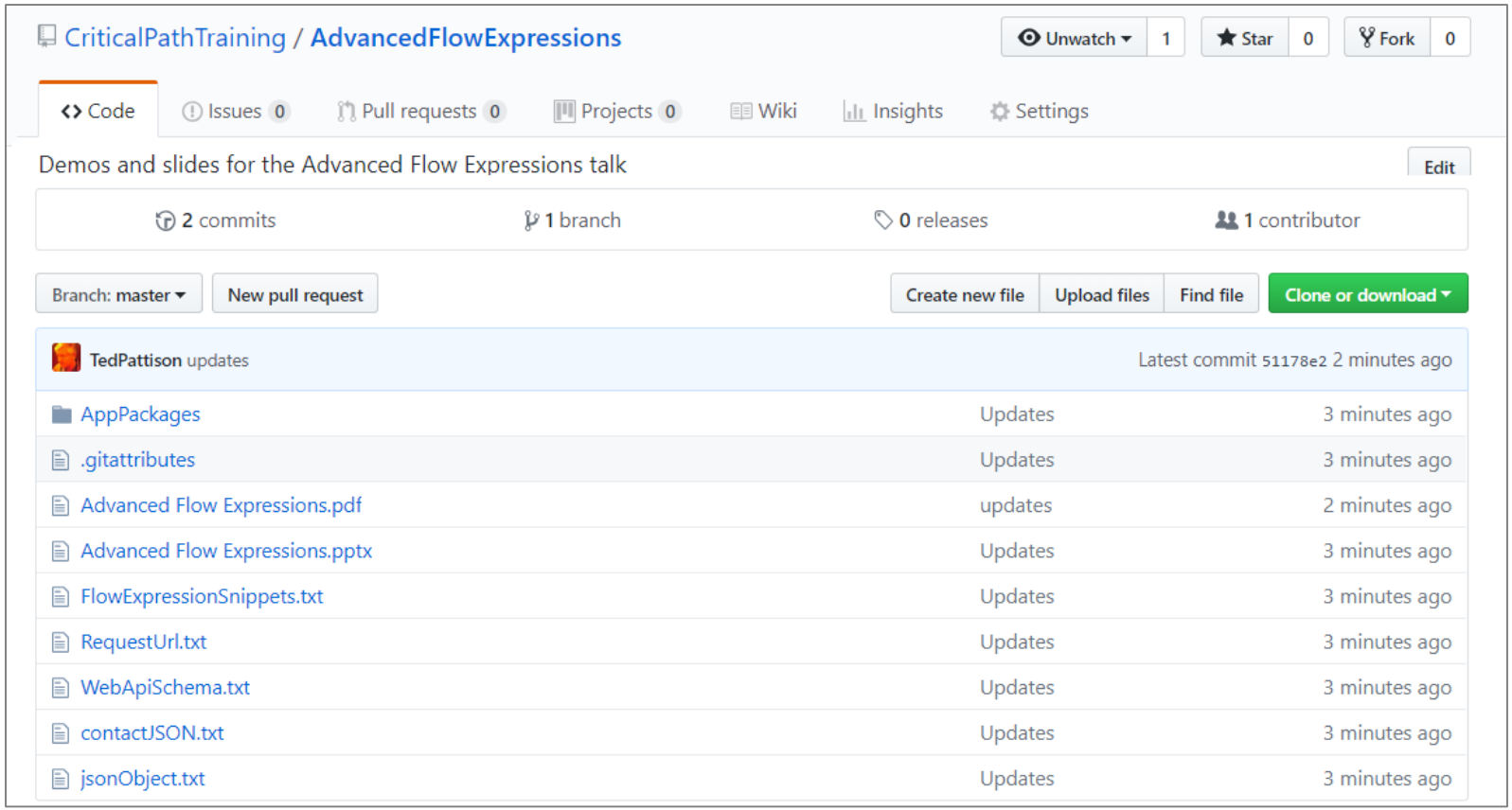
Writing Advanced Formulas in Microsoft Flow

Ted Pattison
Power platform MVP

PHOENIX, AZ | OCTOBER 15-18, 2018

GitHub Repo

- All slides and demo files available for download
  - https://github.com/CriticalPathTraining/AdvancedFlowExpressions



Agenda

- Flow Fundamentals
- Writing Flow Expressions
- Control of Flow
- Processing Data and Preparing Content
- Converting Between Types
- Advanced Techniques

Thanks for coming up with the idea for this session

Stephen Siciliano, Principal Group PM Manager

Deep Dive into PowerApps and Flow

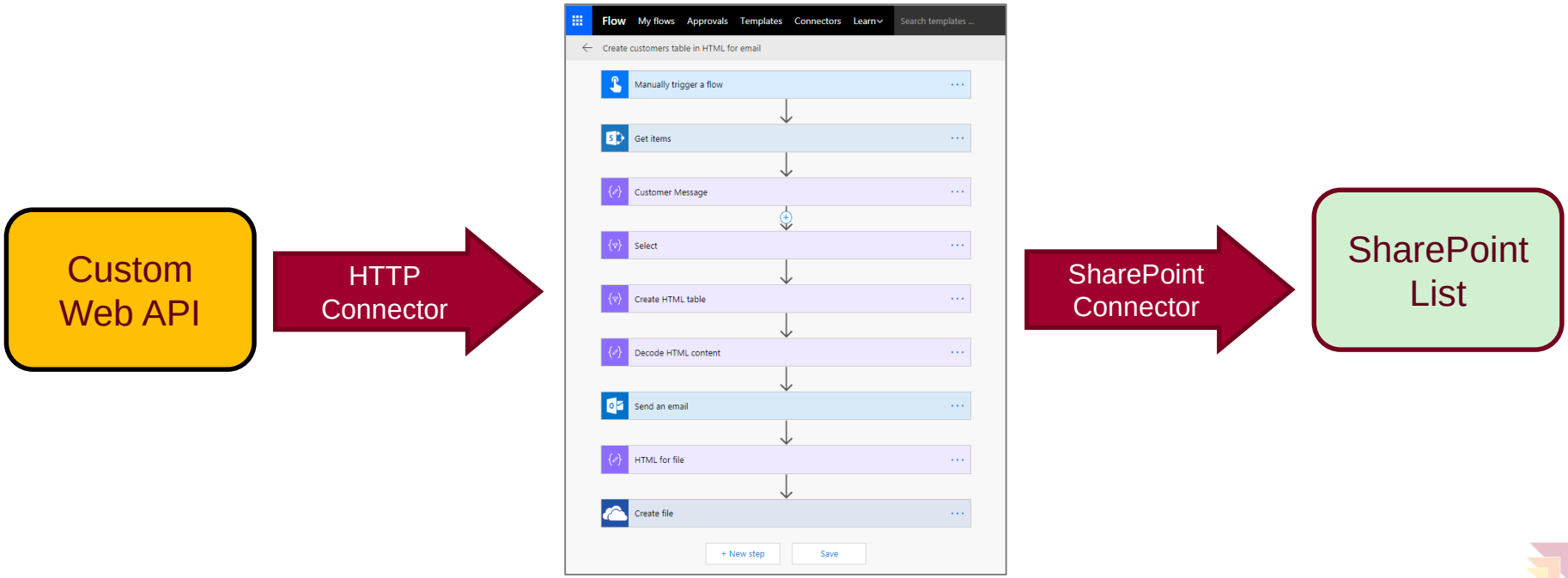
- Two action-packed days of building PowerApps and Flows
  1. Getting Started with PowerApps Studio
  2. Designing PowerApps using Advanced Techniques
  3. Building PowerApps for SharePoint Online
  4. Introduction to Microsoft Flow
  5. Designing Flows to Automate an Approval Process
  6. Building PowerApps and Flows for Power BI
  7. Working with the Common Data Service for Apps
  8. Managing Application Lifecycle with PowerApps and Flow

- More info
  - https://www.criticalpathtraining.com
  - info@criticalpathtraining.com

CriticalPath TRAINING

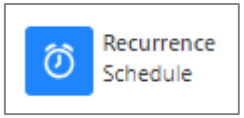
Building Blocks of Flow

- **Triggers** - events that start a flow
- **Actions** - tasks and operation executed by flow
- **Services** - sources and destinations for data
- **Connectors** - wrappers to communicate with service APIs

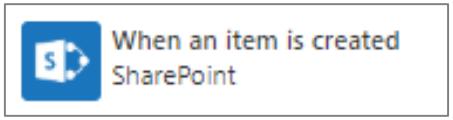


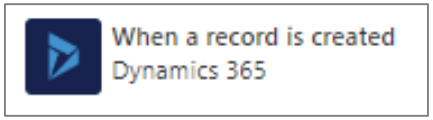
Flow Trigger Types

- Scheduled Flow Triggers
  - Runs periodically based on an interval

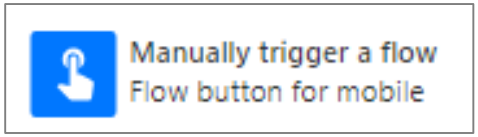


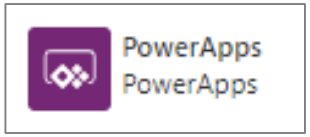
- Automated Flow Triggers
  - Runs when something happens





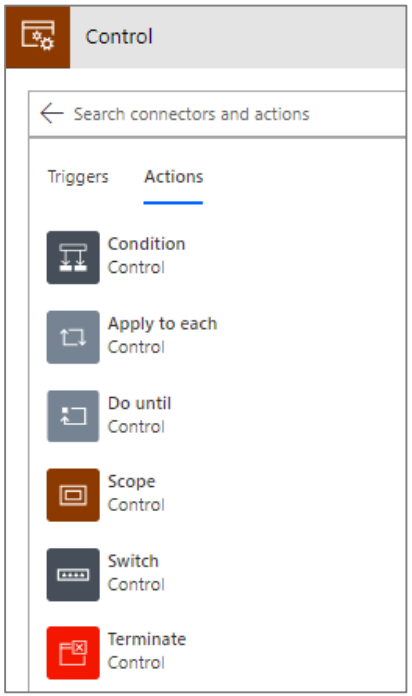
- On-demand Flow Triggers
  - Runs when a user clicks a button

