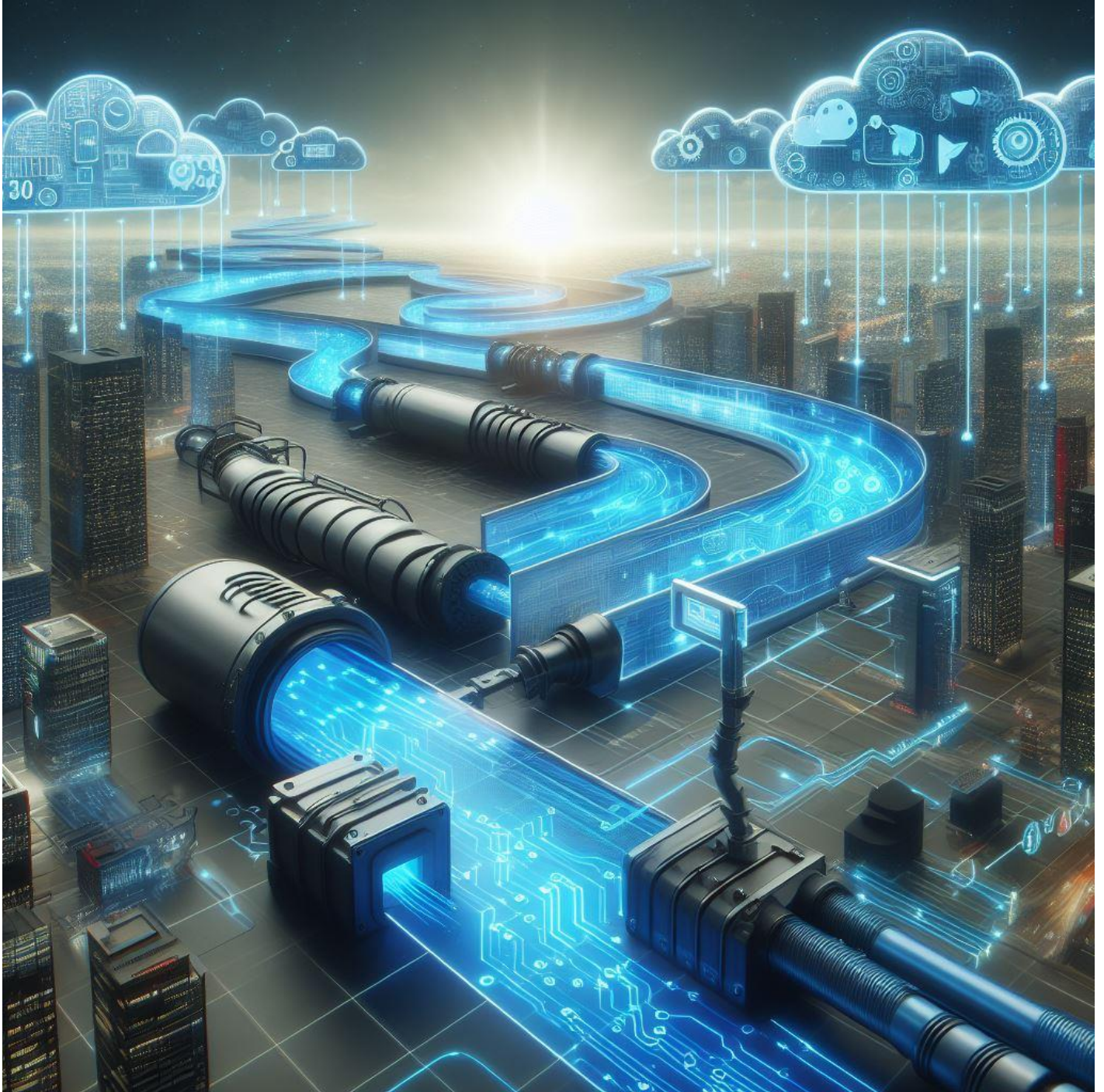


Knowledge Mining

Data Factory services, captures, and code



John Jemson – February 2014

Data Factory service: `adf-knowledge-mining-dev`

Contents

Introduction	4
Manage – Pre-pipeline processing.....	5
Author – Create your datasets.....	6
DS_AI_AnalyzeText	6
DS_AI_FormRecognizer.....	6
DS_ASQL_DataMining.....	6
DS_ASQL_LCS	7
Author – Create your Pipelines	7
PL_0_Master	8
PL_1STG_Ingest_LCS.....	8
Copy Child	8
Copy Significant Adults	8
Copy Case Worker.....	9
Copy Documents	9
Copy Case Notes	10
PL_2_1_ENT_Ingest_STG	10
PL_2_2_ENT_OCR_Documents.....	11
Lookup	12
ForEach activities	13
Web Post to AI service	13
On fail.....	14
On success.....	14
Get processing result	15
Update document content	15
PL_2_3_ENT_EntityRecognition_CaseNotes_async.....	17
Lookup case notes.....	17
ForEach activities	18
POST analyse-text	18
On fail.....	19
On success.....	20
Get processing result	20
Process results	20
On fail.....	21
PL_2_3_ENT_EntityRecognition_Documents_async	22
Lookup Documents	22
ForEach activities	23
POST analyse-text	23
On fail.....	24
On success.....	25

Get processing result	25
On fail	26
Appendix	27
SQL Queries	27
Copy Child - source	27
Copy Significant adults – source	27
Copy case worker – source	27
Copy documents – source	28
Copy case notes – source	29
Stored Procedures	30
sp_load_all	30
sp_load_casenote	30
sp_load_caseworker	31
sp_load_child	32
sp_load_document	33
sp_load_significantAdult	34
update_DocumentContent	35
update_DocumentProcessingError	35
process_childCasenoteResponse	35
update_CaseNoteEntityError	36
process_childDocumentResponse	36
update_DocumentEntityError	37
process_childCaseNoteResponse	37
Process_childDocumentResponse	38
Functions	38
fnCasenotesToEntityRecognise	38
fnDocumentsToEntityRecognise	39
splitString	39
getNetworkNodes	40
getNetworkNodesNoCaseNotes	41
Scrap book notes	44

Introduction

This document provides a brief overview of the process undertaken to create an end-to-end data pipeline for the Knowledge Mining project. The process ingests data from the lcs log ship clone, copied to Azure SQL database 'sqlldb-lcs-ehm' at sql-data-mining.database.windows.net within the Knowledge Mining resource group.

The process aims to extract child and associated case notes, documents and entities linked to children. The pipeline utilises Entity Recognition and Optical Character Recognition AI skills to mine for information unobtainable by traditional reporting methods. Required data is queried and written into 'sqlldb-data-mining' and mined data upserted into those records. The final output can be read by Power BI from the database and visualised.



Manage – Pre-pipeline processing

Linked Services

- | | | |
|------------------------|---|---|
| • LinkAIServices | - | REST service calling AI APIs |
| • LinkLcsSource | - | Azure SQL database link to Azure cloud SQL copy of db |
| • LinkSqlDataMiningOut | - | Azure SQL database link to target db for DF output |
| • LinkStLcsData | - | Link to storage account (file store) |

Integration runtime

- | | | |
|---|---|---|
| AutoResolveIntegrationRuntime | - | Azure built in for basic (public servicing) |
| IntegrationRuntime-adf-knowledge-mining-dev01 | - | Self-hosted, configured for our private network |

Global Parameters - Name the AI service key for reference and use the key here. This means the key need only be updated here rather than multiple instances through the pipelines when/should the key be changed.

Remember that document counts might not link up due to doc files types for example




Author – Create your datasets

DS_AI_AnalyzeText – These have been superseded and not used; however, they may be utilised in the future and have been retained. Used to create data sets for the results from the AI service. The result comes back in JSON.

To call and endpoint in Data Factory, there are two ways of doing this

- 1) Create a linked rest service with a dataset on it
- 2) Or create in the fly without linked service

This will depend on what you are doing with results from service. When using ‘copy data’ within Azure, these must be datasets.



REST
DS_AI_AnalyzeText

ConnectionParameters

Linked service *

LinkAIServices

Test connection

Integration runtime *

integrationRuntime-adf-knowledge-..

Edit

Base URL

https://cog-data-mining.cognitiveservice...


Relative URL

language/:analyze-text?api-version=202...

Preview data

language/:analyze-text?api-version=2022-05-01

DS_AI_FormRecognizer - (See above)



REST
DS_AI_FormRecognizer

ConnectionParameters

Linked service *

LinkAIServices

Test connection

Integration runtime *

integrationRuntime-adf-knowledge-..

Edit

Base URL

https://cog-data-mining.cognitiveservice...

Relative URL

formrecognizer/documentModels/prebui...


Preview data

formrecognizer/documentModels/prebuilt-read:analyze?api-version=2023-07-31

DS_ASQL_DataMining

Connection to target database when processing data. Currently this is a static clone of the LCS log ship taken in July 2023.

OFFICIAL




Azure SQL Database
DS_ASQL_DataMining

Connection

Schema


Parameters

Linked service *


 LinkSqlDataMiningOut

Test connection
 Edit
 + New
 [Learn more](#)

Integration runtime *


 integrationRuntime-adf-knowledge-..

Edit

Table

@dataset().TargetSchema


@dataset().TargetTable

Preview data

☒ Enter manually

DS_ASQL_LCS

Connection to source database when extracting data. Dbo.isperson is a pointer, SQL queries within the pipelines will extract the necessary data.




Azure SQL Database
DS_ASQL_LCS

Connection

Schema


Parameters

Linked service *


 LinkLcsSource

Test connection
 Edit
 + New
 [Learn more](#)

Integration runtime *


 integrationRuntime-adf-knowledge-..

Edit

Table

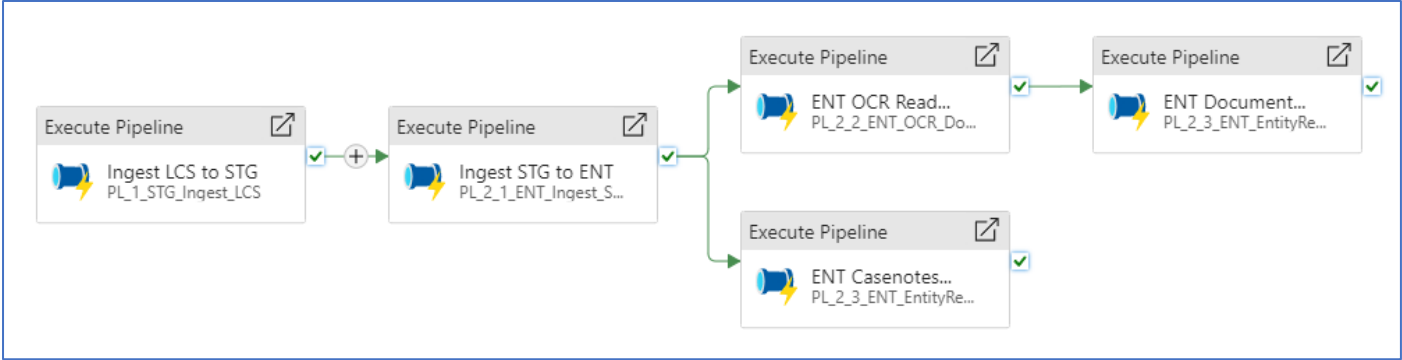
dbo.isperson

Refresh
 Preview data

☐ Enter manually


Author – Create your Pipelines


PL_0_Master – when triggered will run all pipelines in their sequence. Trigger when confident all processing can commence, otherwise, trigger the pipelines individually as necessary.



PL_1STG_Ingest_LCS – triggers the ingestion of the data tables into the staging layer. This is done concurrently so all five processes will begin at once.

Copy data

 Copy child



Copy Child

General, mapping, Settings and User properties are standardised, with no additional configuration.

