DEFINRA

Error Mixology: Crafting Resilient Cloud Flows with Power Automate

Agnius Bartninkas | COO @ Definra

The guy in front of you

Agnius Bartninkas

COO and Co-Founder @ **Definra**

- Microsoft Business Applications MVP (Power Automate)
- Used to be the Excel guru of my office, but now am the grumpy guy telling everyone to stop using Excel
- The most experienced PAD (formerly Softomotive) user in Lithuania. Most other users were trained by me
- A “certified” beer expert. Now an aspiring cocktail maker.

www.linkedin.com/in/agnius-bartninkas

Microsoft® Most Valuable Professional

Agenda for today

Some “theory”:

- Types of error handling available
- Expressions to use
- The issues with some of those expressions

Demos:

- The “Try, Catch, Finally” approach
- A basic approach for simple flows
- Handling individual items in loops
- A branching approach

An alternative method using the Flow Run Dataverse table

Q&A

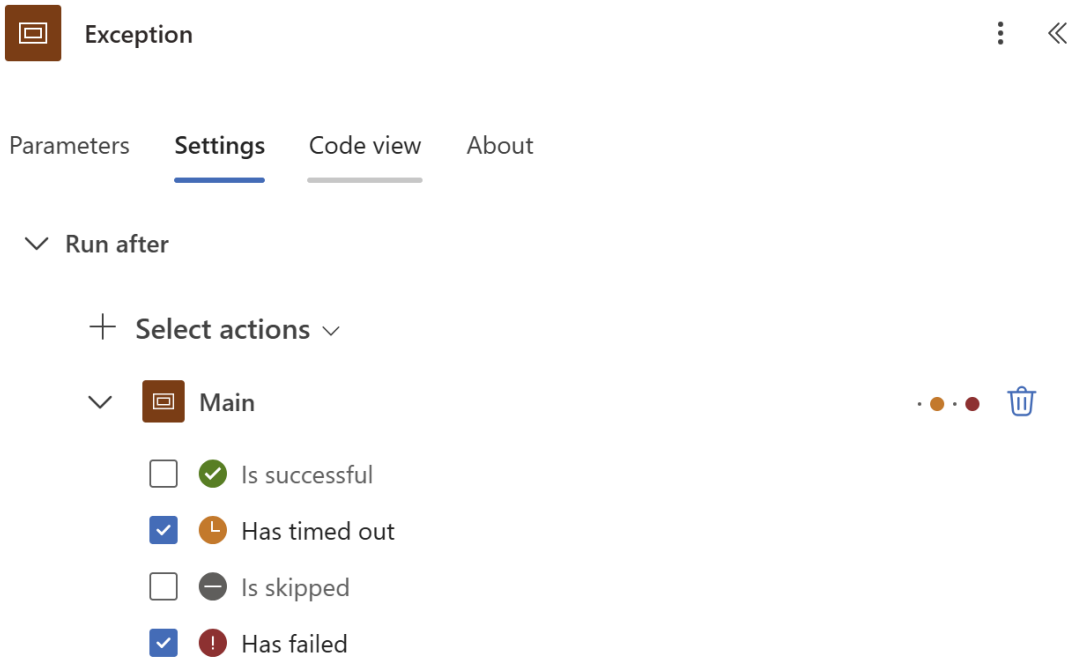
Error Handling

Handling errors is key to all automations, especially anything running in an enterprise environment.

But with Power Automate being a tool that is marketed as extremely citizen-developer-friendly, **lots of users don't know how to handle errors - or that they need to do it at all.**

Run-after Settings

Handling errors in PA is done via the “run after settings” on the following action.



Types of Error Handling

DEFINRA

The main options on how we can handle errors in Power Automate flows are:

- **None** – no error handling, meaning the flow fails on any error (not recommended).
- **Action level** – applying rules after every action in the flow that can possibly fail (not recommended).
- **Containers** – grouping actions within a container (scope, condition, loop) and applying error handling to the entire container (recommended).

The Cool Expressions

To capture an error and get its details, we need to use one of the two expressions:

- **actions()** – for action level error handling
- **result()** – for containers, such as scopes, conditions or loops


```
"outputs": {
    "statusCode": 400,
    "headers": {
        "Cache-Control": "max-age=0, private",
        "Vary": "Origin",
        "X-NetworkStatistics": "0,525568,0,0,58,0,23725",
        "X-SharePointHealthScore": "3",
        "X-MS-SPConnector": "1",
        "X-SP-SERVERSTATE": "ReadOnly=0",
        "DATASERVICEVERSION": "3.0",
        "SPClientServiceRequestDuration": "50",
        "SPRequestDuration": "51",
        "X-DataBoundary": "EU",
        "X-1DSCollectorUrl": "https://eu-mobile.events.data.microsoft.com/OneCo
        "X-AriaCollectorURL": "https://eu-mobile.events.data.microsoft.com/Coll
        "SPRequestGuid": "c4901068-ce9f-46a5-8eeb-a492dd6e9eef",
        "request-id": "c4901068-ce9f-46a5-8eeb-a492dd6e9eef",
        "MS-CV": "aBCQxJ/OpUaO66SS3W6e7w.0",
        "Strict-Transport-Security": "max-age=31536000",
        "X-Frame-Options": "SAMEORIGIN",
        "Content-Security-Policy": "frame-ancestors 'self' teams.microsoft.com
        "MicrosoftSharePointTeamServices": "16.0.0.24524",
        "X-Content-Type-Options": "nosniff",
        "X-MS-InvokeApp": "1; RequireReadOnly",
        "P3P": "CP=\"ALL IND DSP COR ADM CONo CUR CUSo IVAo IVDo PSA PSD TAI TE
        "X-AspNet-Version": "4.0.30319",
        "X-Powered-By": "ASP.NET",
        "Timing-Allow-Origin": "*",
        "x-ms-apihub-cached-response": "false",
        "x-ms-apihub-obo": "false",
        "Date": "Thu, 15 Feb 2024 16:32:28 GMT",
        "Content-Length": "230",
        "Content-Type": "application/json",
        "Expires": "Wed, 31 Jan 2024 16:32:28 GMT",
        "Last-Modified": "Thu, 15 Feb 2024 16:32:28 GMT"
    },
    "body": {
        "status": 400,
        "message": "Column 'Available' does not exist. It may have been deleted
    }
},
```

The Not-So-Cool Part of These Expressions

There are at least four different output schemas of the output depending on the actions* they come from:

Type 1: error.message (e.g Compose, Send an Email (V2))

Type 2: outputs.body.message (e.g Send an HTTP request to SharePoint, Get items)

Type 3: outputs.body.error.innerError.message (e.g Create item)

Type 4: outputs.body.error.message (e.g. List rows in Dataverse table)


```
    "error": {
        "code": "OpenApiOperationParameterTypeConversionFailed",
        "message": "The 'inputs.parameters' of workflow operation 'Send_an_ema
    }
```

```
"body": {
    "status": 400,
    "message": "Column 'Available' does not exist. It may have been deleted by
}
```

```
"body": {
    "error": {
        "code": "0x0",
        "message": "Could not find a property named 'Available' on type 'Mic
    }
}
```

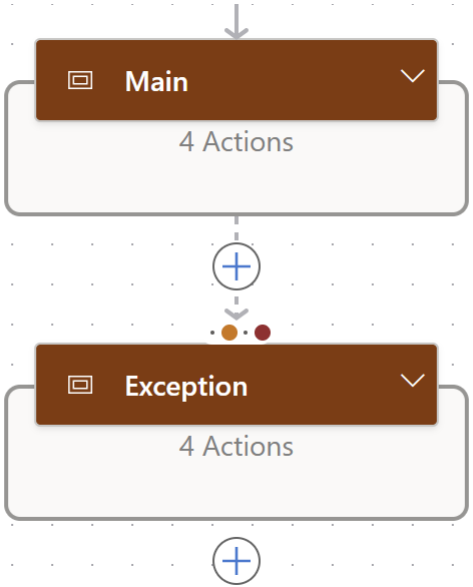

*Shoutout to David Wyatt for testing and finding the first three. He blogs about Power Automate and shares cool content here: https://dev.to/wyattdave

Scopes

Using *result()* in scopes is much easier, as we can group several actions into a single scope and have one expression to capture any errors inside it.

However, there are also some exceptions to this, too:

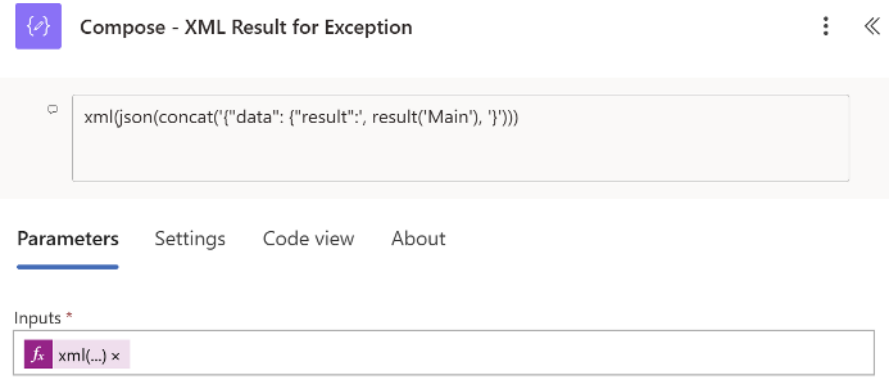
- It works best when no nesting of containers applies
- Loops have a different schema

