Streamline Operations with Salesforce Email Services:

AUTOMATE, OPTIMIZE, AND EXCEL

Robert Davis
Solution Architect
Hike 2

SOUTHEAST DREAMIN'


airSlate
Sweep
sync
pendo
testsigma
CITRINCOOPERMAN®
salesforce
Score Keeper
for Salesforce
DataGroomr
Data Quality Perfected by AI
SOUTHEAST DREAMIN'

Why Email Services?

We send approximately
90 trillion emails per year

We send an estimated
294 billion emails
every day

8 million sent every sec

110 emails daily by corp. user

Source:
Vingapp.com

Inbound Email Services - Out of the Box

Pros

- A native Salesforce feature that converts incoming emails into **Cases**.
- No Apex code needed.
- Great for support scenarios.
- Automatically creates Cases from email.
- Maintains email threads via Ref ID.
- Can attach files.
- Can assign Cases using rules.
- Fast setup

Limitations

- Only creates Cases, not Leads, Contacts, etc.
- Parsing email content into structured fields is limited without extra automation (e.g., Flow).
- Not great for non-service teams (like Sales or Custom Object needs).
- Limits on Case Settings coming in to new Case.

Real-World Story: The Arbitrator's Inbox

📪 **Real-World Example: One person, one inbox, holding up the whole process**

A legal firm receives critical email threads between lawyers and arbitrators…

Scenario: Legal firm receives arbitration-related emails

Problem: Sole person manually forwards messages to the legal team
Limitations of Email-to-Case: No support for custom workflows, other objects

Solution (Apex Email Service):

- ✅ Automatically attaches emails to the correct Case
- ✅ Creates follow-up Tasks
- ✅ Adds metadata for accountability and audit trail

Email Services - On Demand / Apex

Pros

- Full control over how emails are processed.
- Great for custom logic (e.g., parsing CSV attachments, conditional routing).
- You can implement retry logic, error handling, custom logging.
- Can run external calls to other systems.

Limitations

- Requires Apex and tests.
- Must manage governor limits and potential errors carefully.
- More complex deployment and testing process.
- CI/CD Friendly
- Scalability
- Complex Logic

Real-World Story: Flow Error Logging: Catching What the User Never Reports

📪 **Real-World Example: Sometimes the most important errors are the ones nobody sees**

Scenario: Emails sent when automations fail

Problem: Visibility may be lost in email. How to prioritize on issues.

Limitations of Email-to-Case: No support for custom workflows, other objects

Solution (Apex Email Service):

- ✅ Created a custom **Email Service** to receive Flow error messages
- ✅ Parsed body, subject line, and headers
- ✅Logged errors to **Flow_Error__c** object for triage and reporting
- ✅ Now searchable, reportable, and trackable over time

Key Components of an Apex Email Service

```apex
public with sharing class AutomationEmailService implements Messaging.InboundEmailHandler {
    public Messaging.InboundEmailResult handleInboundEmail(Messaging.InboundEmail email, Messaging.InboundEnvelope envelope) {

        Messaging.InboundEmailResult result = new Messaging.InboundEmailResult();

        try {
            // Log the incoming email details using Nebula Logger
            Logger.error('Inbound Email Received');
            Logger.setField(LogEntryEvent__e.Email_Subject__c, email.subject);
            Logger.setField(LogEntryEvent__e.Email_From__c, email.fromAddress);
            Logger.setField(LogEntryEvent__e.Email_Body__c, email.plainTextBody);
            Logger.setField(LogEntryEvent__e.Email_To__c, String.join(email.toAddresses, ','));
            Logger.saveLog();


            result.success = true;
        } catch (Exception e) {
            Logger.error('Error handling inbound email: ' + e.getMessage()+' stack trace: '+e.getStackTraceString());
            Logger.saveLog();

            result.success = false;
            result.message = 'Error: ' + e.getMessage();
        }

        return result;
    }
}
```

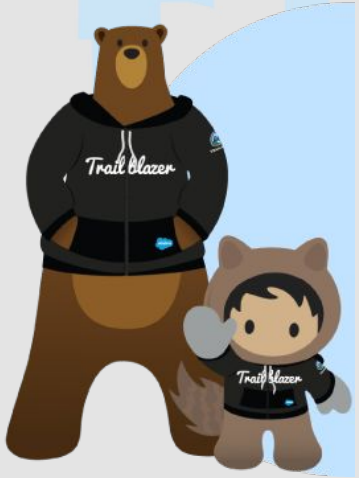
- handleInboundEmail — the **required method name**.
- Messaging.InboundEmail — contains the actual email content (subject, body, attachments, etc.).
- Messaging.InboundEnvelope — contains metadata like fromAddress, toAddress, etc.
- Messaging.InboundEmailResult — must be returned, and typically you just set result.success = true.

