|  |  |  |
| --- | --- | --- |
|  | Date | 9th January 2019 |
|  | Reference | Brit Data Library – Reference Architecture |
|  | Version | V0.4 |
|  | File | Data Library - Reference Architecture - 0.5.Docx |



**

**Contents**

1. Core concepts 3

2. Conceptual architecture 3

3. Tool set review 4

4. Logical architecture 6

4.1. Data Sources 7

4.2. Data Lake 13

4.3. Data Modelling 20

4.5. Data Access 22

5. Cost model 24

6. Access and security 26

6.1. Data Security 26

6.2. Network Security 27

7. Operations 34

7.1. Azure DevOps 34

7.2. Standards 34

7.3. Monitoring of deployments 34

7.4. Platform Testing: 35

7.5. Resilience, Failover and DR: 35

8. Scalability 38

9. Requirements Matrix 39

**Document Control**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Who | What |
| 0.1 Draft | 30 November 2018 | Angus Mack | First draft |
| 0.2 Draft | 03 December 2018 | Angus Mack | Enabled track changes, updated curated data modelling, … |
| 0.3 Draft | 08 January 2019 | Angus Mack | Further updates based on network and DevOps discussions |
| 0.4 Draft | 09 January 2019 | Angus Mack | Updates to Exposure pattern, typos, diagrams, ToC and other corrections |
| 0.5 Draft | 28th February 2019 | Angus Mack | Updates to reflect latest changes in approach (Twitter/AddressCloud) + numerous other minor factual corrections |

# Core concepts

This reference architecture and design stage will define the initial architecture and high-level design of the Data Library and include a review of candidate technologies and a selection of the key technology components.

The architecture will incorporate the following elements:

* Data processing layers, for example raw, cleansed and access/presentation layers
* Integration with the MyMI Data Warehouse, including the integration of MyMI business

logic into the Data Library

* Integration of data quality and data reconciliation logic into the Data Library
* Resource management and Azure consumption cost monitoring
* DevOps management. including continuous integration/continuous deployment (Cl/CD)

pipelines

* Platform segregation and security.

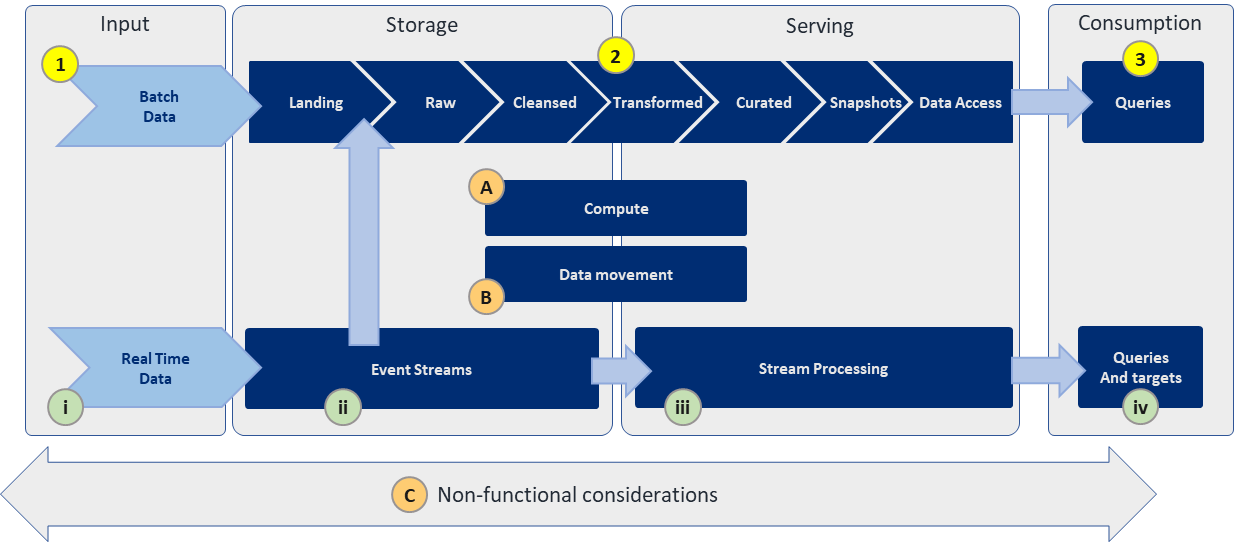
The success criteria for this POC has been written by Graham Binner and this is included in the Section 9, where each point is cross referenced against the features and services that implement it.

MyMI is the current Data Warehouse built on SQL Server 2016 and serves as the current provider of data to the MI/BI function within Brit Insurance.

# Conceptual architecture

The Data Library Data Lake architecture will be based upon the Altius Reference Data Platform architecture and extended and modified as per the requirements outlined above and below.

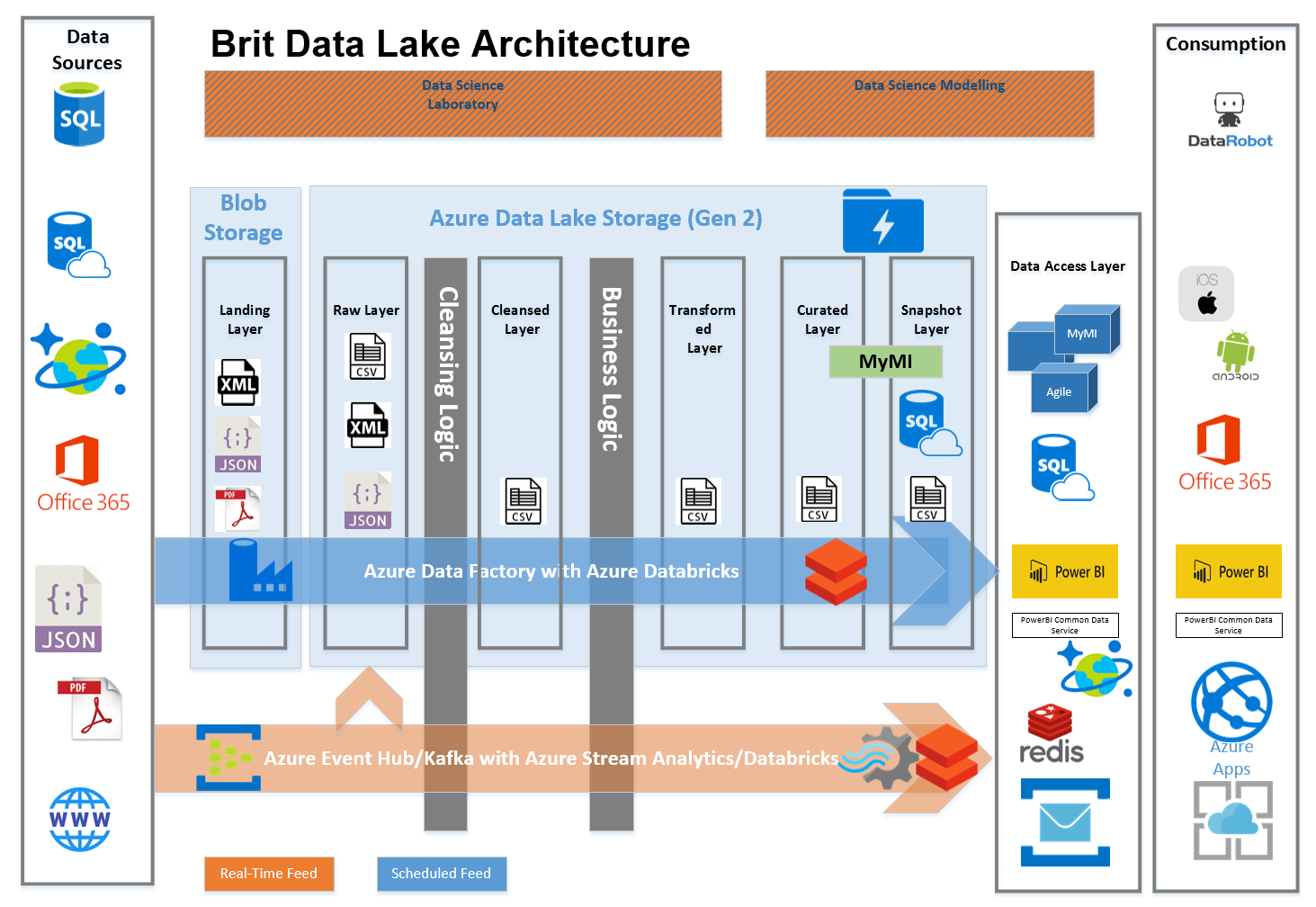
The basic concept is illustrated below:



Brit Insurance have deviated slightly from the Altius reference layer structure, based on their business needs.

|  |  |
| --- | --- |
| **Altius Reference Layer** | **Brit Data Library Layer** |
| Landing | Landing |
| Raw | Raw |
| Cleaned | Cleansed |
| Aligned | Transformed |
| Conformed | Curated |
| - | Snapshots |
| - | Data Access |

The customisation of the Altius reference architecture for Brit Data Library Data Lake aligns as below:



# Tool set review

Azure first, SaaS/PaaS first. This is the wider Brit strategy, and Azure has been selected as the platform of choice for this project.

Given the general wind down of ADLA and ADLS Gen1 availability, combined with the need to build a platform that will be supported over the next 5+ years, Databricks will be deployed to cover those situations where a high-power compute and ELT/ETL/large-scale mapping and other operations on the source data sets is required.

Azure Data Lake Storage Gen2 will be combined with Azure Data Factory for data storage and movement throughout the layers of the Data Lake.

The use of Azure Databricks may highlight some shortcomings compared with ADLA e.g. loss of investment in USQL scripts, increase in effort to learn technology and to reimplement these scripts in Python for example.

This is expected to be short-lived and the emerging adoption of Python as the scripting language of choice means that this is a sound investment in time and resources.

Region of choice: UK South. Although it lacks “Hero region” status\* this is the current location of Power BI tenant and there is already a virtual network route configured over Express Route into Azure subscription.

\*Hero regions are specific Data Centres – one in each geo-graphic region – that have all of the available Azure services deployed and a wider selection of SKUs for each service.

The current list of Azure services to be deployed is:

* Azure Blob Storage (v2)
* Azure Data Lake Store (Gen2)
* Azure Data Factory (v2)\*\*
* Azure Databricks (Python)
* Azure SQL DB
* Azure CosmosDB (Cassandra API)
* Azure Functions (C#)
* Azure Event Hub (Kafka enabled)
* Azure Stream Analytics
* Azure App Service Plan

The detailed list of SKUs will come out of the delivery Sprints and once the final design has been tested and approved.

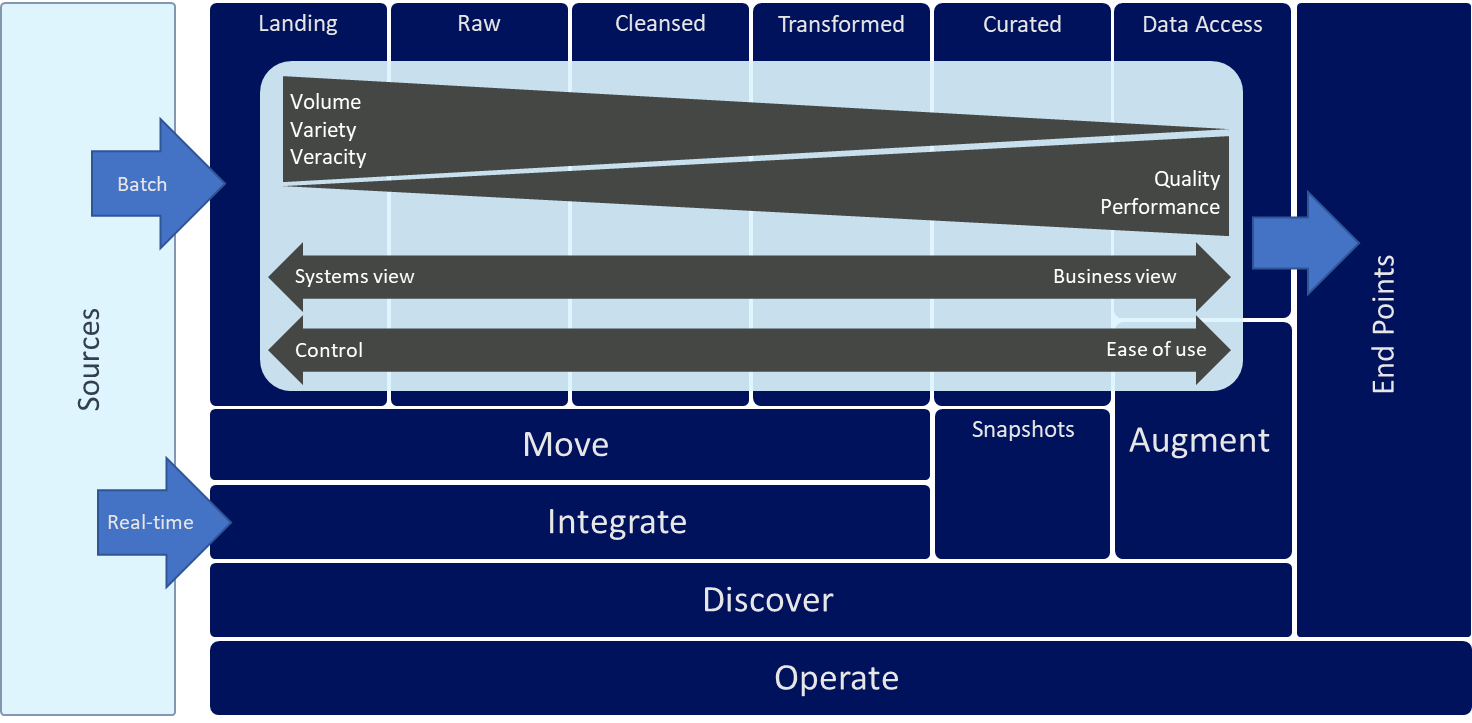
\*\*There is a possibility in the future to use the new Azure Data Factory Data Flows functionality – currently in public preview - as a way to do some of the work that would be done by Databricks notebooks and - more traditionally - SSIS packages on premise.

