
ORACLE®

Oracle Digital Assistant

The Complete Training

Custom Component SDK

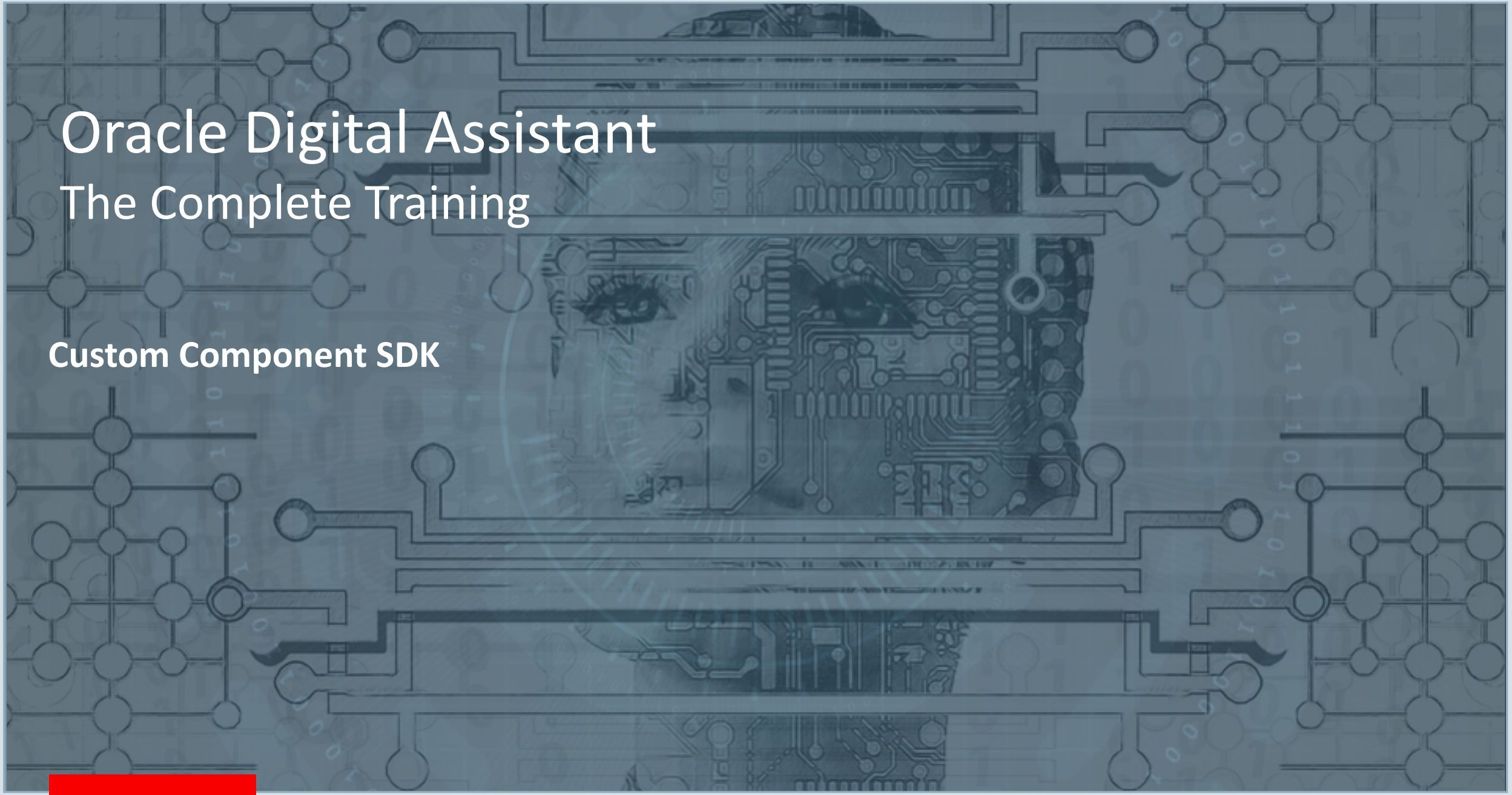


Image courtesy of pixabay.com

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

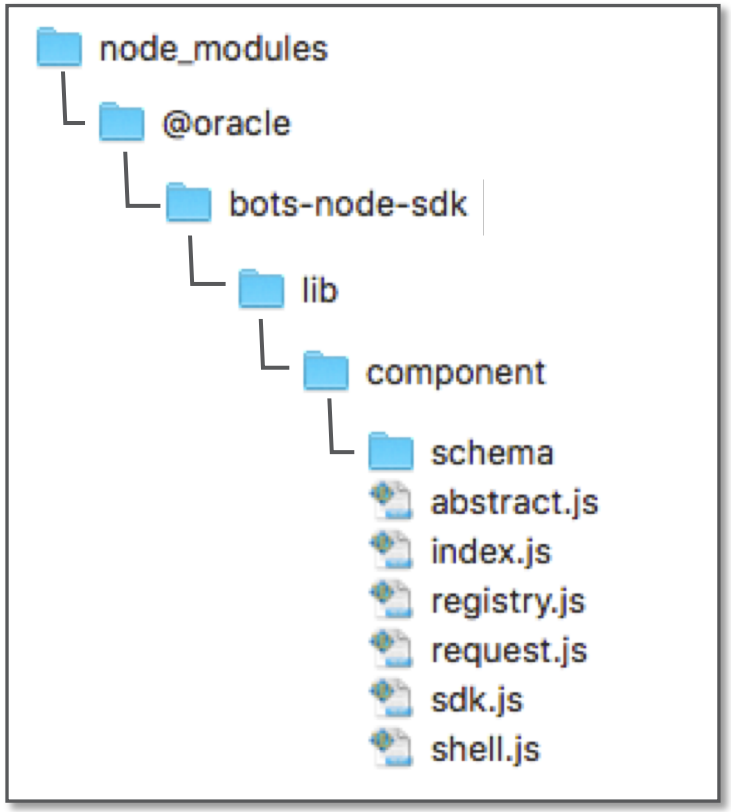
1. SDK, the complete tour for developers
2. Conversation Message Model (CMM)
3. CMM writing bot responses
4. CMM reading user messages

Topic agenda

1. SDK, the complete tour for developers
2. Conversation Message Model (CMM)
3. CMM writing bot responses
4. CMM reading user messages

Role of the custom component SDK

- Contained in Oracle Bots Node.js SDK
- Abstracts the bot payload
- Expose functions to
  - Read from the bot payload
  - Write to the component response payload
- Grant access to Message Model
  - Simplifies creation of rich responses
    - Text, Cards, Attachment, Actions
- Helps to determine type of bot request
  - Action, Text, Attachment, Location etc.



Accessing bot messages

conversation.rawPayload()	{"messagePayload":{"text":"hello world","type":"**text**"},"profile":{"firstName":"john","lastName":"doe", "timezoneOffset":-3600000,"locale":"en-US"},"userId":"8547335"}
conversation.messagePayload()	{"type":"**text**","text":"hello world"}

Accessing messages – text, postback, location, attachment

conversation.text()	• Returns text message or null if message type is not text
conversation.postBack()	• Postback messages are sent from action items (e.g. button) • Returns a JSON object with key value pairs • Returns null if message is not a postback
conversation.location()	• Returns JSON object with longitude and latitude information • Returns null if message is not a location message
