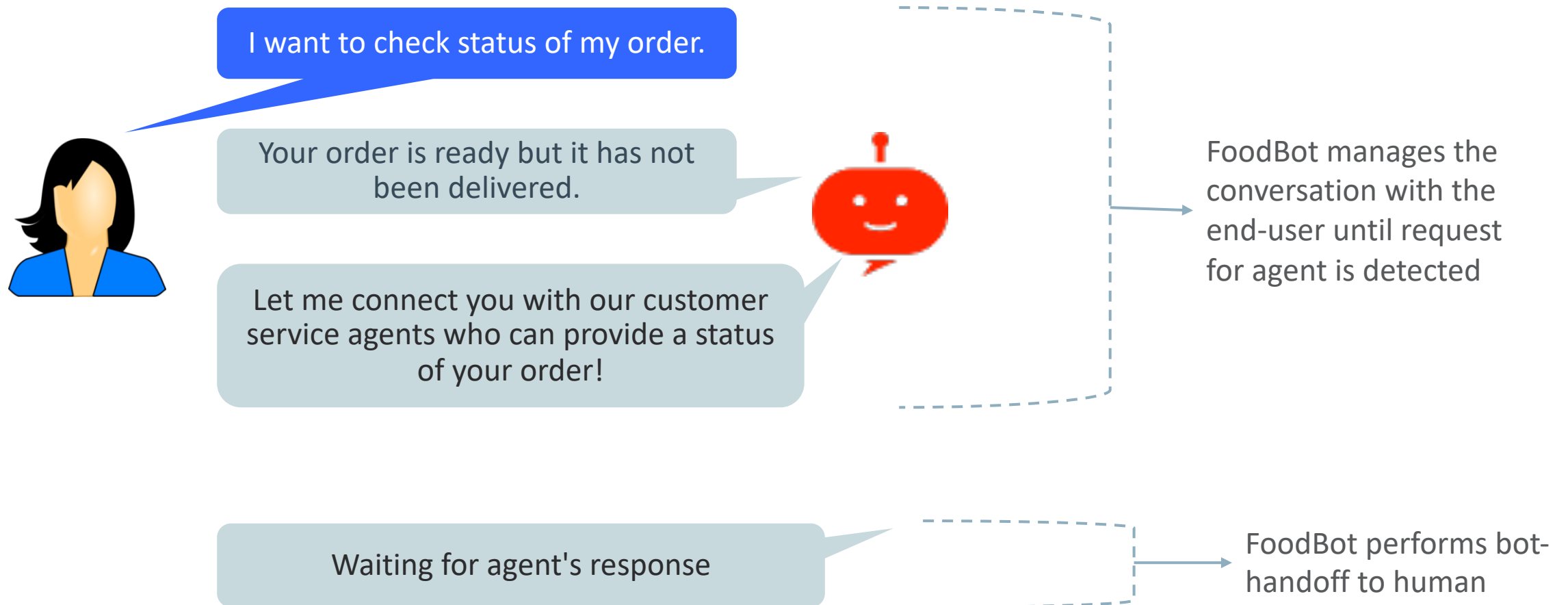
zendesk

GENESYS™

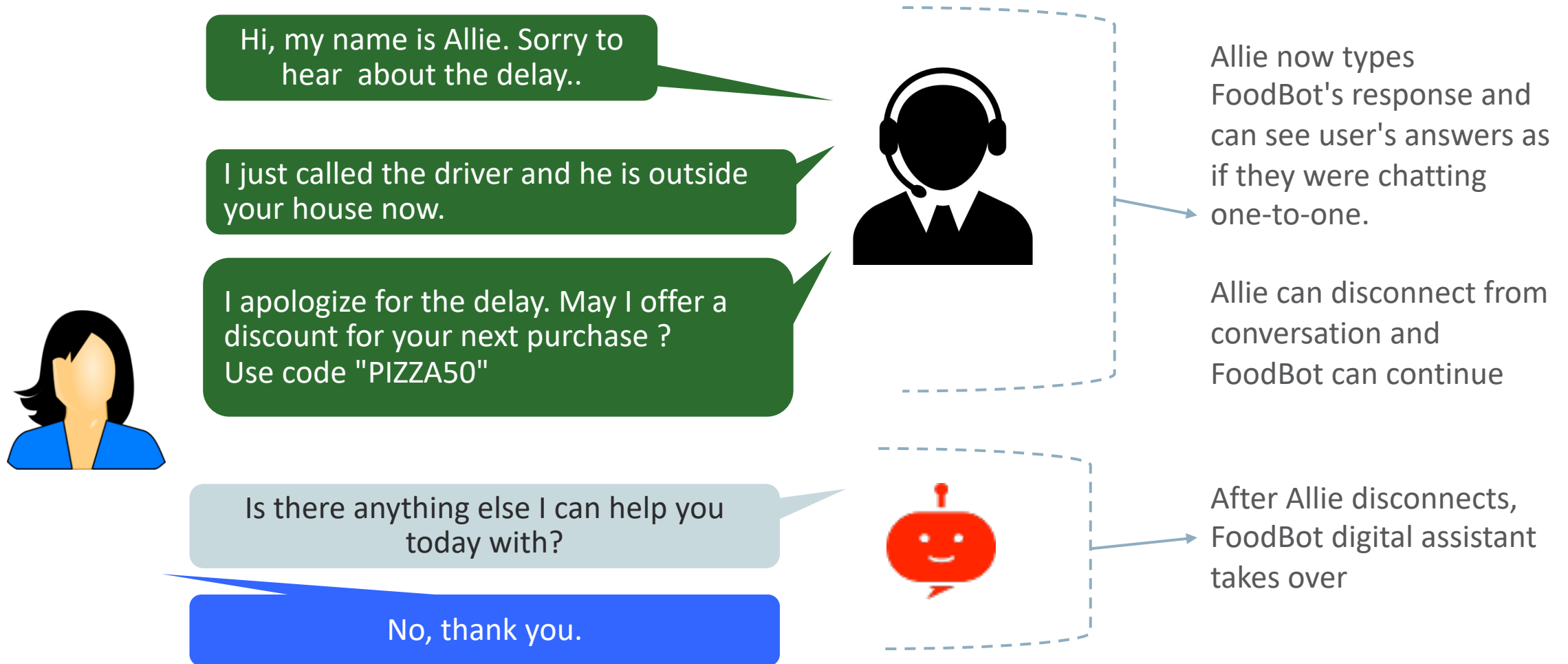
# Topic agenda

- 1 Overview of agent integration
- 2 Use case in action**
- 3 How it works
- 4 Context transfer using custom properties
- 5 Using custom properties for queue routing

# Use case – FoodBot agent integration



# Use case – FoodBot agent integration



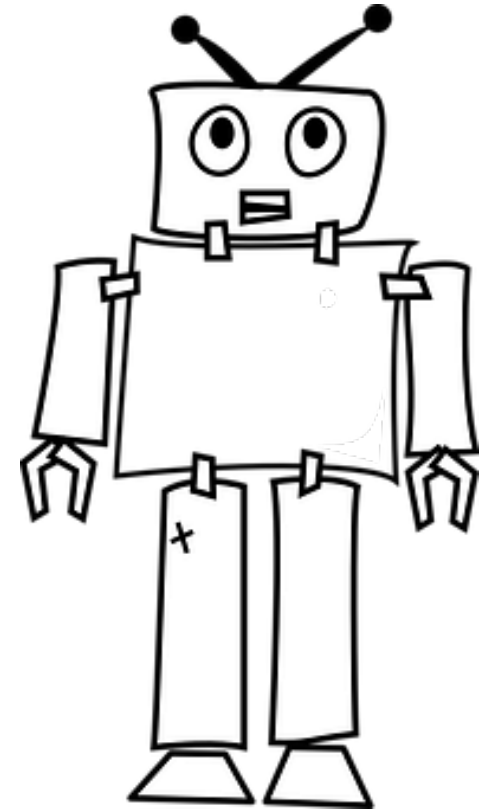
# Agent integration benefits

- Users communicate with a human agent using the same conversation channel they used when chatting with the bot
  - No context switch
  - Same messenger
- Agents
