Build Oracle Intelligent Bots on Oracle Mobile Cloud Service
What Is the Oracle Intelligent Bots Platform?
Channel Configurator
Dialog Flow Engine
Custom Components in the Dialog Flow
Support for the NLP Engine
Chatbot Building Blocks: Intent, Entities, Utterances, and Dialogs
Intent: What Does the User Want?
Utterances: What Does the User Say?
Entities: Can You Tell Me More?
Dialogs: Do You Want Me to Order a Banana Instead of a Pizza?
Train Your Bot to Be Smarter: Machine Learning and NLP
Design a Bot
Build a Bot
Built-in Components
Custom Components
Create Dialogs and Define the Dialog Flow
Test the Bot
Configure Channels
Build Instant Apps
Add Bot Analytics
Try It Yourself
Add a Bot Q&A
Link a Custom Component to Your Bot
What Is a Custom Component Service?
GET Method
POST Method
Implement Custom Components
Understand the Architecture of Custom Components
Component Service
Shell
Registry
SDK
MessageModel
Shared APIs
REST Connector
Build a Custom Component
Summary
Additional Resources
Build Oracle Intelligent Bots on Oracle Mobile Cloud Service

The Oracle Intelligent Bots platform is an integrated feature of Oracle Mobile Cloud Service. With it, you can develop bots that can hold natural language conversations and help users to complete tasks. You can then connect the bots to your enterprise system to answer user queries and complete tasks such as booking a ticket, finding a work order status, or providing the latest inventory update.

By using artificial intelligence (AI) and natural language processing (NLP) powered by neural networks and machine learning, Oracle Intelligent Bots can detect the user’s intent and respond with relevant information.

The platform makes it simple and easy to build and train intelligent bots without the need for AI specialists. After you build and test your bots, you can deploy them on voice and text messaging channels, a custom mobile app, or a website.

Oracle Mobile Cloud Service consists of several lines of business components, including Mobile Core, Intelligent Bots, Customer Analytics, and Client Development Tools. Developers, architects, and development and IT operations teams are the primary users of these components.
What Is the Oracle Intelligent Bots Platform?

Oracle Intelligent Bots is a conversational platform for the enterprise. It comes with a complete set of tools, framework, hosting, and integration features to provide a comprehensive bot development environment. This general-purpose, scalable, and powerful platform helps you develop chatbots and integrate with popular text and voice-based messaging services, such as SMS, Slack, Skype, Amazon’s Alexa, Facebook Messenger, Microsoft Teams, Apple’s Siri, and Google Assistant as well as with software development kits (SDKs) that allow you to extend your existing mobile and web applications with chat capabilities.

By packaging Oracle Intelligent Bots with Oracle Mobile Cloud Service, Oracle helps you connect to powerful enterprise back-end systems. With those systems, you can feed the bots data to train, learn, and understand natural language conversations between user and bot.

As a bot developer, you can also take advantage of features such as geo-location services, push notifications, support for mobile client platforms, analytics, connectors, and security. With those features, you can build truly intelligent and powerful bots capable of handling complex user queries.

### Oracle Intelligent Chatbot Service: Key Components

<table>
<thead>
<tr>
<th><strong>Channel Configurator</strong></th>
<th>Unifies cross-channel chat interface between bots and humans.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dialog Flow Engine</strong></td>
<td>Declarative bot builder UI for botflow. State machine that executes context-driven workflows with scoped variables.</td>
</tr>
<tr>
<td><strong>AI Engine</strong></td>
<td>Active machine learning algorithms that identify relationships and extract insights from unstructured data.</td>
</tr>
<tr>
<td><strong>Channel Configurator</strong></td>
<td>SDK to consume back-end APIs that satisfy humans requests via the bot.</td>
</tr>
</tbody>
</table>

### Channel Configurator

Oracle Intelligent Bots provides easy integration with a number of text and voice-based messaging services. You can also run your bot on other messaging services and websites by configuring a channel with webhooks.

### Dialog Flow Engine

As the interface between the messaging channels and the back-end enterprise services, the dialog flow engine is the core of bots. As a state machine, it executes the dialog flows that drive the bot-human conversation. The dialog flow engine also keeps track of the user’s context within the dialog flow. It lets users switch from one action to another, and then pick up right where they left off in the dialog flow. The dialog engine also helps maintain user memory to help deliver a better user experience when the user re-engages with the bot.