The scale of data operations required would have to be aligned to the possible compute and processing power available within Data Flows. More research will be needed to evaluate this option. N.B. this has nothing to do with the data flows feature in PowerBI.

# Logical architecture

For the scope of this work in the POC, five data sources have been identified to be accessed and stored in the Data Library.

The core principals of data storage and flow shown below will be applied to the Data Library Data Lake platform once it goes to production.



The data sources in scope and targeted for the POC by Brit, in order of importance are:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Data Source | Source Format | Volume | Structure (Variety) | Velocity | POC Benefits | Business Area |
| Eclipse  (GGWP, GNWP, Policy Info) | SQL Server | Low | Structured | Semi Real-Time/Real-Time | Example of using SQL Source.  Business logic layer.  History partitions  Consider trickle-feed mechanism to speed up MyMI load.  Consider integration of Data Lake with real-time feed for OPUS and other apps | Policy administration, premium, claims |
| Oceanwide | XML | Low | Semi-structured | Monthly | Example of semi-structured data | Claims – BGSA |
| Exposure | SQL Server | High – 6 billion rows per year + 3 billion (current yr)  (Estimated 750 million rows per model, 6 models per quarter – 2 retained. Potentially 750m \* 4 quarters \* 2 models + 4 current models =  jobs per year) | Structured | Quarterly | High data volumes are a good test for scalability.  Prove out use of Azure Databricks and compare to POC already built with USQL/ADLA | Cat Lines - Property |
| Address Cloud | JSON via RESTful API | Low / Medium | Semi-Structured | Tbc | Prove mechanism to load JSON from external sites to Brit data lake using RESTful API  Demonstrate suitability of Data Lake architecture for requirements of Data Analytics Team | Property |
| Twitter | Web *Feed (JSON)* | High | *Semi-*structured | *Real-Time* | Example *of:*  Realtime*, JSON, external web feed* | TBD by Data Analytics Team |

## Data Sources

High level decision that incoming XML to landing zone will be written as JSON directly to the Raw Layer in the data lake.

* + 1. **Eclipse**
       1. Overview + Proof points

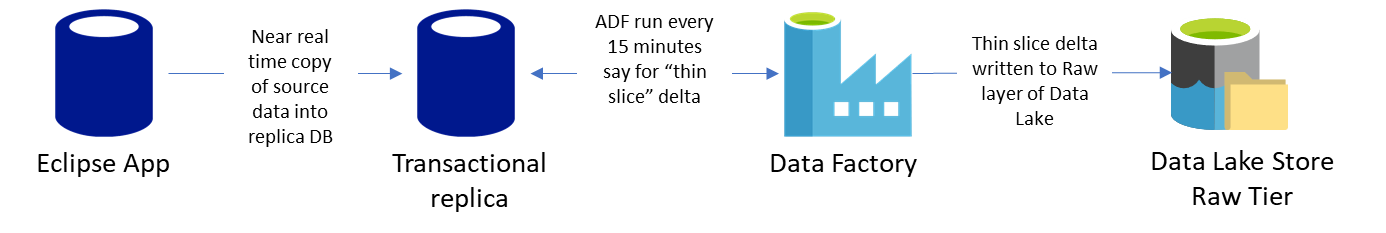
Main store for policy administration, premium and claims data.

The reason for accessing this is that it is a SQL data source that is updated regularly. We want to demonstrate a trickle pattern of ingest and also show how business logic can be applied to this source and Oceanwide.

* + - 1. Ingestion

Current limitations and issues with MyMI are that it relies on a bulk load in off-peak time. The introduction of a new time zone in Singapore combined with increasing data volumes means this approach is impractical.

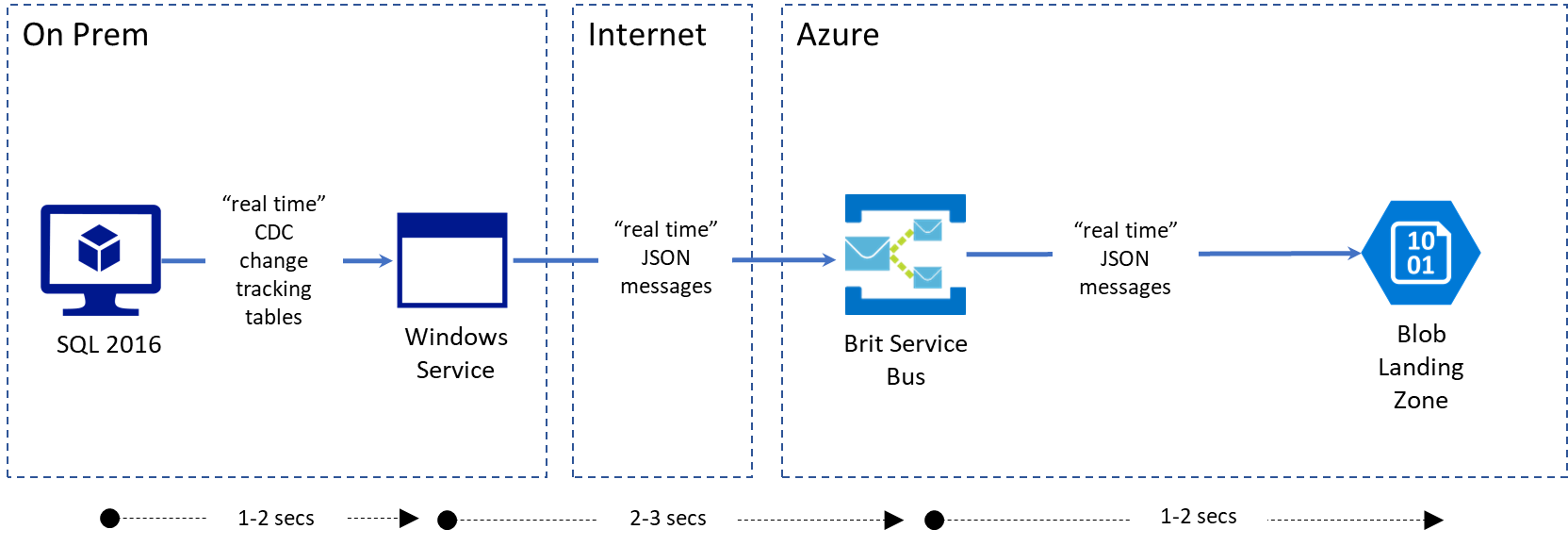
The first option considered was to use a standard Data Factory copy pipeline (on the transactional replica database) and a 15-minute time slice to write to the Data Lake Raw tier.



This was rejected as it didn’t really address the requirement for near real time updates, nor did it do anything innovative and prove a new ingest pattern.

The approach decided upon is to enable Change Data Capture (CDC) on the source database and relevant tables and then read from these CDC tables on a regular basis e.g. every 5 seconds.

These events are sent to a topic on the existing Brit Service Bus, where they are read and written directly to Data Lake Landing Zone.



There are several excellent blog posts and resources about this approach:

<https://mrfoxsql.wordpress.com/2017/07/12/streaming-etl-send-sql-change-data-capture-cdc-to-azure-event-hub/>

<https://azure.microsoft.com/en-us/resources/samples/event-hubs-dotnet-import-from-sql/>

<https://github.com/rolftesmer/SQLCDC2EventHub>

The advantage is that latency is reduced to a few seconds (<10s) and the overall “strain” on the data pipeline is reduced by removing the bulk load.

An Azure Function was considered as the service to poll the CDC tables and retrieve the data but the added overhead of hosting a Function 1) in an App Service Environment (ASE – very expensive) or 2) making two hops through a VNET in West Europe and then one in UK South so that it can use the Express Route into the Brit DC was considered too risky and time consuming for this PoC.

Using the example source code already provided in the GitHub link above – a C# console application – the basic functionality needed has already been demonstrated. There is some overhead in writing and maintaining source code however once written it could be used repeatedly and should require very little effort to keep running.

* + - 1. Data History

In MyMI repeated daily snapshots are taken of Eclipse, Oceanwide, Velocity etc. and these are used to build reports over a defined reporting period e.g. quarterly.

In order to negate the need for these and to leverage the power of the Data Lake the data history will be implemented in the Snapshots layer.

The data required for the various periodical reports will be built from the in-scope sources in a controlled and repeatable manner.

* + - 1. Dimensional Modelling

Slowly Changing Dimensions - Type 2 (Graham Binner will define the exact dimensional model during the Sprint planning phase).

The dimensions targeted for this POC are as follows:

* Policy (attribute: Inception date)
* Currency
* Year of account
* Policy & policy Line
* Group class
* Policy status

Calculated outputs based on business logic are:

* Gross gross written premium (original currency)
* Gross net written premium (original currency)
  + - 1. Business Logic

This is currently carried out in the Data Warehouse MyMI. It makes sense to move this into the processing pipelines of the Lake – in the Transformed layer.

Given the choice to use Databricks as the main processing engine for the data in the Lake the business rules would need to be implemented in Databricks notebooks using Python. They will be written into a common library that will be shared across data source ingest pipelines.

There is the option to call stored procedures in remote databases from Databricks, but this would defeat the point of the distributed processing capabilities of Spark and be very inefficient.

* + - 1. Repeatable ingest pattern

This pattern could be applied to other systems e.g. Velocity and other near real time SQL-like sources.

Moving data from Raw to Curated – in the ADF pipelines, the data is split then re-joined where they have a dependent layer e.g. 3 transformed sources need to go to the Curated layer.

* + 1. **Oceanwide**
       1. Overview and Proof points

This is a different underwriting system based on a subset of the US business (yacht insurance premiums).

It is a third-party SQL server-based system.

There are daily incremental SQL backup files, with a weekly full backup. This backup is restored locally and the tables contain XML columns, which contain the relevant data.

This demonstrates the handling of semi-structured data, with the conformation of multiple data sources and the sharing of business logic across source systems.

It will also show the handling of variant schemas in the same source: the structure of the XML stored in the columns of the tabular data is not the same for each record type.

