

ORACLE®

Oracle Digital Assistant

The Complete Training

Dynamic Entities



Image courtesy of pixabay.com

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

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.

Program agenda

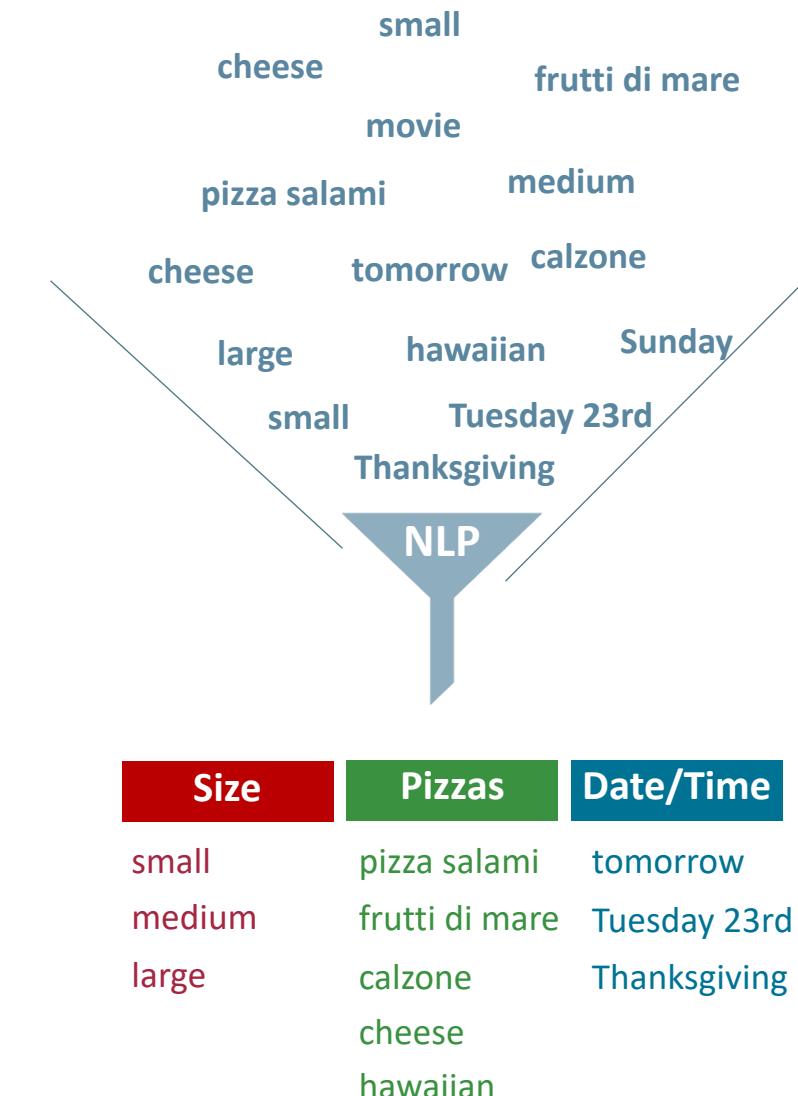
- 1 ➤ Feature overview
- 2 ➤ Oracle Digital Assistant REST APIs
- 3 ➤ Step-by-step example

Program agenda

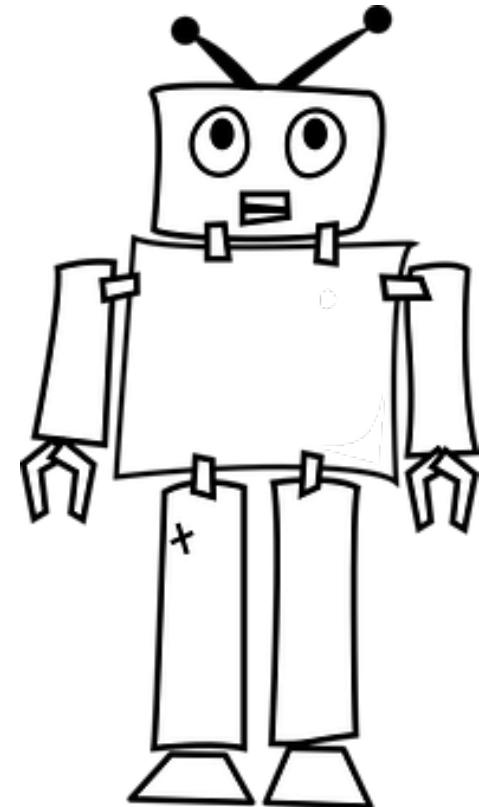
- 1 ➤ Feature overview
- 2 ➤ Oracle Digital Assistant REST APIs
- 3 ➤ Step-by-step example

Introduction to entities

- A variable piece of information of an intent
 - Helps add relevance to the intent
 - Associated with an intent
 - Extracted from user message through NLP
- Types
 - Regular entities
 - System entities, custom entities
 - Composite bag entities
 - Business domain objects
 - Dynamic entities
 - Data created, modified and deleted through API



Dynamic entities are value-list entities for which you can add, **modify**, and delete **data at run time**



Dynamic entity use cases

- Entities that change values over time
 - Low-frequency changes (once per hour, day, week, month)
 - Shared data, not user session based
- Examples
 - Product Names
 - Events
 - Buildings, departments, train stations, etc.

