



nordic.
summit

Error-Proof Your Automations: A Deep Dive into Advanced Error Handling and Logging in Power Automate Desktop

DIAMOND SPONSOR



The Digital
Neighborhood

PLATINUM SPONSOR



Microsoft

GOLD SPONSORS



SILVER SPONSORS



BRONZE SPONSORS



Agnius Bartninkas

COO @ DEFINRA

The most experienced PAD (formerly *Softomotive*) user in Lithuania. Most other users were trained by me.

Also, a “certified” beer expert.



ab@robovirgin.com



<https://www.linkedin.com/in/agnius-bartninkas/>



Not on X



<https://robovirgin.com/>



Agenda

- Some “theory”
 - Types of error handling available in PAD
 - Options available for logging errors
- Demos
 - Different ways to handle errors
 - The different options for logging error messages
 - Logging runtime non-error messages
 - Taking screenshots on error
 - Monitoring logs in ELK
- Q&A

Types of error handling

The main options on how to handle errors in PAD are:

- None – the flow fails on any error (not recommended)
- Action level – special rules set on each action (not recommended, except for special cases)
- Error blocks – setting rules for blocks of actions or entire sub-flows (recommended)

Action level error handling

+ Very powerful, lots of rules

- Time-consuming to build and maintain
- Does not apply to all actions
- Cannot handle unexpected logic errors

The screenshot shows the 'Execute SQL statement' configuration window. At the top, it indicates that the following rules will apply if the action fails. The 'Retry policy' is set to 'None'. Under the 'All errors' section, there are three rules defined: 1) 'Run subflow' with 'GetErrorMessage' as the action; 2) 'Variable' 'Log_Message' with a value of '{x}' mapped to '%ErrorMessage%' with a value of '{x}'; 3) 'Run subflow' with 'LogToFile' as the action. At the bottom of this section, there are two buttons: 'Continue flow run' and 'Throw error'. The 'Advanced' section lists three error types: 'Can't connect to data source', 'Invalid connection string', and 'Error in SQL statement', each with 'New rule' and 'Clear all' options. At the bottom of the window, there are three buttons: 'Return to parameters', 'Save', and 'Cancel'.

Error blocks

- + Very powerful
- + Easy to set up and maintain
- + Can handle all exceptions
- Less options
- Same rules apply to all actions

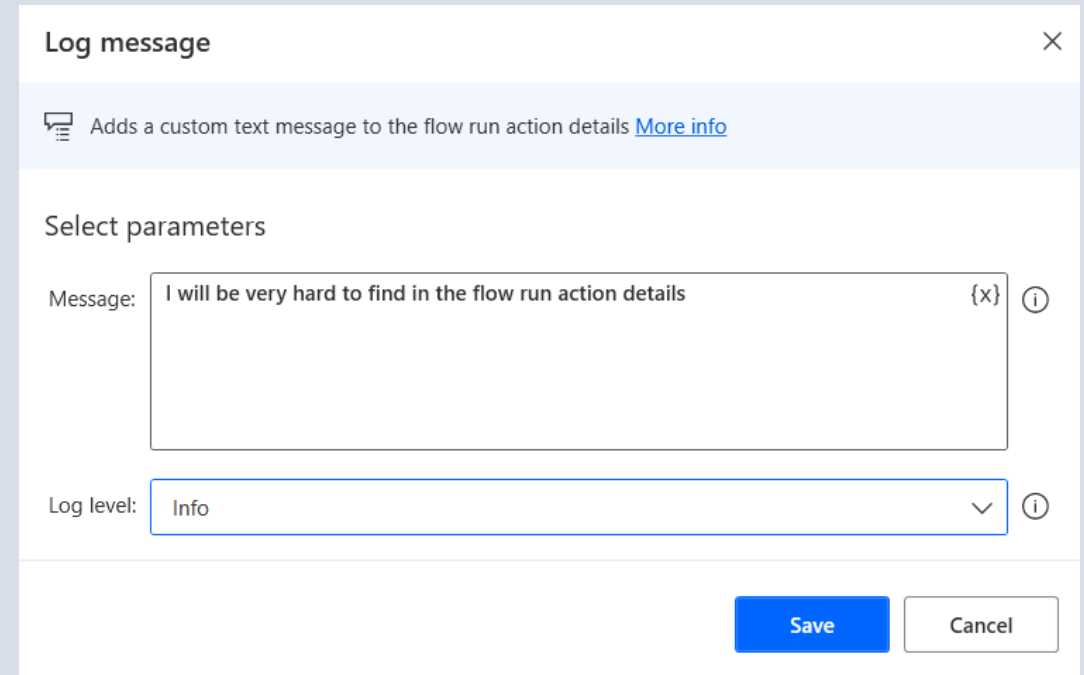
The screenshot shows the 'On block error' configuration dialog. At the top, there is a title bar with a close button. Below it, a shield icon indicates that the block marks the beginning of an error handling block, with a 'More info' link. The 'Select parameters' section includes a 'Name' field containing 'No error shall pass'. Below this is a rule configuration area with a '+ New rule' button and a 'Clear all' button. The rule configuration includes a 'Variable' field with 'Log_Message' selected, a 'to' field with 'TRACE' selected, and a 'Run subflow' dropdown menu with 'LogToFile' selected. There are two radio buttons for 'Continue flow run' (selected) and 'Throw error'. Below these is an 'Exception handling mode' dropdown menu with 'Go to end of block' selected. At the bottom, there is a 'Capture unexpected logic errors' toggle switch which is turned on, and 'Save' and 'Cancel' buttons.

