Data Audit and Cleanse Power Automate Job

Technical Documentation

Table of Contents

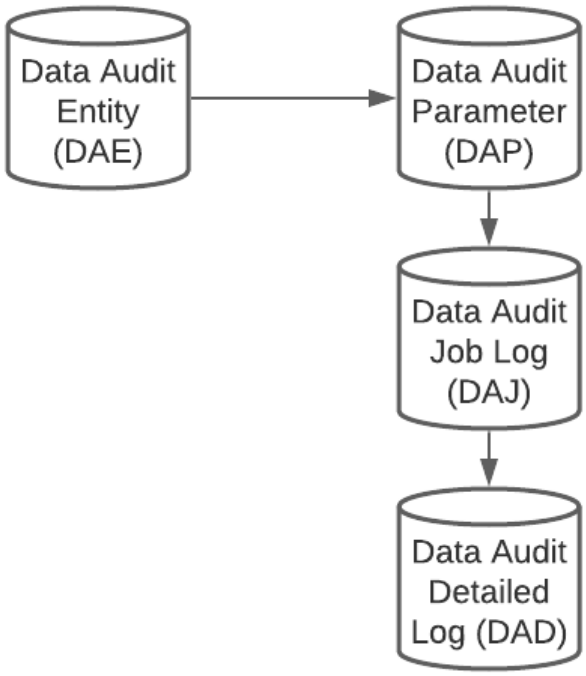
Overview

This document describes the technical processing flow of the FlowMergeDataAuditFetchCase Power Automate job. This flow loops through a series of (DAP) records which each run a query against the CRM database. These queries each return rows that need to be acted upon and represent either data anomalies or stale data. If anomalies are found, (DAD) records and a Case are created, and assigned to Tier 1 support to resolve.

Data Audit Architecture

Data Model

The Data Audit process uses four custom entities within CRM to define and record the results of the jobs.



The DAP entity is the “driving table”. One record contains one “Data Audit job specification” in the form of Advanced Find XML (or SQL – future)

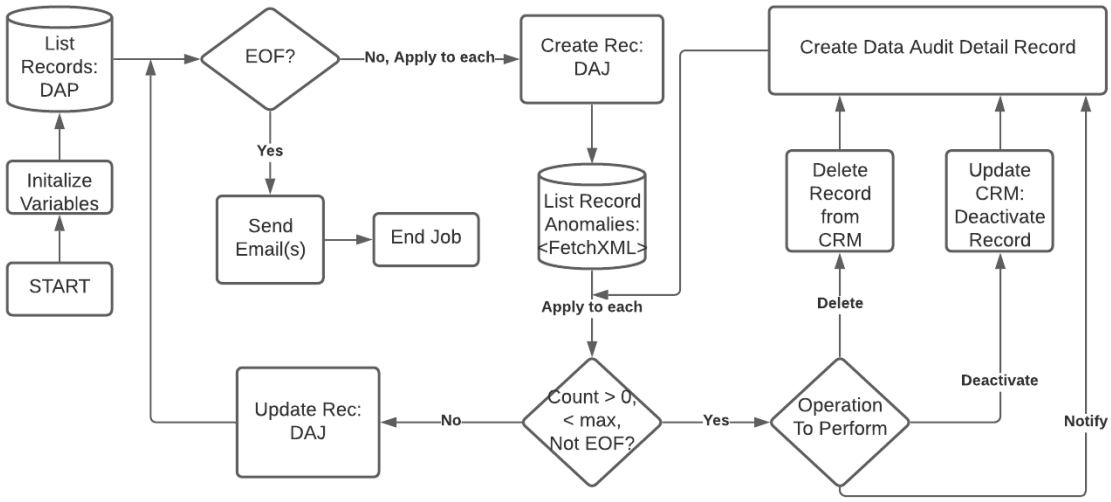
The DAE entity contains fields that enable the Power Automate job to generically process requests for different entities.

Each time a DAP job is run one DAJ log will be created.

For each data anomaly, or record to be deactivated or deleted, a DAD record is created.

Data Audit High-Level Flow Diagram

A high-level flow diagram of this Power Automate flow shows two loops: one for each record in the DAP entity, and one for each record returned by the Query from the DAP record. (Note: in version two of this job we added the ability to process a SQL query. That does not modify the process flow (the SQL query resides in the same location as the "List Record Anomalies: Fetch <XML>" step.) Due to a MSFT bug with solution promotion, we have removed the ability to process a SQL query.