Messaging.InboundEmail Fields Available

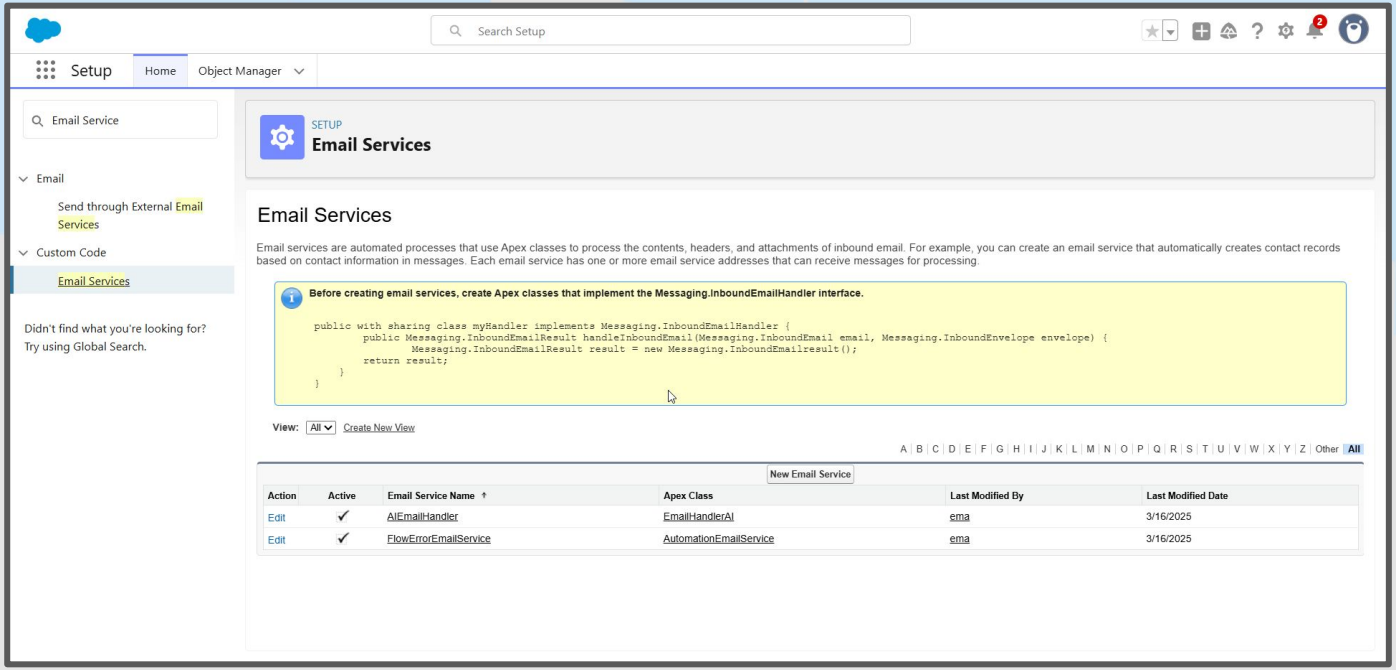
- **binaryAttachments**: An array of binary attachments included in the email. Each attachment is represented by a Messaging.InboundEmail.BinaryAttachment object, containing details like body, fileName, and mimeType.
- **ccAddresses**: An array of strings representing the CC (carbon copy) email addresses specified in the incoming email.
- **fromAddress**: A string containing the email address of the sender.
- **fromName**: A string containing the display name of the sender, if available.
- **headers**: A string capturing the raw email headers of the incoming email.
- **htmlBody**: A string containing the HTML version of the email body, if the email was sent in HTML format.
- **inReplyTo**: A string representing the 'In-Reply-To' header of the email, which can be used to identify the message to which this email is replying.
- **messageId**: A string containing the unique identifier of the email message.
- **plainTextBody**: A string containing the plain text version of the email body.
- **subject**: A string representing the subject line of the incoming email.
- **textAttachments**: An array of text attachments included in the email. Each attachment is represented by a Messaging.InboundEmail.TextAttachment object, containing details like body and fileName.
- **toAddresses**: An array of strings representing the primary recipient email addresses specified in the incoming email.

