APPMIXER

- It is a workflow engine together with a web user interface that allows end-users to create business processes.
- It is easy to drag and drop UI without writing a single line of code.

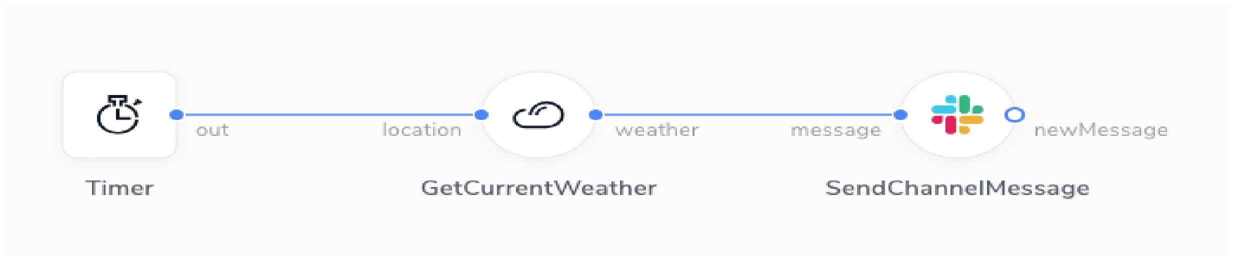
Features:

- Error Handling
- Message Passing
- Custom Components
- Quotes and Limits
- Authentication
- Monitoring.

Components:

- Components are the building blocks of Appmixer.
- Each component in a flow reacts on incoming messages, processes them and produces outgoing messages.
- User can wire components together to define complex behaviour and workflows.

Eg:



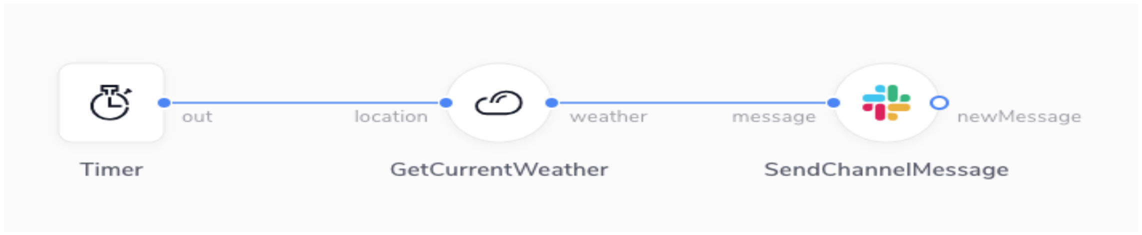
In the above example.There are three components in the flow.The first one is Timer,it generates the messages and send messages to the output port in the regular intervals.As soon as the message leaves it travels through the connected port as mentioned above in GetCurrentWeather component and it starts processing it requests current weather information from the api. Once the response is received from the api.The component is continues to the next Port as specified as SendChannelMessage. The process then repeats for all the other connected components until no message is generated on an output port or there is no other component connected.

Properties Of Components:

- Components don't know about each other.
- Components are black-boxes to the Appmixer engine.

Flow:

The components of each in the flow are interconnected with the source and their properties and the data will transfom through the input ports.



Basic Structure:

- Components in Appmixer are structured into "services", "modules" and "components" hierarchy.
- Each service can have multiple modules and each module can have multiple components.
- For example, "Google" service can have "gmail", "calendar" or "spreadsheets" modules and "gmail" module can have "SendEmail", "NewEmail" and other components.

Manifest :

The component manifest provides information about a component (such as name, icon, author, description and properties) in a JSON text file. The manifest file must be named **component.json**.

- **Name:**
  The name of your component. The name must have the following format:
  [vendor].[service].[module].[component]. Note that all the parts of the name must contain alphanumeric characters only.
  For example:
  { "name": "appmixer.twitter.statuses.CreateTweet" }.

