Agenda: Azure App Service – Logic Apps

- Introduction
- Creating a Simple Logic App
- Handling array of Items using for-each and condition
- Understanding Triggers and Actions
- Using Custom API App in Logic App

Introduction to Logic Apps

- Logic Apps are used to quickly build powerful solutions integrating various SaaS and enterprise applications.
- Logic apps use a **workflow engine** and a **visual designer** to design **business processes** graphically, and then connect them through connectors so that users can access data and required services. All this is achieved **without writing a single line of code.**
- The functionality of the out of the box connectors is based on the APIs that can **trigger** new instances of the workflow based on a specific event.
- Each step in the workflow is an action that accesses data or services through the connector.
- Best of all, Logic Apps can be combined with **built-in Managed APIs** to help solve even tricky integration scenarios with ease.

As mentioned, with logic apps, we can automate business processes.

Here are a couple examples:

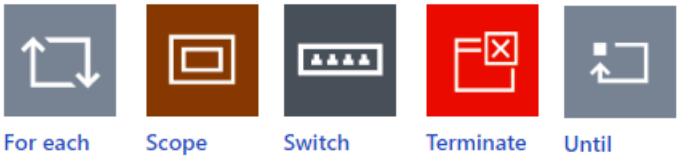
- Fetch phone numbers of New Contacts added into the CRM system like Sales Force or Dyanimcs, process them and automatically send them Welcome message by SMS.
- We can automatically fetch new records in a SQL DB and then send email alert to users.
- Automatically find negative posts on facebook wall and insert the same to database and delete.
- Monitor tweets for a specific subject, analyze the sentiment, and create alerts or tasks for items that need review.

Why Logic Apps?

- Logic Apps allow developers to design workflows that start from a **trigger** and then execute a series of **steps**. Each step invokes an API while securely taking care of authentication and best practices, like check pointing and durable execution.
- You don't have to worry about **Developing, hosting, scaling, managing, maintaining, and monitoring** your apps. Logic Apps handles these concerns for you.
- You pay only for what you use based on a **consumption pricing model**.
- In many cases, you won't have to write code. But if you must write some code, you can create code snippets with **Azure Functions** and run that code on-demand from logic apps.

Triggers and Actions

- **Triggers** - A trigger starts a new instance of a workflow based on a specific event, like the arrival of an e-mail or a change in your Azure Storage account or a Post on your Facebook wall.
- **Actions** - Each step after the trigger in a workflow is called an action. Each action typically maps to an operation on your managed or custom API apps. There are built-in actions for structuring and controlling the actions in your logic app's workflow. For example, you could insert a Condition to evaluate a condition and run different actions based on whether the condition is true or false. Other built-in actions are: For each, Scope, Switch, Terminate, and Until.



Managed Connectors

Managed connectors play an integral part when you create automated workflows with Azure Logic Apps. By using connectors in your logic apps, you expand the capabilities for your on-premises and cloud apps to perform tasks with the data that you create and already have.

Logic Apps offers ~200+ connectors, including:

- **Managed API connectors.** This includes Azure Blob Storage, Office 365, Dynamics, Power BI, OneDrive, Salesforce, and SharePoint Online.
- **On-premises connectors.** This includes SQL Server, SharePoint Server, Oracle DB, Twitter, Salesforce, Facebook, and file shares.
- **Integration account connectors.** Available when you create and pay for an integration account, these connectors transform and validate XML, encode, and decode flat files, and process business-to-business (B2B) messages with AS2, EDIFACT, and X12 protocols.
- **Enterprise connectors**. Provide access to enterprise systems such as SAP and IBM MQ for an additional cost.

Advantages:

- Logic Apps can be designed end-to-end in the browser using the design tool provided in Azure portal.
- Logic Apps make it an easy to connect disparate systems. Eg: Want to create a task in your CRM software that is based on the activity from your Facebook or Twitter accounts.
- Gallery of Templates are provided to rapidly create common solutions.
- Logic Apps is designed to work with API apps; you can easily create your own API app to use as a custom API. Build a new app just for you, or share and monetize in the marketplace.
- Logic Apps can easily leverage the power of BizTalk, Microsoft's industry leading integration solution to enable integration professionals to build the solutions they need.