  - Receive user request and accept or decline the conversation
    - In the latter case, the bot takes over again
  - Have access to the complete conversation history of user interacting with the bot
  - Can execute bot flows to help users
- Once agent disconnect from conversation, the bot can continue

# Topic agenda

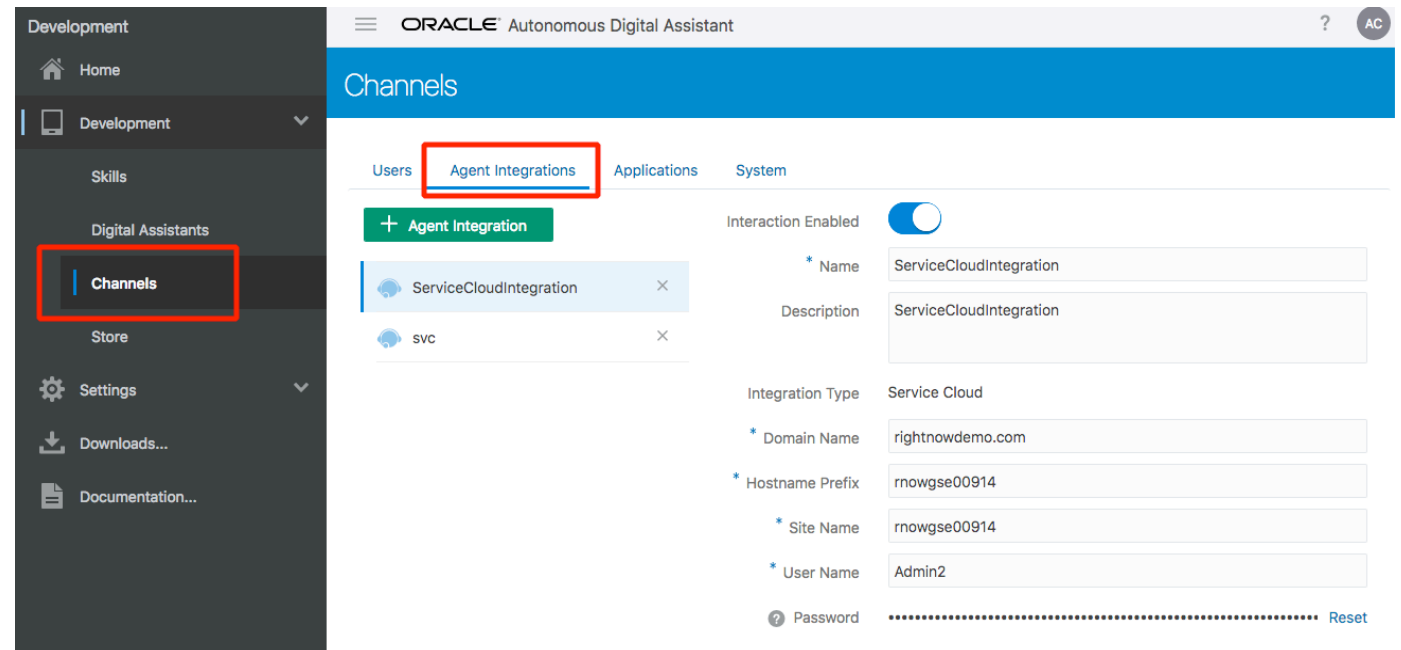
- 1 Overview of agent integration
- 2 Use case in action
- 3 How it works**
- 4 Context transfer using custom properties
- 5 Using custom properties for queue routing

Oracle Digital Assistant uses **specific**  
agent **webhook channels** to integrate  
with **agent systems**.