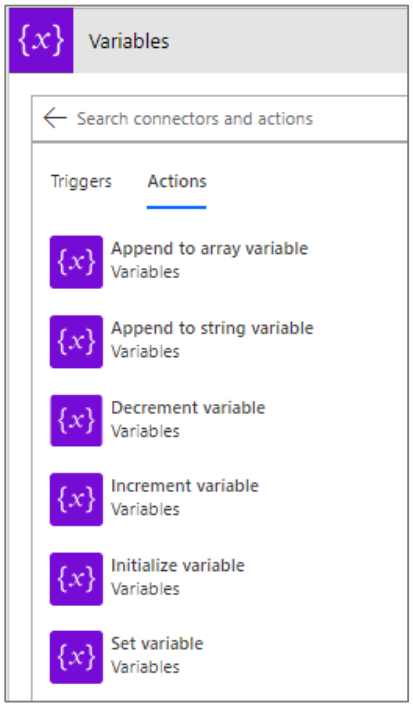


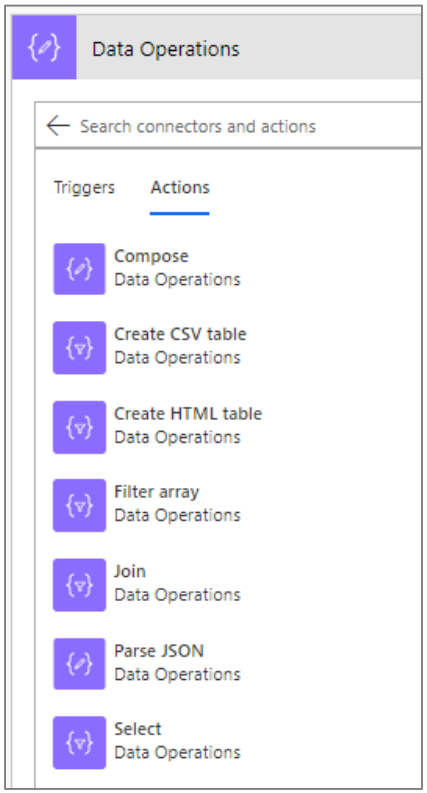


Core Action Categories

- **Control**: actions to provide control-of-flow
- **Variables**: actions to manage state within flow lifetime
- **Data operations**: action to process data & prepare content

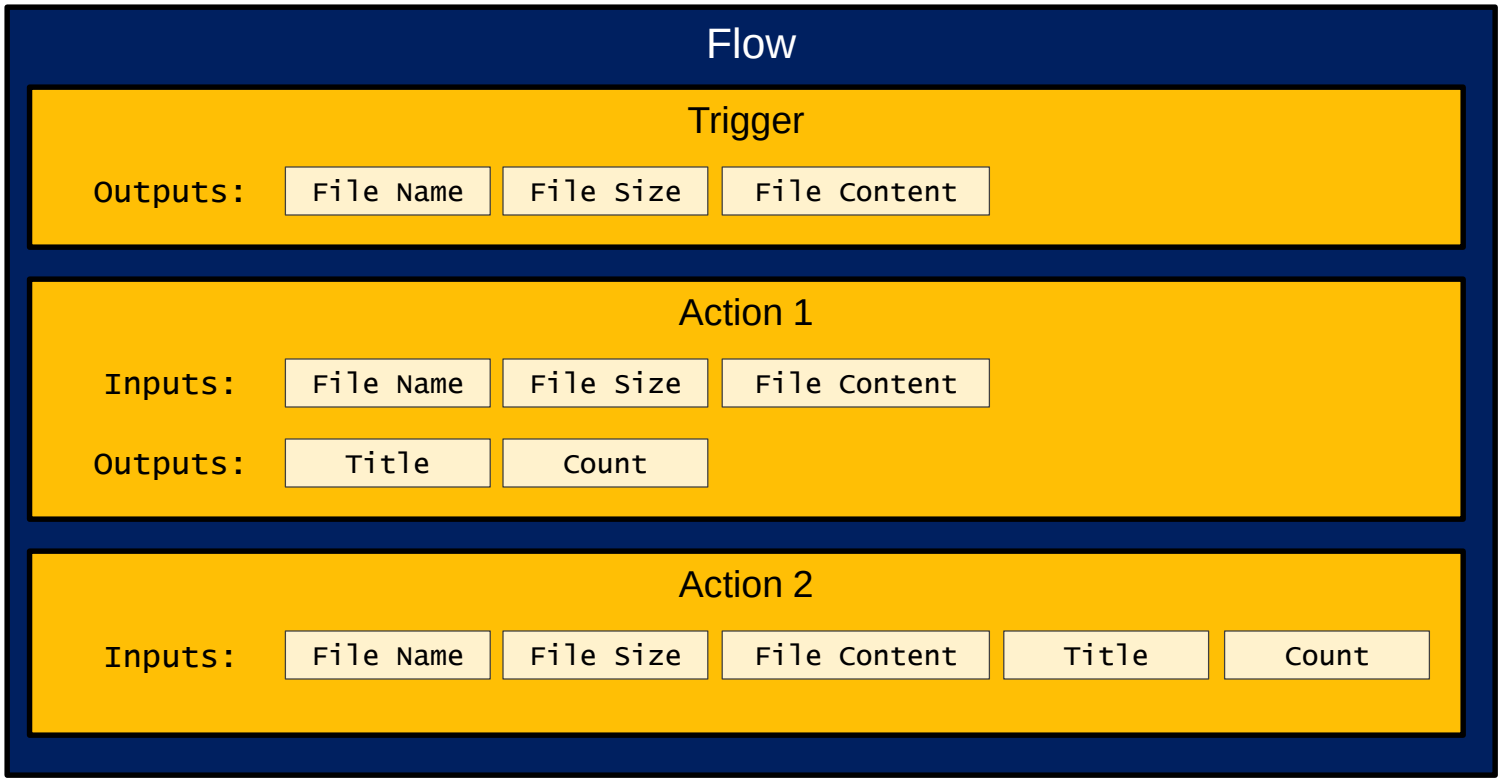






Data Automatically Flows from Step to Step

- Data in flows added by step outputs
  - Data added in step output is available in later steps
  - It's easy to configure step input data using output data in previous steps
  - Certain outputs displayed/hidden based on types of input and output



DEMO

Demo 1
Changing the Flow Trigger Type

Agenda

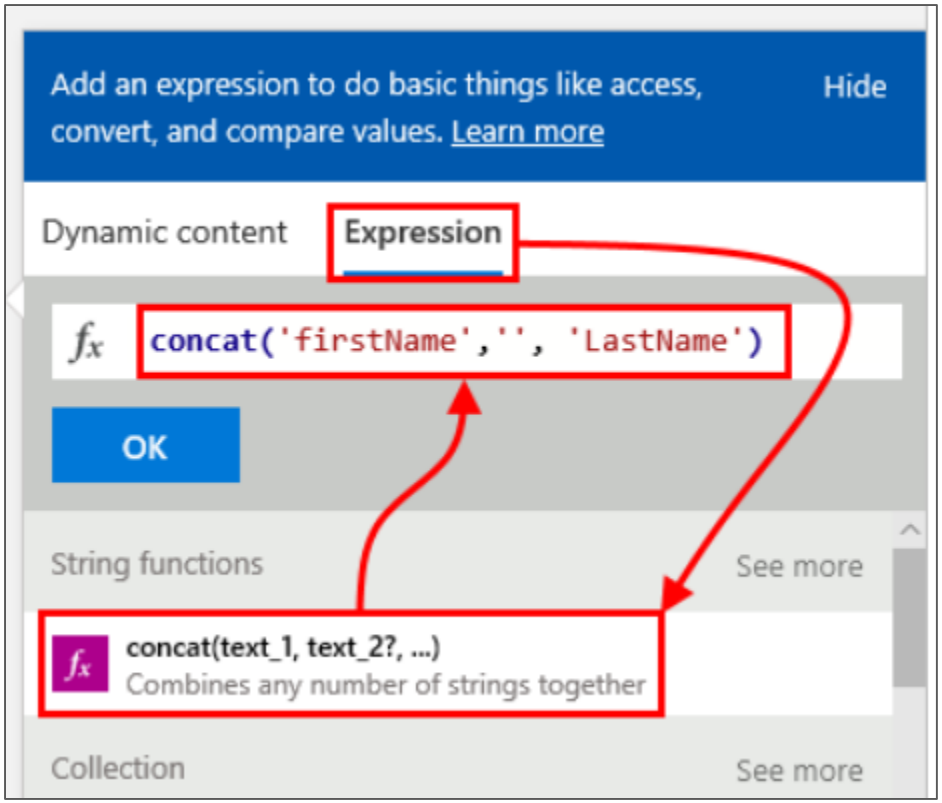
- ✓ Flow Fundamentals
- ➢ Writing Flow Expressions
- Control of Flow
- Processing Data and Preparing Content
- Converting Between Types
- Advanced Techniques

Writing Flow Expressions

- Scenarios for writing Flow expressions
  - Perform string manipulation
  - Generate a GUID or a random number
  - Convert types
  - Perform simple inline calculations
  - Handling optional values
  - Writing conditional statements using "If" statements
  - Working with arrays

Writing Expressions

- Expressions written in fx textbox
- Click OK to enter expressions



Workflow Definition Language (WDL)

- Flow expressions written in Workflow Definition Language
  - Same language used in Azure Logic Apps
  - WDL is more powerful yet more complicated than PowerApps
  - WDL does not overload operators like PowerApps does
  - WDL requires single quotes instead of double quotes

```
fx "this is a test"
```

