7/21/2010

|  |
| --- |
| Common Hosted ETL Framework(CHEF) | Balaji Mishra |



|  |  |
| --- | --- |
| MSIT India | CHEF Architecture |

Contents

[1. CHEF Architecture 2](#_Toc269456276)

[2. CHEF Control Structure 2](#_Toc269456277)

[2.1. Windows Event Viewer Entry 2](#_Toc269456278)

[2.2. Check for SSIS 2](#_Toc269456279)

[2.3. SQL Jobs 2](#_Toc269456280)

[2.4. CHEFConfiguration Table 3](#_Toc269456281)

[2.5. CHEFRequestQueue Table 3](#_Toc269456282)

[2.6. CHEFLog Table 4](#_Toc269456283)

[3. XML Metadata for an Application 5](#_Toc269456284)

[4. CHEF Core Engine 6](#_Toc269456285)

[4.1. Key Activities 6](#_Toc269456286)

[5. CHEF Work Bench 6](#_Toc269456287)

[5.1. Key Elements of the UI 6](#_Toc269456288)

[6. Appendix 7](#_Toc269456289)

[6.1. References 7](#_Toc269456290)

[6.2. Contributors & Documentation 7](#_Toc269456291)

# CHEF Architecture

Introduction:

Every Data Loading may consist of one or more **Processes**. And each Process may have one or more **Steps**. The hierarchy of isolation between Process and Steps could go deeper, but for the basic needs the level of isolation that is needed for supporting the **Re-Startability** from the point of failure the Step may be a good boundary. The Step may be as simple as executing an SSIS package, invoking a SQL Job, running a SQL script etc.

Most scenarios all these can be achieved in multiple ways. The need of a Framework is to standardize these into a singular and uniform process so that there is consistency between different data loading processes, which improves the Maintainability, Supportability, Extensibility besides providing basic utilities to avoid Repeatability for cases where it requires to do Data Transfers for 100s of tables or source files. In these kinds of scenarios having a Metadata driven approach to load data between Source and Destination helps alleviate lots of repetitive work and need for frequent changes into the ETL tool or programs.

The intention of this framework is to support the basic minimum need required by TDW and have an uniform and consistence Data Loading experience apart from an Integrated Operations Dashboard

# CHEF Control Structure

## Windows Event Viewer Entry

During CHEF installation based on the Configuration value, create an Application Category in the Windows Event Viewer

## Check for SSIS

It should check/prompt if SSIS Services is not installed in the SQL Server (Custom SSIS packages need to be stored in the SQL Server(for SQL 11 Compatability))

## SQL Jobs

Three SQL Jobs viz., CHEFUtility Job, Control Job and Executor Job would be created during the installation of the Framework.

* 1. The Control Job would be having a infinite loop and watch the status of the Executor Job and the incoming request for Executing any Data Loading request from a CHEFRequestQueue table with a suitable wait time that could be configured during installation or by manually changing the value
     1. Control Job picks the requests from the CHEFRequestQueue table if any and invokes the Executor Job if it is not in running status
     2. Control Job idles if there is a new request but the Executor Job is in runnable status i.e., processing some Data Loading
     3. Control Job would update that status in the CHEFRequestQueue table based on the progress or finish of the Executor Job
  2. The Executor Job would run the executable to create the Dynamic SSIS Package or invoke a SSIS Package which in turn would invoke the executable to create the Dynamic SSIS Package at run time to do the specific or end to end Data Load for one or more source and destination
  3. The Executor Job will have Three Steps each making an entry to the CHEFLog table for the status of the ETL viz., On Start, On Stop/Cancel, On Success/End and On Failure
  4. The On Fail or On Stop/Cancel step would make an additional insert for the unfinished task(s) with Status as Started but not finished to the CHEFLog table. Besides this would raise the error with a Severity of 19 or more so as to log into the Windows Event Viewer if it is configured to do so
  5. The CHEFUtility job may be scheduled to run once a year(depending on the no. of executions it may have to be changed) or invoked on need basis, will be responsible for doing one or more of the following tasks
     1. Archive the Queue & Log data to a respective Archive tables, so that the current tables has less data and can be accessed faster based on suitable configuration value e.g., if the no. of rows exceed 1 million etc
     2. Reset the identity column value of the Log & Queue table if the values reach near its limit