Messaging.InboundEnvelope Fields Available

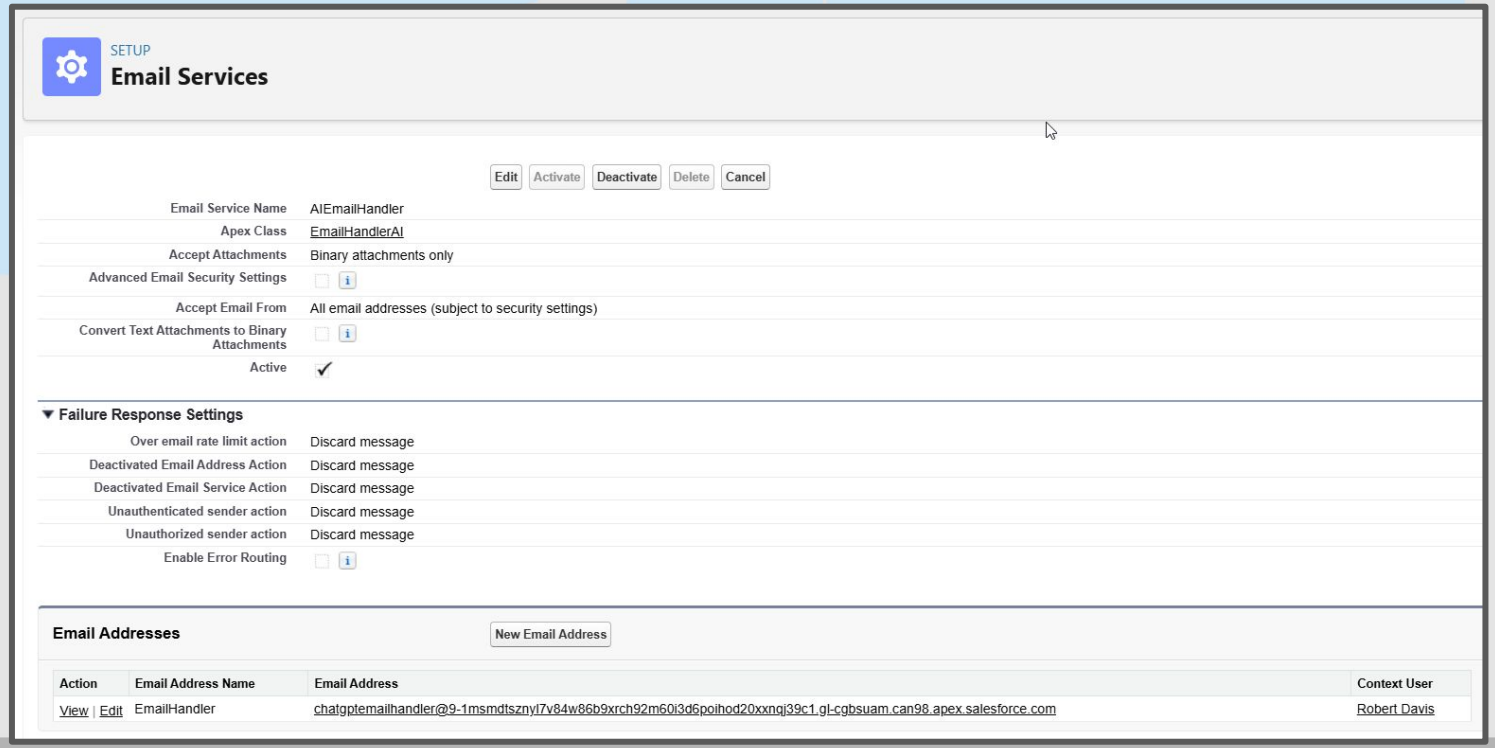
- `fromAddress`: A string containing the email address of the sender, as specified in the envelope.
- `toAddress`: A string containing the email address of the recipient, as specified in the envelope.