The major sections of the Data Audit flow are diagrammed above and listed in the table below. Following this table, each major flow step is explained via screenshots from the Power Automate job.

Data Audit Major Flow Steps

These steps are defined in further detail in the pages below; this chart identifies the most important steps.

Step#	***Flow Label***	***Purpose***
1	Initialize Variables	Must initialize variables to use later in the program.
2	List Records: DAP	Queries the D365 CRM "Data Audit Parameter" (DAP) table. Each record in this table identifies the query and operation to perform on the result set. Only those that should run today are returned, based on the Last Run Date and Schedule.
3	Apply to each	Each record returned in the "List Records: DAP" result set is processed in a loop.
4	Create DAJ Rec	Each time a DAP record is processed, a "Data Audit Job Log" record is produced to capture the results while processing the DAP Query.
5	Condition: Query Method	New in version two, this determines whether or not a SQL or Fetch XML query will be run. The syntax for executing a SQL query is different than processing Fetch XML.
6	List Records: Anomalies	The Query String field in the DAP table contains the FetchXML from an advanced find query in CRM. This identifies records of interest, or data anomalies. (The original purpose of this job was to detect record anomalies but can now also be used to cleanse data.)
7	Apply to each	Each record returned in the "List Records: Anomalies" result set is processed in a loop.
~~8~~	~~List SQL Records: Anomalies~~	~~The Query String field in the DAP table contains the FetchXML from an advanced find query in CRM. This identifies records of interest, or data anomalies. (The original purpose of this job was to detect record anomalies but can now also be used to cleanse data.)~~
~~9~~	~~Apply to each SQL~~	~~Each record returned in the "List Records: Anomalies" result set is processed in a loop.~~
10	Operation to Perform	Each DAP record identifies the operation to perform on each records from the "List Records: Anomalies" result set. The operations are: Delete (a record from CRM), Deactivate (a record from CRM) or Notify (creates a DAD record for users to manually review.)
11	Create DAD Record	Each record returned in the "List Records: Anomalies" query will cause a new "Data Audit Detail" (DAD) record to be created. For Delete and Deactivate Operations, an inactive DAD record is created, to record the transaction results. For Notify Operations the DAD record will store a link to the CRM record and identify the reason the record is here (account 'out of business', e.g.), which will enable a user to fix the data anomaly manually.
12	Update Rec: DAJ	The record created in step #4 is now updated with the results (#records created, updated, deleted, timing and error count) of processing the query. (The job then processes the next DAP record, from #3 above.)
13	Send Email(s)	When the last DAP record has been processed, an email is generated and sent to me with the results. Emails will also be generated to notify users responsible for correcting the data anomalies.

Data Audit Detailed Steps

Each major step in the Data Audit flow is expanded upon in the pages below.

Initialize Variables

All Power Automate jobs must begin with the initialization of variables. The variables used in this flow and their usage is shown in the table below.

Variable Name	***Usage/ Definition/ Example***	***Where Set***	***To What Value***	***Used In***
DAP GUID	DAP GUID	DAP Loop	@{items('Apply_to_each_DAP_record')?['bhn_dataauditparameterid']}	
DAP Count	Counts #DAP Recs to be processed	After List Records: Get DAP	# of DAP recs	Condition > 0, proceed
Entity Singular Name	'contact'	DAP Loop	@{items('Apply_to_each_DAP_record')?['aliasBHN_Entity.bhn_singularentityname']}	DAD Rec create
Plural Entity Name	'contacts'	DAP Loop	@{items('Apply_to_each_DAP_record')?['aliasBHN_Entity.bhn_pluralentityname']}	Compose4; "**ListRecordsanomalies**"
EntityNumberFieldName	Accountnumber	DAP Loop	@{items('Apply_to_each_DAP_record')?['aliasBHN_Entity.bhn_fieldnameforentitynumber']}	
EntityNameFieldName	'fullname'	DAP Loop	@{items('Apply_to_each_DAP_record')?['aliasBHN_Entity.bhn_fieldnameforentitynamevalue']}	
Entity GUID field name	'accountid'	DAP Loop	@{items('Apply_to_each_DAP_record')?['aliasBHN_Entity.bhn_fieldnameforguid']}	
Entity Number	ACT-#	Anomaly Loop	`items('Apply_to_each_data_anomaly')?[variables('EntityNumberFieldName')]`	DAD Rec create
Entity GUID	(32-char unique id)		`Items('Apply_to_each_data_anomaly')?[variables('Entity GUID field name')]`	DAD Rec create