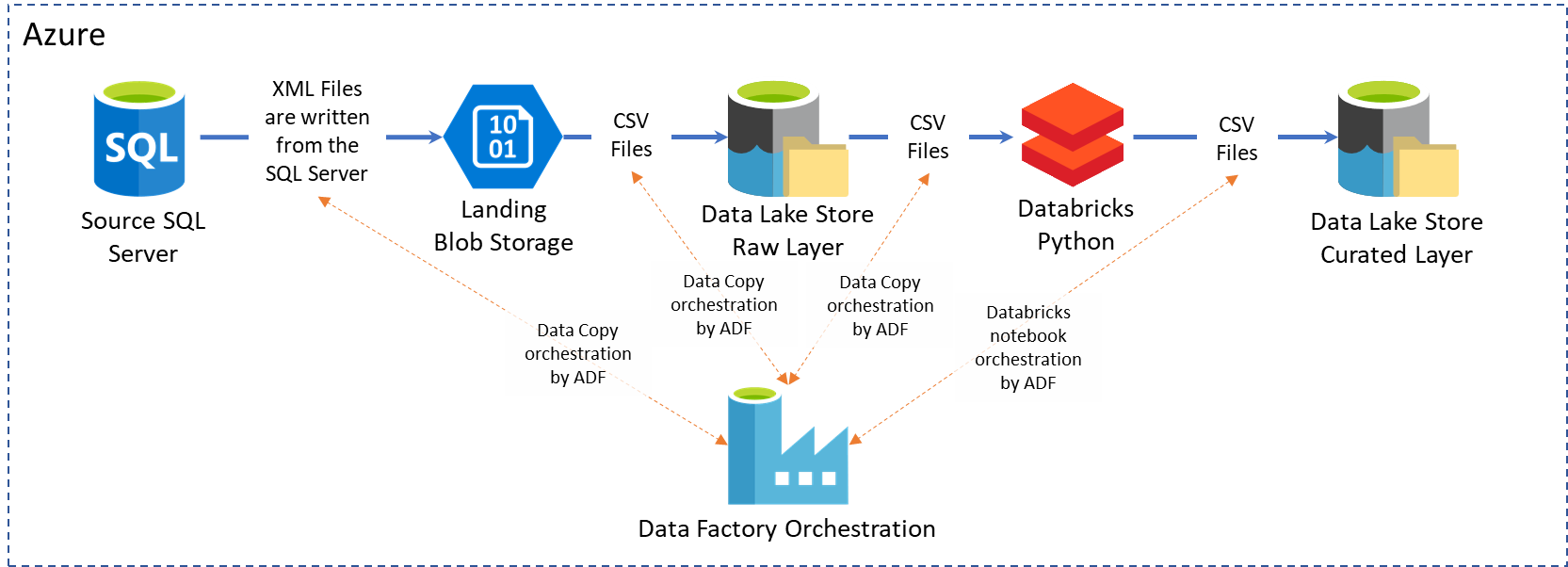
* + - 1. Ingestion

For this POC phase XML files will be provided to the Landing layer, one file per transaction, possibly two files, where one file with the top-level metadata that is common and one with the relevant transaction XML.

The data is stored in XML columns in the SQL source – this data is serialised web form data.

Not all XML nodes are populated, there are also arrays of transactions in the XML data. Not all arrays are the same schema, and objects in same array may have same schema, and some array elements have more or fewer rows.

Data Factory starts a Data Copy task to export data from the Oceanwide source SQL Server and writes the exported content to the Landing Zone. These files are then copied to the Raw Layer of the Data Lake by Data Factory. Databricks notebooks are the executed by ADF, passing in the locations of the folders of the files and these carry out the required cleansing, transformations and conformations to deliver the final data to the Conformed Layer of the Data Lake.



* + 1. **Exposure**
       1. Overview and proof points

This source contains a prediction of the total financial exposure, which is based on external events plus internal sources (listed above).

* Catrader (SQL)
* Touchstone (SQL)
* Manual (SQL)

This is updated quarterly. It has both an adjusted (Catrader + Touchstone + Manual) and an unadjusted model (which does not include the manual feed) calculated over a quarter.

The function is to predict Brit’s total exposure based on the contracts that have been written.

It is a SQL Server source – currently SSIS is used to write to the Raw layer of the MyMI.

For the PoC, Data Factory will be used with a self-hosted Integration Runtime that will pull the relevant subset of the contents of the Exposure tables. This data will be written to the Raw Layer of the Lake. There are 6, possibly 7, dimensions being used to define the subset of data needed from the source Exposure tables.

Rishi Gupta at Brit has already done a lot of work using ADLS Gen1 and USQL scripts combined with Data Factory to import and process data.

This functionality is ready to test porting to Databricks.

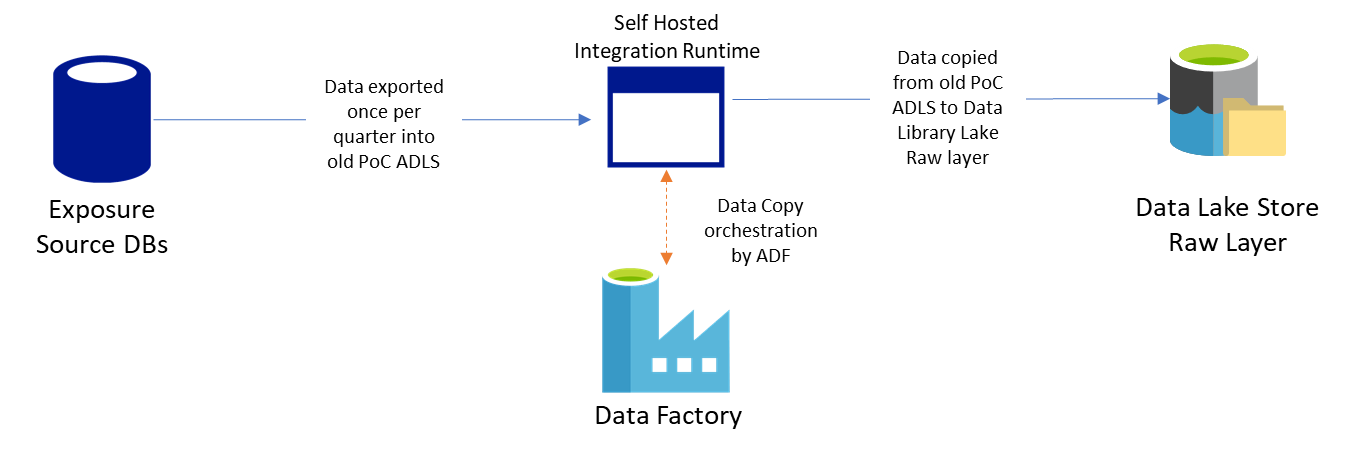
Python will be used in Databricks notebooks as well as porting the USQL scripts to SQL in Databricks notebooks.

In was initially thought that the performance on SQL on Databricks might be too slow to compare with USQL on ADLA. However, looking at the Azure Databricks documentation here <https://databricks.com/blog/2015/04/24/recent-performance-improvements-in-apache-spark-sql-python-dataframes-and-more.html> Spark SQL is actually very performant and there is definite benefit in trying the conversion of the USQL jobs to Spark SQL. This will be a useful test in the Sprints.

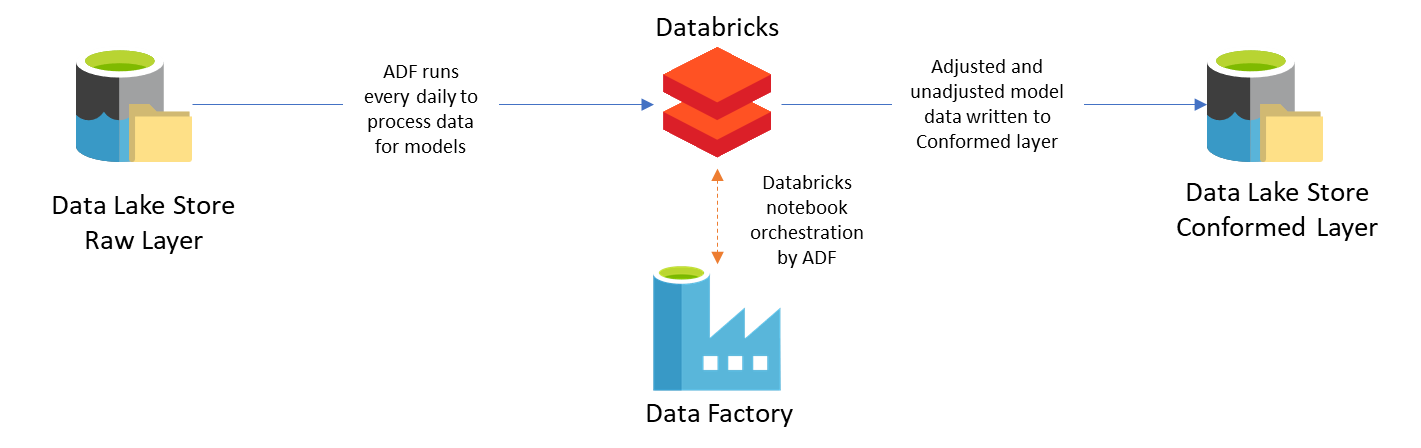
The other main proof point of this source is to look at a price/ease of use comparison with ADLS Gen1 vs ADLS Gen2 with Databricks and the relative costs in terms of effort associated with each approach. This will come out in the Cost Analysis phase.

* + - 1. Ingestion

The Exposure source data tables will be brought into the Raw layer of the Data Lake using a Data Factory pipeline.



Once landed in the Raw Layer an additional Data Factory pipeline will process the Exposure data and move it through the Lake layers to build the adjusted and unadjusted model data.



* + 1. **Twitter**
       1. Overview and proof points

The Data Science team at Brit have a requirement to ingest data from multiple sources including data from APIs.

In order to prove the design patterns of capturing data from both a streaming API and static REST API the Twitter firehose was chosen for streaming. Hazard Hub below is the static API endpoint chosen.

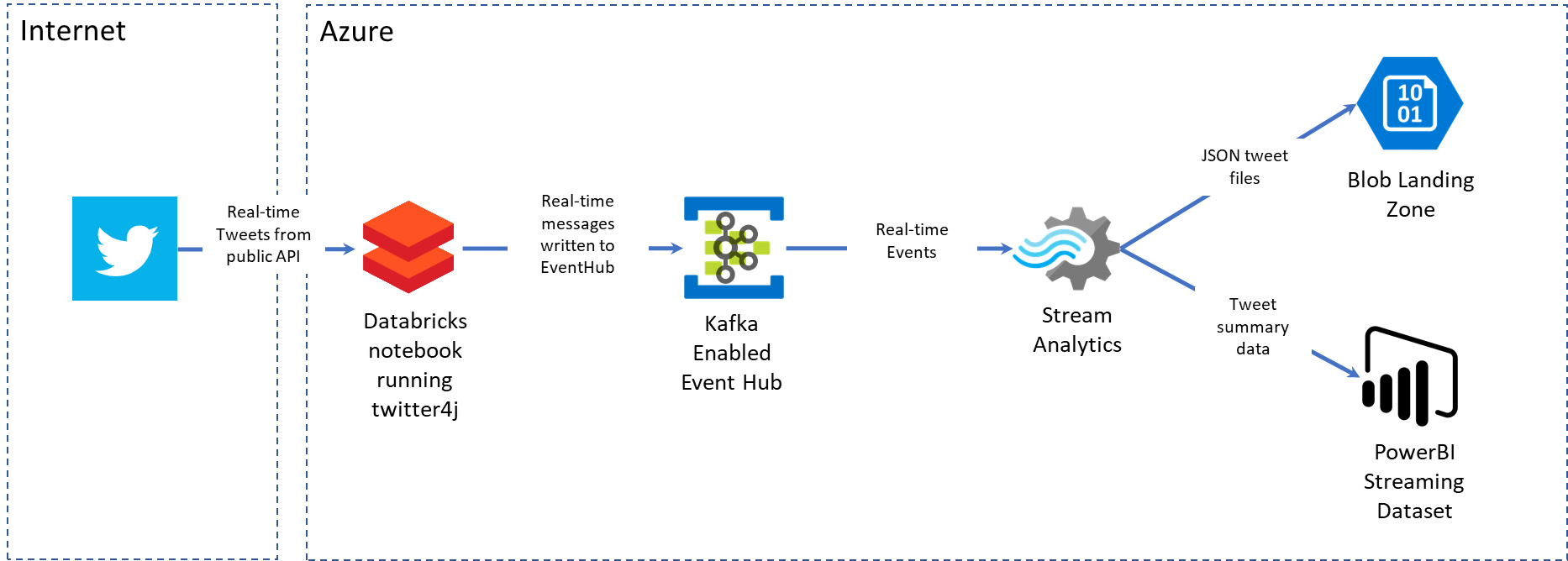
Although this source may not have immediate business benefit on its own, it serves to prove the capability of handling a streaming real-time source.

* + - 1. Ingestion

The client listening to the Twitter API needs to be kept open all the time. It was initially thought that this could be achieved with Azure Functions but in testing this proved impossible due to the timeouts imposed on this serverless service.

Instead, a Databricks notebook running on a cluster is able to achieve this. Real time tweets are streamed to an EventHub and then processed with Stream Analytics, writing both the raw output to the Blob Landing zone and a subset of the data to a streaming dataset in PowerBI.

Kafka is of interest in other areas of Brit and Azure EventHub is capable of supporting both its on message format and Kafka message format.



The streamed event data is serialised to Blob storage.