Invalid: no double quotes

```
fx 'this is a test'
```

Valid

```
fx 'this is a ' + 'test'
```

Invalid : **+** operator not supported

```
fx 'this is a ' & 'test'
```

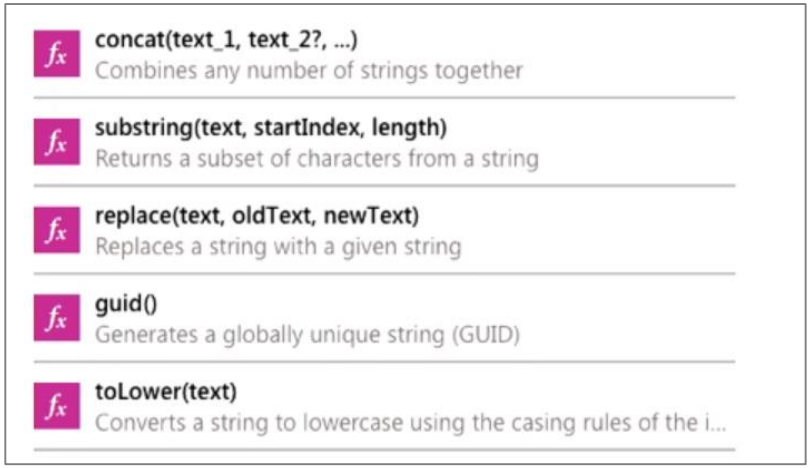
Invalid : **&** operator not supported

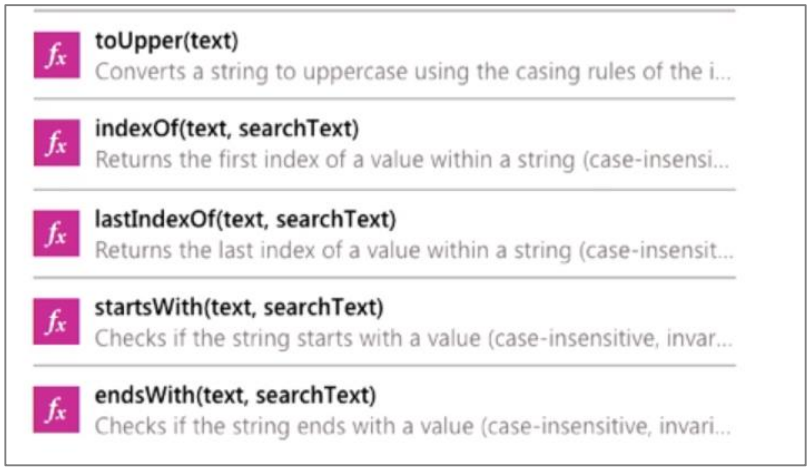
```
fx concat('this is a ', 'test')
```

Valid

Working with Strings

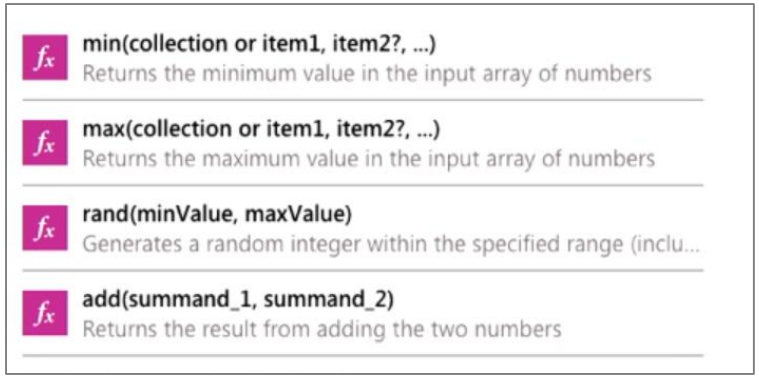
- Parse text together using **concat()**
- Parse out text using **substring()**
- Convert casing using **toLower()** and **toUpper()**
- Search string using **indexOf** and **startsWith()**
- Create new GUID identifier using **guid()**

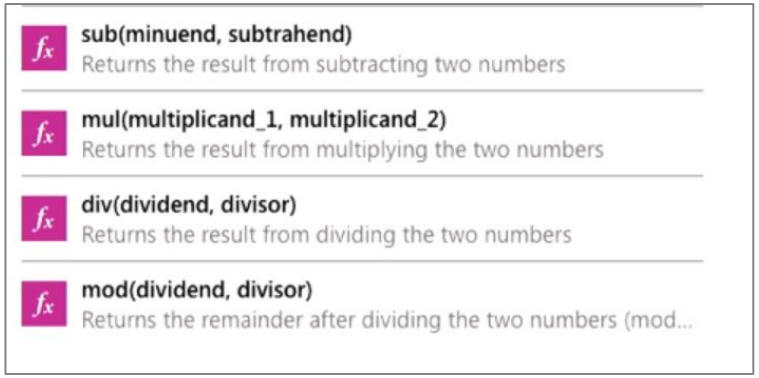




Performing Arithmetic Operations

- You cannot use standard arithmetic operators
  - No support for familiar operators such as **+**, **-**, ***, **/**
  - This does not work: **2 + 2**
  - This works: **add(2, 2)**





DEMO

Demo 2
Adding SharePoint List Items

Agenda

- ✓ Flow Fundamentals
- ✓ Writing Flow Expressions
- ➢ Control of Flow
- Processing Data and Preparing Content
- Converting Between Types
- Advanced Techniques

Understanding Arrays in Flow

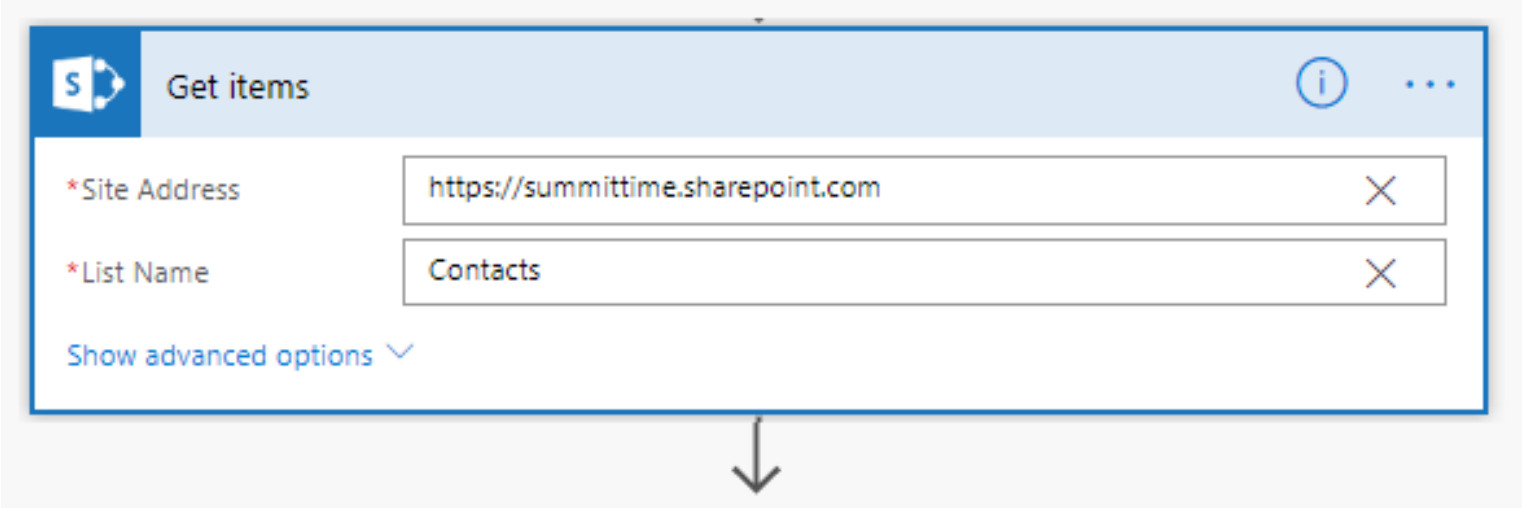
- Flow arrays are zero-based
  - Primitive value arrays

0	Daugherty
1	Hernandez
2	Mack
3	Wiley

  - Object arrays

	Last Name	First Name	Company	Business Phone	Home Phone
0	Daugherty	Cindy	Wonka Industries	1(337)111-4444	1(337)111-7777
1	Hernandez	Zane	Vandelay Industries	1(757)666-3333	1(757)777-1111
2	Mack	Chang	Wonka Industries	1(480)111-4444	1(480)777-0000
3	Wiley	Ramona	Ecumena	1(201)777-8888	1(201)777-2222

Accessing an Array using ['value']

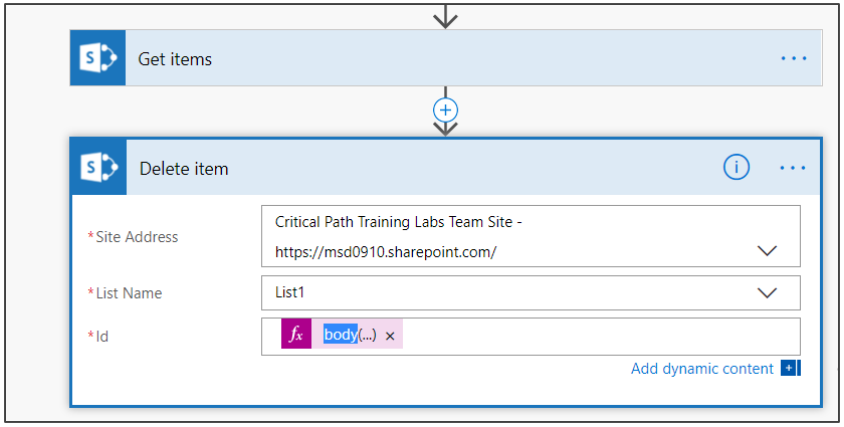


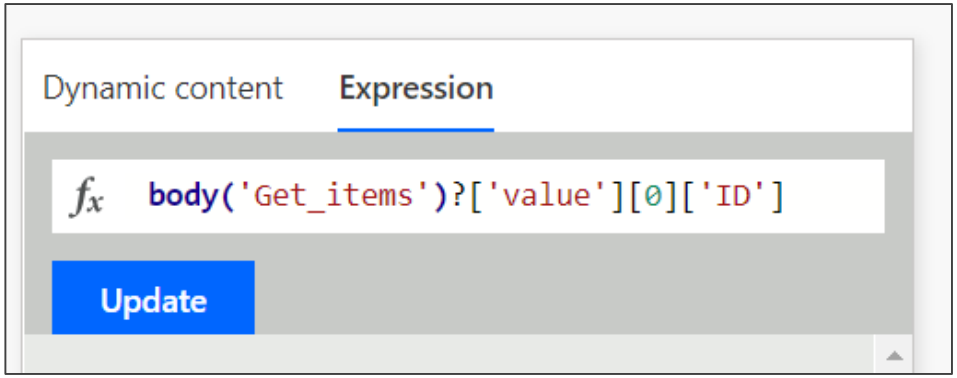

```
body('Get_items')?['value']
```

	Last Name	First Name	Company	Business Phone	Home Phone
0	Daugherty	Cindy	Wonka Industries	1(337)111-4444	1(337)111-7777
1	Hernandez	Zane	Vandelay Industries	1(757)666-3333	1(757)777-1111
2	Mack	Chang	Wonka Industries	1(480)111-4444	1(480)777-0000
3	Wiley	Ramona	Ecumena	1(201)777-8888	1(201)777-2222

