Project Standards and Guidelines

Contents

[1 Purpose 3](#_Toc41294400)

[2 Project Development Initiation 3](#_Toc41294401)

[3 Requirement Gathering 4](#_Toc41294402)

[4 Engineering Guidelines 4](#_Toc41294403)

[4.1 Resource Group (AICP) 4](#_Toc41294404)

[4.2 Resource (AICP) 5](#_Toc41294405)

[4.3 Resource Specific Conventions 5](#_Toc41294406)

[4.3.1 Azure Data Factory 5](#_Toc41294407)

[4.3.2 Azure Data Lake Store 6](#_Toc41294408)

[4.3.3 Databricks 8](#_Toc41294409)

[4.3.4 HDInsight Cluster (AICP) 9](#_Toc41294410)

[4.4 Auditing Strategy (AICP) 9](#_Toc41294411)

[5 Visualization Guidelines 9](#_Toc41294412)

[6 Project Specific Documentation 11](#_Toc41294413)

[7 Best Practices 11](#_Toc41294414)

[7.1 Spark 11](#_Toc41294415)

[7.2 Scala 12](#_Toc41294416)

[7.3 Python 12](#_Toc41294417)

# Purpose

The purpose of this document is to define the development guidelines and standards which can be followed across all the projects. This should help team for effective delivery of the project.

# Project Development Initiation

Follow below guidelines for CDL and AICP platform project initiation:

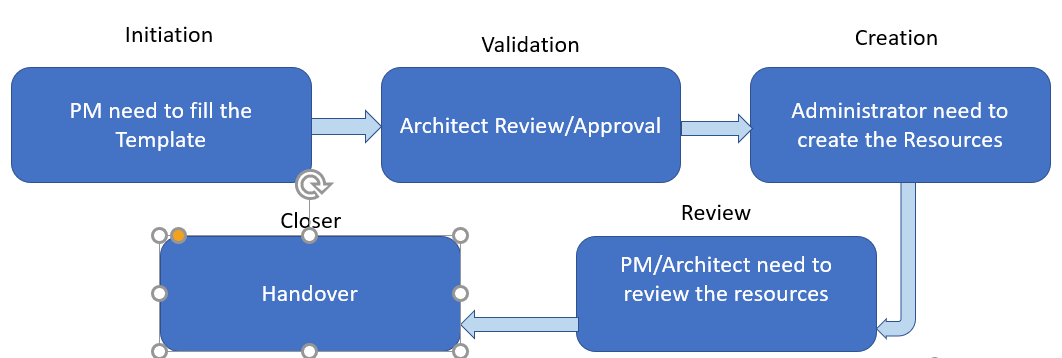
**CDL :**

Refer attached document to know the CDL workflow.



**AICP :**

Once the project is approved on AICP , below steps needs to be initiated by Project Manager to enable the project development on Azure Platform :



**Azure Template**:

|  |  |
| --- | --- |
| **Tag Name** | **Description** |
| Project Name |  |
| Data Architect Approver |  |
| Project Owner |  |
| Business Unit |  |
| Business Owner |  |
| Environment |  |
| Department |  |
| Azure Locations |  |
| Policy Id |  |
| Project Administrator |  |
| Development Team |  |
| Support Team |  |
| Other Specific Requirement |  |

# 

# Requirement Gathering

Document attached below has set of queries which can be raised against source team for data clarity.



# Engineering Guidelines

## Resource Group (AICP)

Resource group will be created and maintained by the Azure admin group.

The naming conventions for resource groups are as follows:

AZ-RG-DCAD-{project}-{environment}

Example : AZ-RG-DCAD-AGBBigDataLongevity-DEV

|  |  |
| --- | --- |
| Project | Descriptive name for the project |

|  |  |
| --- | --- |
| Environment | Description |
| DEV | The development environment where permissions are relaxed for some developers creating applications or business intelligence applications. |
| UAT | The UAT environment where permissions are set to read only and all deployments are controlled. |
| PROD | The production environment. Developers may have some read only access to this environment and all deployments are controlled. |

## Resource (AICP)

Resources will be maintained by the project team. If you need a new resource apart from ADF/Databricks, please request Azure Admin along with proper justification for its creation.

The naming conventions for resource are as follows:

aicp-{name}-{project}-{region}-{environment}

|  |  |
| --- | --- |
| Name | Descriptive name for the resource being created |

|  |  |
| --- | --- |
| Project | Descriptive name for the project |

|  |  |
| --- | --- |
| Environment | Description |
| Dev | The development environment where permissions are relaxed for some developers creating applications or business intelligence applications. |
| UAT | The UAT environment where permissions are set to read only and all deployments are controlled . |
| Prod | The production environment. Developers may have some read only access to this environment and all deployments are controlled . |

|  |  |
| --- | --- |
| Region | Region for the project |

**Examples:**

* aicp-df-customercentricity-prod -> This resource is a data factory servicing the production environment for project customer centricity.
* aicp-sqldb-customercentricity-prod -> This resource is a sql database servicing the production environment for project customer centricity.

