
ORACLE®

Oracle Digital Assistant

The Complete Training

Human Agent Integration

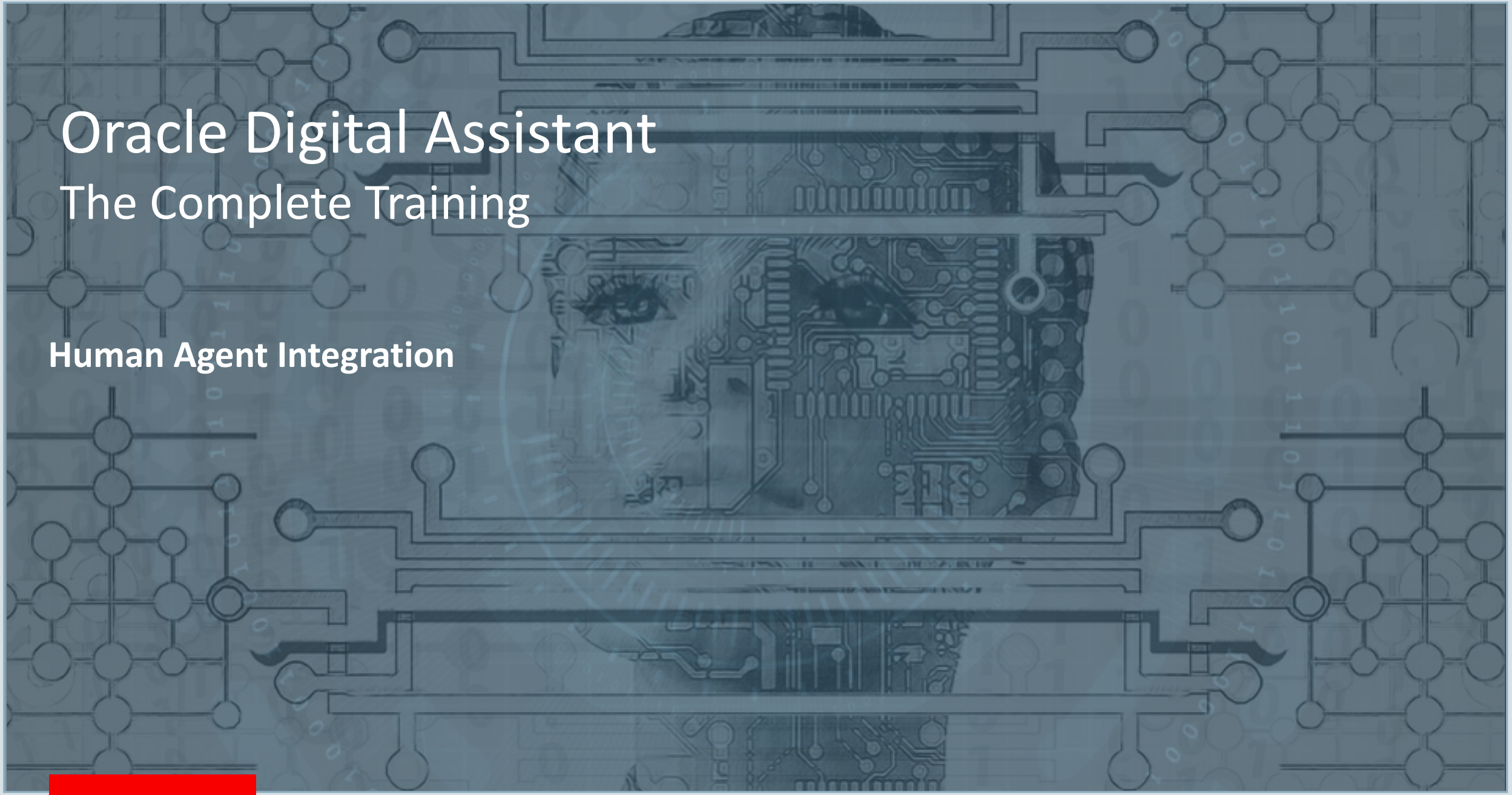


Image courtesy of pixabay.com

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. 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