# Configure agent integration

- Bot developer will add a special agent integration channel to their bot
- Channel initiates conversations and sends/receives agent messages
- Implements a webhook under the covers



# Configure agent integration

- Refer to Service Cloud instance to get details for hostname and site name prefix
  - Get this info from your Service Cloud admin
  - Hostname or sitename typically same as interface name
- [https://<ServiceCloud>.rightnowdemo.com/services/soap/connect/chat\\_soap?wsdl](https://<ServiceCloud>.rightnowdemo.com/services/soap/connect/chat_soap?wsdl)

The screenshot shows the Oracle Autonomous Digital Assistant configuration page for Agent Integrations. The page has a header with the Oracle logo and 'Autonomous Digital Assistant'. Below the header is a blue bar with the word 'Channels'. The main content area has tabs for 'Users', 'Agent Integrations', 'Applications', and 'System'. Under 'Agent Integrations', there is a green '+ Agent Integration' button and a list of existing integrations: 'ServiceCloudIntegration' and 'svc'. To the right, the configuration form for a new integration is shown. It includes a toggle for 'Interaction Enabled' (which is turned on), a 'Name' field with the value 'ServiceCloudIntegration', and a 'Description' field with the value 'ServiceCloudIntegration'. Below these is the 'Integration Type' dropdown, which is set to 'Service Cloud'. A red box highlights the fields for 'Domain Name' (rightnowdemo.com), 'Hostname Prefix' (rnowgse00914), 'Site Name' (rnowgse00914), and 'User Name' (Admin2). At the bottom, there is a 'Password' field with a masked password and a 'Reset' link.