conversation.attachment()	• Users may use the CR component to upload image, video, files and audio content. The url of the attachment content is sent to the component. • Returns JSON object with the url of a user provided attachment • Returns null if message is not an attachment message

Sending response messages to the bot

conversation. reply(…)	• Returns message directly to user • Arguments can be of string, object and MessageModel type • Can be called multiple times for a single response • Sets keepTurn to false
conversation. reply(String)	• Sends a simple text message in a bubble • No formatting. Just plain text.
conversation. reply(ConversationMessage)	• Conversion Message Model message • Supports card - , attachement -, location - , postBack -, raw responses

Working with variables and properties

conversation.variable('variable_name')	• Reads the value from a named contex variable
conversation.variable('variable_name', value)	• Updates a named context variable • If the variable doesn't exist, it will be created
conversation. properties().property_name	• Reads value from a input parameter

Accessing intents and entities from NLP processing

var nlpresult = conversation.nlpResult(string)	• Accesses the nlpresult instance for the nlpresult variable referenced in the argument (e.g. iResult) • Argument can be empty if only a single nlpresult variable is used in a dialog flow.
nlpresult.entityMatches(string)	• Returns an array of entities extracted from the user string that match the provided entity name • If no value is provided then all entities found in the user string are returned
nlpresult.intentMatches()	• Returns array of intents resolved for the processed user string • Intent is a JSON object with an "intent" and a "score" property
nlpresult.topIntentMatch	• Returns the top resolved intent as a JSON object with an "intent" and score" property

