## How to track business processes across multiple batch processes?

* Assess existing logging and tracing infrastructure.
* Use Correlation ID’s across jobs to log and monitor status of a business process.
* The correlation id can be generated by entry point of business process or when a user request is accepted by front end system. Correlation ids will be mapped to an instance of business process.
* Develop a monitoring system (refer SEM based products like Splunk). If there are any similar systems across WK, those can be adopted.
* The system should provide mechanism to define a virtual business or background process which needs to be tracked across multiple applications. It should be possible to compose business process with parent – child relationship.
* The virtual process will consists of steps (mapped to a batch process or application) which can be tracked. There should be provision to set timeout for a step, retry counts etc.
* Batch processes and applications will emit messages to queues (messaging infrastructure)/or logs which can be tracked by monitored system. The messages should adhere a specific formats, and should not log sensitive information.
* The monitoring system should provide mechanism to watch queues (messaging infrastructure) to track dead letter queues.
* The monitoring system can be extended as process manager, which can not only monitor messages/events but can also send commands or trigger a batch job.
* Recommended patterns for implementation are Sagas, Mediator, and Scheduler Agent Supervisor.

## How to secure communication between services?

Note: It is assumed that both WK service and partner service are hosted with HTTPS

**Option A: Certificate based authentication**

Offline Process: Establishing Trust (Partner onboarding)

* Partner will create a self-signed certificate and share the public key with WK
* WK will register the certificate against the partner with its system, For example, p01 with Key k01.

At Runtime

* While calling the WK Service, Partner service will encrypt the message with its private key (not known to WK)
* Partner will also pass it’s identify via header. For example p01.
* WK will decrypt the message with associated key from partner and ensure that message has indeed received from a particular partner.
* As a further enhancement, during offline process additional metadata like states, counties associated for that partner can be captured. This metadata can be used to validated/authorize partner.

**Option B: JWT tokens issued by trusted security token provider**

Offline Process

* The service has trust relation established with STS

At Runtime

* While calling the WK Service, Partner service will first approach STS to get a token for WK Service.
* STS (on authentication – Cert based) can issue a token with claims like (name, or custom claims like allowed states, allowed properties).
* WK Service will check the claims from JWT token.
* This approach of federated user/service identity can used to secure various services to authenticate and authorize internal as well as external services.

**Option C: Issue a token and randomly generated URL for callback.**

Part 1: WK calls Partner service

* Provide required information (Lien details, state, county, some unique identifier)
* Provide a token (encrypted using a key – not known to partners)
* A random generated URL like <http://address/callback/9923e165-bc4e-4c4c-8fee-12d6efbeb056>. The randomly generated url needs to more than a GUID.
* It will also internally maintain a mapping between the randomly generated id (9923e165-bc4e-4c4c-8fee-12d6efbeb056) and some unique identifier)

Part 2: Partner service calls back

* Provide status
* Pass token as apart of header
* Pass unique identifier
* WK will validate the mapping of token, random url, and unique identifier. For further security the random url will be expired once used.
* This is the least secure option of three, but can be considered as time gap arrangement

## Adopting patterns for Reliability, Availability, Resiliency

1. Circuit Breaker – While consuming external services, using frameworks like Polly helps applications to prevent performing an operation which is likely to fail. Polly framework offers multiple resilience policies like retry, circuit breaker etc.
2. External Configuration Store – One of the pain point was configuration management across multiple batch processors and applications. An external configuration store can help to reduce this pain, application can request required configuration from store. This can be also extend as secret manager.
3. Breaking monolithic application into smaller separately deployable units (Microservices) – If an application needs to collect data from user and submit it to 3rd party service. The default option will be consuming the 3rd party service from web application itself. This may work, but may impact resiliency if the 3rd party service is down. To improve resiliency, web application can be divided into two part. The front end application can collect data from end users, store it locally, later a background process can submit the data to 3rd party service.