

AOC
EDE Data Validation
High Level Design (Guidelines)

May 14, 2016

Prepared by

Vijay Kumar

Administrative Office of the Courts

1206 Quince St.

Olympia, WA 98504-1170

Revision History

Document Version #	Document Date	Summary of Changes	Revised By
0.1	09/22/2016	Initial Draft	Vijay Kumar

AOC Internal Review/Approval List

This document requires the following AOC internal approvals.

Name	Title	Date
Vijay Kumar	Solution Architect	
Eric Kruger	Program Architect	

AOC Internal Distribution List

This document has been distributed to:

Name	Title/Role
Kevin Ammons	Program Manager
Kumar Yajamanam	Architecture and Strategy Manager

Table of Contents

Revision History..... 2

AOC Internal Review/Approval List..... 2

AOC Internal Distribution List..... 2

Introduction..... 6

 Purpose..... 6

 Scope..... 6

 Business Drivers..... 7

 Risks..... 7

 Constraints..... 7

 Assumptions..... 7

Solutions..... 10

 Rules Management..... 10

 Scope..... 10

Rules Management..... 11

Rules Management Alternative Design..... 13

 Recommended Design..... 14

 Exception Handling and Logging..... 14

 Risks..... 15

 Benefits..... 15

 Rules Execution Engine..... 16

 Scope..... 16

Business Rules Execution Engine..... 17

 Recommended Design..... 20

 Exception Handling and Logging..... 20

 Why this Design..... 21

Risks 21

Benefits 21

Notification Service 22

Scope 22

Person/Actor Management and Notifications 23

 Recommended Design 26

 Exception Handling and Logging 27

 Why this Design 27

 Risks 27

 Benefits 27

 Address Cleansing 28

 Scope 28

EDR Address/Email/Phone Verification Context 29

 Why Stored Procedure for EDR Address/Email/Phone status update 30

 Recommended Design 33

 Exception Handling and Logging 34

 Risks 34

 Benefits 34

 Scoring & Identify Associations 35

 Scope 35

Person Scoring and Identifying Associations 36

 Why this Design 39

 Recommended Design 40

 Exception Handling and Logging 40

 Risks 40

Benefits 40

Introduction

The Expedited Data Exchange (EDE) Program along with multiple sub-projects was established to provide a mechanism for Washington courts that choose not to utilize an AOC provided Case Management System (CMS) solution to still provide statewide data in accordance with the published Judicial Information System (JIS) Data Standards for Alternative Electronic Court Record Systems. The core of this effort is the creation of the Enterprise Data Repository (EDR) which contains the data elements that conform to the published standards and will be the collection point for all case-management data regardless of the system in which the data was originally collected. This sub-project, Data Validation and Reporting Management (DVRM) is to validate the data submitted, tag any suspected data, and notify the source system owner for review and corrections, as needed, to any statewide data in the EDR.

As courts discontinue their use of the existing JIS CMS systems (SCOMIS & DISCIS), the need for statewide data to be available to all courts continues. Although courts still using the JIS CMS systems will have access to other courts' data using JIS, they will no longer have access to the data for any non-JIS court, nor will the courts that have moved off have access to the data for the courts still in JIS. These needs have expedited the construction of the EDR.

As courts migrate off of JIS systems to a variety of disparate systems, it becomes increasingly important that data in a statewide context undergo increased validation to ensure that the information stored in the EDR is consistent across all data sources.

Purpose

The purpose of this document is to provide a high-level design of the Data Validation sub-project. The intended audience is EDE Program staff including but not limited to Architects, Technical Leads, and Project Managers. The intent here is to provide sufficient direction for a technical lead to move forward in gathering detailed information that will assist them in producing the specifications necessary for developers and other implementation staff in fulfilling the needs of the EDE program and the Data Validation Track.

Scope

The EDR is the data repository and services for collecting and sharing data with courts and other stakeholders. Within the EDR, AOC will implement the Data Validation functions that will enable business processes to ensure well-vetted data across multiple systems. These functions will inspect and tag the data, as well as report back to source systems any issues found with submitted data. This document describes the DVRM processes and functions, plus lists the known requirements for this work.

Business Drivers

- Courts going off of JIS and implementing their own systems
- AOC's acquisition of COTS systems to replace existing JIS applications
- Must be able to visualize diverse data as a standardized view
- Need to have reliable information for public safety, judicial decision making and statistical analysis
- Need for a framework to improve data quality over time
- Need to be able to change and adapt quickly
- Need for data quality in EDR to be as good as, or better than, that in JIS.
- Need for a data quality framework to allow management of data validation rules and ongoing implementation by AOC staff.