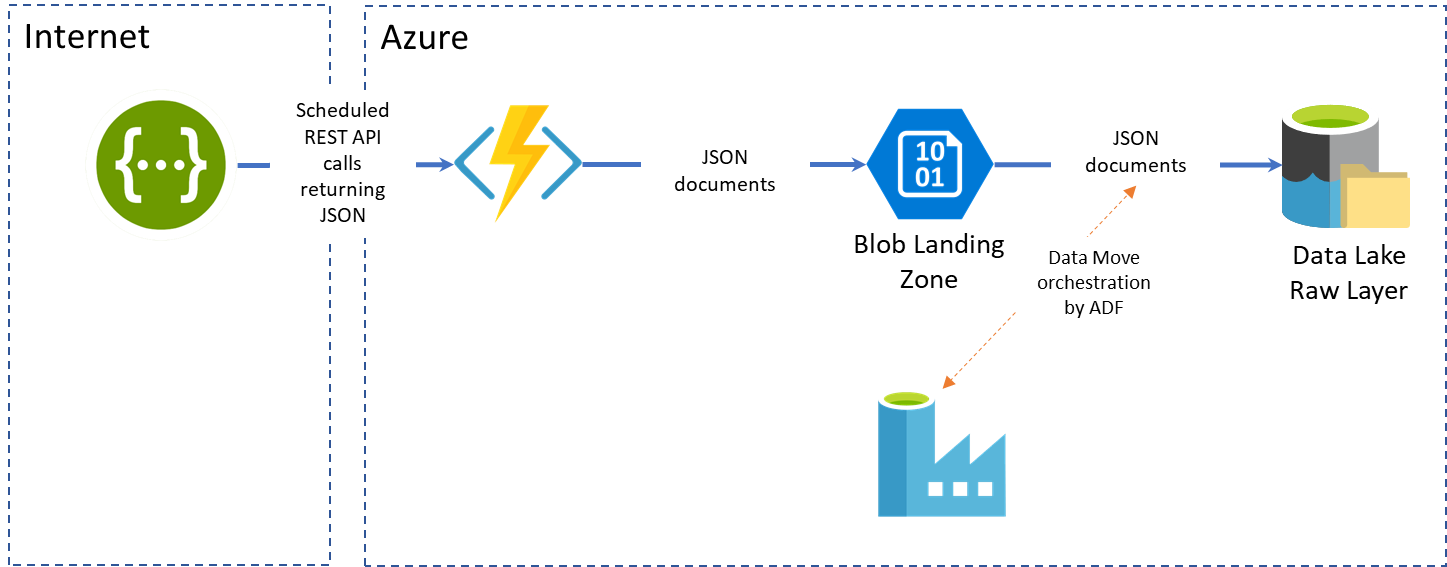
If there is heavyweight business logic that needs to be applied and other more demanding requirements, then Databricks can be used instead of Stream Analytics.

* + 1. **Address Cloud**
       1. Overview and proof points

This is an example of a REST API that will be called on an ad-hoc or scheduled basis.

The data fetched from the API is serialised to the Raw layer.

* + - 1. Ingestion



An Azure Function will call the API periodically and retrieve the event data in JSON format and serialize it as CSV files to the Raw Layer in the Data Lake.

## Data Lake

The Altius Reference Data Platform tiering structure will be applied to the Brit Data Library Data Lake, albeit with slightly modified names as detailed in section above.

There is a requirement to capture metadata in the scope of the POC but not build a catalogue that can be queried.

With this in mind detailed analysis of the source/entity and column metadata has been carried out and will be explained below.

In this section we cover:

1. Data Lake Layer definitions
2. Source Level Metadata
3. Entity Level Metadata
4. Column Level Metadata
5. Data Lineage Metadata
6. Release Catalogue

The tables on the following pages document the detailed architectural design work carried out in order to define the Layers of Data Lake and their function.

The metadata associated with each functional area of the Data Library and their relationships are also documented in the tables below.

These cover:

* Sources
* Entity
* Column
* Data Lineage
* Platform Version Releases

Each layer of the lake will be partitioned as per the Altius Reference Architecture with some modifications.

At the top level partitioning is as follows:

/layer/source/security/entity/

Where “security” is the Brit document security classification and entity is typically the source table name up to the Curated layer.

More details are given in the tables on the following pages.

* + 1. Data Lake Layer Names and Function

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Layer** | **Layer Code** | **Function/Content** | **Storage Type** | **Mandatory** | **Formats** | **Access** | **Folder Structure** |
| Landing | LA | Temporal (30 days max for GDPR), used for holding semi structured/unstructured e.g. XML from TPA | Blob | no | any | IT | /layer/source/security /entity/yyyy/MM/dd/hh/\* |
| Raw | RA | Contain the raw data from all sources. All subsequent layers can be rebuilt from here. | ADLS | yes | CSV, TSV, JSON, XML | IT + Data Science | /layer/source/security/entity/yyyy/MM/dd/hh/ |
| Cleansed | CL | Reformatted to be easier to consume, deduplication, error handling, null values etc. date formatting, unique key columns identified | ADLS | no | CSV | IT + Data Science | /layer/source/security/entity/yyyy/MM/dd/hh/ |
| Transformed | TR | Business logic applied here (source specific schemas) generic structures to support wide usage scenarios | ADLS | no | CSV | IT + Data Science + Reporting Team | /layer/source/security/ /entity/yyyy/MM/dd/hh/ |
| Curated | CU | Dimensional modelling to support agreed reporting scenarios | ADLS | no | CSV | IT | /layer/subject/security/entity/yyyy/MM/dd/hh  e.g. /Curated/MyMI/Public/EclipseDimClaim/2018/11/29/12/ |
| Snapshots | SN | Calendar and FY based snapshots as required | ADLS/SQL DB |  | CSV (SQL subsets) | IT + Data Science + Reporting Team |  |
| Data Access | DA | Data published for consumption by business users, reporting applications and downstream systems | AAS/PowerBI/SQL DB/Cosmos DB/Redis Cache/EventHub/MessageBus |  | as per storage type | All |  |

\*If an entity has any secret columns then the whole entity is classed as secret

* + 1. Source Level Metadata Fields

This table defines the source level metadata items that will be captured for each source.

Where the metadata item cannot be auto generated this will require a manual process for the information to be inputted.

This may be seen as a large overhead, but it will be necessary to ensure the success of the data catalogue and lineage functions.

It may be possible to predefine templates for the sources, so that once inputted once for a source, this information could be made available to future users in order to save on effort on increase accuracy and repeatability of the metadata captured.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Metadata Attribute | Example Value | Can be auto-generated | Raw | Cleansed | Transformed | Curated | Snapshots | Data Access |
| Source Name | Eclipse | Y | Y | Y | Y | Y | Y | Y |
| Layer | Cleansed | Y | Y | Y | Y | Y | Y | Y |
| Format | CSV | N | Optional | Optional | Optional | Y | Y | Y |
| Business Owner | Phil | N | Optional | Optional | Optional | Y | Y | Y |
| Business Area | Bureau | N | Optional | Optional | Optional | Y | Y | Y |
| Description | something useful | N | Optional | Optional | Optional | Y | Y | Y |
| Functional Area | Underwriting | N | Optional | Optional | Optional | Y | Y | Y |
| ValidFromRelease | **13** | **Y (on first load)** | Y | Y | Y | Y | Y | Y |
| ValidToRelease | 14 | N | Y | Y | Y | Y | Y | Y |
| UniqueID (layer+source) | **CL-Eclipse** | **Y** | Y | Y | Y | Y | Y | Y |

* + 1. Entity Level Metadata Definition

This table defines the entity level metadata items that will be captured for each entity.

Where the metadata item cannot be auto generated this will require a manual process for the information to be inputted.

As previously stated, this may be seen as a large overhead, but it will be necessary to ensure the success of the data catalogue and lineage functions.

It may be possible to predefine templates for the entities, so that once inputted once for an entity, this information could be made available to future users in order to save on effort on increase accuracy and repeatability of the metadata captured.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Metadata Attribute** | **Example Value** | **Can be auto-generated** | **Raw** | **Cleansed** | **Transformed** | **Curated** | **Snapshots** | **Data Access** |
| Entity Name | WrittenPremium | Y | Y | Y | Y | Y | Y | Y |
| Source Name | Eclipse | Y | Y | Y | Y | Y | Y | Y |
| Layer | Curated | Y | Y | Y | Y | Y | Y | Y |
| Business Entity Name | Written Premium | N | Optional | Optional | Optional | Y | Y | Y |
| Business Area | BGSA | N | Optional | Optional | Optional | Y | Y | Y |
| Description | something useful here | N | Optional | Optional | Optional | Y | Y | Y |
| Functional Area | Underwriting | N | Optional | Optional | Optional | Y | Y | Y |
| **ValidFromRelease** | **12** | **Y (on first load)** | Y | Y | Y | Y | Y | Y |
| ValidToRelease | 14 | N | Y | Y | Y | Y | Y | Y |
| **UniqueID (layer+source+entity)** | **CU-Eclipse-WrittenPremium** | **Y** | Y | Y | Y | Y | Y | Y |
| ReconciliationEntityID | Tr-Eclipse-WrittenPremium |  |  |  |  |  |  |  |
| **ReconciliationGroupByID** | **None, or list of comma separated list of columns** |  |  |  |  |  |  |  |

* + 1. Col 0adata

This table defines the column level metadata items that will be captured for each column.

Where the metadata item cannot be auto generated this will require a manual process for the information to be inputted.

As previously stated, this may be seen as a large overhead, but it will be necessary to ensure the success of the data catalogue and lineage functions.

It may be possible to predefine templates for the columns, so that once inputted once for a column, this information could be made available to future users in order to save on effort on increase accuracy and repeatability of the metadata captured.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Metadata Attribute** | **Example Value** | **Can be auto-generated** | **Raw** | **Cleansed** | **Transformed** | **Curated** | **Snapshots** | **Data Access** |
| Column Name | GrossNetWrittenPremium | Y | Y | Y | Y | Y | Y | Y |
| Entity Name | WrittenPremium | Y | Y | Y | Y | Y | Y | Y |
| Source Name | Eclipse | Y | Y | Y | Y | Y | Y | Y |
| Classification | Official | Y | Y | Y | Y | Y | Y | Y |
| Layer | Curated | Y | Y | Y | Y | Y | Y | Y |
| IsUniqueIdentifierComponent | Yes | N | Optional | Y | Y | Y | Y | Y |
| DataType | Numeric | N | Optional | Optional | Optional | Y | Y | Y |
| Usage | PrimaryKey, Attribute, ForeignKey, AggregatableMeasure, SemiAggregatableMeasure | N | N | N | Optional\* | Y | Y | Y |
| Business Column Name | Gross Net Written Premium | N | Optional | Optional | Optional | Y | Y | Y |
| Business Owner | Dave | N | Optional | Optional | Optional | Y | Y | Y |
| Business Area | Bureau | N | Optional | Optional | Optional | Y | Y | Y |
| Description | free text description of column | N | Optional | Optional | Optional | Y | Y | Y |
| **ValidFromRelease** | **12** | **Y (on first load)** | Y | Y | Y | Y | Y | Y |
| ValidToRelease | 14 | N | Y | Y | Y | Y | Y | Y |
| **UniqueID (layer+source+entity+column)** | **CU-Eclipse-WrittenPremium-GrossNetWrittenPremium** | **Y** | Y | Y | Y | Y | Y | Y |

Primary Key = UniqueID + ValidFromRelease, \*the use of the Usage column here implies a drift away from schema-on-read, which is against Data Lake design principals

* + 1. Data Lineage Metadata

This table defines the metadata required to implement data lineage for each column.

The purpose is to ensure that for any column it will be possible to determine from where this information has been copied or derived.

Where the metadata item cannot be auto generated this will require a manual process for the information to be inputted.

For data lineage the majority can be automatically generated with the only manual information being a description of the process.

Based on the column type there will be a decision as to whether this can be reconciled using a hash total. If not, then an appropriate script will need to be written and stored on the Data Lake. There it can be referenced in the table below and used to perform the required reconciliation steps.