Email Service Setup



Email Service Setup

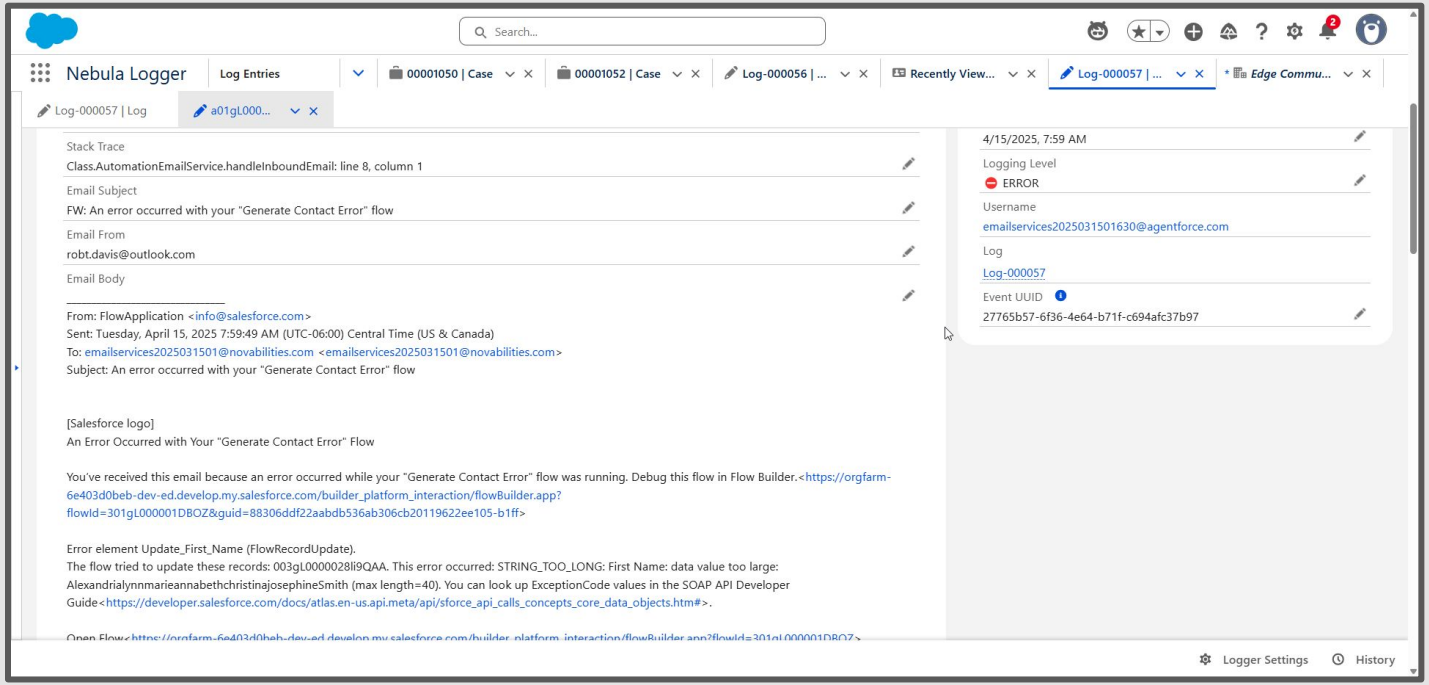


Review the AutomationEmailService

```
public with sharing class AutomationEmailService implements Messaging.InboundEmailHandler {
    public Messaging.InboundEmailResult handleInboundEmail(Messaging.InboundEmail email, Messaging.InboundEnvelope envelope) {

        Messaging.InboundEmailResult result = new Messaging.InboundEmailResult();

        try {
            // Log the incoming email details using Nebula Logger
            Logger.error('Inbound Email Received');
            Logger.setField(LogEntryEvent__e.Email_Subject__c, email.subject);
            Logger.setField(LogEntryEvent__e.Email_From__c, email.fromAddress);
            Logger.setField(LogEntryEvent__e.Email_Body__c, email.plainTextBody);
            Logger.setField(LogEntryEvent__e.Email_To__c, String.join(email.toAddresses, ','));
            Logger.saveLog();


            result.success = true;
        } catch (Exception e) {
            Logger.error('Error handling inbound email: ' + e.getMessage()+' stack trace: '+e.getStackTraceString());
            Logger.saveLog();

            result.success = false;
            result.message = 'Error: ' + e.getMessage();
        }

        return result;
    }
}
```

- Review in VS Code
- Might use truncation in case too long.
- Create an Event and forward Automated Process User Email Address

Capture Errors Over Time



- Flow errors often go unnoticed unless users report them.
- Able to Log and reduce inbox and see errors over time.
- Searchable and reportable.
- Easier triage and faster response.

SOUTHEAST DREAMIN'

Real-World Story: Snap, Send, Save: Simplifying Expense Tracking

📪 **Real-World Example: No app. No clicks. Just an email with a receipt.**

No Receipt left behind

Scenario: Remove Friction of uploading receipts, process Expenses quicker.

Problem: Uploading expense reports may require using desktop app and not terribly user friendly.

