# APS DR framework- Implementation using ADF pipelines

# OBJECTIVE

The framework presents the Disaster Recovery (DR) solution architecture for Microsoft’s Analytics Platform System (APS). While there are several DR Architectures available for APS; this IP focuses on Dual Load approach implementation. The Dual Load solution focusses on loading and tracking each flat data file into both the Primary & Secondary APS systems with an ETL workflow implementation using Azure Data Factory (ADF)

DUAL LOAD DR FRAMEWORK

Dual Load framework is accomplished by following processes:

1. A process that pulls all the source system data to be processed in a central location. When dual loading two systems, the dual ETL process will not pull the data at the same rate or at exactly the same time. Therefore, having a central process that pulls the data and generates a flat file that both systems use, the potential of one system being different than the other will be lessened. Using flat files allows easy method to ensure both the Primary and Secondary PDW systems stay 100% in sync.
2. A central Monitoring or control DB process is used to keep track of when each flat file gets loaded on each of the APS systems.
3. The dual load process implemented with 3 separate workflows or base pipelines:
   1. **Init Workflow Pipeline** begins the dual load process by retrieving a list of flat files stored in a directory and recording the location and filename in the control db.
   2. **Load Process Workflow Pipeline** reads the central control DB and determines the flat files that need to be loaded into the PDW. Should one of the PDW go offline, no steps will need to be taken. When the PDW comes back online, the workflow will process the files that have not been processed for the given PDW.
   3. **Archive Workflow Pipeline** reviews the control tables to determine what flat files have been processed by both systems. If the file has been processed by both systems, the flat file will be moved to an archive location. If the file has only been processed by one system, the files will remain on the disk until the other system can process the file.

DESIGN CHALLENGES using ADF

DR framework processing should not be limited to pre-defined source data but must be able to process new flat files originating from source system. For this to be accomplished using ADF technology- ADF components including pipelines, activities and datasets need to be created specific to each input data files. But, ADF in current form lacks out of the box support for looping construct or flow variables across activities. Per product team- "Both these features (looping, state passing, etc) are in design phases as part of us expanding the ADF app model. Too early to say yet though re: release dates, etc. "

**Solution:** To manage the requirement to be able to process changing or new files from source dynamically I needed to “unroll” the loop into N pipelines (N being dynamic number based on new files at source). The dynamicity of the solution was accomplished by creating/updating/monitoring Data Factory, Pipelines, Activities, Datasets & Linked Services programmatically using ADF SDK).

SETUP

* Azure Storage account for blob data files and Azure SQL for Control DB can be created directly from Azure Portal
* All DDLs & stored procedures (available in VS project) can be loaded on Control DB by connecting thru SQL Server management Studio. Sample DMLs are also provided for sample data loading.
* Creating Data Factory:

Prerequisites

* Visual Studio 2015
* install Azure .NET SDK (from [MS download site](https://azure.microsoft.com/en-us/downloads/))
* install NuGet packages for Azure Data Factory
  + Click Tools, point to NuGet Package Manager, and click Package Manager Console.
  + In the Nuget Package Manager Console, download the latest ADF Management nuget.

Install-Package Microsoft.Azure.Management.DataFactories  
Install-Package Microsoft.IdentityModel.Clients.ActiveDirectory

* (Optional) Download latest Azure Data Factory plugin for Visual Studio 2015 (Tools -> Extensions and Updates -> Online -> Visual Studio Gallery ->Microsoft Azure Data Factory Tools for Visual Studio)
* For create new Data Factory or tear down existing factory & Linked Services, use AzureDataFactoryFoundry program
* To create all ADF pipelines required for Dual Load Process- Execute Init, Load Process, and Archive Pipeline programs in sequence .
* Further, Scheduling the execution of pipelines based on business requirement can be done by either: (1) modifying pipeline start and end times, or (2) scheduling section of activities, or (3) changing dataset availability section

FINAL SOLUTION

Here is the link to Github Repository for the solution:

<https://github.com/manav0/HybridDR_ADF_Repository>

Above repository has complete Visual Studio project containing

* All the code artifacts for Hybrid DR implementation using Azure and Azure Data Factory .NET SDKs
* All the table DDLs and stored procedures for Control Database.

Configuration:

* The same code can be configured for different Azure subscriptions by modifying the Subscription & Tenant Ids config fields in App.config.
* The connection strings for Control database and Azure Blob storage account can be modified in DualLoadConfig.cs.
* Data Factory component names, sql tables, or updates to queries can also be performed centrally in DualLoadConfig.cs.

REFERENCES

ADF SDK 

<https://msdn.microsoft.com/en-us/library/dn883654.aspx>

Link for getting the ADF SDK as nugget package 

<https://www.nuget.org/packages/Microsoft.Azure.Management.DataFactories/>

Basic Example for ‘create and update’ of ADF components programmatically 

<https://azure.microsoft.com/en-us/documentation/articles/data-factory-create-data-factories-programmatically/>