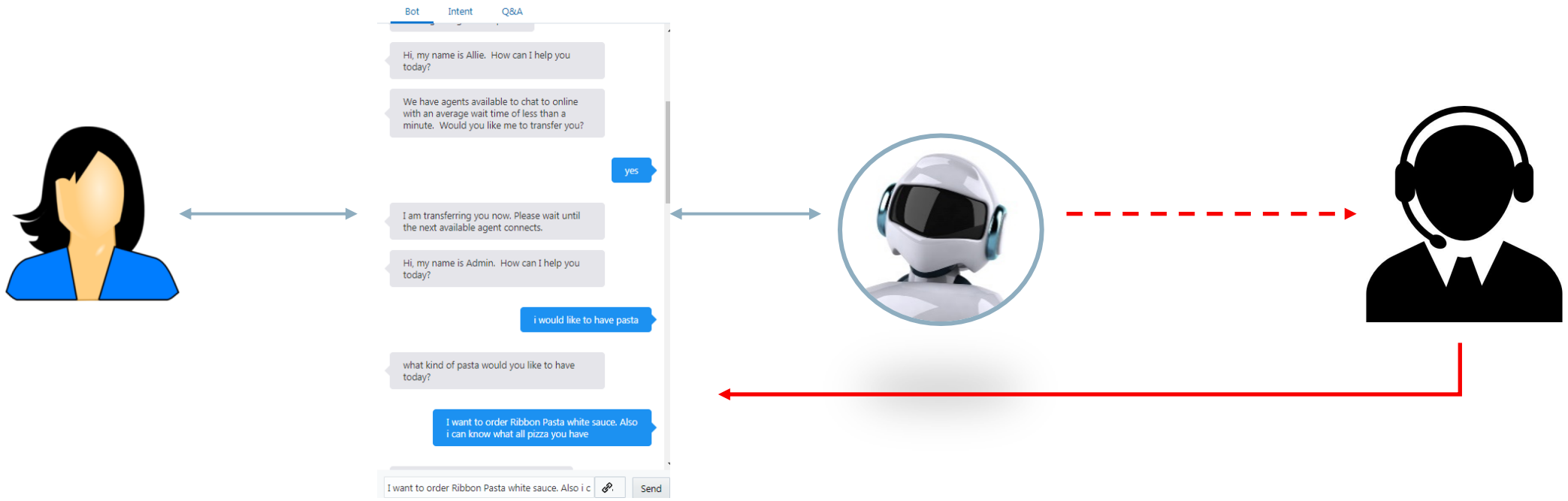
1. Overview of agent integration
2. Use case in action
3. How it works
4. Context transfer using custom properties
5. Using custom properties for queue routing

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**

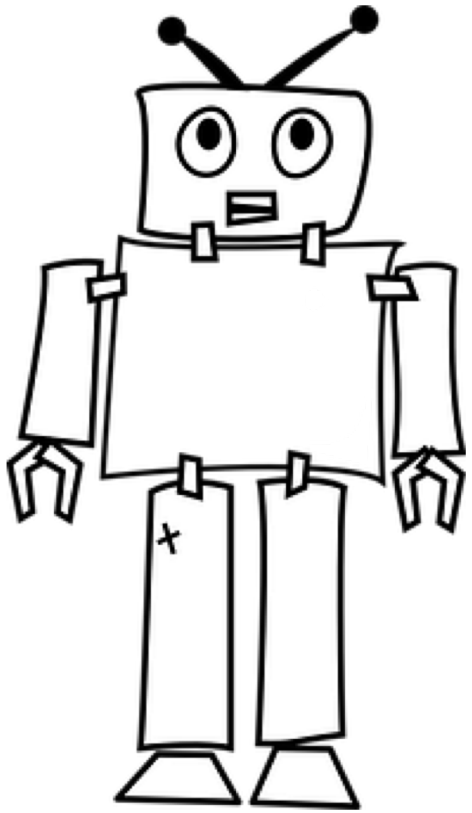


Image courtesy of Farbentek at FreeDigitalPhotos.net

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.

ORACLE SERVICE CLOUD

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

I want to check status of my order.

Your order is ready but it has not been delivered.

Let me connect you with our customer service agents who can provide a status of your order!

FoodBot manages the conversation with the end-user until request for agent is detected

Waiting for agent's response

FoodBot performs bot-handoff to human

Use case – FoodBot agent integration

Hi, my name is Allie. Sorry to hear about the delay..

I just called the driver and he is outside your house now.

I apologize for the delay. May I offer a discount for your next purchase ? Use code "PIZZA50"

Allie now types FoodBot's response and can see user's answers as if they were chatting one-to-one.

Allie can disconnect from conversation and FoodBot can continue

Is there anything else I can help you today with?

No, thank you.

After Allie disconnects, FoodBot digital assistant takes over

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**.