Risks

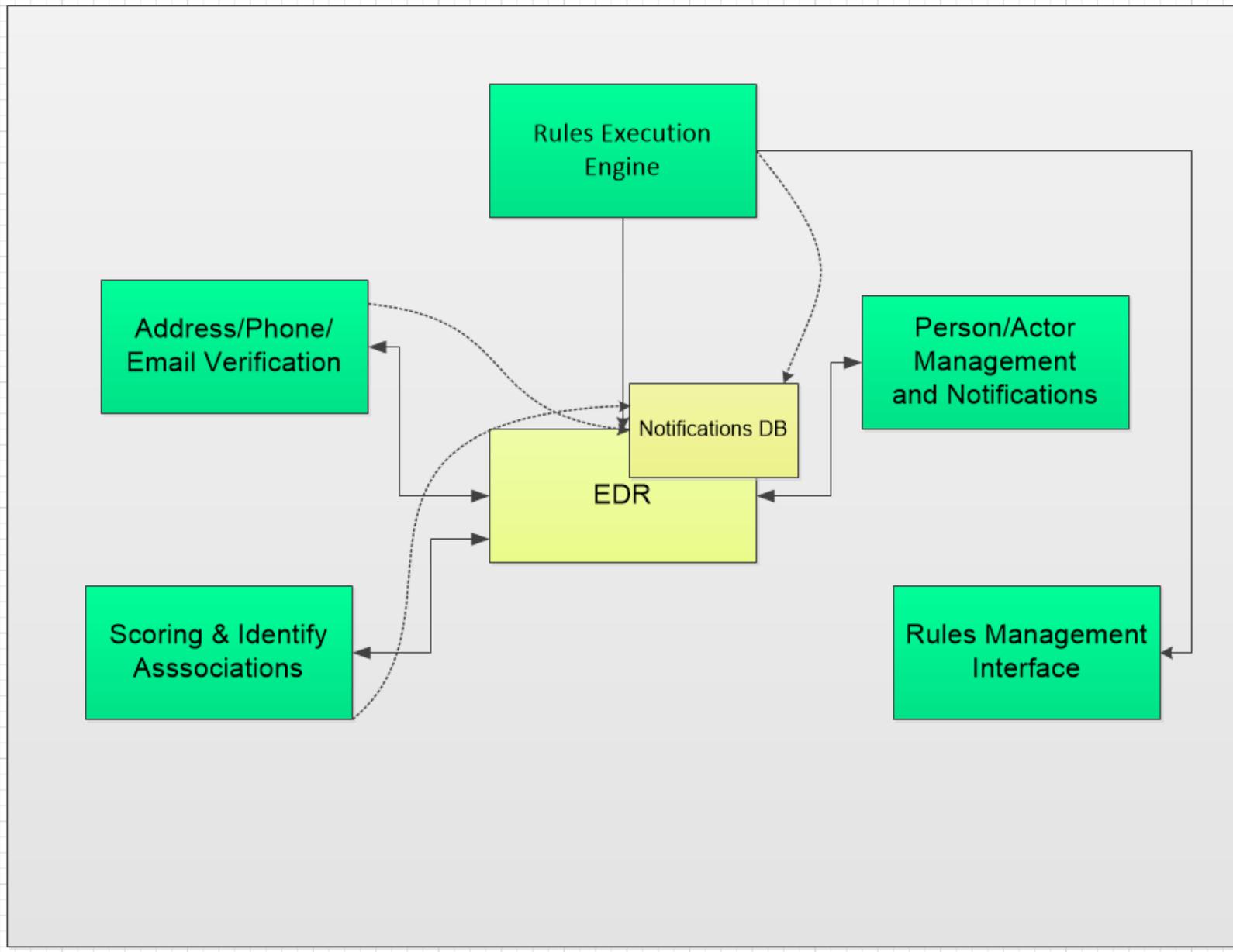
- Existing JIS data quality issues will not be resolved prior to loading from the JIS into the EDR and will cause the same/similar issues in the EDR.
- Data Validation error correction by court clients will not be resourced, as it is a new business function.
- Data validation rules and person matching rules require governance processes that are not yet established, thus causing possibly unacceptable delay in solution development.

Constraints

- Technology: SQL Server 2012, OData, OAuth, Entity Framework.
- Time: All design, construction and implementation work must be done and ready by the time the EDR/EDE system is ready for Go-Live, targeted for March 2017.
- Budget: Limited to what has already been scoped out and the EDE budget must be able to fund all aspects of the EDE Program as approved by the Legislature.

Assumptions

- The first courts outside of JIS to share data with the EDR will be King County District Court (KCDC) and King County Superior Court (KCCO).
- Data Validation will not delete user data; if necessary, it will flag the record as being corrected and write the corrected values to a new row.