### Custom Components in the Dialog Flow

Components are the units of work within the dialog flow, and they perform the various functions required by your bot.
Oracle Intelligent Bots provides a set of built-in components to carry out common tasks, like asking the user to enter text or prompting for a response. Although these built-in components can help you tie together the various steps in your bot-human conversation, you can also add custom components so that your bot can perform specific tasks, like checking bank balances or booking fares. Using the bot SDK, you can build components that execute custom business logic within the dialog flow. You do that by calling the custom code APIs that run it in the Oracle Mobile Cloud Service back end.

Learn more about custom APIs and custom components.

Support for the NLP Engine

To interpret user requests, bots use their own language intelligence (LI) framework based on the NLP concepts of intents and entities. An intent brings together user input into a specific category, which helps the bot to take action or perform a task based on the user input. Entities enable you to label and classify the vital pieces of information that are automatically extracted from the user’s requests. In a banking transaction, for example, an entity named “account” identifies terms like savings, checking, and CD.

Using named entity recognition, the LI framework combs through the user input and extracts the words and phrases identified by the entities. These words and phrases become the parameters that make the dialog flow engine execute an action or prompt the user for input.

Chatbot Building Blocks: Intent, Entities, Utterances, and Dialogs

Chatbots interact with humans in conversational mode on messaging services, such as Facebook Messenger or Slack, and voice-based platforms, such as Amazon Alexa and Google Home. To conduct a meaningful conversation with a human, the bot must understand what the human wants and then respond.

In a bot-human conversation, the human input is categorized into intent, entities, utterances, and dialogs.

Intent: What Does the User Want?

When you create a chatbot application in Oracle Intelligent Bots, your first task is to create intents. An intent illustrates your use case by describing the various actions that your bot helps users complete. Intent describes not only what your bot can do but also enables the bot to recognize user input by training it with sample utterances. Utterances are simply the set of statements that the user is likely to make when interacting with the bot.
The NLP algorithm behind a chatbot application takes an utterance and classifies it to a predetermined intent. This intent then tells the chatbot application what task to perform and how to complete the task. For example, when a user says, “I want to order a pizza,” the chatbot application maps this user input or utterance to the OrderPizza intent and then instructs the bot to perform the actions associated with this intent.

Utterances: What Does the User Say?

Machine learning algorithms must be trained with data to understand user input and context. Utterances are a quick way to train the ML model without requiring huge amounts of data. Utterances are common sentences that might be associated with the intent. Think of an utterance as a kind of sample data that the bot’s AI element uses to train itself. Utterances tell the bot what the user wants, but, given that users can ask questions in several different ways, utterances might not be expressed clearly. For example, these utterances have the same meaning: “Can you order a pizza for me?” and “Order me a pizza.”

Utterances train the bots to interpret the meaning of user-generated sentences. Typically, ML algorithms are trained with a set of seven to 10 utterances for each intent. When trained, the algorithms are able to classify user input even if the input were not included in the original corpus of data. As users engage with the bot and ask different questions or utterances, the bot continues to learn and increase its vocabulary as well as accuracy of predicting the right intent.

Entities: Can You Tell Me More?

An entity adds context and metadata for an intent. A bot uses entities to fill in the specifics of the user request so that it can complete the task. For example, if the intent is OrderPizza, the entities for the intent could be PizzaType (veg, nonveg), PizzaSize (small, medium, large), and Toppings (sausages, olives).

Dialogs: Do You Want Me to Order a Banana Instead of a Pizza?

Dialogs are core to the conversational experience. They define the conversation between the user and the bot. After the AI engine identifies the intent of the user input and extracts the entities, the dialog engine analyzes the input to determine if all the information is available to take an action or if additional input is required from the user to perform the action. During a conversation, a user might change the topic and the bot must handle this switch gracefully. For example, when ordering a pizza, a user might ask the bot to order a can of diet soda and ask whether the pizza vendor accepts credit cards. This example reflects a real-life scenario in which users often switch topics. Because bots deal with humans, they must be prepared to handle sudden changes of topics in a conversation and remember to complete the earlier conversation when a user changes topics.
Train Your Bot to Be Smarter: Machine Learning and NLP

An intelligent bot must read, understand, and process human language intelligently. This task invariably involves transforming the bot-human conversation into a format that the bot can easily understand. To accomplish this task, you must harness the power of machine learning and NLP.