Actors



Skill Designer

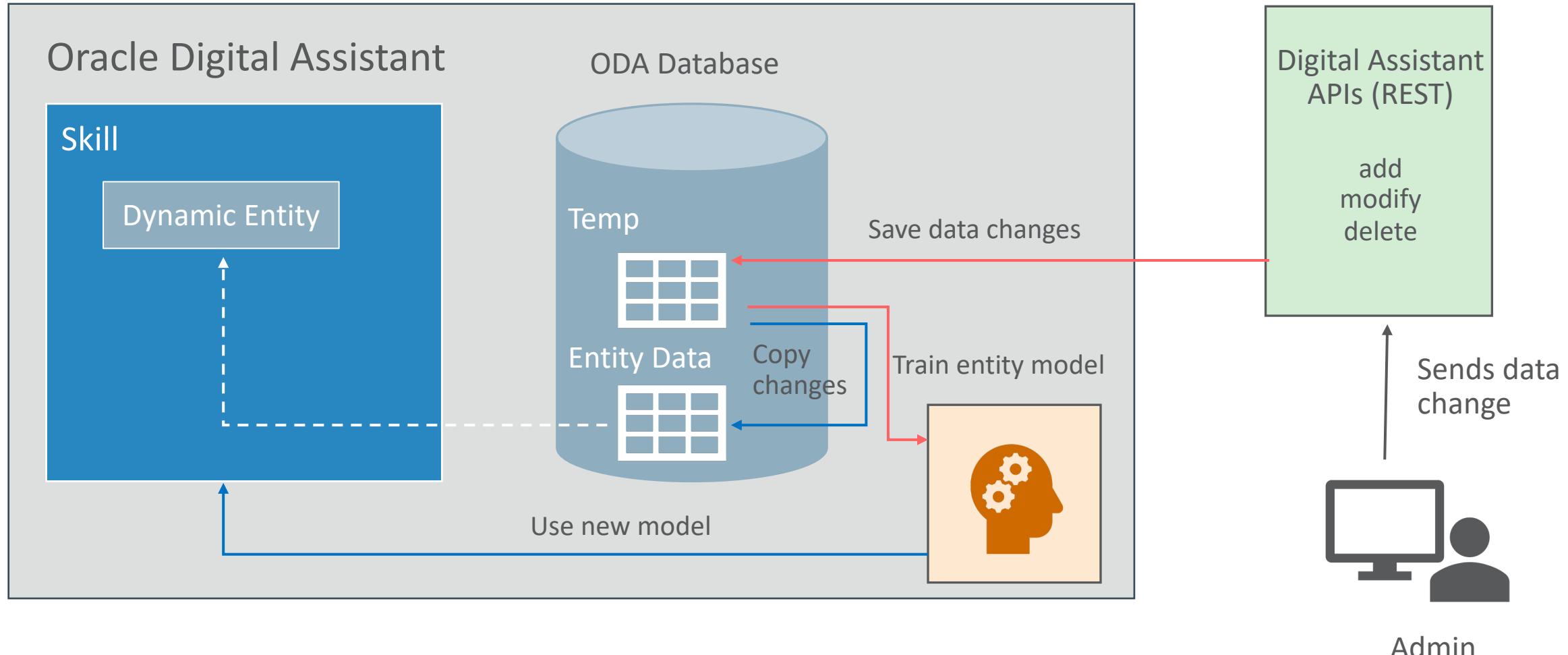
- Develops skill
- Creates dynamic entity
- Optionally, adds initial set of data to dynamic entity for testing



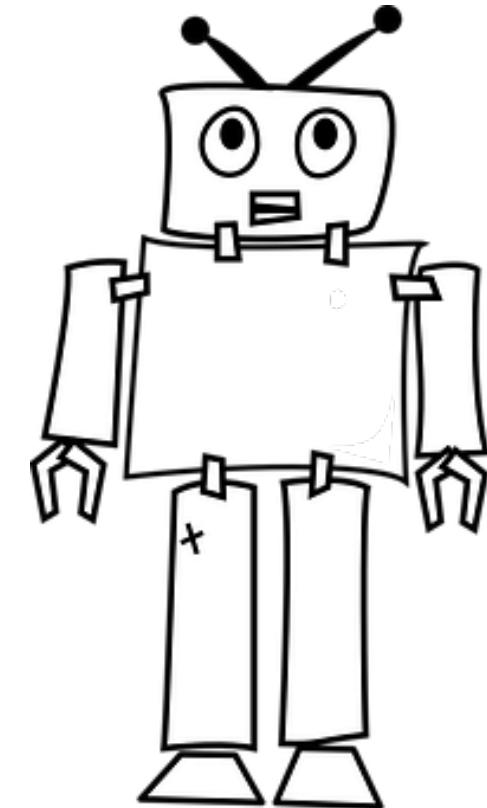
Service Administrator

- Creates script(s) to populate and maintain dynamic entities
 - Add data, modify data, delete data
 - Integrates backend services
- Looks for automating updates
 - E.g. upon change in database table
 - E.g. time of the day

Architecture



Dynamic entity training uses elastic search as the model.



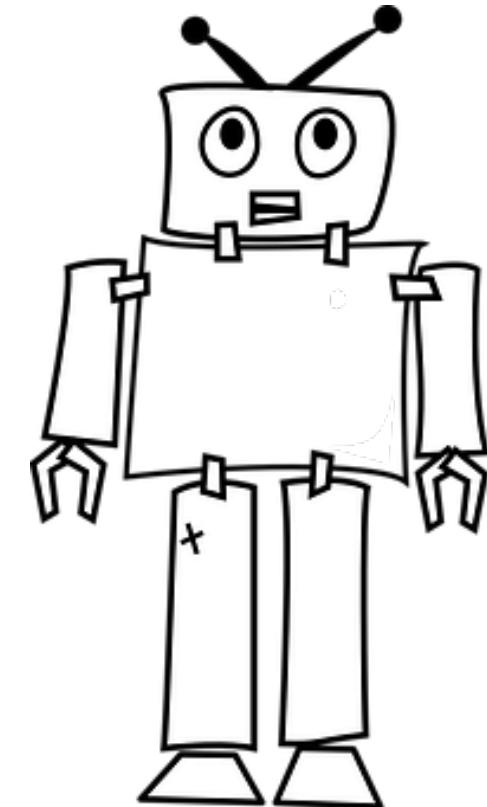
Program agenda

- 1 ➤ Feature overview
- 2 ➤ Oracle Digital Assistant REST APIs
- 3 ➤ Step-by-step example