Retrieving List Items

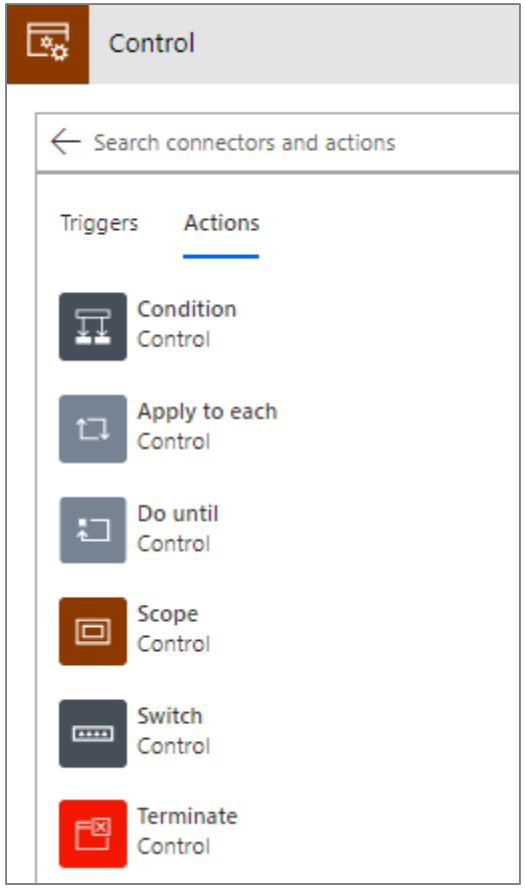
- Use **first()** and **last()** to get lead at head or tail
- Individual items retrieved using zero-based array syntax
  - SharePoint list item array - `body('Get_items')?['value']`
  - First item field value - `body('Get_items')?['value'][0]['ID']`



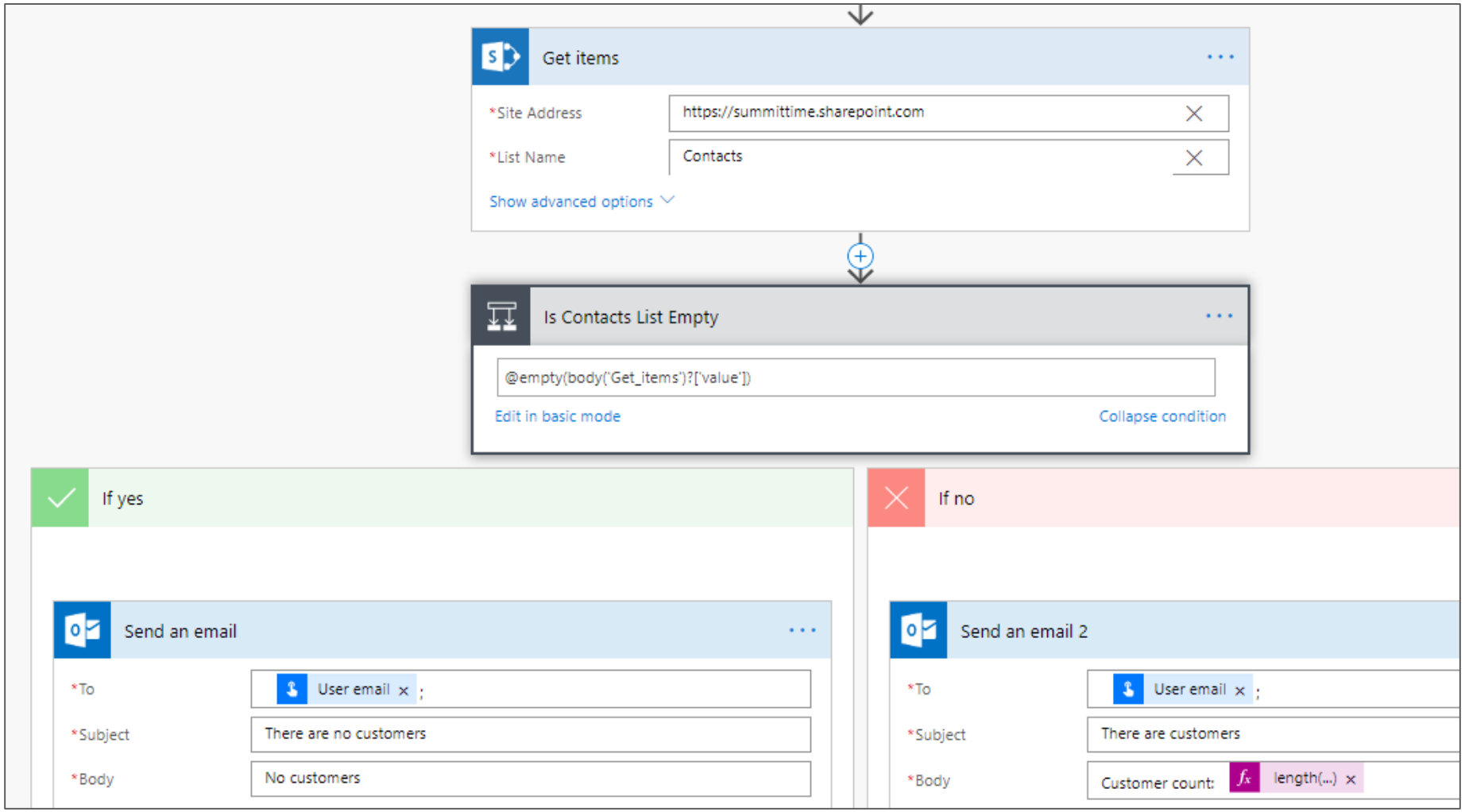


Control of Flow

- Condition
  - Provides logical structure for If Then Else
- Apply to each
  - Enumerate through collection (e.g. list items)
- Do until
  - Repeat until condition changes
- Scope
  - Create an action container with a private scope
- Switch
  - Select Case flow
- Terminate
  - Completes a flow

