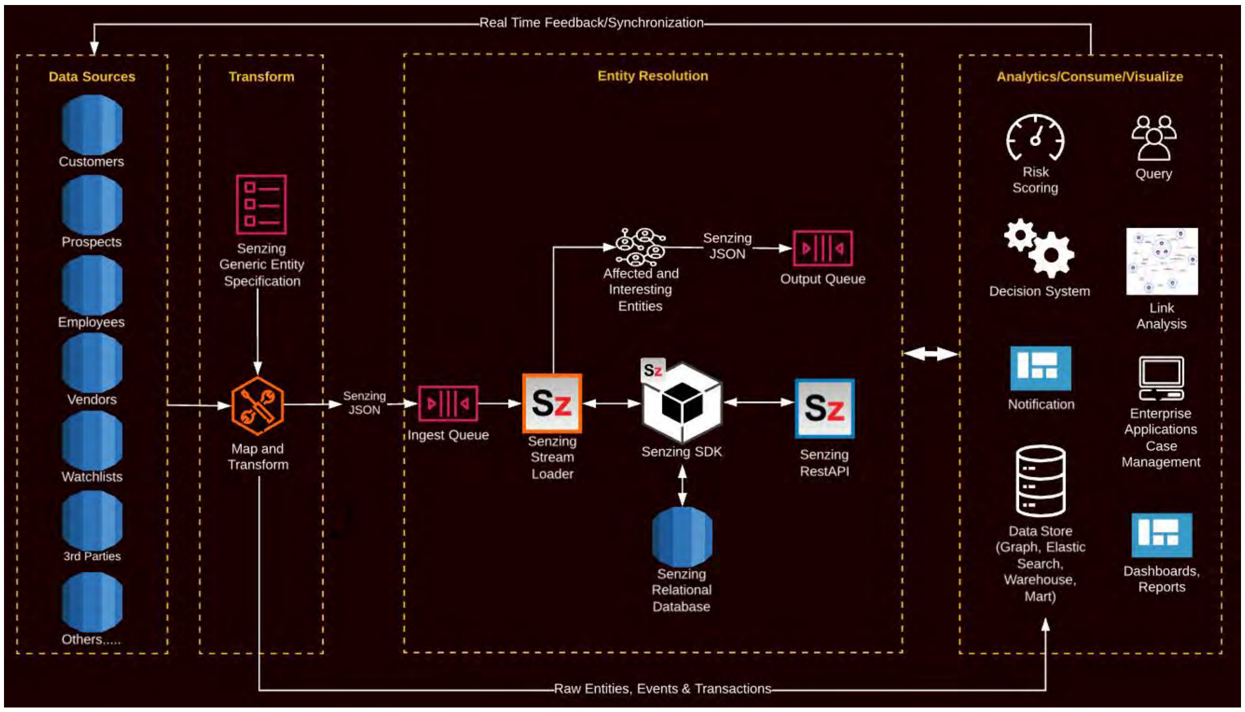
Senzing

ARCHITECTURAL PATTERN FOR PERPETUAL INSIGHTS

Use this architecture for systems that are making real-time decisions about new or existing entities.

The perpetual insights pattern covers common use cases such as: Know Your Customer (KYC), Customer On-Boarding, Transaction Monitoring, Background Checks, Supply Chain Vetting, etc.



Data Sources: A contributing data source will contain structured person or organization information. A data source will be a data store (e.g., SQL, NoSQL, NEO4J, MongoDB, HADOOP), a file (e.g., CSV), or a queue containing records. Loading historical data is handled by submitting all records in the data source to the transform function. After historical data has been loaded, adds, changes and deletes caused by ongoing business operations are to be submitted as transactions. Some data sources like Master Data Management (MDM) and Know Your Customer (KYC) systems may also receive real-time updates from the entity resolution process e.g., ensuring new duplicates are not being created.

Transform: Each data source must be mapped into Senzing JSON as defined by the Senzing Generic Entity Specification. Some popular data mappers are available in our open source community here. We encourage you to contribute to this open source effort. As entities are mapped to JSON they are placed onto the Ingest Queue (e.g., RabbitMQ, Kafka, AWS SQS). Example docker demonstrations can be found here.

Entity Resolution: The Senzing Stream Loader reads records off the Ingest Queue. The Stream Loader calls the Senzing SDK where the entity resolution takes place (more here). Senzing stores all the received records, the entity resolution assertions and the entity resolved graph in a relational database (e.g., PostreSQL, AWS Aurora, Db2, MySQL). See Senzing system requirements here. Results from each entity resolution transaction are posted, in real time, to the Output Queue in Senzing JSON. This will include both affected and interesting entities (more here).

Analytics, Consume & Visualize: Real-time processes e.g., risk scoring and decisioning systems will read the Output Queue and direct further action, as desired e.g., responding to an MDM or KYC source system in real-time. The Output Queue can also keep other data stores in sync – which could include NEO4J, Elasticsearch, reservoirs, lakes, warehouses, marts, etc. The Senzing RestAPI and synchronized data store are called by end user applications providing such capabilities as user search, graph visualization, enterprise applications e.g., case management, dashboards, reporting, etc.