Source

General

Source

Sink

Mapping

Settings

User properties

Source dataset *

DS_ASQ_LCS

Open

New

Use query

Table

☒ Query

Stored procedure

Query *

select child.PersonID as id,
child.IntegID as caseReference,
CAST(child.LACStart as DATE) as dateOpened,
CAST(child.LACEnd as DATE) as dateClosed,
child.Surname as surname,
child.Forename as forename,
child.Gender as gender,
CAST(child.DOB as DATE) as dateOfBirth
from isperson child

Edit

Query timeout (minutes) ⓘ

120

Isolation level ⓘ


Select...

Partition option ⓘ

☒ None

☐ Physical partitions of table ⓘ

☐ Dynamic range ⓘ

 Please preview data to validate the partition settings.

Additional columns ⓘ

+

New

Sink

General

Source

Sink

Mapping

Settings

User properties

Sink dataset *

DS_ASQ_L_DataMining

Open

New

Dataset properties ⓘ

Name	Value
TargetTable	child
TargetSchema	STG

Write behavior

☒ Insert

☐ Upsert

☐ Stored procedure

Bulk insert table lock ⓘ

☐ Yes

☒ No

Table option

☐ None

☒ Auto create table ⓘ

Pre-copy script ⓘ

truncate table STG.child

Write batch timeout ⓘ

e.g. 00:30:00

Write batch size ⓘ

Max concurrent connections ⓘ

Disable performance metrics analytics ⓘ

☐

Copy data

 Copy significant adults



Copy Significant Adults

General, mapping, Settings and User properties are standardised, with no additional configuration.

Source

GeneralSourceSinkMappingSettingsUser properties

Source dataset *

DS_ASQL_LCS

OpenNew

Use query

Table

Query

Stored procedure

Query *

select rel.PersonId as childId,
adult.personId as significantAdultId,
adult.surname,
adult.forename,
adult.gender,
reltype.GeneralDesc as relationship,
CAST((CASE rel.ParentalResp WHEN 'Y' THEN 1 ELSE 0 END) AS BIT) as
parentalResponsibility -- Need a boolean out from this
FROM dbo.icsrela rel
JOIN dbo.isperson adult on rel.RelatedID = adult.PersonID
JOIN ICSRelationshipType reltype on rel.RelType = reltype.RelationKey
WHERE getDate() between StartDate and ISNULL(EndDate, '20790101')

Query timeout (minutes)

120

Isolation level

Select...

Partition option

None

Physical partitions of table

Dynamic range

Please preview data to validate the partition settings.

Additional columns

New

Sink

GeneralSourceSinkMappingSettingsUser properties

Sink dataset *

DS_ASQL_DataMining

OpenNew

Dataset properties

NameValue

TargetTablesignificantadult

TargetSchemaSTG

Write behavior

Insert

Upsert

Stored procedure

Bulk insert table lock

Yes

No

Table option

None

Auto create table

Pre-copy script

truncate table STG.significantadult

Write batch timeout

e.g. 00:30:00

Write batch size

Max concurrent connections

Disable performance metrics analytics

Copy data



Copy case worker



Copy Case Worker

General, mapping, Settings and User properties are standardised, with no additional configuration.

Source

GeneralSourceSinkMappingSettingsUser properties

Source dataset *

DS_ASQL_LCS

OpenNew

Use query

Table

Query

Stored procedure

Query *

select distinct caseworker.PersonID as childId,
staff.StaffID as allocatedWorkerId,
staff.Surname as surname,
staff.Forename as forename,
staff.Email as email,
roletype.[description] as [role],
department.DeptDesc as team
from dbo.icscaseworker caseworker
JOIN dbo.wxstaff staff on caseworker.WorkerID = staff.StaffID
JOIN dbo.wfpicklistitem roletype on caseworker.CWRole =
roletype.code and pickid = 'ICSRoleTypes'
JOIN dbo.wxdept department on caseworker.WorkerDeptID =
department.DeptID
WHERE GETDATE() BETWEEN caseworker.StartDate and
ISNULL(caseworker.EndDate, '20790101')

Query timeout (minutes)

120

Isolation level

Select...

Partition option

None

Physical partitions of table

Dynamic range

Please preview data to validate the partition settings.

Additional columns

New

Sink

GeneralSourceSinkMappingSettingsUser properties

Sink dataset *

DS_ASQL_DataMining

OpenNew

Dataset properties

NameValue

TargetTablecaseworker

TargetSchemaSTG

Write behavior

Insert

Upsert

Stored procedure

Bulk insert table lock

Yes

No

Table option

None

Auto create table

Pre-copy script

truncate table STG.caseworker

Write batch timeout

e.g. 00:30:00

Write batch size

Max concurrent connections

Disable performance metrics analytics

Copy data



Copy documents



Copy Documents

General, mapping, Settings and User properties are standardised, with no additional configuration.

Source (see complete SQL in appendix)

GeneralSourceSinkMappingSettingsUser properties

Source dataset *

DS_ASQL_LCS

Open

New

Use query

Table

Query

Stored procedure

Query *

SELECT d.docid as docid
, d.filetype as fileType
, d.docdate as docDate
, d.notes
, 'person' as linkType
, dl.refkey as personid
FROM dmdocument d
JOIN dmdoclink dl on dl.linkdocid = d.docid
WHERE refclass = 'com.ics.DBPerson'
UNION
SELECT d.docid as docid
, d.filetype as fileType
, d.docdate as docDate
, d.notes

Query timeout (minutes)

120

Isolation level

Select...

Partition option

None

Physical partitions of table

Dynamic range

Please preview data to validate the partition settings.

Additional columns

New

Sink

GeneralSourceSinkMappingSettingsUser properties

Sink dataset *

DS_ASQL_DataMining

Open

New

Dataset properties

Name

Value

TargetTabledocument

TargetSchemaSTG

Write behavior

Insert

Upsert

Stored procedure

Bulk insert table lock

Yes

No

Table option

None

Auto create table

Pre-copy script

truncate table STG.document

Write batch timeout

e.g. 00:30:00

Write batch size

Max concurrent connections

Disable performance metrics analytics

Copy data



Copy case notes

Copy Case Notes

General, mapping, Settings and User properties are standardised, with no additional configuration.

Source

GeneralSourceSinkMappingSettingsUser properties

Source dataset *

DS_ASQL_LCS

Open

New

Use query

Table

Query

Stored procedure

Query *

SELECT distinct
interview.personid as childId,
casenote.CaseNoteId as noteId,
interview.StartDate as contactDate,
casenotetype.[description] as typeOfContact,
casenote.Reason as reasonForContact,
casenote.Notes as notes
FROM dbo.wfainterviewed interview
JOIN dbo.wfcasenote casenote on interview.CaseNoteId =
casenote.CaseNoteId
JOIN dbo.wfpicklistitem casenotetype on casenote.ContactType =
casenotetype.code and pickid = 'ICSCaseNoteTypes'

Query timeout (minutes)

120

Isolation level

Select...

Partition option

None

Physical partitions of table

Dynamic range

Please preview data to validate the partition settings.

Additional columns

New

Sink

GeneralSourceSinkMappingSettingsUser properties

Sink dataset *

DS_ASQL_DataMining

Open

New

Dataset properties

Name

Value

TargetTablecasenote

TargetSchemaSTG

Write behavior

Insert

Upsert

Stored procedure

Bulk insert table lock

Yes

No

Table option

None

Auto create table

Pre-copy script

truncate table STG.casenote

Write batch timeout

e.g. 00:30:00

Write batch size

Max concurrent connections

Disable performance metrics analytics

PL_2_1_ENT_Ingest_STG – triggers a stored procedure to load data in the staging layer, into the enterprise layer. (This will be utilised to manage incremental loading).

Stored procedure

Load all from STG to ENT

General

Settings

User properties

Linked service *

LinkSqlDataMiningOut

Integration runtime *

integrationRuntime-adf-knowledge-...

Stored procedure name *

[ENT].[sp_load_all]

☒ Enter manually

Stored procedure parameters

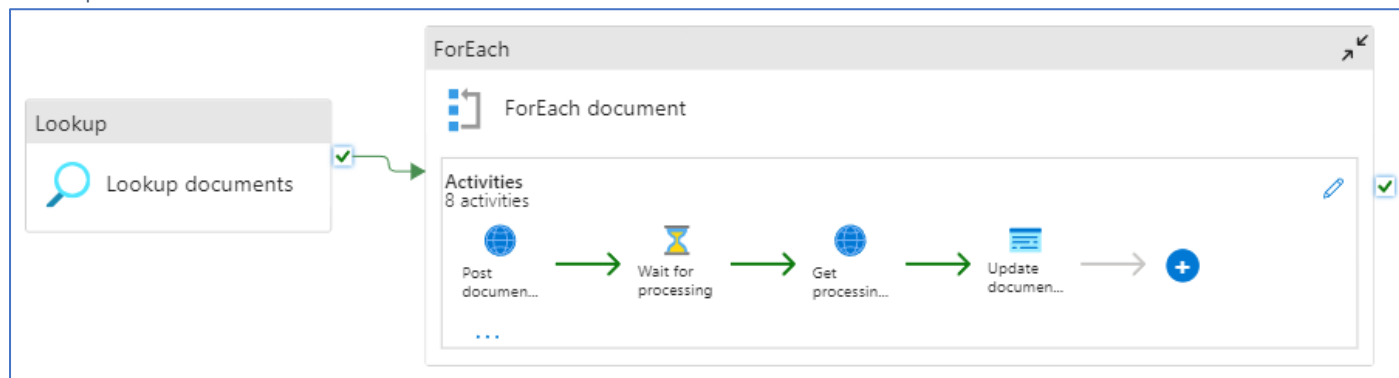
Import

New

Stored procedure triggered: [ENT].[sp_load_all] (See appendix for Stored Procedures)

PL_2_2_ENT_OCR_Documents – triggers a series of processes that retrieve documents IDs from table ‘ENT.vwDocumentsToOCR’, retrieves the documents and sends them to the OCR service. The process sends the document away and returns after a short period of time for an update. If the document has been ‘OCRd’, the result will be populated in the ***** table.

Lookup



Settings for the lookup. This is linked to a ForEach, the rest of the processing sits within this action.

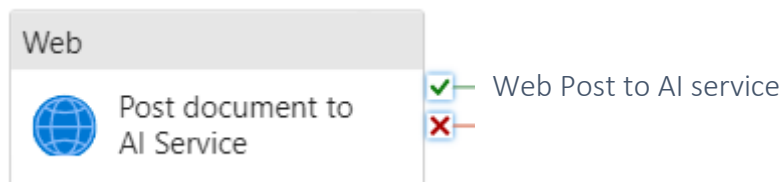
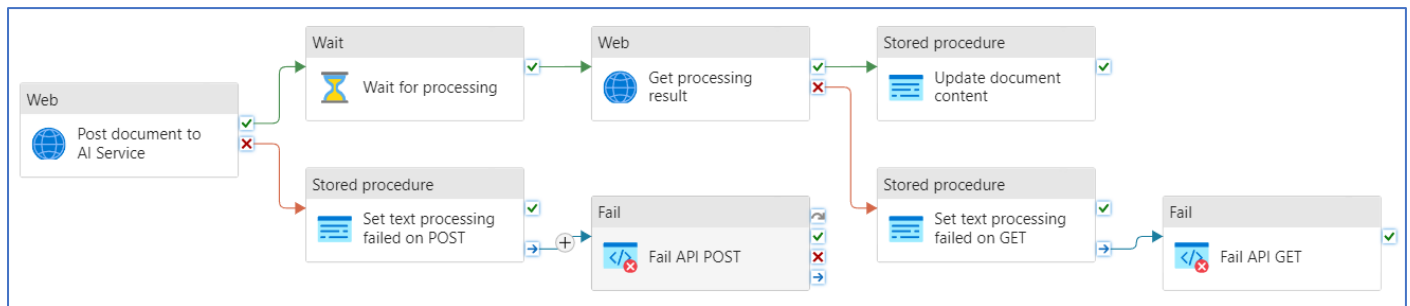
The screenshot shows the 'Settings' tab of the 'Lookup documents' activity. The 'Source dataset' is set to 'DS_ASQL_DataMining'. Below it, 'Dataset properties' are listed: 'TargetTable' is 'document' and 'TargetSchema' is 'ENT'. The 'First row only' checkbox is unchecked. 'Use query' is selected with the 'Query' radio button. The 'Query' text box contains the SQL query: `@concat('select top ', string(pipeline(...`. The 'Query timeout (minutes)' is set to 120. The 'Isolation level' is set to 'Select...'. The 'Partition option' is set to 'None'. A message at the bottom says: 'Please preview data to validate the partition settings.'