Variable Name	***Usage/ Definition/ Example***	***Where Set***	***To What Value***	***Used In***
Anomaly Record Count	How many records were returned by the DAP Query?		`length(body('Listrecordsanomalies')?['value'])`	Compared to Max Count; Re-initialized @ end of DAP loop
Anomaly Entity Name value	'Contact'		`Items('Apply_to_each_data_anomaly')?[variables('EntityNameFieldName')]`	DAD Rec create
DAD Create Count	How many DAD records were created in this pass?	Anomaly Loop/Switch Condition	+ 1 if "DAD In Process Recs Exists Count" = 0	DAJ Rec create; Re-initialized @ end of DAP loop
DAD Update Count	How many DAD records were updated (done if this anomaly has already been reported)	Anomaly Loop/Switch/ Condition	+ 1 if "DAD In Process Recs Exists Count" > 0	DAJ Rec create; Re-initialized @ end of DAP loop
DAD In Process Recs Exist Count	If > 0, then a DAD record already exists, so update	Anomaly Loop/Switch (Notify)	`length(body('List_records')?['value'])`	If 0, will update a Notify action
Deactivate Record Count	Number of CRM records deactivated	Anomaly Loop/Switch (Deactivate)	+ 1 each time through this path of the switch	
Delete Record Count	Number of CRM records deleted	Anomaly Loop/Switch (Delete)	+ 1 each time through this path of the switch	
Notify Record Count		Anomaly Loop/Switch (Notify)	+ 1 each time through this path of the switch	

Variable Name	***Usage/ Definition/ Example***	***Where Set***	***To What Value***	***Used In***
DADstatecode	The CRM DAD table statecode value (option set number).	Anomaly Loop/Switch	(In Switch, set to valid statecode for entity and operation.) Stored on the DAE table. Operation Type of Notify creates an active DAD record. Delete and Deactivate create inactive DAD records.	
DADstatuscode	Status option set value the flow will set the DAD record to.	Anomaly Loop/Switch	(In Switch, set to valid statecode for entity and operation.) Stored on the DAE table.	
CRM BASE URL	Identifies which system we are in, DEV, UAT, or PROD		@{items('Apply_to_each_BHN_Configuration_(even_though_it_will_only_be_one_record)')?['bhn_value']}	List Rec for BHN Configs
Hotlink	clicking this will open the CRM record.	Anomaly Loop	@{variables('CRM BASE URL')}/main.aspx?pagetype=entityrecord&etn=@{variables('Entity Singular Name')}&id=@{variables('Entity GUID')}	DAD Rec create

Record ID		Anomaly Loop/Switch	{
			"statecode": 1,
			"statuscode": 2
			}	

Entity Disabled State Code		Init loop	1	
Entity Disabled Status Code		Init loop	2	
DAJ Message	Message written to a DAJ record			
DAJ Error Count	# of errors while processing this job.			

Variable Name	*Usage/ Definition/ Example*	*Where Set*	*To What Value*	*Used In*
HtmlTable	An array variable that captures the values from each DAP processed, sent in an email at end of job.			
Case Priority GUID	The Priority of the Case (GUID)			
Case Requestor GUID	The requestor (GUID) of the CASE created			
Case Category GUID	The Category of the CASE created			
Case Subcategory GUID	The Subcategory of the CASE created			
Case Account GUID	The dummy account linked to the CASE			

List Records: Get the CRM BASE URL from BHN Configuration

As the step says, the URL of the CRM system is stored in a CRM custom entity called "BHN Configuration". This step finds that URL. We don't know the GUID directly, so am doing a List records step that will return one row.

List Records: Get DAP

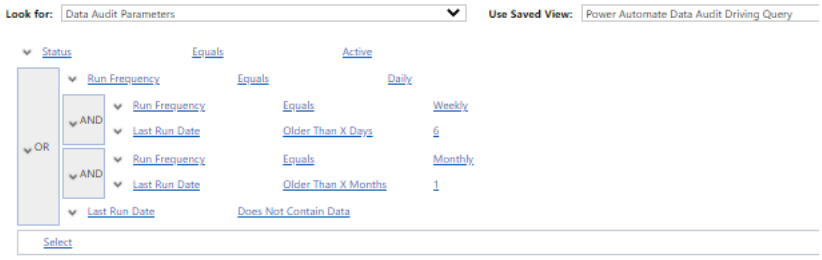
This List Records block finds all of the Data Audit jobs that are ready to be run in this pass. The data returned by this step is processed as the "outer loop".