## Resource Specific Conventions

### Azure Data Factory

|  |  |
| --- | --- |
| ADF Object | Schema |
| Pipeline | PL\_{Source DB Name}\_{Table Name}\_{Destination DB Name}\_{Frequency} |
| Input Data Set | DS\_IN\_{Database name}\_{Table Name} |
| Output Data Set | DS\_OUT\_{Database name}\_{Table Name} |
| Linked Service | For Databases – LS\_{Database Type}\_{Database Name}  Example: LS\_SQL\_dbname  For Services – LS\_{Service Type}\_{Service Name}  Example: LS\_ADLS, LS\_ADW\_SQLDWBI |

Database Name – The name of the database where the table is stored. If the storage is Azure Data Lake Storage, then the name will be ADLS-{Zone Name}.

Dashes are not allowed in ADF V2.

Naming convention for pipelines :

* Pipelines that include multiple tables or steps should identify the subject area or process instead of the {Table\_Name}.

Example: PL\_AGILE\_SPEC\_CREDIT\_RADIANT is a pipeline that copies Spec Credit tables from Agile to Radiant.

Example: PL\_IDM\_AttributeEntityTables\_ADLS\_CURATED\_DAILY copies IDM Attribute and Entity tables to the ADLS Curated zone traversing through Raw and Staging.

* Pipeline Activities should be named appropriately.

Examples: copy\_dimMasterItem, copy\_AuthorizedExpenseModel, copy\_mtl\_transactions

DO NOT USE: Activity-Copy-Input-Data-Set, Activity-0-USERID\_NAMESRELATION

* Consolidate Pipelines

For source to ADLS (Raw and Staging), use one pipeline per data source per refresh period.

For curating data, multiple pipelines might be required per subject area or project.

Frequency – The frequency of execution for the pipeline.

* INITIAL – A one-time execution of the pipeline
* MINUTELY – The pipeline runs every x minutes
* HOURLY – The pipeline runs every x hours
* DAILY – The pipeline runs once daily
* WEEKLY – The pipeline runs once weekly
* MONTHLY – The pipeline runs once monthly
* INTRADAY – The pipeline runs every x hours in a time window

**Data Factory accessing ADLS – To use managed identity in linked service while making the connection, provide the factory name to Azure admin so that the role can be added properly on ADLS-Gen2 for accessing the data.**

### Azure Data Lake Store

The following naming conventions will be used for the Azure Data Lake Store.

There will be 3 major containers :

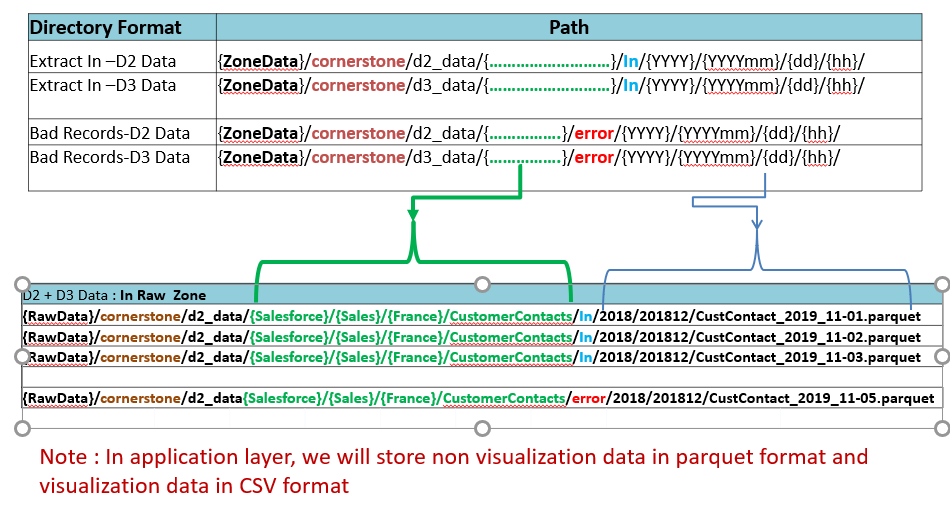
* Data container name : **data**
* Log container name : **log**
* Script container name : **script**

Under **data container**, data lake store will consist of 3 zones: Raw, Staging, Curated

#### Zone Definitions

* Raw Zone
  + Stores the data from the source systems in raw format
  + Mostly written by ADF copying tables from the source systems into flat files in parquet format.
  + Not suitable for analytics/reporting
* Staging Zone
  + Temporary data zone for additional reporting/transformations
  + Holds data that may be used for multiple processes
  + Is not required to be used
* Curated Zone
  + Reporting consumption zone for Power BI, SQL DW
* Reference Zone
* Stores all reference data required for data processing
* POC Zone
* Only available for dev environment to performing any poc

#### Directory Layout



Azure admin will create project specific folder and will give appropriate access under all 3 containers (data/log/script) .