Although machine learning enables bots to learn things by themselves and apply the learning to accomplish tasks without human intervention, NLP provides a system for the bots to read and understand human conversations. This process involves understanding the context of the conversation, determining the intent, identifying the entities, and defining the dialog flow of the conversation.

Oracle Intelligent Bots uses the power of machine learning to extract meaning from bot-human conversations and train the bots to become smarter when accomplishing common tasks. Learn how Oracle Intelligent Bots uses machine learning to drive conversations and train your bot.

Design a Bot

When you design a bot, you must adopt certain design principles to make your bot the preferred mode for addressing your user’s business needs, rather than waiting in line to talk to an agent in a customer support center. To ensure the primacy of your bot, it’s imperative to have the right platform upon which to build it.

Oracle Intelligent Bots equips you with the tools that you need. With Oracle Mobile Cloud Service, your bot learns and adapts with real data that plugs into the back-end system of your organization. You can also create custom components that make REST API calls to back-end services to complete tasks and use the data to train the bots.

An important parameter to measure the effectiveness of your bot is how quickly it solves customer issues. No amount of NLP capability or access to big data can compensate for a bot that can’t solve common customer issues.

However, by using Oracle Intelligent Bots, you maximize the chances of creating a highly successful bot. You leverage the limitless power of AI and machine learning in the form of components and access to big data within an organization.
Build a Bot

Before you start creating your first bot, you must have an Oracle Cloud account. If your company already has a subscription for Oracle Mobile Cloud Service, you can start building your bot right away. If not, sign up for a free 30-day Oracle Cloud account.

Building a bot involves a series of sequential tasks that help you put together the building blocks of a chatbot application.

1. Create intents to represent tasks performed by the bot. The tasks are based on user input during a bot-human conversation. An intent must map to a task, such as OrderPizza or CheckBalance.

2. Add utterances to an intent to train the bot to understand different user expressions that mean the same task. To build robust training, you can add up to two dozen utterances for each intent.

3. Create entities to add context to the intent itself. Entities help describe the intent and enable your bot to complete a user request. The service has built-in entities and custom entities.

4. Train your bot by providing examples to improve its NLP capabilities. After you add your intent, utterances, and entities, train the bot to ensure that it's behaving as expected. To test and train your bot, use the intent tester that's built into Oracle Intelligent Bots.

5. Integrate components so that your bot can do user-requested tasks. By creating intents and entities, you enable your bot to understand what the user wants, but it can’t perform the tasks without the components. For example, if the user says, “I want to order a pizza,” the bot can map this input to the OrderPizza intent. Before the bot completes the task, it must invoke built-in and custom components that make REST API calls, such as authorizing payment for the pizza and getting the user’s pizza preferences.

The high-level steps for creating an intelligent chatbot

1. Creating intents
2. Training the bot
3. Creating entities
4. Integrating custom components
5. Creating dialog flows
6. Testing the bot
7. Configuring channels
Built-in Components

Oracle Intelligent Bots provides a set of built-in components so that bots can perform these common tasks:

**Authorization**: Enables your bot to secure access tokens from OAuth providers, like Google, Twitter, or LinkedIn

**NLP**: Sets the variable values from the intent and the entities. The NLP component includes subcomponents (System.Intent, System.SetVariable, System.ResetVariables, System.MatchEntity) that allow bots to perform such tasks as detecting user intent and matching a variable to a context that’s defined for an entity

**Output**: Allows a bot to prompt a user to enter text, shows a list of options, and displays messages (System.Text, System.List, System.Output)

**Conditional**: Routes the conversation flow based on whether the user input matches a predetermined value (System.ConditionEquals, System.ConditionExists, System.Switch, System.CopyVariables)

Oracle Intelligent Bots also provides the following set of powerful user interface components:

- **System.Text**: Prompts the user to enter text
- **System.List**: Prompts the user with a list option
- **System.Output**: Displays a message
- **System.CommonResponse**: Outputs content-rich messages
- **System.Interactive**: Integrates your bot with an instant app

You can use the System.CommonResponse model to build a specialized user interface that can include text, action buttons, images, and cards all without writing custom code. Instead, you define the component’s properties and metadata.