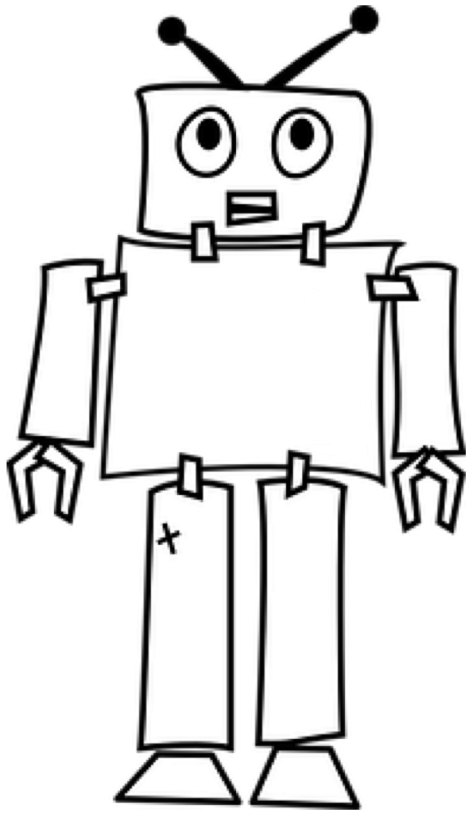
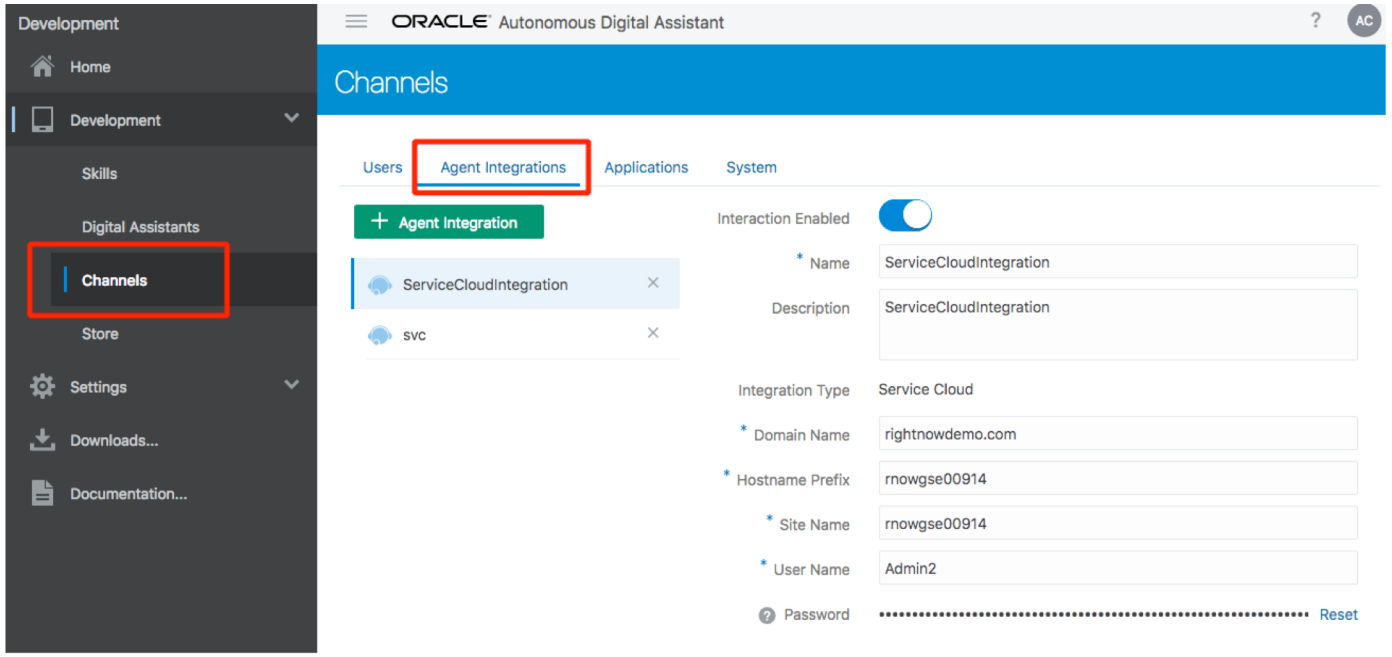