It is recommended that the reconciliation scripts be stored in the Azure DevOps repo so that it can be versioned appropriately.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Metadata Attribute** | **Example Value** | **Can be auto-generaeted** | **Raw** | **Cleansed** | **Transformed** | **Curated** | **Snapshots** | **Data Access** |
| ValidFromRelease | 12 | Y (on first load) | N | Y | Y | Y | Y | Y |
| ValidToRelease | 14 | N | N | Y | Y | Y | Y | Y |
| DestinationUniqueID (layer+source+entity+column) | CU-Eclipse-WrittenPremium-GrossNetWrittenPremium | Y | N | Y | Y | Y | Y | Y |
| SourceColumnUniqueID | TR-Eclipse-WrittenPremium-GrossNetWrittenPremium | Y | N | Y | Y | Y | Y | Y |
| Transformation | CopyValues | N | N | Y | Y | Y | Y | Y |
| Description | Copies values from transformed to curated | N | N | Y | Y | Y | Y | Y |
| ReconcileHashTotal\* | Yes/No | N | N | Y | Y | Y | Y | Y |
| ReconciliationScriptLocation | [/DataLake/scripts/reconcile\_script](file://lond2/scripta) | N | Y\*\* | Y | Y | Y | Y | Y |

\*the lineage is not captured from landing/direct from source to Raw layer

\*\*where possible a "source to Raw" reconciliation file will be provided

* + 1. Release Catalogue

The table below shows the catalogue of Data Lake releases.

|  |  |  |  |
| --- | --- | --- | --- |
| **DateTime** | **Environment** | **ReleaseNumber** | **DevOps Build Number** |
| 12/12/2018 12:52 | Production | 12 | 1234 |
| 14/12/2018 17:32 | UAT | 14 | 2345 |
| 03/01/2018 10:12 | Dev | 17 | 2348 |

Whenever a change is made to the structure of the data sources or the outputted data layers in Curated or Data Access are updated then a release will be made, and the corresponding release number increased.

This will be tied to build pipeline ID in Azure DevOps for full accountability of the changes made.

There will be incremental release numbers shared by all environments. This negates the need to use the environment as an extra key in the metadata and lineage tables above, as the release number will be unique regardless of whether it was on dev, UAT or production.

This will also support versioning and the validity of data in the Snapshots layer.

## Data Modelling

* + 1. Diagrams

For Eclipse, Oceanwide and Exposure the data model diagrams will be developed during the Sprint planning phase and implemented in the Data lake during the POC Sprints.

* + 1. Dimensions & Measure Groups (Facts)

In the Raw layer there will be one file per dimension/fact/measure group, per load, each with a valid from date in the name of the file.

**Data Lake Path example:**

/Raw/Eclipse/Official/Policy/2018/11/27/12/

**Example file name:**

Policy\_yyyyMMdd\_HHmm.csv

**Example of policy dimension content changing over time:**

Created: /Raw/Eclipse/Official/Policy/2018/11/27/12/Policy\_20181127\_1250.csv

**PolicyID PolicyRef PolicyStatus InceptionDate**

123456 p123456 Quote 20190101

123457 p123457 Quote 20190101

Updated: /Raw/Eclipse/Official/Policy/2018/11/27/**13**/Policy\_20181127\_**13**50.csv

**PolicyID PolicyRef PolicyStatus InceptionDate**

123456 p123456 Written 20190101

Deleted: /Raw/Eclipse/Official/Policy/2018/11/27/**13/**Policy\_20181127\_**13**50\_**deleted**.csv

**PolicyID**

123457

Policy ID is business key

In the Curated layer we will have one file per dim/fact/group per load with a valid from date e.g. in the name of the file.

**Data Lake Path example:**

/Curated/Eclipse/Official/Policy/2018/11/27/12/

**Example file name:**

Policy\_yyyyMMddHHmm.csv

**Example of policy dimension content changing over time:**

Created: /Curated/Eclipse/Official/Policy/2018/11/27/12/Policy\_20181127\_1000.csv

**DL\_PolicyID PolicyID PolicyRef PolicyStatus InceptionDate IsDeleted**

1 123456 p123456 Quote 20190101 False

2 123457 p123457 Quote 20190101 False

Updated: /Curated/Eclipse/Official/Policy/2018/11/27/**13**/Policy\_20181127\_**1355**.csv

**DL\_PolicyID PolicyID PolicyRef PolicyStatus InceptionDate IsDeleted**

3 123456 p123456 Written 20190101 False

2 123457 p123457 Quote 20190101 True

The deletion of a record is also handled in the Curated layer by setting the IsDeleted flag.

DL\_PolicyID – Data Lake/Data Library policy ID, acts as a surrogate key.

Implicitly, every dimension is Type 2 at the Curated layer.

* + 1. Surrogate Keys

Naming convention - DL<EntityName><ID> (camel case for EntityName)

* + 1. Business Keys

One column - every table from curated (& snapshot) onwards will have this one columm called BusinessKey

Naming convention - concatenation of all business key values delimited by ";" in a defined order (TBD).

* 1. Snapshot Layer
     1. Determining correct values

For a given point time, for each business key take a combination of the latest available file date, within this taking the latest value of the BusinessKey ID.

* + 1. Populating the Snapshot layer

**Background:**

Any reload of an existing time slice for any source/entity should trigger a rebuild of all the dependent downstream (or parallel e.g. curated star schema dependent entities) layers including snapshots

This is an exceptional case.

**Build:**

For every business key we get the maximum surrogate key ID where the time <= snapshot time.

Regardless of whether the dimension or fact or time slice in snapshot layer has a complete set of facts and dimensions.

The Data Lake may need to support downstream tools e.g. cubes/AAS so will need to build a mechanism to generate valid type-2 dimension tables

**Naming convention:**

/snapshot/MyMI/Internal/Policy/2019/02/03/18/Policy\_20190203\_1800.csv

6 hourly folders:

00 06 12 18

Contents:

For example, the 06 contains from 00:01 to 06:00.

## Data Access

Data Access will be as per the existing methods:

* Azure Analysis Services
* PowerBI

As mentioned before, the use of Power BI Dataflows will be explored in order to allow the querying of the data directly from the Lake

Data access for the POC will be to the predefined 5 sources.

**Data Science – Analysis**

If the Data Science team need a new semi-permanent or permanent data source, then a formal request will be raised for that source to be onboarded into the Data Lake.

If a temporary new source is needed e.g. weather data, then it is up to the Data Science team to create and store this data feed locally to their environment e.g. on their VM.

Ultimately the team managing the Data Lake will be the overall guardians of the data sources provided by the Data Library and surfaced through the Data Access Layer.

# Cost model

*As of this document release, the detailed approach to the Azure cost model is currently being circulated and once approved the detailed output of that analysis will be documented here.*

In general there are some standing costs for Azure services e.g. data storage per GB and then there are also transactional costs e.g. data movement costs for ingress and egress.

For the Azure run costs Brit will:

* Identify how the Azure service are broken down
* Use Azure resource tags to identify cost centres and owners
* Use the Cost Centre to divide costs per source/project (for cross billing)
* Where appropriate deploy and instance of a service per project e.g. one Data Factory instance dedicated to ingest and onboard one source
* Use the previous real costs from the work done using ADLS Gen1 and ADF with Exposure in order to better understand the guidelines in the context of Brit project delivery
* The final cost model will be presented after the end of four Sprints in March 2019

Monitoring and reporting of costs:

* Use the built-in cost monitoring tools (was Cloudyn).
* Use the PowerBI reports for Azure costs
* Exporting the raw costs data to the Data Lake to ensure all historical information is captured and available to report against
* Set up alerts to indicate when Azure spend goes above a predefined threshold

Cost optimisation:

* Utilise built in features of ADLS Gen2 to move data from Hot -> Cold -> Archive tiers as appropriate
* Utilise built in features of Azure storage services to purge data automatically e.g. Landing zone in Blob storage to delete after 30 days
* Turn off workloads automatically after period of inactivity e.g. Databricks clusters
* Consider use of soft deletes – potentially turning this off will reduce costs, but risks could outweigh benefits for data integrity

Resource Tags:

All resources deployed into Azure must be tagged with the following tags:

|  |  |
| --- | --- |
| **Tag Name** | **Example Value** |
| contact\_details | graham.binner@britinsurance.com |
| location | UK South |
| app | datalib |
| contact | Graham Binner |
| description | Brit Data Library (data lake) platform |
| environment | dev |
| source | terraform |
| project | datalibrary |
| costcentre | H15 |

With these tags for every resource it will be possible to assign costs appropriately.

# Access and security

## Data Security

The following high-level principals will be observed:

* Principal of Least Permission (POLP)
* User access secured using Azure Active Directory
* Existing defined Groups (MyMI Users group, MyMI Power Users group) within Brit
* In Finance the levels are more granular.

The Brit Data Classification levels are as follows:

* Public
* Internal (to Brit, user has been added to AD)
* Confidential (user must be member of relevant group)
* Secret (named individuals)

These are defined in the “Data Classification Policy” document on the IT Hub.

**Data Handling**

The security levels and handling will be implemented as per the Data Classification Policy.

For example, Crypto handling – all data is encrypted at rest (and in transit) using the standard Azure Services - this is currently “good enough”. Security officer at Brit has said that when the regulations catch up this will need to be updated but this approach will be compliant for the duration of the POC.

The relevant Project Manager at Brit is responsible for classifying any new data coming into the Data Lake on a per project/source basis.

**Use of this classification in Data Lake store**

By folder.

Landing area - data retention, 30 days (for housekeeping now, rather than GDPR)

**GDPR**

Personal information as captured under fair use will be stored in the Lake e.g. Claims data:

On the claim form there is likely to be a declaration on fair use\* e.g. Brit has the right to use personal data in the execution of the claim/policy.

*\*This declaration is crucial and will need to be captured at source to ensure relevant security and rights are applied.*

If Brit then wanted to do some analysis on this (e.g. “how many people with blue eyes have heart attacks?”) then Brit don’t have the right to do that.

If the data was obfuscated then this application wouldn’t be in breach, or a further request for extra use of the user’s data would be required.

**Data backups**

Where possible data duplication should be minimised.

In principal for Brit, all data is retained forever.

Backups are for Disaster Recovery, which go back 7 days for example.

If you need to delete data, then delete it from the live system, 7 days later data in backups will be deleted as well.

**Archives**

These differ from backups as they contain real data. One should be able to do eDiscovery on archives (see below). How far back should one be able to go back? 30 days is a safe limit.

As an example, and archive of a Legacy systems: a database has grown too large, someone takes an ***archive*** copy, this contains data that you cannot find in your live system.

This is what needs to be made available for eDiscovery (and/or GDPR compliance).

Segment this in a logical way to aid the discovery process.

Archives vs Backups. Where possible, keep backups not Archives as it simplifies the identification of data stored.

**Geo replication and location**

Keeping data in Europe is the safest approach. Data can be stored in the US as well under Privacy Shield agreement, if there are good reasons for providing data closer to the location of consumption.

The most important thing is to know where the data is and where it has come from.

## Network Security

There are various Azure service types (PaaS, Data Factory Integration Runtimes, etc.) that need to communicate with each other and also with the on-premise services too.

The matrix grid below shows those services that make outbound connections and the service they connect with.

From this we are able to determine the most appropriate way to secure that connection.

By default, where a Managed Service Identity exists for an Azure service then this should be used. This negates the need for usernames and passwords to be stored and managed.

A Service Principal and Azure Active Directory (AAD) username and password are the next preferred option.