ORACLE Autonomous Digital Assistant

Channels

Users Agent Integrations Applications System

+ Agent Integration

ServiceCloudIntegration ×

svc ×

Interaction Enabled ☒

\* Name ServiceCloudIntegration

Description ServiceCloudIntegration

Integration Type Service Cloud

\* Domain Name rightnowdemo.com

\* Hostname Prefix rnowgse00914

\* Site Name rnowgse00914

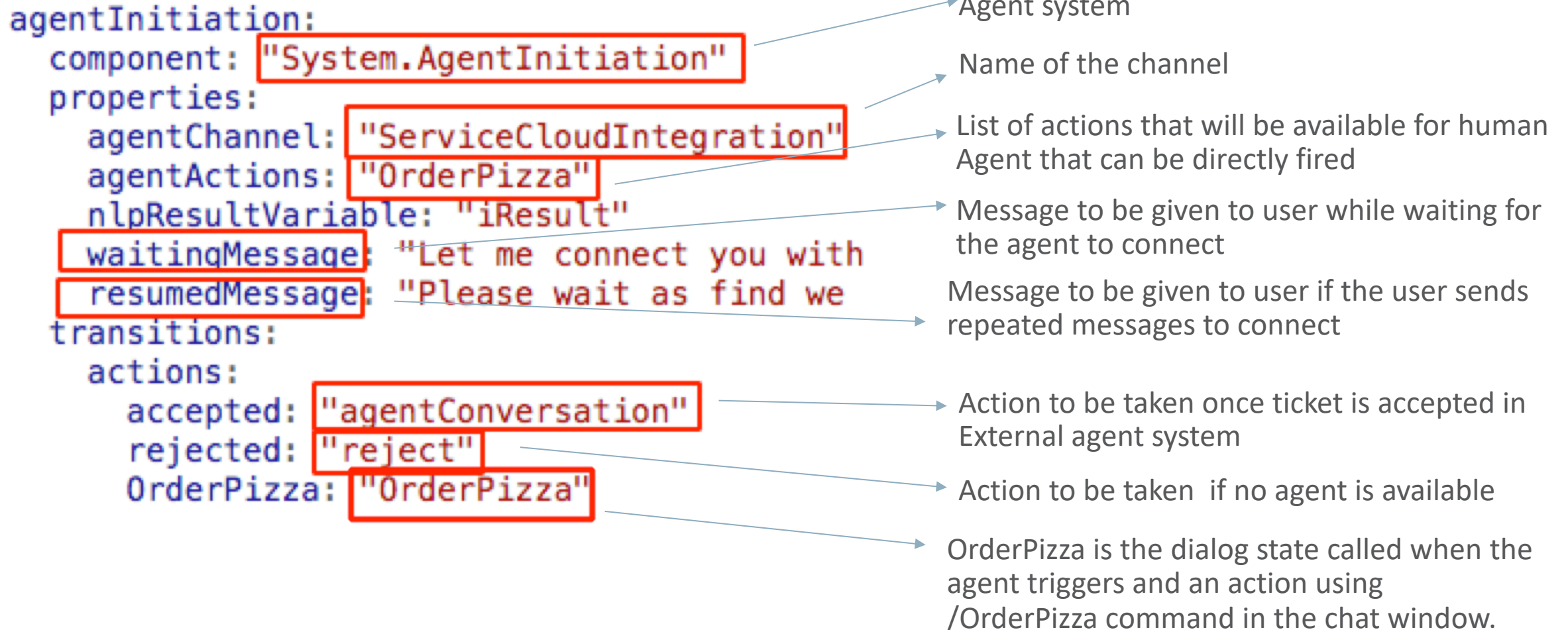
\* User Name Admin2

? Password ..... Reset

# Enable bot to human interaction from BotML

- Through BotML you decide when you allow the user to be passed to agent
  - When a user conversation is passed to an agent
  - Use case is handled only by an agent
  - Sentiment analysis - user is angry/upset?
- In the dialog flow two System components enable bot to human interaction
  - System.AgentInitiation
  - System.AgentConversation

# Agent initiation in dialog flow



# Start agent conversation in dialog flow

```
agentConversation:
```

```
  component: "System.AgentConversation"
```

Manages message processing between the user and the agent

```
  properties:
```

```
    agentChannel: "ServiceCloudIntegration"
```

Name of the channel

```
    nlpResultVariable: "iResult"
```

```
    exitKeywords: "bye, bye, good night, end, quit"
```

Keywords when used by the user terminates the conversation with the agent.

```
    conclusionMessage: "Have a nice day."
```

```
  transitions:
```

```
    next: "endPrompt"
```

When user types any of these words the conversation is ended.

# A typical user – bot – agent conversation

CrcPizzaSkill\_AB Tester

I am looking for some offers I can't find online

We have a special offer for you today, get 15% off on your order today.


Great Show me the pizzas now

Connecting you back to the Bot..

Here are our pizzas you can order today

CHEESE

Classic marinara sauce topped with whole milk mozzarella cheese.



Order Now

ORACLE

Chats

You requested a chat.

Abhay Bhavsar

00:02:26

Queue: Default Chat Queue

Email: abhay.bhavsar@oracle.com

Chat subject: Chat Session

14:29:17 [00:01:07] **Abhay Bhavsar:** I am looking for some offers I can't find online

14:29:43 [00:01:33] **Admin:** We have a special offer for you today, get 15% off on your order today.

14:30:32 [00:02:22] **Abhay Bhavsar:** Great Show me the pizzas now

14:30:54 [00:02:44] **Admin:** /OrderPizza

14:30:59 [00:02:49] **System:** The end user Abhay Bhavsar has disconnected from chat 485

Public Response

00000

Contact

Abhay Bhavsar

Incident

190117-000000

Date Requested

01/17/2019 12:58 AM

Interface

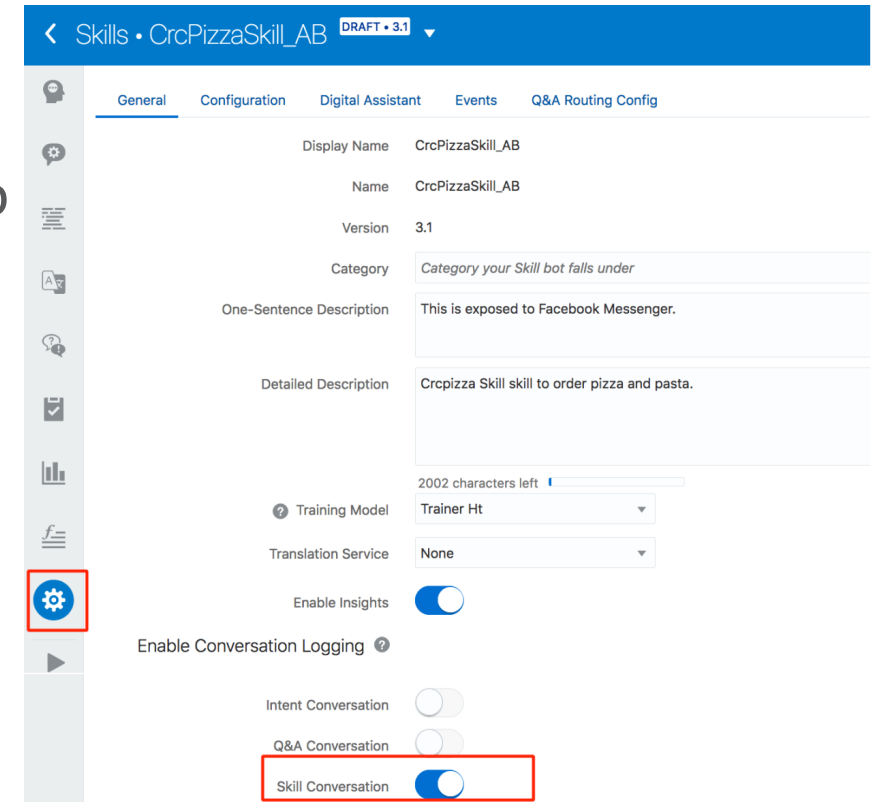
rnowgse00914

# Topic agenda

- 1 Overview of agent integration
- 2 Use case in action
- 3 How it works
- 4 Context transfer using custom properties**
- 5 Using custom properties for queue routing

# Context transfer using custom properties

- Ideally the bot should pass to the agent:
  - User information
  - Context information – reason for user being directed to the live agent
  - Current bot – user conversation history
  - Action that the agent can trigger to return the handoff to the bot
  - Whether the conversation can be passed to a specific agent/department



The screenshot shows the configuration page for a skill named 'CrcPizzaSkill\_AB' in the Oracle CX Cloud interface. The page has a blue header with a back arrow, the skill name, and a 'DRAFT • 3.1' status. A left sidebar contains icons for various functions, with the settings gear icon highlighted by a red box. The main content area has tabs for 'General', 'Configuration', 'Digital Assistant', 'Events', and 'Q&A Routing Config'. The 'General' tab is active, displaying fields for 'Display Name', 'Name', 'Version', 'Category', 'One-Sentence Description', and 'Detailed Description'. Below these are settings for 'Training Model', 'Translation Service', 'Enable Insights', and 'Enable Conversation Logging'. At the bottom, there are three toggle switches for 'Intent Conversation', 'Q&A Conversation', and 'Skill Conversation', with the 'Skill Conversation' toggle highlighted by a red box and currently turned on.