Project team will be responsible to create and manage folders underneath project folder .

Example : {RawData}/cornerstone/d2\_data/{Salesforce} will be created by Azure Admin.

### Databricks

* Notebook names should be UpperCamelCase
* When a notebook is being created for each entity, the name should be the folder name and the entity name
* If a notebook is not entity specific, its name should be descriptive of its process, including a verb and the zones involved
* Column and Table names are UpperCamelCase
* Python used in notebooks should follow [PEP-8 style guideline](https://www.python.org/dev/peps/pep-0008/)s for formatting and names
* Variables and functions should be lower case with words separated by underscores

Example: python\_variable = [1, 2, 3]

* Classes should be lowerCamelCase
* SQL commands should be UPPER CASE
* When using SQL, be generous with white space and indentation to improve readability
* Logging mechanism should be implemented properly for catch the Exception.
* Documentation should be in markdown
  + Include a header and overview at the beginning explaining the purpose of the notebook
  + Include documentation before each command­­­­
* Develop shared code in Shared folder
* Use built-in functions and libraries to transform the data over user defined functions
  + Refer to [Sparks](https://spark.apache.org/docs/latest/api/python/index.html) and [Databricks](https://docs.databricks.com/) documentation
  + Default libraries are approved. See list of [default libraries here](https://docs.databricks.com/release-notes/runtime/5.2.html#installed-python-libraries) (check runtime version)
  + All new libraries must be approved, and documentation updated to state approved libraries
* Refer below attached guideline for ADLS connection setup using pass through method.



### HDInsight Cluster (AICP)

Cluster can be utilized to run any Spark or Pyspark code which can be kept on ADLS-Gen2 script container (Cluster is already been mounted to ADLS-Gen2 script container).

To use cluster, RSA token needs to be generated and shared with Azure admin(Nishant Soni) to get access.

Follow steps in below document to generate the RSA token :



Please note that common cluster will be used across all projects for development whereas dedicated cluster will be built for production.

## Auditing Strategy (AICP)

Capture all ADF pipeline details as part of audit in below sql server through execution of stored procedure maintained by Azure admin team:

**SQL Server :** SSDB\_AICP

**Schema :** dev/uat/prd

**Tables:-**Audit Table

**Column :**

* DATAFACTORY\_NAME
* PIPELINE\_NAME
* START\_DT
* END\_DT
* DURATION
* STATUS
* FOLDER\_PATH
* FILE\_NAME
* NB\_BYTES\_COPIED
* SOURCE
* CLASSIFICATION(D2/D3)
* ERROR
* REJECTED FILE PATH

# Visualization Guidelines

Refer below guideline document for Power BI report creation :



# Project Specific Documentation

Following sets of documents should be available as part of project deliverables:

* Technical Spec – This document should explain the complete flow of the technical process involved for the delivery of the project.

Template : 

* Unit Test Case/Results – This document should capture various test scenarios and its results to ensure data handling has been done properly.

Template : 

# Best Practices

## Spark

* Use Broadcast joins when joining a huge fact and dimension table for better performance.
* Predicate push down improves spark SQL (Avoiding full table scan by filtering).
* Parquet (Delta) format is recommended as optimal file storage format, as its 7x faster than CSV or JSON
* Execution plan of spark SQL queries can be measured by using Explain () or EXPLAIN function
* When we are joining objects, go for narrow transformations then wide transformations, Because In narrow transformations there won’t be any shuffling, but in wide transformations there will be shuffling.
* Join always smaller table against larger table.
* Partitioning on tables based on columns, ex: - Partition by Time period or Partition by states. Partitions minimizes the data scan on Tables
* Use cache () or persist () wherever needed but flush the least used memory all the time
* Auditing for Mismatch of data, Ex: - Record count, financial balancing using SQL views
* Try to make sure that your code is written in a way that minimizes the amount of shuffling that may occur
* Collect () function can be used to measure the running time of each individual stage of the spark job
* Use ReduceByKey instead of GroupByKey
* Don’t run long queries on SQL source systems, bring the data into the Spark platform and then apply necessary joins and filters

## Scala

Use below guide for Scala best practices :



## Python

Use below guide for Python best practices :