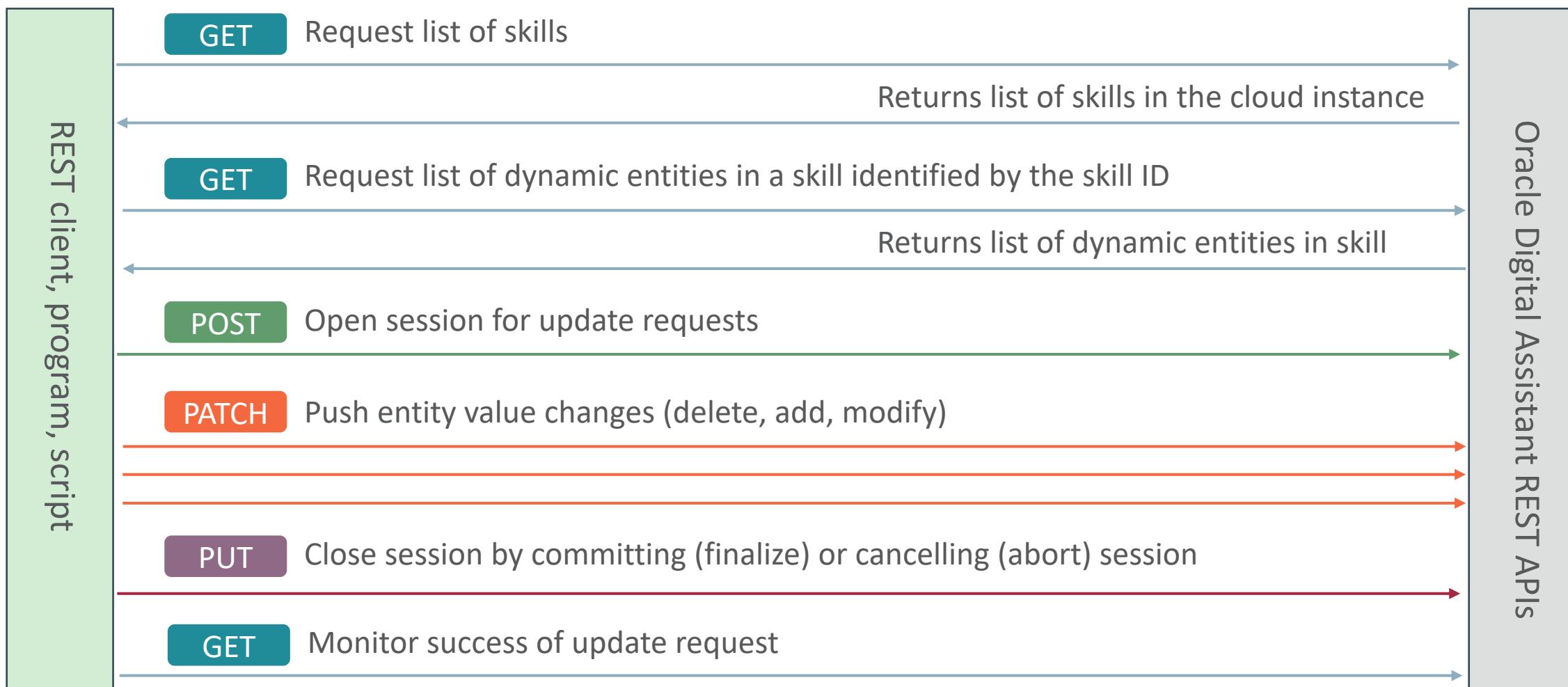
About Oracle Digital Assistant REST APIs

- APIs for managing an Oracle Digital Assistant instance
 - List digital assistants and skills
 - Export skill insights data
 - Export digital assistant insights data
 - Export skill session conversation log(s)
 - Manage values of dynamic entities
- POST, PUT, and PATCH requests may not require a payload
- Query parameters can be used to sort and filter returned lists
- Requests can be issued from any REST client

Oracle Digital Assistant REST API calls require an authorization token to be sent with the request.



Dynamic entity REST API workflow sequence



Dynamic entity APIs

GET

/api/v1/bots/{botId}/dynamicEntities

Returns the list of dynamic entities for the given skill for you to find a dynamic entity's ID

POST

/api/v1/bots/{botId}/dynamicEntities/{entityId}/pushRequests?copy=true | false

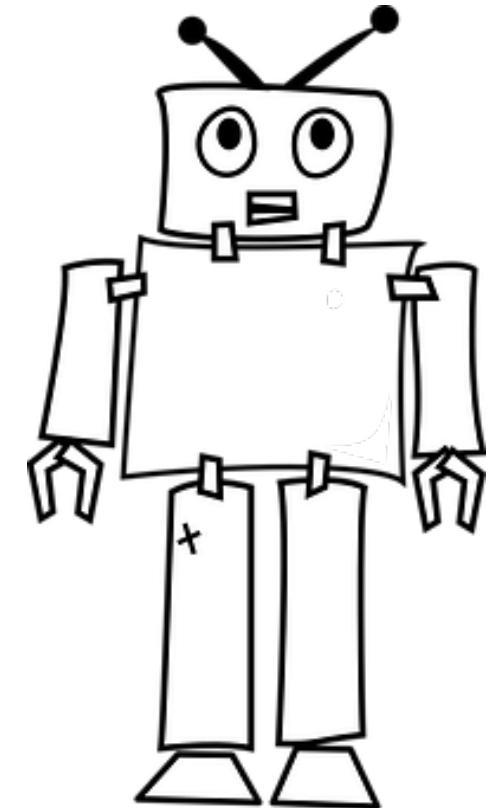
Creates a push request. After you send this request, you can push the entity values to delete, add, and modify.



... /pushRequests?copy=true | false

Set copy = true for requests that should preserve existing dynamic entity values. Use copy = false (or omit the copy parameter) for PATCH requests that replace all existing values with new values.