## CHEFMetadata Table

* 1. This table would store the configuration values that typically are supplied during installation and would be used life long
  2. It would be most likely a single row table if there is one XML Metadata File or a row per Processes, besides a row for Global Configuration
  3. Few examples of life time configuration values are ArchiveRowLimit, ResetIdentity flag when the identity column of the Log & Queue tables reaches near 90% of its limit, ApplicationEventViewerName, XML column with all the Metadata for the Application
  4. It stores the XML metadata for each of the process of an data loading process

## CHEFRequestQueue Table

* 1. This table would store the request to execute all the Steps in a Process or all steps from a Specific Step onwards.
  2. This request can come from an UI or manual entry into the table or through a Scheduled Job which puts an entry in to this table
  3. The schema of this table will be as follows (subjected to change later; script to be attached)

QueueID int identity(1,1) not null

,CalendarMonth tinyint NOT NULL, --e.g., 1-Jan, 2-Feb

,CalendarYear smallint NOT NULL, --e.g., 2009, 2008

,ProcessID smallint not null --e.g.,10-OPUS Loading, 20-Excise Loading etc

,StartStepID smallint not null --e.g., 1000-Truncate Opus Tables; 1010-Load Opus Staging, 2010-Truncate Excise Tables etc

,RequestStatus tinyint not null -- 0:Queued, 1:Started, 2:Finished, 3:Stopped, 4:Failed

,ScheduleDate datetime not null --by default getdate() with time in hh only

,LineageID uniqueidentifier NOT NULL, --e.g., 36CFA4E7-7248-49C6-8149-BE780C7ED808

,RequestedBy nvarchar(50) not null default GETDATE()

,RequestedDate datetime not null default SUSER\_SNAME()

NB:

1. The LineageID is created per process execution request, currently this is there in all tables in TDW, by keeping it here, this would not be required in all tables, instead the QueueID would be stored in the Log table to find the Last Load Lineage
2. The ScheduleDate column can also be used for scenarios where the Filter to fetch Delta is a datetime in a day.
3. The combination of CalendarMonth, CalendarYear, ProcessID, RequestStatus & ScheduledDate should be unique
4. Request Queue to have new entry for reload/restart scenario

## CHEFLog Table

* 1. This table would store the Log for all activities both success and failure related to Data Loading using CHEF
  2. There would be TWO inserts for each activity viz., Start and Finish(/Stop/Fail)
  3. The data in this table for any Process would be stored in a inward hierarchical format e.g.,

OPUS Load Started

Step-1: Load Stage Started

Step-1\_1: Truncate dbo.abc Table Started

Step-1\_1: Truncate dbo.abc Table Finished

Step-1\_2: Load dbo.abc Table Started

Step-1\_2: Load dbo.abc Table Finished

Step-1: Load Stage Finished

Step-2: Load Warehouse Started

Step-2\_1: Delete dbo.abc Table Started

Step-2\_1: Delete dbo.abc Table Finished

Step-2\_2: Load dbo.abc Table Started

Step-2\_2: Load dbo.abc Table Finished

Step-2: Load Warehouse Finished

OPUS Load Finished

1. There would be wrapper SPs to insert data into this table
2. There would be wrapper UDFs/SPs/Views to read data from this table
3. The schema of this table (already in use in TDW)

[LogID] [int] IDENTITY(1,1) NOT NULL,

[QueueID] [int] NOT NULL, --FK References CHEFRequestQueue(QueueID)

[ProcessStep] [varchar](255) NOT NULL, --Unique for ProcessID e.g.,Load\_ACCCAT1010,Load\_EXRATE01Currency,Truncate\_ACCCAT1010,Update\_ACCCAT1010