Custom Components

With Oracle Intelligent Bots, you can create custom components that are specific to a business need. For example, you build a custom component to find out the customer’s name, age, and account details, and verify the customer’s age or return account information.

Custom components don’t reside in Oracle Intelligent Bots. Instead, this functionality is provided through back-end services that are accessed through and returned from the Component Service REST service. Component Service exposes two REST endpoints: GET and POST. The GET method returns a list of all custom components that are hosted within the component service. The POST method invokes a specific component that’s required to complete a task.

You must configure your custom components before your bot can use them.
Create Dialogs and Define the Dialog Flow

During a bot-human conversation, a dialog flow manages the conversation between the two parties. Whereas websites and apps are UI-based, a bot interacts with a human by using dialogs that enable you to logically separate various areas of the bot functionality and choreograph the conversational flow. For example, you can create one dialog to display the pizza menu and another dialog to order the pizza.

Dialogs may or may not have a graphical user interface. They may contain buttons, text, and other elements such as voice commands or they may be entirely speech-based. Dialogs also contain actions to perform tasks such as invoking other dialogs or processing user input.

Dialogs are connected to each other by a dialog flow. Within this dialog flow, you can specify when to invoke a built-in component and when to invoke a custom component.

In Oracle Intelligent Bots, you define dialog flows with the Bot Builder tool.

This tool is Oracle’s implementation of YAML (YAML Ain’t Markup Language). JavaScript Object Notation (JSON) is a subset of YAML and provides a much faster way to encode or decode, consume memory, and serialize the information. In addition, YAML provides better expression clarity to build a user interface for dialogs using a declarative tool.

The YAML definition has two main parts: context and states. You define the variables that are available across the session within the context node, and you define the definition of the flow itself in the states section.

Learn how to write a dialog flow and understand the details of a dialog flow syntax.

Test the Bot

A well-designed bot must not only solve common customer issues, but it must also gracefully handle situations where users switch topics or use inappropriate language. The bot must continue to evolve with training and display improved intelligence with every interaction. Building a bot involves a number of fuzzy decisions that you must test to ensure that the bot is robust and serves its primary purpose to solve customer issues.

Here are some best practices for testing a bot:

**Brevity:** Start your bot conversation with a brief hello (no more than 90 characters per message). A long opening sentence might discourage your users or they might skim the content.

**Consistency:** Have users enter details in a consistent format. For example, your bot can ask them to provide travel dates in dd/mm/yy format.

**Flexibility:** Make sure that your bot understands natural-language input. For example, the user ignores the preferred travel date format and instead says, “I want to fly to New York tomorrow and return the day after.”
Your bot should understand that the user wants to book a ticket to New York for the next day and return a day later.

**Device interoperability:** Make sure that your bot works on all intended devices and interfaces. It should be able to chat on company websites and social media pages.

**Dialog flow:** Rigorously test your dialog flow definition to ensure that your bot can gracefully handle different states in the conversation. Users can switch between intents during a conversation. Your bot should be able to hold the conversation and then return to the original task. For example, if a user asks a bot to set a reminder, the bot asks the user for the date and time. While the bot is still waiting for input about the reminder, the user asks the bot to update an existing reminder. Decide if you want to support switching intents and then design your dialog flow accordingly.

**Abandon flow:** Have you ever chatted with a friend when she suddenly stops responding because she received a phone call? This scenario will also play out with your chatbot. Have a timeout interval for each input and remind the user when there’s inactivity.

**Graceful exit:** If your bot doesn’t understand a request, it should give options to the user. You might want your bot to hand a conversation over to a human agent, highlight help functionality, or refer users to buttons, quick replies, and the persistent menu.

**Learn** how to test your bot when you create intents and add entries.

**Configure Channels**

Your bot can run on any messaging service that supports webhooks. Before you deploy your bot, you must configure the channels where you want to support your users. Oracle Intelligent Bots provides a number of channels for the bot including Web, Mobile (iOS and Android), Facebook Messenger and a generic webhook channel at the time of publishing this report. Oracle continues to add channels at regular intervals. Review the service website for the latest support list.

