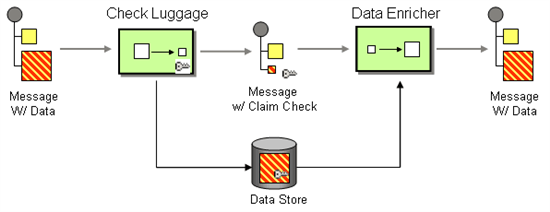
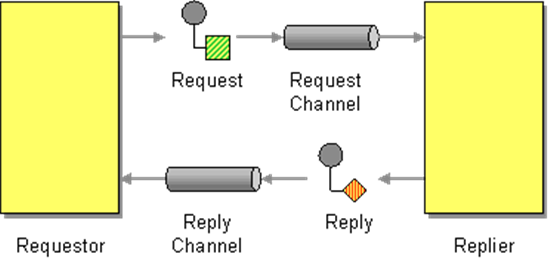
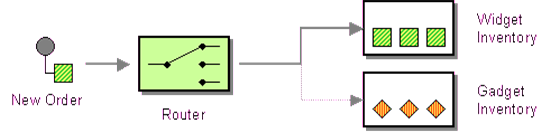
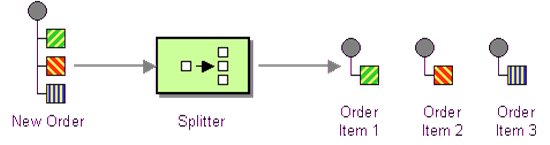
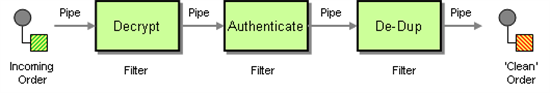
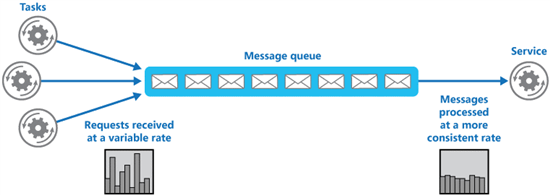
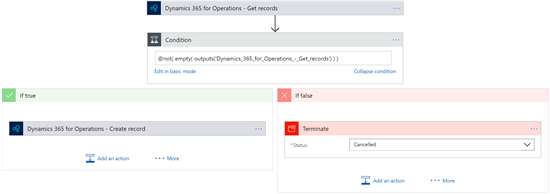
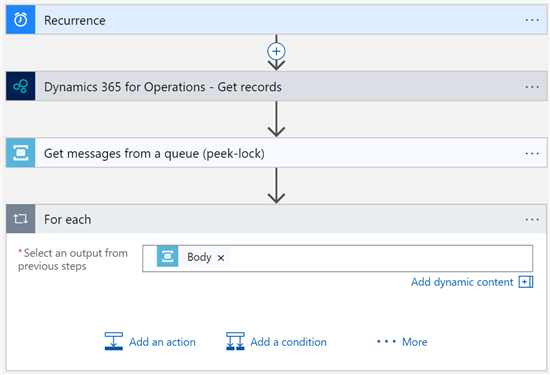
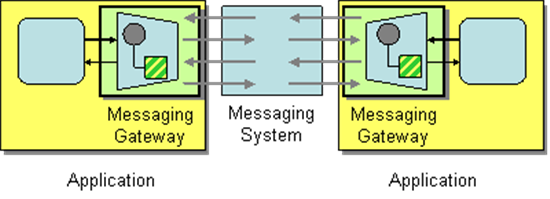
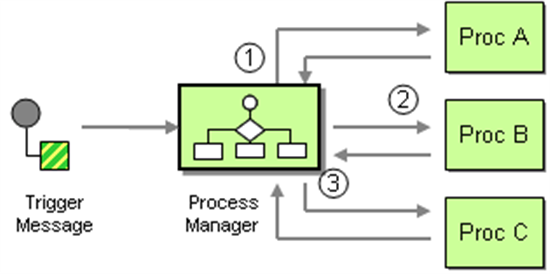
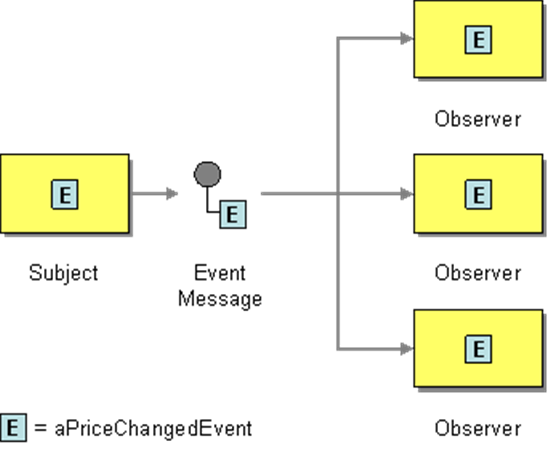
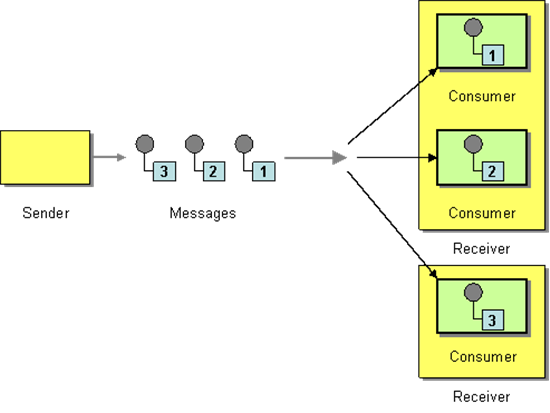
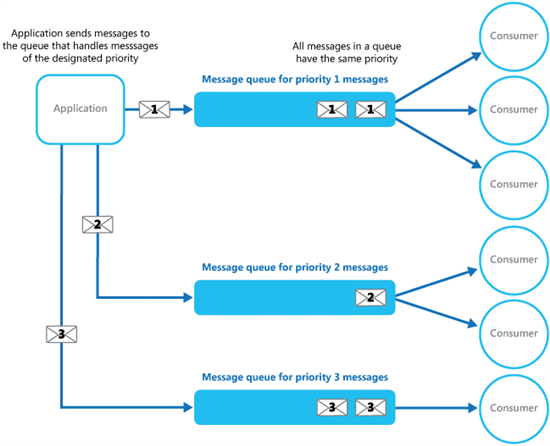
1. Introduction
   1. Overview of the workshop
   2. Objectives of the workshop
   3. Prerequisites for attendees
   4. Activities
      1. Activity 1: Have the attendees introduce themselves and share their experience level with Logic Apps, Dynamics Finance and Operations, and integration.
   5. What are Logic Apps?
   6. What can they Do?
   7. Power Automate vs Logic Apps Vs Function Apps
2. Azure infrastructure for integrations
   1. What is an App registration?
   2. Azure Queues
      1. https://learn.microsoft.com/en-us/azure/storage/queues/storage-queues-introduction
   3. Topics
      1. https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-queues-topics-subscriptions#topics-and-subscriptions
   4. Event Grids
      1. https://learn.microsoft.com/en-us/azure/event-grid/overview
   5. Event Hubs
      1. https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-about
   6. Blob Storage