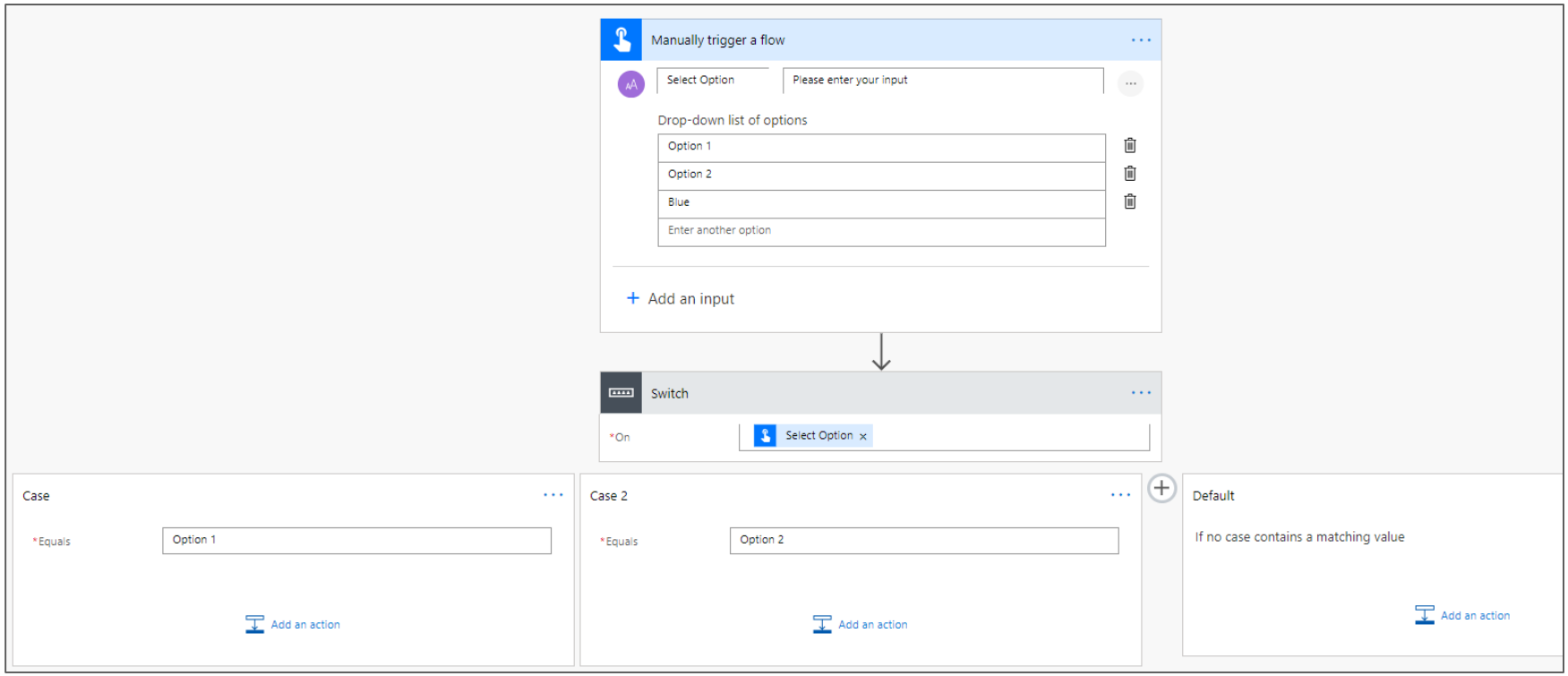


Condition Action



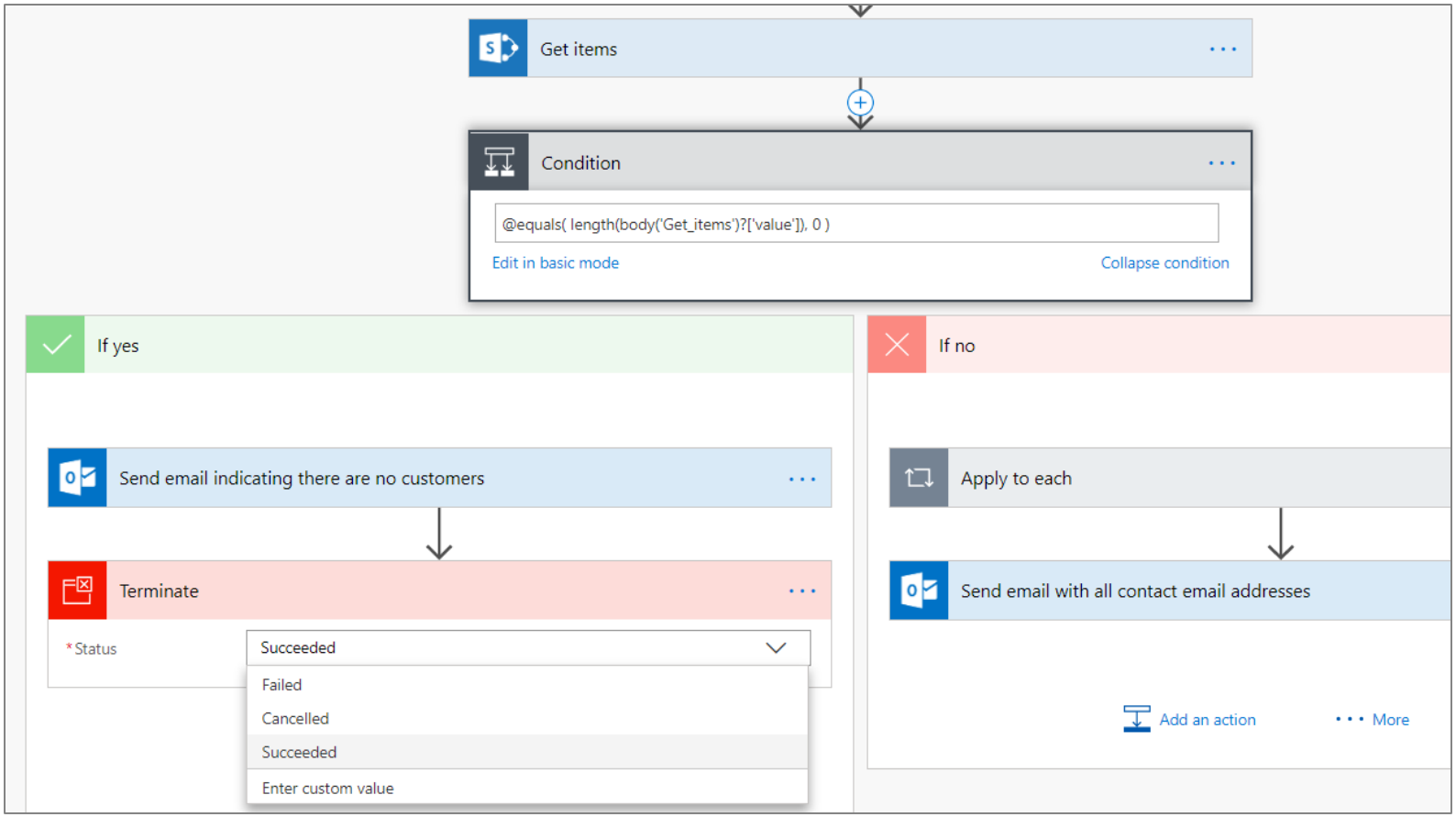
Switch Action

- Switch actions provides cases
  - Each case represents separate execution path
  - Only one execution path will execute
  - Default case executes when no other case is matched



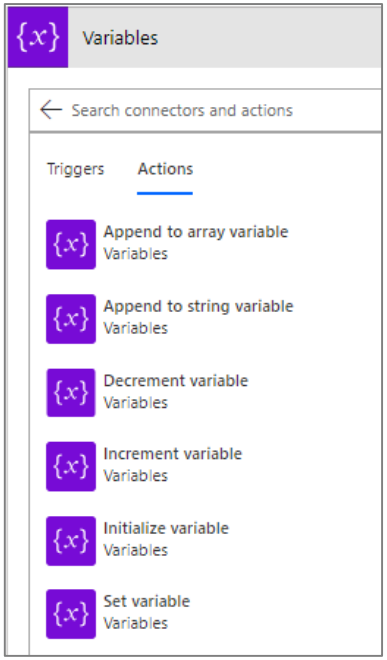
Terminate action

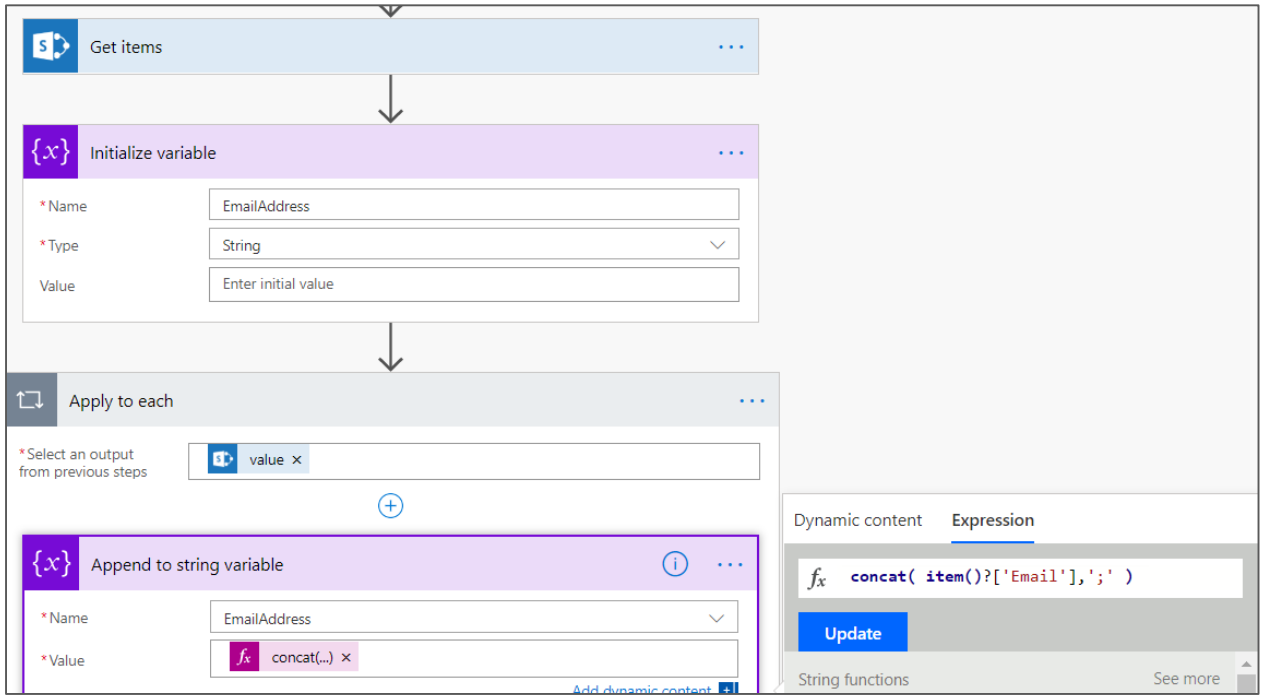
- Used to stop a flow at any point
  - Terminate status can be set to Succeeded, Cancelled, Failed



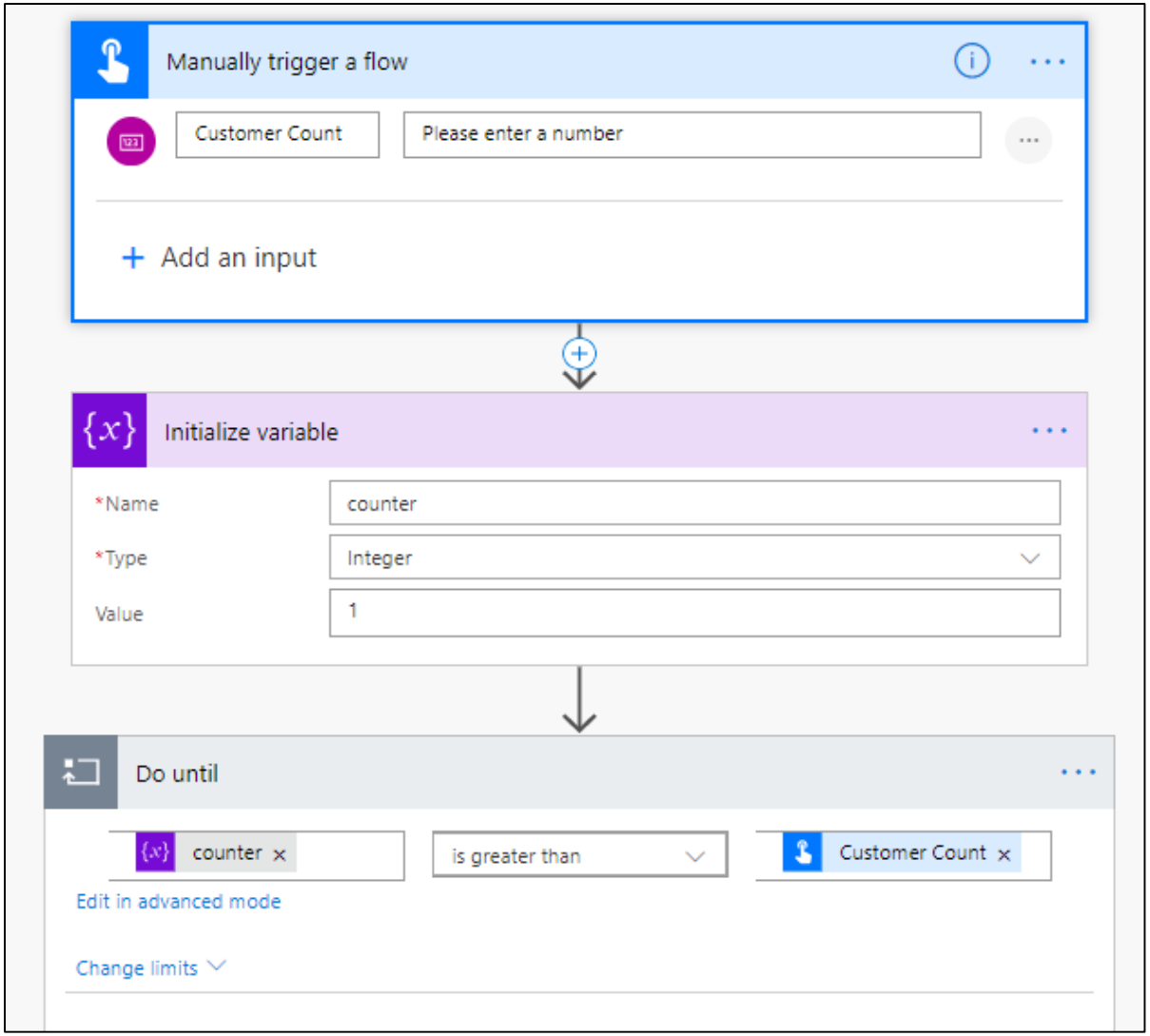
Tracking State using Variables

- Variables used to track state during flow lifetime
  - Initialize Variable used to create variable with Type and Value
  - Other variable actions uses to update variable values
  - By default, variable stored within flow until end of flow lifetime
  - Variables can be initialized inside Scope action to reduce lifetime