**Running** your bot on Facebook Messenger involves setting up a Facebook page, a Facebook app, a page access token, an app secret, the webhook URL, and the verify token, and then testing your bot on Facebook Messenger.

**Configuring** a webhook channel involves accessing a publicly accessible HTTP messaging server that uses a webhook to relay messages between the user device and your bot. You implement the webhook by using POST calls to send and receive bot messages. You publish the inbound and outbound calls in JSON format that the bot can read. The outbound JSON payload contains a text response and a choice response, and the inbound JSON payload contains the X-Hub-Signature header and payload details.

**Using** SDKs, you can integrate your bot with iOS apps, Android apps, and web pages. For any of these integrations, you need to generate the App Id by creating a Web, iOS, or Android channel.
Build Instant Apps

Bots are designed to handle free-flowing, natural language conversations to process user requests. However, the bots sometimes need to collect precise information, such as credit card details or passport information. To help your bot’s users enter this type of information, your bot can call an instant app, which provides forms with labels, options, choices, checkboxes, data fields, and other UI elements.

Oracle Intelligent Bots provides the Instant App builder for a quick and easy way to build instant apps. You can create or customize an instant app from the library on the Install App landing page. Configure the app settings to manage general information about your app, such as name, ID, and internal description.

Specify the layout of the app by selecting panes and elements and adding identifying information, such as Pane IDs. Instant app layouts are highly customizable to suit your business needs.

Define the events and actions that should occur when the user interacts with the instant app. For example, when the user clicks a radio button or when the instant app is sent to the customer, the App Sent event is triggered.

Set up the parameters to pass data from a bot to an instant app. You can then use this data in elements and as part of the JavaScript snippets.

You can use JavaScript snippets to launch another instant app from your current app, set element values, enable elements, set element labels, call external Web APIs, play sound, and send Short Message Service (SMS) alerts.

Add Bot Analytics

By using the Analytics API in a custom component, you can analyze your bot’s performance. The Oracle Mobile Cloud Service platform provides the `conversation.oracleMobile.analytics.postEvent()` object to send data about your bot to the Oracle Mobile Cloud Service Analytics application. You just need to make sure that the application is associated with the back end that’s running the custom component. You also need to make sure that you configure a custom schema so that the application can handle the data.

You also need to make sure that you configure a custom schema so that the application can handle the data.

Learn how to add analytics to the PizzaBot.
Try It Yourself

Try building a chatbot that helps users to order pizza. In this tutorial, you’ll be deconstructing and then rebuilding the pizza bot.

Add a Bot Q&A

The Oracle Intelligent Bot Q&A framework enables you to add simple question and answer support to your bot. The Q&A feature allows the bot to answer general interest questions based on seeded questions and answers or refer to content like frequently asked questions (FAQs) or other knowledge-based documents.

You can start using the Q&A feature by enabling it from the user interface and loading the source files that hold the categorized question and answer pairs. You can tweak the Q&A modules and data sources to optimize accuracy. You can then test the Q&A and configure the dialog flow to support the Q&A.

Learn more about Q&A.

Link a Custom Component to Your Bot

The Oracle Intelligent Bots platform provides a powerful feature to build your own custom components. You can use custom components to add business logic to a chatbot conversation, query and update back-end systems, or add functionality not included as part of the built-in components. For example, if you need to display weather data in reply to a customer query, you can create a custom component to call a third-party API to get the weather details for a given city and country.

Custom components execute the business logic within a Node container. The components are invoked from the Dialog Engine during a conversation between a chatbot and a user. When invoked, the component sends a request through a REST call in a JSON file to a back-end system. The back-end system returns the response through a REST call, also in a JSON file. The chatbot application then displays the information to the user within a messaging channel, such as Facebook. Currently, you can write custom components in JavaScript and Node for Oracle Intelligent Bots.

What Is a Custom Component Service?

Custom components reside within a component service. A component service is a REST service that holds several custom components. You need to register a component service with the Oracle Intelligent Bots to use the custom components. It is important to note that the custom components do not reside on the same host as the Oracle Intelligent Bots. The components are hosted on different Cloud Services, such Oracle Mobile Cloud Service, Oracle Application Container Cloud Service, or other similar REST servers on the Internet.
A component service exposes two REST methods: GET and POST.

