Alexa, talk to ERPNext!

Motivation

Voice assistants are becoming integral part of business workflows. In this talk we will explore how Alexa and ERPNext can be integrated with each other.

Key components includes:

1. Voice Assistant: Alexa
2. Glue: AWS Lambda
3. Custom ERPNext API: Built using Custom App

Why integrate voice with ERPNext?

1. Quick query {for High level management}
2. Natural Response {vs Review of multiple reports}
3. Implement Domain Specific Concepts {e.g. Red flags for inventory}

Some of the queries that get frequently asked include:

1. Sales: What is sales during X? {Daily, Weekly, Monthly, Quarterly}
2. Inventory: What are some red flags?
3. CRM: How many leads did we generate in last X period? {Daily, Weekly, Monthly, Quarterly}

We will approach implementing this from bottom up i.e. **from Custom ERPNext API to ERPNext**

API Layer in ERPNext

Create custom app named `alex_integration` using following

```
cd /home/frappe/frappe-bench
bench new-app alexa_integration
cd apps/alexa_integration/alexa_integration
touch api.py
```

Next we will create a white-listed API as follows

```
import frappe

def sales_for(period):
    from datetime import datetime, timedelta

    if period == "daily":
        start = str(datetime.now().date()-timedelta(days=1))
    elif period == "weekly":
        start = str(datetime.now().date()-timedelta(days=7))
    elif period == "monthly":
        start = str(datetime.now().date()-timedelta(days=30))
    else:
        start = str(datetime.now().date()-timedelta(days=90))

    totals = list(frappe.db.get_values("Sales Order",
       filters={
        "status": ("in", ["Completed", "To Deliver and Bill", "
        "transaction_date": ['>=', start],
        }, fieldname=["Total"]))
    return sum([item[0] for item in totals])
```

```
@frappe.whitelist(allow_guest=True)
def sales(period):
    return sales_for(period)
```

Note: Don't forget decorator `@` <- I've wasted 2 hours for this silly mistake

Install the app using

```
bench --site <what_ever_site> install-app alexa_integration
```

Test your API endpoint using

```
curl -d “usr=SuperAdministrator&pwd=SuperSecurePasswd”\
http://localhost:<port_no>/api/method/alexa_integration\
.api.sales?period=weekly
```

Glue: AWS Lambda

TLDR; version of AWS Lambda: We will have **our service** call **your service**

More details Why AWS Lambda? 4 Benefits to Building Your Custom Alexa Skill with AWS Lambda

1. You Can Focus on Building Your Voice Experience
2. You Don’t Need to Provide an SSL Certificate
3. Lambda Streamlines Access Control and Security
4. Lambda Is Part of the AWS Free Tier

So what does it really look like?

```
def query_erpnext(final_url):
    """
    wrapper around existing API built
    using custom app alexa_integration
    """
    f = urllib.request.urlopen(final_url)
    data = json.loads(f.read().decode('utf-8'))
    return data['message']
```

```
def on_intent(intent_request, session):
    intent = intent_request['intent']
    intent_name = intent_request['intent']['name']

    # Dispatch to your skill's intent handlers
    if intent_name == "QuerySalesDaily":
        final_response = "Your sales today is 2000"
        return generate_response(final_response)
    elif intent_name == "QuerySalesWeekly":
        final_response = "You weekly sales is %s" % query_erpne
        return generate_response(final_response)
    elif intent_name == "QuerySalesMonthly":
        final_response = "You monthly sales is %s" % query_erpr
        return generate_response(final_response)
        ...
```

Voice Assistant: Alexa

Workflow for connecting Alexa to AWS Lambda

1. Create Custom Skill
    i. Make sure Language matches the one on your Device
2. Key Concepts
    i. Invocations
    ii. Intents
    iii. Utterances
    iv. End Point