Data Validation includes the following components:

- **Rules Management** – This enables the business to capture, store, and prioritize rules that data must adhere to.
- **Rules Execution Engine** – Based on identified rules, data is evaluated and the results are written to the database.
- **Address Cleansing** – This component provides address standardization, cleansing, and verification of deliverability for provided addresses based on conformance to the United States Postal Service (USPS) address standards (see [CASS](#)).
- **Scoring & Identity Associations** – This component utilizes various matching algorithms which use multiple attributes (including demographic and contact information) to determine the likelihood of identities matching.
- **Person/Actor Management and Notifications** – Source systems receive notification when rules identify that data needs further review or some other action is needed.

The data validation components will interact with the other EDE components as follows:

1. EDR data will be created, read, updated and deleted using OData Services (except Address cleansing).
2. All data will be saved in the EDR unless there are data type restrictions that prevent transaction completion (non-numeric data in a numeric columns, invalid date/time, etc.).
3. Incoming data will be flagged as “new,” requiring validation.
4. New/changed data will be validated as soon as possible.
5. The results of the data validation will be stored in the EDR database. For example, if Actor Hair Color is required, and not provided, then an error record will be written: “Actor Hair Color must not be blank.”
6. An update to a case will trigger the revalidation of all case data in accordance with case-data rules.
7. An update to a person will trigger revalidation of all person data in accordance with validation rules for a person, scoring, and then the identity matching process.
8. The identity scoring process will assign a number from zero to 100 to indicate the “completeness” of actor data, with zero being totally incomplete and 100 being perfect.
9. An update of an address will trigger the address validation. The address source is saved in the EDR, then the postal address standardization is performed. If the source address is valid, then it is flagged as such. If the address requires correction to achieve postal standards, a new address is stored for the actor in the EDR.
10. Each data validation error will have a unique error code. Each error code will have meta-data associated with it to provide the severity of the error, a long description, and method for user notification.
11. Data-validation statuses are marked with a begin-date and time when created. Those that are reviewed shall be marked with an appropriate status (such as completed, reviewed but not changed, etc.).

12. Validation rules are defined, documented, and tracked as they are managed through the data governance process. Once the rule is approved, some process is executed (coding, configuration, other) to implement the rule.
13. Data validation errors, address corrections, and identity matching will create notifications to originator of the data.
14. Notifications are viewed using some type of portal. Once viewed, a notification can be marked with an appropriate status (such as cleared, reviewed but not changed, etc.).

Solutions

Possible solutions are identified in this document immediately following the requirements for each component.

Rules Management

Data Validation rules need a method to be managed. The ideal solution would be one where the validation rules are defined in a repository, and that repository is used to validate the EDR database. A sophisticated rules-management and data-validation solution is a desirable long-term goal; however, it might not be feasible to implement in the time and budgetary constraints. Requirements identified as 'mandatory' must be met. Requirements identified as 'optional', are desirable if they can be met within time and budgetary constraints.

The data validation will consist of the following types of rules:

1. Valid source reference values (that is, a reference value maps to a standard reference value)
2. Unconditional existence (must exist, not be blank, be greater than zero, etc.)
3. Conditional existence (e.g., value must not be blank if another specified data item is not blank)
4. Simple business rules (e.g., date must be less than current date; value must be > specified number: etc.)
5. Complex business rules (e.g., value must be X if another value is greater than or equal to Y and was entered prior to a specified date)
6. Referential integrity rules (valid parent-child relationships).

Scope

The scope of high-level design is to obtain Rules Management. The context section that follows provides limited insight into how interactions with this component may proceed, but efforts that fulfill the specific requirements may differ from what is presented here.

Rules Management

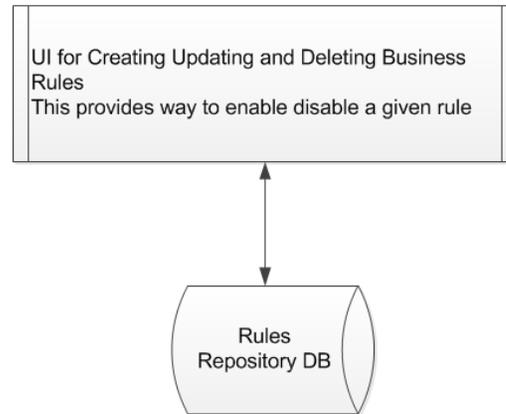


Figure 3 EDR Rules Management

Figure 3 above identifies sub-project Rules Management in the EDR Validation Track. It identifies components that support Rules Management. This Design will utilize ASP.NET Portal as a tool to display rules getting executed by a business rules engine. It will provide an interface for business analysts' (BAs') use in enabling or disabling a given rule, or editing corresponding configuration values.