Name	Value
TargetTable	document
TargetSchema	ENT

Query text

```
@concat('select top ',
string(pipeline().parameters.numDocumentsToProcess),
' documentId, fileType, url from ENT.vwDocumentsToOCR')
```

ForEach activities



Will follow processing path based on success or failure. General settings below.

General
Settings
User properties

URL * ⓘ
https://cog-data-mining.cognitiveservice...
Information will be sent to the URL specified. Please ensure you trust the URL entered.

Method * ⓘ
POST

Body
@string(json(concat('{"urlSource": "', ...

Authentication ⓘ
None

Headers * ⓘ
+ New | Delete

Name	Value
Ocp-Apim-Subscripti	@pipeline().globalParameters.AI_Ser...
Content-Type	application/json

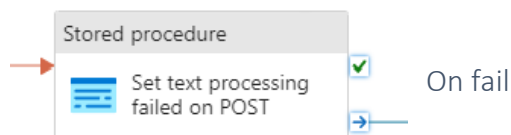
Advanced

Integration runtime * ⓘ
AutoResolveIntegrationRuntime (Ma..y Edit

Completed strings...

- URL: `https://cog-data-mining.cognitiveservices.azure.com/formrecognizer/documentModels/prebuilt-read:analyze?api-version=2023-07-31`
- Body: `@string(json(concat('{"urlSource": "', item().url, '" }')))`
- Name: `Ocp-Apim-Subscription-Key` Value: `@pipeline().globalParameters.AI_Services_Key`

On Fail (see ForEach activity path above)



General Settings User properties

Linked service * LinkSqlDataMiningOut Test connection Edit

Integration runtime * integrationRuntime-adf-knowledge-... Edit

Stored procedure name * [ENT].[update_DocumentProcessingError]

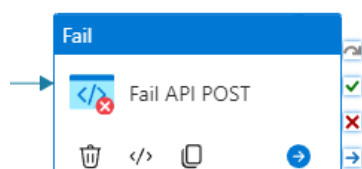
☒ Enter manually

Stored procedure parameters

Import New Delete

<input type="checkbox"/>	Name	Type	Value
<input type="checkbox"/>	documentId	Guid	@item().documentId
<input type="checkbox"/>	errorMessage	String	@activity('Post document to AI Servi...

- Stored procedure triggered: [ENT].[update_DocumentProcessingError] (See appendix for Stored Procedures)
- errorMessage 'value': @activity('Post document to AI Service').error.Message



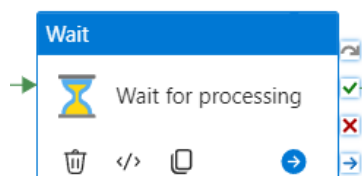
General Settings User properties

Fail message * @activity('Post document to AI Servi...

Error code * 0

Fail message : @activity('Post document to AI Service').error.Message

On success



General Settings User properties

Wait time in seconds * 15

When the REST API 'Post document to AI service' is successful, the process will wait 15 seconds before taking further action. This allows sufficient time for the API post to be received and understood.



General **Settings** User properties

URL: @activity('Post document to AI Servi...
 ⚠ Information will be sent to the URL specified. Please ensure you trust the URL entered.

Method * ⓘ: GET

Authentication ⓘ: None

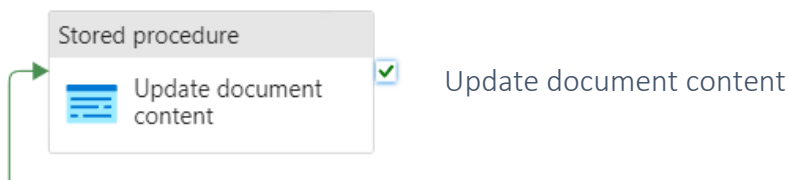
Headers * ⓘ: + New | Delete

<input type="checkbox"/>	Name	Value
<input type="checkbox"/>	Ocp-Apim-Subscripti	@pipeline().globalParameters.AI_Ser...

Advanced

Integration runtime * ⓘ: ☒ AutoResolveIntegrationRuntime (Ma... Edit

- URL: @activity('Post document to AI Service').output.ADFWebActivityResponseHeaders['Operation-Location'] (also set under 'User properties')
- Name: Ocp-Apim-Subscription-Key Value: @pipeline().globalParameters.AI_Services_Key



General **Settings** User properties

Linked service * ⓘ: LinkSqlDataMiningOut Test connection Edit

Integration runtime *: ☒ integrationRuntime-adf-knowledge-... Edit

Stored procedure name *: [ENT].[update_DocumentContent]
☒ Enter manually

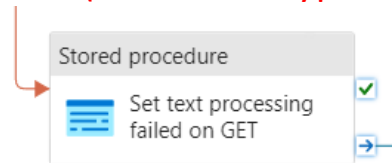
Stored procedure parameters ⓘ

Import + New | Delete

<input type="checkbox"/>	Name	Type	Value
<input type="checkbox"/>	content	String	@activity('Get processing result').out...
<input type="checkbox"/>	documentId	Guid	@item().documentId

- Stored procedure triggered: [ENT].[update_DocumentContent] (See appendix for Stored Procedures)
- @activity('Get processing result').output.analyzeResult.content

On Fail (see ForEach activity path above)



General Settings User properties

Linked service * LinkSqlDataMiningOut Test connection Edit

Integration runtime * integrationRuntime-adf-knowledge-... Edit

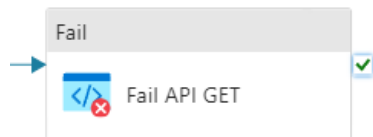
Stored procedure name * [ENT].[update_DocumentProcessingError]
 ☒ Enter manually

▼ Stored procedure parameters

← Import + New Delete

<input type="checkbox"/>	Name	Type	Value
<input type="checkbox"/>	documentId	Guid	@item().documentId
<input type="checkbox"/>	errorMessage	String	@activity('Get processing result').err...

- Stored procedures triggered: [ENT].[update_DocumentProcessingError] (See appendix for Stored Procedures)
- @activity('Get processing result').error.Message



General Settings User properties

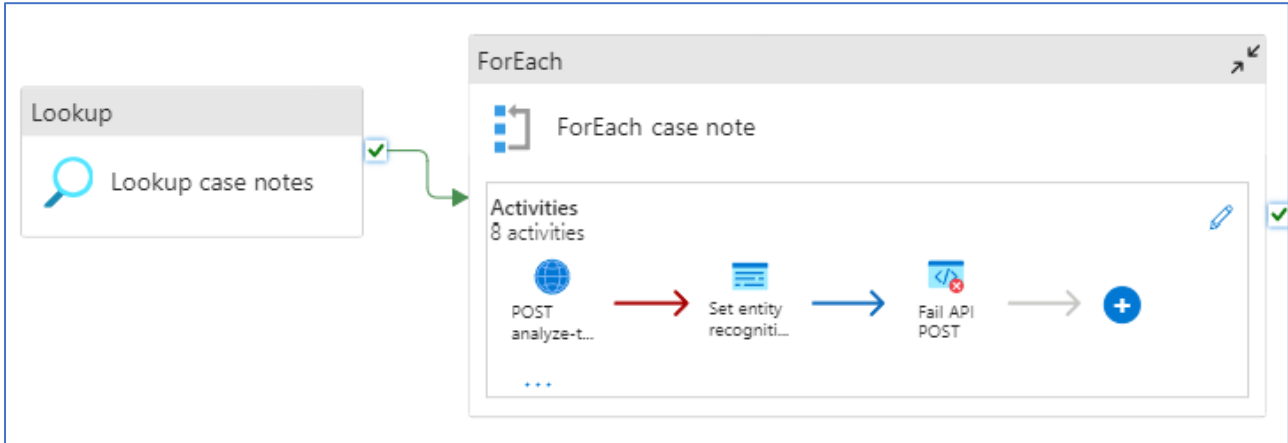
Fail message * @activity('Get processing result').err...

Error code * 0

- @activity('Get processing result').error.Message

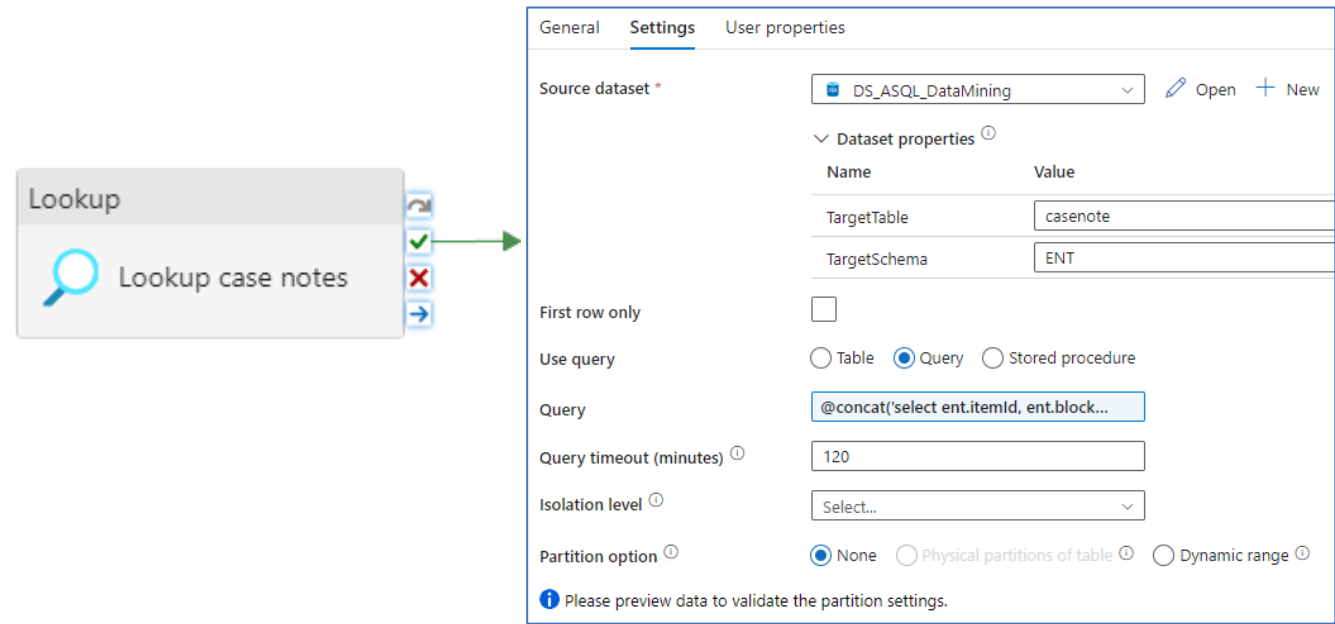
PL_2_3_ENT_EntityRecognition_CaseNotes_async triggers entity recognition on case notes things happen and such, the result will be populated in the ***** table.