Only one push request can be active
for a specific dynamic entity at a time



Dynamic entity APIs

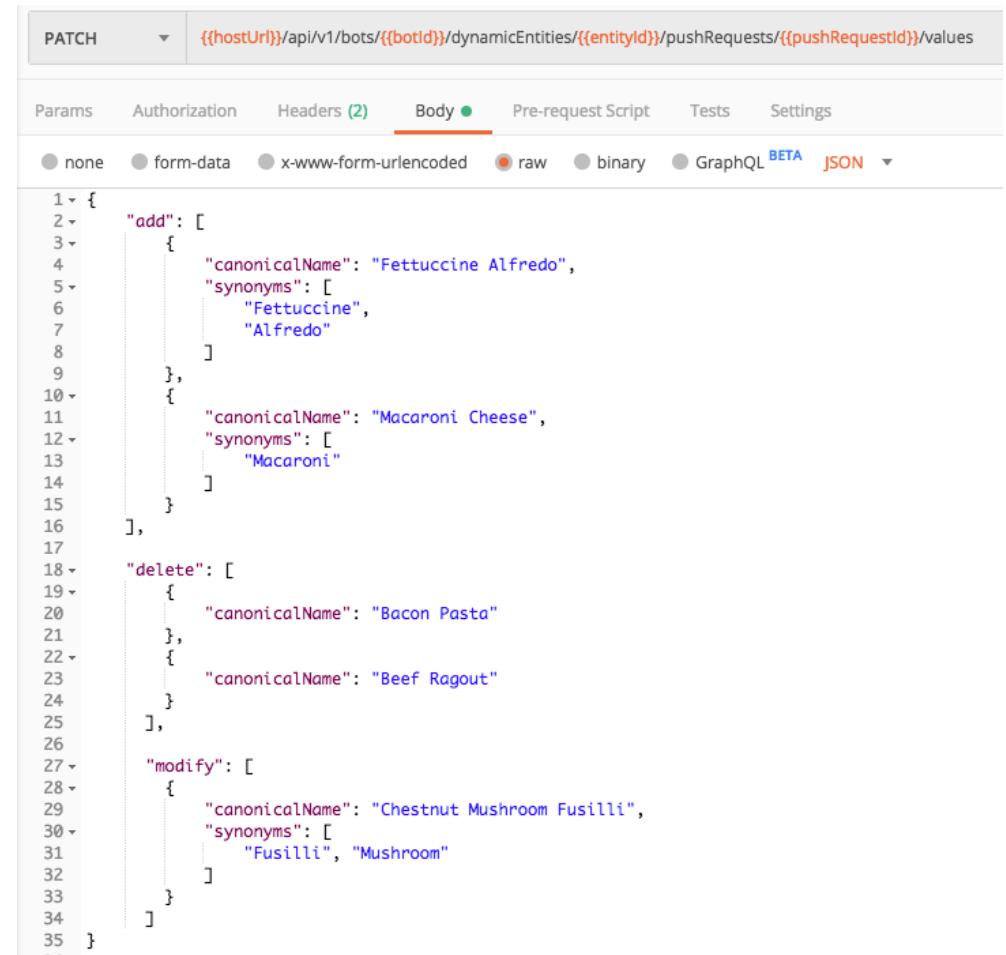
PATCH

/api/v1/bots/{botId}/dynamicEntities/{entityId}/pushRequests/{pushRequestId}/values

Provides an 'INPROCESS' push request with the entity values to add, delete, and modify. You use this operation to push data to an 'INPROCESS' *push request* until it is finalized or aborted. Data can be pushed in multiple requests.

Push data request payload example

```
{  
  "delete": [  
    { "canonicalName": "Bacon Pasta"}, { "canonicalName": "Beef Ragout"}],  
  
  "add": [  
    { "canonicalName": "Fettuccine Alfredo",  
      "synonyms": ["Fettuccine", "Alfredo"]  
    },  
    ... ],  
  
  "modify": [  
    { "canonicalName": "Chestnut Mushroom Fusilli",  
      "synonyms": ["Fusilli", "Mushroom"]  
    },  
    ...  
  ]  
}
```



The screenshot shows a Postman API request configuration. The method is set to PATCH, and the URL is {{hostUrl}}/api/v1/bots/{{botId}}/dynamicEntities/{{entityId}}/pushRequests/{{pushRequestId}}/values. The 'Body' tab is selected, showing a JSON payload. The payload is a complex object with three main sections: 'add', 'delete', and 'modify'. The 'add' section contains two entities: 'Fettuccine Alfredo' with synonyms 'Fettuccine' and 'Alfredo', and 'Macaroni Cheese' with synonym 'Macaroni'. The 'delete' section contains two entities: 'Bacon Pasta' and 'Beef Ragout'. The 'modify' section contains one entity: 'Chestnut Mushroom Fusilli' with synonyms 'Fusilli' and 'Mushroom'. The JSON is numbered from 1 to 35.

```
1  {  
2    "add": [  
3      {  
4        "canonicalName": "Fettuccine Alfredo",  
5        "synonyms": [  
6          "Fettuccine",  
7          "Alfredo"  
8        ]  
9      },  
10     {  
11       "canonicalName": "Macaroni Cheese",  
12       "synonyms": [  
13         "Macaroni"  
14       ]  
15     }  
16   ],  
17  
18   "delete": [  
19     {  
20       "canonicalName": "Bacon Pasta"  
21     },  
22     {  
23       "canonicalName": "Beef Ragout"  
24     }  
25   ],  
26  
27   "modify": [  
28     {  
29       "canonicalName": "Chestnut Mushroom Fusilli",  
30       "synonyms": [  
31         "Fusilli",  
32         "Mushroom"  
33       ]  
34     }  
35   ]  
36 }
```



... /pushRequests?copy=true

If you omit the copy query parameter or set it to false, then the modify and delete operations will not work. Copy=true must be set if you want to modify or delete values.



First delete, then update values.

Dynamic entities have a limit of 10,000 values per entity. The limit is checked for each PATCH request. Developers should therefore execute delete statements first.

Dynamic entity APIs

PUT

/api/v1/bots/{botId}/dynamicEntities/{entityId}/pushRequests/{pushRequestId}/{action}

Finalize or abort the push request. Set the {action} to DONE to close an open push request and then start training the entity. Set the {action} to ABORT to abort the push request. You can't abort a push request if it has completed.

Dynamic entity APIs

GET

/api/v1/bots/{botId}/dynamicEntities/{entityId}/pushRequests

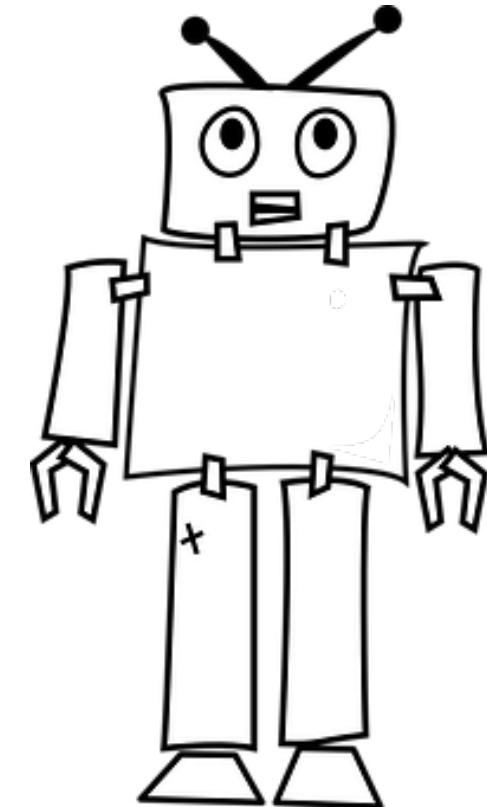
Returns list of push requests for a specific entity, which includes the ID and the status of each request. You can use query parameters to filter and sort the returned list.