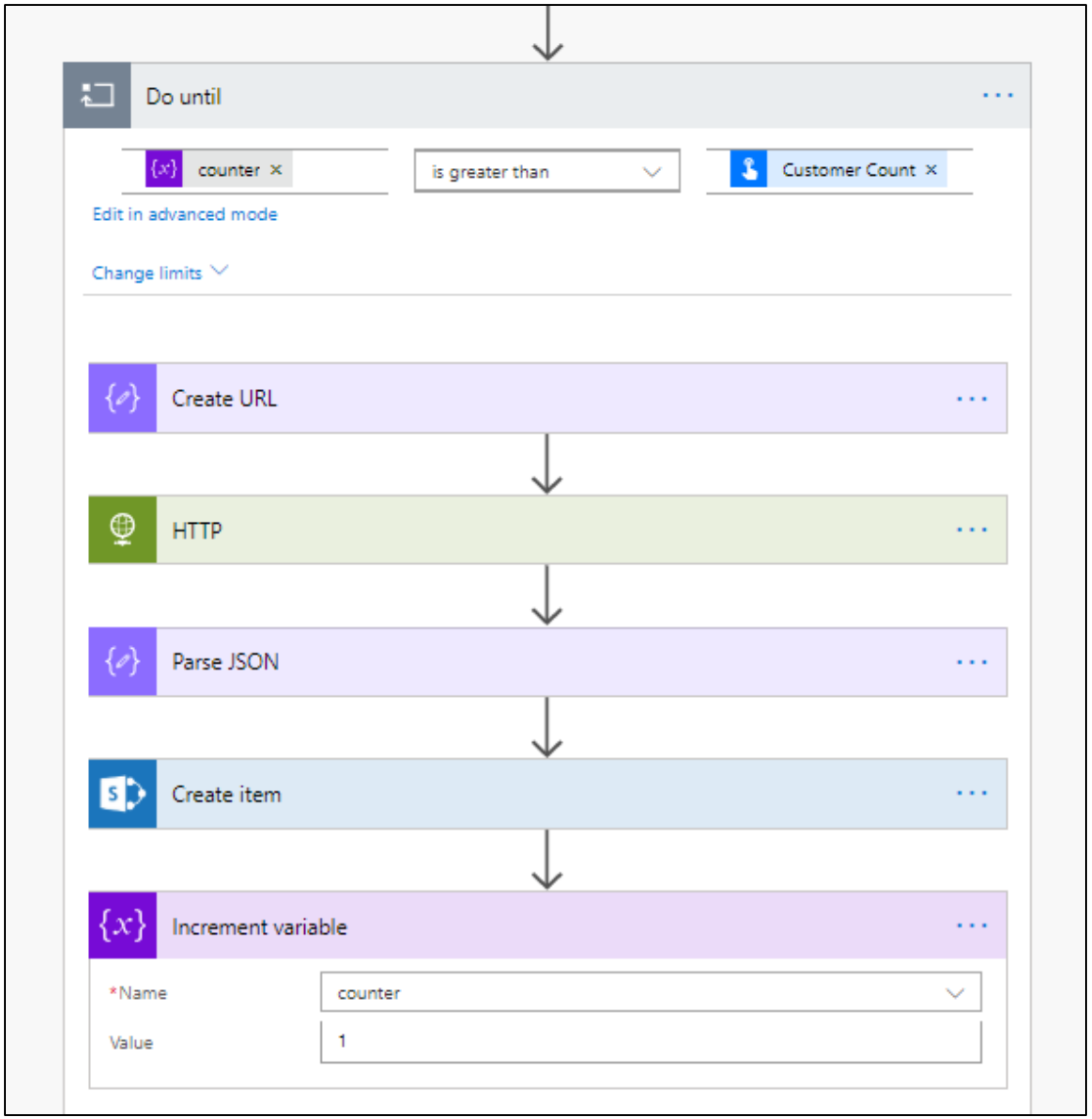




Do Until Action with Counter Variable

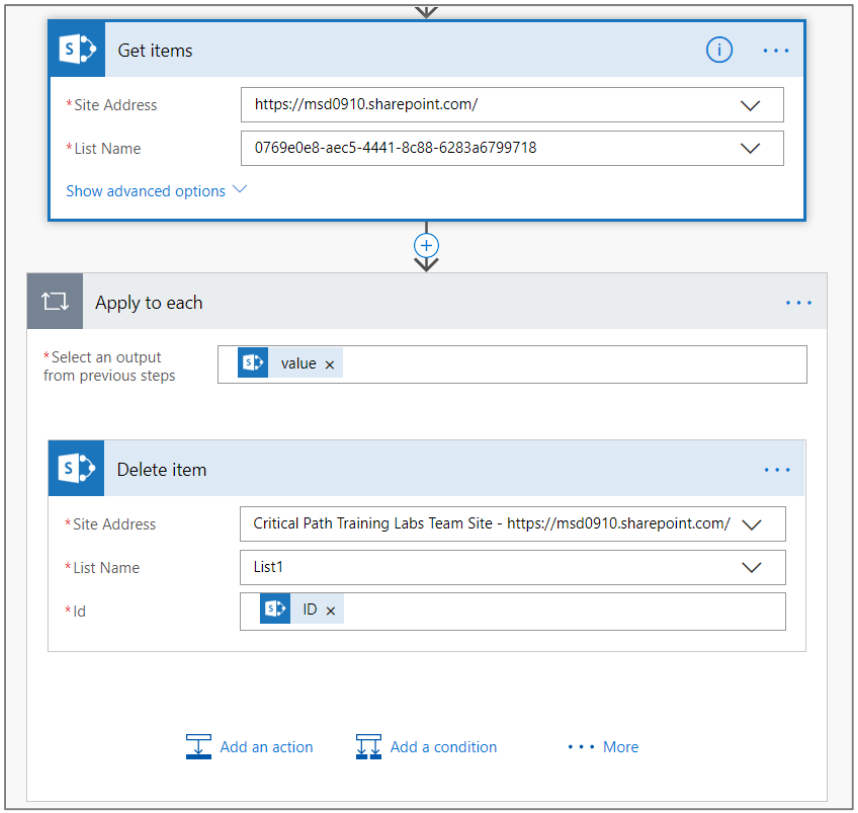


Executing Operations inside Do Until Loop



Using Apply to Each

- Automatically added when list is used from output
- Destination step enumerates over list items



DEMO

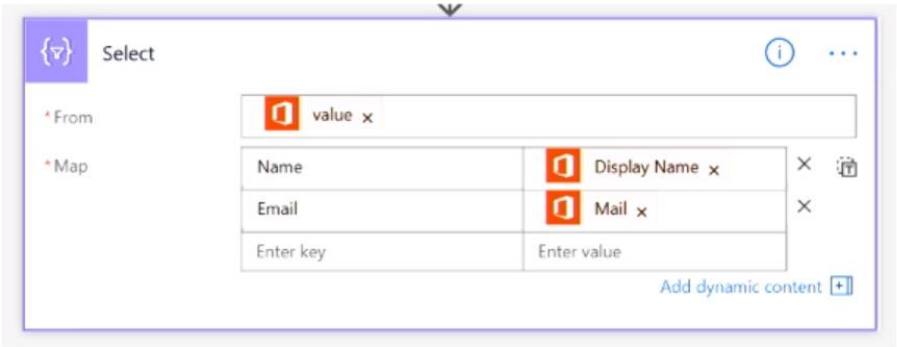
Demo 3
Apply to Each and Do Until

Agenda

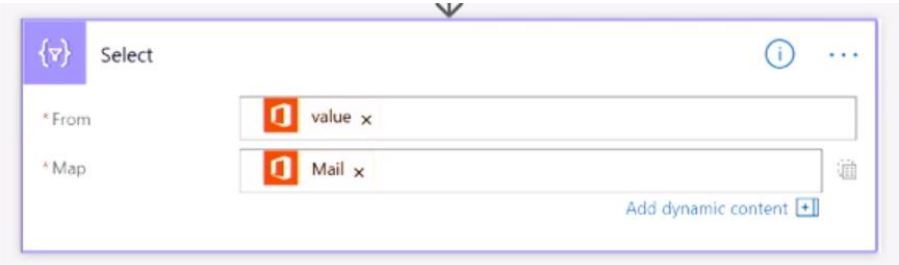
- ✓ Flow Fundamentals
- ✓ Writing Flow Expressions
- ✓ Control of Flow
- ➢ Processing Data and Preparing Content
- Converting Between Types
- Advanced Techniques

Transforming Arrays

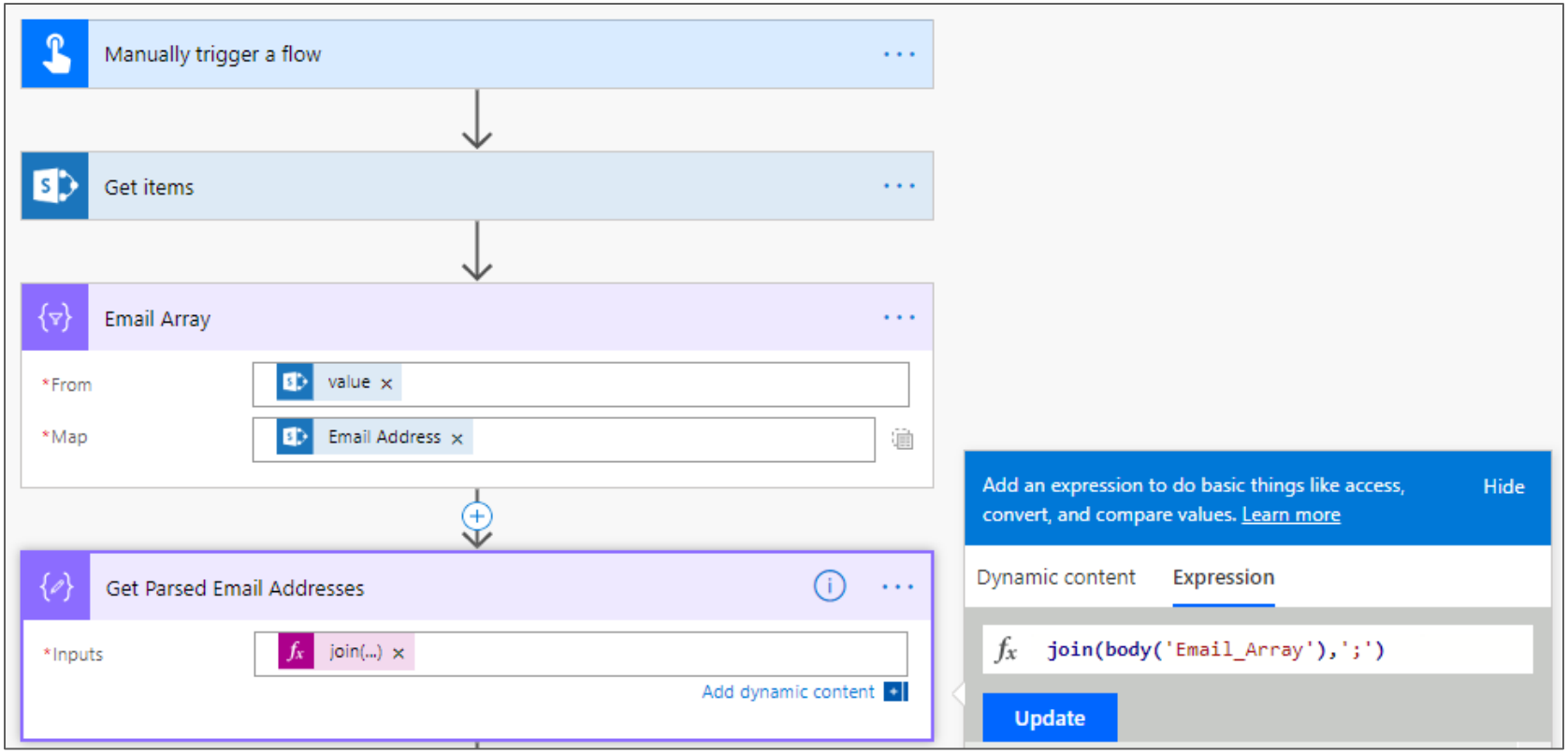
- Use Select action
  - Two input modes: fill key-value pairs or typing directly
- Create array object objects
  - Useful for passing array to another action