The FetchXMLQuery is taken from the "Power Automate Data Audit Driving Query" advanced find view (see below).

The Expand Query gets field values from the "Data Audit Entity" table (linked N:1 from Data Audit Parameter to Data Audit Entity) that identifies the technical name for the primary key, status codes, and other values that vary by entity.

Power Automate Data Audit Driving Query

This query returns all jobs ready to run during the next execution of this Power Automate job. Essentially, if it is a Daily job, it is always ready to run. If it is a weekly job, it will run if the LastRunDate was a week ago. If a monthly job, it will run if the LastRunDate was a month ago.



z

Set Variable: DAP Count

This expression, in the "Set Variable: DAP Count" block

```
length(body('List_records:_Get_DAP')?['value'])
```

counts the number of records returned by the Driving Query above.

Condition: IF DAP Count gt 0, RUN Else STOP

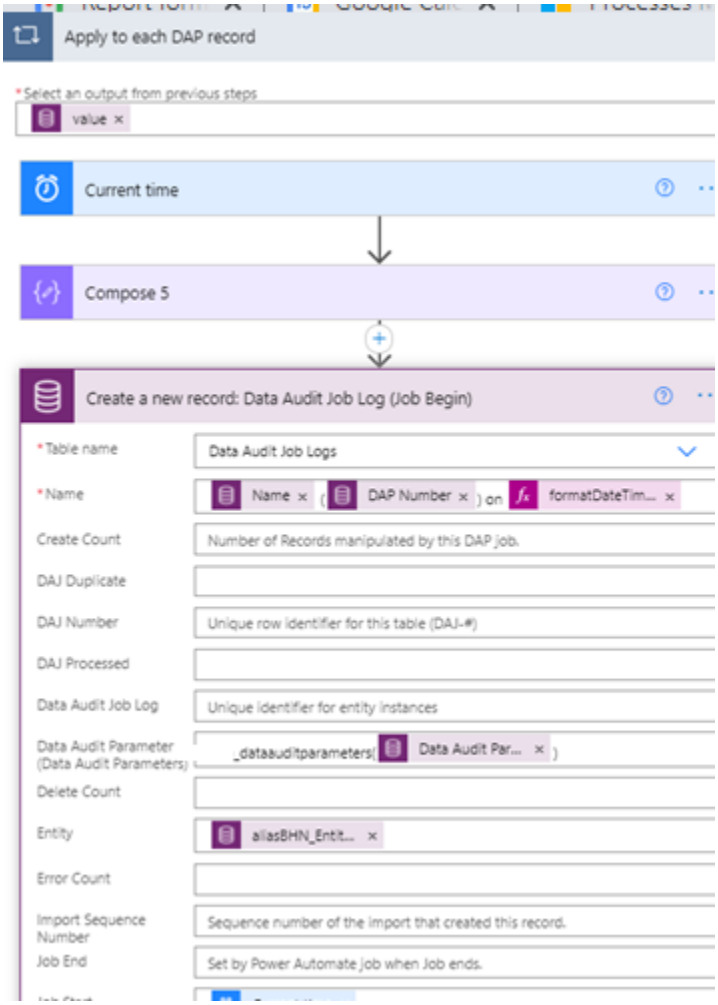
If there are no rows returned, the job stops.

Apply to each DAP Record

This is the first loop in this Power Automate job. Each DAP record (returned by the driving query) is processed one at a time. There are some key decision paths in this loop; this is where the bulk of the work is done.

Create DAJ Rec

The first step is to create the Data Audit Job Log record. One DAJ record exists each time a DAP is processed.