Skills • CrcPizzaSkill\_AB DRAFT • 3.1

General Configuration Digital Assistant Events Q&A Routing Config

Display Name CrcPizzaSkill\_AB

Name CrcPizzaSkill\_AB

Version 3.1

Category *Category your Skill bot falls under*

One-Sentence Description This is exposed to Facebook Messenger.

Detailed Description Crcpizza Skill skill to order pizza and pasta.

2002 characters left

Training Model Trainer Ht

Translation Service None

Enable Insights ☒

Enable Conversation Logging ⓘ

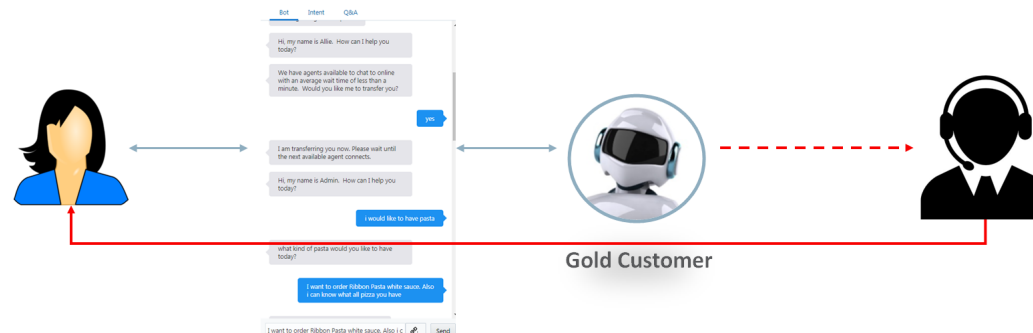
Intent Conversation ☐

Q&A Conversation ☐

Skill Conversation ☒

# Three ways of passing context

- Conversation logs
  - “Out of the box” functionality that passes conversation history
  - Need to turn on conversation logging
- Primary attributes
  - Fields already defined in service cloud that can be set from the bot
- Custom attributes
  - Newly created properties that you define in service cloud for a specific need



# Context passed in conversation logs

## Testing TutorialCbPizzaBot Skill

Place an order for 2 pitchers

The Water Jug shipment with 2 items will be sent right away

Goodbye.

talk to an agent

talk to an agent

Conversation

States

### Chats

Unrestricted

Joe Doe 00:01:01 1

Queue: Product Support  
Email: joe.doe@oracle.com  
Chat subject: talk to an agent

bot: Hi, how can I help you?  
user: Place an order for 2 pitchers  
bot: The Water Jug shipment with 2 items will be sent right away  
bot: Goodbye.  
user: talk to an agent  
bot: Hi, how can I help you?  
user: talk to an agent

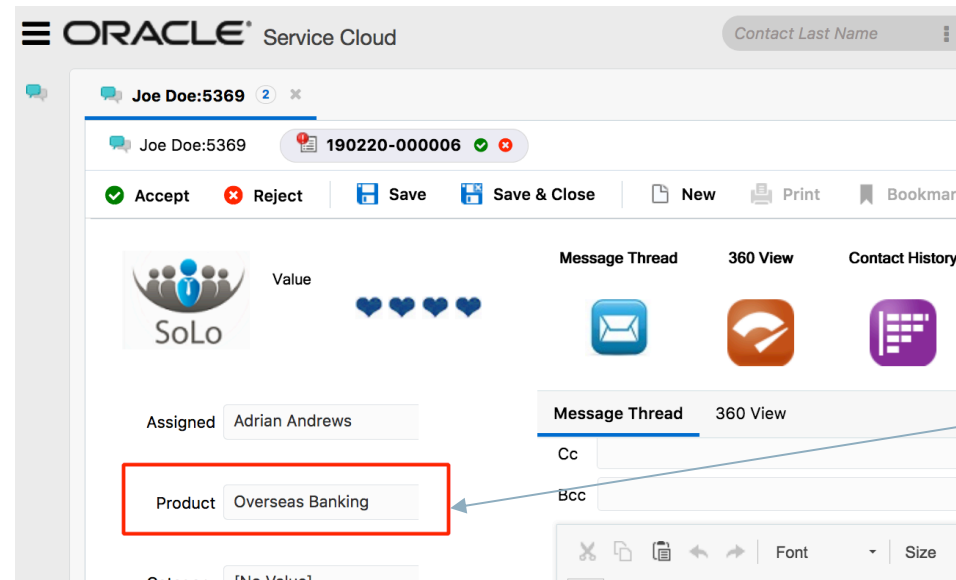
Here are the available agent actions (Please send one of the available actions to bot using  
backslash e.g. /actionName):  
/Order : Order

