GS518 – Integrating External APIs with Logic Apps and Electronic Reporting

In a world of connected systems and compliance-driven automation, businesses often need to call **external APIs** to fetch authentication tokens, retrieve transaction statuses, or communicate with government gateways. Traditionally, this required custom development, but not anymore.

In this article, we'll show how to:

- Trigger an **Azure Logic App** from Dynamics 365 Finance
- Send a request with **custom headers and body** using Electronic Reporting (ER)
- Use **Electronic Messaging (EM)** to orchestrate the request-response lifecycle
- Parse and save the **API response** directly into D365 tables, no custom code

---

🛠️ Step-by-Step Setup of Azure Logic App for Access Token Request

🔁 Scenario:

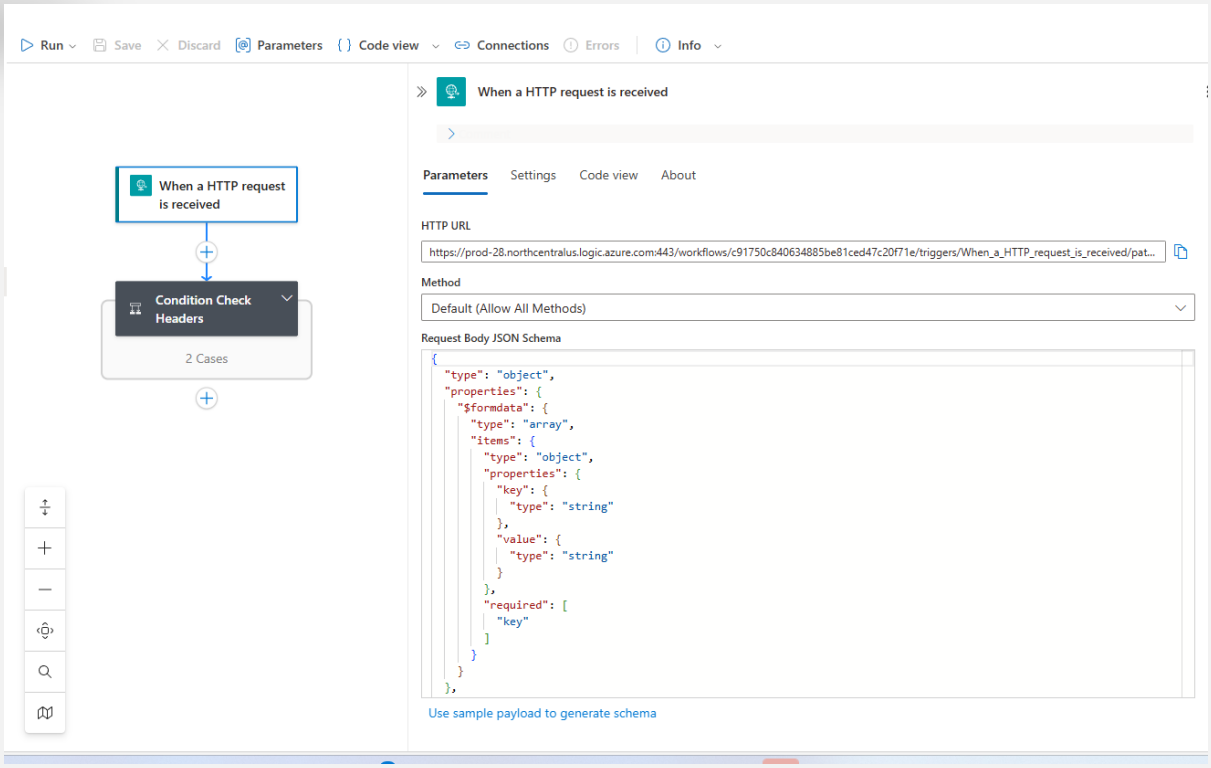
We want Dynamics 365 to send a secure API request to this Logic App to get an **access token**. The Logic App will:

- Check if the request contains correct **security headers**
- If yes, return a sample **token response**
- If no, return an **error message**

☑️ Step 1: When an HTTP request is received

📍 *Purpose*:
This is the **trigger** that starts the Logic App. It waits for a request from an external system (like D365 or Postman).

💡 *Why it's needed*:
You need an endpoint that Dynamics 365 can "call" to request a token.

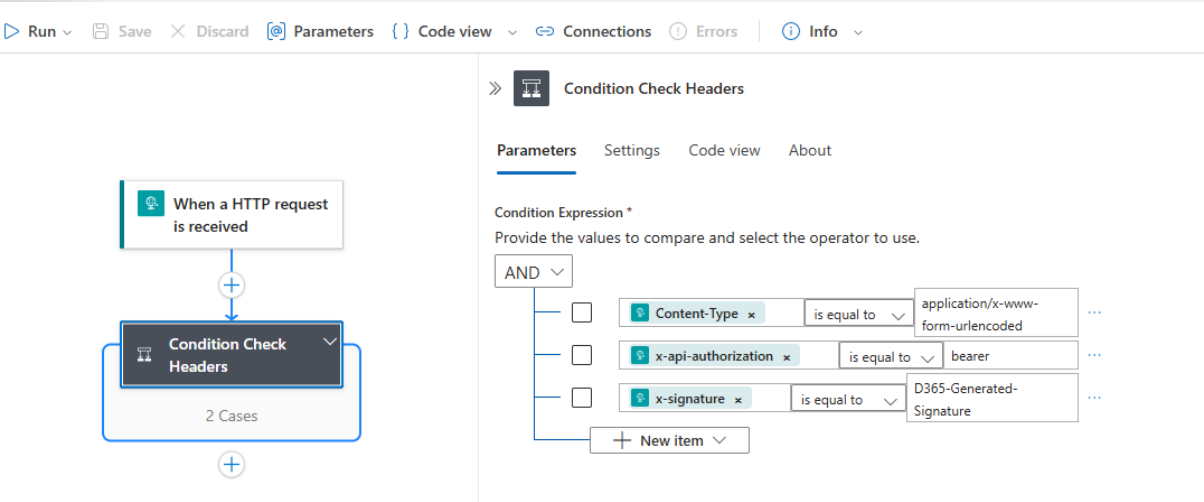


☑ Step 2: Condition Check – Validate Headers

📍 *Purpose*:
These checks whether the incoming request includes all four **mandatory headers**:

Header Name	Expected Value
Content-Type	application/x-www-form-urlencoded
x-api-authorization	bearer
x-signature	D365-Generated-Signature

💡 *Why it's needed*:
To make sure the request is coming from a **trusted source** and has **all required fields** for security and business logic.



☑ Step 3 (If Headers Are Valid): Return Token Response

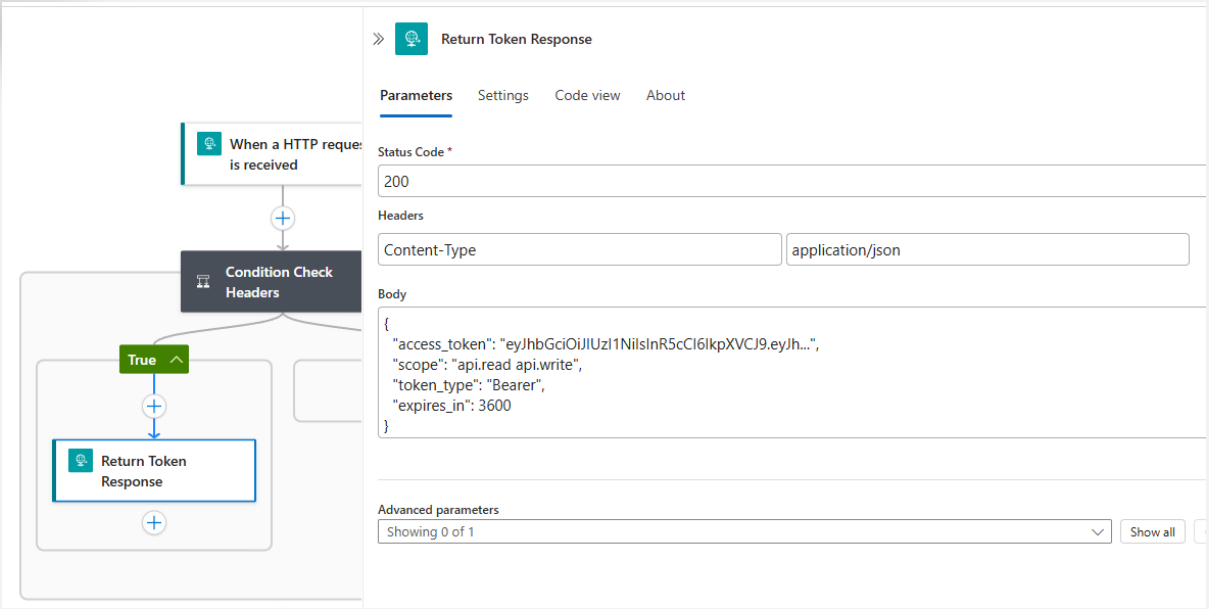
📍 *Purpose*:
If all headers match, the Logic App will respond with a **predefined JSON token**:

json

CopyEdit

```
{
 "access_token": "12c3d4567-token-sample",
 "scope": "am_application_scope default",
 "token_type": "Bearer",
 "expires_in": 2521
}
```

💡 *Why it's needed*:
This token can now be used by Dynamics 365 in further API requests, like submitting invoices or querying third-party services.



❌ Step 4 (If Headers Are Missing or Wrong): Return Error Response

📍 *Purpose*:
If any header is missing or incorrect, the Logic App returns an error like:

json

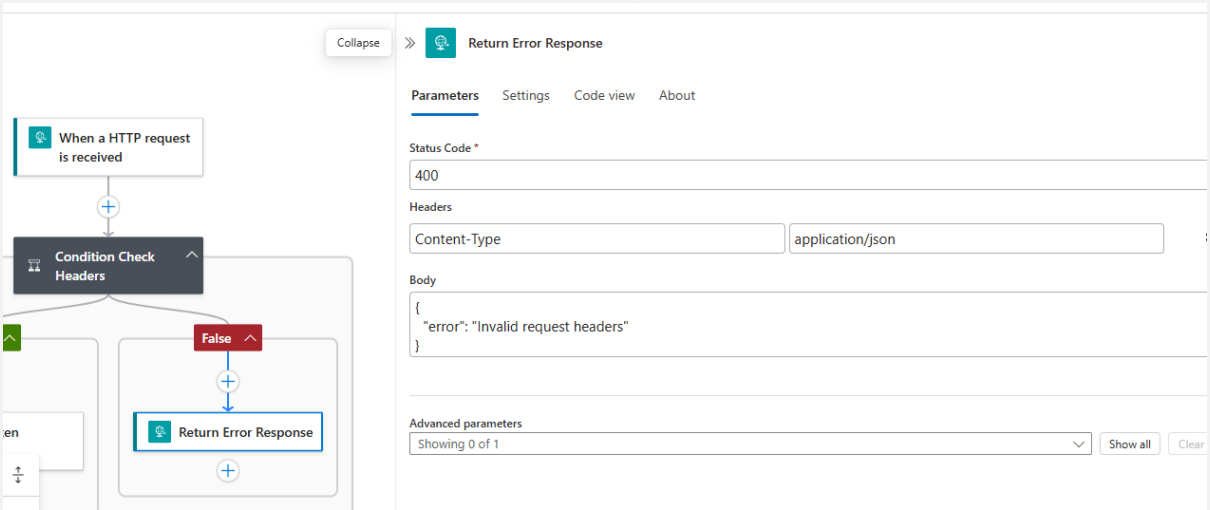
CopyEdit

{

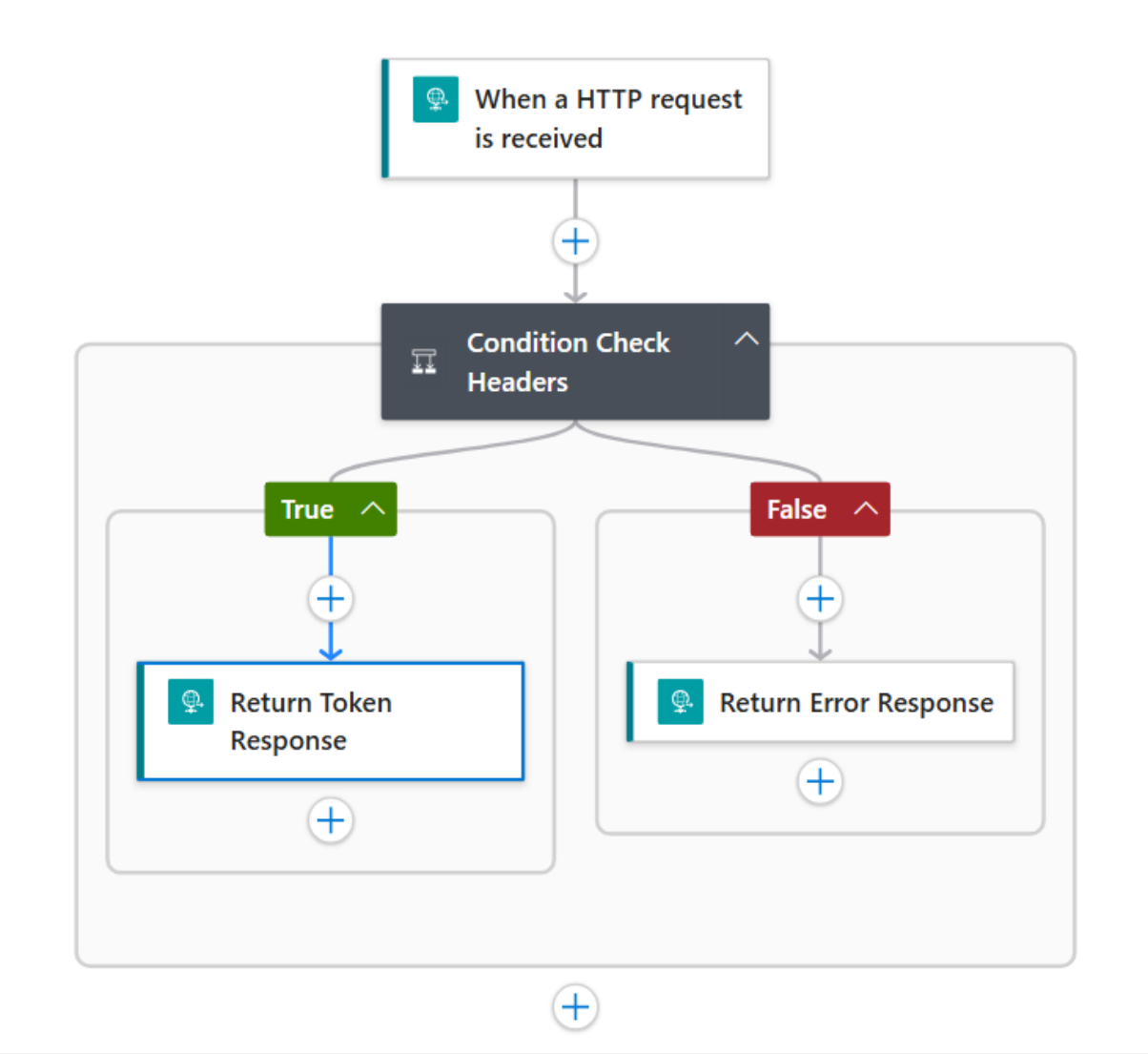
"error": "Invalid request headers"

}

💡 *Why it's needed*:
This prevents unauthorized or incorrect calls from proceeding.



Final Logic app



Step-by-Step Testing with Postman – Validate Logic App Response

After setting up your Azure Logic App it's time to test it using **Postman**. We'll simulate two scenarios:

Scenario 1: Valid Headers – Return Token Response

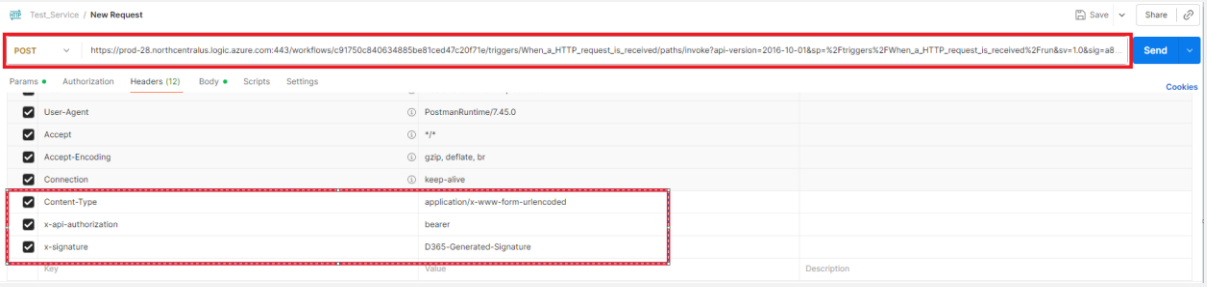
This test checks if the Logic App returns a token when the **correct headers** are provided.

Request Setup in Postman:

- **Method**: POST
- **URL**: *Paste your Logic App HTTP endpoint URL*

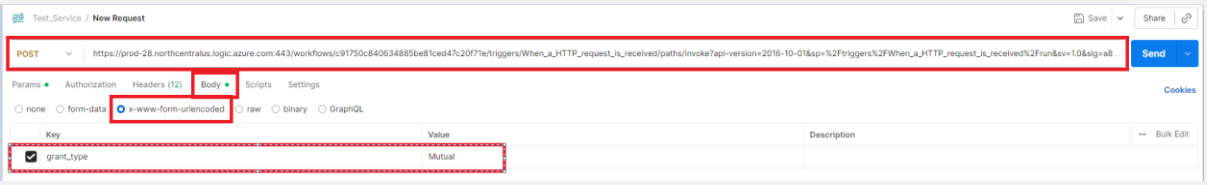
Request Headers:

Header Name	Value
Content-Type	application/x-www-form-urlencoded
x-api-authorization	bearer
x-signature	D365-Generated-Signature



📝 **Body (raw text):**

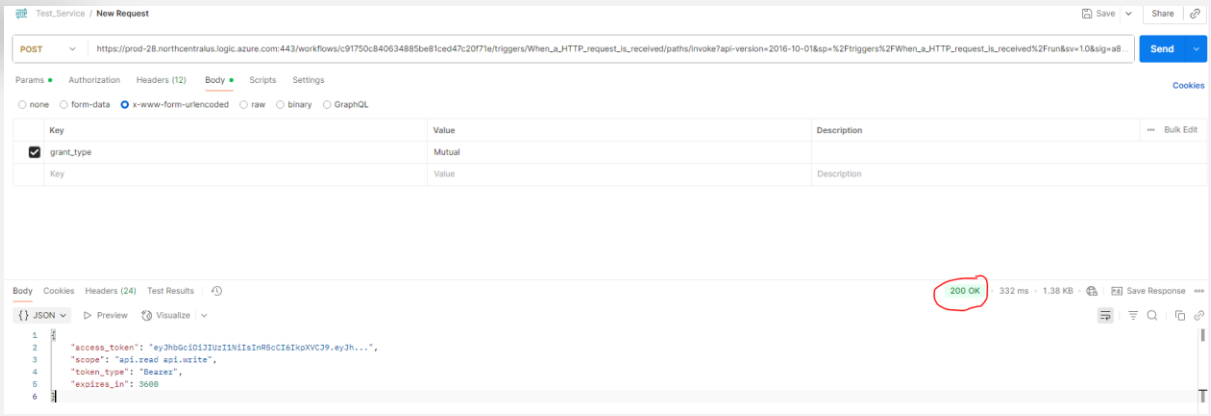
grant_type=Mutual



📩 **Expected Response:**

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJh...",
  "scope": "api.read api.write",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

✅ *This confirms that the Logic App validated the headers and returned the correct token.*



✖ *Scenario 2: Invalid Headers – Return Error Response*

This test checks if the Logic App rejects the request when **headers are missing or wrong**.

🔁 **Change One Header:**

For example, change x-api-authorization to:

makefile

CopyEdit

x-api-authorization: Bearer123 ← Incorrect value

🚨 **Expected Error Response:**

json

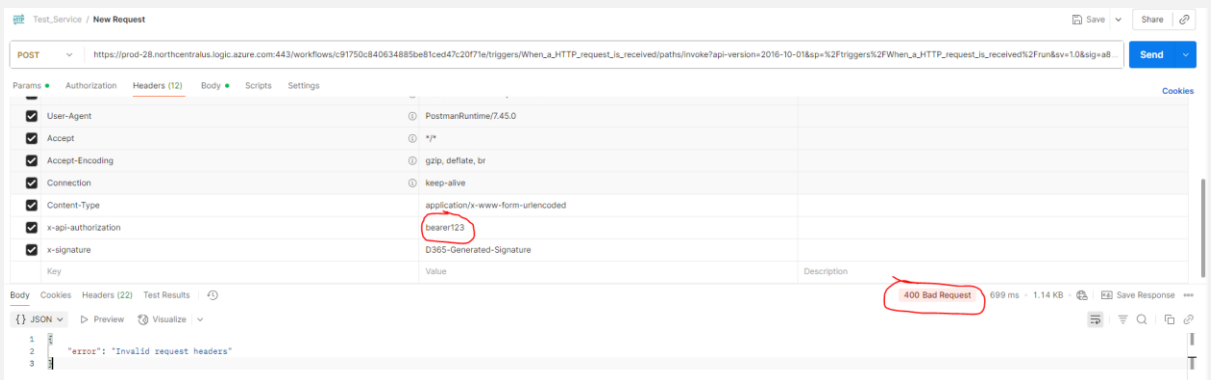
CopyEdit

{

"error": "Invalid request headers"

}

🔒 *This confirms the Logic App's security layer is working as expected.*



⚙ Integrating Logic App with D365FSCM Using ER and Electronic Messaging

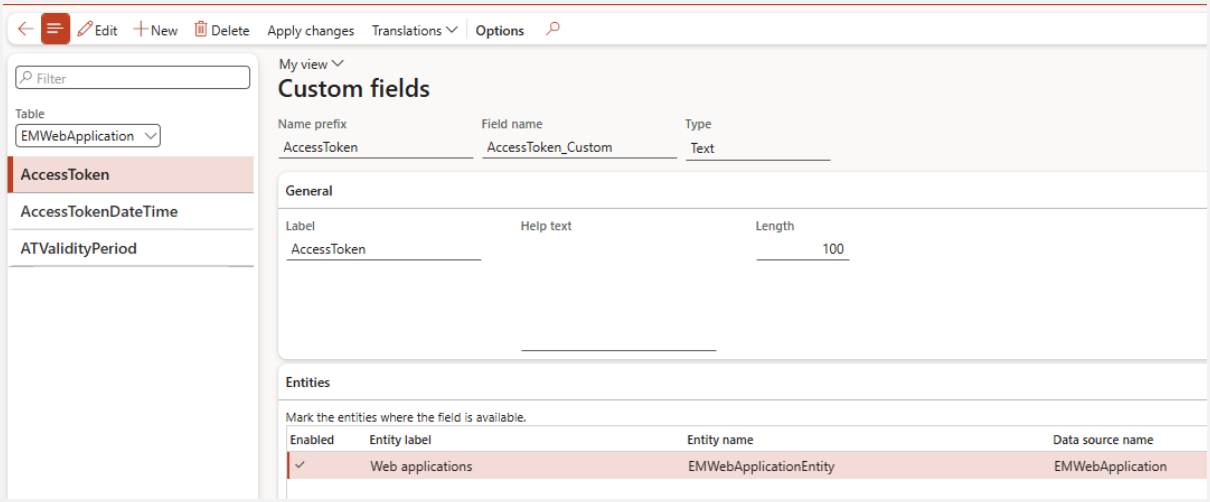
In this section, we'll walk through how to configure Dynamics 365 FSCM to **trigger your Logic App**, receive a **JSON token response**, and **store that response** in your system — all using standard **Electronic Reporting (ER)** and **Electronic Messaging (EM)**, with no X++ code.