Image courtesy of Farbentek at FreeDigitalPhotos.net

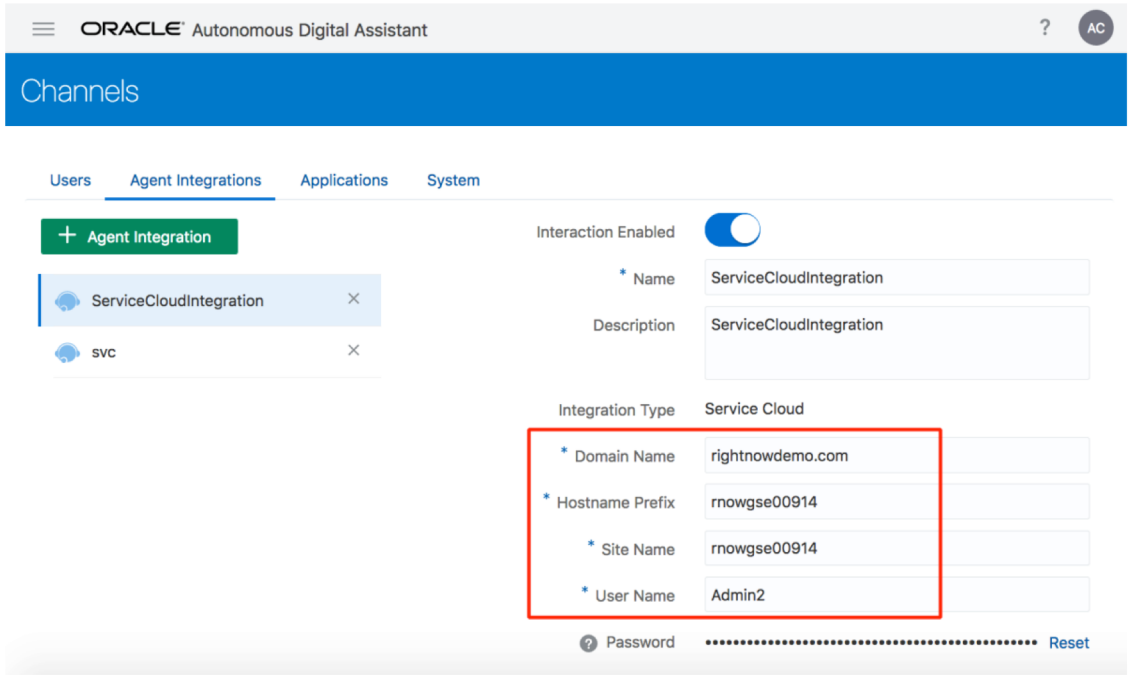
Configure agent integration

- 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

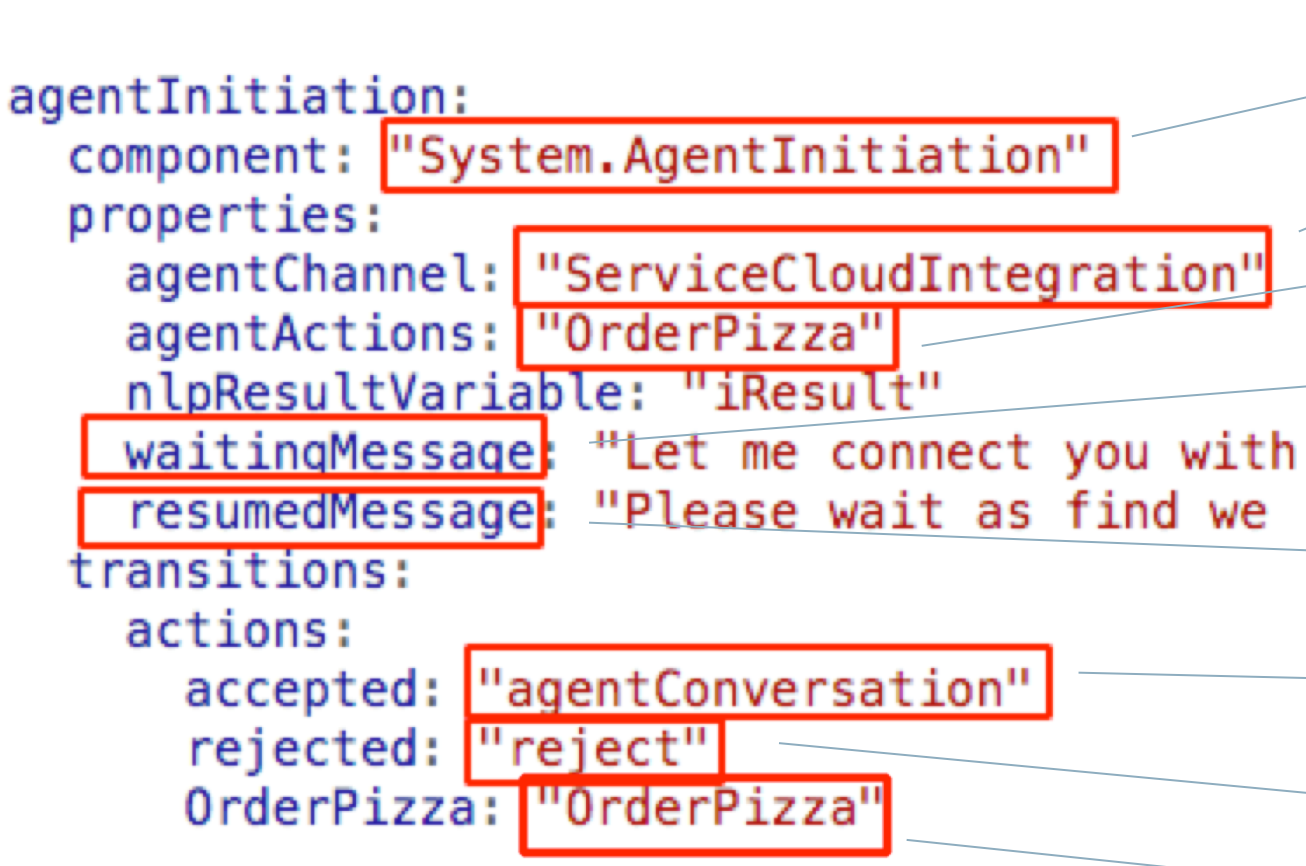
- Refer to Service Cloud instance to get details for hostname and site name prefix
  - Get this info from your Service Cloud admin
  - Hostname or sitename typically same as interface name
    - https://<ServiceCloud>.rightnowdemo.com/services/soap/connect/chat_soap?wsdl



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



Initiates the handshake with external Agent system

Name of the channel

List of actions that will be available for human Agent that can be directly fired

Message to be given to user while waiting for the agent to connect

Message to be given to user if the user sends repeated messages to connect

Action to be taken once ticket is accepted in External agent system

Action to be taken if no agent is available

OrderPizza is the dialog state called when the agent triggers and an action using /OrderPizza command in the chat window.

Start agent conversation in dialog flow

```
agentConversation:
  component: "System.AgentConversation"
  properties:
    agentChannel: "ServiceCloudIntegration"
    nlpResultVariable: "iResult"
    exitKeywords: "bye, bye, good night, end, quit"
    conclusionMessage: "Have a nice day."
  transitions:
    next: "endPrompt"
```