      1. https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction
   7. HTTPS
      1. Logic Apps HTTP trigger
      2. Function App
      3. Code sample
         1. Logic app that gets a HTTP Trigger
         2. Posts it to Teams
   8. Power Automate
   9. Activities
      1. Activity 2: discuss the different Azure infrastructure options available for Logic App integrations (App Register, Azure Queues, Topics, Event Grids, Event Hubs, Blob Storage, HTTPS, Power Automate)
      2. Activity 3: Create Azure queue, create logic app to process message in queue
      3. ~~Puzzle questions – provide scenarios and select the azure infrastructure and F&O integration tooling~~
   10. Logic Apps
       1. Dataverse
          1. <https://learn.microsoft.com/en-us/azure/connectors/connect-common-data-service?tabs=consumption>
       2. Business Events
          1. Text

             Description automatically generated
          2. https://dynvision365.com/2022/06/08/using-azure-logicapps-and-d365finops-business-events-for-data-export-scenarios/
3. Integration tooling in F&O
   1. Data Events
      1. <https://learn.microsoft.com/en-us/dynamics365/fin-ops-core/dev-itpro/business-events/data-events>
      2. Examples
         1. Vendor Create
         2. Customer Update
         3. Released Product Delete
      3. Code Sample
         1. Logic App to email data event payload
   2. Business Events
      1. <https://learn.microsoft.com/en-us/dynamics365/fin-ops-core/dev-itpro/business-events/home-page>