The portal will fetch rules from the rules repository database (DB). This DB acts as a persistent storage of rules and their configurable values.

PROS:

- In this design, the user is provided with a user interface (UI) to view rules, edit configuration values, and enable/disable individual or sets of rules.
- It provides to the business analyst a 360-degree view of the rules engine to keep track of what is running in the system.

CONS:

- This is an additional effort to provide a rules engine view to end-user business analysts. A business rules engine can function without this.
- This can cause confusion for some business analysts, who may wrongly think that if they create a rule in this portal, it will be automatically implemented by the business rules engine.

Rules Management Alternative Design

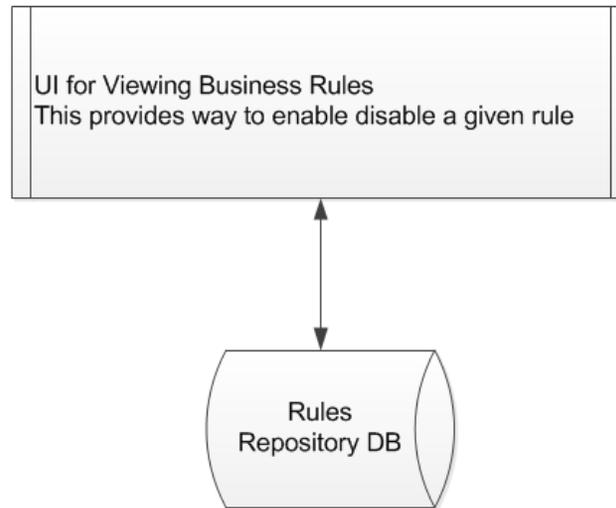


Figure 4 EDR Rules Management Alternative Design

Figure 4 above identifies a sub-project Rules Management within the EDR Validation Track. It identifies components that support Rules Management.

This Design will utilize ASP.NET Portal as a tool to display rules that are executed by the business rules engine. It will act as a rules viewer in which BAs can view and enable or disable a given rule.

The portal will fetch rules from the rules repository DB. This DB acts as a persistent storage of rules.

PROS:

- In this design, the user is provided with a user interface in which to view and enable/disable Individual rules or a set of rules.
- It provides a 360-degree view of the rules engine to the business analyst to keep track of what is running in the system.
- This is simple and aligns with the business rules engine, in that no rules can be added unless coded in the business rules engine.

CONS:

- This is an additional effort to provide a rules engine view to end-user business analysts. A business rules engine can function without this.
- This User Interface (UI) does not provide any view that would allow update of potentially configurable rules values.
- Because there is no current UI for entry of business rules into a business rules DB, it may be necessary to create additional SQL Script to insert rules into the DB.

Recommended Design

- This design (Design 2) is less complex and does not create any confusion for business analysts in terms of editing existing rules or creating new rules using the given UI.
- This design avoids any potential accidental updates to configuration values, which would disrupt business.

Exception Handling and Logging

All exceptions/errors will be captured using the standard logging tool (Log4NET) used in EDR.

Risks

- Access to this Portal should be restricted to designated business analysts only, since any enabling/disabling of rules needs to be carefully considered and implemented.

Benefits

This solution is custom built for the unique needs of AOC and, hence, is more flexible in implementing AOC's experience and years-long lessons learned.

Rules Execution Engine

These are the processes and components that actually conduct the inspection of the data, compare the data to the rules, and provide a mechanism for identifying whether (and to what degree) the data complies with the rules.

Scope

The scope of high-level design is to obtain a business rules execution engine. This applies to person and case data present in EDR. The context section that follows provides limited insight into how interactions with this component may proceed, but efforts that fulfill the specific requirements may differ from what is presented here.