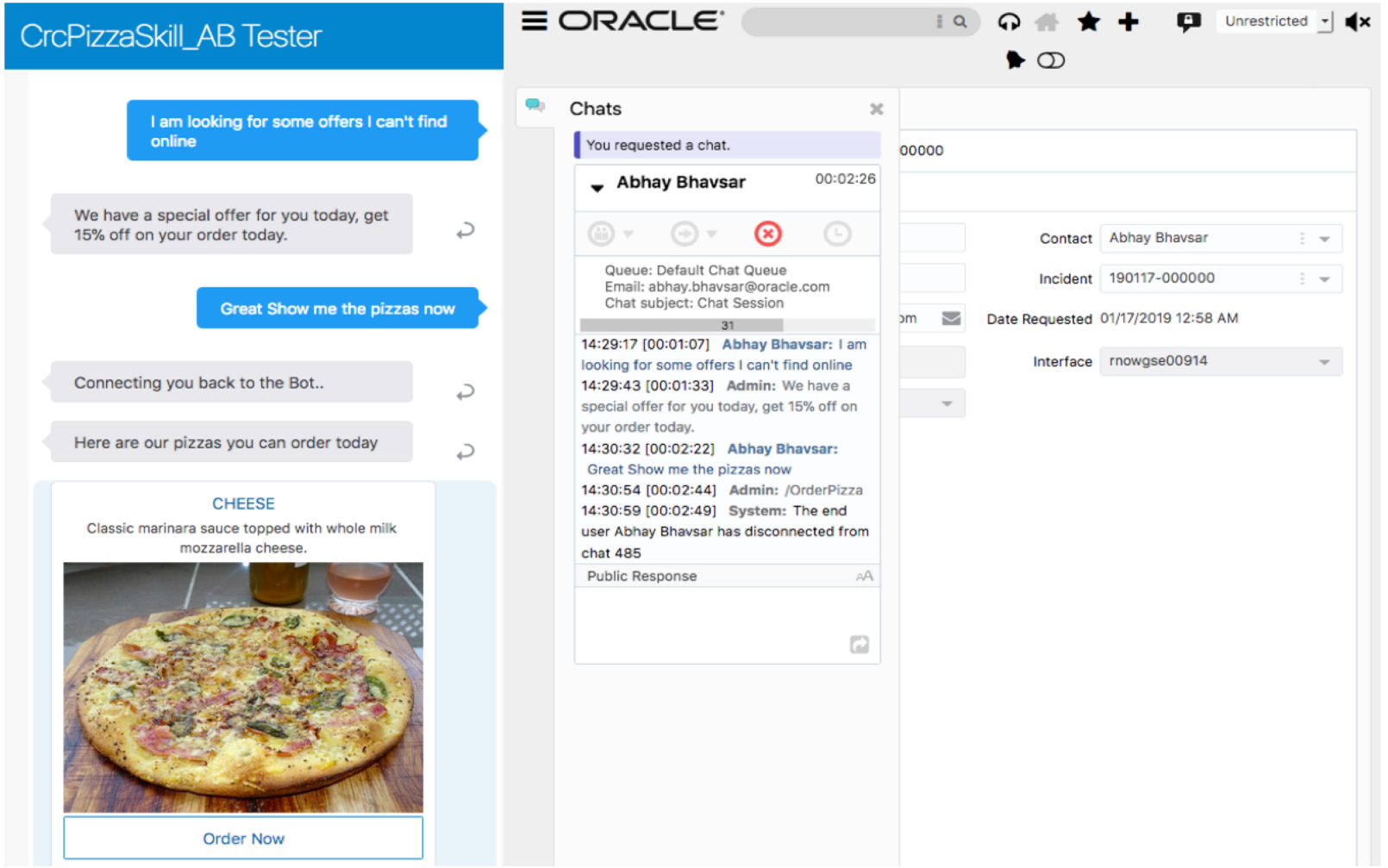
Manages message processing between the user and the agent