Creating a Logic App

1. Select New, Web + Mobile, and select Logic App
2. Select Logic App → Settings → Logic App Designer (Development Tools)
3. Click + → Select Facebook "When there is a new post on my timeline"
4. Click Sign in to Facebook and provide Credentials of your Facebook account.
5. + New Step → Add an Action → search SMTP Send Email → Provide SMTP Credentials for sending email
6. Set To=<any email>, Subject = "You have a new post " [full name], Body = [Status Message]
7. For testing: Set status in Facebook and check above mentioned email inbox.

8. Create Azure SQL Server and Database and copy the connection string in Notepad
9. Go to SQL Server → Select Server → Settings → Firewall → Add client IP → Save
10. Connect to Server using SQL server Management Studio from local machine
11. Create Table as below

```
CREATE TABLE FacebookUpdates
(
        ID int NOT NULL Primary Key IDENTITY,
        StatusPosted Varchar(1000),
        PostedAt varchar(1000)
)
GO
```

12. In Logic App Designer, Add SQL Insert Row Trigger and provide the table name and set values for column **StatusPosted** = [Status Message], **PostedAt** = @utcnow()

Example2: Handling Array of Items using ForEach Step.

Logic App for Sending Emails to all rows of EmailAlerts table having AlertSent=False

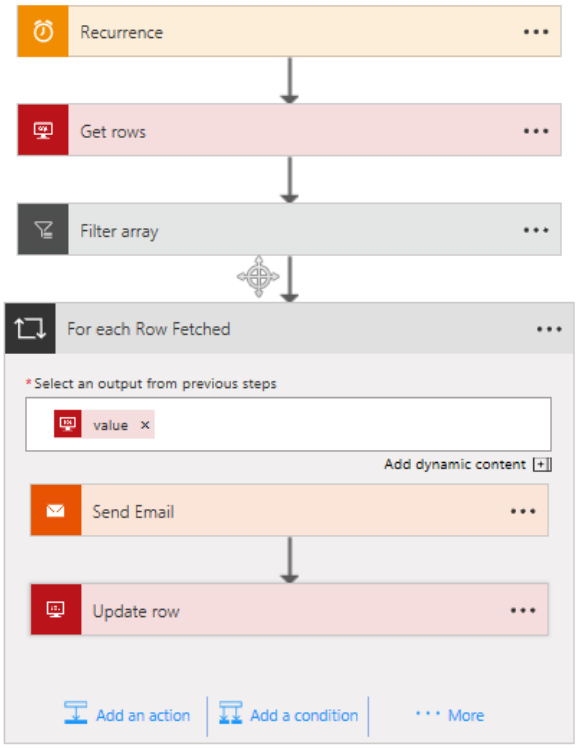
1. Create Table as below

```
CREATE TABLE EmailAlerts
(
        ID int NOT NULL PRIMARY KEY IDENTITY (1, 1),
        ToAddress varchar(50) NOT NULL,
        MailSubject varchar(50) NOT NULL,
        MailBody varchar(MAX) NOT NULL,
        EmailSent bit NOT NULL
)
GO
```