Lookup case notes



Parameters				Variables	Settings	Output
+ New Delete						
<input type="checkbox"/>	Name	Type	Default value			
<input type="checkbox"/>	numCasenotesToProcess	Int	10			
<input type="checkbox"/>	textLimit	Int	124000			

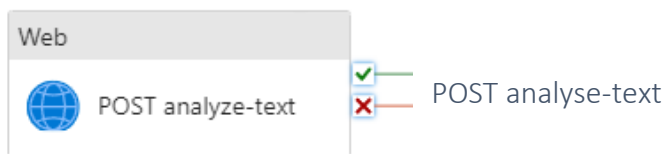
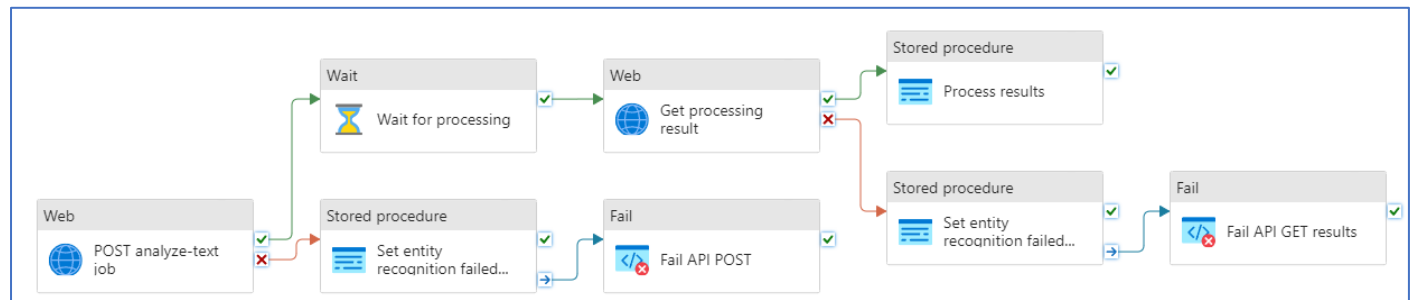
Settings for the lookup. This is linked to a ForEach, the rest of the processing sits within this action.



Query text

```
@concat('select ent.itemId,  
ent.blockNumber,  
ent.jsonText  
from ENT.fnCasenotesToEntityRecognise(',  
string(pipeline().parameters.numCasenotesToProcess),  
,',',  
string(pipeline().parameters.textLimit),') ent')
```

ForEach activities



General Settings User properties

URL * ⓘ

⚠ Information will be sent to the URL specified. Please ensure you trust the URL entered.

Method * ⓘ

Body

Authentication ⓘ

Headers * ⓘ

+ New | Delete

<input type="checkbox"/>	Name	Value
<input type="checkbox"/>	Ocp-Apim-Subscripti	@pipeline().globalParameters.AI_Ser...
<input type="checkbox"/>	Content-Type	application/json

Advanced

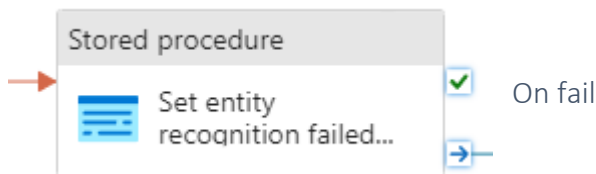
Integration runtime * ⓘ Edit

- URL: <https://cog-data-mining.cognitiveservices.azure.com/language/analyze-text/jobs?api-version=2023-04-01>
- Name: Ocp-Apim-Subscription-Key Value: @pipeline().globalParameters.AI_Services_Key

- Body:

```
@string(json(concat('{
"displayName": "Entity Recognition",
"analysisInput": {
  "documents": [' , item().jsonText, ' ]
},
"tasks": [
  {
    "kind": "EntityRecognition",
    "taskName": "Entity Recognition Task",
    "parameters": { }
  }
]
}'))
```

On Fail (see ForEach activity path above)



General			Settings	User properties
Linked service *		LinkSqlDataMiningOut		Test connection Edit
Integration runtime *		integrationRuntime-adf-knowledge-..		Edit
Stored procedure name *		[ENT].[update_CaseNoteEntityError]		
		<input checked="" type="checkbox"/> Enter manually		
Stored procedure parameters ⓘ				
<div> <div>← Import</div> <div>+ New</div> <div>🗑 Delete</div> </div>				
<input type="checkbox"/>	Name	Type	Value	
<input type="checkbox"/>	casenoteld	Guid	@item().itemId	
<input type="checkbox"/>	errorMessage	String	@activity('POST analyze-text').error....	

- Stored procedures triggered: [ENT].[update_CaseNoteEntityError] (**See appendix (*) for Stored Procedures**)
- errorMessage: @activity('POST analyze-text').error.Message

Fail

Fail API POST

Settings

Fail message *

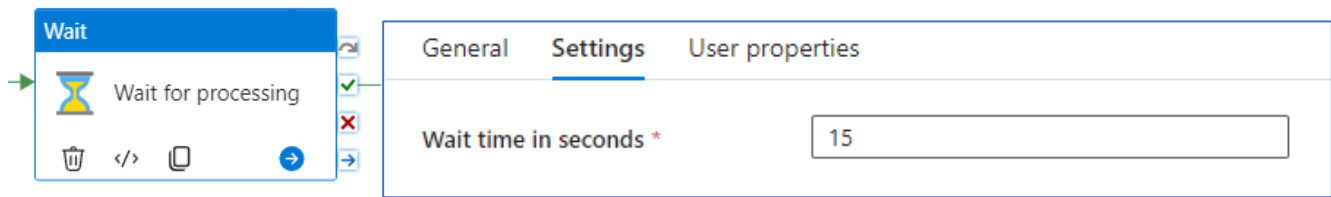
@activity('POST analyze-text').error....

Error code * ⓘ

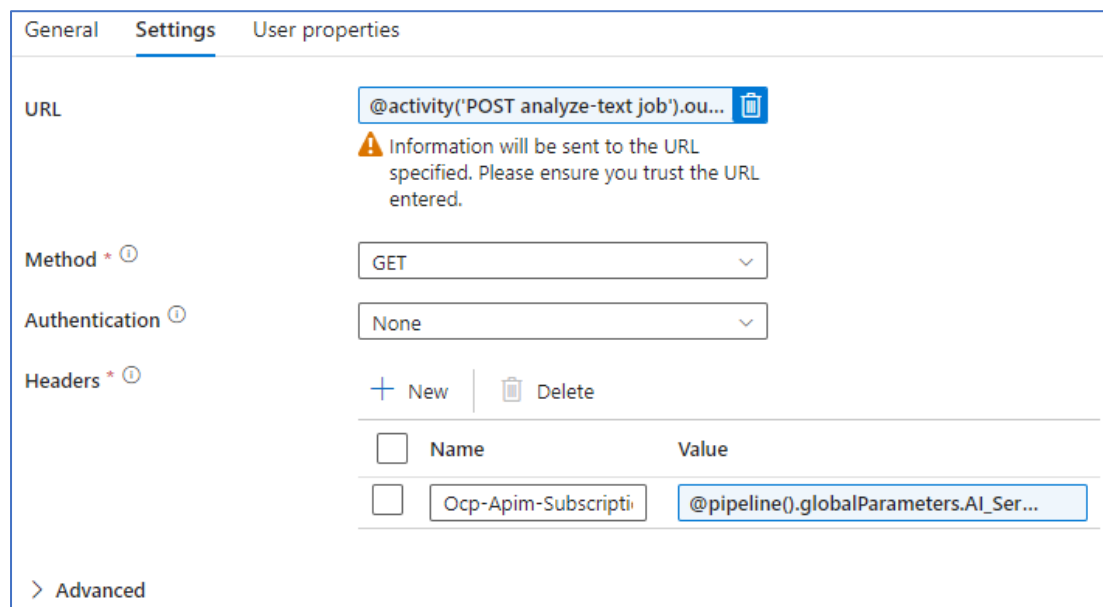
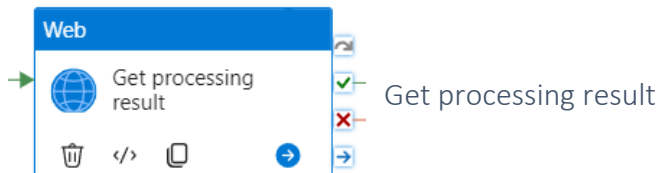
0

- @activity('POST analyze-text').error.Message

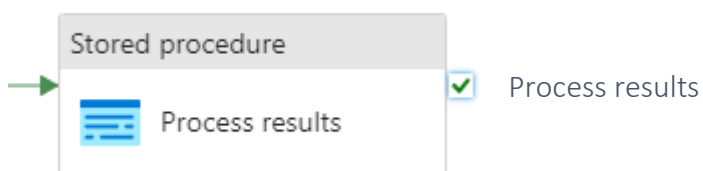
On success



When the REST API 'Post document to AI service' is successful, the process will wait 15 seconds before taking further action. This allows sufficient time for the API post to be received and understood.



- URL: @activity('POST analyze-text job').output.ADFWebActivityResponseHeaders['Operation-Location'] (also set under 'User properties')
- Name: Ocp-Apim-Subscription-Key Value: @pipeline().globalParameters.AI_Services_Key



General **Settings** User properties

Linked service * ^① LinkSqlDataMiningOut Test connection Edit

Integration runtime * integrationRuntime-adf-knowledge-... Edit

Stored procedure name * [ENT].[process_childCaseNoteResponse]
☒ Enter manually

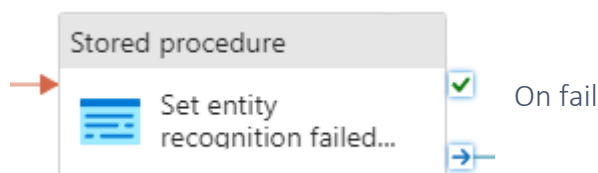
▼ Stored procedure parameters ^①

← Import + New | Delete

<input type="checkbox"/>	Name	Type	Value
<input type="checkbox"/>	apiResponse	String	@string(activity('POST analyze-text')....
<input type="checkbox"/>	blockNumber	Int32	@item().blockNumber
<input type="checkbox"/>	itemId	Guid	@item().itemId

- Stored procedures triggered: [ENT].[process_childCaseNoteResponse] (See appendix for Stored Procedures)
- apiResponse: @string(activity('POST analyze-text').output)

On Fail (see ForEach activity path above)



General **Settings** User properties

Linked service * ^① LinkSqlDataMiningOut Test connection Edit + New

Integration runtime * integrationRuntime-adf-knowledge-... Edit

Stored procedure name * [ENT].[update_CaseNoteEntityError]
☒ Enter manually

▼ Stored procedure parameters ^①

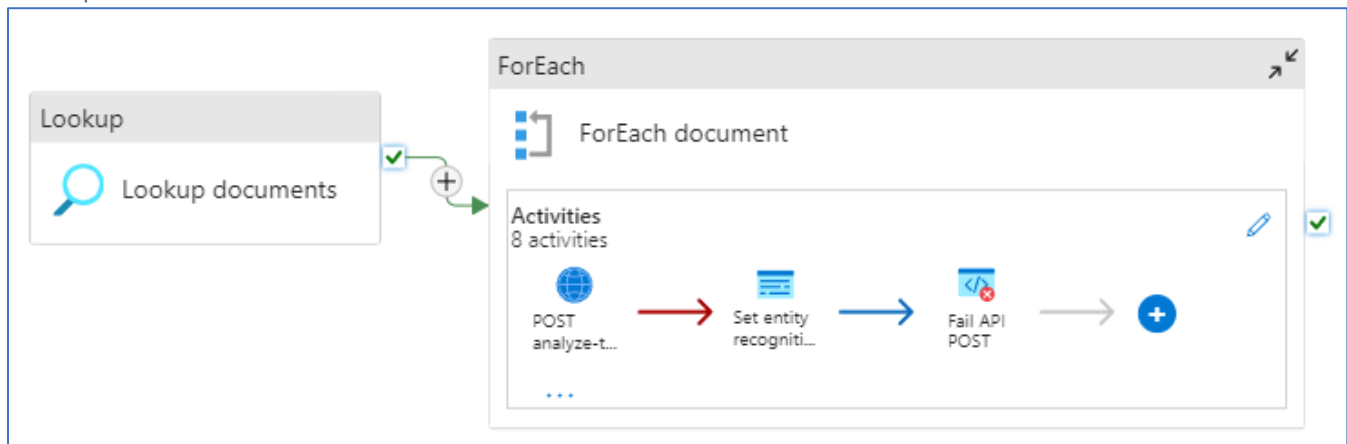
← Import + New | Delete

<input type="checkbox"/>	Name	Type	Value
<input type="checkbox"/>	casenoteld	Guid	@item().itemId
<input type="checkbox"/>	errorMessage	String	@activity('Get processing result').err...