      2. Examples
         1. Collect Letter Created
         2. Interest Note Posted
         3. Purchase Order Confirmed
      3. What’s the difference between Data Events and Business events?
         1. BE is event is confirmed complete. Ie; Sales Order Invoice Posted
         2. DE is a record was written to a table (messaged via data entity). Ie; record in SalesInvoiceJour was written.
         3. BE generally more performant \*
         4. DE contain entire record so can run into performance issues
      4. Code Sample
         1. Logic App to email data event payload
   3. Virtual Entities
      1. <https://learn.microsoft.com/en-us/dynamics365/fin-ops-core/dev-itpro/power-platform/virtual-entities-overview>
      2. <https://learn.microsoft.com/en-us/dynamics365/fin-ops-core/dev-itpro/power-platform/enable-virtual-entities>
   4. Dual Write
      1. <https://learn.microsoft.com/en-us/dynamics365/fin-ops-core/dev-itpro/data-entities/dual-write/dual-write-overview>
   5. Recurring Integrations
      1. https://learn.microsoft.com/en-us/dynamics365/fin-ops-core/dev-itpro/data-entities/recurring-integrations
   6. DMF REST API
      1. https://learn.microsoft.com/en-us/dynamics365/fin-ops-core/dev-itpro/data-entities/data-management-api
   7. Custom Services
      1. SOAP
         1. https://learn.microsoft.com/en-us/dynamics365/fin-ops-core/dev-itpro/data-entities/custom-services
      2. JSON
         1. https://learn.microsoft.com/en-us/dynamics365/fin-ops-core/dev-itpro/data-entities/custom-services
   8. Activities
      1. Activity 4: (group) discuss the different integration tooling options available in Dynamics Finance and Operations
      2. Activity 5: Create data event endpoint with azure service bus queue, key vault, infrastructure
      3. ~~Walkthrough: configure dual write~~
4. Common Logic App Patterns
   1. Claim-check
      1. “Luggage handling”
      2. 
      3. A common pattern will be to store large messages in a separate (BLOB) storage during message processing.
      4. Code Sample
         1. Get BE payload, use OData to get entire record
   2. Request and reply
      1. 
      2. the logic app receiving messages will also update the sender after the message was successfully transferred.
      3. Code sample
         1. Send posting to F&O, wait for affirmative response
   3. Content Based Routing
      1. 
      2. The Content-Based Router examines the message content and routes the message onto a different channel based on data contained in the message
      3. Code Sample
         1. Create logic app to route a customer create message
         2. Create logic app to take routed message and process it
   4. Splitter
      1. 
      2. A Splitter is useful to break out a single message into a sequence of sub-messages that can be processed individually.
      3. Code Sample
         1. Submit a message with multiple messages in it, create logic app to split messages, create another to process messages
   5. Pipes and Filters
      1. 
      2. Each logic app will perform a simple operation.
      3. The location where a message gets stuck, indicates what action had failed.
      4. Each step in the process can be individually enabled / scaled / improved.
      5. Code Sample
         1. Create logic app to convert from format 1 to format 2
         2. Create another logic to de-dupe
         3. Create another logic app to process message
   6. Load Leveling
      1. 
      2. Code Sample
         1. Create logic app will only process the top 50 messages every minute
   7. Idempotent Receiver
      1. 
      2. check whether a sales order is already existing, when not existing create the sales order.
      3. Code sample
         1. Create a logic app that will check if an order exists in data lake
   8. Circuit Breaker
      1. 
      2. running a small health check up front
      3. code sample
         1. create logic app to check health of service before processing message
   9. ~~Messaging Gateway / Enterprise Service Bus (ESB)~~
      1. 
      2. ~~instead of creating a direct interface between each pair of applications, create an interface from each application to a central gateway with the role of transporting the message to other systems.~~
   10. Process Manager
       1. 
       2. A Process manager Logic App might be implemented to guide a single message through all required steps. This will centralize coordination but will also create a single point of complexity (read: failure).
       3. Code Sample
          1. Get an order message with full details
          2. Create customer
          3. Create order
          4. Post confirmation for order
   11. Event Messager (Event Hub)
       1. 
   12. ~~Competing Consumers~~
       1. 
       2. ~~Main advantage of this model, will be the fact that all instances will be processing on full capacity, until all work has been completed. This model will scale to a very high throughput.~~
   13. Priority Queue
       1. 