Trigger dialog flow navigation

conversation.transition(string)

- Custom components may define supported action strings in their metadata that dialog flow designers use to determine the next state to navigate to
- Custom component developers use the transition(...) function to return an action string
- Its up to the dialog flow designer to use the action string for navigation or to ignore it

Request user input

conversation.keepTurn(true)	• Used by components that need to print a message to the user but don't require the user to provide an input in response • Use case would be to print a statement
conversation.keepTurn(false)	• Most commonly used • User is required to provide an input in response to a component response • Use case would be a list of value printed by the component that the user should select a vaue from

The combination of **keepTurn(boolean) and transition(string)** determines when and how the dialog flow navigates to a next state

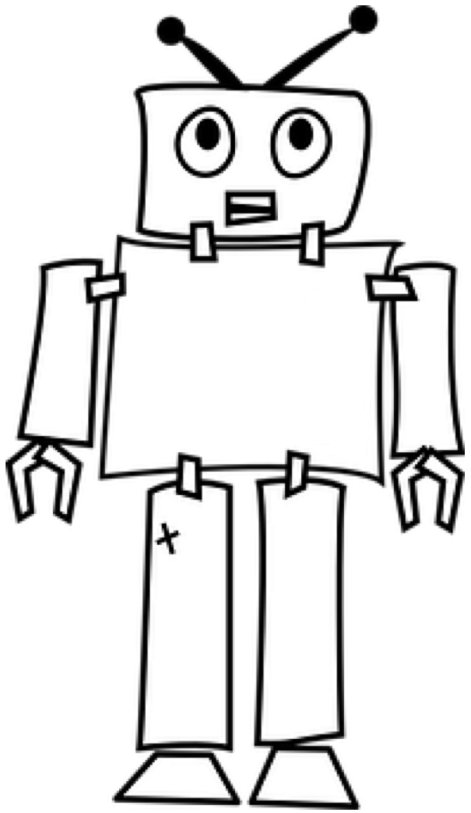
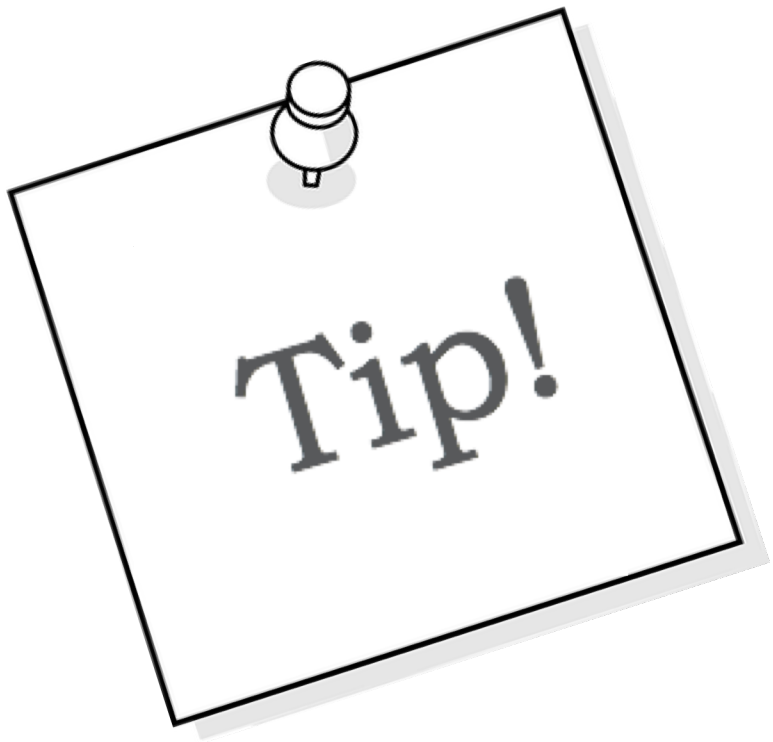