Business Rules Execution Engine

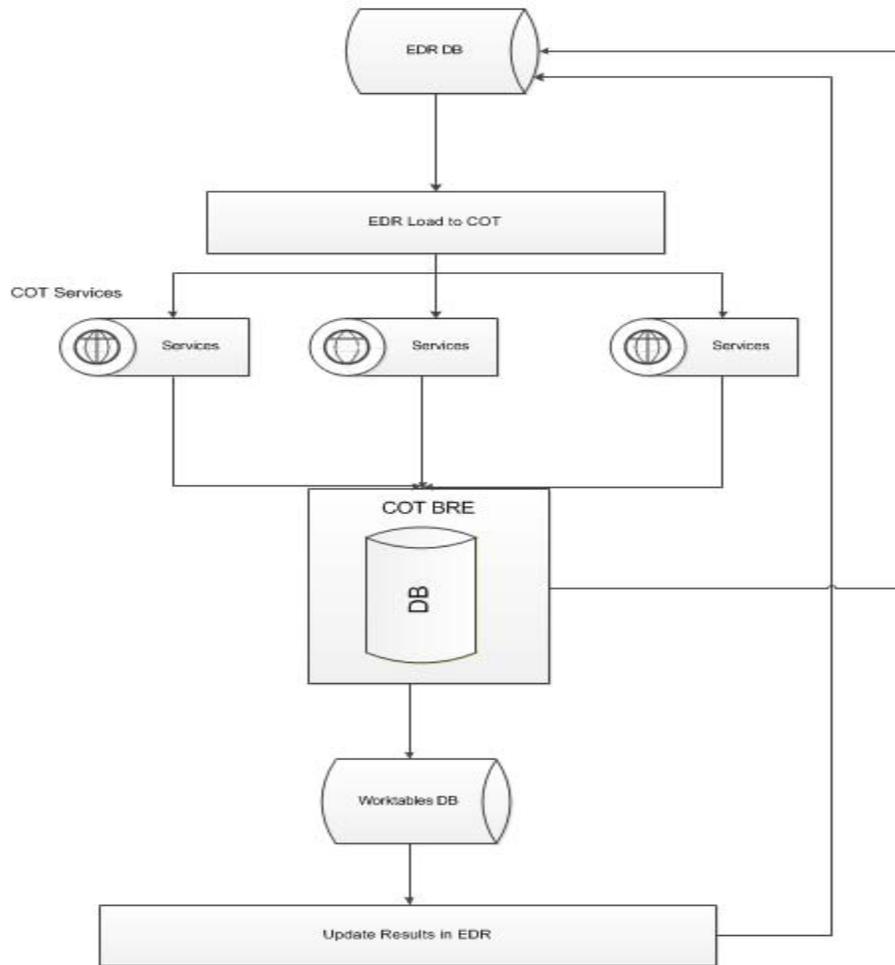


Figure 5 EDR Business Rules Execution Engine

Figure 5 above identifies a-sub project Business Rules Execution Engine in the EDR Validation Track. It identifies components that support rules execution on EDR Data. This Design will utilize a set of .NET components and COTS package as a tool to implement business rules on EDR data.

A custom process will fetch person and case records from EDR using OData Service Call. Data will be mapped to COTS service request for data validation. Findings of validation will be updated back into EDR using an OData Call, and the notification record (if any) will be created in a notifications DB (part of EDR DB).