We'll split it into 7 clear steps:

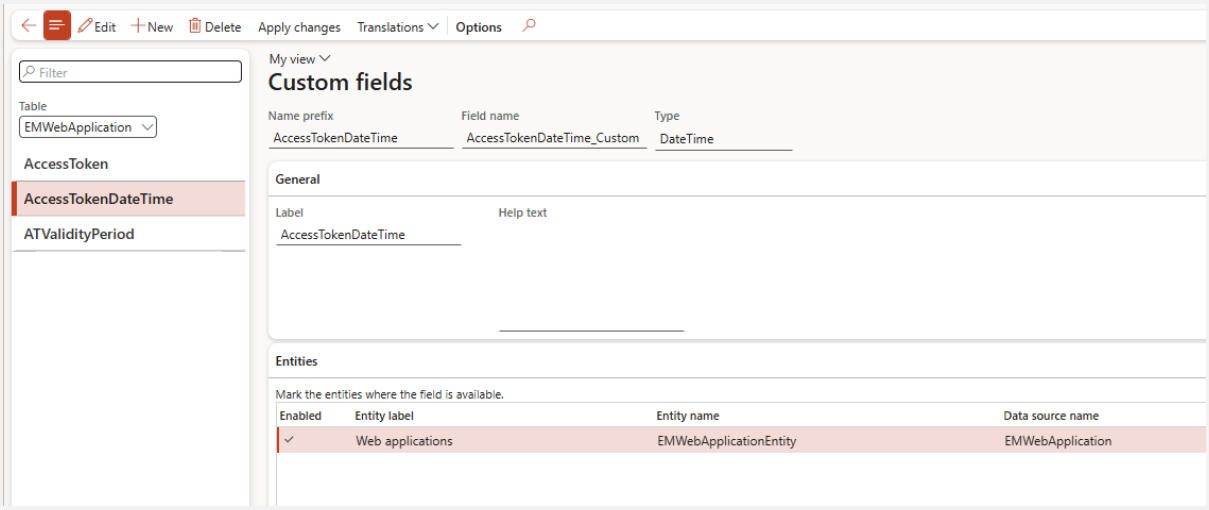
Prerequisite

Create 3 custom fields with the same name. These fields have been used to capture Logic app response

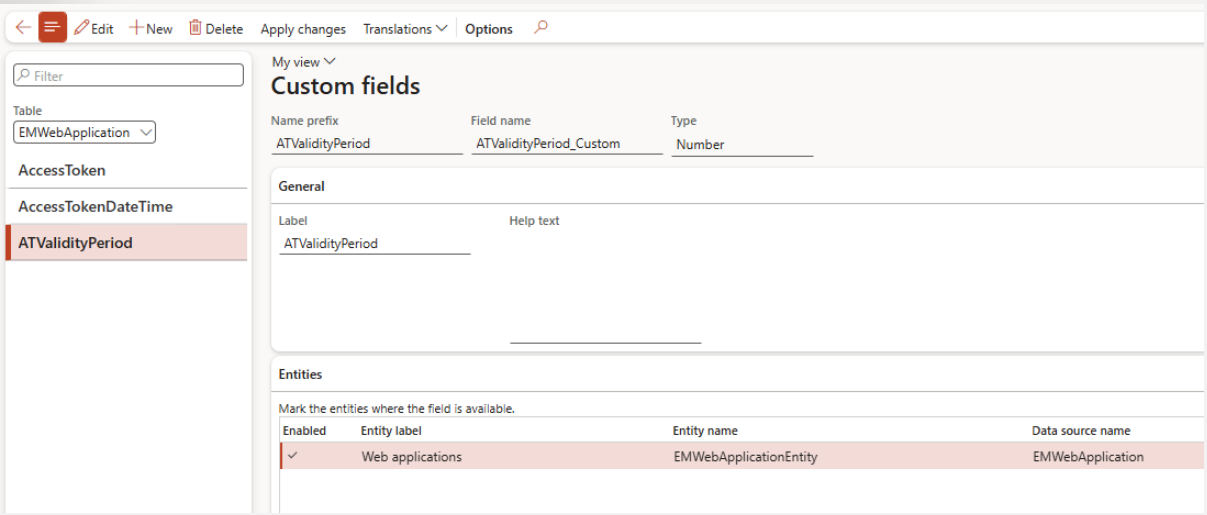
AccessToken



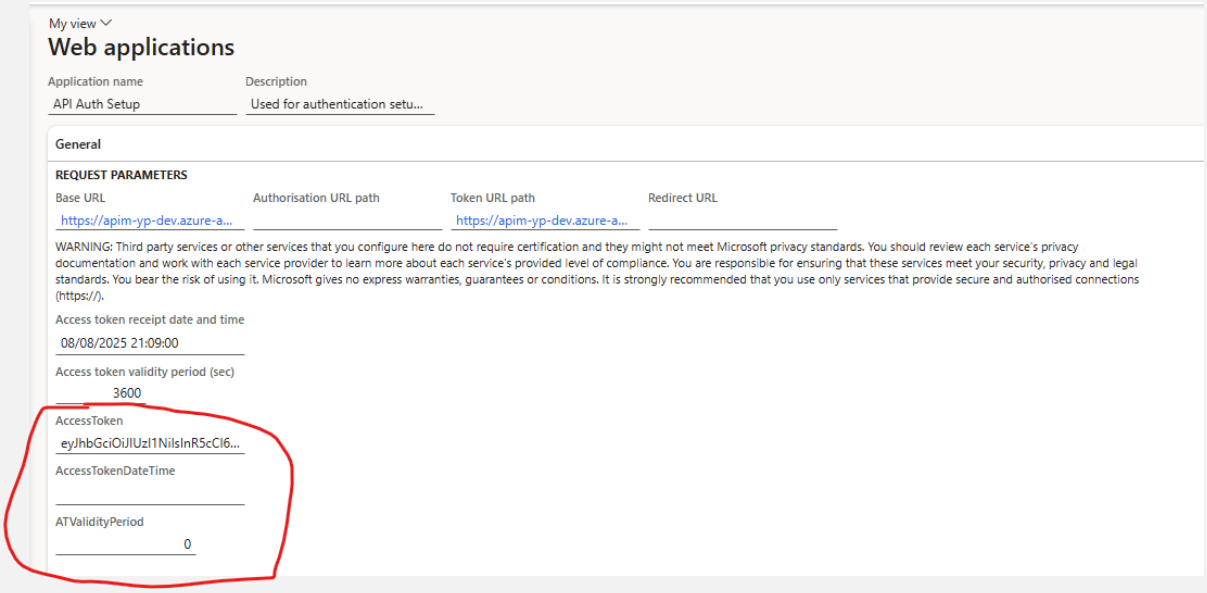
AccessTokenDateTime



ATValidityPeriod



These fields are added on Web application screen

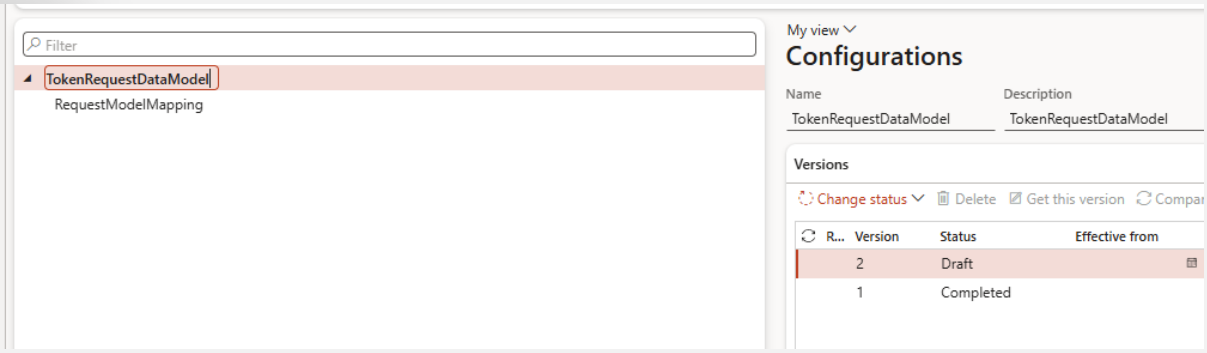


1. ER Formats

Download and Import all configuration in your environment - Download ER Configurations

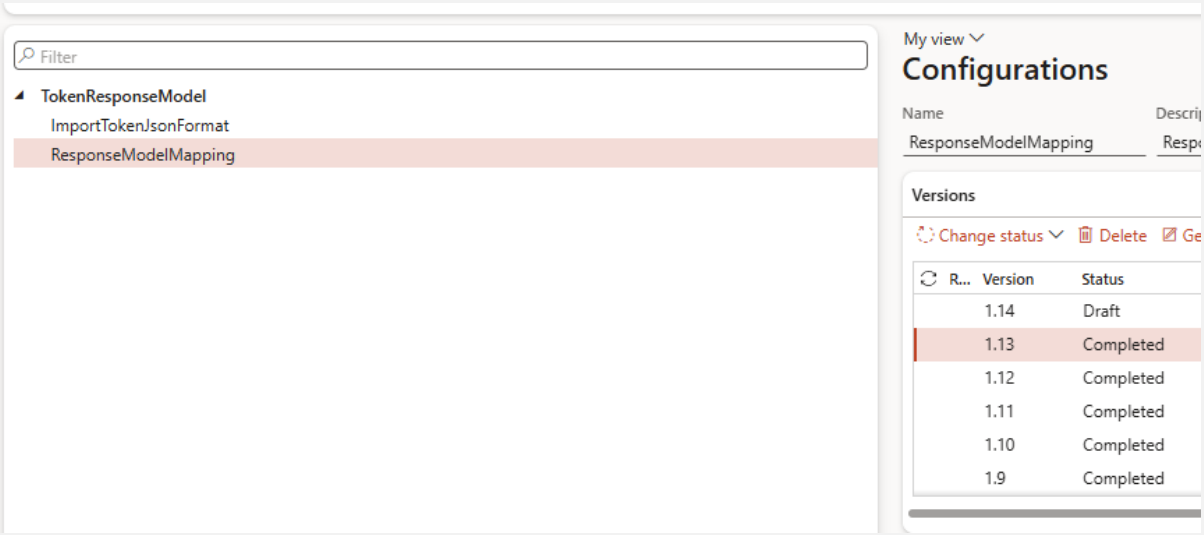
Export Configuration

Defines the JSON request (headers, body) that D365FSCM will send to the Logic App. Following configurations are in above location so make sure you import them



Import Configuration

Defines how the JSON response from the Logic App will be read and mapped into D365 tables. Following configurations are in above location so make sure you import them

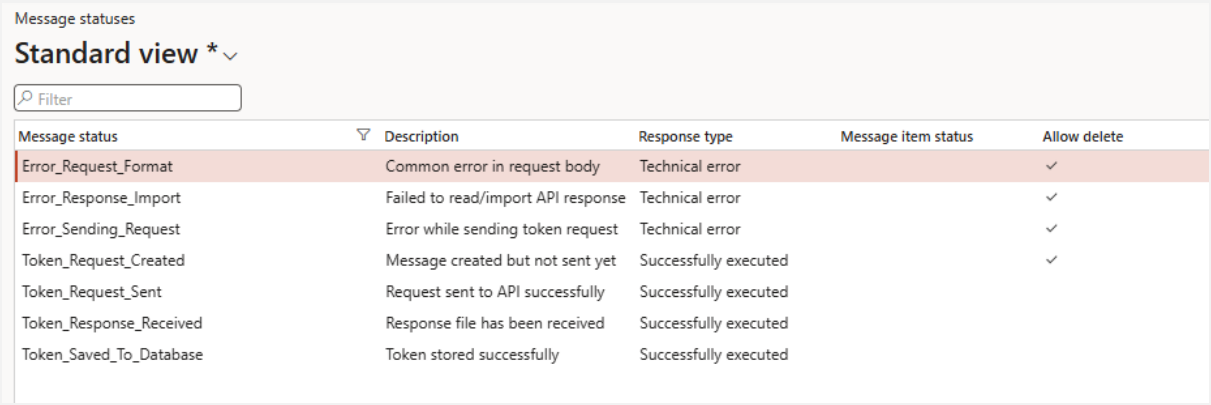


2. Electronic Message Statuses

Tracks each stage of the process. Helps monitor progress and troubleshoot errors.