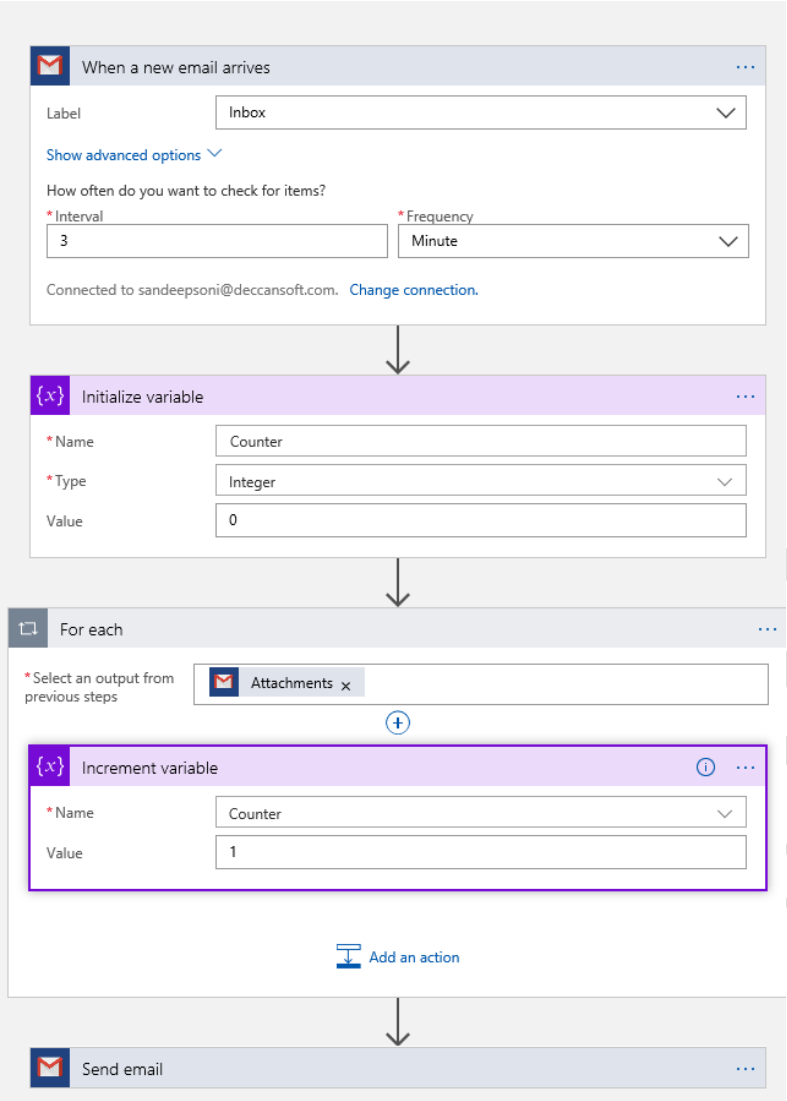
2. Insert some records into the above table.

```
INSERT INTO EmailAlerts (ToAddress, MailSubject, MailBody, EmailSent) VALUES
('decnsoft@hotmail.com','Sub1','This is message body - 1',0)
GO
INSERT INTO EmailAlerts (ToAddress, MailSubject, MailBody, EmailSent) VALUES
('decnsoft@hotmail.com','Sub2','This is message body - 2',0)
GO
INSERT INTO dbo.EmailAlerts (ToAddress, MailSubject, MailBody, EmailSent) VALUES
('decnsoft@hotmail.com','Sub3','This is message body - 3',0)
GO
```

3. Select New, Web + Mobile, and select Logic App
4. Select Logic App → Settings → Triggers and Actions → Logic App Designer
5. Click + → Select **Recurrence**, Frequency = Minute, Interval=5
6. Click + → Add Action → SQL Get rows,
    1. provide the SQL Connection Details
    2. Table name = "EmailAlerts",
    3. Click on Show Advanced Options, Filter Query=AlertSent eq false (Note: false should be in lowercase)
7. Click + New step → More → Add a for each
    1. Select an output from previous steps = Get rows, values
    2. Add an Action → Search Send Email → Provide details for sending email
        - From Address = <Email address whose settings are provide in email connector>
        - Set ToAddress, Subject and Body with Outputs from Get rows.
    3. Add an action search SQL – Update row
        - Select Existing Connection / Create a Connection
        - TableName = EmailAlerts
        - ToAddress, MailSubject, MailBody = values from Get row action
        - AlertSent = true
    4. Save and Run the Trigger.