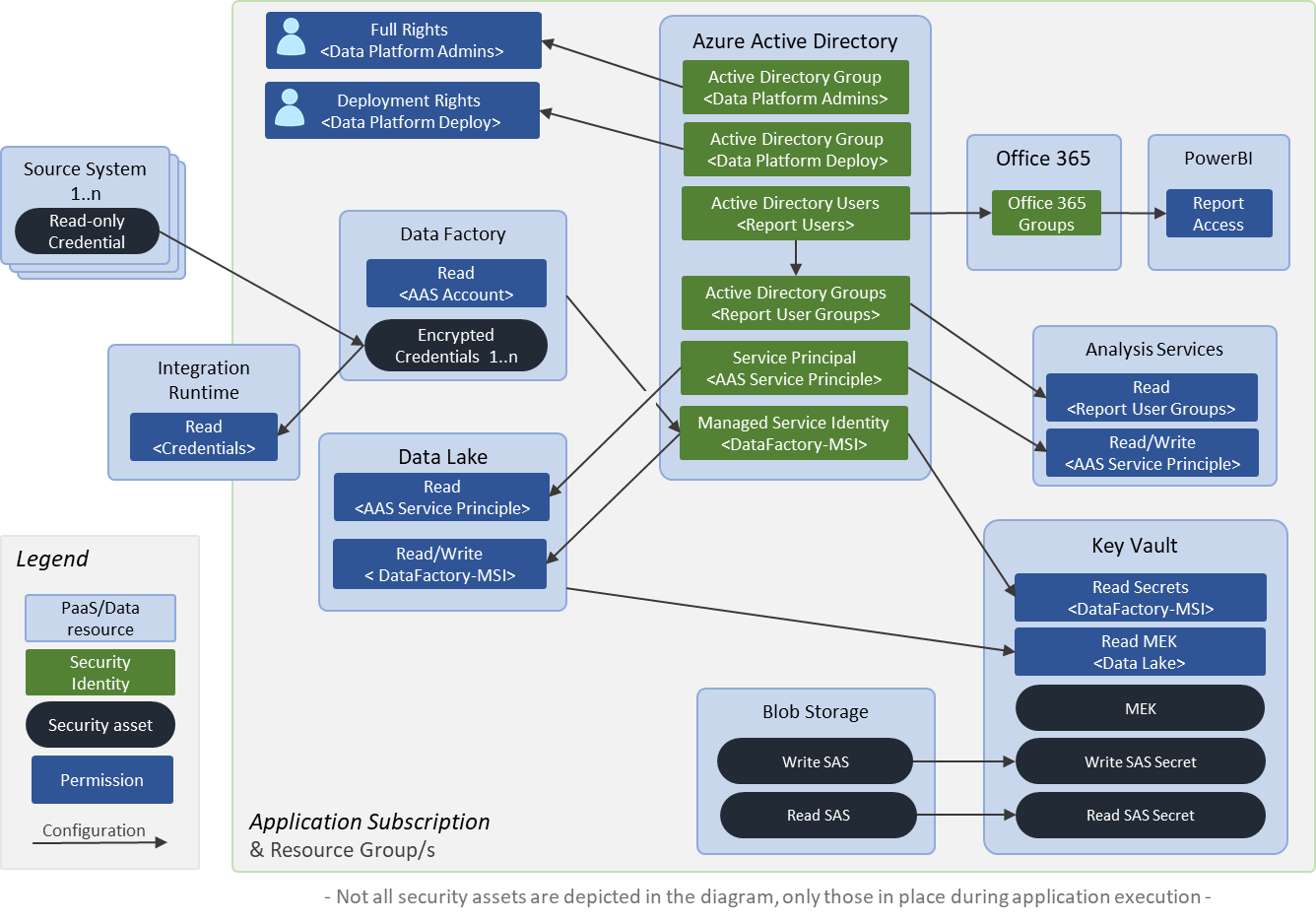
Some services are not secured by an identity - these are usually protected by a firewall or a whitelist of IP addresses that are permitted to make connections.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Azure Analysis Services** | **Azure Data Factory** | **Azure Data Factory - Public IR** | **Azure Data Factory - Self hosted IR** | **Azure SQL DB** | **Cosmos DB** | **Data Bricks (Cluster)** | **Data Lake Store v2** | **Data Science VM** | **Event Hubs** | **Functions** | **On Premise SQL DB** | **On Premise SSIS** | **On Premise Workstations** | **Power BI** | **Public API - Live (Push)** | **Public API - Pull** | **Stream Analytics** | **Vendor Azure SQL DB** | **Virtual Machine** |
| **Azure Analysis Services** |  |  |  |  | X |  |  | X |  |  |  |  |  |  |  |  |  |  |  |  |
| **Azure Data Factory v2** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Azure Data Factory - Public IR** |  |  |  |  | X |  | X | X |  |  |  |  |  |  |  |  |  |  |  |  |
| **Azure Data Factory - Self hosted IR** |  |  |  |  |  |  |  |  |  |  |  | X |  |  |  |  |  |  | X |  |
| **Azure SQL DB** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Cosmos DB** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Data Bricks (Cluster)** |  |  |  |  |  | X |  | X |  |  |  |  |  |  |  |  |  |  |  |  |
| **Data Lake Store v2** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Data Science VM** |  |  |  |  |  |  |  | X |  |  |  |  |  |  |  |  |  |  |  |  |
| **Event Hubs** |  |  |  |  |  |  |  | X |  |  |  |  |  |  |  |  |  |  |  |  |
| **Functions** |  |  |  |  |  |  |  |  |  | X |  | X |  |  |  | X | X |  |  |  |
| **On Premise SQL DB** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **On Premise SSIS** |  |  |  |  | X |  |  | X |  |  |  |  |  |  |  |  |  |  |  |  |
| **On Premise Workstations** | X | X |  |  | X | X | X | X |  | X | X |  |  |  |  |  |  | X |  | X |
| **Power BI** | X |  |  |  | X |  |  | X |  |  |  |  |  |  |  |  |  |  |  |  |
| **Public API - Live (Push)** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Public API - Pull** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Stream Analytics** |  |  |  |  |  |  |  | X |  | X |  |  |  |  | X |  |  |  |  |  |
| **Vendor Azure SQL DB** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Virtual Machine** |  | X |  |  |  |  |  |  |  |  |  | X |  |  |  |  |  |  | X |  |

This table then explains in more detail how these connections are secured at a service level.

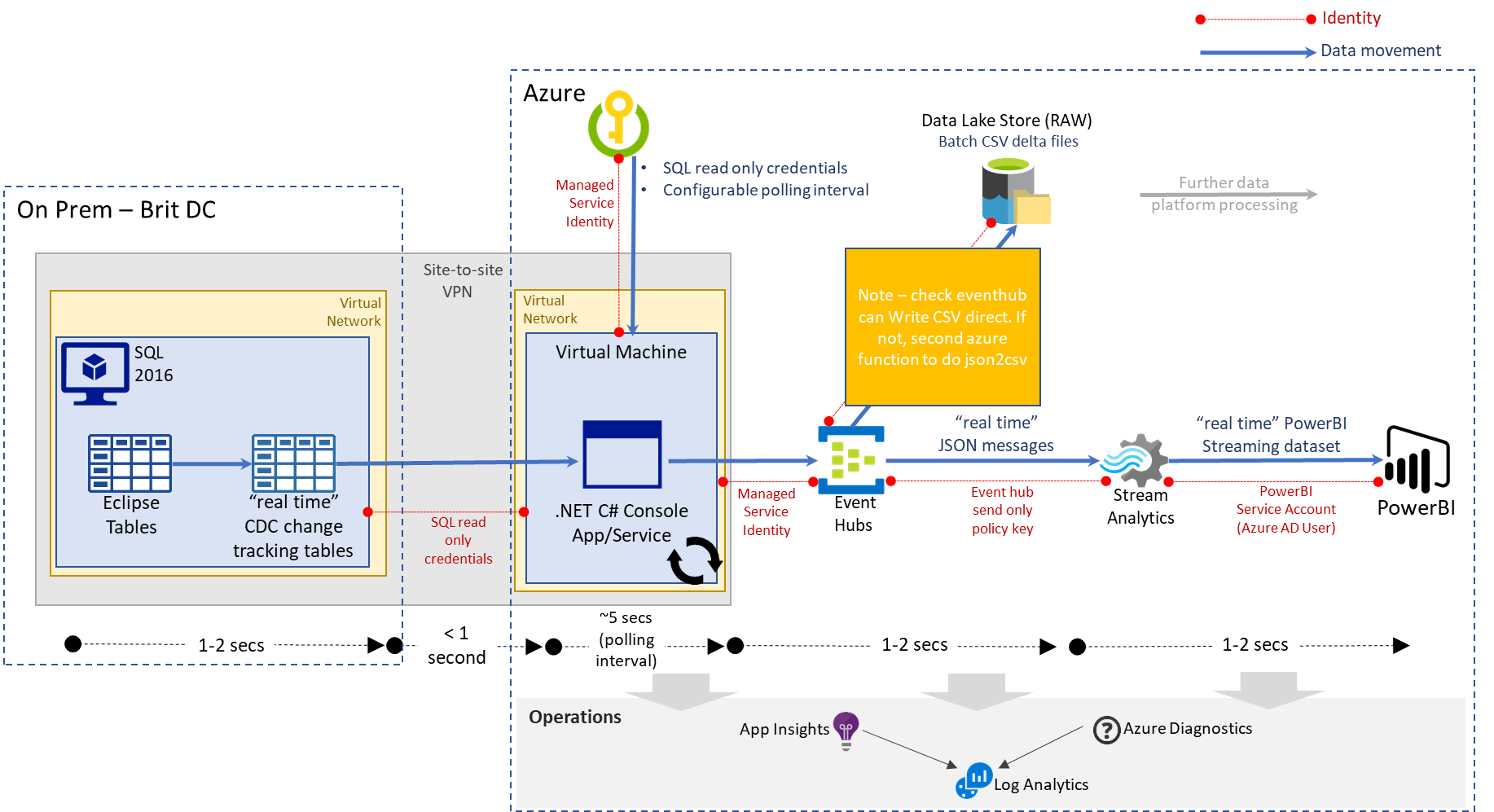
|  |  |  |  |
| --- | --- | --- | --- |
| **Resource** | **Security** | **Inbound Connection** | **Outbound Connections** |
| Fixed Access Token / Key - Azure Key Valut | Stored in the Azure Key Vault | Functions - MSI Event Hub -  Databricks Cluster Stream Analytics |  |
| Azure Analysis Services | 1. Hosted on Azure Public Domain 2. Secured by Firewall - requires client IP(s) to be whitelisted 3. Access to the models secured by Azure Active Directory | On Premise Workstations (2,3) Power BI - via Gateway (2,3) | Azure SQL DB - Azure AD Service Principle Azure Datalake Store - Azure AD Service Principle |
| Azure Data Factory | Hosted on Azure Public Domain Secured by Azure AD Store metadata (Credentials) to connect to linked services via Integration Runtimes |  |  |
| Azure Data Factory Public IR |  | Azure Data Factory V2 (Azure Managed) | Azure SQL DB - MSI Azure Datalake Store v2 - MSI Azure Databricks Cluster - MSI |
| Azure Data Factory Self Hosted IR | Added to Express Route VPN Azure AD security for access control | Azure Data Factory V2 (Azure Management - Reg Key) | On Premise Workstations/Servers - VPN and Express Route Vendor hosted Azure SQL DB - Whitelisting static IP |
| Azure SQL DB | Hosted on Azure public domain Secured by Firewall Database Access secured by Azure AD On Premise firewall requires configration to allow connection to Azure SQL server on port 1433. (Do we enable it enterprise wide or on per request basis?) Zscaler proxy IP ( range ?) - whitelisted in Azure SQL server Firewall Enable all Azure Services?? | Azure Analysis Services - Azure AD User/ Service Principle Azure Data Factory Public IR - MSI On Premise SSIS / Workstations - Azure AD User Power BI - via Gateway |  |
| Cosmos DB | Firewall Enable all Azure Services via Resource Token Broker (web service) | Azure Databricks - Resource Token Broker (Custom App) - TBD\* \* Securing access to the custom app needs further investigation On Premise Workstations - Using Resource Token (key) |  |
| Data Bricks (Cluster) | Secured by Data Bricks V Net Access is controlled by Azure Active Directory (Restricted Access) | Azure Data Factory Public Runtime - MSI On Premise Workstations - Azure Active Directory / Azure | Cosmos DB - Resource Token Broker (Custom App) - TBD\* Data Lake Store - Azure Datalake Store v2 - Azure Active Directory / Service Princple |
| Data Lake Store v2 | Firewall Enable all Azure Services secured by Azure AD Authetication Shared Access Token | Azure Analysis Services - Azure AD User/ Service Principle Azure Data Factory Public IR - MSI On Premise SSIS / Workstations - Azure AD User Power BI - via Gateway Databricks Cluster - Azure AD Service Principle Data Science VM - Azure AD User Event Hub - SAS Token Stored in Key Vault Stream Analytics - TBD |  |
| Event Hub | 1. Firewall - Whitelist Client IP(s) 2. Shared Access Token | Stream Analytics - (2) Functions - (2) | Data Lake Store v2 - SAS Token of the Blob Storage Key Vault |
| Functions | Hosted on App Service Plan On Premise Access Fucntions to be included in the Express Route VPN | On Premise Workstations | Event Hub - Secured Access Token (Key Vault) On Premise SQL DB - Via Express Route VPN Public API (Pull & Push) - Internet / Tokens stored in Key Valut Key Vault - MSI |
| Stream Analytics |  | On Premise Workstations | Data Lake Store v2 - SAS Token of the Blob Storage Event Hub Power BI |
| Virtual Machine | Secured by Firewall Connected to Express Route VPN Secured by Azure AD resticted access | On Premise Workstations | On Premise SQL DB - Express Route VPN Vendor Azure SQL DB - Outbound static IP to be whitelisted on the Vendor's side |

Below shows the use of user credentials, service principals and managed service identities within in Azure in order to provide secure access from service to service within the infrastructure.

