Regexp entities

Some entities need to match patterns rather than specific terms. For example, national identification numbers, IDs, license plates, and so on. With *regexp entities*, you can provide regular expressions (https://github.com/google/re2/wiki/Syntax) for matching.

Where to find this data

When building an agent, it is most common to use the Dialogflow Console (visit documentation (/dialogflow/docs/console), open console (https://dialogflow.cloud.google.com)). The instructions below focus on using the console. To access entity data:

1. Go to the Dialogflow Console (https://dialogflow.cloud.google.com).
2. Select an agent.
3. Select **Entities** in the left sidebar menu.

If you are building an agent using the API instead of the console, see the EntityTypes reference (/dialogflow/docs/reference/common-types#entitytypes). The API field names are similar to the console field names. The instructions below highlight any important differences between the console and the API.

Compound regular expressions

Each regexp entity corresponds to a single pattern, but you can provide multiple regular expressions if they all represent variations of a single pattern. During agent training, all regular expressions of a single entity are combined with the alternation operator (`|`) to form one *compound regular expression*.

For example, if you provide the following regular expressions for a phone number:

- `^[2-9]\d{2}-\d{3}-\d{4}$`
- `^(1?(-?\d{3})-?)?(\d{3})(-?\d{4})$`

The compound regular expression becomes:

- `^[2-9]\d{2}-\d{3}-\d{4}$|^(1?(-?\d{3})-?)?(\d{3})(-?\d{4})$`

The ordering of regular expressions matters. Each of the regular expressions in the compound regular expression are processed in order. Searching stops once a valid match is found. For example, for an end user expression of "Seattle":

- `Sea|Seattle` matches "Sea"
- `Seattle|Sea` matches "Seattle"

Special handling for speech recognition

If your agent uses speech recognition (also known as audio input, speech-to-text, or STT), your regular expressions will need special handling when matching letters and numbers. A spoken end-user utterance is first processed by the speech recognizer before entities are matched. When an utterance contains a series of letters or numbers, the recognizer may pad each character with spaces. In addition, the recognizer may interpret digits in word form. For example, an end-user utterance of "My ID is 123" may be recognized as any of the following:

- "My ID is 123"
- "My ID is 1 2 3"
- "My ID is one two three"

To accommodate three digit numbers, you could use the following regular expressions:

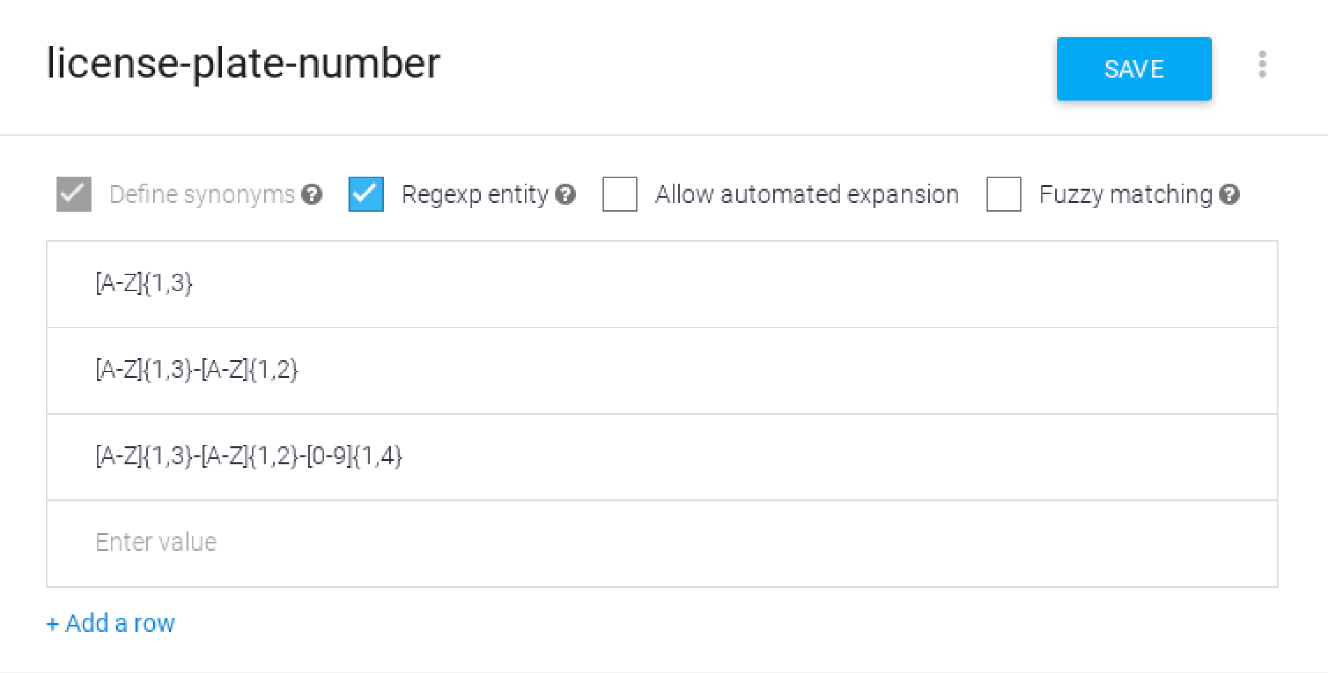
```
\d
```

```
|one|two|three|four|five|six|seven|eight|nine) (zero|one|two|three|four|five|six|sev
```

Create a regexp entity

To create a regexp entity:

1. Open an existing entity or create a new one.
2. Check **Regexp entity**.
3. Enter one or more regular expressions in the entries table.
4. Click **Save**.



license-plate-number

☑ Define synonyms ☑ Regexp entity □ Allow automated expansion □ Fuzzy matching

```
[A-Z]{1,3}
[A-Z]{1,3}-[A-Z]{1,2}
[A-Z]{1,3}-[A-Z]{1,2}-[0-9]{1,4}
```

If you are using the API to create or update entities, use `KIND_REGEXP` for the entity kind field.

Limitations

The following limitations apply:

- Fuzzy matching (/dialogflow/docs/entities-fuzzy) cannot be enabled for regexp entities. These features are mutually exclusive.
- Each agent can have a maximum of 50 regexp entities.

- The compound regular expression (#compound) for an entity has a maximum length of 1024 characters.

Previous
← Session entities (/dialogflow/docs/entities-session)

Next
Fuzzy matching (/dialogflow/docs/entities-fuzzy) →

Last updated 2020-08-20 UTC.