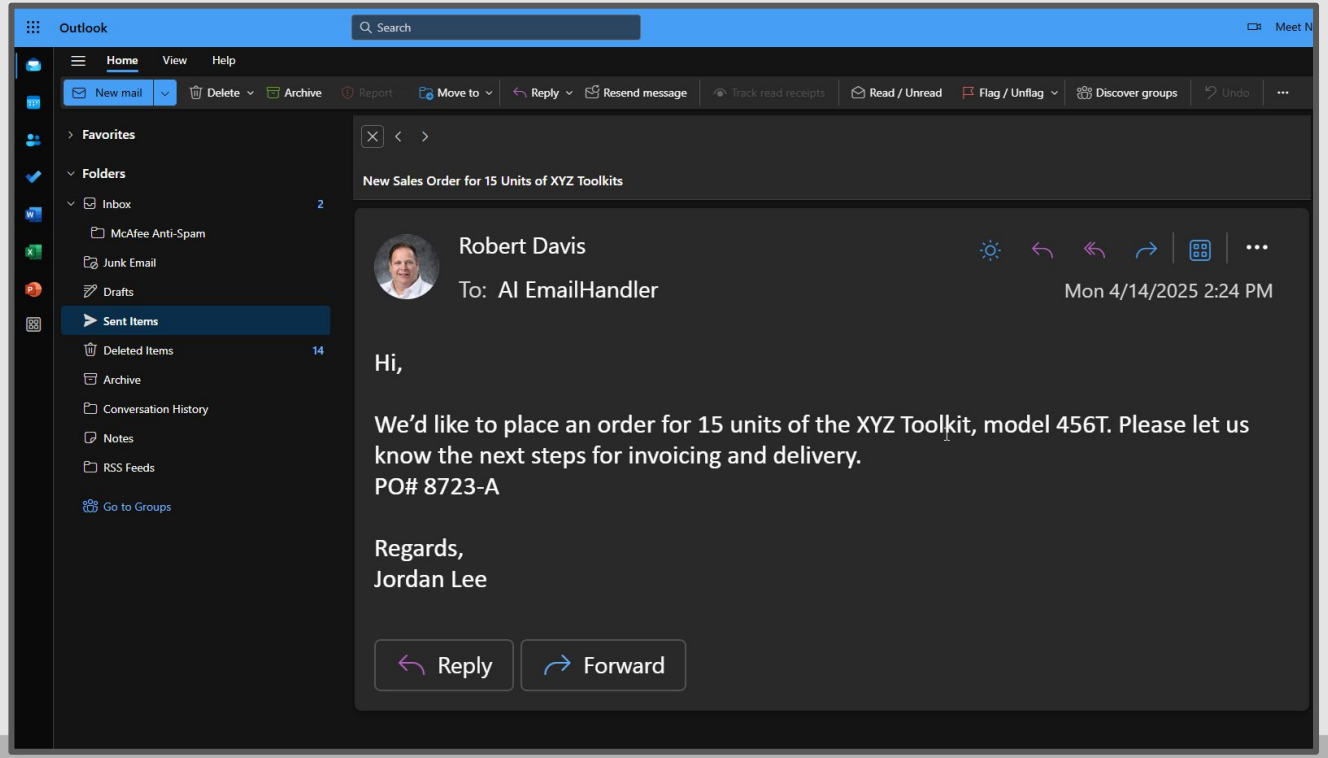
Limitations of Email-to-Case: No support for custom workflows, other objects
Solution (Apex Email Service):

- Sales reps emailed photos of receipts from the road
- Email sent to `expenses@company.com`
- Apex Email Service logic:
  - ✅ Match to existing open `Expense__c` by sender
  - ✅ If none, create new `Expense__c` record
  - ✅ Attach image file to record