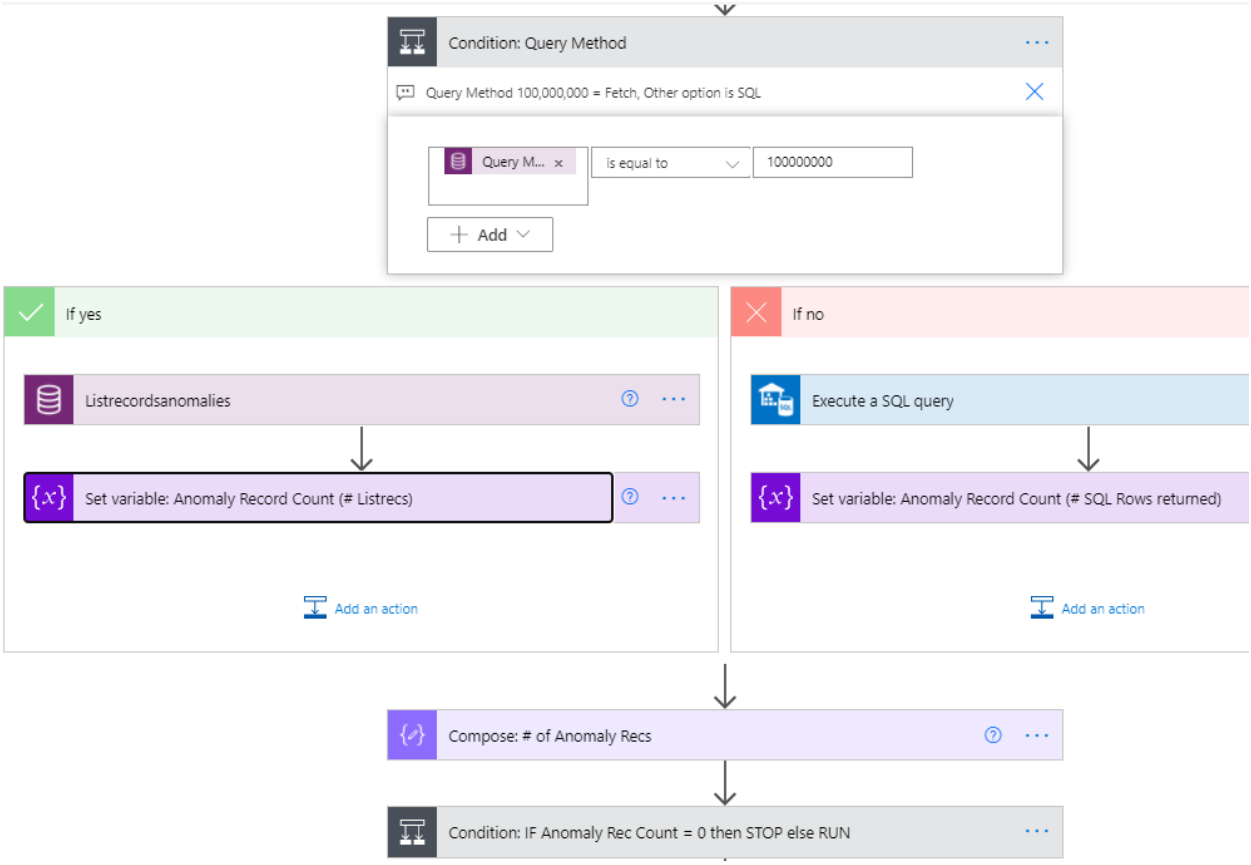
This ‘shell’ record will be updated after the DAP record is processed, identifying the number of records processed when running the query contained within the DAP record, and identifying any records created or errors logged.

Set variables for Entity fields

After the DAJ record is created, the field values from the Data Audit Entity linked to this DAP record are assigned to variables.

Condition: Query Method

In version 2, we added the ability to execute a SQL query to find anomalies. (Again, this was not implemented due to the MSFT solution bug.) We can now use advanced find Fetch XML, a SQL query or SQL stored procedure as the source for a DAP query. By using variables, we can reuse much of the logic and flow. This section of this document refers to this part of the flow:



Only two conditional/switch branches are required to:

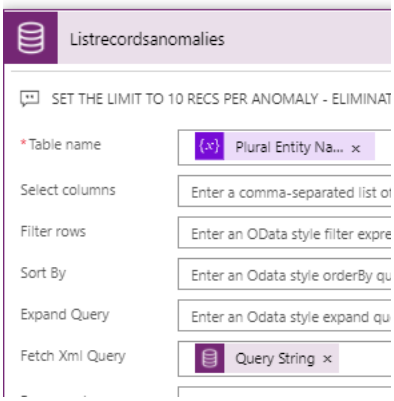
(a) execute a SQL query against Azure versus running a Fetch XML query against the CRM database, and
(b) process the results (different syntax, of course, depending on which action you use).

By defining variables to store the values of the fields returned by the queries, the rest of the flow can then use one set of variables.

This condition is to determine if we are executing a SQL query or a Fetch XML Query.