Name of the channel

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

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

A typical user – bot – agent conversation

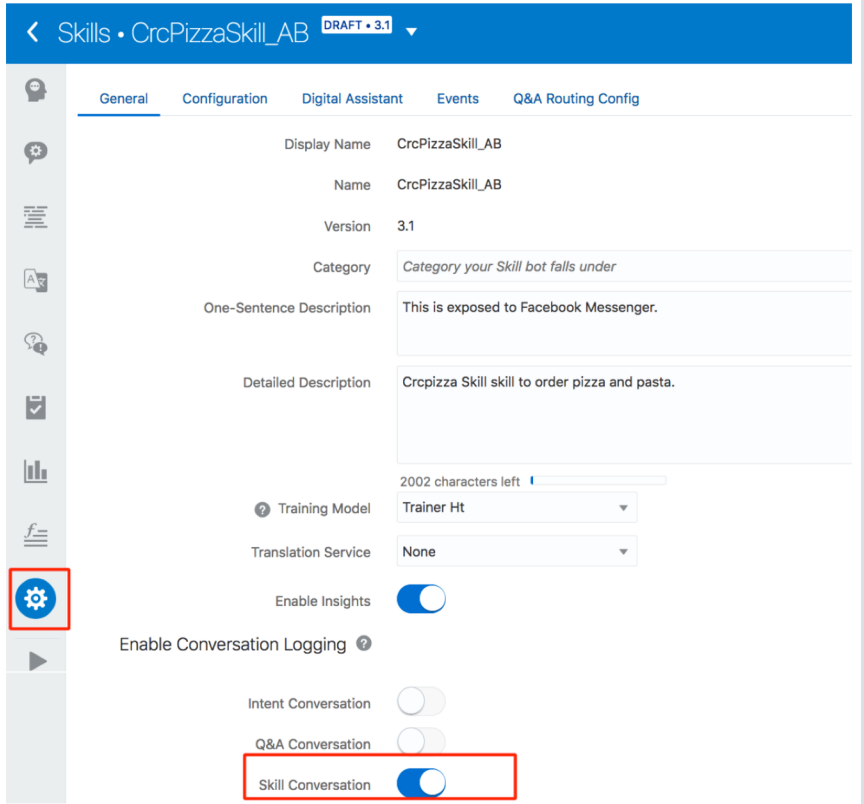


Topic agenda

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

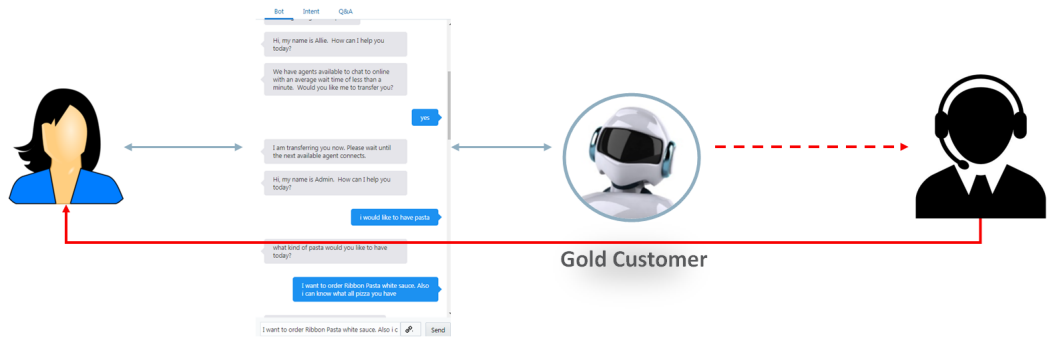
Context transfer using custom properties