- Later enhancement: OCR to auto-fill vendor, amount, and date

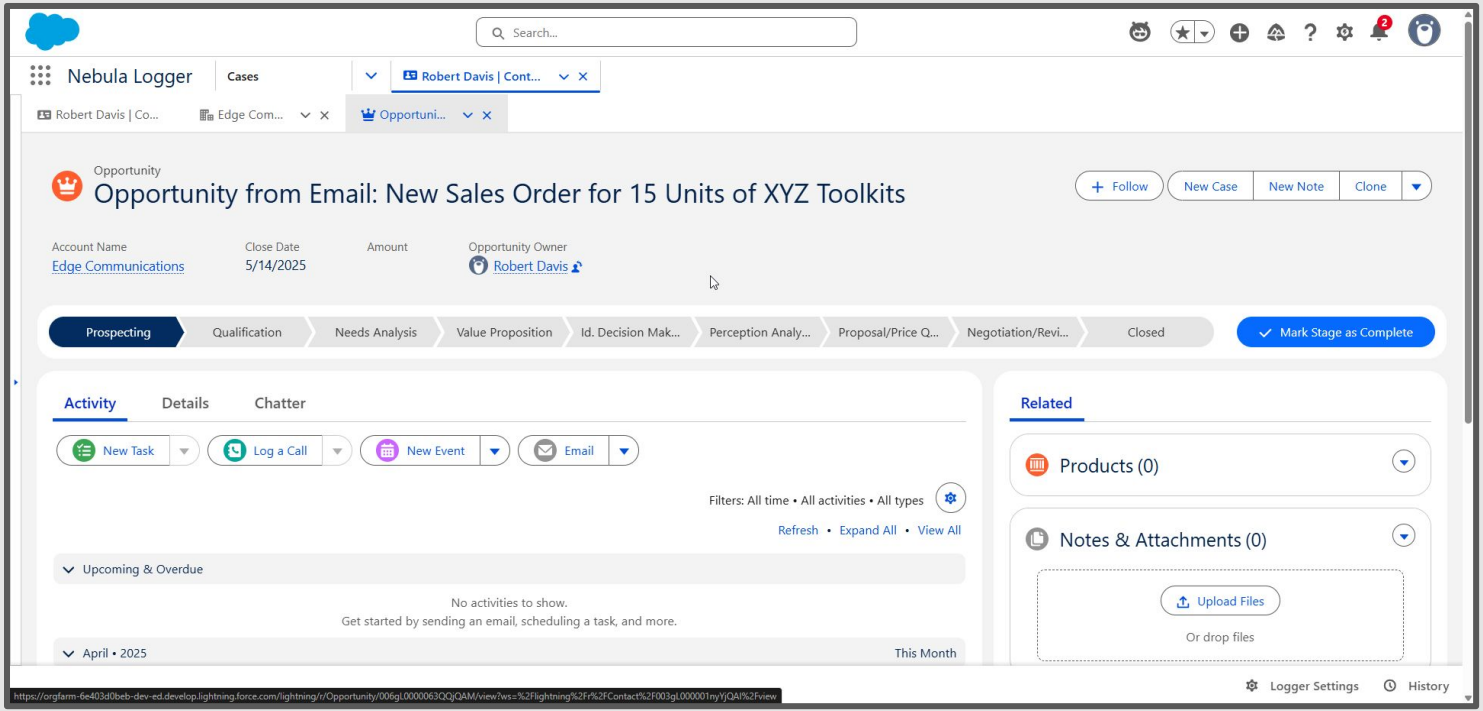

DEMO TIME
makeameme.org

Send / Forward Email



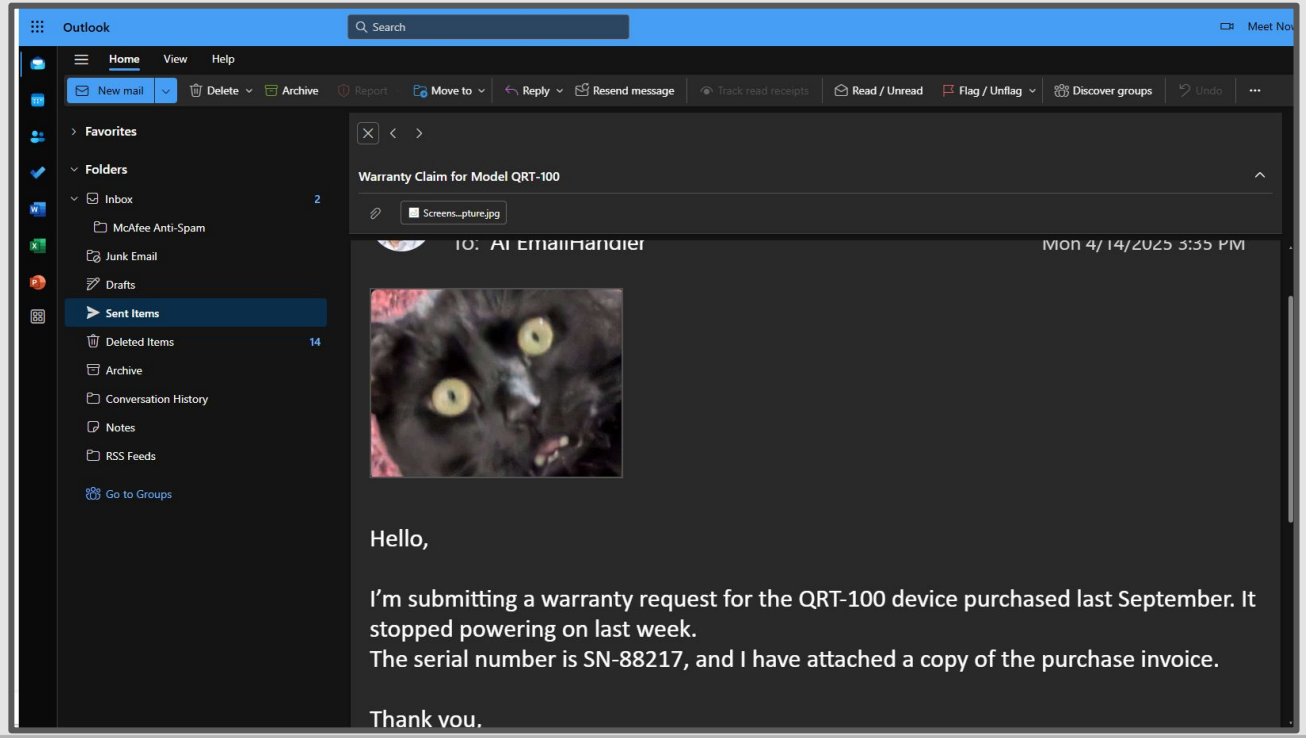
- Create an Opportunity not an option for out of the box
- See in industries where there is a unique identifier