Logic App With Variables and For Each



Check traffic on a schedule with Azure Logic Apps

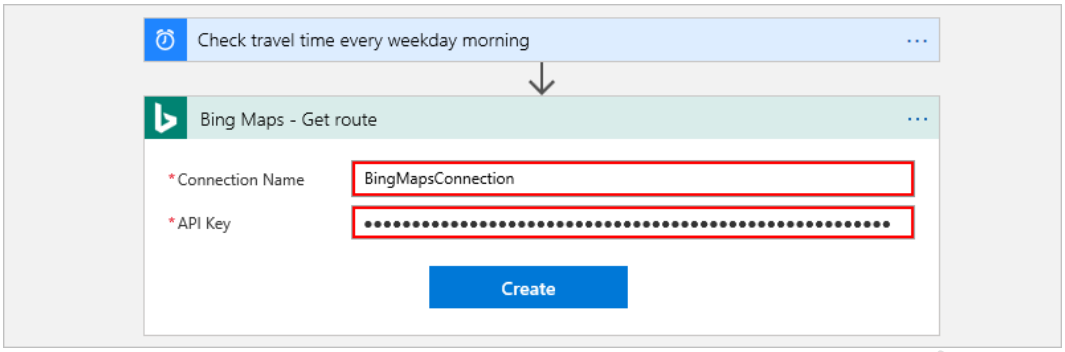
Step1: Creating a Bing Maps Key

1. Go to the Bing Maps Dev Center at https://www.bingmapsportal.com/.
   - If you have a Bing Maps account, sign in with the Microsoft account that you used to create the account or create a new one. For new accounts, follow the instructions in Creating a Bing Maps Account.
2. Select **My keys** under **My Account**.
3. Select the option to create a new key.

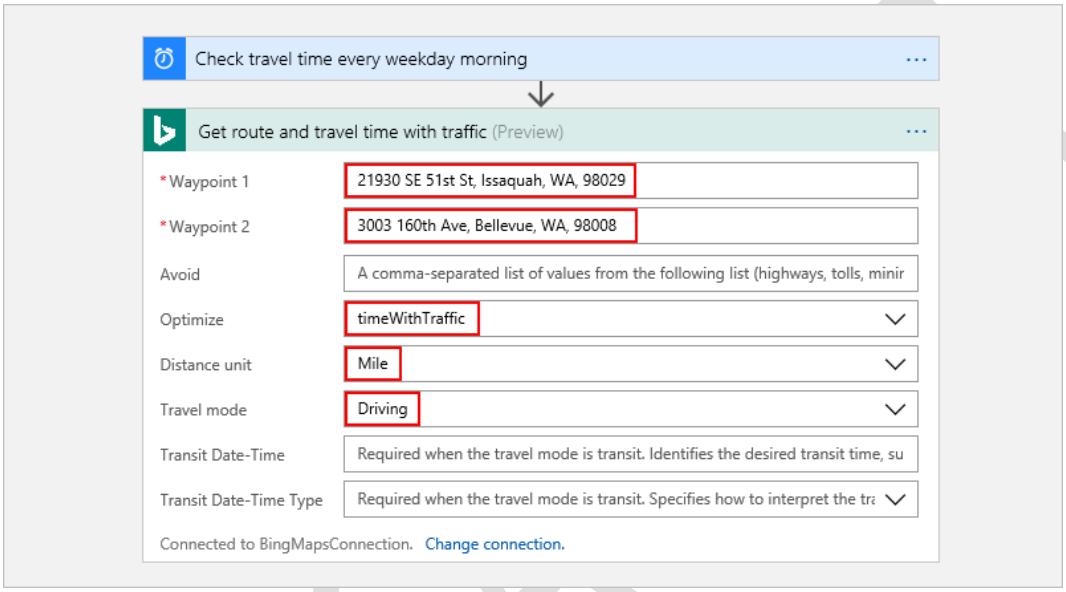
Sample Key: ArnSE6qQ6QDVWURsyUKYYgptwXJQ5WlahKNxQCOPUIOE6Bd06ttSZWuoSSSItHaF

