|  |  |  |
| --- | --- | --- |
| Corrections Data Architecture Specifications | October 12  2017 | |
|  | |  |

***Version 1***

The Brite Group Confidential and Proprietary Information;

Distribution or Disclosure is prohibited without TBG Prior Permission.

Document Modification History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version Number** | **Revised By** | **Revised On** | **Description** | **Revised Volumes and Sections** |
| 1.0 |  |  | Initial Version |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Architecture Specifications**

**Table of Contents**

[Document Modification History 1](#_Toc495561043)

[Architecture Specifications 2](#_Toc495561044)

[Table of Contents 2](#_Toc495561045)

[1.0 Overview 3](#_Toc495561046)

[2.0 Guiding Principles 4](#_Toc495561047)

[3.0 Architecture 5](#_Toc495561048)

[3.1 Raw Data Store 6](#_Toc495561049)

[3.2 Enhanced Data Store 7](#_Toc495561050)

[3.3 Materialized Data Marts 8](#_Toc495561051)

[3.4 Virtualized / Federated Data Store 9](#_Toc495561052)

[3.5 Metadata Store 10](#_Toc495561053)

[3.6 Data Portal 11](#_Toc495561054)

[3.7 Sandbox 12](#_Toc495561055)

[4.0 Frameworks 13](#_Toc495561056)

[4.1 Audit Trail Framework 13](#_Toc495561057)

[4.2 Exception Handling Framework 14](#_Toc495561058)

[4.3 Logging Framework 15](#_Toc495561059)

[5.0 Assumptions 16](#_Toc495561060)

# Overview

The document will provide a high level architectural specifications for the end to end data management for BJS next generation data platform to cater to the correction data pipelines. The objective of this document is to provide descriptions and purpose for each subcomponent and tiers of the architecture.

# Guiding Principles

* A single source of the truth.
* Consistent data management, governance, process and tools.
* Consistent alignment on KPIs across the organization and ready access to those KPIs.
* Align with key BJS IT initiatives and strategic investments.
* Shift BJS scientists’ workload from manual data ingest and preparation tasks to statistical analysis and research.
* A reduction in time to deliver statistical reporting and analytics.
* Better data currency and transparency.
* Improve insight through advanced analytics.
* Explore new and open technologies to promote next generation solutions (e.g. Big Data, machine learning, AI, self-serve analytics).
* Be agile and deploy capabilities incrementally based on prioritized business needs.
* All in coming data needs to be stored in its raw form with all versions as it goes through updates over time, along with capability to query and generate point in time snapshots.
* End to end workflow should be as much automated as possible, with minimal manual intervention except situations like decision making, compliance approval etc.
* All data, system and operational exceptions should be logged and stored in a reportable fashion.
* Build on existing platforms (strategic investments) but be open to integrate with new and upcoming technologies building best in class hybrid solutions.
* Do not take excessive risks by adopting unproven and brand new technologies but we want to be fast followers.
* Solution should easily adapt to future data changes. Build flexible, dynamic, configurable metadata driven solution.
* Solutions should be scalable both horizontally and vertically.

# Architecture

Following is the high-level end to end architecture diagram of the Corrections data pipeline. The architecture is comprised of several tiers with specific purpose and design considerations as explained in subsequent sections.



*Fig 1: End to End Architecture Picture for Corrections Data Pipeline*

## Raw Data Store

The data store is the area where the raw incoming data sets are landed. The collection files are received and copied as is into this area. The dataset can come in using a variety of methods like FTP, DB dumps, API pull etc.



*Fig 2: Raw Data Store*

The files are copied into the filesystem followed up by a number of checks. The initial list of checks are classified as structural sanity checks which will verify the structural integrity of the files coming in before loading them into the Landing database tables.

Once the data is loaded into the landing tables, second level of automated checks will be triggered which will execute a series of rules as configured in the metadata tables which will include data mapping rules, data profiling rules etc.

Third and final step in this process will be the load to the Staging tables, during which the data will be casted into a **standardized schema** followed by execution of a series of data validation rules. Staging database is responsible for holding the current input data set only until that is loaded into the “Standardized Data Store (SDS)” also known as “System of Record (SOR)”. Staging tables are not expected to hold any historical data. Stage tables will be exposed through the Data Portal for Business Analysts to perform data updates before that data is pushed into SDS.