- Stored procedures triggered: [ENT].[update_CaseNoteEntityError] (See appendix (*) for Stored Procedures)
- errorMessage: @activity('Get processing result').error.Message

PL_2_3_ENT_EntityRecognition_Documents_async triggers entity recognition on documents things happen and such, the result will be populated in the ***** table.

Lookup Documents



Parameters				Variables	Settings	Output
+ New				Delete		
<input type="checkbox"/>	Name	Type	Default value			
<input type="checkbox"/>	numDocumentsToProcess	Int	10			
<input type="checkbox"/>	textLimit	Int	124000			

Settings for the lookup. This is linked to a ForEach, the rest of the processing sits within this action.

Lookup

Lookup documents

General

Settings

User properties

Source dataset *

DS_ASQI_DataMining

Open

New

Dataset properties

Name	Value
TargetTable	document
TargetSchema	ENT

First row only☐

Use query

Table

Query

Stored procedure

Query

@concat('Select itemid, blockNumbe...

Query timeout (minutes) ¹

120

Isolation level ¹

Select...

Partition option ¹

None

Physical partitions of table ¹

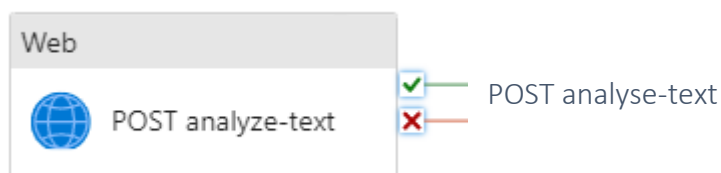
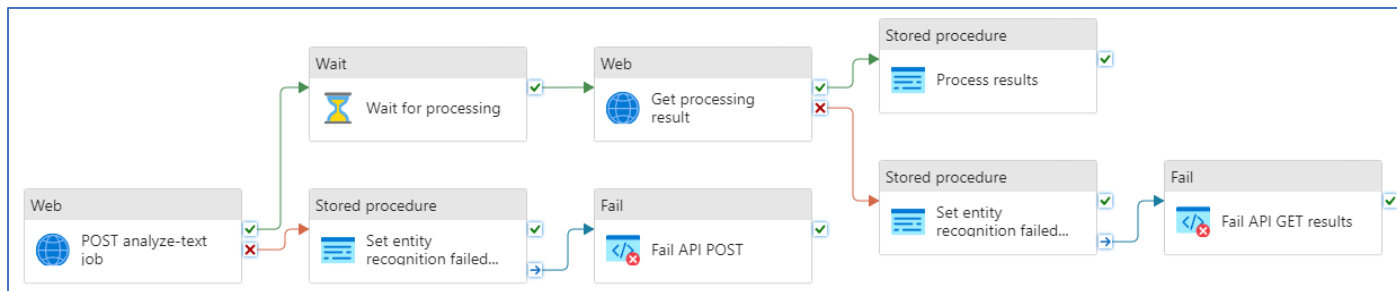
Dynamic range ¹

Please preview data to validate the partition settings.

Query text

```
@concat('Select itemId, blockNumber, jsonText
from ENT.fnDocumentsToEntityRecognise(',
string(pipeline().parameters.numDocumentsToProcess),
', ',
string(pipeline().parameters.textLimit),
')')
```

ForEach activities



General Settings User properties

URL * ⓘ

⚠ Information will be sent to the URL specified. Please ensure you trust the URL entered.

Method * ⓘ

Body

Authentication ⓘ

Headers * ⓘ

+ New | Delete

Name	Value
<input type="checkbox"/> Ocp-Apim-Subscripti	<input type="text" value="@pipeline().globalParameters.AI_Ser..."/>
<input type="checkbox"/> Content-Type	<input type="text" value="application/json"/>

Advanced

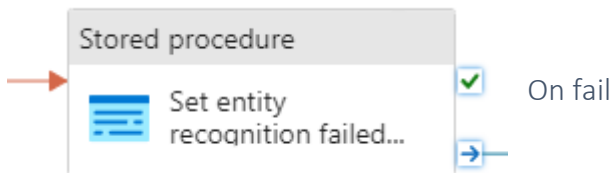
Integration runtime * ⓘ Edit

- URL: <https://cog-data-mining.cognitiveservices.azure.com/language/analyze-text/jobs?api-version=2023-04-01>
- Name: Ocp-Apim-Subscription-Key Value @pipeline().globalParameters.AI_Services_Key:

- Body:

```
@string(json(concat('{
"displayName": "Entity Recognition",
"analysisInput": {
  "documents": [' , item().jsonText, ' ]
},
"tasks": [
  {
    "kind": "EntityRecognition",
    "taskName": "Entity Recognition Task",
    "parameters": { }
  }
]
}'))
```

On Fail (see ForEach activity path above)



General **Settings** User properties

Linked service * ⓘ LinkSqlDataMiningOut Test connection Edit

Integration runtime * integrationRuntime-adf-knowledge-... Edit

Stored procedure name * [ENT].[update_DocumentEntityError] ☒ Enter manually

Stored procedure parameters ⓘ

Import + New Delete

<input type="checkbox"/>	Name	Type	Value
<input type="checkbox"/>	documentId	Guid	@item().itemId
<input type="checkbox"/>	errorMessage	String	@activity('POST analyze-text').error....

- Stored procedure triggered: [ENT].[update_DocumentEntityError] (See appendix (*) for Stored Procedures)
- @activity('POST analyze-text').error.Message

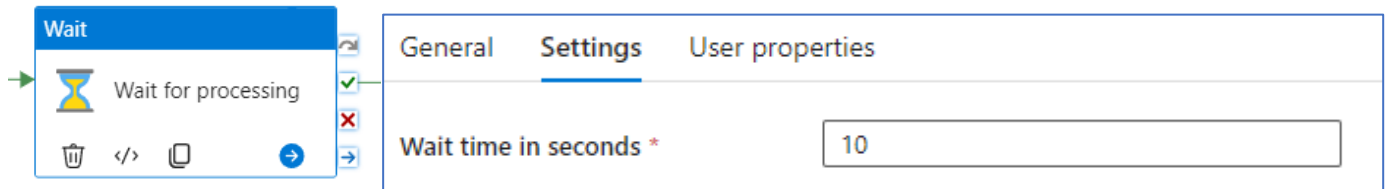
General **Settings** User properties

Fail message * @activity('POST analyze-text').error....

Error code * ⓘ 0


- @activity('POST analyze-text').error.Message

On success



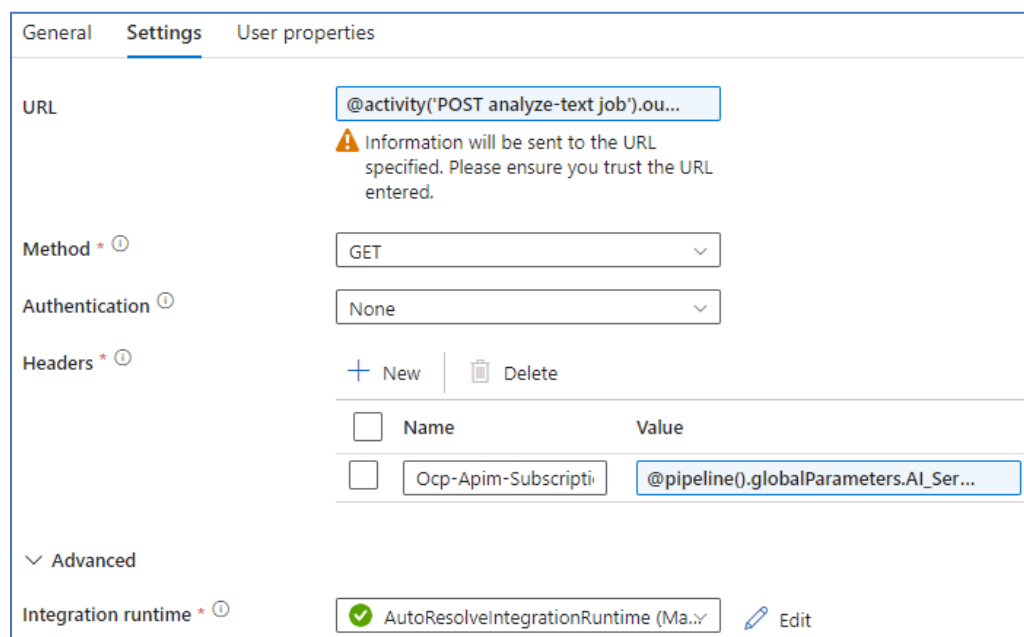
The screenshot shows the 'Wait' activity configuration. The 'Settings' tab is active, displaying 'Wait time in seconds *' with a value of 10. The 'General' and 'User properties' tabs are also visible.

When the REST API 'Post document to AI service' is successful, the process will wait 10 seconds before taking further action. This allows sufficient time for the API post to be received and understood.



The screenshot shows the 'Web' activity configuration. The 'Get processing result' activity is selected, and the 'Settings' tab is active. The 'URL' field is set to '@activity('POST analyze-text job').ou...'. The 'Method' is set to 'GET' and 'Authentication' is set to 'None'. The 'Headers' section shows a table with one header row and one data row.

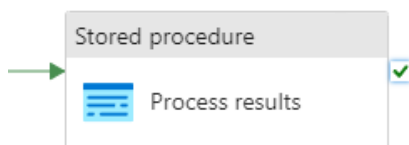
Name	Value
Ocp-Apim-Subscripti	@pipeline().globalParameters.AI_Ser...



The screenshot shows the 'Web' activity configuration. The 'Settings' tab is active, displaying 'URL' as '@activity('POST analyze-text job').ou...', 'Method' as 'GET', and 'Authentication' as 'None'. The 'Headers' section shows a table with one header row and one data row.

Name	Value
Ocp-Apim-Subscripti	@pipeline().globalParameters.AI_Ser...

- URL: @activity('POST analyze-text job').output.ADFWebActivityResponseHeaders['Operation-Location']
- Name: Ocp-Apim-Subscription-Key Value @pipeline().globalParameters.AI_Services_Key:



The screenshot shows the 'Stored procedure' activity configuration. The 'Process results' activity is selected, and the 'Settings' tab is active. The 'URL' field is set to '@activity('POST analyze-text job').ou...'. The 'Method' is set to 'GET' and 'Authentication' is set to 'None'. The 'Headers' section shows a table with one header row and one data row.

Name	Value
Ocp-Apim-Subscripti	@pipeline().globalParameters.AI_Ser...

General **Settings** User properties

Linked service * ⓘ LinkSqlDataMiningOut [Test connection](#) [Edit](#)

Integration runtime * ✓ integrationRuntime-adf-knowledge-... [Edit](#)

Stored procedure name * [ENT].[process_childDocumentResponse]
☒ Enter manually

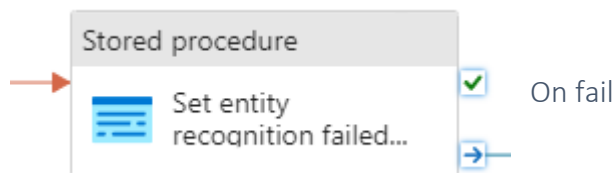
▼ Stored procedure parameters ⓘ

← Import + New | Delete

<input type="checkbox"/>	Name	Type	Value
<input type="checkbox"/>	apiResponse	String	@string(activity('POST analyze-text')....
<input type="checkbox"/>	blockNumber	Int32	@item().blockNumber
<input type="checkbox"/>	itemId	Guid	@item().itemId

- Stored procedure triggered: [ENT].[process_childDocumentResponse] (See appendix (*) for Stored Procedures)
- apiResponse: @string(activity('POST analyze-text').output)

On Fail (see ForEach activity path above)



General **Settings** User properties

Linked service * ⓘ LinkSqlDataMiningOut [Test connection](#) [Edit](#)

Integration runtime * ✓ integrationRuntime-adf-knowledge-... [Edit](#)

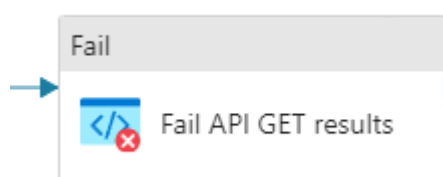
Stored procedure name * [ENT].[update_DocumentEntityError]
☒ Enter manually

▼ Stored procedure parameters ⓘ

← Import + New | Delete

<input type="checkbox"/>	Name	Type	Value
<input type="checkbox"/>	casenotId	Guid	@item().itemId
<input type="checkbox"/>	errorMessage	String	@activity('Get processing result').err...