- **Label:**
  The label of your component. If not label is specified, then last part of name will be used when component is dropped into Designer. If your component name is appmixer.twitter.statuses.CreateTweet then CreateTweet will be name of the component unless you specify label property

For example:

{ "label": "Create Tweet" }

- **Icons**:

  Icons are representing the component UI. It must be in the Data URI image format as described here:

  For Example:

  {

  "icon": "data:image/svg+xml;base64,PD94bWwgdmV..."

  }

- **Marker**:

  To give some more extra context..The marker must be in the Data URI image format.

- **Author**:

  It defines the author of the component name.

  For Example:

  {
  "author": "David Durman <david@client.io>"
  }

- **Description**:

  It describes about the component.

- **Properties:**

  The configuration properties of the component. Note that unlike properties specified on input ports (described later on in the documentation), these properties cannot be configured by the user to use data coming from the

components back in the chain of connected components. In other words, these properties can only use data that is known before the flow runs.
It includes as textarea,number,datetime,multiselect,toggle,colorpalette,select button group,expression,filepicker and soon.

Behaviour:

Components receive incoming messages, process them, and generate outgoing messages. The way messages are processed is called component behaviour. It defines what components do internally and how they react to inputs.

Dependencies:

Component NodeJS modules can use 3rd party libraries which are defined in the standard package.json file.

An example:

```
{
"name": "appmixer.twilio.sms.SendSMS",
"version": "1.0.0",
"private": true,
"main": "SendSMS.js",
"author": "David Durman <david@client.io>",
"dependencies": {
"twilio": "^2.11.0"
}
}
```

Authentication:

Components that require authentication from the user must implement an authentication module for the service/module they belong to. The authentication module must be named auth.js and must be stored under either the service or module directory (i.e. [vendor]/[service]/auth.js or [vendor/[service]/[module]/auth.js. Appmixer currently supports four types of authentication mechanisms that are common for today's APIs.

Module Structure:

The type of authentication mechanism. Any of apiKey, pwd, oauth (consumer id, consumer Secret) and oauth2(client id,client secret).

Installation:

First, unzip the appmixer.zip archive and change to the *appmixer/* directory.

Install and Start Appmixer:

```
docker-compose --project-name appmixer up
```

Stop Appmixer:

```
docker-compose --project-name appmixer stop
```

Using Appmixer SDK:

The Appmixer JavaScript SDK allows you to embed any of the UI widgets of Appmixer into your own web products. You can also take advantage of the SDK methods to communicate with the Appmixer engine REST API.

Using Appmixer API:

The Appmixer API allows you to access all the features that the UI works with via a REST API. If you followed the "Getting Started" section, you should have a user signed-up in Appmixer. In order to access the data of the user via the API, you need to have their access token. Use the following API call to get the token (don't forget to replace the "abc@example.com" and "abc321" with your own user's username and password).

Appmixer Deployment Models:

- Public Cloud.
- Private Cloud.
- VPC.
- Hybrid Cloud.

Appmixer Supporting Technologies:

- Rabbit MQ(Message Broker)
- Mongo DB(Database)
- Elastic Search(Search engines)
- Redis(Distributed Key Value)

By Using the get,post, patch and delete methods for storing and retrieving the data for all api purposes .

UI & Widgets:

- Flow Manager
- Designer
- Insights Logs
- Insights DashBoard
- Accounts
- Storage
- People Tasks
- Connectors
- Integrations
- Wizard.

Utilities:

- ### Email:

The SendEmail component uses our Mandrill API KEY by default. It is recommended to change that to your own Mandrill API KEY if you want to keep using Mandrill as the email service provider. You can do that from the Backoffice.

- ### Language:

The Language module uses by default Appmixer credentials.

- ### Tasks:

This component is more like a template, something you can use and built your own version that better fits into your platform.

- ### Tests:

Appmixer contains components that we use to test components. You can use them to test your own custom component. There is a *ProcessE2EResults* component which sends result from a test run (flow run) into a predefined list of email addresses.

- **Weather:**

  To get the api key of the current status of the weather in the location.