       2. The Priority queue is a pattern where Incoming messages are sorted into multiple queues based on their priority
          1. Create 3 queues and a logic app for each queue as pairs (queue 1 pairs with logic app 1, q2 with LA2, and so on)
          2. LA2 can’t run until q1 is empty.
          3. LA3 can’t run until q2 is empty
   14. Activities
       1. Activity 5: Discuss examples of when to use each example
          1. Examples:
             1. Create examples
       2. Activity 6: Create workflow for decision making process
5. Considerations
   1. Best practices for designing and building integrations with Logic Apps
      1. List 1
         1. Use a consistent naming convention for connectors, triggers, and actions to improve readability and maintainability.
            1. <https://blog.sandro-pereira.com/2022/02/25/logic-app-best-practices-tips-and-tricks-1-logic-app-naming-convention/>
            2. <https://blog.sandro-pereira.com/2022/02/27/logic-app-best-practices-tips-and-tricks-2-actions-naming-convention/>
            3. https://github.com/Azure-Samples/azure-logic-apps-deployment-samples/blob/master/naming-convention.md
         2. Leverage the built-in error handling capabilities of Logic Apps to handle errors and exceptions.
            1. <https://learn.microsoft.com/en-us/azure/logic-apps/monitor-logic-apps-log-analytics>
            2. <https://learn.microsoft.com/en-us/azure/logic-apps/monitor-logic-apps-log-analytics>
            3. <https://www.serverlessnotes.com/docs/logic-apps-exception-handling-using-configure-run-after-settings>
         3. Use the Test feature in Logic App Designer to test your integration before deploying it to production.
            1. <https://learn.microsoft.com/en-us/azure/logic-apps/test-logic-apps-mock-data-static-results?tabs=consumption>
         4. Utilize variables and expressions to make your integration more dynamic and reusable.
            1. https://learn.microsoft.com/en-us/azure/logic-apps/logic-apps-create-variables-store-values
         5. Use the Batch operations to process large sets of data more efficiently.
            1. <https://learn.microsoft.com/en-us/azure/logic-apps/logic-apps-batch-process-send-receive-messages>
            2. Or use something else like an Azure Data Factory?
         6. Use the built-in connectors and custom connectors where possible to improve the performance and security of your integration.
            1. <https://learn.microsoft.com/en-us/azure/logic-apps/custom-connector-overview>
            2. Built-in connectors are optimized for high performance and reliability, and have been thoroughly tested for scalability, which reduces the risk of performance issues in your integration.
            3. Built-in connectors use secure authentication and encryption methods to ensure that sensitive data is protected during transmission and storage. Additionally, custom connectors can be built to meet specific security requirements and can be tightly integrated with other security measures, such as Azure Active Directory, to further improve security.
         7. Monitor your Logic Apps using Azure Monitor or another monitoring solution to ensure they are running as expected.
            1. https://learn.microsoft.com/en-us/azure/logic-apps/monitor-logic-apps-log-analytics
         8. Use Azure Key Vault to securely store secrets and keys
            1. https://365bythijs.be/2021/11/16/interacting-with-key-vault-from-logic-apps-securely/
         9. Use Source control like Azure DevOps or GitHub to keep track of versions and manage deployments of your Logic App.
            1. https://automationadmin.com/2021/08/source-control-logic-apps#gsc.tab=0
         10. Use Azure Resource Manager (ARM) templates for automating deployment and scaling of Logic Apps.
             1. https://medium.com/@sanjeevm/ci-cd-with-azure-logic-apps-2b8ed0fa2c2d
   2. Strategies for scaling and managing Logic Apps in production
      1. Use Azure Resource Manager (ARM) templates: Use ARM templates to deploy and manage Logic Apps, as it allows for versioning and repeatable deployments.
      2. Use Azure Monitor: Use Azure Monitor to collect and analyze log data, and set up alerts for important events.
      3. Use Azure Resource Manager (ARM) policies: Use ARM policies to enforce compliance and governance policies on resources.
      4. Use Azure Automation: Use Azure Automation to automate repetitive tasks and to manage the lifecycle of resources.
      5. Use Azure Key Vault: Use Azure Key Vault to store and manage secrets and certificates that are used by Logic Apps.
      6. ~~Use Azure ExpressRoute: Use Azure ExpressRoute to connect to on-premises resources, and use it to create private connections between Azure datacenters and on-premises resources.~~
      7. Use Azure DevOps: Use Azure DevOps to manage the code, testing, and deployment of Logic Apps.
      8. ~~Use Azure Backup: Use Azure Backup to back up Logic Apps and associated data.~~
      9. ~~Use Azure Scale set: Use Azure Scale set to automatically scale Logic App instances based on demand and configure auto-scaling rules.~~
   3. What about API Management?
      1. Azure API Manager (APIM) is a service that helps developers to publish, secure, and manage APIs. It allows you to create, manage, and publish APIs, as well as to set policies, monitor usage, and secure access to the APIs. Logic Apps, on the other hand, is a service that enables developers to create workflows and integrate systems, services, and applications.