Step 2: Create a Logic App

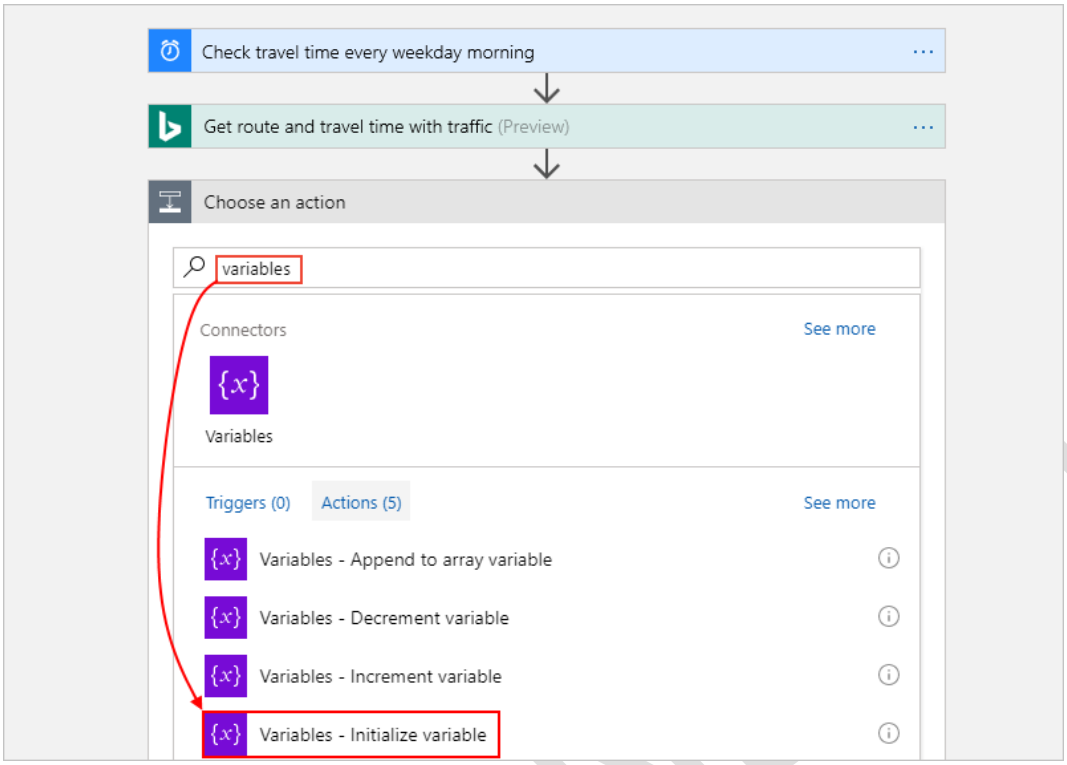
1. Create a blank logic app.
2. Add a recurrence trigger that works as a scheduler for your logic app.
3. Search for "maps", and select this action: **Bing Maps - Get route**



4. Provide details for the **Get route** action as shown and described here, for example:



5. Add an action that creates a variable, converts the travel time from seconds to minutes, and saves that result in the variable.