Created Opportunity

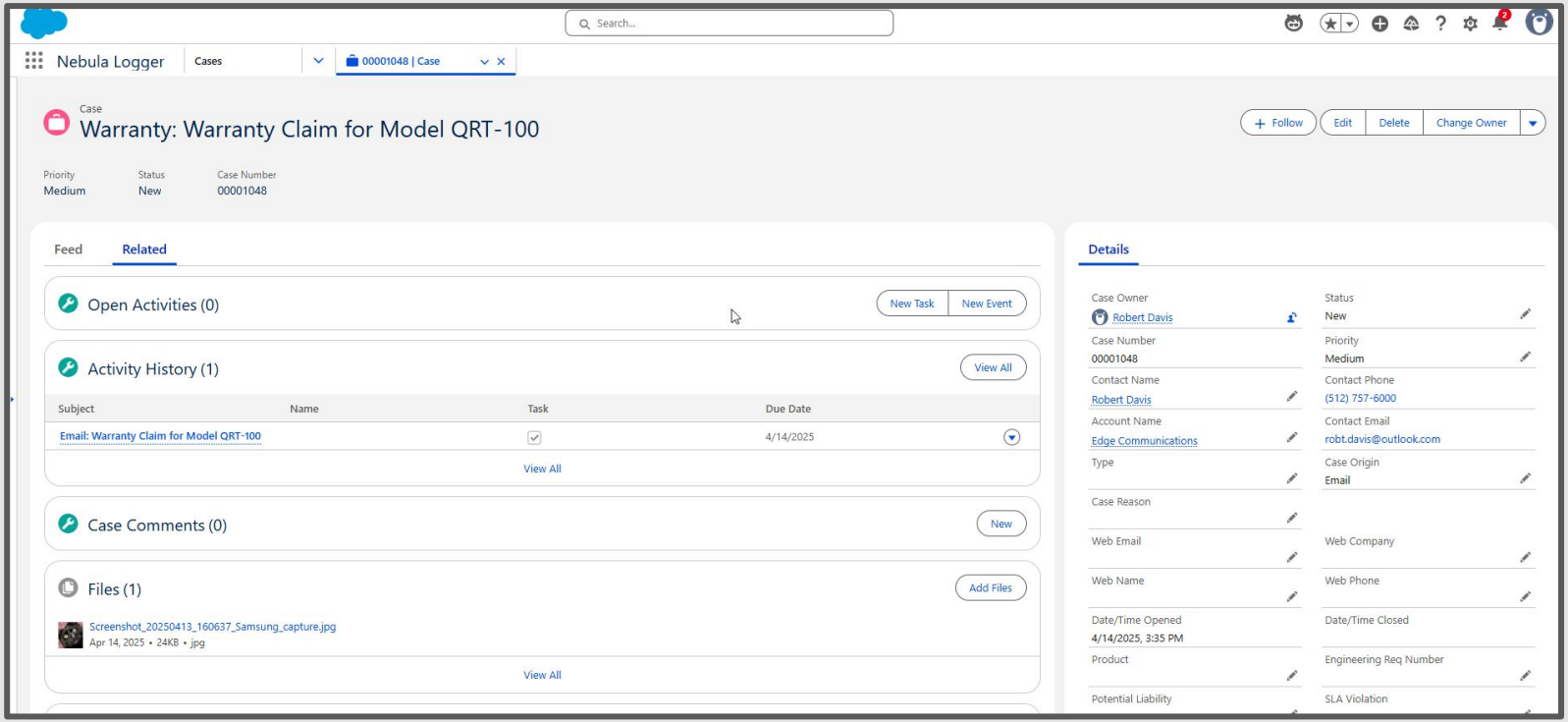


SOUTHEAST DREAMIN'

Send / Forward Email



Case Created



SOUTHEAST DREAMIN'

EmailHandlerAI

```
public class EmailHandlerAI implements Messaging.InboundEmailHandler {

    public Messaging.InboundEmailResult handleInboundEmail(Messaging.InboundEmail email, Messaging.
InboundEnvelope envelope) {
        Messaging.InboundEmailResult result = new Messaging.InboundEmailResult();

        // Callout to ChatGPT to classify email
        String emailType = EmailClassifier.classifyEmail(email.subject, email.plainTextBody);

        if(emailType == 'Product Inquiry'){
            SalesforceRecordCreator.createCase(email, 'Product Inquiry:');
        }
        else if(emailType == 'Sales Order'){
            SalesforceRecordCreator.createOpportunity(email);
        }
        else if(emailType == 'Warranty'){
            SalesforceRecordCreator.createCase(email, 'Warranty:');
        }
        else if(emailType == 'Request for More Information'){
            SalesforceRecordCreator.createCase(email, 'Request for More Information:');
        } else{
            SalesforceRecordCreator.createCase(email, 'Unable to Determine Email Type:');
        }

        result.success = true;
        return result;
    }
}
```

- Implements Messaging.InboundEmailHandler
- Passes off to a callout to do the classification
- Comes back and creates a Case or Opportunity

EmailClassifier

```
public class EmailClassifier {
    public static AI_Integration_Config__mdt
 config;

    public static String classifyEmail(String
 subject, String body) {

// Initialize configuration - get the values for c
onfig
        initializeConfig();

        // Prepare and send request
        HttpResponse response =
sendClassificationRequest(subject, body);

        // Log the response
        logResponse(response);

        // Process the response
        return processResponse(response);
    }
```