      2. When using Azure APIM with Logic Apps, you can use APIM to manage and secure access to the Logic App's endpoint. This allows you to control access to the Logic App's endpoint, set policies, and monitor usage of the Logic App's endpoint. You can also use APIM to create and manage API keys, which can be used to authenticate and authorize access to the Logic App's endpoint.
      3. Additionally, APIM allows you to create and manage custom connectors that can be used in Logic Apps. This allows you to expose internal services and systems as APIs, which can then be consumed by Logic Apps. In this way, Logic Apps can integrate with internal services and systems that are not directly accessible through connectors or triggers.
      4. You can also use APIM to create and manage API definitions, which can be used to create Logic Apps. This allows you to create Logic Apps from API definitions, which can be used to automatically generate the logic app's workflow and integration.
      5. Overall, Azure API Manager and Logic Apps are two services that can be used together to enable end-to-end API management and integration scenarios.
   4. Security considerations for Logic Apps integrations
      1. Secure connections: Use secure connections, such as HTTPS or SFTP, to transfer data between systems and services also use secure storage.
      2. Authentication: Use authentication to secure access to the Logic App's endpoint and to the systems and services that the Logic App integrates with.
      3. Authorization: Use authorization to control access to the Logic App's endpoint and to the systems and services that the Logic App integrates with.
      4. Encryption: Use encryption to protect data in transit and at rest, both within the Logic App's endpoint and within the systems and services that the Logic App integrates with.
      5. Token management: Use token management to securely store and manage access tokens, and to validate and refresh tokens as needed.
      6. Key management: Use key management to securely store and manage encryption keys and other secrets.
      7. Vulnerability management: Use vulnerability management to identify and address security vulnerabilities in the Logic App's endpoint and in the systems and services that the Logic App integrates with.
      8. Monitoring and logging: Use monitoring and logging to detect and respond to security incidents, and to improve security over time.
      9. Compliance: Make sure that your integration meets the compliance requirements of your organization, and the industry standards.
   5. Integration with on-premises systems using the hybrid integration runtime
      1. Can’t any more
   6. F&O Best Practices
      1. Loosely Coupled
      2. Use Reliable message processing
      3. “Idempotent”, 1 input – 1 output
      4. Design for Reuse / Abstract out
      5. Add Security
      6. Add Exception-handling plus run after
      7. Add Monitoring
      8. Use Design Patterns
      9. Use Logical Name and Comments
      10. Use Parameters
   7. Activities
      1. Activity 7: Create a logic app to remove from a queue, process, place in another queue, create another logic app to process message
6. Deep dive into Logic App components
   1. Components
      1. Trigger types
         1. Recurrence
         2. HTTP
         3. APIConnection
         4. Manual
         5. HTTPWebhook
         6. APIConnectionWebhook
   2. Designing integrations with examples
      1. List from previous presentation
   3. Review of all Logic App components
      1. Implementing error handling and retry policies in Logic Apps
      2. Monitoring and logging for Logic Apps integrations
7. CI/CD with DevOps and Azure Monitor
   1. Integration with Azure DevOps and Azure Monitor for continuous integration and delivery (CI/CD) and monitoring
      1. <https://learn.microsoft.com/en-us/azure/logic-apps/monitor-logic-apps-log-analytics>
      2. <https://learn.microsoft.com/en-us/azure/logic-apps/set-up-devops-deployment-single-tenant-azure-logic-apps?tabs=azure-cli>
   2. Integration with Azure Resource Manager templates for deployment and management of Logic Apps
      1. <https://learn.microsoft.com/en-us/azure/logic-apps/logic-apps-azure-resource-manager-templates-overview>
      2. <https://learn.microsoft.com/en-us/training/modules/create-deploy-logic-apps-using-arm-templates/>
      3. <https://vaibhavgujral.com/2020/07/17/deploying-a-logic-app-using-azure-devops-yaml-pipeline/>
8. Real-world eCommerce integration
   1. Overview of the integration
   2. Components of the integration
   3. Activities
      1. Activity 8: Demo of the integration
9. Q&A and wrap-up
   1. Summary of the workshop
   2. Q&A session for attendees
   3. Next steps for attendees
   4. Activities
      1. Have the attendees ask questions and share feedback about the workshop.
      2. Have the attendees share next steps or plans for applying what they've learned to their own work or projects.