Public Response

# Passing context using primary fields

- Service Cloud already defines useful information in primary fields
  - e.g. ProductID, ContactID, OrganizationID
- These fields already exist in the Service Cloud dashboard
- These can be populated directly from the dialog flow

In this case `productID` 178 refers to  
**Overseas Banking**



```
metadata:  
  platformVersion: "1.1"  
  main: true  
  name: "CrcPizzaBot_AB"  
  context:  
    variables:  
      iResult: "nlresult"  
      myCustomProps: "map"
```

```
setupCustomProps:  
  component: "System.SetVariable"  
  properties:  
    variable: "myCustomProps"  
    value:  
      customerInformation:  
        interfaceID:  
          name: "solo_financial_1"  
          productID:  
            id: 178  
      transitions: {}
```

# Passing context using primary fields

- Some fields get populated automatically through conversation history
  - Firstname, Lastname, EmailAddress
- For others you explicitly define the fields to be updated and their values in the flow
- Refer to SOAP API to understand properties and types
  - [https://<Service Cloud>/services/soap/connect/chat\\_soap?wsdl=server](https://<Service Cloud>/services/soap/connect/chat_soap?wsdl=server)
- Look for section ChatCustomerInformation

```
<!-- ===== -->  
<!-- Chat Customer Information -->  
<!-- ===== -->
```

```
<xs:complexType name="ChatCustomerInformation">
```

# Passing context using primary fields

- Note that the InterfaceID is of type NamedID so it can be referred to by using an id as well as name, while the ProductID can only be referred to by id since it is of type ID.

```
<xs:element name="InterfaceID" type="rnccm:NamedID" minOccurs="1" maxOccurs="1"/>
<xs:element name="ContactID" type="rnccm:ID" minOccurs="0" maxOccurs="1" />
<xs:element name="OrganizationID" type="rnccm:ID" minOccurs="0" maxOccurs="1" />
<xs:element name="Question" type="xs:string" minOccurs="0" maxOccurs="1"/>
<xs:element name="ProductID" type="rnccm:ID" minOccurs="0" maxOccurs="1"/>
<xs:element name="CategoryID" type="rnccm:ID" minOccurs="0" maxOccurs="1"/>
```

```
<xs:complexType name="NamedID">
  <xs:sequence>
    <xs:element name="ID" type="ID" minOccurs="0" maxOccurs="1"/>
    <xs:element name="Name" type="xs:string" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="ID">
  <xs:attribute name="id" type="xs:long" use="optional"/>
</xs:complexType>
```

# Passing context using primary fields

- In dialog flow we have a map variable with a particular format
  - customerInformation – The format of the customer information is as per the SOAP WSDL
    - Note that interfaceID can be referred by name as well as id unlike productID

```
setupCustomProps:  
  component: "System.SetVariable"  
  properties:  
    variable: "myCustomProps"  
    value:  
      customerInformation:  
        interfaceID:  
          name: "solo_financial_1"  
        productID:  
          id: 178  
      transitions: {}
```

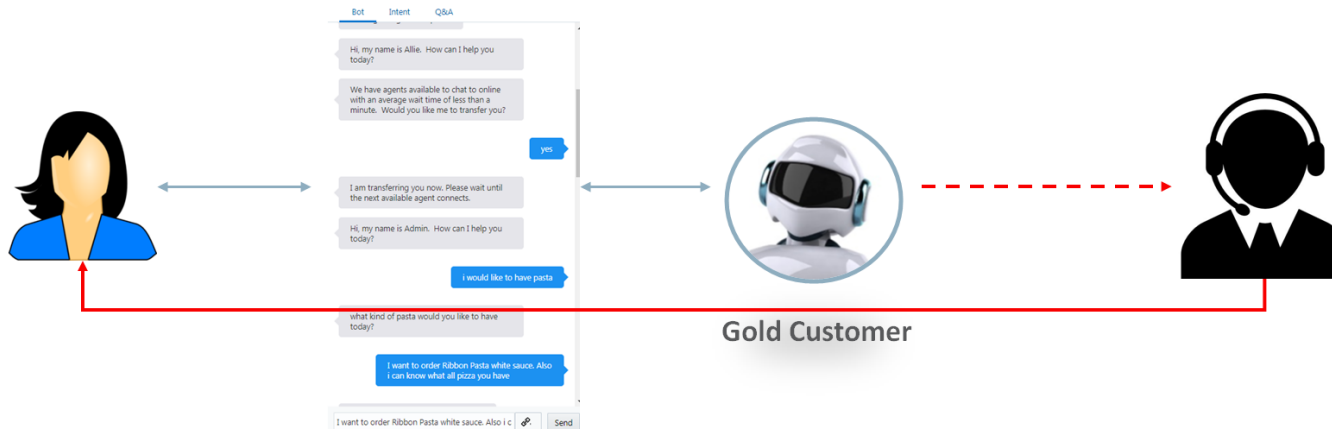
• myCustomProps is the properties object. It holds all the primary as well as custom properties

• interfaceID is the name of the interface defined in Service Cloud. The Oracle Service Cloud SOAP API defines the structure of this object.

• Product ID points to the Service Product ID in Service Cloud.