Message status	Description	Response type	Message item status	Allow delete
Error_Request_Format	Common error in request body	Technical error		Yes
Error_Response_Import	Failed to read/import API response	Technical error		Yes
Error_Sending_Request	Error while sending token request	Technical error		Yes
Token_Request_Created	Message created but not sent yet	Successfully executed		Yes
Token_Request_Sent	Request sent to API successfully	Successfully executed		No
Token_Response_Received	Response file has been received	Successfully executed		No

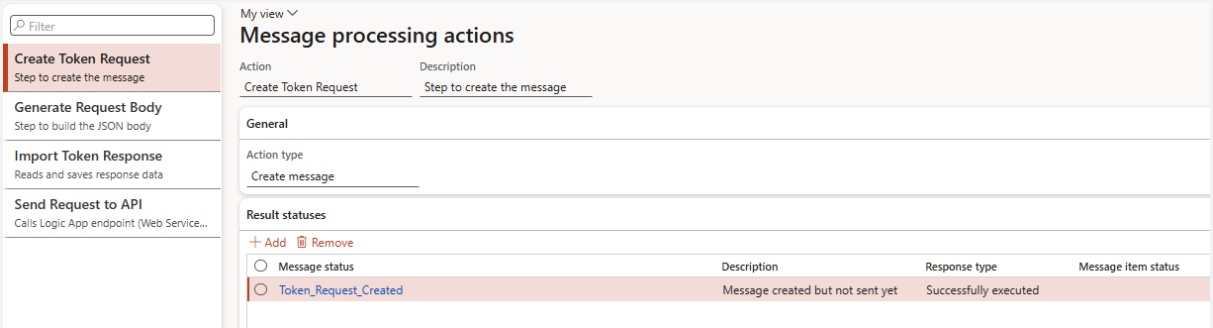
Token_Saved_To_Database	Token stored successfully	Successfully executed	No

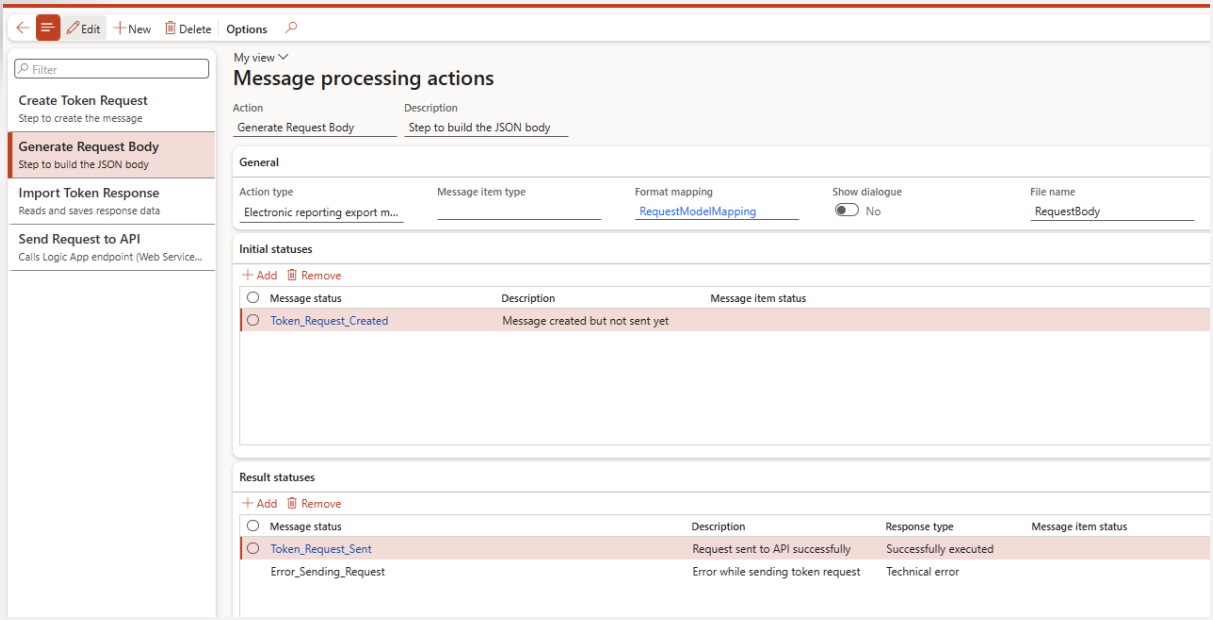


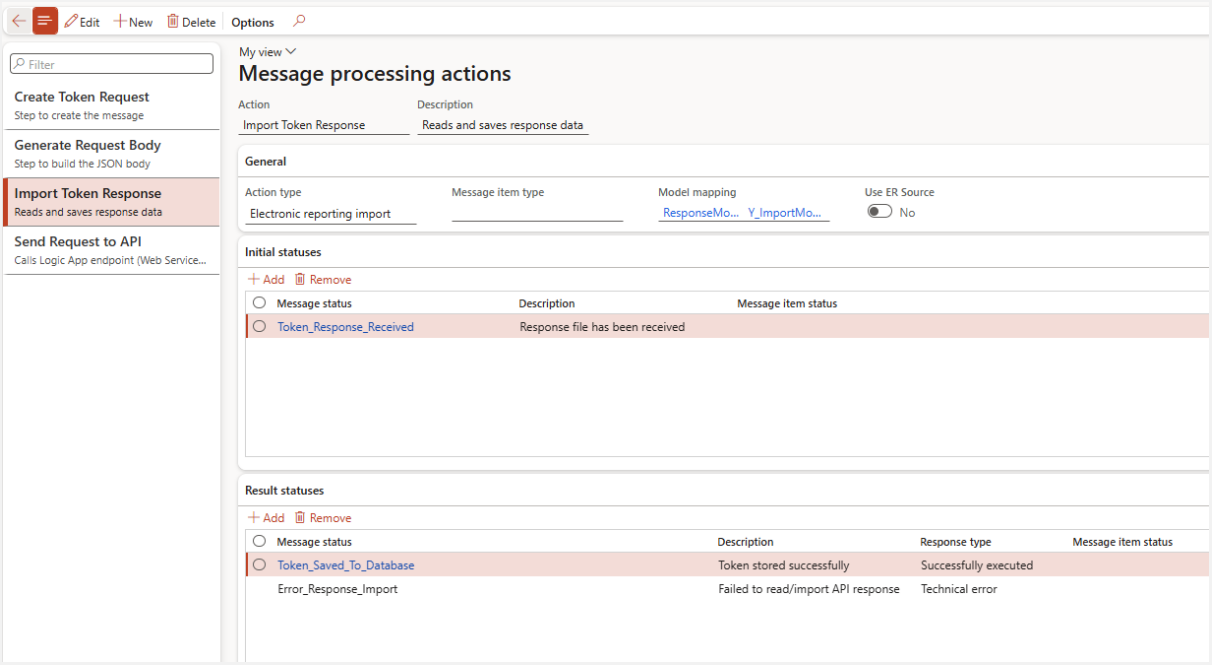
3. Electronic Message Processing Actions

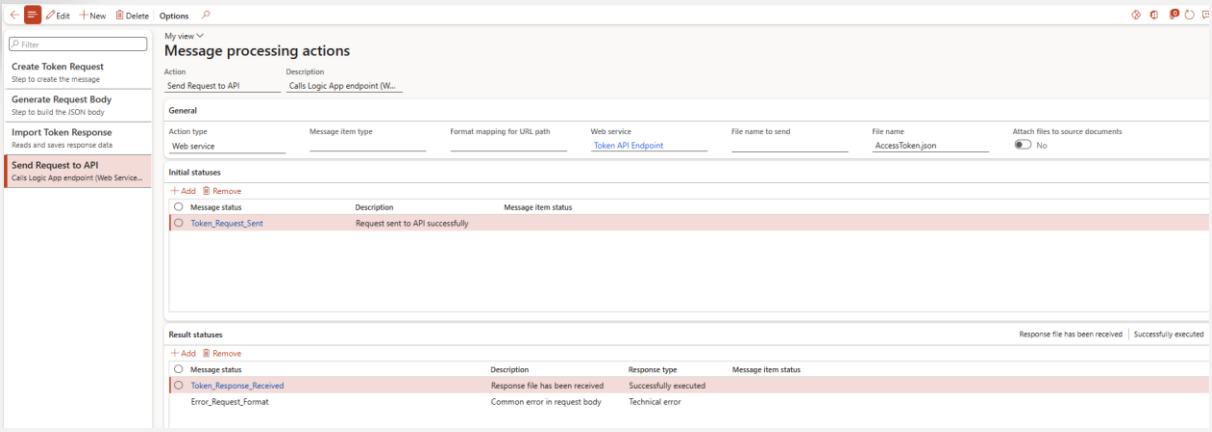
Breaks the process into smaller tasks: create the message, generate request, call the Logic App, and import the response.

Electronic Message Actions	Purpose / Description
Create Token Request	Step to create the message
Generate Request Body	Step to build the JSON body
Import Token Response	Reads and saves response data
Send Request to API	Calls Logic App endpoint (Web Service Action)




Edit
New
Delete
Options
Filter
Create Token Request
Step to create the message
Generate Request Body
Step to build the JSON body
Import Token Response
Reads and saves response data
Send Request to API
Calls Logic App endpoint (Web Service...
My view
Message processing actions
Action
Generate Request Body
Description
Step to build the JSON body
General
Action type
Electronic reporting export m...
Message item type
Format mapping
RequestModelMapping
Show dialogue
No
File name
RequestBody
Initial statuses
Add
Remove
Message status
Description
Message item status
Token_Request_Created
Message created but not sent yet
Result statuses
Add
Remove
Message status
Description
Response type
Message item status
Token_Request_Sent
Request sent to API successfully
Successfully executed
Error_Sending_Request
Error while sending token request
Technical error

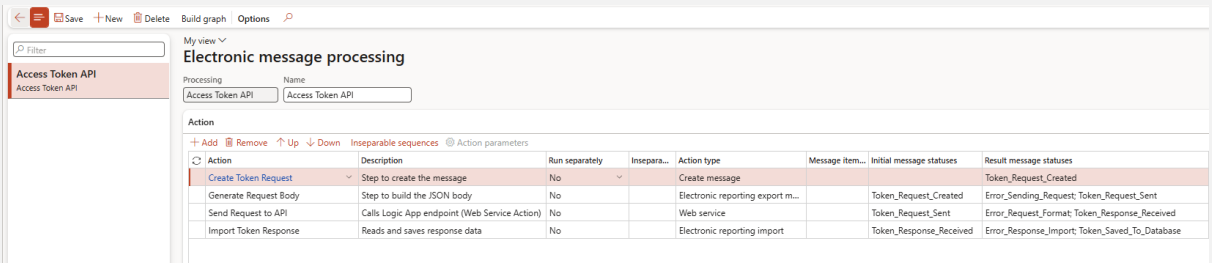

Edit
New
Delete
Options
Filter
Create Token Request
Step to create the message
Generate Request Body
Step to build the JSON body
Import Token Response
Reads and saves response data
Send Request to API
Calls Logic App endpoint (Web Service...
My view
Message processing actions
Action
Import Token Response
Description
Reads and saves response data
General
Action type
Electronic reporting import
Message item type
Model mapping
ResponseMo...
Y_ImportMo...
Use ER Source
No
Initial statuses
Add
Remove
Message status
Description
Message item status
Token_Response_Received
Response file has been received
Result statuses
Add
Remove
Message status
Description
Response type
Message item status
Token_Saved_To_Database
Token stored successfully
Successfully executed
Error_Response_Import
Failed to read/import API response
Technical error



4. Electronic Message Processing (Pipeline)

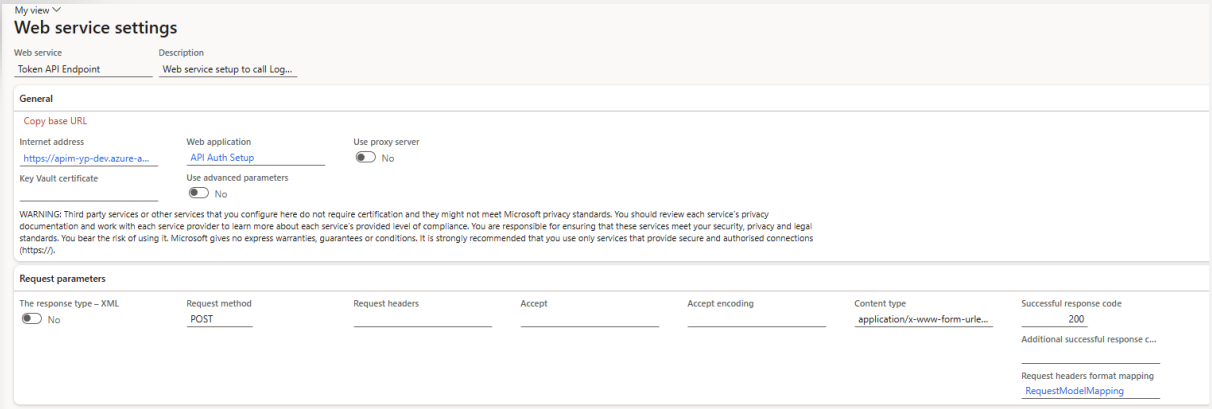
The full execution pipeline – defines the sequence of actions from start to finish so the process runs in the right order.