- Stored procedure triggered: [ENT].[update_DocumentEntityError] (See appendix (*) for Stored Procedures)
- @activity('Get processing result').error.Message



General **Settings** User properties

Fail message * @activity('Get processing result').err...

Error code * ⓘ 0

- @activity('Get processing result').error.Message

Appendix

PL 1 STG Ingest LCS

SQL Queries

Copy Child - source

```
select child.PersonID as id,
child.IntegID as caseReference,
CAST(child.LACStart as DATE) as dateOpened,
CAST(child.LACEnd as DATE) as dateClosed,
child.Surname as surname,
child.Forename as forename,
child.Gender as gender,
CAST(child.DOB as DATE) as dateOfBirth
from isperson child
```

Copy Significant adults – source

```
select rel.PersonId as childId,
adult.personId as significantAdultId,
adult.surname,
adult.forename,
adult.gender,
reltype.GeneralDesc as relationship,
CAST((CASE rel.ParentalResp WHEN 'Y' THEN 1 ELSE 0 END) AS BIT) as parentalResponsibility
FROM dbo.icsrela rel
    JOIN dbo.isperson adult
        on rel.RelatedID = adult.PersonID
    JOIN ICSRelationshipType reltype
        on rel.RelType = reltype.RelationKey
WHERE getdate() between StartDate and ISNULL(EndDate, '20790101')
```

Copy case worker – source

```
select distinct caseworker.PersonID as childId,
staff.StaffID as allocatedWorkerId,
staff.Surname as surname,
staff.Forename as forename,
staff.EMail as email,
roletype.[description] as [role],
department.DeptDesc as team
from dbo.icscaseworker caseworker
    JOIN dbo.wxstaff staff
        on caseworker.WorkerID = staff.StaffID
    JOIN dbo.wfpicklistitem roletype
        on caseworker.CWRole = roletype.code
        and pickid = 'ICSRoleTypes'
    JOIN dbo.wxdept department
        on caseworker.WorkerDeptID = department.DeptID
WHERE GETDATE() BETWEEN caseworker.StartDate
and ISNULL(caseworker.EndDate, '20790101')
```

Copy documents – source

```
SELECT d.docid as docId
      , d.filetype as fileType
      , d.docdate as docDate
      , d.notes
      , 'person' as linkType
      , dl.refkey as personId
FROM dmdocument d
     JOIN dmdoclink dl on dl.linkdocid = d.docid
WHERE refclass = 'com.ics.DBPerson'
UNION
SELECT d.docid as docId
      , d.filetype as fileType
      , d.docdate as docDate
      , d.notes
      , 'assessment' as linkType
      , a.formlinkkey personId
FROM dmdocument d
     JOIN dmdoclink dl on dl.linkdocid = d.docid
     JOIN wfacssessment a on a.formno = dl.refkey
WHERE refclass = 'com.liquidlogic.assessment.DBAssessment'
UNION
select d.docid as docId
      , d.filetype as fileType
      , d.docdate as docDate
      , d.notes
      , 'meeting' as linkType
      , mp.participkey personId
FROM dmdocument d
     JOIN dmdoclink dl on dl.linkdocid = d.docid
     JOIN ppmeeting m on m.meetingid = dl.refkey
     JOIN ppmeetparts mp on mp.meetingid = m.meetingid
WHERE dl.refclass = 'com.liquidlogic.path.DBPathMeeting'
UNION
SELECT d.docid as docId
      , d.filetype as fileType
      , d.docdate as docDate
      , d.notes
      , 'missing' as linkType
      , mp.personid as personId
FROM dmdocument d
     JOIN dmdoclink dl on dl.linkdocid = d.docid
     JOIN icsmissingperson mp on mp.missingpersonid = dl.refkey
WHERE dl.refclass = 'com.ics.ICSMissingPerson'
```

Copy case notes – source

```
SELECT distinct
interview.personid as childId,
casenote.CaseNoteId as noteId,
interview.StartDate as contactDate,
casenotetype.[description] as typeOfContact,
casenote.Reason as reasonForContact,
casenote.Notes as notes
FROM dbo.wfainterviewed interview
      JOIN dbo.wfcasenote casenote on interview.CaseNoteID = casenote.CaseNoteID
      JOIN dbo.wfpicklistitem casenotetype on casenote.ContactType = casenotetype.code and
pickid =  'ICSCaseNoteTypes'
```


PL 2 1 ENT Ingest STG

Stored Procedures

sp_load_all

```
CREATE PROC [ENT].[sp_load_all]
AS
BEGIN
    exec ENT.sp_load_casenote
    exec ENT.sp_load_caseworker
    exec ENT.sp_load_child
    exec ENT.sp_load_document
    exec ENT.sp_load_significantAdult
END
GO
```

sp_load_casenote

```
CREATE PROC [ENT].[sp_load_casenote]
AS
BEGIN
MERGE ENT.casenote as tgt
USING
(SELECT
    [childId] as internalChildId
    ,[noteId]
    ,[contactDate]
    ,[typeOfContact]
    ,[reasonForContact]
    ,[notes]
FROM STG.casenote notes
) as src
on src.noteId = tgt.noteId and src.internalChildId = tgt.internalChildId
WHEN MATCHED
THEN
UPDATE
set
tgt.contactDate = src.contactDate,
tgt.typeOfContact = src.typeOfContact,
tgt.reasonForContact = src.reasonForContact,
tgt.notes = src.notes,
tgt.modifiedAt = getdate()
WHEN NOT MATCHED
THEN INSERT ([internalChildId]
    ,[noteId]
    ,[contactDate]
    ,[typeOfContact]
    ,[reasonForContact]
    ,[notes]
    ,processedFlag
    ,errorFlag
    ,modifiedAt)
values (src.[internalChildId]
    ,src.[noteId]
    ,src.[contactDate]
    ,src.[typeOfContact]
    ,src.[reasonForContact]
    ,src.[notes]
    ,0
    ,0,
    getdate()); END
```

```

GO
sp_load_caseworker
CREATE PROC [ENT].[sp_load_caseworker]
AS
BEGIN
MERGE ENT.caseworker as tgt
USING
(
    SELECT
        [childId] as internalChildId
        ,[allocatedWorkerId]
        ,[surname]
        ,[forename]
        ,[email]
        ,[role]
        ,[team]
    FROM STG.caseworker
) as src
on src.internalChildId = tgt.internalChildId and src.allocatedWorkerId =
tgt.allocatedWorkerId and src.role = tgt.role
WHEN MATCHED
THEN
UPDATE
set
tgt.surname = src.surname,
tgt.forename = src.forename,
tgt.email = src.email,
tgt.team = src.team,
tgt.modifiedAt = getdate()
WHEN NOT MATCHED
THEN INSERT ([internalChildId]
,[allocatedWorkerId]
,[surname]
,[forename]
,[email]
,[role]
,[team]
,modifiedAt)
values (src.[internalChildId]
,src.[allocatedWorkerId]
,src.[surname]
,src.[forename]
,src.[email]
,src.[role]
,src.[team]
,getdate() );
END
GO

```

```

sp_load_child
CREATE PROC [ENT].[sp_load_child]
AS
BEGIN
MERGE ENT.child as tgt
USING
(
    SELECT
        [id] as internalChildId
        ,[caseReference]
        ,[dateOpened]
        ,[dateClosed]
        ,[surname]
        ,[forename]
        ,[gender]
        ,[dateOfBirth]
    FROM STG.child
) as src
on src.internalChildId = tgt.internalChildId
WHEN MATCHED
THEN
UPDATE
set
tgt.[caseReference] = src.[caseReference]
    ,tgt.[dateOpened] = src.[dateOpened]
    ,tgt.[dateClosed] = src.[dateClosed]
    ,tgt.[surname] = src.[surname]
    ,tgt.forename = src.[forename]
    ,tgt.[gender] = src.gender
    ,tgt.[dateOfBirth] = src.[dateOfBirth]
    ,tgt.modifiedAt = getdate()
WHEN NOT MATCHED
THEN INSERT ([internalChildId]
    ,[caseReference]
    ,[dateOpened]
    ,[dateClosed]
    ,[surname]
    ,[forename]
    ,[gender]
    ,[dateOfBirth]
    ,modifiedAt)
values (src.[internalChildId]
    ,src.[caseReference]
    ,src.[dateOpened]
    ,src.[dateClosed]
    ,src.[surname]
    ,src.[forename]
    ,src.[gender]
    ,src.[dateOfBirth]
    ,getdate()
);
END
GO

```

```

sp_load_document
CREATE PROC [ENT].[sp_load_document]
AS
BEGIN
MERGE ENT.document as tgt
USING
(
    SELECT
        [docId] as internalDocId
        ,[fileType]
        ,[docDate]
        ,[notes]
        ,[linkType]
        ,[personId] as internalChildId
        ,[text]
        FROM STG.document
) as src
on src.internalDocId = tgt.internalDocId and src.internalChildId = tgt.internalChildId and
src.linkType = tgt.linkType
WHEN MATCHED
THEN
UPDATE
set
    tgt.[fileType] = src.[fileType]
    ,tgt.[docDate] = src.[docDate]
    ,tgt.[notes] = src.[notes]
    ,tgt.[text] = src.[text]
    ,tgt.modifiedAt = getdate()
WHEN NOT MATCHED
THEN INSERT ( [internalDocId]
    ,[fileType]
    ,[docDate]
    ,[notes]
    ,[linkType]
    ,[internalChildId]
    ,[text]
    ,modifiedAt)
values (src.[internalDocId]
    ,src.[fileType]
    ,src.[docDate]
    ,src.[notes]
    ,src.[linkType]
    ,src.[internalChildId]
    ,src.[text]
    ,getdate()
);
END
GO

```

```

sp_load_significantAdult
CREATE PROC [ENT].[sp_load_significantAdult]
AS
BEGIN
MERGE ENT.significantAdult as tgt
USING
(
    SELECT distinct
        [childId] as internalChildId
        ,[significantAdultId] as internalSignificantAdultId
        ,[surname]
        ,[forename]
        ,[gender]
        ,[relationship]
        ,[parentalResponsibility]
        FROM STG.significantAdult
) as src
on src.internalChildId = tgt.internalChildId and src.internalSignificantAdultId =
tgt.internalSignificantAdultId and src.relationship = tgt.relationship
and src.parentalResponsibility = tgt.parentalResponsibility
WHEN MATCHED
THEN
UPDATE
set
    tgt.[surname] = src.[surname]
    ,tgt.[forename] = src.[forename]
    ,tgt.[gender] = src.[gender]
    ,tgt.modifiedAt = getdate()
WHEN NOT MATCHED
THEN INSERT ([internalChildId]
    ,[internalSignificantAdultId]
    ,[surname]
    ,[forename]
    ,[gender]
    ,[relationship]
    ,[parentalResponsibility]
    ,modifiedAt
)
values (src.[internalChildId]
    ,src.[internalSignificantAdultId]
    ,src.[surname]
    ,src.[forename]
    ,src.[gender]
    ,src.[relationship]
    ,src.[parentalResponsibility],
    getdate());
END
GO

```

```

update_DocumentContent
CREATE proc [ENT].[update_DocumentContent]
(
    @documentId uniqueidentifier,
    @content nvarchar(max)
)
as
BEGIN
update ENT.document
set [text] = @content
where @documentId = documentId
END
GO