- 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

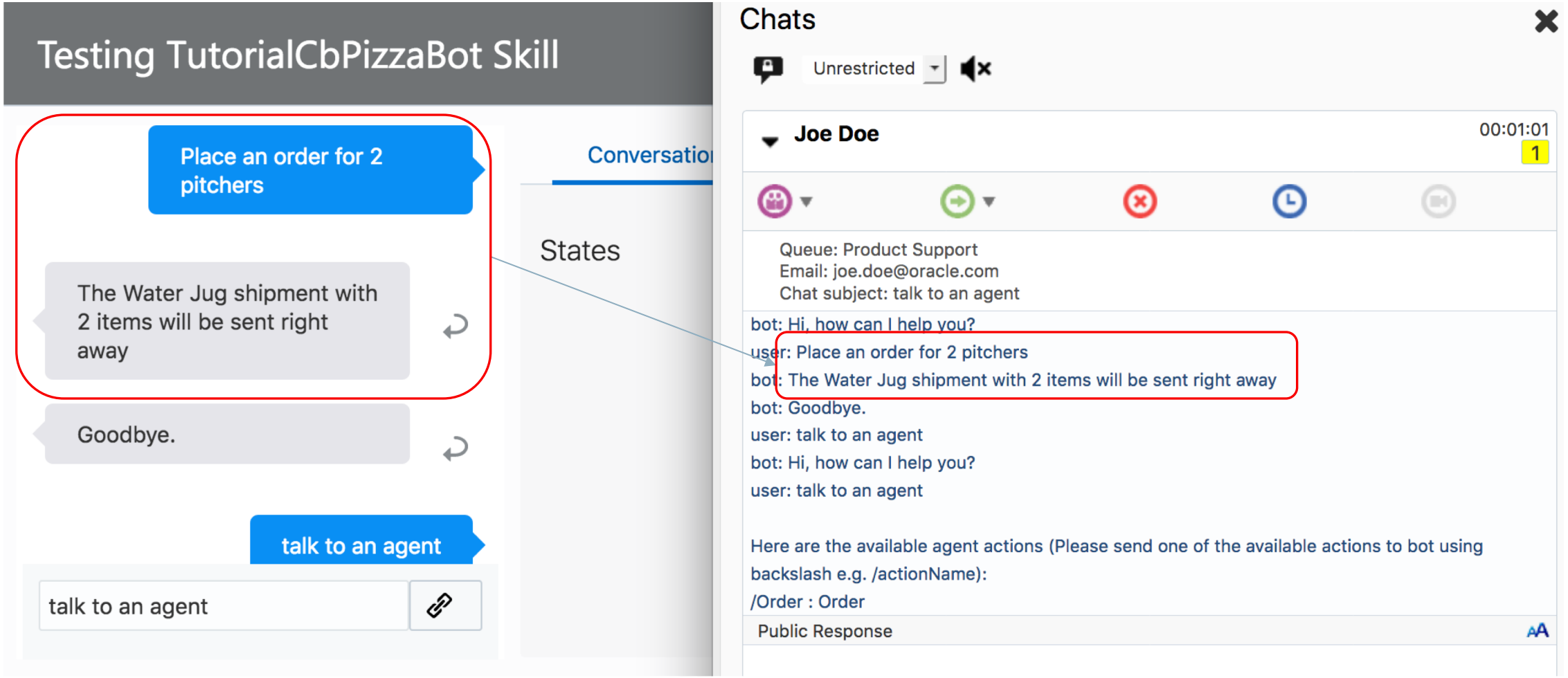


Three ways of passing context

- 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



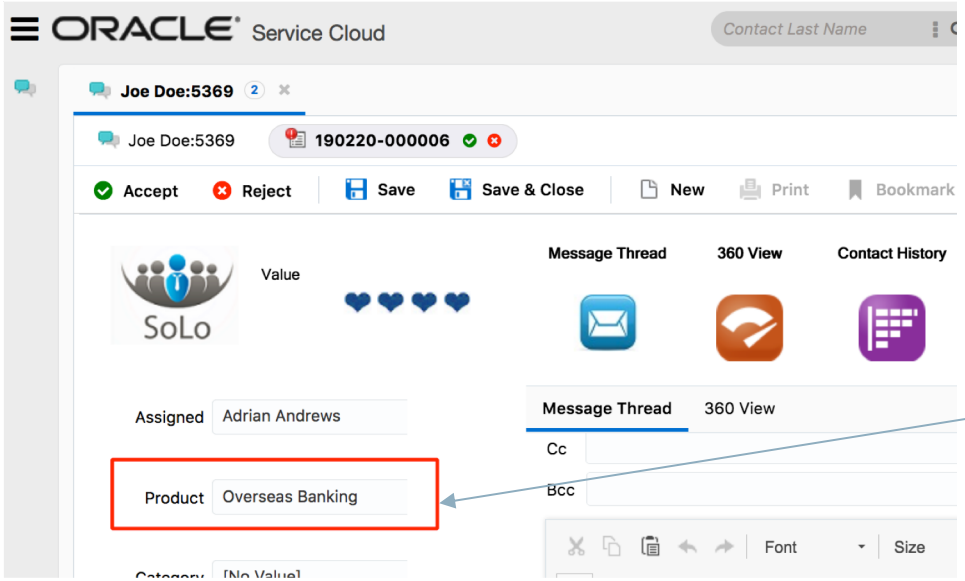
Context passed in conversation logs



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: "nlpresult"
    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 they values in the flow
- Refer to SOAP API to understand properties and types
  - 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

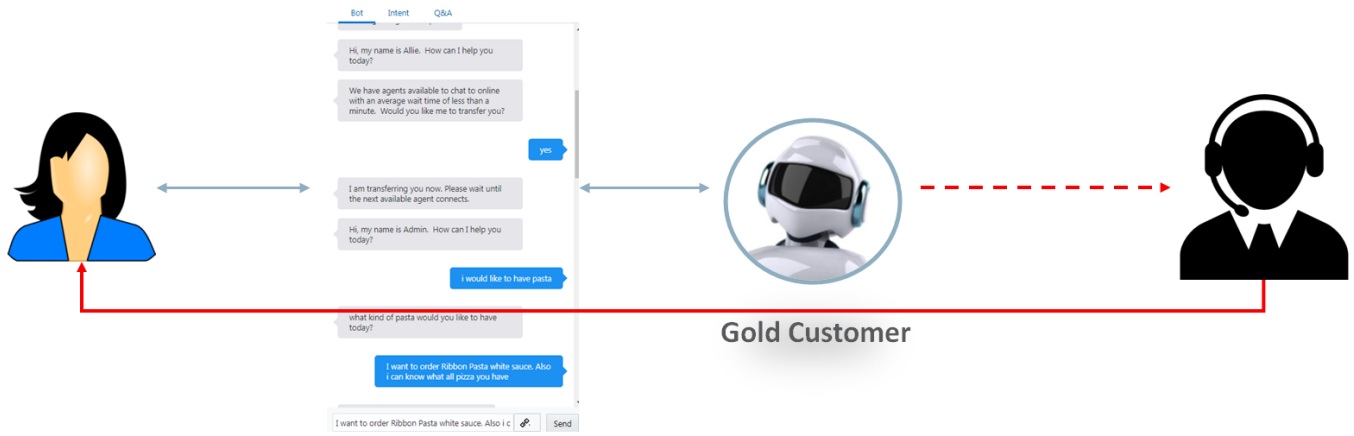
- In dialog flow we have a map variable with a particular format
  - customerInformation – The format of the customer information is as per the SOAP WSDL
    - Note that interfaceID can be referred by name as well as id unlike productID

```
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.

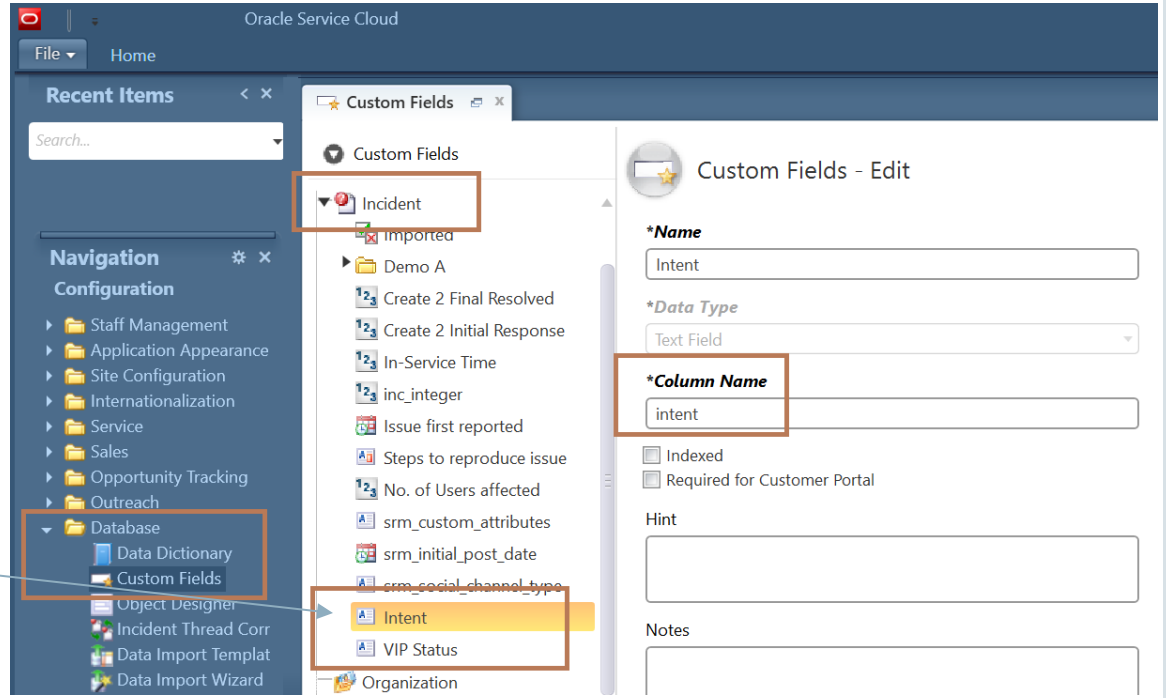
Passing context using custom fields

- 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

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

