Side-by-side Comparison of Amazon Lex and Google DialogFlow

**Lex: DialogFlow:**

* All intents are available all the time and all exist at the root level
* Visual editor UI is difficult to work with – blocks sometimes reappear once they’ve been deleted.
* A condition block cannot have the same parent.
* Only one confirmation block per intent
* Documentation is unreliable, recently moved from v1 to v2 and v2 docs are not always accurate/up to date
* Will invoke lambda at each step when you turn it on at the intent level, must uncheck at each step of intent otherwise it will make a call after each turn and give an error.
* Builds take a long time and must be done before being able to test any new changes
* Cannot store objects in sessionAttributes, strings only
* Work gets duplicated (not sure if it’s because 2 people have consoles open)
* Intents live at the page level, meaning that you must explicitly add them to each turn of the conversation if you want the bot to listen for it.
* UI editor is much cleaner than Lex
* Route groups exist – actionable flows you can create that cascade down the flow they’re added to.
* Flows – modular sets of actions that can be added to any step in other flows.
* Less-defined block purposes; only have pages which can be purposed for anything

|  |  |  |
| --- | --- | --- |
|  | Google DialogFlow | Amazon Lex |
| Documentation | Smiling face outline with solid fill  Documentation has been very clear and concise. Everything that was documented has worked as designed. | Sad face outline with solid fill  Documentation for V2 is lacking compared to Googles – ex. Missing clear definitions for certain response properties |
| Cloud Function integration | Sad face outline with solid fill  Deploy take longer in Google Cloud function than AWS  DialogFlow requires HTTP connections for WebHook.  \*\* although a pain, this could be helpful if needing to connect to services in AWS. | Smiling face outline with solid fill  Lambda deploys are quick.  Lex can natively invoke Lambdas rather than requiring an HTTP gateway for the executions |
| Response formats | Smiling face outline with solid fill  Response format was much cleaner, everything is treated as a parameter with different data types. | Sad face outline with solid fill  Response format was much more convoluted with sessionAttributes and slots. They don’t co-mingle and they’re not interchangeable. |
| Variables/Properties | Smiling face outline with solid fill  DialogFlow allows for complex data types. Underlying data types of Number, String | Sad face outline with solid fill  Only allows for String data type for sessionAttributes whereas slots can be Number or Strings.  When dealing with numeric values, we had to cast strings to numbers in order to do operations but Lex will convert it back to a String which makes operations complicated (think comparisons x > 2) |
| User Friendliness | Smiling face outline with solid fill  Does a really good job of being up to date with what everyone is seeing while looking at the page. | Sad face outline with solid fill  Sometimes blocks are added which weren’t originally added.  Blocks sometimes are duplicated and are not seen by one user but seen by another.  Properties are sometimes not shown in the correct state to the user but when the page is refreshed (or another user looking at the page) will show up properly. |
| Build time | Smiling face outline with solid fill  Builds are not required for each change within the bot. There is learning that happens when you make changes but NLU only learns what’s been changed. | Sad face outline with solid fill  ENTIRE bot must be built when any change is made to anything within Lex ecosystem. This build time expands as the bot grows. |
| Route Organization | Smiling face outline with solid fill  Visual builder is much cleaner. We can create route groups that are modular and can be placed anywhere in a flow.  Flows allow us to call into other flows and allow us to do “routes” | Sad face outline with solid fill  Visual builder is very confusing.  Lines are not visually understandable after reaching a certain number of elements.  Some blocks (confirmation, closing remark) can only be used once in an intent  Some blocks (conditionals) cannot have multiple things leading to it. |
| Intents | Smiling face outline with solid fill  Intents live at the page level so must be specifically placed if you want the bot to listen for it.  Can have multiple intent “routes” per flow. | Sad face outline with solid fill  Intents live at the root level, meaning that if the conversation is not inside of a specific intent, then ALL intents are available to be listened for.  Within one intent, cannot have any other intent “routes.” |
| Environment Set up | Requires the DialogFlow API to be enabled. Create a DialogFlow Project and then can create an Agent. The Agent will need to be in the Global environment in order to get a phone number for testing | Requires IAM permissions for lex:\* as well as for lambda to ensure the creation is successful. Also, we had to create a role with permissions to upload to CloudWatch. After that, creating a bot was the only further step. |
| Integration with Cisco |  |  |
| Testing capabilities | Smiling face outline with solid fill  Comes built-in with testing in mind. When using the Test section can “save” a walk through and have it re-executed for testing.  Can also have environments and those environments can have tests run against them on a daily/deployment basis. | Sad face outline with solid fill  We have not found any testing capabilities. |
| Logging | Sad face outline with solid fill  Visual logging is nice; however, it would sometimes miss tracking something that was said. | Smiling face outline with solid fill  CloudWatch is nice and there is a lot you can do with it.  Can migrate logs from CloudWatch to RedShift for data lake style long term storage and searching. |
| Debugging | Smiling face outline with solid fill  Has the ability to track all parameters during a test session.  Has ability to track which flow and even page you are on at any point in the conversation.  Can step through each part of the decision tree while testing. | Sad face outline with solid fill  You can see slots and their values during a test session.  Can see what intent you are in.  Cannot see steps within the intent. Cannot play any part of the decision. Cannot see sessionAttributes. |
| Analytics | Smiling face outline with solid fill  Great analytics, lots of stats available; abandon rates, percentages of path choices, unhandled messages, and percentage intent handling.  All analytics are easily exportable to DB service for querying. | We have not seen any analytics available |
|  |  |  |