- Create a simple array of strings, numbers, Booleans, etc
  - Useful for creating simple list (e.g. email addresses)

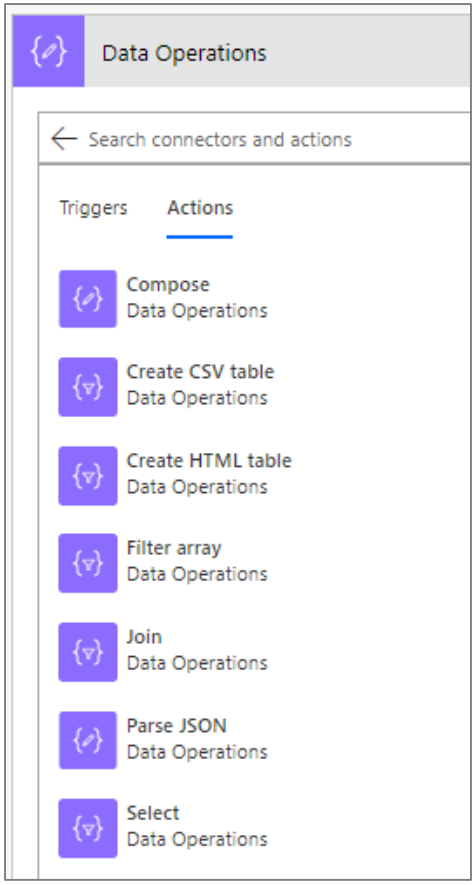


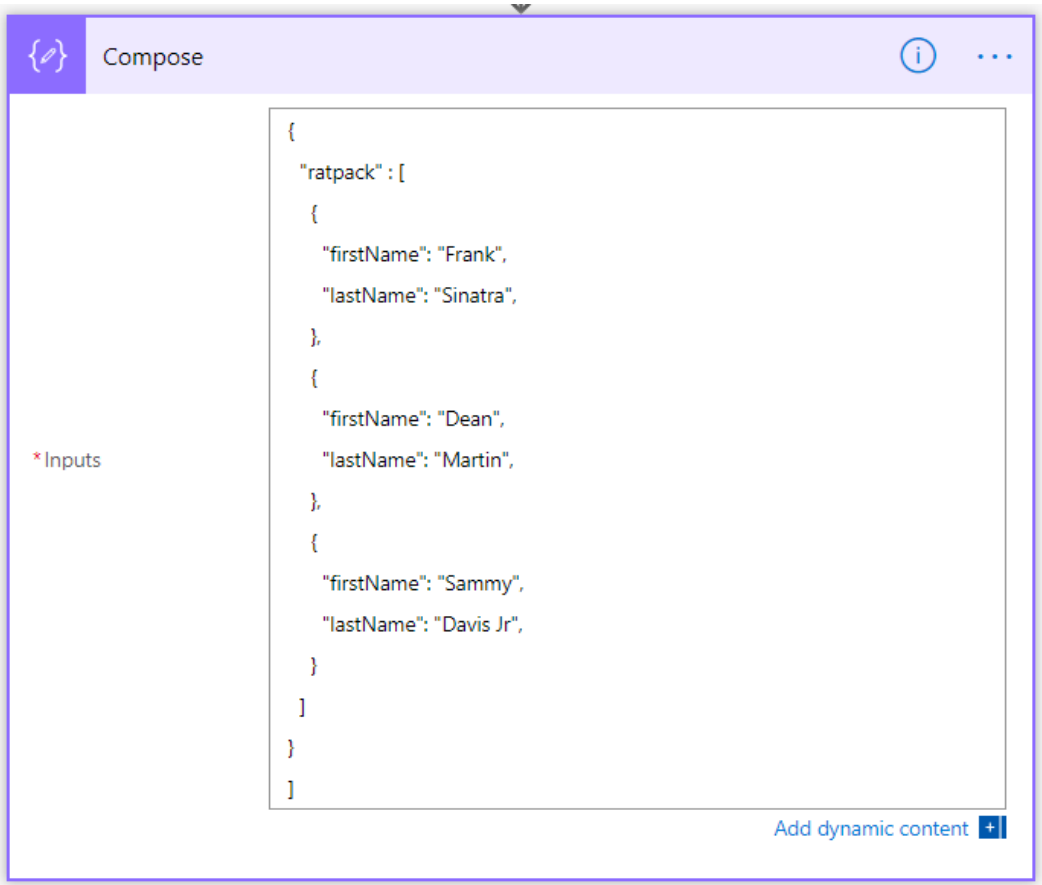
Converting an Array using Select



Data Operations

- Used to process data and prepare content





DEMO

Demo 4
Generating an HTML Table

Agenda

- ✓ Flow Fundamentals
- ✓ Writing Flow Expressions
- ✓ Control of Flow
- ✓ Processing Data and Preparing Content
- ➢ Converting Between Types
- • Advanced Techniques

Handling Type Conversion

- Some conversion is automatic
  - Sometimes conversions are performed for you
  - In other cases, you must explicitly convert between types

string(value)
Convert the parameter to a string

float(value)
Convert the parameter argument to a floating-point number

bool(value)
Convert the parameter to a Boolean

base64(value)
Returns the base 64 representation of the input string

base64ToBinary(value)
Returns a binary representation of a base 64 encoded string

base64ToString(value)
Returns a string representation of a base 64 encoded string

binary(value)
Returns a binary representation of a value

dataUriToBinary(value)
Returns a binary representation of a data URI

dataUriToString(value)
Returns a string representation of a data URI

dataUri(value)
Returns a data URI of a value

uriComponent(value)
Returns a URI encoded representation of a value

uriComponentToBinary(value)
Returns a binary representation of a URI encoded string

uriComponentToString(value)
Returns a string representation of a URI encoded string

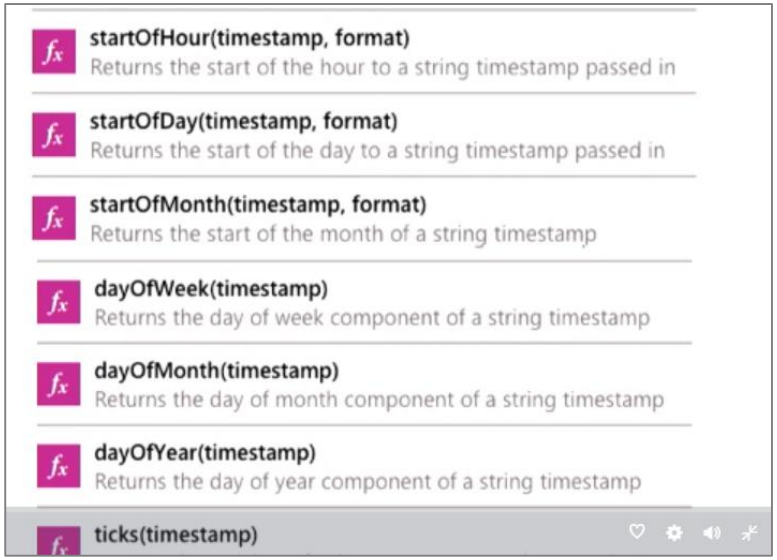
Flow Type Conversion Matrix

UI adds automatically

To → / From ↓	String	Base 64	Binary content	Data URI	URI comp.	Floating-point	Integer	Bool.	Array	JSON Object	XML content
String	Yes	base64()	binary()	dataUri()	uriComponent()	float()	int()	bool()	split() json()	json()	xml()
Base 64	base64ToString()	Yes	base64ToBinary()	*	*	*	*	*	*	*	*
Binary content	string()	base64()	Yes	dataUri()	uriComponent()	*	*	*	*	*	*
Data URI	dataUriToString()	*	dataUriToBinary()	Yes	*	*	*	*	*	*	*
URI comp.	uriComponentToString()	*	uriComponentToBinary()	*	Yes	*	*	*	*	*	*
Floating-point	Yes	base64()	binary()	dataUri()	uriComponent()	Yes	No	No	No	No	No
Integer	Yes	base64()	binary()	dataUri()	uriComponent()	Yes	Yes	No	No	No	No
Bool.	Yes	base64()	binary()	dataUri()	uriComponent()	No	No	Yes	No	No	No
Array	join() string()	*	*	*	*	No	No	No	Select Action	Select or Compose	xml()
JSON object	string()	*	*	*	*	No	No	No	Select or Compose	Compose Action	xml()
XML content	string()	*	*	*	*	No	No	No	xpath()	xpath()	Logic apps only