Each step will be executed based on instructions saved in metadata tables and will capture output metrics which will be saved back into metadata tables for offline reporting.

## Enhanced Data Store

Enhanced Data Store is the area where the data is held over several time periods, or in other words holds historical data across all collection pipelines.



*Fig 3: Enhanced Data Store*

This area is comprised of 2 data stores:

**Standardized Data Store (SDS):** This database is very similar in structure to the Staging database as they will both follow the standardized schema; however, this database will hold historical data sets and will also hold multiple versions of the same data set as it goes through updates over time if any. This database will be exposed through the Data Portal for Analysts to perform data edits at any point in time as the need arises.

**Data Archive:** This database will receive its input from SDS based on either manual triggers or automated schedule, and will hold historical information for a longer duration than its source. Also in the process of data being pushed into this database, system will execute configured data transformations in order to enhance the data set.

## Materialized Data Marts

This is the hub of all Data Marts that are being built for Analytics and Reporting purposes across all collection pipelines.



*Fig 4: Materialized Data Marts*

Data transformation pipelines will be built which will operate based on configured business rules, transformations, aggregations, filtering rules and create purpose built Data Marts which will serve the needs of specific Data / Analytic tools. Certain canned datasets will be generated from these data marts for archival, but most importantly these data marts will be exposed through the Data Portal via the Data APIs for the end customer to extract data based on their needs.

## Virtualized / Federated Data Store

This area is key in building the next generation data hub for BJS where new virtual datasets can be built by comingling data from both internal and external datasets.



*Fig 5: Virtualized / Federated Data Store*

This area provides a key capability in creating solutions using the freshest data set by merging and transforming data from SDS as well as external datasets like FBI, census and other similar sources. Data transformations will be built using a data federation tool that would deliver the new datasets without moving the existing data bases or landing the data in any intermediary data stores, delivering performance and agility.

## Metadata Store

This area is the central store for all the configurations, business rules, transformation rules, system configurations, and any other information that is required for the system to operate under its highly dynamic and automated operation requirements. This will also store the outcomes of checks performed, exceptions and logging information as well for the end to end system.



*Fig 6: Metadata Store*

**Data Definitions:** Structural information of all incoming datasets and rules for verification. This will define the structure and taxonomy of all incoming data sets.

**Structural Check Results:** Outcomes of performing structural checks which are exposed through the Data Portal for visualization.

**Xwalk Rules:** Cross walk rules to map collection variables across multiple time periods and versions of surveys to the standardized schema.

**Validation Rules:** Data Validation and profiling rules to be configured for dynamic execution on incoming data.

**Validation Results:** Data Validation outcomes which are exposed through the Data Portal for visualization.

**Publish Rules:** Rules to transform, aggregate, filter data etc. as it is loaded into the purpose-built Data Marts.

**Metadata Management Interface:** This would be an amalgamation of tools, comprising of both custom-built interfaces as well as COTS product interface. The metadata we will store in this tier is a combination of reference data, business rules, validation rules, profiling rules, transformation rules and cross walk rules. All these categories of metadata will be managed and used by the best fit solution.

## Data Portal

This is a web portal which will serve as a central place for the end customers, analysts and IT personnel to visualize, edit, manage and consume the data and analytics served by the collection pipelines.



*Fig 7: Data Portal*

The data portal will be a central place to visualize several different functionalities. Below are the categories.

**Visualize Data Validation / Profiling Results:** This section would be several interactive reports and dashboards to visualize all the operations that were performed in the Raw Data Store, namely, results from the structural checks, data validation outcomes and profiles of data sets based on the rules. We will also have other report like exceptions, scheduling and workflows, activity and any other custom reports. These will be built using a standard visualization product.

**Data Management and Query Interface:** This is a custom web application that will provide data query as well as data management functionality. The query functionality will be purely driven by use cases defined and not open the database for custom querying. This interface will allow data edits also based on predefined use cases but will enforce the data audit and tracking framework and other house cleaning rules and activities behind the scenes to maintain a higher level of quality and traceability. These interfaces will be allowed to perform these operations on both the Staging as well as SDS databases.