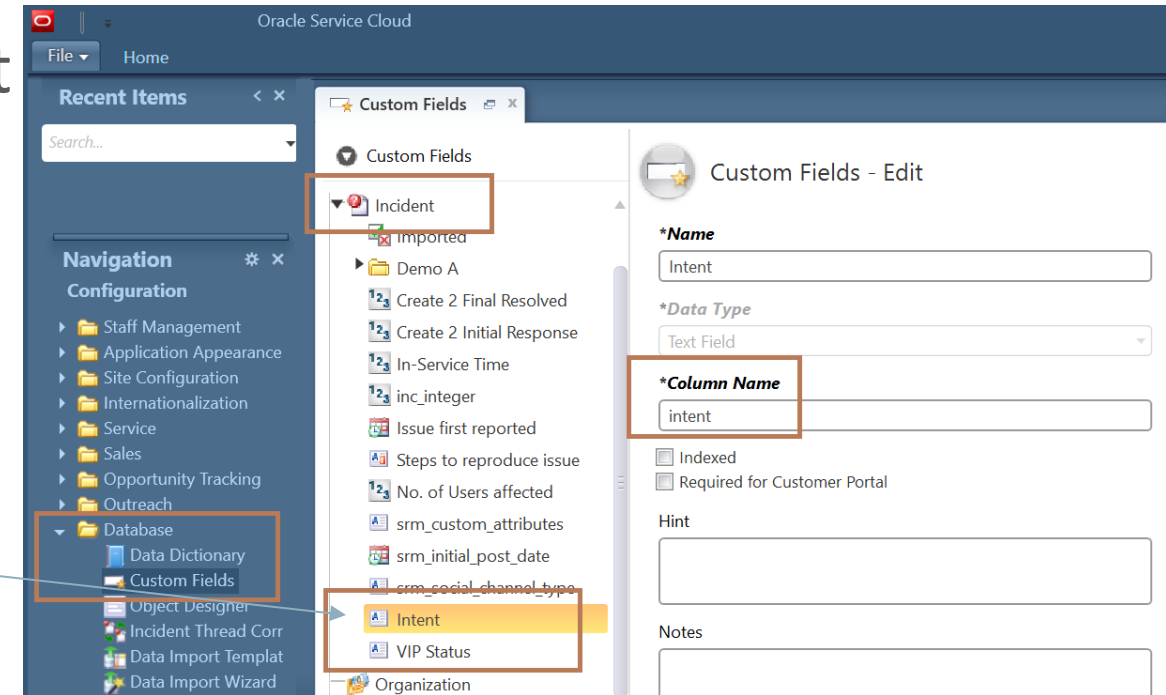
# Passing context using custom fields

- Custom fields allow use case specific information to be passed between the skill and Service Cloud
  - e.g. pass an order number from the bot to the agent
- Service cloud displays this information to the agent
- Routing rules can be used to route calls to agents based on custom fields
  - GOLD customer gets routed to a department for handling high profile customers



# Passing context using custom fields

- When a chat request is accepted an incident is created, the incident dataset gets populated from the custom properties. Both primary and custom fields gets populated.
- New custom fields can be created in the Incident table
- In dialog flow custom fields are referred to C\$<CustomFieldName>



# Passing context using custom fields

- The documentation provides access to the complete Service Cloud data
- Oracle Service Cloud provides REST APIs for listing custom fields
  - Developer can discover all custom properties
  - <https://docs.oracle.com/en/cloud/saas/service/18c/cxsvc/api-queryresults.html>

JSON

Raw Data

Headers

Save

Copy

▼ items:

▼ 0:

tableName: "incidents"

count: 16

▼ columnNames:

0: "imported"

1: "cs\_accepted\_offer\_yn"

2: "cs\_accepted\_install\_yn"

3: "cs\_install\_date\_dtm"

4: "cs\_made\_offer\_yn"

5: "cs\_offer\_name\_text"

6: "cs\_offer\_price\_text"

7: "cs\_offer\_install\_price\_text"

8: "cs\_offer\_tax\_text"

9: "cs\_offer\_total\_text"

10: "cs\_payment\_method\_menu"

11: "status\_crosschannelreport"

12: "c2fr"

13: "c2init\_resp"

14: "in\_service"

15: "inc\_integer"

16: "issue\_first\_reported"

17: "steps\_to\_repro"

18: "no\_of\_users\_affected"

19: "srm\_custom\_attributes"

20: "srm\_initial\_post\_date"

21: "srm\_social\_channel\_type"

22: "intent"

23: "vipstatus"

# Passing context using custom fields

- Custom properties defined as a map in dialog flow
- Pass variable with a particular format to support each custom field
  - customFields, custom fields
- custom properties are referred to by the name `c$<CustomFieldName>`
- The data type of these properties is defined by the SOAP WSDL
  - [https://<Service Cloud>/services/soap/connect/chat\\_soap?wsdl=server](https://<Service Cloud>/services/soap/connect/chat_soap?wsdl=server)

```
metadata:
  platformVersion: "1.1"
main: true
name: "CrcPizzaBot_AB"
context:
  variables:
    iResult: "nlresult"
    myCustomProps: "map"

setupCustomProps:
  component: "System.SetVariable"
  properties:
    variable: "myCustomProps"
    value:

customerInformation:
  interfaceID:
    name: "solo_financial_1"
  productID:
    id: 178

customFields:
  - name: "c$intent"
    dataType: "STRING"
    dataValue:
      stringValue: "${intentName}"
  - name: "c$vipstatus"
    dataType: "STRING"
    dataValue:
      stringValue: "GOLD"
transitions: {}
```

# Passing context using custom fields

JSONRaw DataHeaders

SaveCopy

▼ items:

▼ 0:

tableName:"incidents"

count:16

▼ columnNames:

0:"imported"

1:"cs\_accepted\_offer\_yn"

2:"cs\_accepted\_install\_yn"

⋮

20:"srm\_initial\_post\_date"

21:"srm\_social\_channel\_type"