Business Rules Execution Engine Alternative Design

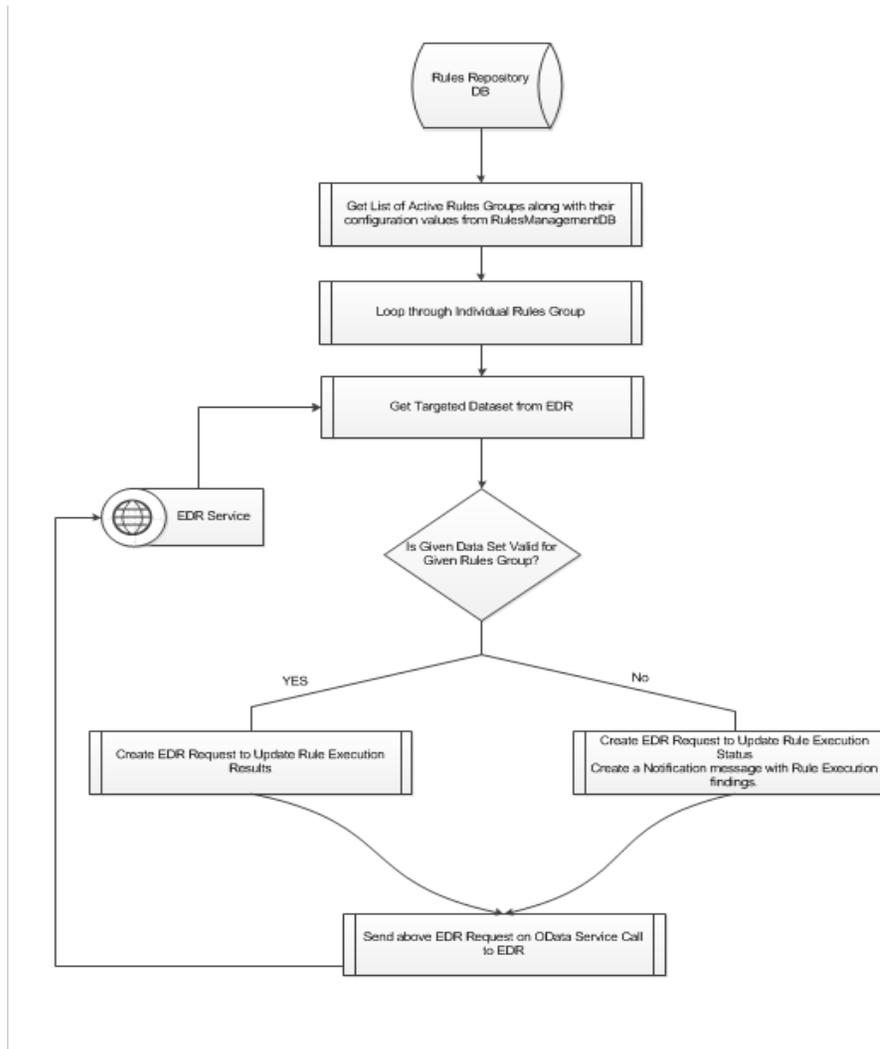


Figure 6 EDR Business Rules Execution Engine

Figure 6 above identifies a sub-project Business Rules Execution Engine in the EDR Validation Track. It identifies components that support Rules Execution on EDR Data. This Design will utilize Windows Service and a set of .NET executables developed using C# as a tool to implement a business rules execution engine on EDR data. It will leverage multi-threading and asynchronous calls to OData service for parallel processing.

The tool will fetch person and case records as a JavaScript Object Notation (JSON) object from EDR using OData Service Call. JSON object will be loaded into corresponding person and case objects to be used in data validation. Validation findings will be updated back into EDR using an OData call, and the notification record (if any) will be created in the notifications DB.

Once rules execution is finished, the record is marked as “processing.”

Recommended Design

- This design (Design 2 Alternative Design) is better structured and has more flexibility in terms of implementing complex rules.
- This design is implemented using the latest available technology, while leveraging enhanced hardware and operating-systems capabilities.
- This will be much easier to maintain, since it is custom built for given AOC requirements by one of the widely used technologies like Microsoft .NET C#.
- This Design is extendable to accommodate more rules in the future, as they become necessary.

Exception Handling and Logging

All exception/error records will be updated with EDR validation error status in EDR, along with corresponding error message.

This application will use Log4NET for logging with the same template as is used in other applications.

Why this Design

- This design divides the whole process into individual components which are responsible for a given functionality, allowing development and deployment independently. This makes the solution more robust and easier to develop and maintain.
- This design is proposed if all options to buy or rent commercial off-the-shelf products (COTS) are exhausted and a decision is made for In-house development.
- In the long run, this design would be easier to maintain.

Risks

- Quality and success of this design will depend on using a good combination of C# programming features like multi-tasking, asynchronous service calls, and optimized code.
- This is a new application-development effort, hence development needs to be planned and monitored closely to avoid any slippage in timelines.

Benefits

This solution is custom built for unique needs of AOC, making it more flexible in implementing AOC's experience and lessons learned over the years. This solution is scalable and multi-threaded, and so is a good fit for AOC volume requirements.

Notification Service

The notification component is the mechanism in which the EDR communicates with source systems as well as with the AOC Data Quality program.

Scope

The scope of high-level design is to obtain a Notifications Service. This applies to all notifications sent out as a part of the data validation process. The context section that follows provides limited insight into how interactions with this component may proceed, but efforts that fulfill specific requirements may differ from what is presented here.