Working with Dates and Time

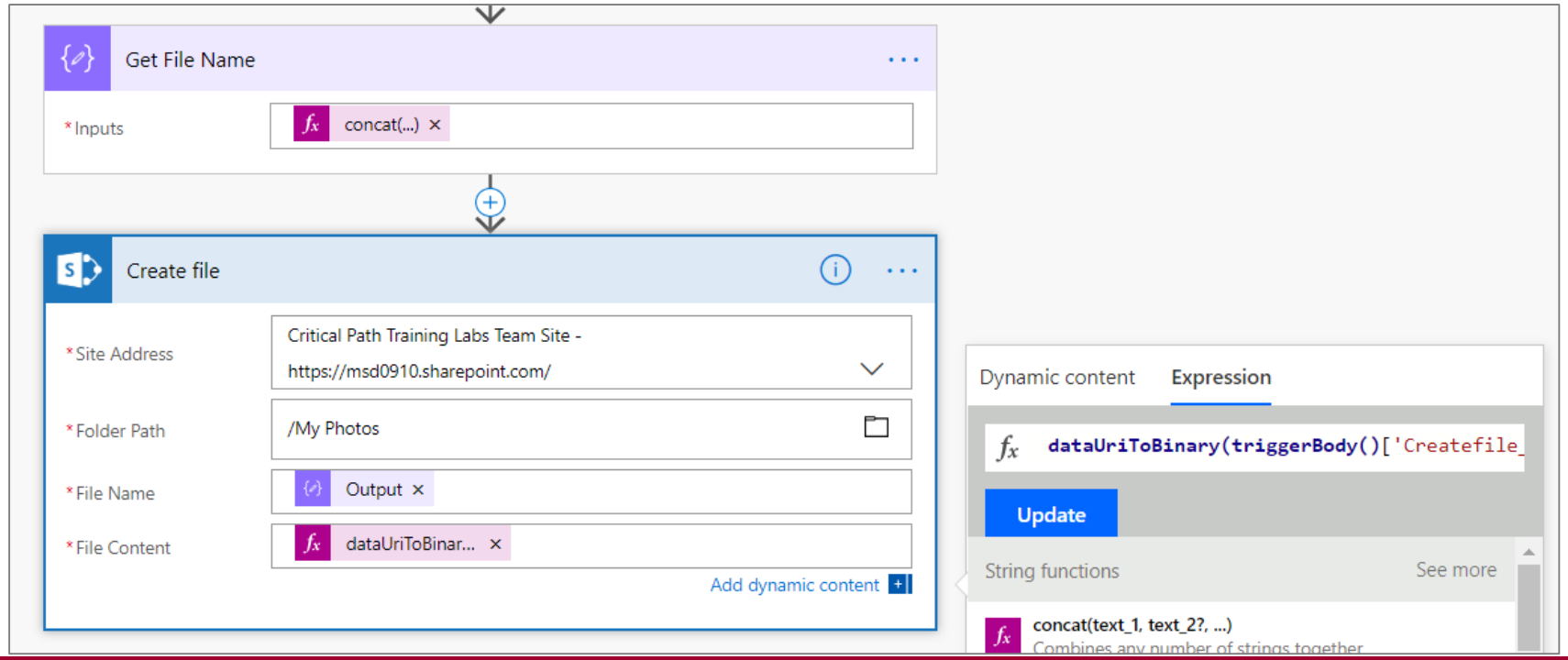
- Get Greenwich Meantime using **utcnow()**
- Use **add*()** functions to move time back/forward
- **convertTimeZone()** used to handle local times
- **formatDateTime()** used to format



dataUriToBinary()

- PowerApps photos require conversion
  - Allows you to upload phots to SharePoint
  - Accomplished using **dataUriToBinary()** function

```
dataUriToBinary(triggerBody()['Createfile_FileContent'])
```



DEMO

Demo 5
Uploading Mobile Photos to SharePoint

Agenda

- ✓ Flow Fundamentals
- ✓ Writing Flow Expressions
- ✓ Control of Flow
- ✓ Processing Data and Preparing Content
- ✓ Converting Between Types
- ➢ Advanced Techniques

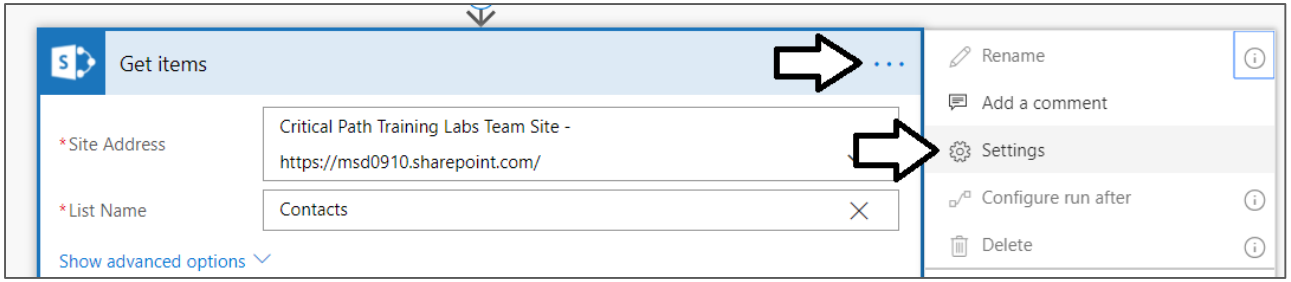
Normal action execution

- Standard behavior of a flow
  - Action steps execute in sequential order
  - Flow terminates if error occurs (failure or timeout)

- After flow runs, every action left in 1 of 4 possible states

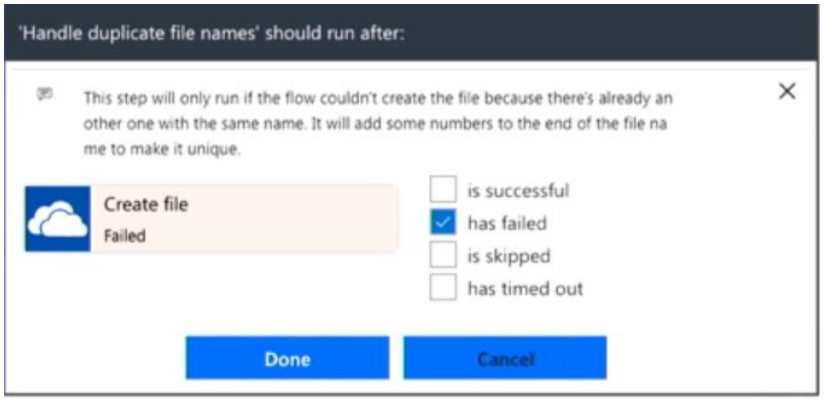
Action Settings

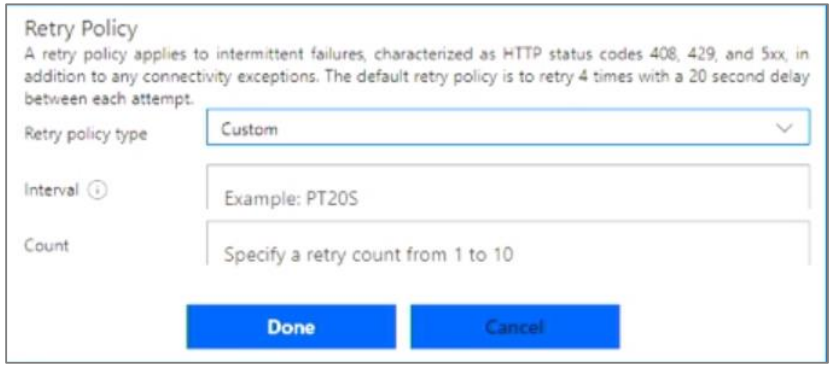
- Settings let you configure
  - Async Actions
  - Timeouts
  - Retry Policy
  - Sequential Behavior
  - And more!



Error Handling

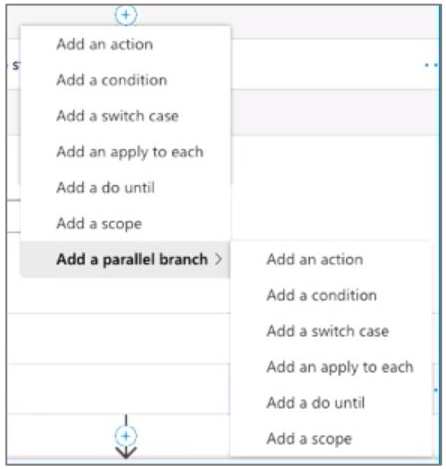
- Select the **Run after** option from action menu
  - Choose which error conditions, the arrow will turn dotted red
  - Use parallels for errors that are not at end of flow
  - Retry policy by default handles transient failures
  - Recommended to select exponential as they last a long time

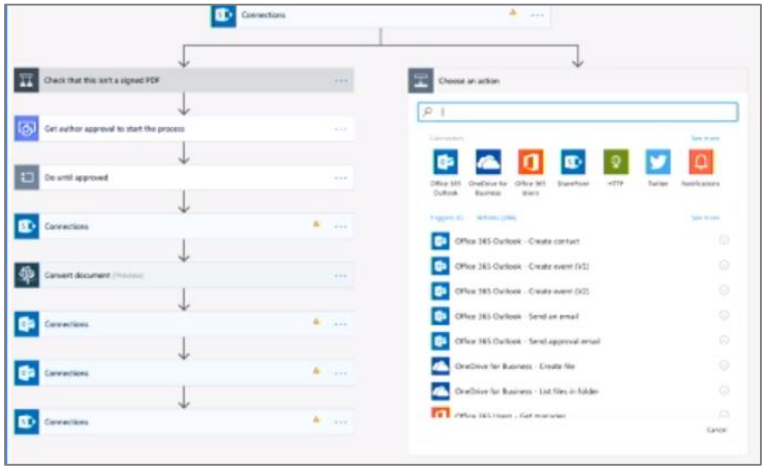




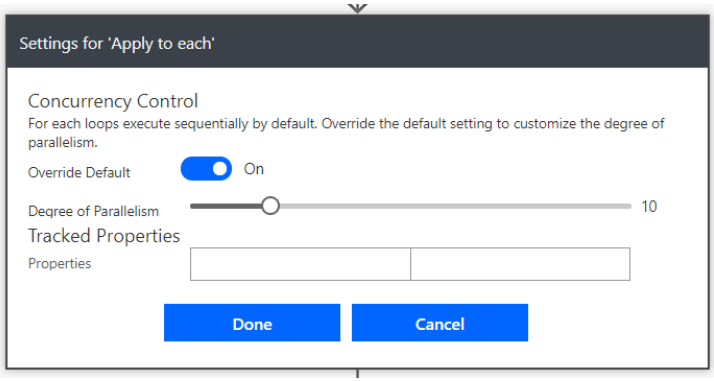
Parallel Execution

- Add parallel branch from above using ⊕





- Apply to each is sequential by default
  - Adding parallel execute to Apply to each



Calling Flows using the HTTP action

In the parent workflow:

HTTP

* Method: POST

* Uri: https://prod-56.westus.logic.azure.com:443/workflows/419593519f754af7a37f07ab5736bb98/triggers/manual/paths/invoke?api-version=2016-06-01&sp=%2Ftriggers%2Fmanual%2Frun&sv=1.0&sig=vYgMxpt4ArsyNbXFxnZMFeZcgoGx2KqW-ttAKJr3fGQ

Headers: Enter key | Enter value

Body:
```
{
"message" : "message"
}
```

Add dynamic content

Show advanced options

In the child workflow:

Request

HTTP POST URL: https://prod-56.westus.logic.azure.com:443/workflows/419593519f75...

Request Body JSON Schema

Example:
```
{
    "type": "object",
    "properties": {
        "address": {
            "type": "string"
        }
    },
    "required": ["address"]
}
```

Use sample payload to generate schema

Show advanced options

Response

* Status Code: 200

Headers: Enter key | Enter value

Body:
```
{
    "message": "Success!"
}
```

Add dynamic content

Summary

- ✓ Flow Fundamentals
- ✓ Writing Flow Expressions
- ✓ Control of Flow
- ✓ Processing Data and Preparing Content
- ✓ Converting Between Types
- ✓ Advanced Techniques