Action	Description	Run separately	Action type
Create Token Request	Step to create the message	No	Create message
Generate Request Body	Step to build the JSON body	No	Electronic reporting e
Send Request to API	Calls Logic App endpoint (Web Service Action)	No	Web service
Import Token Response	Reads and saves response data	No	Electronic reporting ir



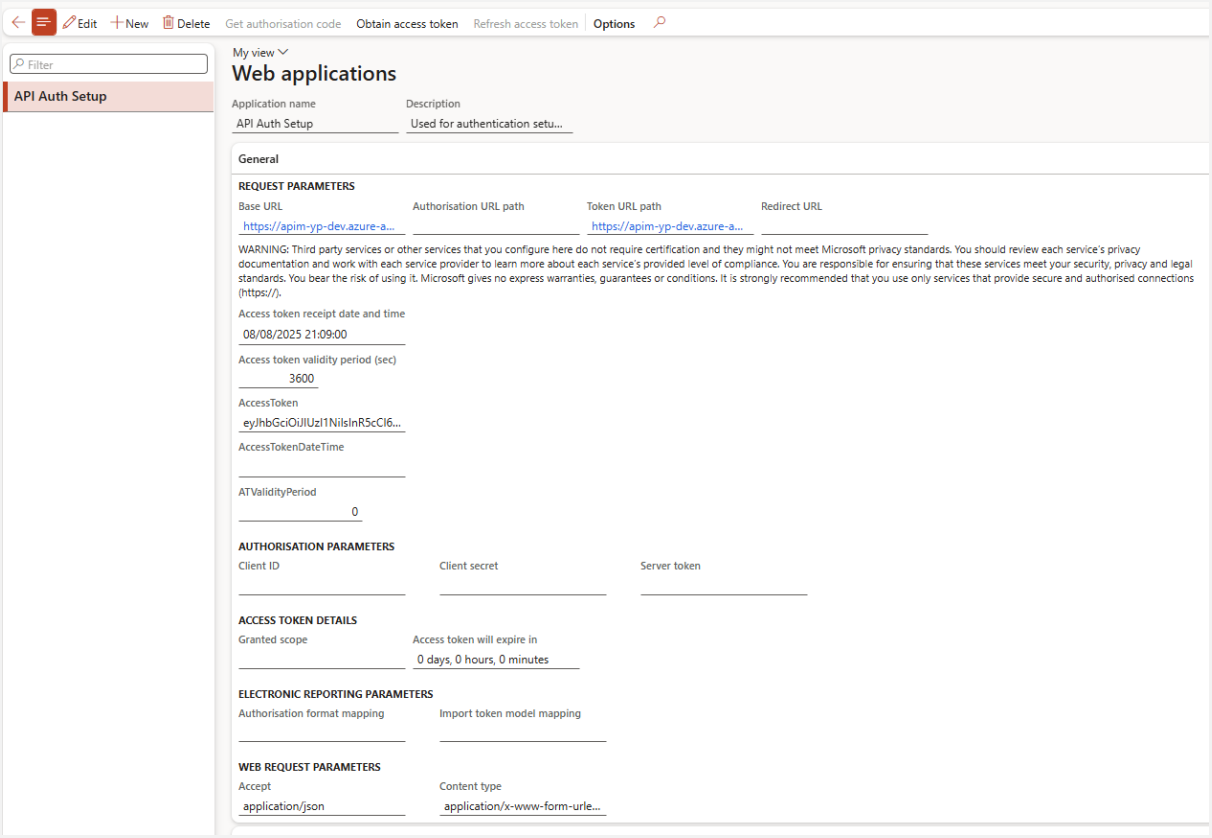
5. Configure Web Service (Secure connection to Webservices)

Stores the Logic App endpoint and connection details, ensuring secure HTTPS calls from D365FSCM.



6. Web Application (Auth2.0)

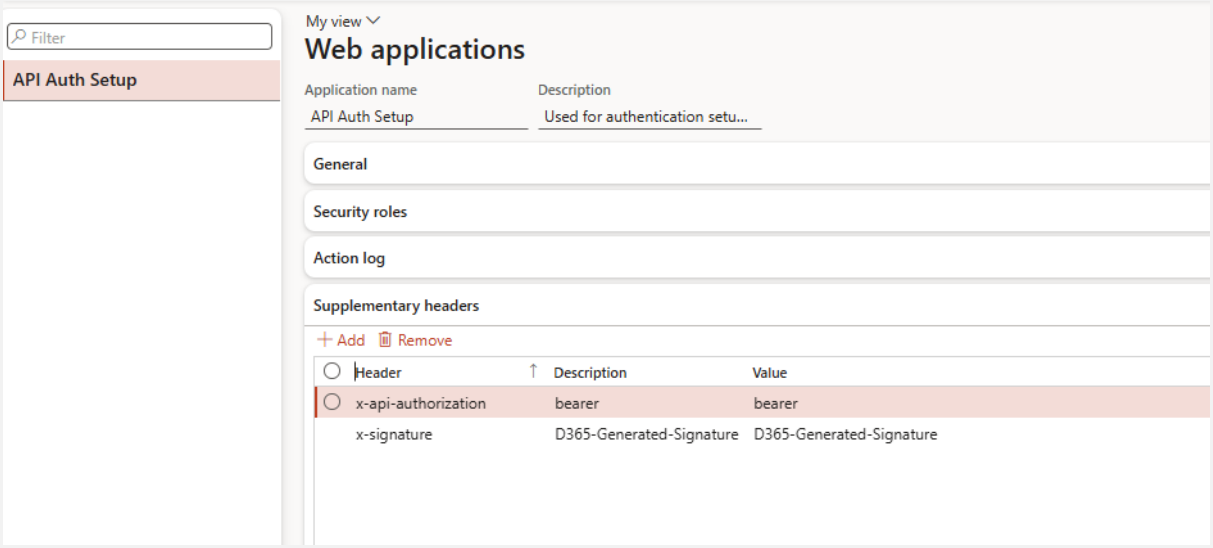
Holds credentials and authentication settings (e.g., OAuth 2.0 parameters) needed to communicate with the API securely.



Add Web applications supplementary headers

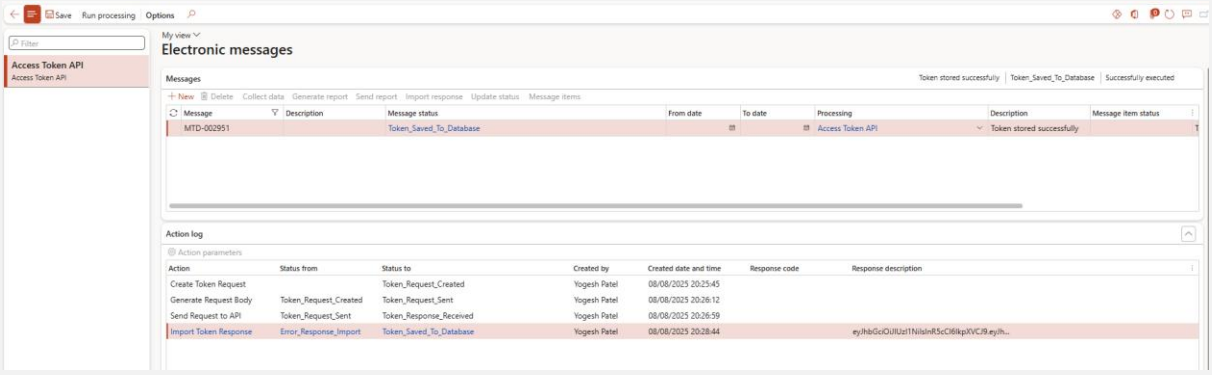
Header	Description	Value
x-api-authorization	bearer	bearer

x-signature	D365-Generated-Signature	D365-Generated-Signature



7. Electronic Message (Container)

The working record for each API call – stores request, response, logs, and attachments for audit and troubleshooting.

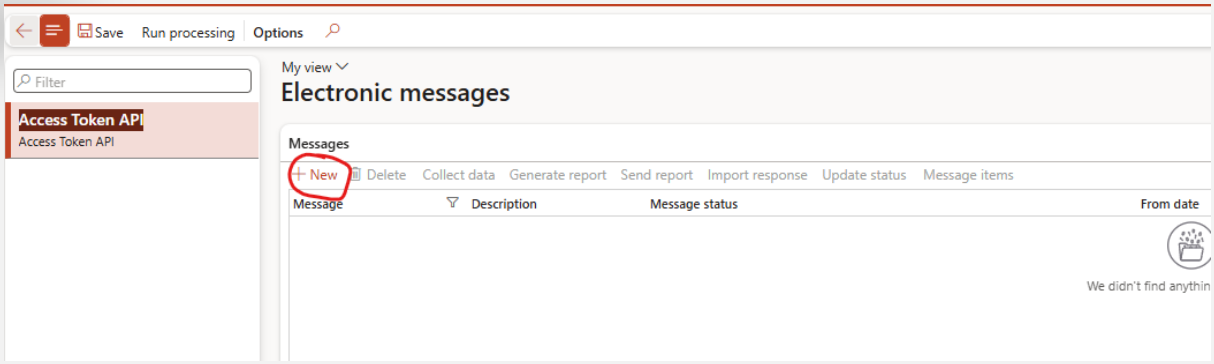


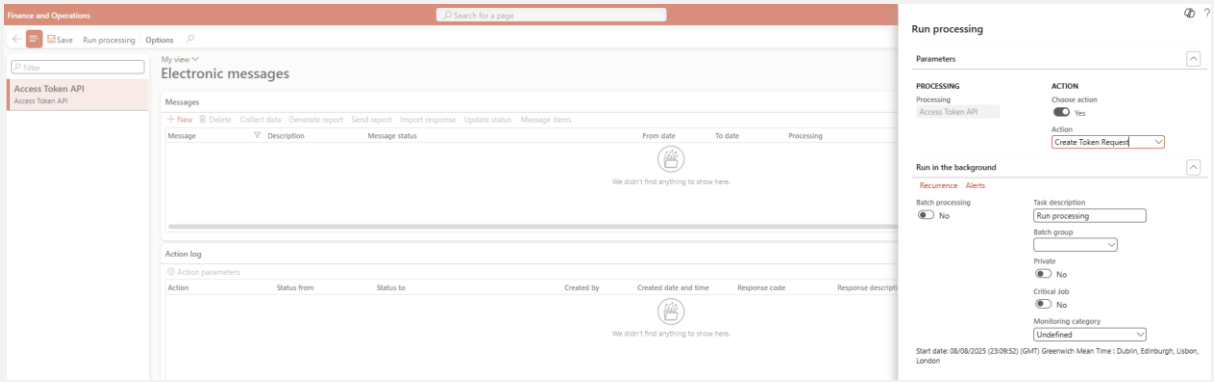
Step-by-Step Testing with Electronic Message – Validate Logic App Response

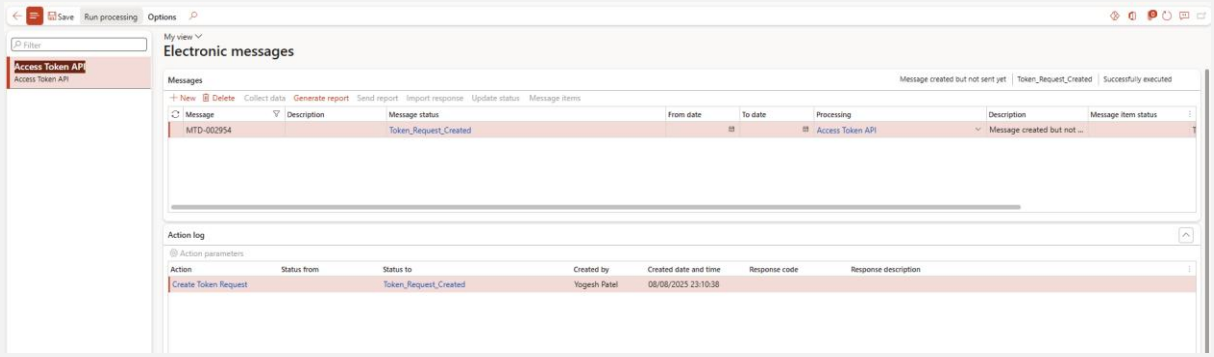
Run the pipeline inside D365FSCM to send a request to the Logic App, retrieve the JSON token, and confirm it's stored in your web application table. If the Logic App returns an error (e.g., wrong headers), the EM log will capture it for review.