List Records: Anomalies

This step processes a Fetch XML query. The simplicity and power of this step is awesome. With just two variables, the entity that we are processing (the "Look for" entity in Advanced Find), and the query string (the Fetch XML from the Advanced Find), we can run any advanced find query against any entity (as long as things are defined properly in the DAP and DAE tables).



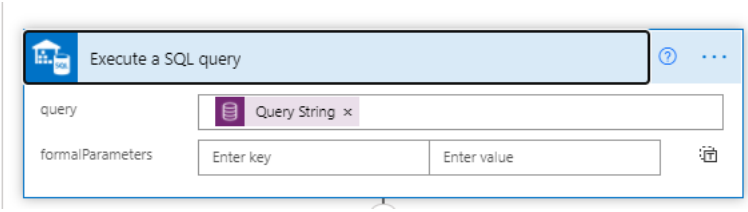
Set Variable: Anomaly Record Count

The FetchXML query syntax to count the number of records is:

```
length(body('Listrecordsanomalies')?['value'])
```

Execute a SQL query

This new step processes a SQL query. This is even easier than using Fetch XML: you simply pass the "Query String" variable, (which contains the Query String from the DAP, which is the SQL query) into the Query string in the SQL action block.



Note that we count the records returned by either query, and store the result in the same variable.

Set Variable: Anomaly Record Count

The SQL set variable is slightly different syntax:

```
length(body('Execute_a_SQL_query')?['resultsets']?['Table1'])
```

Condition: If Anomaly Recs count = 0 then Stop

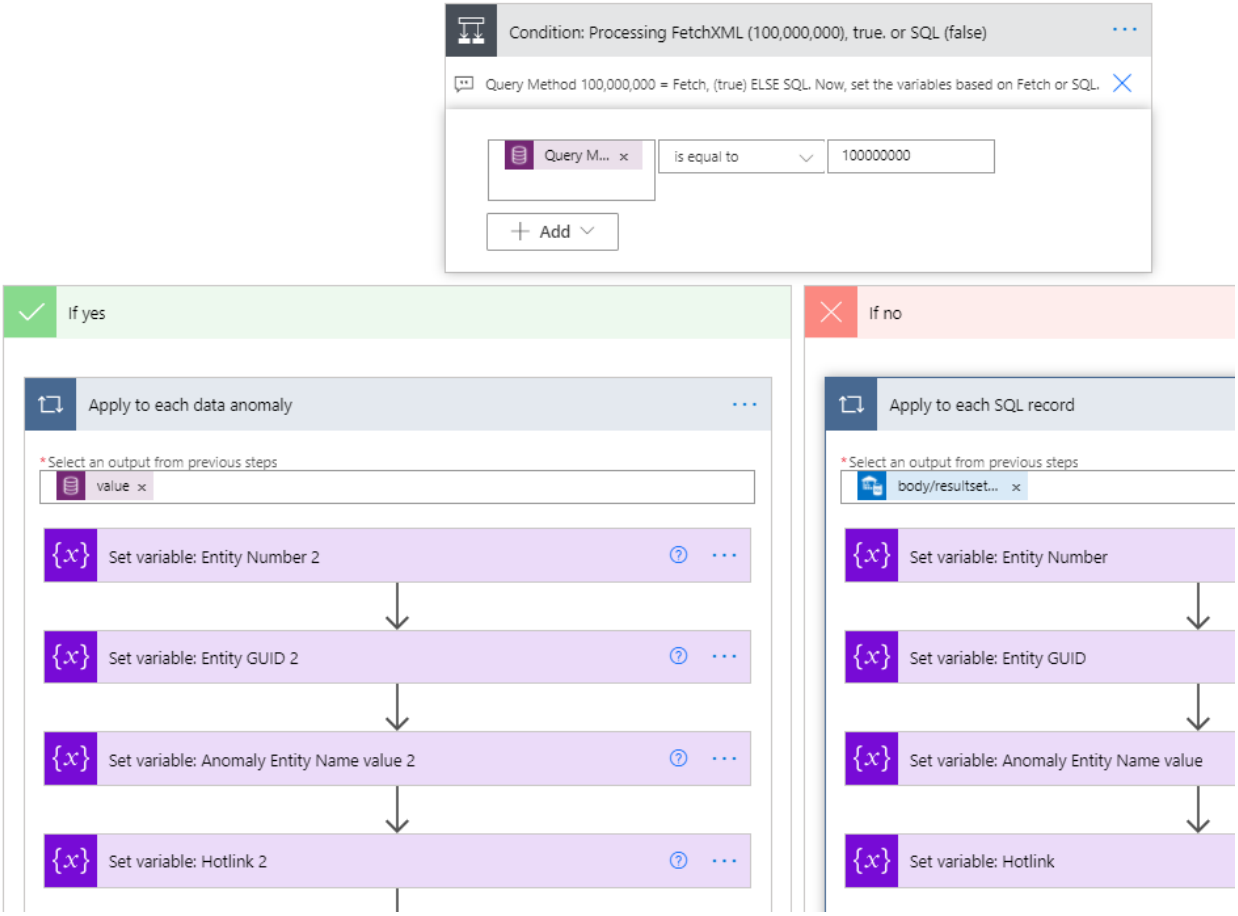
Again, after we process the query indicated by the DAP, we count how many records are returned. If 0, then we simply update the DAJ log, and exit this loop (which will cause us to process the next DAP record).