GET

/api/v1/bots/{botId}/dynamicEntities/{entityId}/pushRequests/{pushRequestId}

Returns information for a specific push request. Shows status as INPROGRESS, TRAINING, COMPLETED, or ABORTED.

Dynamic entity data can be changed
for skills in publish and draft state.



Program agenda

- 1 ➔ Feature overview
- 2 ➔ Oracle Digital Assistant REST APIs
- 3 ➔ Step-by-step example

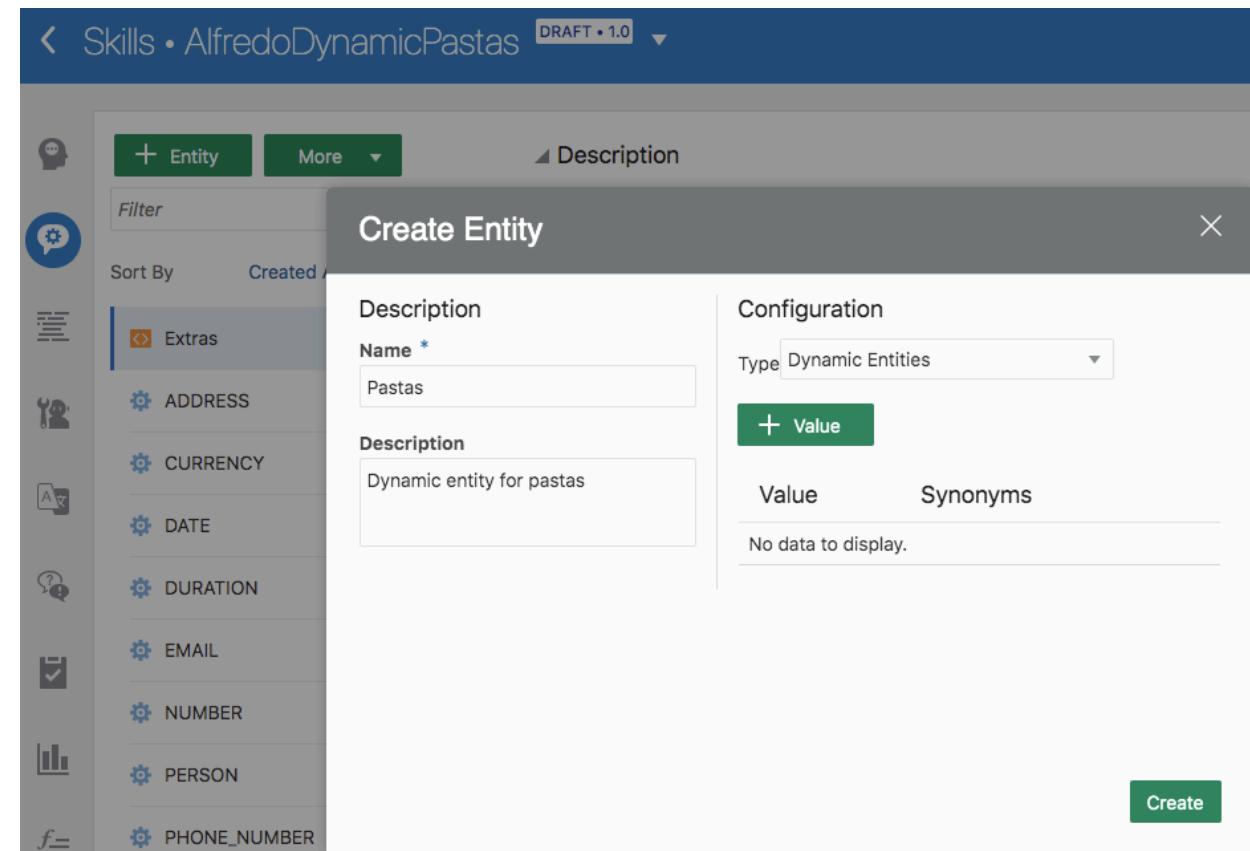
Use case

- Build a new dynamic entity for pastas
- Query list of dynamic entities
- Create push request
- Push entity data
- Finalize push request