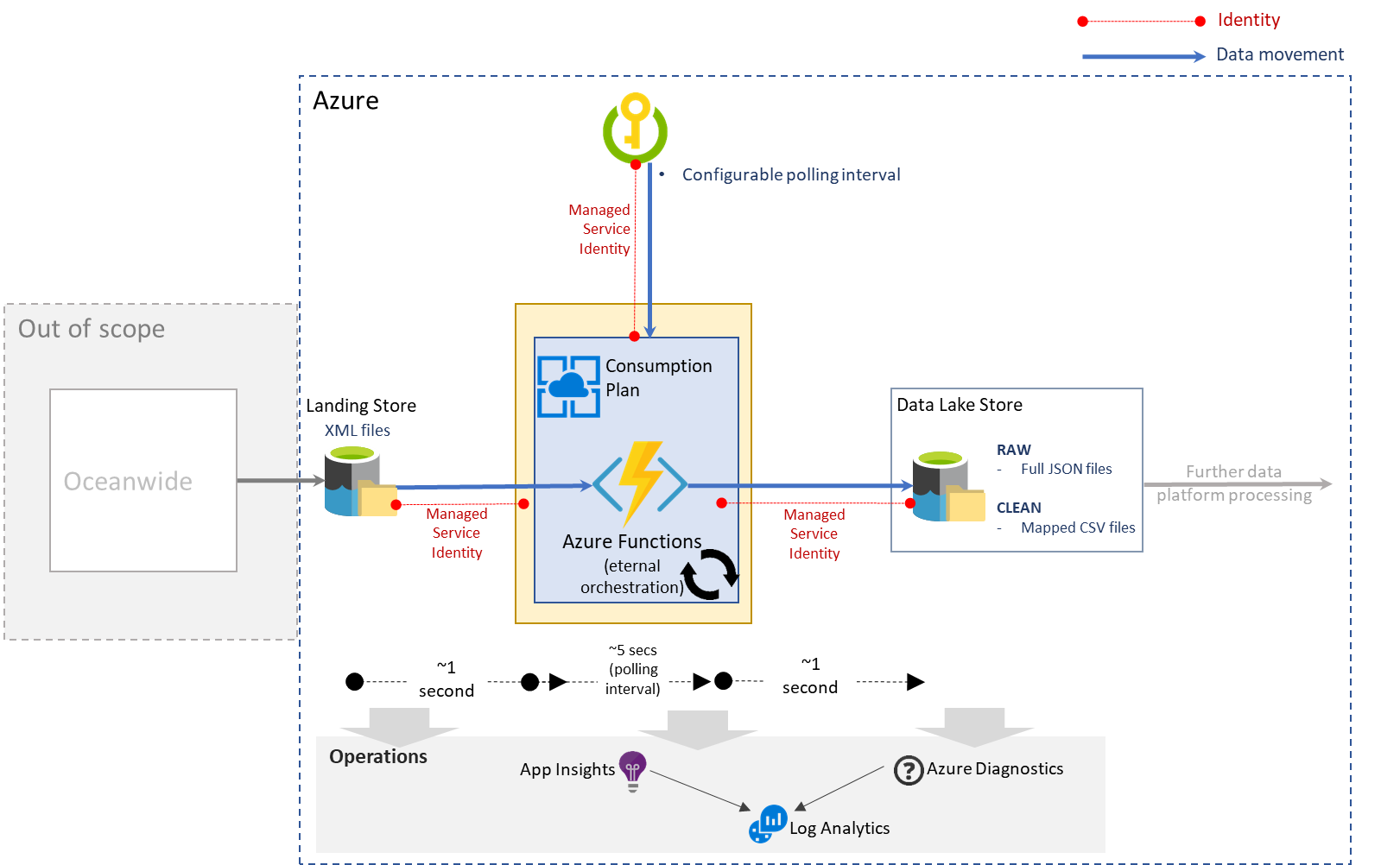


Where possible the use of Managed Service Identities (MSIs) will be used.

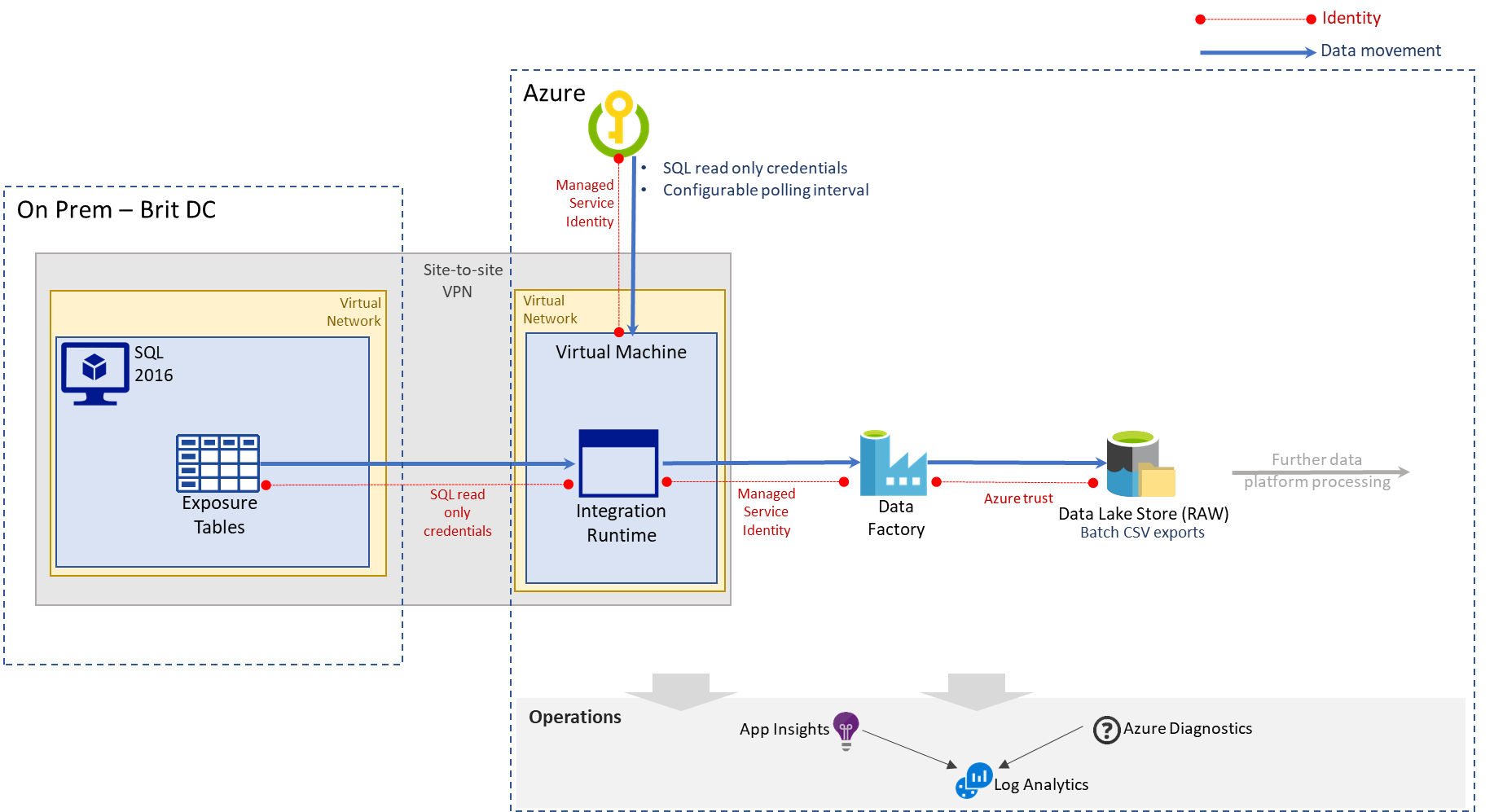
The network security flow for Eclipse:



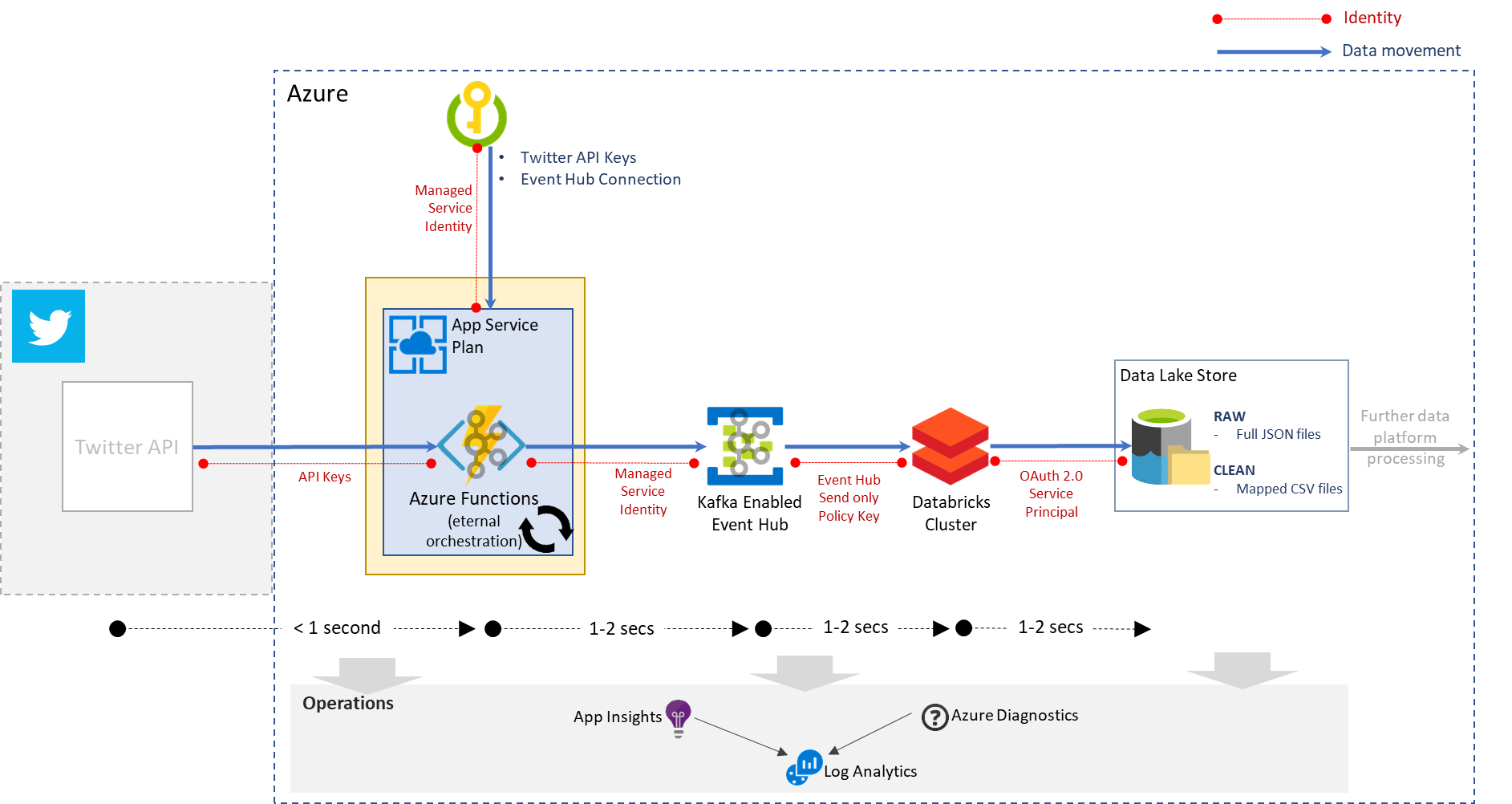
The network security flow for Oceanwide:



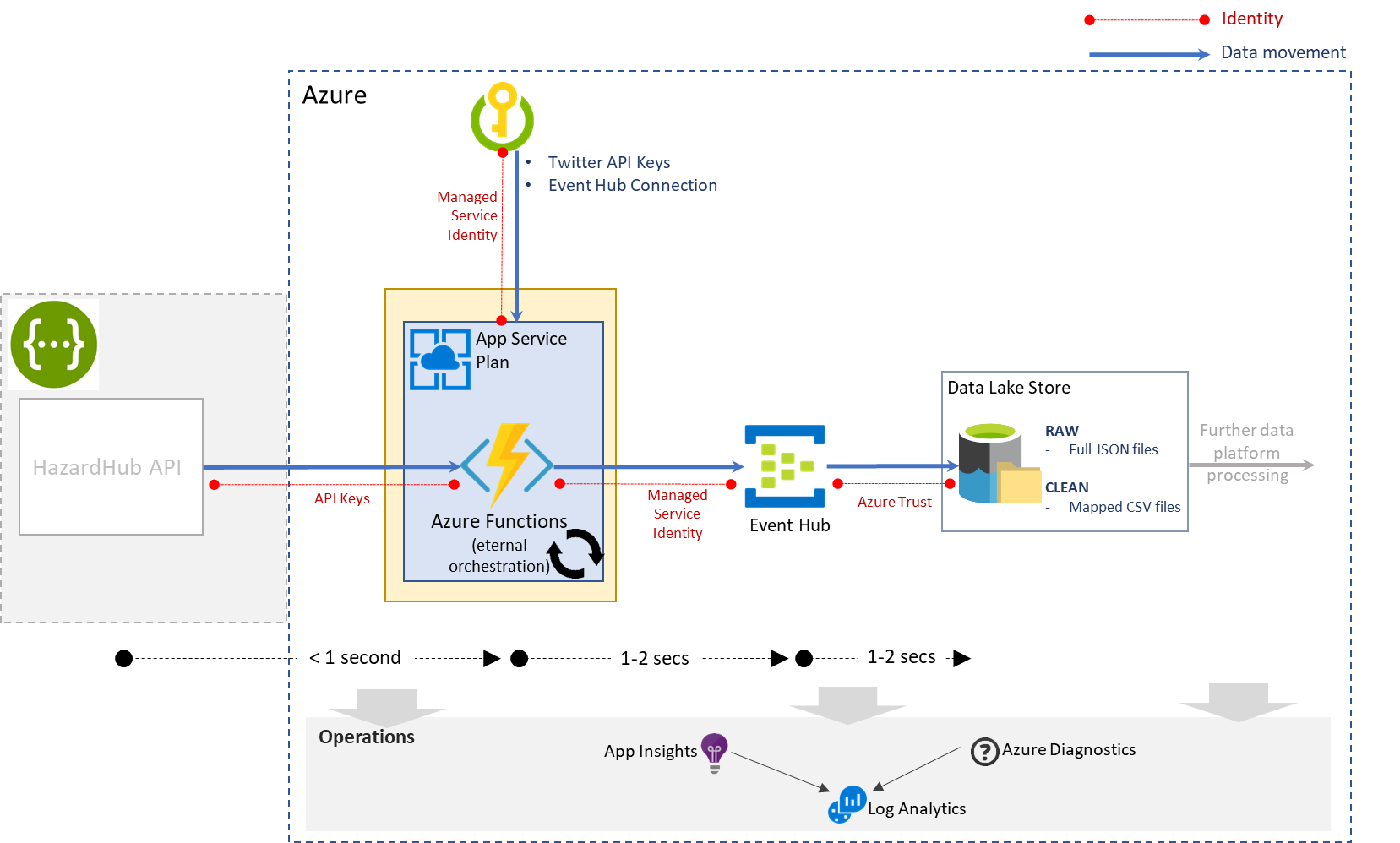
The network security flow for Exposure:



The network security flow for Twitter:

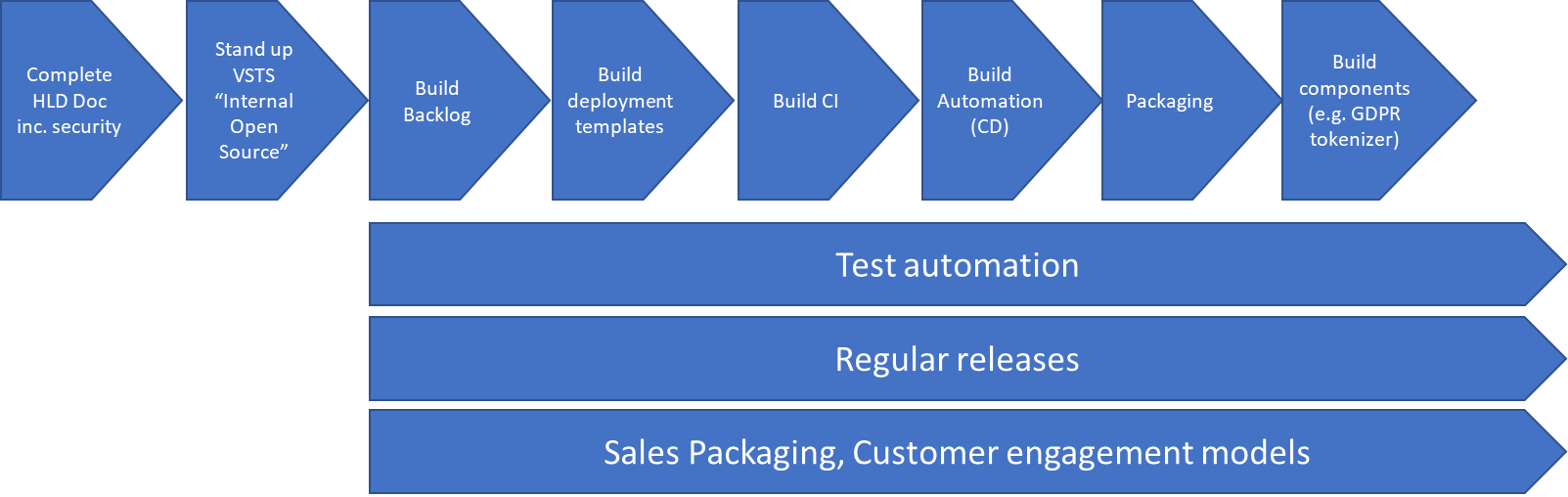


The network security flow for HazardHub:



# Operations

The diagram below shows how the Data Library Data Lake will be developed and deployed using a standard DevOps methodology.



## Azure DevOps

This will be used in the scope of the POC to store code, build and deploy the platform as well as plan the Sprints.

DevOps **Repos**: Azure hosted Git repositories for Terraform templates, PowerShell scripts, Databricks notebooks, Azure Functions etc.

DevOps **Pipelines**: For building the infrastructure and deploying to Azure resource groups using Azure-hosted build agents.

DevOps **Boards**: Used for Sprint planning and execution, a backlog will be generated and the Sprints planned and tracked here.

## Standards

All Azure resources will be tagged as per the table in the Cost Model section.

Terraform is the provisioning tool of choice across Brit.

The Data Library services deployed in development will be done “by hand” i.e. through the portal, the test environment will be an exact mirror of development automated using Terraform.

The main focus will be on Azure services functionality and the data ingestion pipelines, data lake storage and the auto reconciliation of the dimensions and fact measures.

## Monitoring of deployments

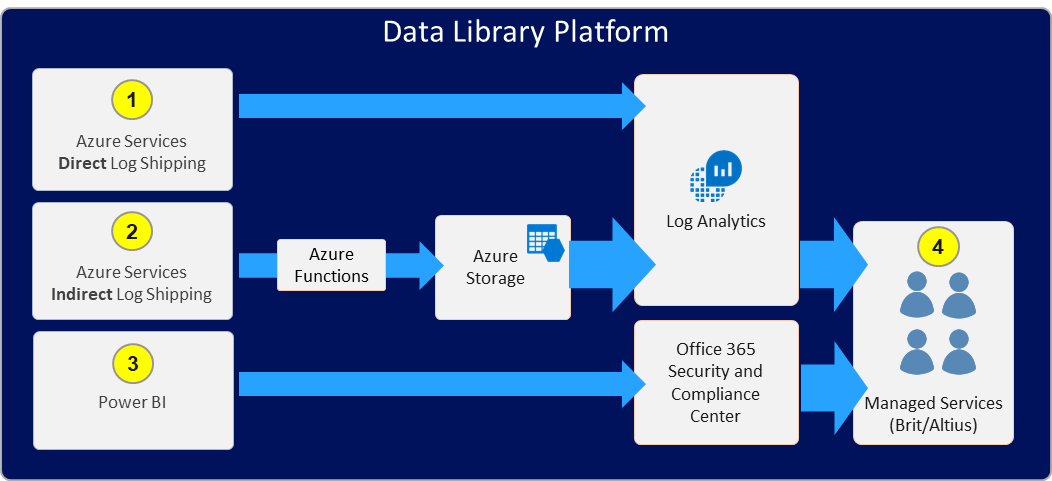
The use of Operations Management Suite (OMS) in Azure is becoming the standard.

OMS now incorporates Log Analytics and where possible Azure services should be configured to log directly to OMS.

Where this is not possible e.g. in an Azure function, log files should be written to blob storage and then imported into OMS.

For other Microsoft services e.g. PowerBI that are not directly part of Azure the relevant management suite for that service should be used.

This is depicted below:



It should be note that Brit currently use a 3rd party on premise SIEM tool, but they are looking to choose a new cloud hosted tool and have this deployed within 6 to 8 months.

When Azure services are deployed the appropriate security events will be raised in Security Centre in Azure and these will be pulled into the wider Brit tool, once it has been selected and rolled out.

## Platform Testing:

* Availability of services (actually monitoring)
* Reconciliations (these will be scripted and automated)

Reconciliations will be stored in CosmosDB using the Cassandra key/value pair API (also accessible from Databricks using Spark)

<https://docs.microsoft.com/en-us/azure/cosmos-db/cassandra-spark-databricks>

## Resilience, Failover and DR:

Each service has its own method of ensuring uptime (resilience) and also approach for surviving a complete outage of the DC (failover/DR).

By default, all services will be deployed to the primary DC – West Europe. Failover will happen to the other DC in this region – North Europe.

This is with the exception of the Virtual Machine that host the Data Factory Integration Runtimes, which communicate with the on-premise services. This will be deployed in UK South due to the availability of the direct connectivity from Azure to these services that has already been enabled i.e. Express Route.

**Azure Data Lake Storage Gen2**

Based on Blob storage service.

In non-production the Locally Redundant Storage (LRS) SKU will be used.

In production the Geo Redundant Storage (GRS) SKU will be used – this will automatically fail over from one data centre in a region to another data centre in the same region.

<https://docs.microsoft.com/en-us/azure/storage/common/storage-redundancy-grs>

Soft deletes should be enabled so that content is never lost.

**Azure Analysis Services**

<https://docs.microsoft.com/en-us/azure/analysis-services/analysis-services-bcdr>

Redundancy can be achieved by creating additional, secondary servers in one or more regions. When creating redundant servers, to assure the data and metadata on those servers is in-sync with the server in a region that has gone offline, you can:

* Deploy models to redundant servers in other regions. This method requires processing data on both your primary server and redundant servers in-parallel, assuring all servers are in-sync.
* Backup databases from your primary server and restore on redundant servers. For example, you can automate nightly backups to Azure storage, and restore to other redundant servers in other regions.

In either case, if the primary server experiences an outage, the connection strings must be changed in reporting clients to connect to the server in a different regional data centre.

This change should be considered a last resort and only if a catastrophic regional data centre outage occurs. It's more likely a data centre outage hosting your primary server would come back online before the connections on all clients could be updated.

To avoid having to change connection strings on reporting clients, a server alias can be created for the primary server. If the primary server goes down, the alias can be changed to point to a redundant server in another region. This change can be automated by coding an endpoint health check on the primary server. If the health check fails, the same endpoint can direct to a redundant server in another region.

**Data Factory**

For resilience, multiple data factories will be used per environment, dependent on the sources and their associated refresh times.

Duplicate pipelines will be created in the secondary data centre but not started until they are required.

**SQL Server**

For the POC these will be backed up in the usual way as they contain only metadata. In production the use of geo-replication should be considered, but there are cost implications which is why simple backups for the PoC phase will be used.

**Cosmos DB**

By default this is geo redundant and can be configured for further redundancy but there are associated data costs.

**App Service Plan**

Give that we are not deploying web applications but Azure Functions these will be deployed as single instances. In a failover/DR scenario the function would be recreated using the deployment templates and the calling services switch over to use this.

**Resource Groups**

One resource group for each non-prod environment.

In production there will be multiple resource groups.

The use of Log analytics, OMS, Security Centre and integration with existing Brit’s choice of monitoring solution/tools will provide active methods of reporting failure.

Testing of service availability:

* CICD pipeline takes care of deployment
* Alerting when service availability changes

Testing of data quality:

* Reconciliations (built into Sprint planning for POC as this is vital to success of the Data Library)

# Scalability

Onboarding of new data sources using repeatable patterns reduces the time it takes to ingest new data.

The use of PaaS allows the Data Lake to respond to varying demands:

* Databricks – increase size and power of cluster, add more clusters
* Data Factory – scale up or down the number of integration runtimes for parallel processing
* Analysis Services – scale up or down the size of SKU (associated cost)
* SQL DB - scale up or down the size of SKU
* CosmosDB - scale up or down the size of SKU
* Event Hub & Stream Analytics - scale up or down the size of SKU
* App Service Plans (for Functions) - scale up or down the size of SKU

# Requirements Matrix

This is now stored as a separate file called “Requirements and scope matrx.xlsx”

in the same folder as this document in Teams.

Altius London

69 Carter Lane

London

EC4V 5EQ

Altius India

A5/903, Kumar Palmgrove

Kondhwa Bk

Pune 411048  
Maharashtra, India

Altius Consulting

River Court

Mill Lane

Godalming

Surrey, GU7 1EZ

Confidential. © Copyright 2016 Altius Ltd