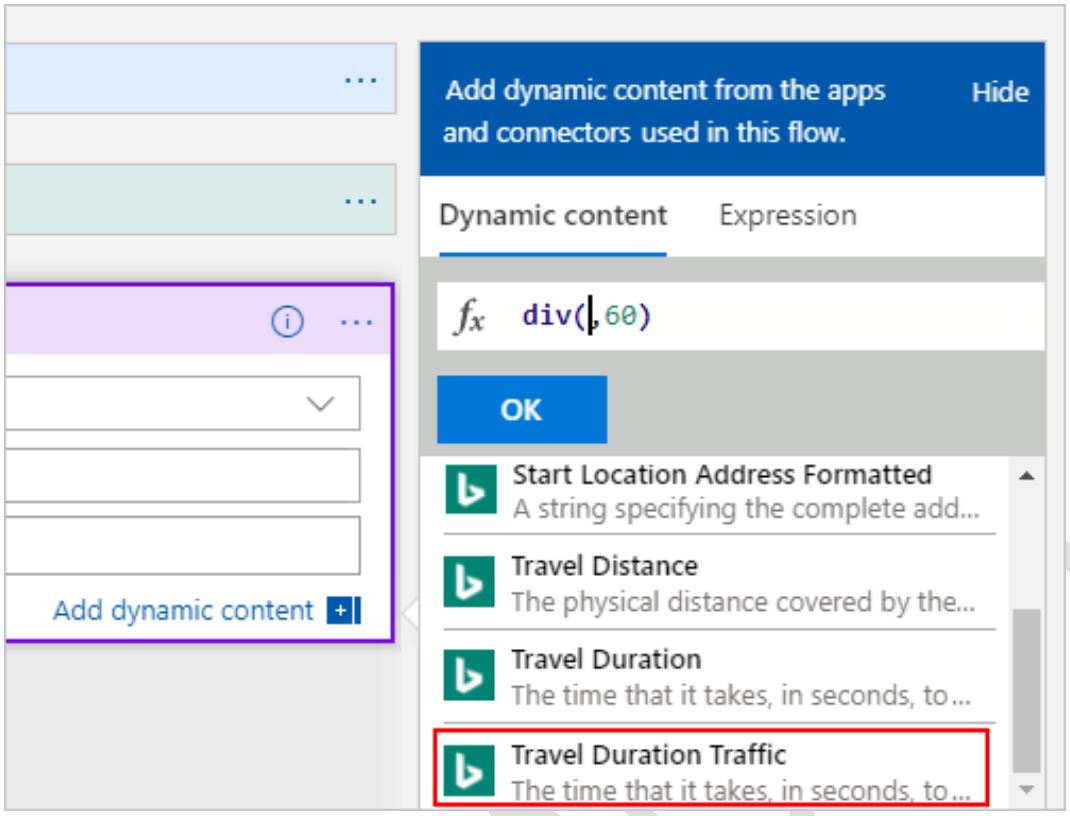


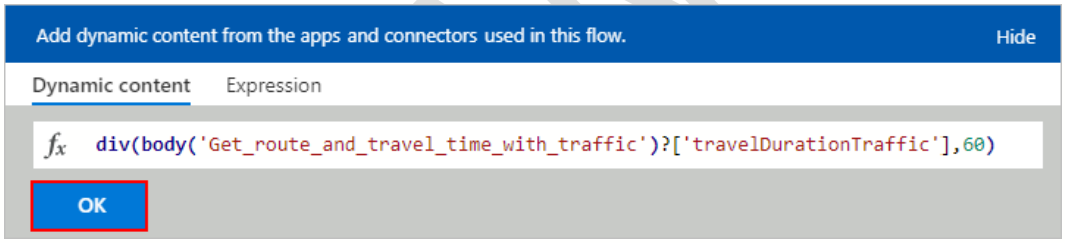
6. Provide the details for your variable as described here:

   Name = travelTime

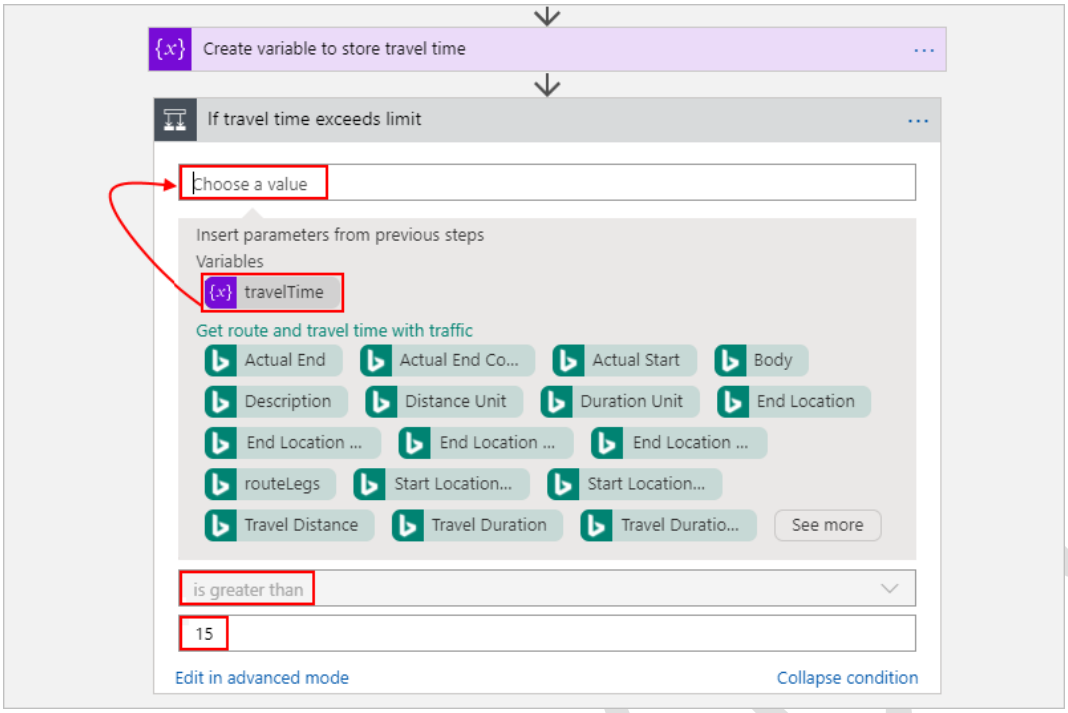
   Type = Integer

   Value = An expression that converts the current travel time from seconds to minutes (see steps under this table