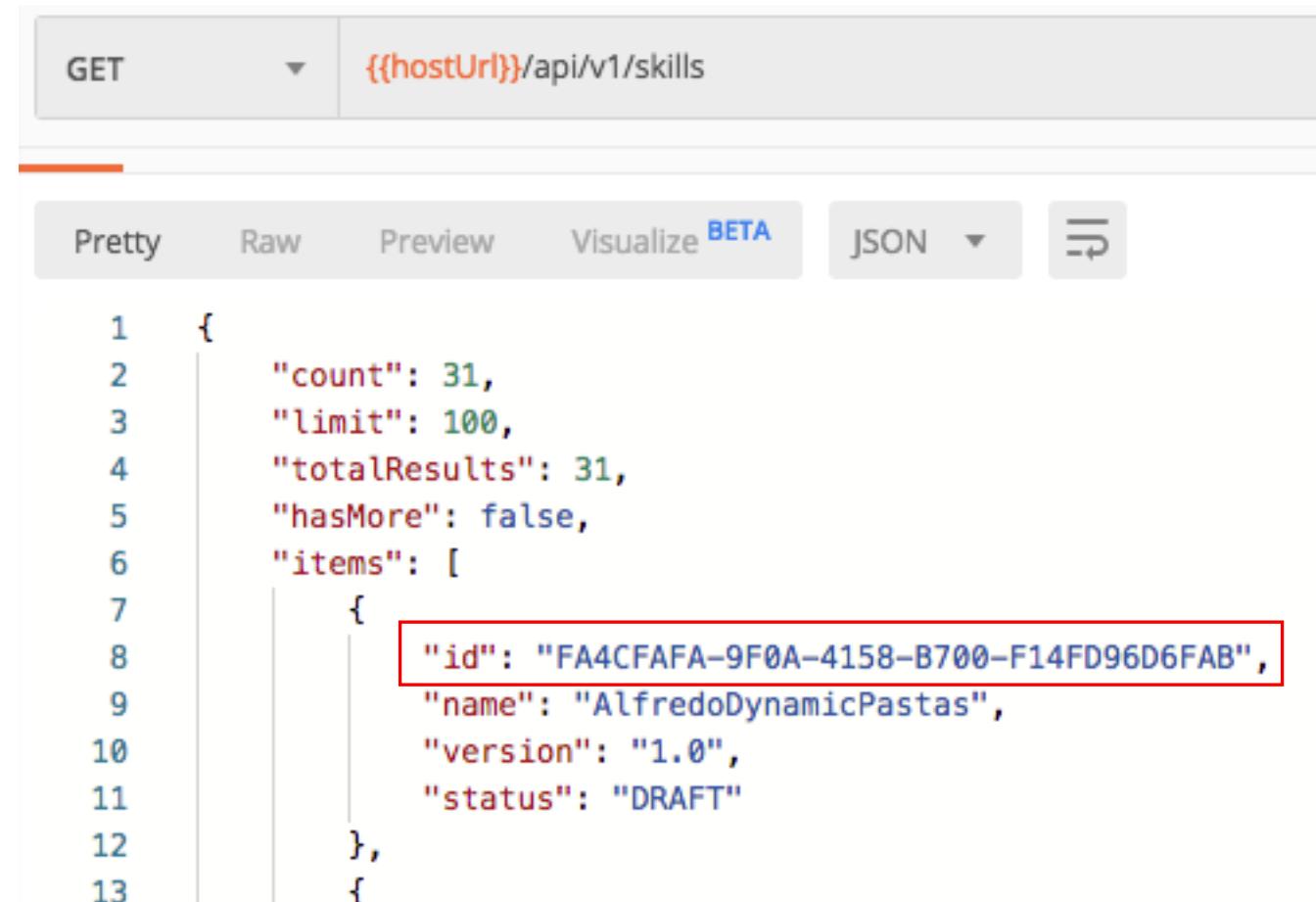
Building dynamic entities

- Create new entity
- Set entity "Type" to **Dynamic Entity**
- Optionally, define an initial list of values
 - E.g. use as mockup data



Query skill id using Oracle Digital Assistant REST APIs

- Get access token
- Determine skill ID
 - /api/v1/skills



The screenshot shows a REST API tool interface. At the top, there is a 'GET' button and a URL field containing `{{hostUrl}}/api/v1/skills`. Below the URL are several tabs: 'Pretty' (selected), 'Raw', 'Preview', 'Visualize BETA', 'JSON' (with a dropdown arrow), and a copy icon. The main area displays a JSON response with line numbers on the left. A specific skill entry is highlighted with a red box:

```
1  {
2      "count": 31,
3      "limit": 100,
4      "totalResults": 31,
5      "hasMore": false,
6      "items": [
7          {
8              "id": "FA4CFAFA-9F0A-4158-B700-F14FD96D6FAB",
9                  "name": "AlfredoDynamicPastas",
10                 "version": "1.0",
11                 "status": "DRAFT"
12             },
13         {
```

Query dynamic entities for a skill

- Query dynamic entities
 - Use skill ID as argument
 - /api/v1/bots/{botId}/dynamicEntities/
- Take note of dynamic entity ID

The screenshot shows a POSTMAN interface with the following details:

- Method:** GET
- URL:** {{hostUrl}}/api/v1/bots/{{botId}}/dynamicEntities/
- Headers:** (8)
 - Authorization: Bearer {{accessToken}}
 - Key: Value
- Temporary Headers:** (7)
- Body:** (Pretty, Raw, Preview, Visualize BETA, JSON, `={}`)
- Response Body (Pretty Print):**

```
1 {
2   "count": 1,
3   "limit": 100,
4   "totalResults": 1,
5   "hasMore": false,
6   "items": [
7     {
8       "id": "6A813AE3-B3BF-4950-AD05-5618F98C515C",
9       "name": "Pastas",
10      "type": "DYNAMICVALUES"
11    }
12  ]
```

Create push request for entity

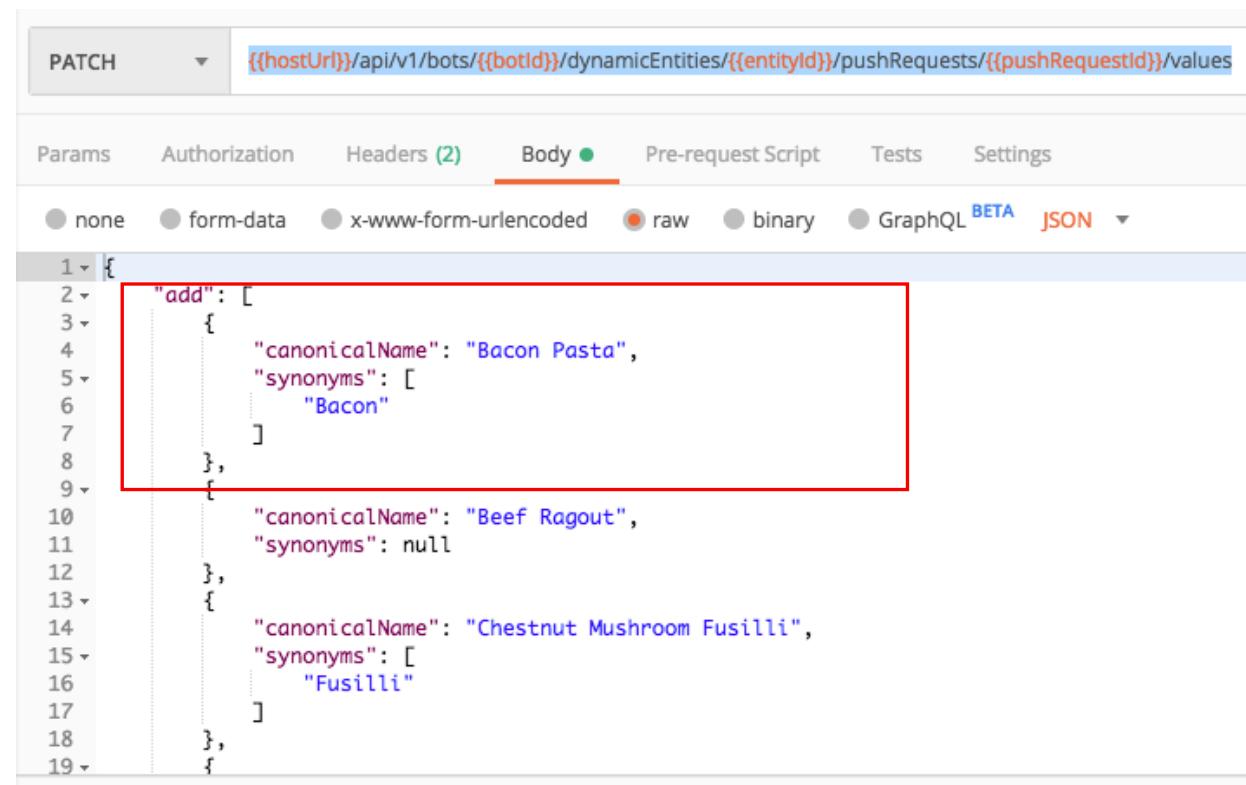
- Push request creates a 'session'
 - `/api/v1/bots/{botId}/dynamicEntities/{entityId}/pushRequests`
 - Entity values can be added, deleted, modified in multiple PATCH requests
 - Model will be trained at end of push request
 - Push request ID needs to be sent with data manipulation requests