**GET Method**

The GET method is used to retrieve a list of custom components hosted by the service. Along with the component list, it provides information about each component such as the component name, input properties, and action strings returned by the component. This information is provided in a JSON format and is interpreted at design time by Oracle Intelligent Bots.

**POST Method**

The POST method is used to invoke a custom component by appending the component name to the REST URI. During a dialog flow, component properties are passed in a payload from the dialog flow in Oracle Intelligent Bots to the component.

**List of values returned by the POST method.**

- Invokes Custom Component
- Component Name Passed in URI
- Properties Passed in Payload
- Flow Context Passed in Payload

**A sample payload sent to a custom component using the POST method.**

The payload sent with each POST request to the component service contains information about the component properties as well as context information, such as the query text sent by the user. The query text is used to find the intent, which then determines the conversation flow that is executed.

**List of values returned by the GET method.**
Implement Custom Components

In Oracle Mobile Cloud Service, the Oracle Intelligent Bots component service is implemented as a custom API.

The following sequence explains the runtime process:

1. Oracle Intelligent Bots chatbot sends a request to the component service, a custom API in Oracle Mobile Cloud Service, with the name of a component to invoke.

2. A mobile back end that hosts the component service authenticates the API before it can access the component service. A mobile back end is a secured API container that the chatbot needs to authenticate to access the component service. It is similar to a webserver. It handles the communication between the API and the bot.

3. The custom API in Oracle Mobile Cloud Service defines the component service and is exposed as a REST service. The custom API uses the GET method to derive the list of custom components available in a component service. The URI is composed of the Oracle Mobile Cloud Service host, the mobile custom path followed by the name of the API followed by components.

Example of GET URI:
https://mcs-host-url:port/mobile/custom/api_name/components (for GET request)

The Oracle Intelligent Bots UI calls the GET endpoint at design time to discover the components parameters and its possible responses.

Example of POST URI:
https://mcs-host-url:port/mobile/custom/api_name/components/name_of_component (for POST requests)

4. The Node container in Oracle Mobile Cloud Service processes the JavaScript implementation code of the custom component.

If back-end service access is required, you can use Oracle Mobile Cloud Service connectors to access REST services, SOAP services, Fusion App, and Integration Cloud Service.
Understand the Architecture of Custom Components

The custom component architecture consists of the component service, Shell, Registry, SDK, and Message Model.

Component Service

A REST API service that hosts the custom components. Component services don’t reside within the bots but hosted on a separate Node container. The Node container can be part of Oracle Mobile Cloud Service or any other node container. You can implement the component service using any language that supports REST APIs. Oracle Mobile Cloud Service provides an SDK specifically to implement component services.

Shell

The Shell is a JavaScript object that routes the GET and POST requests. It produces a list of custom components in response to the GET call made by bots when you register a component service. The Shell invokes a component using the component’s name that’s appended to the POST call (POST uri/components/{ComponentName}). To respond to these requests, the Shell component references a file in the Registry component that maps the component names to their corresponding JavaScript implementation files. The Shell object is shipped with Oracle Intelligent Bots platform.

Registry

Since each component is saved as a JavaScript file, it becomes necessary for the components to be registered with the component service. The registry file contains name-value pairs in form of JSON object properties. Each attribute name represents a custom component, for example, Balance Retrieval can be the name of a custom component and invoked to retrieve the account balance. The attribute value is a file reference to the JavaScript file that contains the component’s implementation code. Each component is described by a name-value pair in which the ‘name’ is the name of the component like ‘Balance Retrieval’ and the value is a return function with a reference to the JavaScript module location relative to the registry.js file (./).