[ProcessDate] [datetime] NOT NULL,        --e.g., 2009-09-14 00:14:78 245 hrs

[StatusID] [tinyint] NOT NULL,        --e.g., 1:Started, 2:Finished, 3:Stopped, 4:Failed

[RowsAffected] bigint NOT NULL, --e.g., 100 --At the Start it is estimated rows and at the Complete/Fail it is Actual rows

[Description] [varchar](1000) NOT NULL,        --e.g., Load data to dbo.AdjustmentTypeHierarchy from FeedStore Started. Estimated Rows(100).

NB:

1. The Description and ProcessStep names will be generated dynamically by the framework during the package creation and supplied to an SP for insert into the Log table
2. If some Transformations are done through Custom SP or SSIS package then these inputs should be supplied as per the above guideline, without which only the outer call to the execution of the Custom SSIS/SP will be logged to the Log table

# XML Metadata for an Application

Typical Key Inputs that's needed for Data Transformations which could be metadata driven

1. Process Name

This would be the Dynamic SSIS package that will be created at runtime

1. Step Name
   1. This would be Container Name and point of restartability or reload starting point in case of failure
   2. This would precede and succeed with a SQL Task to log the start and finish of the Step
2. Transformation Type(s)

That would be supported by the CHEF initially

* 1. SSIS Package -- Custom Excise Package, should be stored in SQL Server(for SQL 11 Compatibility) not in File System

(This should be done in case of complex transformations or Tasks outside the Data Flow or SQL Tasks)

1. SQL Task
   1. Stored Procedure
   2. SQL Query
2. Data Flow Task
   1. Source
      1. Table
      2. Flat File
      3. Excel File
   2. Destination
      1. Table
      2. Flat File
      3. Excel File --this would enable to use the CHEF framework to create the Excel file dynamically for the Ad-Hoc reporting need in TDW

1. Source & Destination Server Name
2. Source & Destination Database Name
3. Source & Destination Table Schema Name
4. Source & Destination Table Name
5. Source & Destination Column List

with the respective Data Types

1. Source File Path in case of Flat or Excel File
2. Source File Name in case of Flat or Excel File
3. Source Sheet Name in case of Excel File
4. Source Attribute/Column

with respective Data Types for the Flat File or Excel Source

1. Structure of the XML Metadata

Expected for an Application (with sample data for understandability)

<?xml version="1.0" encoding="utf-8"?>

<CHEFMetaData ApplicationName="TDW">

<!--CHEF will create Package for each Process-->

<Process DefaultAllowTruncate="True|False" VerboseLogging="True|False">

<!-- SPARK: Explict way to configure transaction-->

<ConnectionSet>

<SQLConnection key="" ServerName="" DatabaseName=""/>

<FolderConnection key="" Path=""/>

<FileConnection key="" FileName=""/>

<FlatFileConnection key="" FileName=""/>

</ConnectionSet>

<!--Name is description. ID is unique & number type-->

<Step ID="" Name="" TypeID="" TypeName="">

<SQLTask TargetConnection="key" SQLStatement=""/>

<!-- SPARK: How to handle Error Redirection-->

<DataFlow SourceConnection="key"

TargetConnection="key"

SourceName=""

TargetName=""

SourceType="Folder | Excel | FlatFile | Table | SELECTSQL"

TargetType="Excel | FlatFile | Table"

PickColumnsFromTarget="True | False"

ColumnDelimeter=""

RowDelimeter="">

<ColumnMapping>

<Mapping SourceColumn="" TargetColumn=""

SourceType="" TargetType=""

SourceLength="" TargetLength=""

AllowTruncate="False|True"/>

</ColumnMapping>

</DataFlow>

<DataFlowSet SourceConnection="key"

TargetConnection="key"

SourceType="Folder | Excel | FlatFile | Table | SELECTSQL"