The screenshot shows the Postman application interface for a POST request. The URL is `{{hostUrl}}/api/v1/bots/{botId}/dynamicEntities/{entityId}/pushRequests`. The 'Headers' tab is selected, showing an 'Authorization' header with the value 'Bearer {{accessToken}}'. The 'Body' tab is selected, displaying a JSON response with a red box highlighting the 'id' field.

```
1 {
2   "createdOn": "2019-10-09T09:32:27.435Z",
3   "updatedOn": "2019-10-09T09:32:27.435Z",
4   "links": [
5     {
6       "rel": "self",
7       "href": "https://idcs-oda-ceb728a9bc3541b5a3bafc75e11d4ef3-s0.data.digitalassist"
8     },
9     {
10      "rel": "canonical",
11      "href": "https://idcs-oda-ceb728a9bc3541b5a3bafc75e11d4ef3-s0.data.digitalassist"
12    }
13  ],
14  "id": "4D15B527-F0D1-4164-8DC7-CE508D35EE89",
15  "status": "INPROGRESS",
16  "statusMessage": "New push request is created, by user action."
17 }
```

Push data to dynamic entity

- /api/v1/bots/{botId}/dynamicEntities/{entityId}/pushRequests/{pushRequestId}/values
- Use pushRequestId when pushing data to update and change entities
- Data can be pushed in multiple PATCH request calls



The screenshot shows a POSTMAN interface with a PATCH request. The URL is `({hostUrl})/api/v1/bots/{botId}/dynamicEntities/{entityId}/pushRequests/{pushRequestId}/values`. The 'Body' tab is selected, showing a JSON payload with three items under the 'add' key. The second item is highlighted with a red box.

```
[{"add": [{"canonicalName": "Bacon Pasta", "synonyms": ["Bacon"]}, {"canonicalName": "Beef Ragout", "synonyms": null}, {"canonicalName": "Chestnut Mushroom Fusilli", "synonyms": ["Fusilli"]}]}]
```

Finalize push request

- /api/v1/bots/{botId}/dynamicEntities/{entityId}/pushRequests/{pushRequestId}/done
- Push request needs to be finalized
 - Writes data to dynamic entity
 - Trains NLP model
 - Updates entity table upon success
- Nothing gets changed if you pass "abort" instead of "done"

The screenshot shows a REST API tool interface with the following details:

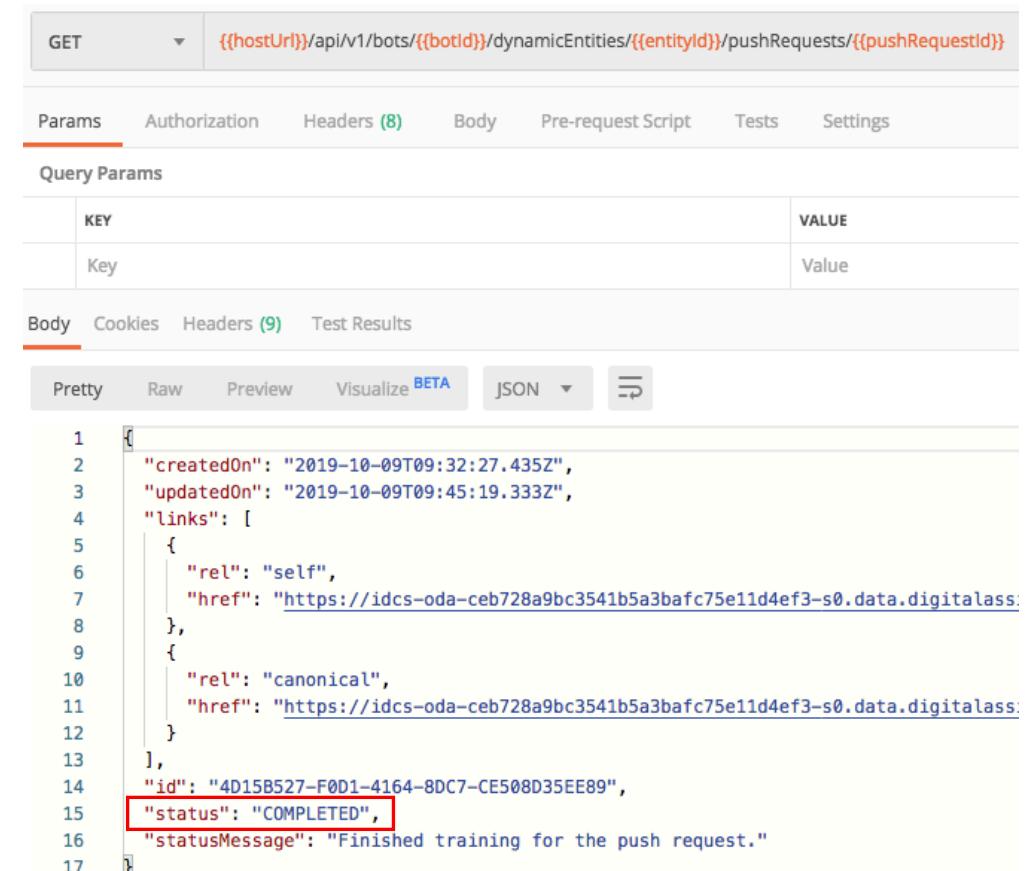
- Method:** PUT
- URL:** {{hostUrl}}/api/v1/bots/{{botId}}/dynamicEntities/{{entityId}}/pushRequests/{{pushRequestId}}/done
- Params Tab:** Active. Shows a table for Query Params with one row: Key (empty) and Value (empty).
- Body Tab:** Active. Shows a JSON editor with the following content:

```
1 {  
2   "createdOn": "2019-10-09T09:32:27.435Z",  
3   "updatedOn": "2019-10-09T09:45:19.094Z",  
4   "id": "4D15B527-F0D1-4164-8DC7-CE508D35EE89",  
5   "status": "TRAINING",  
6   "statusMessage": "Request Pushed into training, on user request"  
7 }
```