Person/Actor Management and Notifications

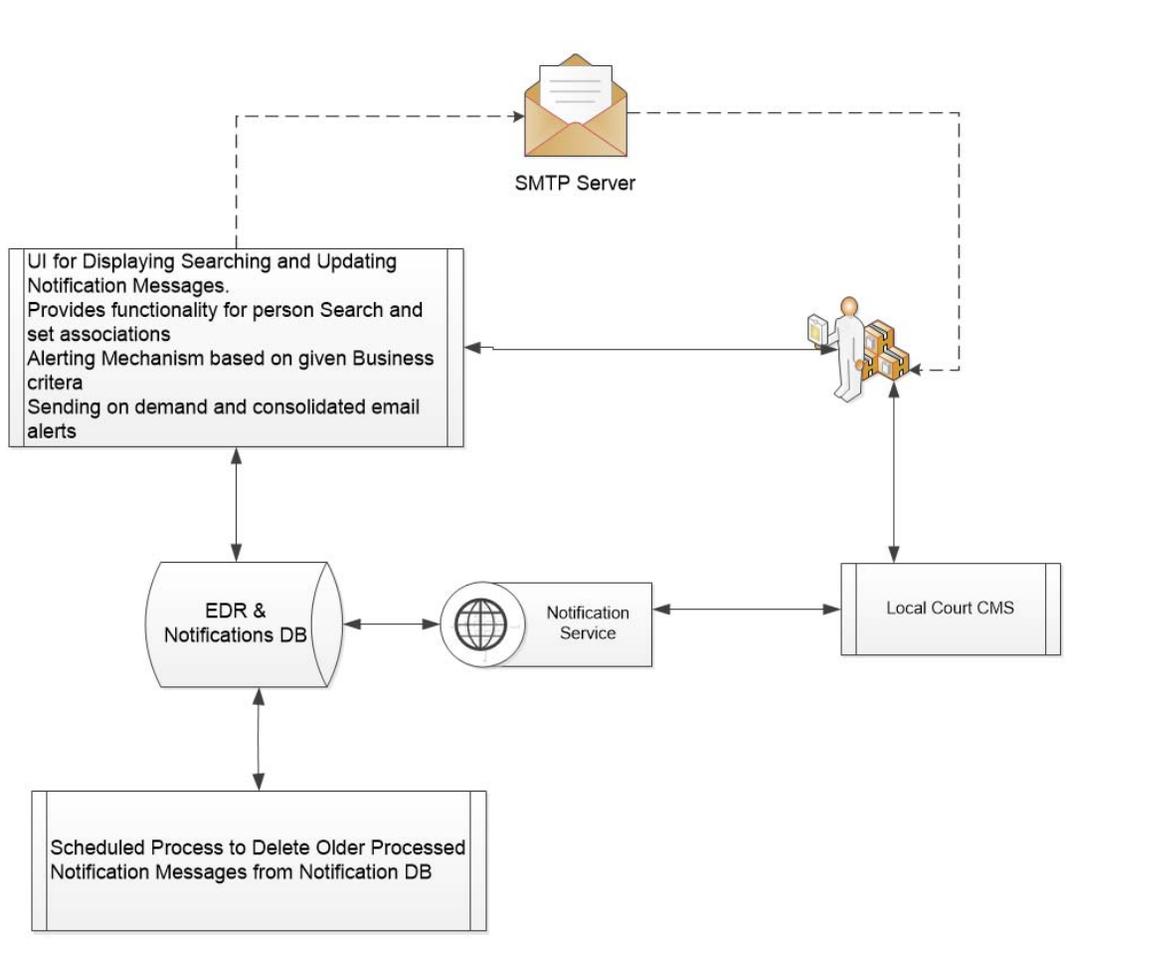


Figure 7 Person/Actor Management and Notification Service

Figure 7 above identifies a sub-project Notification Service in the EDR Validation Track. It identifies components that support sending notifications following execution of business rules, address verification, and the identity-association process.

This Design will utilize ASP.NET portal and a Windows service. EDR clients will be granted role-based access to this ASP.NET portal. Other courts can view and update notification status.

Windows service will also include functionality for grandfathering and deleting old notifications after a configurable time span has passed.

The UI will provide functionality for local court staff to manually set new person associations (“alias relationships” in legacy JIS) or delete existing associations. It will provide Person search functionality, based on criteria defined in use cases.

The tool will send consolidated or on-demand email alerts based on pre-defined requirements. Findings from data validation components are stored in the notifications DB. This portal provides a mechanism to manage and deliver them to the intended audience.