Create New

Creates a new electronic message record to start a fresh process

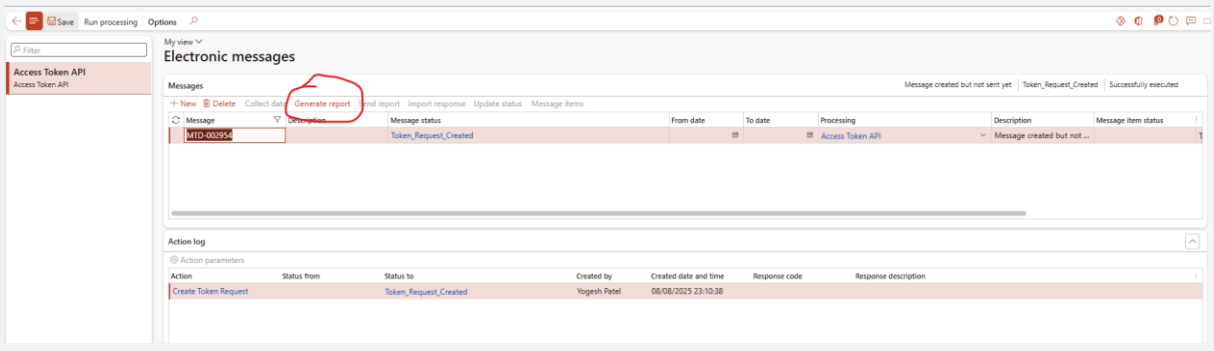


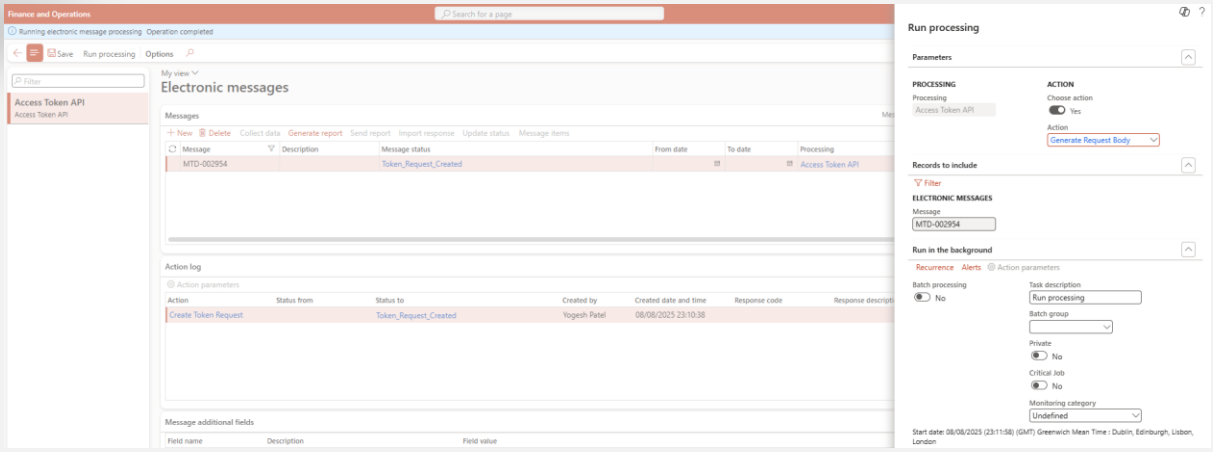


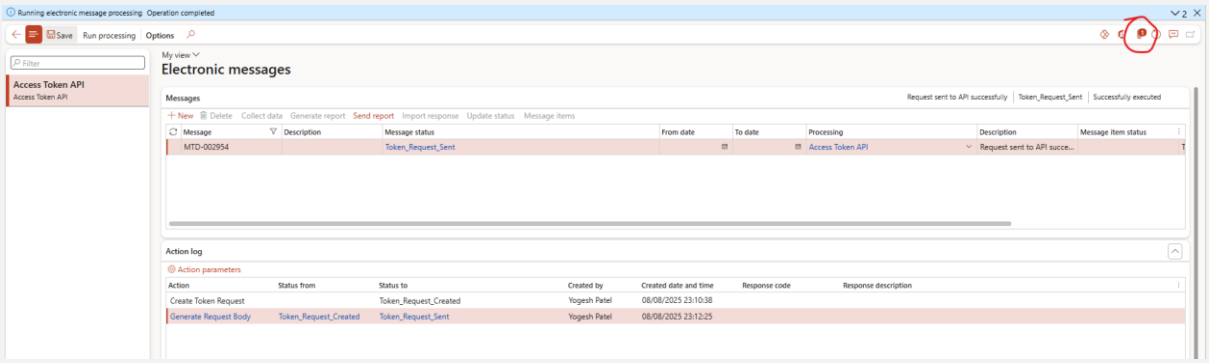


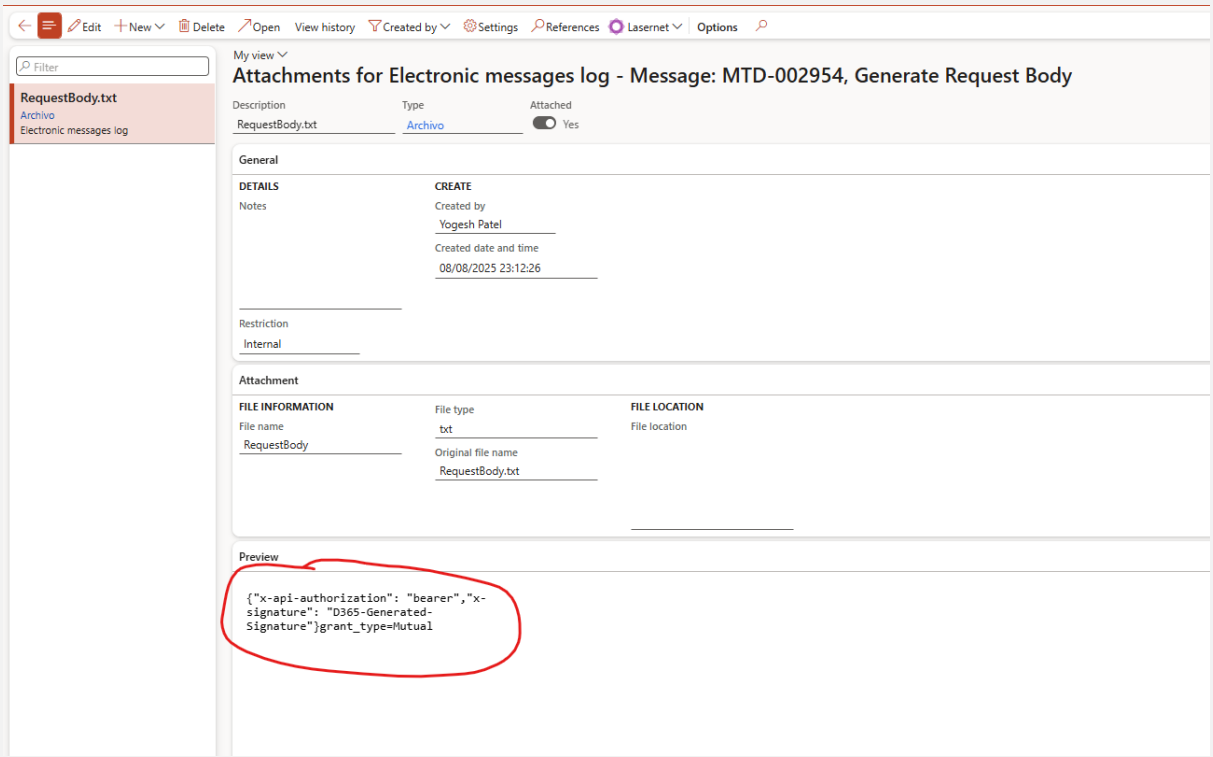
Generate Report

Runs the linked **ER export format** to build the request file or data payload (JSON) that will be sent out.



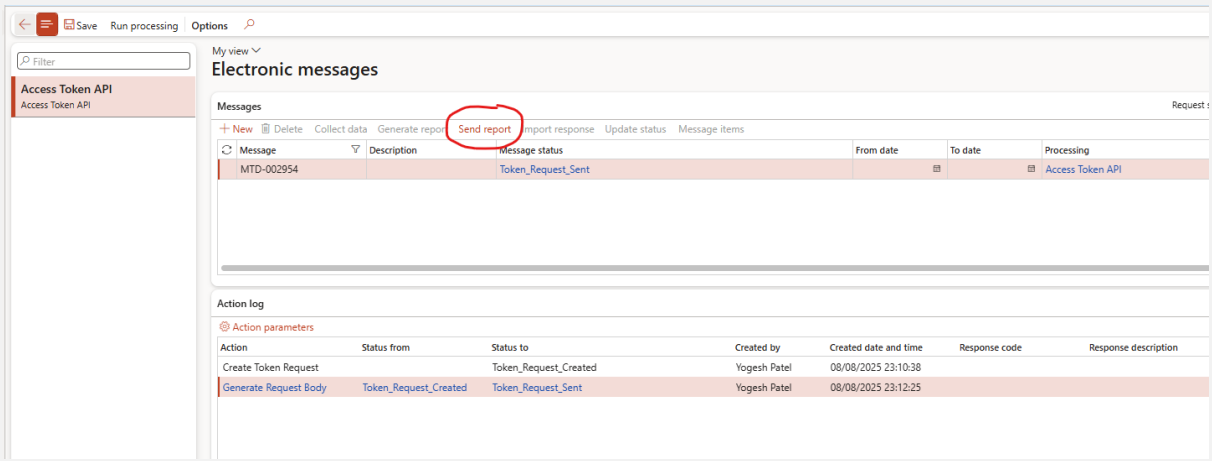

Finance and Operations
Electronic messages
Access Token API
Run processing
Parameters
PROCESSING
ACTION
Generate Request Body
Records to include
ELECTRONIC MESSAGES
MTD-002954
Run in the background
Action log
Create Token Request
Token_Request_Created
Yogesh Patel
08/08/2025 23:10:38
Message additional fields

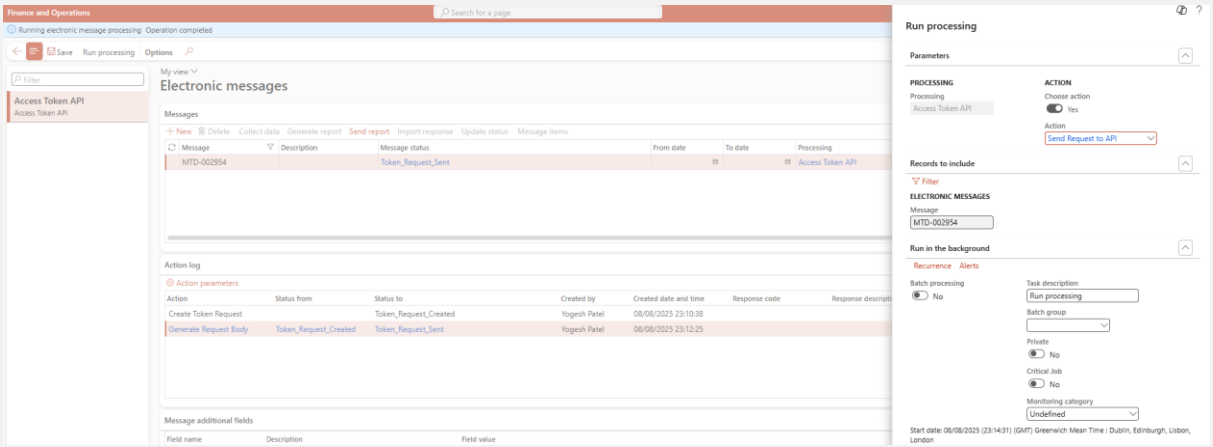

Running electronic message processing Operation completed
Save
Run processing
Options
Electronic messages
Access Token API
Messages
Request sent to API successfully
Token_Request_Sent
Successfully executed
MTD-002954
Token_Request_Sent
Action log
Create Token Request
Token_Request_Created
Generate Request Body
Token_Request_Sent
Yogesh Patel
08/08/2025 23:10:38
08/08/2025 23:12:25