```

```

update_DocumentProcessingError
CREATE proc [ENT].[update_DocumentProcessingError]
(
    @documentId uniqueidentifier,
    @errorMessage nvarchar(max)
)
as
BEGIN
update ENT.document
set OCRErrorFlag = 1,
errorMessages = @errorMessage
where @documentId = documentId
END
GO

```

```

process_childCasenoteResponse
CREATE PROC [ENT].[process_childCasenoteResponse]
(
    @itemId uniqueidentifier,
    @blockNumber int,
    @apiResponse nvarchar(max)
)
AS
BEGIN
MERGE ENT.casenoteEntity as tgt
USING
(
    SELECT
        docs.casenoteId as casenoteId,
        @blockNumber as blockNumber,
        ROW_NUMBER() OVER (PARTITION BY casenoteId ORDER BY casenoteId) as entityId,
        JSON_VALUE(entities.value, '$.text') AS text,
        JSON_VALUE(entities.value, '$.category') as category,
        JSON_VALUE(entities.value, '$.subcategory') as subCategory,
        cast(JSON_VALUE(entities.value, '$.offset') as int) as offset,
        cast(JSON_VALUE(entities.value, '$.length') as int) as length,
        cast(JSON_VALUE(entities.value, '$.confidenceScore') as decimal(10, 2)) as confidenceScore
    FROM ENT.casenote docs
    CROSS APPLY OPENJSON(@apiResponse, '$.results.documents[0].entities') entities
    where casenoteId = @itemId
) as src
on src.casenoteId = tgt.casenoteId and src.entityid = tgt.entityid and src.blockNumber =
tgt.blockNumber
WHEN NOT MATCHED BY TARGET
THEN INSERT(casenoteId, blockNumber, entityid, text, category, subCategory, offset, length,
confidencescore, modifiedAt)
values (src.casenoteId,src.blockNumber, src.entityid, src.text, src.category, src.subCategory,
src.offset, src.length, src.confidencescore, getdate());

update ENT.casenote set entityProcessedFlag = 1 where casenoteId = @itemId
END

```

```

GO

update_CaseNoteEntityError
CREATE proc [ENT].[update_CaseNoteEntityError]
(
    @casenoteId uniqueidentifier,
    @errorMessage nvarchar(max)
)
as
BEGIN
    update ENT.casenote
    set entityProcessedFlag = 1,
    entityErrorFlag = 1,
    errorMessages = @errorMessage
    where @casenoteId = casenoteId
END
GO

process_childDocumentResponse
CREATE PROC [ENT].[process_childDocumentResponse]
(
    @itemId uniqueidentifier,
    @blockNumber int,
    @apiResponse nvarchar(max)
)
AS
BEGIN
    MERGE ENT.documentEntity as tgt
    USING
    (
        SELECT
            docs.documentId as documentId,
            @blockNumber as blockNumber,
            ROW_NUMBER() OVER (PARTITION BY documentId ORDER BY documentId) as entityId,
            JSON_VALUE(entities.value, '$.text') AS text,
            JSON_VALUE(entities.value, '$.category') as category,
            JSON_VALUE(entities.value, '$.subcategory') as subCategory,
            cast(JSON_VALUE(entities.value, '$.offset') as int) as offset,
            cast(JSON_VALUE(entities.value, '$.length') as int) as length,
            cast(JSON_VALUE(entities.value, '$.confidenceScore') as decimal(10, 2)) as confidenceScore
        FROM ENT.document docs
        CROSS APPLY OPENJSON(@apiResponse, '$.results.documents[0].entities') entities
        where documentId = @itemId
    ) as src
    on src.documentId = tgt.documentId and src.entityid = tgt.entityid and src.blockNumber =
    tgt.blockNumber
    WHEN NOT MATCHED BY TARGET
    THEN INSERT(documentId, blockNumber, entityid, text, category, subCategory, offset, length,
    confidencescore, modifiedAt)
    values (src.documentId,src.blockNumber, src.entityid, src.text, src.category, src.subCategory,
    src.offset, src.length, src.confidencescore, getdate());

    update ENT.document set entityProcessedFlag = 1 where documentId = @itemId
END
GO

```


update_DocumentEntityError

```
CREATE proc [ENT].[update_DocumentEntityError]
(
    @documentId uniqueidentifier,
    @errorMessage nvarchar(max)
)
as
BEGIN
update ENT.document
set entityProcessedFlag = 1,
entityErrorFlag = 1,
errorMessages = @errorMessage
where @documentId = documentId
END
GO
```

process_childCaseNoteResponse

```
CREATE PROC [ENT].[process_childCasenoteResponse]
(
    @itemId uniqueidentifier,
    @blockNumber int,
    @apiResponse nvarchar(max)
)
AS
BEGIN
MERGE ENT.casenoteEntity as tgt
USING
(
    SELECT
        docs.casenoteId as casenoteId,
        @blockNumber as blockNumber,
        ROW_NUMBER() OVER (PARTITION BY casenoteId ORDER BY casenoteId) as entityId,
        JSON_VALUE(entities.value, '$.text') AS text,
        JSON_VALUE(entities.value, '$.category') as category,
        JSON_VALUE(entities.value, '$.subcategory') as subCategory,
        cast(JSON_VALUE(entities.value, '$.offset') as int) as offset,
        cast(JSON_VALUE(entities.value, '$.length') as int) as length,
        cast(JSON_VALUE(entities.value, '$.confidenceScore') as decimal(10, 2)) as confidenceScore
    FROM ENT.casenote docs
    CROSS APPLY OPENJSON(@apiResponse, '$.results.documents[0].entities') entities
    WHERE casenoteId = @itemId
) AS src
ON src.casenoteId = tgt.casenoteId and src.entityId = tgt.entityId and src.blockNumber =
tgt.blockNumber
WHEN NOT MATCHED BY TARGET THEN
    INSERT(casenoteId, blockNumber, entityId, text, category, subCategory, offset, length,
confidenceScore, modifiedAt)
    VALUES (src.casenoteId,src.blockNumber, src.entityId, src.text, src.category, src.subCategory,
src.offset, src.length, src.confidenceScore, getdate());

UPDATE ENT.casenote
SET
    entityProcessedFlag = 1,
    modifiedAt = getdate()
WHERE casenoteId = @itemId

END
GO
```

Process_childDocumentResponse

```
CREATE PROC [ENT].[process_childDocumentResponse]
(
    @itemId uniqueidentifier,
    @blockNumber int,
    @apiResponse nvarchar(max)
)
AS
BEGIN
MERGE ENT.documentEntity as tgt
USING
(
    SELECT
        docs.documentId as documentId,
        @blockNumber as blockNumber,
        ROW_NUMBER() OVER (PARTITION BY documentId ORDER BY documentId) as entityId,
        JSON_VALUE(entities.value, '$.text') AS text,
        JSON_VALUE(entities.value, '$.category') as category,
        JSON_VALUE(entities.value, '$.subcategory') as subCategory,
        cast(JSON_VALUE(entities.value, '$.offset') as int) as offset,
        cast(JSON_VALUE(entities.value, '$.length') as int) as length,
        cast(JSON_VALUE(entities.value, '$.confidenceScore') as decimal(10, 2)) as confidenceScore
    FROM ENT.document docs
    CROSS APPLY OPENJSON(@apiResponse, '$.results.documents[0].entities') entities
    where documentId = @itemId
) AS src
ON src.documentId = tgt.documentId and src.entityId = tgt.entityId and src.blockNumber =
tgt.blockNumber
WHEN NOT MATCHED BY TARGET THEN
    INSERT(documentId, blockNumber, entityId, text, category, subCategory, offset, length,
confidenceScore, modifiedAt)
    VALUES(src.documentId,src.blockNumber, src.entityId, src.text, src.category, src.subCategory,
src.offset, src.length, src.confidenceScore, getdate());

UPDATE ENT.document
    SET entityProcessedFlag = 1,
        modifiedAt = getdate()
WHERE documentId = @itemId
END
GO
```

Functions

fnCasenotesToEntityRecognise

```
CREATE FUNCTION [ENT].[fnCasenotesToEntityRecognise]
(@numRows int, @maxBlockLength int = 124000)
RETURNS TABLE
AS
RETURN SELECT
    casenoteId as itemId,
    ss.blockNumber,
    JSON_OBJECT('id':casenoteId, 'text': ss.[text]) as jsonText
FROM ENT.casenote cn
-- Cross apply with the splitting of the long string into individual rows
CROSS APPLY ENT.splitString(cn.notes, @maxBlockLength) ss
WHERE cn.casenoteId in (select top (@numRows) cn2.casenoteId FROM
    ENT.casenote cn2
    WHERE TRIM(ISNULL(cn2.notes, '')) <> '' -- There's some text worth processing
    AND cn2.entityProcessedFlag = 0 -- It's been processed
    AND cn2.entityErrorFlag = 0 -- And there are no errors
)
GO
```

```

fnDocumentsToEntityRecognise
CREATE FUNCTION [ENT].[fnDocumentsToEntityRecognise]
(@numRows int, @maxBlockLength int = 124000)
RETURNS TABLE
AS
RETURN SELECT
    doc.documentId AS itemId,
    ss.blockNumber,
    JSON_OBJECT('id':documentId, 'text': ss.[text]) as jsonText
FROM ENT.document doc
-- Cross apply with the splitting of the long string into individual rows
CROSS APPLY ENT.splitString(doc.[text], @maxBlockLength) ss
WHERE doc.documentId in (select top (@numRows) doc2.documentId
    from ENT.document doc2
    WHERE
        TRIM(ISNULL(doc2.[text], '')) <> '' -- we have text for the document
        AND doc2.entityProcessedFlag = 0 -- not been processed already
        AND doc2.entityErrorFlag = 0 -- and no errors flagged
        AND doc2.OCRErrorFlag = 0 -- and no OCR errors either - CHECK WHETHER WE NEED THIS.
)
GO

```

```

splitString
CREATE function [ENT].[splitString]
(
    @str nvarchar(max),
    @length int
)
RETURNS @Results TABLE([text] nvarchar(max), blockNumber int)
AS
BEGIN
    DECLARE @Sequence INT = 1

    DECLARE @s nvarchar(max)
    WHILE len(@str) > 0
    BEGIN
        -- Take the first @length characters
        if len(@str) > @length
        begin
            SET @s = left(@str, @length)
            if charindex(' ', @s) > 0
            BEGIN
                -- Find the position of the last space, and chop string back to that position
                DECLARE @lastSpace INT
                set @lastSpace = len(@s) - charindex(' ', reverse(@s)) + 1
                set @s = LEFT(@s, @lastSpace)
            END
        end
        ELSE
            set @s = @str
        -- Insert those values into the table
        INSERT @Results VALUES (@s, @Sequence)
        IF(len(@str)<@length)
        BREAK

        SET @str = right(@str, len(@str) - len(@s))
        SET @Sequence = @Sequence + 1
        --INSERT @Results VALUES (':' +@str, @Sequence)
        if @Sequence > 100
            break
    END
    RETURN
END
GO