Condition: If Anomaly Recs count > Max then Stop

Similarly, if the count is greater than the "Max Count" field value from the DAP, then we do not process this job. (This is intended as a backstop: if a query was inadvertently changed, it may create (or worse, delete) many more records than intended.)

Condition: Processing Fetch XML or SQL?

This section of the document is the second branch that processing differently if the Query Method is FetchXML versus a SQL Query. This is because the way you refer to fields differs depending on data source.



This condition block evaluates the "Query Method" field value from this DAP record. If it is equal to Fetch, then we "Apply to each data anomaly", which is linked to the "Listrecordanomalies" block. If the Query Method is SQL, then we "Apply to each SQL record", which is linked to the "Execute a SQL Query" block.

Inside of each "Apply to each" block, we set variables to record the value of each column for each row of data returned. (Now that, regardless of query method, we have the value in a variable, the rest of the flow can just refer to that variable.)

Apply to each data anomaly

For every DAP that uses FetchXML as the query method, this loop processes the result set returned by executing the Fetch XML query against CRM data. For each record returned, the actions within this block are performed. The actions are a series of set variable statements that read one record from the Fetch XML query and populate the Entity # (ACT-#, e.g.), the Entity GUID, the Entity Name value, and the hyperlink for the record.

Apply to each SQL record

Similarly, for every DAP that uses SQL as the query method, this loop processes the result set returned by executing the SQL statement. against CRM data. For each record returned, the actions within this block are performed. The actions are a series of set variable statements that read one record from the Fetch XML query and populate the Entity # (ACT-#, e.g.), the Entity GUID, the Entity Name value, and the hyperlink for the record.

Note that now whichever query method was used, we have the variables populated that drive the remainder of the process.

Switch: Operation Type (affects CRM Data)

This switch looks at the value of the DAP Operation Type field. There are currently 3 options:

- Deactivate: The CRM record will be deactivated. (State and status code variables are set appropriately for the entity being processed.)
- Delete: The CRM record will be deleted
- Notify: A DAD record will be created that references (via a hotlink) the core CRM record.

The Delete branch has switches that (a) handle error trapping if the CRM deletion has issues (b) set a switch if we are deleting a DAD record to NOT create a DAD record letting us know (defeats the purpose of clearing out this DAD log table), and (c) sets a switch so that we do no write a DAD record unless we are in UAT or PROD.

(Note: options (b) and (c) re-use as existing variable “DAD In Process”. This variable was originally created to let the system know when an active DAD record for this DAP and GUID was already written, so that we don’t keep reporting the same problem. I am setting that same variable if the entity we are processing is the DAD entity, as we don’t want to write a DAD when deleting a DAD. And finally the variable is set if we are not in UAT or PROD, so that it will not write a DAD record in other environments.)