Image courtesy of pixabay.com



Always. Call

```
conversation.keepTurn(boolean)
```

after

```
conversation.reply(…)
```

Invalid user input

conversation.invalidUserInput(string)

- Allows non-sequitur handling of user messages that a custom component considers invalid
- String argument is the message printed to the user
- Custom component does not *conversation.transition(...)* but calls *done()* immediately
- Digital Asissistant response to invalidUserInput(string) call and tries to find a skill matching the user messages.
  - If found, non-sequitur routing happens.
  - If not found then the custom component is called again. Thus the string argument so the user knows what went wrong.
- Hint: set keepTurn to false (conversation.keepTurn(false)

Logging

conversation.logger().logLevel(…)

- Writes log messages using the logger configured in the SDK
- Default logger is *console*
- log(...), error(...), warn(...), info(...)

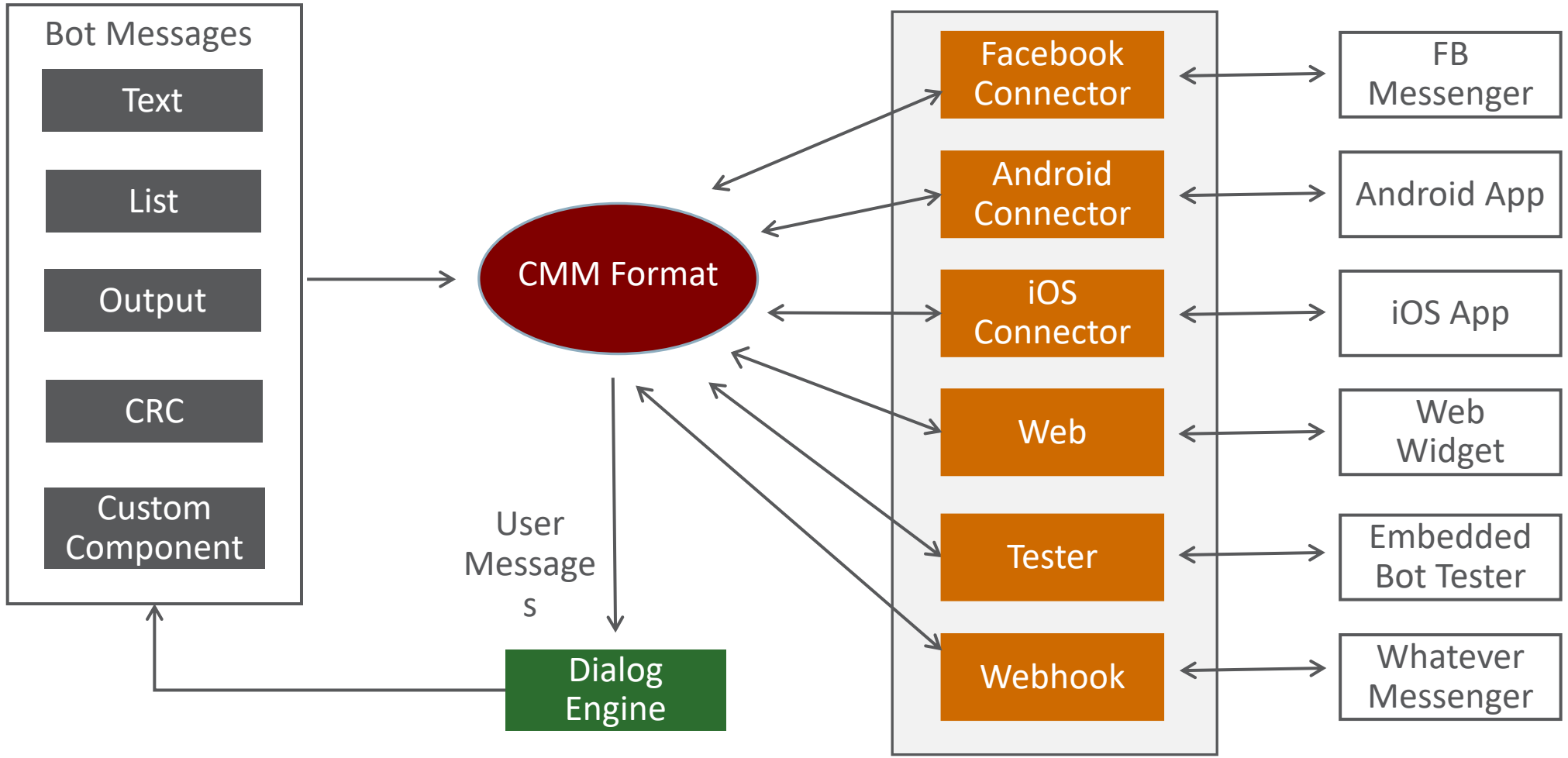
Topic agenda

1. SDK, the complete tour for developers
2. Conversation Message Model (CMM)
3. CMM writing bot responses
4. CMM reading user messages