```
metadata:
  platformVersion: "1.1"
main: true
name: "CrcPizzaBot_AB"
context:
  variables:
    iResult: "nlpresult"
    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

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"
      ...
      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: "nlpresult"
    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"
```

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

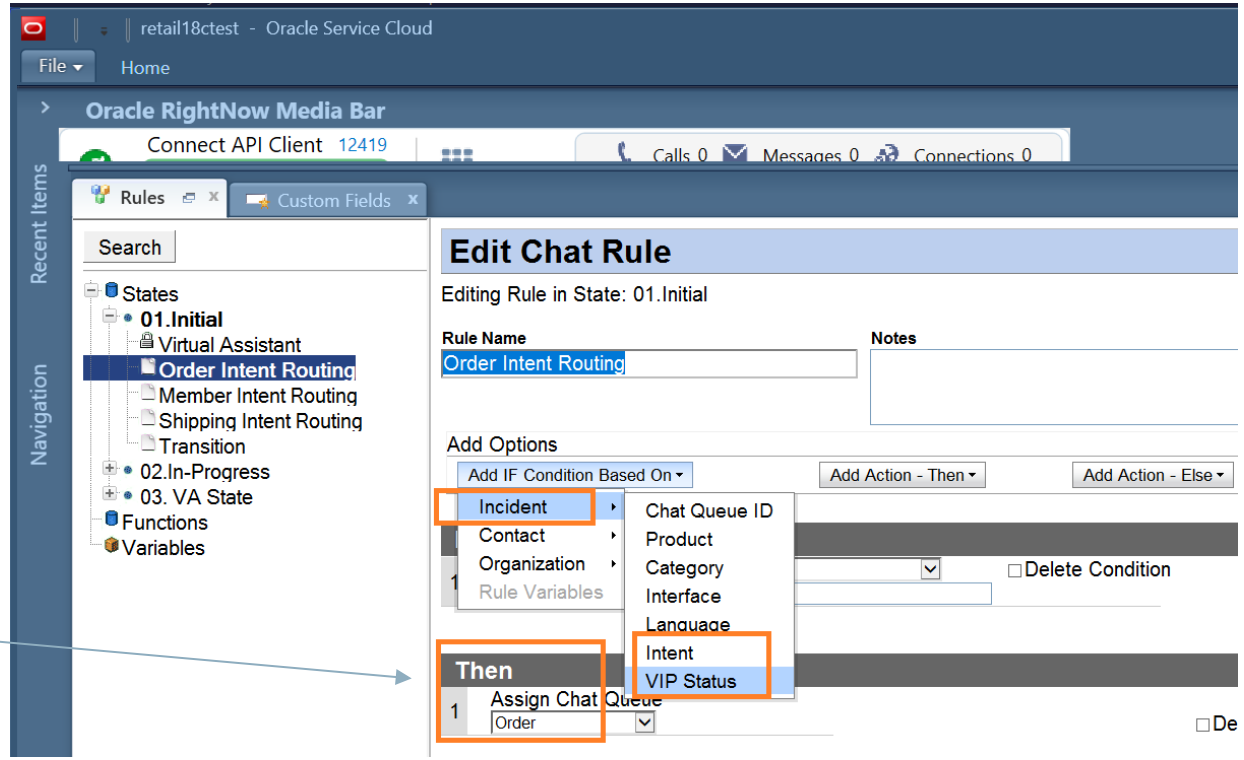
- Based on custom property you may decide to route request to a specific agent queue
  - If VIP status then route to order queue

```
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®