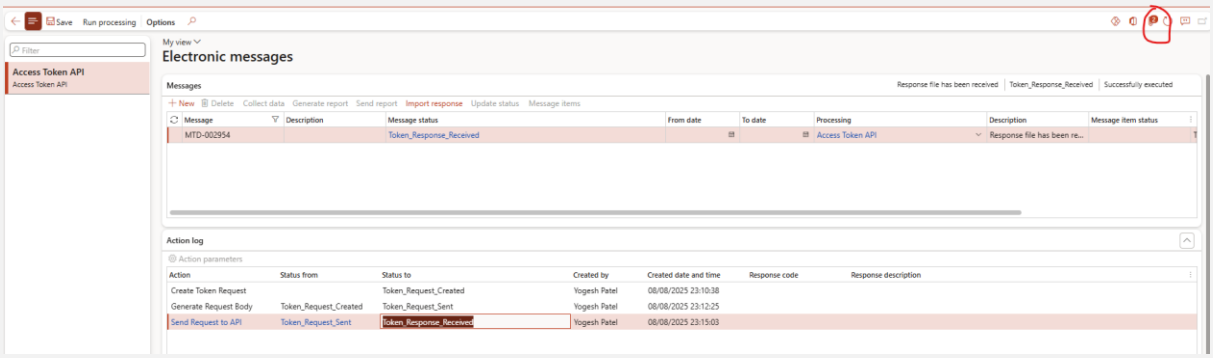

Edit
New
Delete
Open
View history
Created by
Settings
References
Lasernet
Options
My view
Attachments for Electronic messages log - Message: MTD-002954, Generate Request Body
RequestBody.txt
Archivo
Electronic messages log
Description
RequestBody.txt
Type
Archivo
Attached
Yes
General
DETAILS
Notes
CREATE
Created by
Yogesh Patel
Created date and time
08/08/2025 23:12:26
Restriction
Internal
Attachment
FILE INFORMATION
File name
RequestBody
File type
txt
Original file name
RequestBody.txt
FILE LOCATION
File location
Preview
{"x-api-authorization": "bearer","x-signature": "D365-Generated-Signature"}grant_type=Mutual

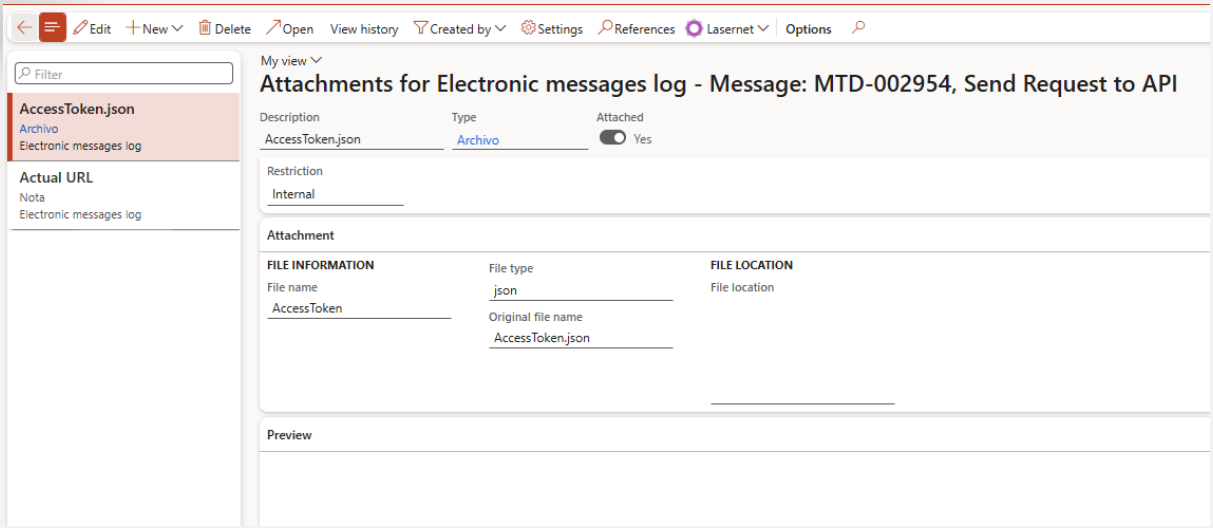
Send Report

Sends the generated request file to the connected web service or endpoint (Logic Apps in this case).



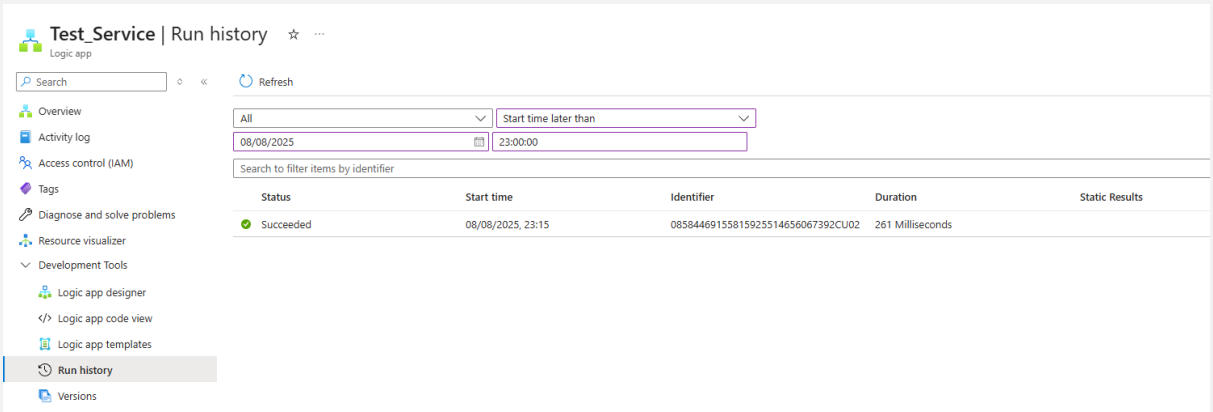


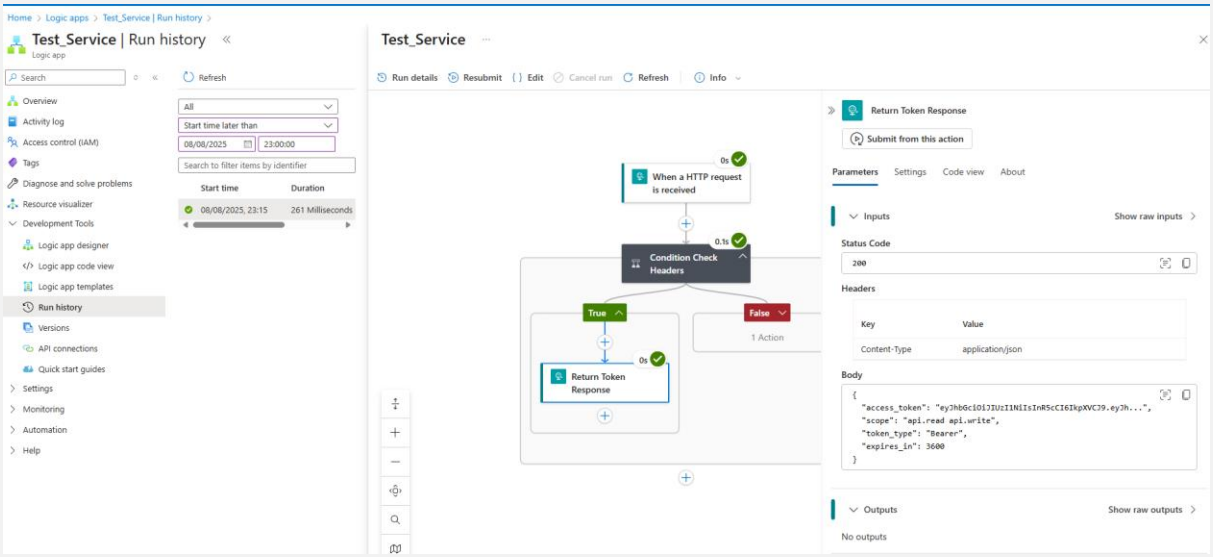




Verify Logic app trigger

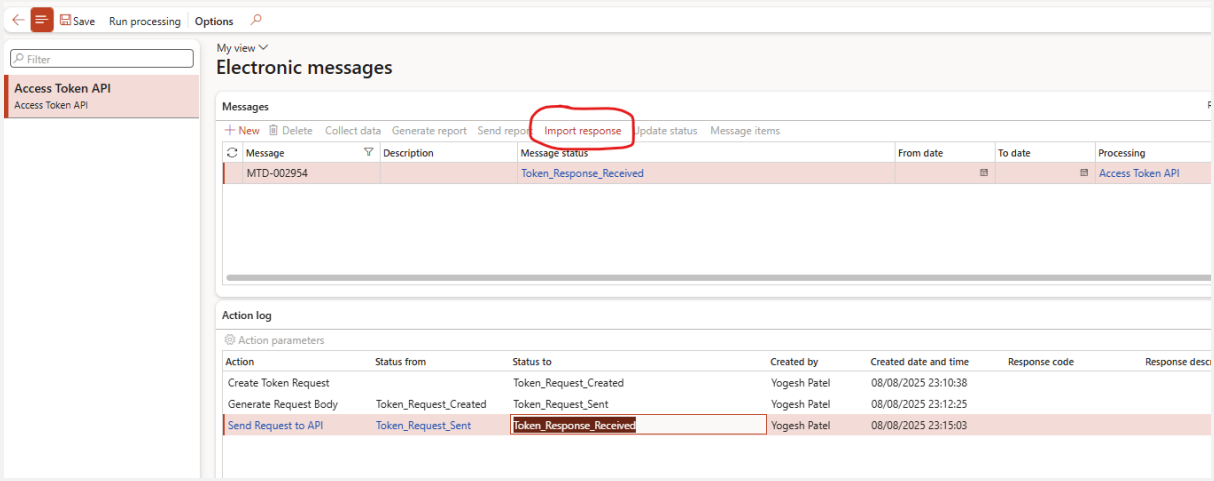
Check Logic app run. Above access token received from Logic app