Conversation Message Model (CMM)

- Consistent messages structure format
- Use of connectors to produce channel specific message responses
- Webhook returns CMM message structure as is
- Design-time support
  - Common Response Component
  - Custom Component (SDK)
    - Exposes MessageModel object to work with CMM messages

Oracle Digital Assistant CMM Architecture



Topic agenda

1. SDK, the complete tour for developers
2. Conversation Message Model (CMM)
3. **CMM writing bot responses**
4. CMM reading user messages

TextConversationMessage

- Writes a text message to the messenger client
  - Simple text, JSON object
- Optionally, takes an array of actions that are rendered below text
  - postbackActionObject, locationActionObject, urlActionObject, shareActionObject

```
var messageModel = conversation.MessageModel();

var payload = 'hello world';
var actions = [];
…
var textResponse = messageModel.textConversationMessage(payload, actions);
conversation.reply(textResponse );
```

CardConversationMessage

- Shows responses as an array of vertical or horizontal cards
- Optionally, takes an array of actions that are rendered on each card
  - postbackActionObject, locationActionObject, urlActionObject, shareActionObject

```
var messageModel = conversation.MessageModel();

var cardObject = messageModel.cardObject("title string", "description",
"imageUrl", "card Url", [array of actions]);

var cards = [];
cards.push(cardObject);

var cardsResponse = messageModel.cardConversationMessage('vertical', cards);
conversation.reply(cardsResponse );
```

AttachmentConversationMessage

- Creates image, video, audio, file response
- Each attachment is rendered in its own bubble

```
var messageModel = conversation.MessageModel();

var type = "image" //video, audio, file
Var docUrl = "http://host:port/my_image.jpg";

var attachmentResponse = messageModel.attachmentConversationMessage(type,docUrl)

conversation.reply(attachmentResponse );
…
```

CMM actions

- Rendered as actionable items in a bot response
  - List item, button, url
- Example: Postback action
  - Individual user payload (string or JSON object) when action is selected
  - `var action = messageModel.postbackActionObject( label, imageURL, payload)`
- Call action
  - A phone number to dial on a mobile device
  - `var action = messageModel.callActionObject( label, imageUrl, phoneNumber)`

CMM actions

- URL action
  - A web address to open in a mobile browser
  - `var action = messageModel.urlActionObject(label, imageUrl, url);`
- Share action
  - Ability to share messages in a messenger
  - `var action = messageModel.shareActionObject(label, imageUrl);`
- Location action
  - Ability to request and receive user GPS location
  - `var action = messageModel.locationActionObject(label, imageUrl);`

Topic agenda

1. SDK, the complete tour for developers
2. Conversation Message Model (CMM)
3. CMM writing bot responses
4. CMM reading user messages

Determine the user input message type

- `conversation.text()` returns text string or null
  - If(conversation.text()) { … handle text response ..}
- `conversation.attachment()` returns attachment string or null
  - If(conversation.attachment()) { … handle attachment response ..}
- `conversation.location()` returns location JSON object or null
  - If(conversation.location()) { … handle location response ..}
- `conversation.postback()` returns postback JSON object or null
  - If(conversation.postback()) { … handle potsback response ..}

Example: handling of a postback message

```
invoke: (conversation, done) => {
  const pizzaType = conversation.properties().pizzaType;
  //handle postback response
  if (conversation.postback()) {
    //postback contains a key-value object
    let postbackpayload = conversation.postback();
    //update 'pizzaType' context variable with 'pizza' key postback value
    conversation.variable(pizzaType,
                          postbackpayload["pizza"]);
    conversation.transition();
    done();
  }
```

Integrated Cloud

Applications & Platform Services


ORACLE®

Oracle Digital Assistant Hands-On

TBD