**Data Access API and Tools:** These are tools and products that BJS exposes to the end customers. These will be a set of interfaces and APIs designed to access and extract the data from the Analytics Data Marts or the SDS.

## Sandbox

This is an area designed to enable the Data Scientists and Data Analysts with end to end capabilities to self-serve and perform exploratory data analysis and drive new use cases and insights. This area would have the capability to bring in data sets from any sources, install new and cutting edge tools and be very agile in building new capabilities without being completely dependent on IT.



*Fig 8: Sandbox Area*

The key here is to be agile ant be not tied down by IT processes. The objective of this area is not to build production processes but to create processes that can be productionalized. Hence users should be allowed to install open source and other types to tools, have connectivity to all available structured and unstructured datasets available within BJS and available publicly. This tier will be empowered with tools that allow self-serve for end users where they can perform end to end data provisioning (ingestion, validation, transformation, filtering, aggregations, modeling and visualization), all without having to rely on IT, hence delivering value much faster than otherwise possible.

# Frameworks

Following are the list of frameworks that will be built within this application to serve supportive functions for the data platform.

## Audit Trail Framework

This framework should be implemented on all database tables across the entire application irrespective of tier or purpose to enforce standard and simplicity. Also, enforcing a single pattern minimizes effort on programming side as well. This framework will be a set of columns as noted below to capture changes and multiple versions of the data as it goes through changes.

|  |  |
| --- | --- |
| Created By | User ID of the person or process who creates the record. When a record is created the “Updated By” and “Update Timestamp” will be left empty (NULL). |
| Creation Timestamp | Timestamp when the record is created. |
| Updated By | User ID of the person or process who updates the record. When a record is updated, we never physically update the record. Instead we will always expire the current row and insert a new row. Row expiration is done by updating the “Updated By” and “Update Timestamp” fields, and also updating the “Active Flag” to “N”. |
| Update Timestamp | Timestamp when the record is updated. |
| Version Number | Version number of the record. This is a numeric value starting with 1 and incrementing by 1. When a record is created for the first time a version number of 1 is put in. As this record gets updated (old record expired and new record inserted) a new version number is generated and put in this column in the new record. |
| Active Flag | This will have a value of “Y” or “N”. All active records will have “Y” and all inactive or expired records will have a value of “N”. |
| Process ID | This will be populated with the current process id like workflow/session execution id. |
| Process Name | A short description of the process which inserted or updated the record. |

## Exception Handling Framework

This framework ensures that we raise, catch and log exceptions uniformly at all steps within the end to end workflow. This uniformity promotes standard and simplicity in all programming modules that will implement this framework. This framework comprises of 3 parts:

**Exception Object:** This tracks a consistent set of metadata for every kind of exception, like exception level (environment, application, record level), severity, originating program, user id, timestamp, workflow, predecessor, error type, error message etc. This list will be enhanced and finalized during the detailed design phase.

**Exception Raising Process:** This is a module used by all steps to raise an exception. Each individual programming interface will trap all exceptions using their local method, but then invoke this global module to raise an exception uniformly. This will ensure that consistent metadata is captured for every exception.

**Exception Catching Process:** This is a module used by all steps to catch an exception log it into a database table. This will ensure we catch and log all exceptions uniformly and ensure standards.

Details about all these 3 modules will be laid out during the design phase.

## Logging Framework

This framework ensures that all application related log information is stored centrally and categorized so that they can be visualized and mined easily downstream as needed. The key feature of this framework would be a configurable set of parameters which can control 2 things:

**Level of logging:** This controls how much or how little information is posted in the logs. This is very helpful for debugging and issue resolution.

**Logging Objects and Modules:** This are the set of objects to ensure consistent set of metadata for the logs and also the process to write the logs from the heterogenous set of tools into a single uniform format.

Details about all these 2 modules will be laid out during the design phase.

# Assumptions

* The system is not required to support data ingestion directly from external systems (e.g. not accessing census bureau and capture data directly).
* The stake holders will be available for key decision making for review and selection of technologies.
* The proposed architecture will be presented by BJS, when required Brite group will assist in presenting the solutions.
* The system shall support integration with data virtualization subsystem.