TargetType="Excel | FlatFile | Table"

PickColumnsFromTarget="True | False"

RunParallel="True | False">

<!--In case of excel Source Name is Sheet name or named range

In case of CSV, TXT, dat is same as file name-->

<DataFlow SourceName=""

TargetName=""/>

</DataFlowSet>

<!--No File based package support-->

<PackageExecution Connection="" PackageName=""/>

</Step>

</Process>

</CHEFMetaData>

1. Structure of the XML Global Config

Expected for an Application (with sample data for understandability)

<CHEFGlobalConfig ApplicationName="TDW">

<GlobalConfiguration Version="1.0" LogLocation="C:\Program Files\CHEF" InstallationBitLocation="C:\Program Files\CHEF" MaxBatchSize="1024" MaxLogTableSize="1024" NotificationAlias="ramsingh" SendNotification="True" ThresholdTimeInMinutes="10" OutputPackageLocation="C:\Program Files\CHEF\Output Package\CHEFPackage.dtsx" />

</CHEFGlobalConfig>

# CHEF Core Engine

This would be the crux of the CHEF in reading the metadata and creating the runtime SSIS Packages with tasks, parallelization, mappings, logging etc.

## Key Activities

1. Insert Log status for Start, intermediate operations/step, Finish to the CHEFLog table for the QueueID
2. Based on the input ProcessID, the first step in CHEF would read the XML Metadata file/column from the CHEFConfiguration table and pickup all the Transformations configured
3. It would create a new SSIS package with all the necessary transformations dynamically
4. It would add SQLTask Steps before and after every Transformation for logging the Status to CHEFLog table
5. In case of Partial Execution or Re-Start request from a specific step onwards for a Process, it would create a new SSIS package with only those steps, but include the basic steps that is needed for any data loading
6. After the SSIS package is created it would run the package
7. Request Queue to have Step Number instead of Step TypeID.(Change to be incorporated in Engine)

# CHEF Work Bench

This would be the UI (Web & Optional Windows) controlling all the transactions & actions and report for the CHEF. Any user is expected to use this interface to carry out all the operational tasks related to Data Loading for an Application.

## Key Elements of the UI

1. Display a list of ETL Operations/Services by reading the data from the XML metadata file
2. Display the Period/Frequency of Load by reading the data from the XML metadata file
3. Allow Start/Scheduling/Stop the Operation/Service
4. Allow Re-Start or Re-Loading from the point of Failure (at least at Task Level & at Sub Task/Step wherever possible )
5. Allow uploading of Flat Files/Excel Files for an Operation/Process
6. For Failed Operations, display all the failure details from Log table & Optionally from the Event Viewer Failed. Optionally show the Spurious/Bad Data that couldn’t be loaded for a Peroid.
7. Show Tasks & Sub Tasks in any Operations with the Status on the Last/Current Execution in Graphical/List
8. Show the Execution History Report of the Data Loading
9. UI Step description to come as hover.

Step row should have following in high level.  
Step Name         Status                   Button(Start/Stop)

1. Log to be displayed for a process step in a separate screen.  
   Log should be derived from the UDF [CHEF].[DataLoadStatus](calendar year, calendar Month, queueID)

# Appendix

## References

## CHEF Design Diagram

## Contributors & Documentation

|  |  |  |
| --- | --- | --- |
| Name | Activity Date | Comments |
| Balaji Mishra | 21st July 2010 | Initiated the CHEF Architecture Document with Ideas & Key High Level Areas |
| Balaji Mishra | 27th July 2010 | Updated the Document with formatting & sections for the Four Key Core Areas in CHEF |
| Balaji Mishra | 25th Aug 2010 | Updated the Document with new Design & name changes |
| Ram Janam Singh | 27th Aug 2010 | Updated the Metadata and Global Config files. |
| Kailash Choudhury | 8th Sept 2010 | Updated RequestQueue, UI and Engine section |
|  |  |  |
|  |  |  |
|  |  |  |