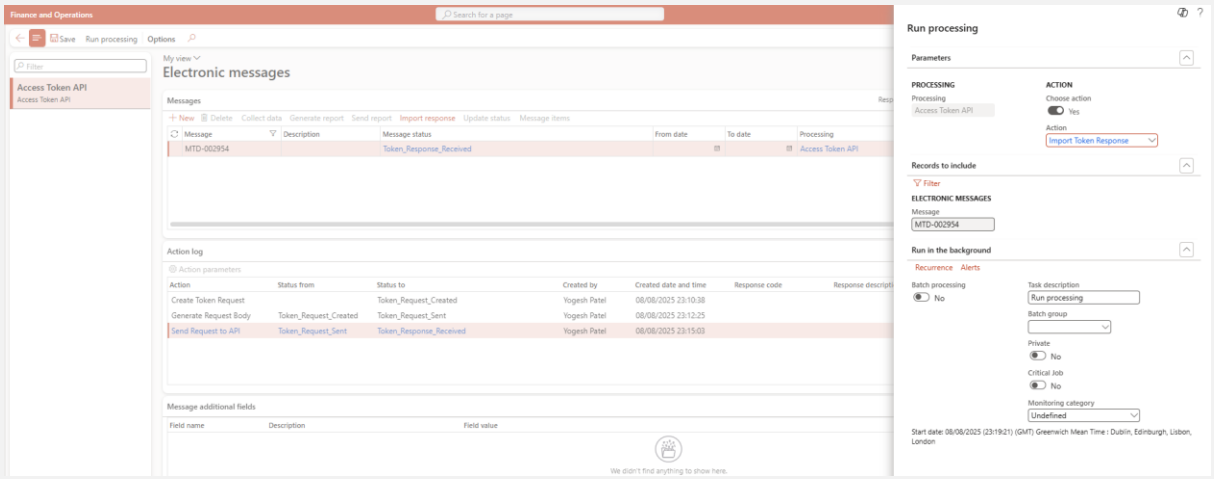


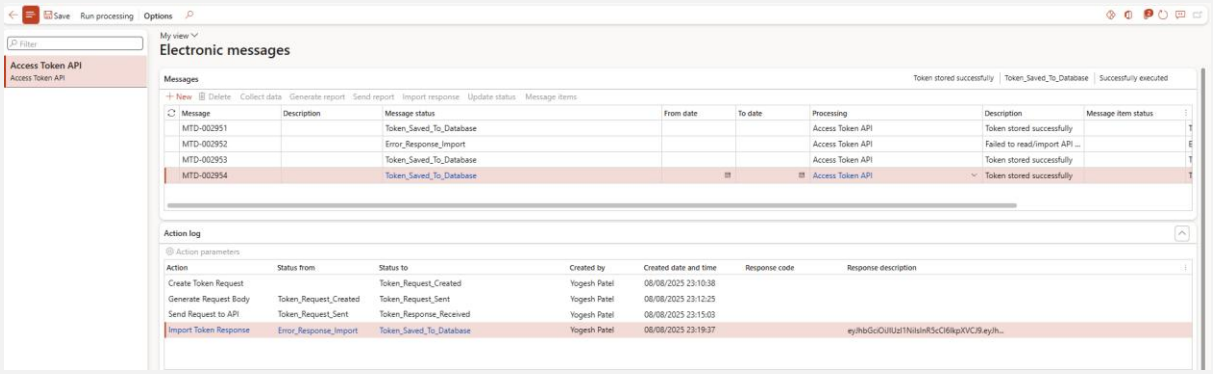


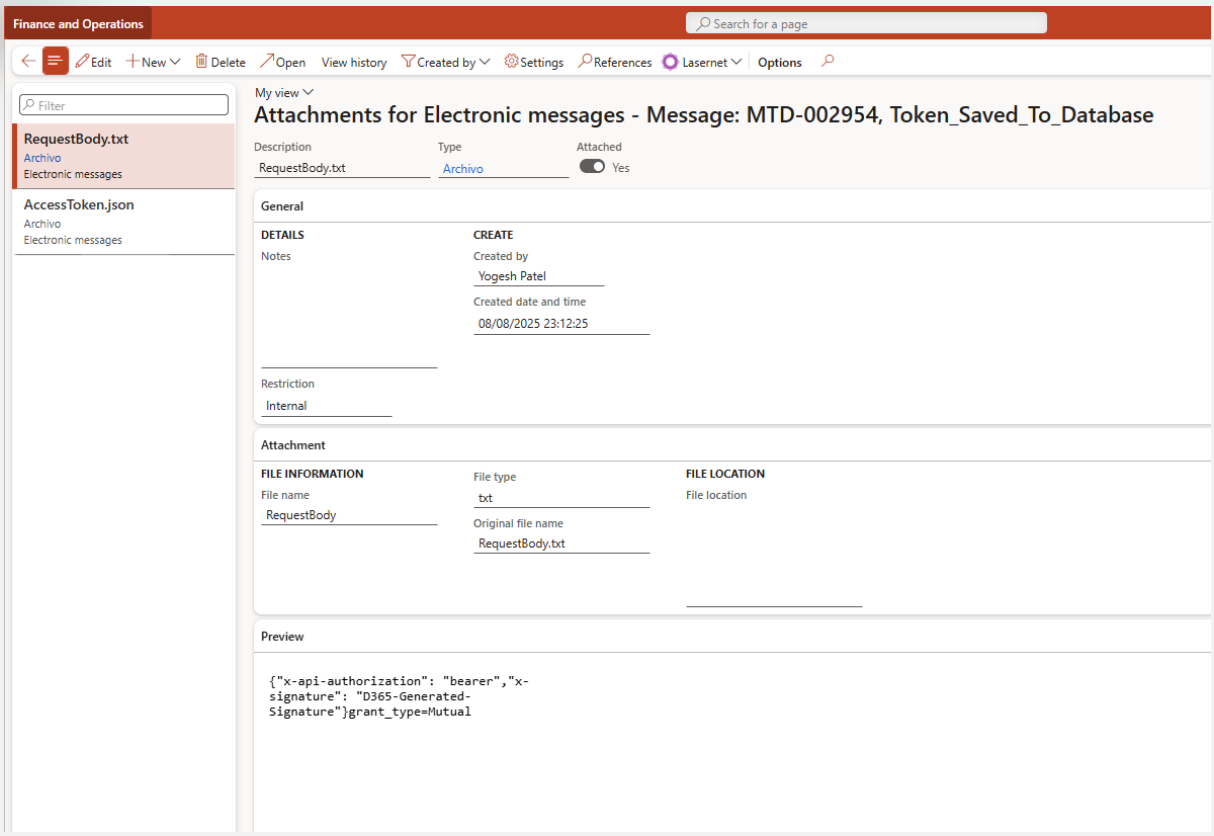
Import Response

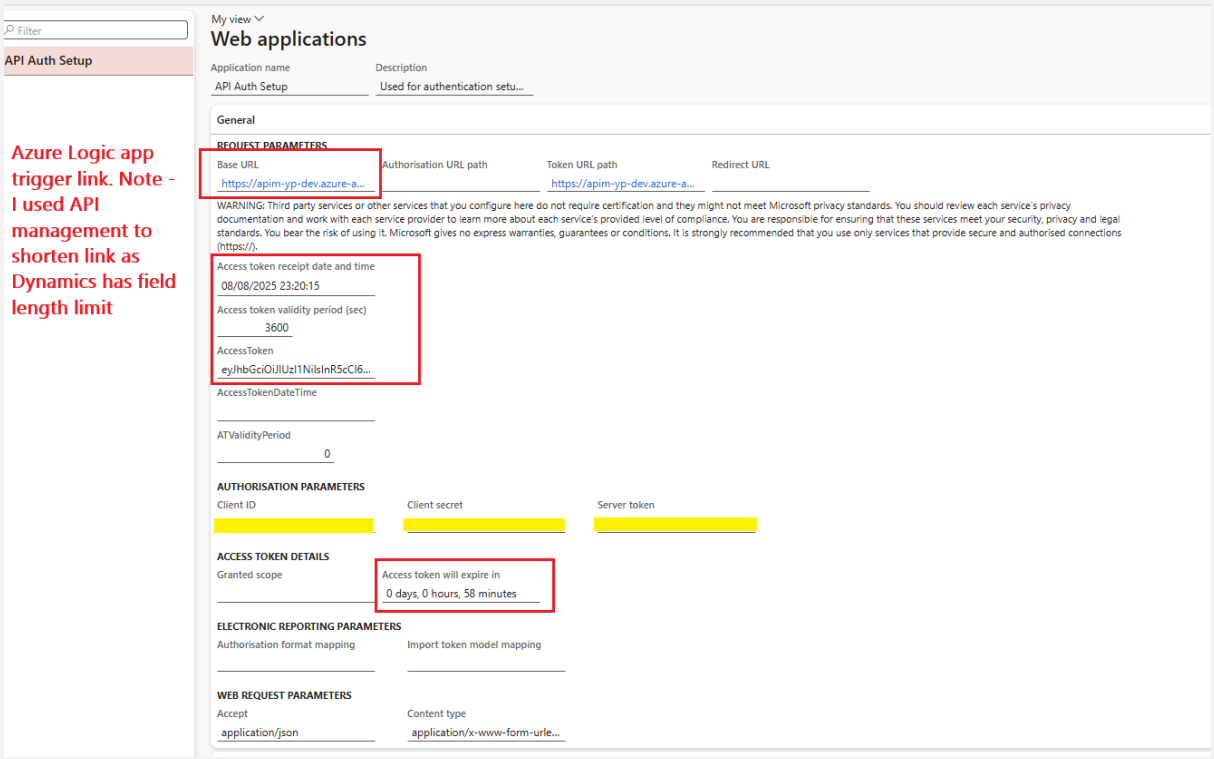
Runs the linked **ER import format** to read the incoming response file from the web service and store the results in D365 tables for tracking or further processing.








Finance and Operations
Search for a page
Edit
New
Delete
Open
View history
Created by
Settings
References
Lasernet
Options
Filter
RequestBody.txt
Archivo
Electronic messages
AccessToken.json
Archivo
Electronic messages
My view
Attachments for Electronic messages - Message: MTD-002954, Token_Saved_To_Database
Description
RequestBody.txt
Type
Archivo
Attached
Yes
General
DETAILS
Notes
CREATE
Created by
Yogesh Patel
Created date and time
08/08/2025 23:12:25
Restriction
Internal
Attachment
FILE INFORMATION
File name
RequestBody
File type
txt
Original file name
RequestBody.txt
FILE LOCATION
File location
Preview
{"x-api-authorization": "bearer","x-signature": "D365-Generated-Signature"}grant_type=Mutual


Filter
API Auth Setup
My view
Web applications
Application name
API Auth Setup
Description
Used for authentication setu...
General
REQUEST PARAMETERS
Base URL
https://apim-yp-dev.azure-a...
Authorisation URL path
Token URL path
https://apim-yp-dev.azure-a...
Redirect URL
WARNING: Third party services or other services that you configure here do not require certification and they might not meet Microsoft privacy standards. You should review each service's privacy documentation and work with each service provider to learn more about each service's provided level of compliance. You are responsible for ensuring that these services meet your security, privacy and legal standards. You bear the risk of using it. Microsoft gives no express warranties, guarantees or conditions. It is strongly recommended that you use only services that provide secure and authorised connections (https://).
Access token receipt date and time
08/08/2025 23:20:15
Access token validity period (sec)
3600
AccessToken
eyJhbGciOiJIUzI1NiIsInR5cCI6...
AccessTokenDateTime
ATValidityPeriod
0
AUTHORISATION PARAMETERS
Client ID
Client secret
Server token
ACCESS TOKEN DETAILS
Granted scope
Access token will expire in
0 days, 0 hours, 58 minutes
ELECTRONIC REPORTING PARAMETERS
Authorisation format mapping
Import token model mapping
WEB REQUEST PARAMETERS
Accept
application/json
Content type
application/x-www-form-urle...
Azure Logic app trigger link. Note - I used API management to shorten link as Dynamics has field length limit

How It All Works at Runtime

1. **User runs processing** from the EM workspace
2. **Request body and headers** are generated by ER
3. **Logic App is triggered**, returns response JSON
4. **Response is parsed**, token stored in D365
5. **Logs are created**, status is updated, attachments stored

Best Practices & Recommendations

Area	Best Practice
Security	Secure Logic App with OAuth or header-based key
Token expiry	Use expires_in to trigger future token refresh
Reusability	Centralize token storage in EMWebApplication table
Environments	Use parameterized Logic App URLs per environment
Audit trail	Always write logs to BusinessDocumentExecutionResultLog or attachments

Real-World Scenarios

- Fetch **access tokens** from customs or banking APIs
- Submit **invoice batches** to eInvoicing platforms with signed headers
- Call **tax rate services** or **validation engines** via Logic Apps
- Push data into **external audit or fraud detection APIs**

Coming Up Next

In GS519 – Managing Errors, Logs, and Submission History in Globalization Studio, we'll explore how to track, troubleshoot, and resolve issues that arise during document submission—using real examples like Spain's FACe and SII submissions.
You'll learn how to:

- Read pipeline logs and error messages
- Analyze submission results and trace failures

- Export logs for audit purposes
- Retry or reprocess failed transactions

📖 *Continue reading: GS519 – Logs and Troubleshooting →*