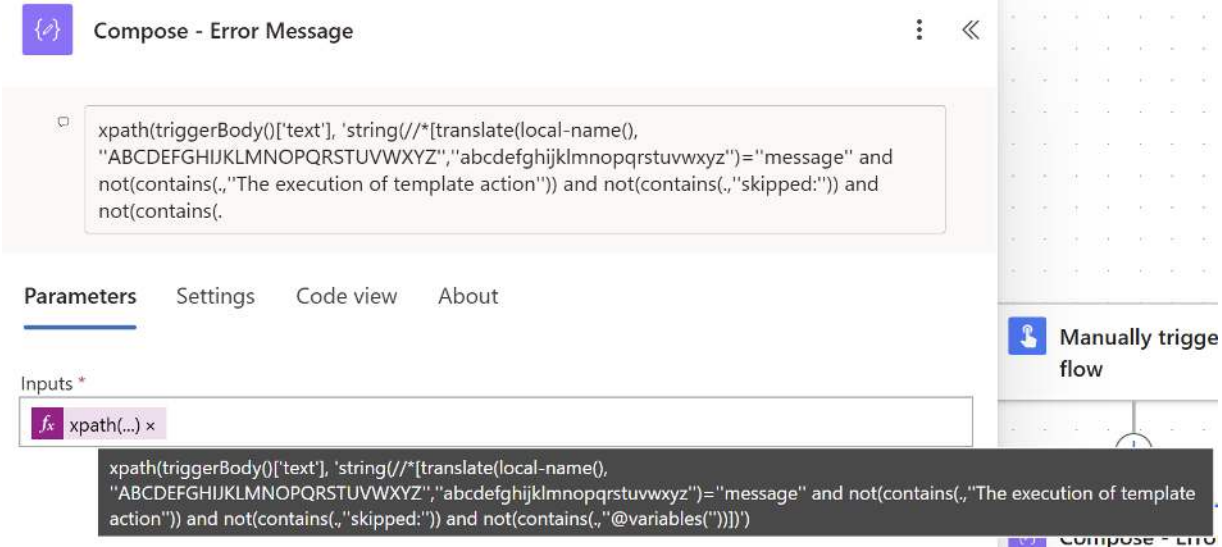


Getting the Error Details

Because of the different schemas, parsing the JSON object becomes tricky and difficult to maintain.

The best approach is thus to convert it to XML and use an Xpath expression to target the relevant node, instead of the full schema.






Demo
Time!

Key Takeaways

The best way to handle errors in PA flows is essentially following these guidelines:

- Handle containers using *result()* instead of actions using *actions()*.
- Wrap the entire flow into a Scope (e.g. ‘Main’).
- Have minimum nesting. Try avoiding loops and conditions inside scopes where possible.
- If you do have nesting, handle the exception at each container level - i.e. if you have a condition or a loop inside a scope, handle the exception for the nested container inside the scope, too.
- Send an email with error details and the flow run link.
- Terminate after handling.

The Main Expressions Used

Converting the result to XML:
xml(json(concat('{"data": {"result":', result('Main'), '}')))

Getting the error details:
xpath(triggerBody()['text'], 'string(//[translate(local-name(), "ABCDEFGHIJKLMNOPQRSTUVWXYZ", "abcdefghijklmnopqrstuvwxyz")="message" and not(contains(.,"The execution of template action")) and not(contains(.,"skipped:")) and not(contains(.,"@variables"))])')*

Getting the flow run URL:
concat('https://make.powerautomate.com/manage/environments/', workflow()?['tags']?['environmentName'], '/flows/', workflow()?['name'], '/runs/', workflow()?['run']['name'])

Before We Wrap-up – The Alternative Way

For simple flows where you don't want them to recover automatically, but just need some reporting on what happened, there is now a new "Flow Run" table available in the Dataverse.

Tables > Flow Run

Table properties — Properties, Tools

Name	Primary column	Description
Flow Run	Name	
Type	**Last modified**	
Elastic	6 months ago	

Schema: Columns, Relationships, Keys

Data experiences: Forms, Views, Charts, Dashboards

Customizations: Business rules, Commands

Flow Run columns and data — Update forms and views, Edit

Workflow	Created On	Error code	Error message	FlowRunId *	Duration	+26 more
Demo - Error Handling: Basic	11/5/2024 1:13 PM	Terminated	{ "code": "500", "message": "Column 'True' does n...	f75c147a-691e-0839-84cc-cdeeea...	8,539	
Demo - Error Handling: Basic + Lo...	11/5/2024 1:12 PM			7dee5e3f-01be-d76a-f613-fcb389...	17,837	
Log to SharePoint	11/5/2024 1:12 PM			b11b8361-90ab-38fe-0f4b-37361...	1,033	
Log to Dataverse	11/5/2024 1:12 PM			77344627-17a9-d57a-55ec-99cba...	4,091	
ParseError	11/5/2024 1:08 PM			b2c356b2-e32c-fd5d-e024-f7ffa2...	237	
AI_Builder_Demo Invoice Processor	10/22/2024 11:55 AM			909d8257-3adc-ebfe-15a6-95e83...	473,869	
AI_Builder_Demo Invoice Processor	10/22/2024 11:48 AM	Terminated		9042c001-2a6c-1f77-face-e89bee...	132,597	
AI_Builder_Demo Invoice Processor	10/21/2024 3:22 PM	Terminated		5bb9d831-8f7e-148c-0548-ed123...	177,957	
Select lookup		*Enter text*	*Enter text*		*Enter number*	

DEFINRA

Let’s chat more?

Rate the session

www.linkedin.com/in/agnius-bartninkas