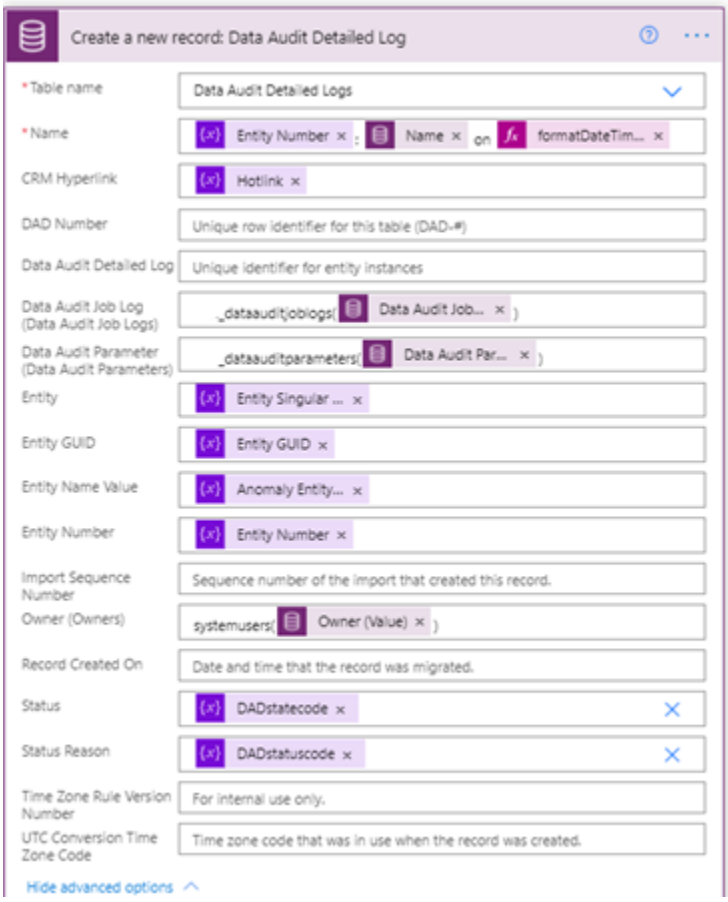
Condition: DAD In Process Recs Count

If this variable has been set to 0, then we create a new DAD record. Else (for conditions noted above), no DAD record is created. (We instead write a note to that record.)

Create DAD Record

Part of this closed loop process is that Tier 1 support will review and correct the data anomalies found by this Power Automate flow. This flow creates Cases and DAD records to record the details of the audit findings. One Case record is created for every non-zero DAJ record (created for each DAP record).

A DAD record is created for every data anomaly encountered in this job. For “Delete” and “Deactivate” operations, the DAD record is set to ‘deactivated’ status, as there is nothing for Tier 1 to do on these. (We are just record the deleted objects for tracking purposes.) For “Notify” operation types, a DAD record is created for each CRM record that met the conditions specified in the Query String in the DAP.



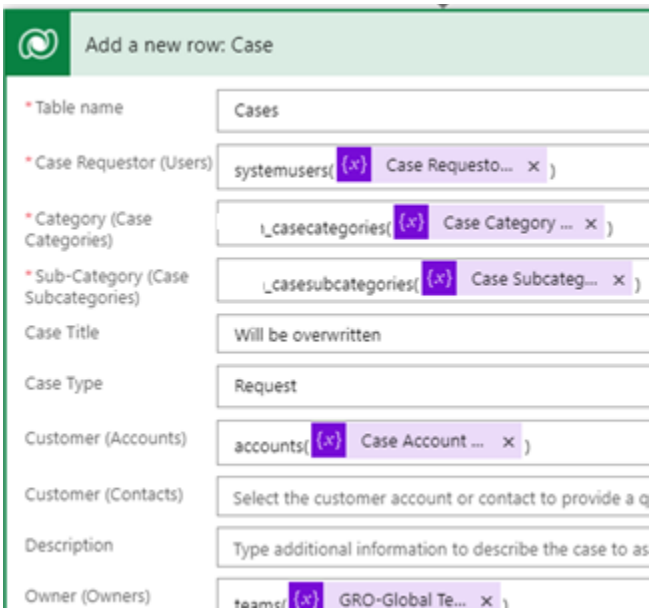
Update Rec: DAJ

After all anomaly records are processed, we update the Data Audit Job Log (DAJ) table with the counts of the records processed.

Condition: If Created DAD Rec (count gt 0) Create Case

If processing this DAP resulted in the creation of DAD records, a CRM Case is created. If not then no Case record is created (the right, or "No" branch).

If a DAD record was created, we find the Case Category, Subcategory, Priority, and Account to link this case to, and create the Case.



Update a record (DAP (Completion Time)

After processing each DAP, we update the DAP record with the completion time. That, along with the Run Requency field (daily, weekly, monthly) determine when this DAP next runs. Daily jobs run each day; weekly jobs run when the LastCompletedDate is a week old, and monthly jobs run when the LastCompletedDate is a month old.

We now loop through the next DAP.

Send Email

After all DAP records have been processed, an email is sent that displays the name of the job and the record counts (#created, #updated, #deleted, #deactivated) from all DAP jobs run during this pass.