Log messages

Handling errors is cool, but if there are no logs, it will usually be quite hard to find the issue - and let alone fix it.

Currently, PAD supports the **Log message** action.

While **Log message** is better than no logging, pretty much any custom logging alternative is better than using **Log message**.



The screenshot shows a 'Log message' dialog box with a close button (X) in the top right corner. Below the title bar, there is a light blue header with a speech bubble icon and the text 'Adds a custom text message to the flow run action details [More info](#)'. The main area is titled 'Select parameters' and contains two fields: 'Message:' with a text input field containing 'I will be very hard to find in the flow run action details' and a character count '{x}' and an information icon (i); and 'Log level:' with a dropdown menu set to 'Info' and an information icon (i). At the bottom right, there are two buttons: 'Save' (blue) and 'Cancel' (white).

Logging errors

Some options that I've seen:

- Logging to SQL databases – great for techies, might be harder to use for non-tech people.
- Logging to SharePoint lists – better for non-tech people
- Logging to CSV files – easy to implement, but a bit limited
- Logging to plain text files – very easy to implement and easy to use

Logging to plain text files

The reason I like plain text files:

- They're very easy to set up in PAD
- **Write text to file** is much less likely to fail due to application errors.

The one limitation of logging to local files over SQL databases and SharePoint lists are the fact that they're local to the machine (unless we push them to some network location).



On block error at sub-flow level + Get last error

On block error

Marks the beginning of a block to handle actions errors [More info](#)

Select parameters

Name:

Variable {x} to {x}

Run subflow

Exception handling mode

Capture unexpected logic errors

ExecuteWebFlow	LogToFile	ConvertPrice	TakeScreenshot	Config
----------------	------------------	--------------	----------------	--------

```

If ( Log_MessageList [0] ='TRACE') =True then
  {x} Set variable
  Assign to variable Log_Level the value Log_MessageList [0]
  {x} Get last error
  Get the last error that occurred and store it into Log_Message and clear the error value.
Else if Log_MessageList .Count <2 then
  {x} Set variable
  Assign to variable Log_Level the value 'WARN'
  Set variable
  Assign to variable Log_Message the value 'Found the list of expected message count to be less than 2. Car
  correctly. Original message: ' Log_MessageList
Else

```

External Config for logger object, screenshots path, etc.


```
"WorkItemProcessor": {
  "PreviousModule": "WorkItemGenerator",
  "MaxErrorCount": {
    "Excel": 10,
    "Web": 10,
    "Axapta": 10,
    "WorkItem": 3
  },
  "Logger": {
    "LogLevels": "TRACE,DEBUG,INFO,WARN,ERROR,FATAL",
    "ErrorLogLevels": "ERROR,FATAL"
  },
  "Environment": "DEV",
  "URL": {
    "Main": "https://www.lego.com/",
    "Base": "https://www.lego.com/en-lt/product/{ProductNumber}"
  },
  "ScreenshotsPath": "C:\\RPA\\NordicSummit\\Screenshots",
  "Recipient": {
    "Error": "ab@robovirgin.com",
    "Info": "",
    "Success": "",
    "Skip": "",
    "Fail": "",
    "Default": "ab@robovirgin.com"
  }
}
```


Log file per day per flow (+ per machine optionally)








{X} **Get Windows environment variable**
Retrieve the value of environment variable 'COMPUTERNAME' and store it into `ComputerName`

{X} **Set variable**
Assign to variable `Log_File` the value `LogsDirectory \"{Date}_{ModuleName}_{Machine}.log'`

 **Replace text**
Replace text '{Date}' with `DateISO8601` in `Log_File` and store the result into `Log_File`

 **Replace text**
Replace text '{ModuleName}' with `ModuleName` in `Log_File` and store the result into `Log_File`

 **Replace text**
Replace text '{Machine}' with `ComputerName` in `Log_File` and store the result into `Log_File`

Name	Date modified	Type	Size
 2023-08-29_WorkItemGenerator_LAPTOP-55SPC6R1.log	2023-08-29 12:02	Log file Source File	35 KB
 2023-09-20_WorkItemGenerator_LAPTOP-55SPC6R1.log	2023-09-20 15:09	Log file Source File	45 KB
 2023-09-20_WorkItemProcessor_LAPTOP-55SPC6R1.log	2023-09-20 15:36	Log file Source File	483 KB
 2023-09-21_WorkItemGenerator_LAPTOP-55SPC6R1.log	2023-09-21 21:31	Log file Source File	43 KB
 2023-09-21_WorkItemProcessor_LAPTOP-55SPC6R1.log	2023-09-21 21:55	Log file Source File	337 KB
 2023-09-22_WorkItemGenerator_LAPTOP-55SPC6R1.log	2023-09-22 11:21	Log file Source File	43 KB
 2023-09-22_WorkItemProcessor_LAPTOP-55SPC6R1.log	2023-09-22 11:44	Log file Source File	138 KB



nordic.
summit

Q&A



nordic
summit

THANK YOU!