```

getNetworkNodes

```
CREATE FUNCTION [PBI].[getNetworkNodes](@internalChildId int)
```

```
RETURNS @results TABLE
```

```
(
    internalChildId INT NOT NULL,
    [Source] NVARCHAR(MAX) NOT NULL,
    [Target] NVARCHAR(MAX) NOT NULL,
    [Type] NVARCHAR(MAX) NOT NULL,
    [Role] NVARCHAR(MAX) NOT NULL,
    [Date] DATE NOT NULL,
    [ConfidenceScore] [decimal](10, 2) NOT NULL,
    SourceWeight DECIMAL(10, 2) NOT NULL,
    TargetWeight DECIMAL(10, 2) NOT NULL
```

```
)
```

```
AS
```

```
BEGIN
```

```
; WITH CTE_NODES AS
```

```
(
```

```
-- Child->caseworker links
```

```
SELECT child.internalChildId, CONCAT(child.forename, ' ', child.surname) as Source,
       CONCAT(cw.forename, ' ', cw.surname) as [Target],
       'Allocated Worker' as Type,
       cw.[role] as [Role],
       CAST(getdate() as date) as [Date],
       1.0 as ConfidenceScore
```

```
from ENT.child child join
```

```
    ENT.caseworker cw on child.internalChildId = cw.internalChildId
```

```
WHERE child.internalChildId = @internalChildId
```

```
UNION
```

```
-- Child->significant adult links
```

```
SELECT child.internalChildId, CONCAT(child.forename, ' ', child.surname) as Source,
       CONCAT(sa.forename, ' ', sa.surname) as [Target],
       'Significant Adult' as Type,
       sa.relationship as [Role],
       CAST(getdate() as date) as [Date],
       1.0 as ConfidenceScore
```

```
from ENT.child child join
```

```
    ENT.significantAdult sa on child.internalChildId = sa.internalChildId
```

```
WHERE child.internalChildId = @internalChildId
```

```
UNION
```

```
-- Child->document links
```

```
SELECT child.internalChildId, CONCAT(child.forename, ' ', child.surname) as Source,
       ISNULL(document.notes, 'Document ' + CAST(internalDocId as NVARCHAR(50))) as [Target],
       'Document' as Type,
       'Document' as [Role],
       CAST(document.docDate as date) as [Date],
       1.0 as ConfidenceScore
```

```
from ENT.child child join
```

```
    ENT.document document on child.internalChildId = document.internalChildId
```

```
WHERE child.internalChildId = @internalChildId
```

```
UNION
```

```
-- Document->entity links
```

```
SELECT document.internalChildId, ISNULL(document.notes, 'Document ' + CAST(internalDocId as
NVARCHAR(50))) as Source,
       entity.[text] as [Target],
       'Entity' as Type,
       entity.category as [Role],
       CAST(document.docDate as date) as [Date],
       entity.confidenceScore as ConfidenceScore
```

```
from ENT.document document
```

```
    JOIN ENT.documentEntity entity on document.documentId = entity.documentId
```

```
WHERE document.internalChildId = @internalChildId
```

```
AND confidenceScore >= 0.8
```

```
AND NOT entity.category IN ('Skill', 'Location', 'Product', 'Event', 'DateTime', 'Quantity',
'PhoneNumber', 'Email', 'URL')
```

```

UNION
-- Child->Casenote links
SELECT child.internalChildId, CONCAT(child.forename, ' ', child.surname) as Source,
       'Case Note ' + CAST(noteId as NVARCHAR(50)) as [Target],
       'Case Note' as Type,
       'Case Note' as [Role],
       CAST(casenote.contactDate as date) as [Date],
       1.0 as ConfidenceScore
from ENT.child child join
     ENT.casenote casenote on child.internalChildId = casenote.internalChildId
WHERE child.internalChildId = @internalChildId
UNION
-- Casenote->entity links
SELECT casenote.internalChildId, 'Case Note ' + CAST(noteId as NVARCHAR(50)) as Source,
       entity.[text] as [Target],
       'Entity' as Type,
       entity.category as [Role],
       CAST(casenote.contactDate as date) as [Date],
       entity.confidenceScore as ConfidenceScore
from ENT.casenote casenote
JOIN ENT.casenoteEntity entity on casenote.casenoteId = entity.casenoteId
WHERE casenote.internalChildId = @internalChildId
AND confidenceScore >= 0.8
AND NOT entity.category IN ('Skill', 'Location', 'Product', 'Event', 'DateTime', 'Quantity',
'PhoneNumber', 'Email', 'URL')
)
INSERT @results
(internalChildId, [Source], [Target], [Role], [Type], [Date], ConfidenceScore, SourceWeight,
TargetWeight)
SELECT
    internalChildId, [Source], [Target], [Role], [Type], [Date], ConfidenceScore,
    CAST(COUNT(*) OVER(PARTITION BY Source) as decimal(10, 2)),
    CAST(COUNT(*) OVER(PARTITION BY Target) as decimal(10, 2)) as TargetWeight
FROM CTE_NODES n
-- Set the node weights
UPDATE @results
set TargetWeight = GREATEST(1.0, TargetWeight / (SELECT MAX(TargetWeight) from @results) * 10),
    SourceWeight = GREATEST(1.0, SourceWeight / (SELECT MAX(SourceWeight) from @results) * 10)
RETURN
END
GO

```

getNetworkNodesNoCaseNotes

```

CREATE FUNCTION [PBI].[getNetworkNodesNoCaseNotes](@internalChildId int)
RETURNS @results TABLE
(
    internalChildId INT NOT NULL,
    [Source] NVARCHAR(MAX) NOT NULL,
    [Target] NVARCHAR(MAX) NOT NULL,
    [Type] NVARCHAR(MAX) NOT NULL,
    [Role] NVARCHAR(MAX) NOT NULL,
    [Date] DATE NOT NULL,
    [ConfidenceScore] [decimal](10, 2) NOT NULL,
    SourceWeight DECIMAL(10, 2) NOT NULL,
    TargetWeight DECIMAL(10, 2) NOT NULL
)
AS
BEGIN
; WITH CTE_NODES AS
(
    -- Child->caseworker links
    SELECT child.internalChildId, CONCAT(child.forename, ' ', child.surname) as Source,
           CONCAT(cw.forename, ' ', cw.surname) as [Target],
           'Allocated Worker' as Type,
           cw.[role] as [Role],

```

```

        CAST(getdate() as date) as [Date],
        1.00 as ConfidenceScore
from ENT.child child join
    ENT.caseworker cw on child.internalChildId = cw.internalChildId
WHERE child.internalChildId = @internalChildId
UNION
-- Child->significant adult links
SELECT child.internalChildId, CONCAT(child.forename, ' ', child.surname) as Source,
    CONCAT(sa.forename, ' ', sa.surname) as [Target],
    'Significant Adult' as Type,
    sa.relationship as [Role],
    CAST(getdate() as date) as [Date],
    1.00 as ConfidenceScore
from ENT.child child join
    ENT.significantAdult sa on child.internalChildId = sa.internalChildId
WHERE child.internalChildId = @internalChildId
UNION
-- Child->document links
SELECT child.internalChildId, CONCAT(child.forename, ' ', child.surname) as Source,
    ISNULL(document.notes, 'Document ' + CAST(internalDocId as NVARCHAR(50))) as [Target],
    'Document' as Type,
    'Document' as [Role],
    CAST(document.docDate as date) as [Date],
    1.00 as ConfidenceScore
from ENT.child child join
    ENT.document document on child.internalChildId = document.internalChildId
WHERE child.internalChildId = @internalChildId
UNION
-- Document->entity links
SELECT document.internalChildId, ISNULL(document.notes, 'Document ' + CAST(internalDocId as
NVARCHAR(50))) as Source,
    entity.[text] as [Target],
    'Entity' as Type,
    entity.category as [Role],
    CAST(document.docDate as date) as [Date],
    entity.confidenceScore as ConfidenceScore
from ENT.document document
    JOIN ENT.documentEntity entity on document.documentId = entity.documentId
WHERE document.internalChildId = @internalChildId
AND confidenceScore >= 0.8
AND NOT entity.category IN ('Skill', 'Location', 'Product', 'Event', 'DateTime', 'Quantity',
'PhoneNumber', 'Email', 'URL')
UNION
-- Child->Casenote links
SELECT child.internalChildId, CONCAT(child.forename, ' ', child.surname) as Source,
    entity.[text] as [Target],
    'Case Note Entity' as Type,
    entity.category as [Role],
    CAST(casenote.contactDate as date) as [Date],
    entity.confidenceScore as ConfidenceScore
from ENT.child child
    JOIN ENT.casenote casenote on child.internalChildId = casenote.internalChildId
    JOIN ENT.casenoteEntity entity on casenote.casenoteId = entity.casenoteId
WHERE child.internalChildId = @internalChildId
AND confidenceScore >= 0.8
AND NOT entity.category IN ('Skill', 'Location', 'Product', 'Event', 'DateTime', 'Quantity',
'PhoneNumber', 'Email', 'URL')
)
INSERT @results
(internalChildId, [Source], [Target], [Role], [Type], [Date], ConfidenceScore, SourceWeight,
TargetWeight)
SELECT
    internalChildId, [Source], [Target], [Role], [Type], [Date], ConfidenceScore,
    CAST(COUNT(*) OVER(PARTITION BY Source) as decimal(10, 2)),
    CAST(COUNT(*) OVER(PARTITION BY Target) as decimal(10, 2)) as TargetWeight
FROM CTE_NODES n

```

```
UPDATE @results
set TargetWeight = GREATEST(1.0, TargetWeight / (SELECT MAX(TargetWeight) from @results) * 10),
    SourceWeight = GREATEST(1.0, SourceWeight / (SELECT MAX(SourceWeight) from @results) * 10)

RETURN
END
GO
```

Scrap book notes

```
select @@servername[Server Name]
, db_name() [DB name]
, u.name [DB Role]
, u2.name [Member Name]
from sys.database_role_members m
    join sys.database_principals u
        on m.role_principal_id = u.principal_id
    join sys.database_principals u2
        on m.member_principal_id = u2.principal_id
where u.name = 'db_owner'
order by [Member Name]
```

The Azure resource group must be set up as a user on the database as it is this 'entity' that calls the database from data factory. The process does not use the users credentials. The linked service entity will require read, write and execute permissions.

The screenshot shows the Azure Data Factory portal interface. On the left, the 'Linked services' tab is selected in the navigation pane. The main area displays a table of linked services:

Name	Type	Related
LinkAIServices	REST	2
LinkLcsSource	Azure SQL Database	1
LinkSqlDataMiningOut	Azure SQL Database	8
LinkStLcsData	Azure Blob Storage	0

The 'LinkSqlDataMiningOut' service is selected, and the 'Edit linked service' pane on the right is open. It shows the configuration for an Azure SQL Database connection:

- Connection string:** Azure Key Vault
- Account selection method:** From Azure subscription
- Fully qualified domain name:** sql-data-mining.database.windows.net
- Database name:** sqldb-data-mining
- Authentication type:** System Assigned Managed Identity
- Managed identity name:** adf-knowledge-mining-dev
- Managed identity object ID:** 66f455ec-1422-446a-bed8-a0f3e4e4434a
- Always encrypted:** Checked
- Key store authentication type:** System Assigned Managed Identity

At the bottom of the pane, there are 'Save' and 'Cancel' buttons, and a 'Test connection' link.

```
create user [adf-knowledge-mining-dev] from external provider --WITH DEFAULT_SCHEMA = dbo;

alter role db_datareader add member [adf-knowledge-mining-dev]

alter role db_datawriter add member [adf-knowledge-mining-dev]

alter role db_owner add member [adf-knowledge-mining-dev] -this will supersede the previous
two lines if required.
```


The screenshot shows the Microsoft SQL Server Management Studio interface. The left pane displays the Object Explorer with the 'sql-data-mining' database selected. The central pane shows a SQL query being executed. The bottom pane displays the results of the query in a table format.

Query:

```

1 select @@ServerName [Server Name], DB_NAME() [DB Name], u.name [DB Role], u2.name [Member Name]
2 from sys.database_role_members m
3 join sys.database_principals u on m.role_principal_id = u.principal_id
4 join sys.database_principals u2 on m.member_principal_id = u2.principal_id
5 where u.name = 'db_owner'
6
7
8
9
10 order by [Member Name]

```

Results:

	Server Name	DB Name	DB Role	Member Name
1	sql-data-mining	sql-data-mining	db_owner	dbo
2	sql-data-mining	sql-data-mining	db_owner	Geraint.Edwards@northyorksgovuk.onmicrosoft.com
3	sql-data-mining	sql-data-mining	db_owner	John.Jemson@northyorks.gov.uk
4	sql-data-mining	sql-data-mining	db_owner	Tom.Moore@northyorks.gov.uk

```

create user [SVC-SP13PowerP@northyorks.gov.uk] from external provider WITH DEFAULT_SCHEMA = dbo;
alter role db_datareader add member [SVC-SP13PowerP@northyorks.gov.uk];

```

```

alter role db_datareader drop member [adf-knowledge-mining-dev]

```