22:"intent"

23:"vipstatus"

```
setupCustomProps:
  component: "System.SetVariable"
  properties:
    variable: "myCustomProps"
    value:
      customerInformation:
        interfaceID:
          name: "solo_financial_1"
        productID:
          id: 178
      customFields:
        - name: "c$intent"
          dataType: "STRING"
          dataValue:
            stringValue: "${intentName}"
        - name: "c$vipstatus"
          dataType: "STRING"
          dataValue:
            stringValue: "GOLD"
      transitions: {}
```

# Passing context using custom fields

- Custom properties passed to Service Cloud via System.AgentInitiation

```
metadata:  
  platformVersion: "1.1"  
  main: true  
  name: "CrcPizzaBot_AB"  
  context:  
    variables:  
      iResult: "nlresult"  
      myCustomProps: "map"  
agentInitiation:  
  component: "System.AgentInitiation"  
  properties:  
    agentChannel: "ServiceCloudIntegration"  
    agentActions: "OrderPizza"  
    nlpResultVariable: "iResult"  
    waitingMessage: "Let me connect you with our Customer"  
    resumedMessage: "Please wait as find we find the best"  
    customProperties: "${myCustomProps.value}"  
  transitions:  
    actions:  
      accepted: "agentConversation"  
      rejected: "reject"  
      OrderPizza: "OrderPizza"
```



A diagram illustrating the flow of custom properties. A red box highlights the `myCustomProps: "map"` field within the `context.variables` section of the `metadata` block. A blue arrow points from this box to another red box that highlights the `customProperties: "${myCustomProps.value}"` field within the `properties` section of the `agentInitiation` block.

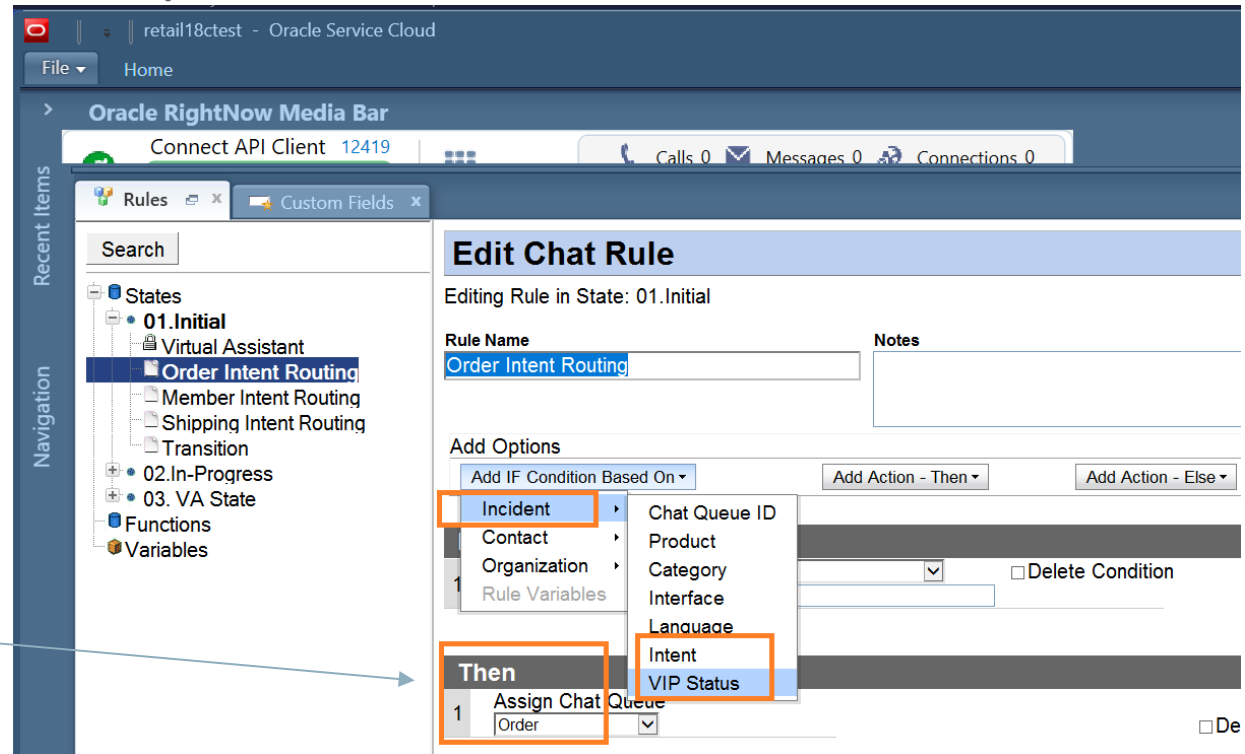
# Topic agenda

- 1 Overview of agent integration
- 2 Use case in action
- 3 How it works
- 4 Context transfer using custom properties
- 5 Using custom properties for queue routing**

# Using custom properties for queue routing

- Based on custom property you may decide to route request to a specific agent queue
  - If VIP status then route to order queue

```
setupCustomProps:  
  component: "System.SetVariable"  
  properties:  
    variable: "myCustomProps"  
    value:  
      customerInformation:  
        interfaceID:  
          name: "solo_financial_1"  
        productID:  
          id: 178  
      customFields:  
        - name: "c$intent"  
          dataType: "STRING"  
          dataValue:  
            stringValue: "${intentName}"  
        - name: "c$vipstatus"  
          dataType: "STRING"  
          dataValue:  
            stringValue: "GOLD"  
      transitions: {}
```



ORACLE®