The Shell component uses the Registry object. Oracle provides the Shell file and you must not edit this file. The Registry file (registry.js) must be in the same folder as the Shell file (shell.js).
In this snippet, the three components, BalanceRetrieval, TrackSpending, and Payments are custom components, each of which maps to a separate JavaScript module. The `require` function includes these separate modules in the registry.js file.

```javascript
'use strict';

module.exports = {
  'BalanceRetrieval': require('./banking/balance_retrieval'),
  'Trackspending': require('./banking/track_spending'),
  'Payments': require('./banking/payments')
}
```

Because the Shell.js component assumes that it shares the same file location as the registry.js, the shell.js file uses the following import statement:

```javascript
var registry = require('./registry');
```

**SDK**

Oracle Mobile Cloud Service provides an SDK to implement the component services. Using a set of helper methods, the components access the context of a bot's message, which comprise of elements that describe the variable values, the language processing results, the extracted entities, and any input parameters defined for the component. The SDK also enables the components to return a response to the bot. To access the SDK and its methods, the invoke function uses an argument called conversation, which is automatically passed with each request along with the essential `done()` callback that signals the Shell when the component has completed its work.

While you can use the Shell and Registry components in any REST framework that produces a JSON object from the incoming request, you can only use the SDK's helper methods if you implement the Component Service in Oracle Mobile Cloud Service. To use the SDK and get ready-made versions of the Shell and Registry, you need the Bots SDK.

**MessageModel**

The MessageModel is a utility class that helps to create and validate a message structure representing a bot message. An instance of this class is instantiated along with the payload that represents the message so that the message can be parsed and validated. This utility is used by the Bots Custom Components Conversation SDK and downloaded as part of the SDK.

Version 1.1 of the Bots SDK lets you leverage the Conversation Message Model (the CMM), a framework that defines various platform-agnostic templates for the messages sent between the bot and its users. Not only does the CMM allow your bot to output messages as loops of cards that have actions configured for both the images and buttons that display within each of them, it also gives your bot other capabilities as well, such displaying context-specific messages and allowing users to share locations or upload audio, video, file, or image attachments. The Bots SDK documentation describes how you integrate the CMM into the code for your custom components, the methods for different types of message formats, and how you can upgrade your custom component service to use the CMM.
Shared APIs

Custom component APIs created for specific tasks, such as ‘BalanceRetrieval’, ‘accountType’, and so on. You create custom component APIs using the Create API functionality. The custom components are stored as JavaScript files in the same directory as the Shell, Registry, Message Model, and SDK components within the component service scaffolding.

REST Connector

An API used to connect to external or third-party REST services, such as Google Books API or Weather API. The REST connector API enables the mobile back-end to communicate with the external services allowing it to send and receive data. The REST Connector API wizard walks you through creating REST Connector APIs, from specifying a remote service and setting security policies to testing your endpoints. Learn more about REST Connector API.

Build a Custom Component

To build a custom component, you need to create a new API in Oracle Mobile Cloud Service and define the REST endpoints—the GET and the POST methods. Next, you need to implement the scaffold file to add a custom component logic and then associate the custom API with a back end and test the API. After testing the API, register the component service with the bot and configure the custom component to be invoked from a dialog flow.

Here are the broad level steps for creating and configuring a custom component:

- Create intent and utterances
- Create entities
- Create custom API
- Define REST endpoints
- Download scaffold
- Implement the custom component
- Associate custom API with back end and test
- Register component service in Oracle Intelligent Bots
- Configure the dialog flow
- Deploy chatbot on Facebook Messenger

Learn about Oracle Mobile Cloud Service.

Free Cloud Trial
Summary

Chatbots are the new sentinels at the forefront of automation and gold star customer experience in the fields of retail, healthcare, travel, finance, insurance, and many more industries. Powered by machine learning and natural language processing, Oracle Intelligent Bots provides an easy and robust platform for building intelligent chatbots. Building a bot requires minimal coding and comes with rigorous training capabilities backed by deep integration into the enterprise systems through Oracle Mobile Cloud Service.

By packaging the Oracle Intelligent Bots platform as part of Oracle Mobile Cloud Service, Oracle provides you with features such as API shaping, geo-location services, push notifications, support for mobile client platforms, analytics based on the user interaction, and, of course, all connectors and security for accessing real back-end systems that are the core of any business. Coupled with this, Oracle Mobile Cloud Service hosts the heavyweight AI processing required to train, learn, and understand natural language conversations between users and bots.

Additional Resources

- Build a Simple Bot with Oracle Mobile Cloud Service
- Create a Custom Component for Your Bot with Oracle Mobile Cloud Service
- Chatbots 101: Just The Facts
- Oracle Mobile Cloud, Enterprise Bots Client SDK for JavaScript
- Custom Component Development Quickstart