- Get api info from Custom Metadata
- Send the Classification to Gemini LLM
- Log the response
- Pass back the classification

SalesforceRecordCreator

```
public class SalesforceRecordCreator {

    public static void createCase(Messaging.InboundEmail email, String type){
        Contact cont = [
            SELECT Id, AccountId
            FROM Contact
            WHERE Email = :email.fromAddress
            LIMIT 1
        ];

        Case c = new Case();
        c.Subject = (type == 'Unable to Determine Email Type' ?
'Unable to Determine Email Type: ' : '') + Type+' '+email.subject;
        c.Description = email.plainTextBody;
        c.ContactId = cont.Id;
        c.AccountId = cont.AccountId;
        c.Origin = 'Email';
        insert c;

        attachEmailAndFiles(c.Id, email);
    }
```

- Create Case
- Create Task
- Create Email Message
- Create Account and Opportunity
- Create Opportunity
- Attach Attachments

THANK YOU / Q & A

Robert Davis
Solution Architect, Hike 2

🙏 “Thanks for spending time with me today — and for letting me talk about something I really enjoy building.

Slides

https://bit.ly/4kBerrm

LinkedIn

https://bit.ly/3XcyG5d

SOUTHEAST DREAMIN'

When to Use Apex Email Services

When out-of-the-box just isn’t enough.

- You need to create something *other than* a Case
- You want to parse email content or attachments
- You need to route messages based on subject or sender
- You want to log Flow or system activity by email
- You want to integrate with AI, OCR, or external systems
- You want to reduce inbox chaos with automation

Resources

- Salesforce CI/CD DevOps - A REAL How To
- Increase Email-to-Case Limit
- Setup Email-to-Case in Salesforce
- Using the InboundEmail Object
- Email Services in Salesforce with simple example
- InboundEmail Object fields

SOUTHEAST DREAMIN'


airSlate
Sweep
sync
pendo
testsigma
CITRINCOOPERMAN®
salesforce
Score Keeper
for Salesforce
DataGroomr
Data Quality Perfected by AI
SOUTHEAST DREAMIN'