The field "status": "TRAINING" is highlighted with a red box.
- Headers Tab:** Shows 9 headers.
- Tests Tab:** Not visible.
- Settings Tab:** Not visible.

Check Status

- {hostUrl}/api/v1/bots/{botId}/dynamicEntities/{entityId}/pushRequests/{pushRequestId}
- Status "COMPLETED" indicates that entity data has been changed



The screenshot shows a REST API tool interface with the following details:

- Method:** GET
- URL:** {{hostUrl}}/api/v1/bots/{{botId}}/dynamicEntities/{{entityId}}/pushRequests/{{pushRequestId}}
- Params:** None
- Headers:** (8)
- Body:** None
- Pre-request Script:** None
- Tests:** None
- Settings:** None

The response body is displayed in JSON format:

```
1  {
2      "createdOn": "2019-10-09T09:32:27.435Z",
3      "updatedOn": "2019-10-09T09:45:19.333Z",
4      "links": [
5          {
6              "rel": "self",
7              "href": "https://idcs-oda-ceb728a9bc3541b5a3bafc75e11d4ef3-s0.data.digitalasset.com"
8          },
9          {
10             "rel": "canonical",
11             "href": "https://idcs-oda-ceb728a9bc3541b5a3bafc75e11d4ef3-s0.data.digitalasset.com"
12         }
13     ],
14     "id": "4D15B527-F0D1-4164-8DC7-CE508D35EE89",
15     "status": "COMPLETED",
16     "statusMessage": "Finished training for the push request."
17 }
```

Updated data also shows at design time

Skills • AlfredoDynamicPastas DRAFT • 1.0 ▾

+ Entity More ▾

Filter

Sort By Created Ascending

Pastas Extras ADDRESS CURRENCY DATE DURATION EMAIL NUMBER PERSON PHONE_NUMBER SET TIME

Description

Name *

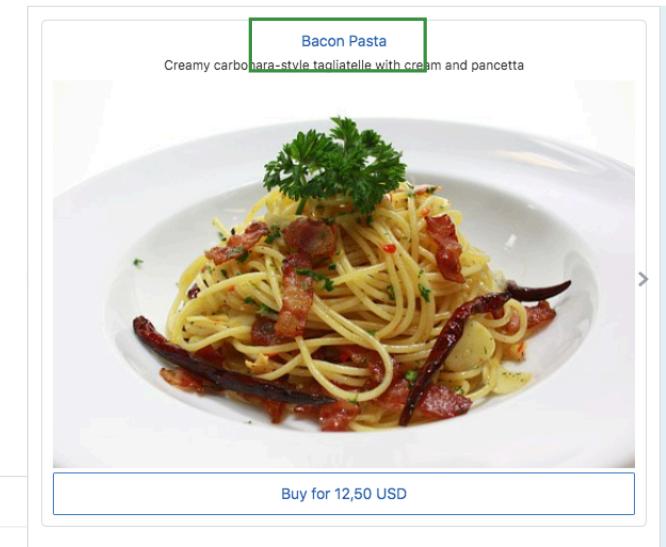
Description

Configuration

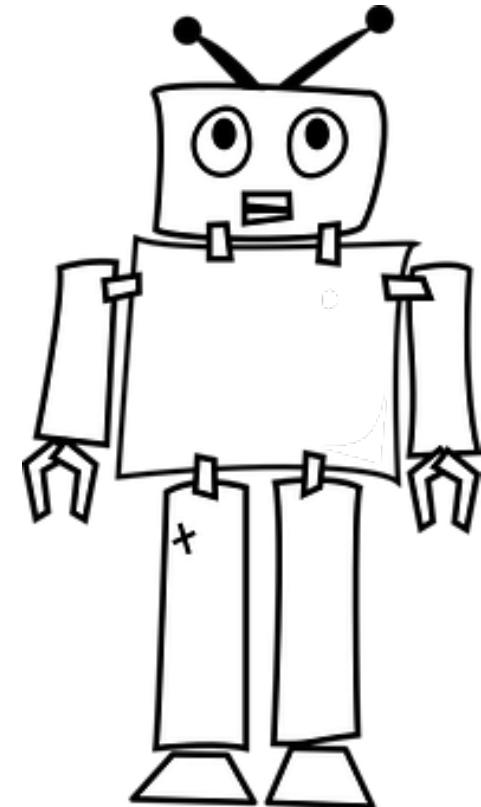
Type * Dynamic Entities

+ Value

Value	Synonyms
Bacon Pasta	Bacon
Beef Ragout	
Chestnut Mushroom Fusilli	Fusilli
Fettuccine Alfredo	Fettuccine, Alfredo
Macaroni Cheese	Macaroni

 Bacon Pasta
Creamy carbonara-style tagliatelle with cream and pancetta
Buy for 12,50 USD

If pushing data changes at design time,
make sure you **reset the conversation
tester** before testing the data changes



Integrated Cloud Applications & Platform Services

ORACLE®