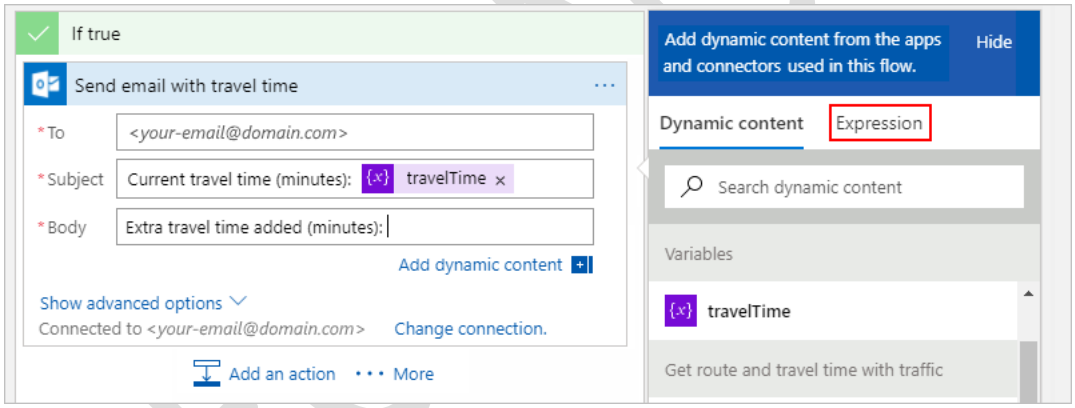


7. Add a condition that compares the travel time against a specified limit.



8. Add an action that sends email if the travel time exceeds the limit.

   Body: Extra travel time added (minutes) = **sub(variable('travelTime'),15)**



9. **Run your logic app.**

Add Custom Code with Azure Function

1. **Create and Azure Function App → Add Azure Function = CounterCharacters**

```
using System.Net;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Primitives;
using Newtonsoft.Json;
public static async Task<int> Run(HttpRequest req, ILogger log)
{
   string name = req.Query["name"];
```

```
    string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
    dynamic data = JsonConvert.DeserializeObject(requestBody);
    name = name ?? data?.name;
    int cnt = name != null?name.Length:0;
    return cnt;
}
```

2. From the **Function Apps** list, select your function app > **Platform features** > **CORS**. → Under **CORS**, add the * wildcard character, but remove all the other origins in the list, and choose **Save**.
3. **Either to an existing Logic App or a New Logic App: Add Action = Azure Function → Add existing functions to Logic App**

Note:

- Workflow Definition Language Reference: https://msdn.microsoft.com/en-us/library/azure/mt643789.aspx
- Workflow Actions and Triggers: https://msdn.microsoft.com/en-us/library/azure/mt643939.aspx

List of Logic API's

- https://azure.microsoft.com/en-us/documentation/articles/apis-list/

Using API App in Logic Apps

1. API App → Settings → CORS → Allowed Origins=* → Save
2. API App → Settings → API definition → Copy URL
3. Create a New Logic App
4. Add Triggers and Actions as per the requirement
5. To add an API App, Under Microsoft Managed API, **Search HTTP + Swagger**
6. For Swagger endpoint url = URL Copied in step2 → Select the appropriate method and Continue…

Functions vs. Logic Apps

Functions and Logic Apps can both create complex orchestrations. An orchestration is a collection of functions or steps, that are executed to accomplish a complex task. With Azure Functions, you write code to complete each step, with Logic Apps, you use a GUI to define the actions and how they relate to one another.

You can mix and match services when you build an orchestration, calling functions from logic apps and calling logic apps from functions. Here are some common differences between the two.

-	**Functions**	**Logic Apps**
State	Normally stateless, but Durable Functions provide state	Stateful

Development	Code-first (imperative)	Designer-first (declarative)
Connectivity	About a dozen built-in binding types, write code for custom bindings	Large collection of connectors, Enterprise Integration P build custom connectors
Actions	Each activity is an Azure function; write code for activity functions	Large collection of ready-made actions
Monitoring	Azure Application Insights	Azure portal, Log Analytics
Management	REST API, Visual Studio	Azure portal, REST API, PowerShell, Visual Studio
Execution context	Can run locally or in the cloud	Runs only in the cloud.