Notifications Service Alternative Design

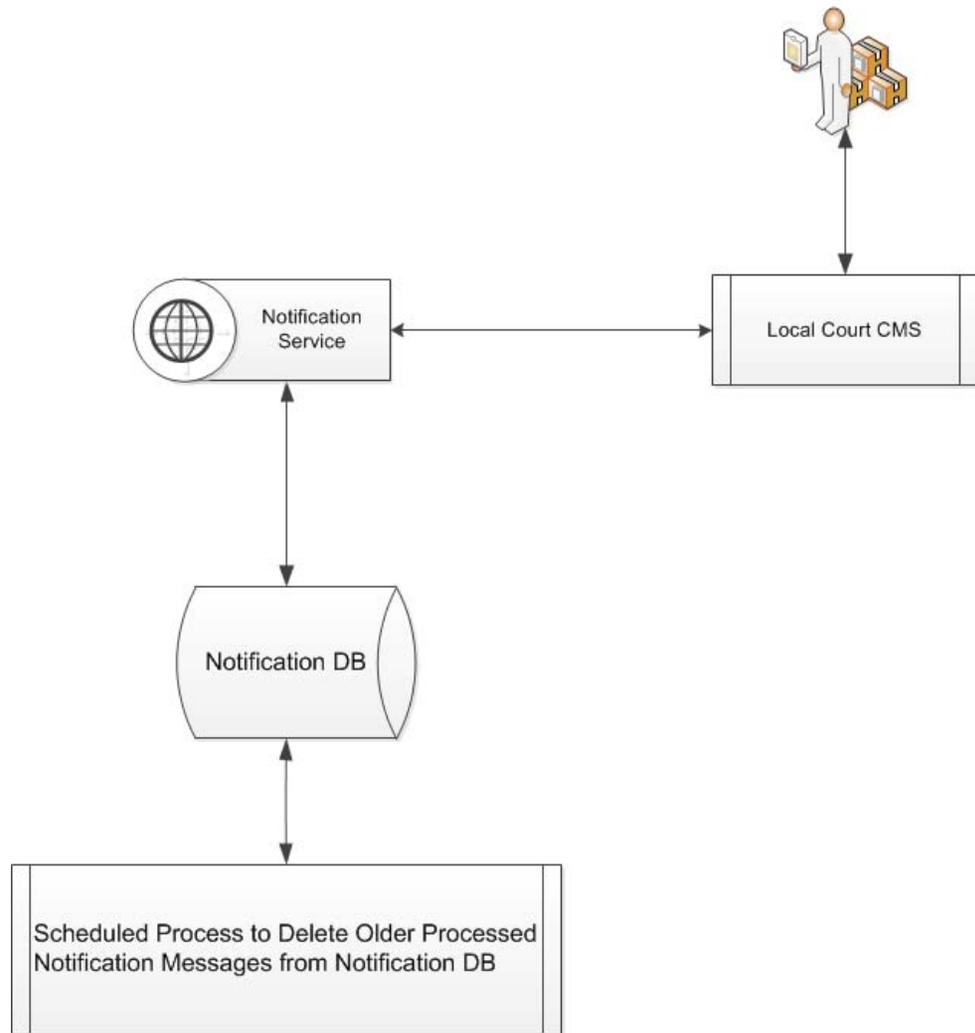


Figure 8 Notification Service alternative Design

Figure 8 above identifies a sub-project Notification Service alternative design in the EDR Validation Track. It identifies components that support Notification Service.

This Design will utilize Representational State Transfer (RESTful) OData Service to send notifications to local courts' case-management systems. It will publish notifications created during the validation process to the corresponding courts. The Service DB will be updated in real time during execution of the validation process, so that local courts will have immediate access to notifications.

Local courts' case-management systems can fetch notifications in JSON or XML format and display them in their own application. Once a user takes an action on a given notification, the local court's CMS will update status in the notification database using notification ID and OData service call.

PROS:

- In this design, the user is provided with real-time data in JSON or XML format to integrate into their own system to avoid switching systems.
- This design is easier to implement (from the AOC prospective), since it leverages UI provided/created by the local court's CMS.

CONS:

- This approach will reduce the notification mechanism to a service, only. Lack of an AOC-provided UI will force consumers (local courts) to immediately either integrate in their CMS or develop a new UI for themselves.
- Some EDR customers who do not have their own CMS or UI will have no other option than for one to be developed for them.

Recommended Design

- This design (Design 1) is more complete in terms of information flow and provides an alternate UI option or at least an intermediary arrangement until a solution for integrating services into the local-court CMS is in place.
- With addition of a few more EDR tables, this design leverages the existing EDR DB, thereby eliminating the need to create and maintain a separate Notifications DB.

Exception Handling and Logging

All exceptions/errors will be captured using the standard logging tool (Log4NET) employed in EDR. Application will leverage logging built in EDR OData Service to capture logging and exceptions in the notification service. Local Court CMS development/exception handling and logging is out of scope for this effort.

Why this Design

- This design provides a mechanism to store, display, and manage notifications in a persistent and secure way. Notification may need a workflow for a given incident, as well as an assignment and tracking mechanism. This design provides both.
- This design sends consolidated email alerts at a pre-configured time or business event.
- This design provides a solution to grandfather and ultimately delete older notifications, making the system self-sustainable and reducing support work in the long run.
- It provides a feature to send email alert on demand, if needed.
- This design is proposed once all options to buy or rent a COTS product are exhausted and a decision is made for In-house development.
- This decision would be easier to maintain in the long run.

Risks

- Quality and success of this design will depend on using a good combination of C# programming features and efficient use of UI technologies.

Benefits

This solution would be custom built for AOC, and hence would be more flexible in meeting AOC's unique needs.

Address Cleansing

The address cleansing component will be used to ensure mailing addresses are accurate and deliverable.

Scope

The scope of high-level design is address/email/phone verification in EDR, and applies to all such contact information present in EDR. The context section that follows provides limited insight into how interactions with validation may proceed, but efforts which fulfill the specific requirements may differ from what is presented here.

EDR Address/Email/Phone Verification Context

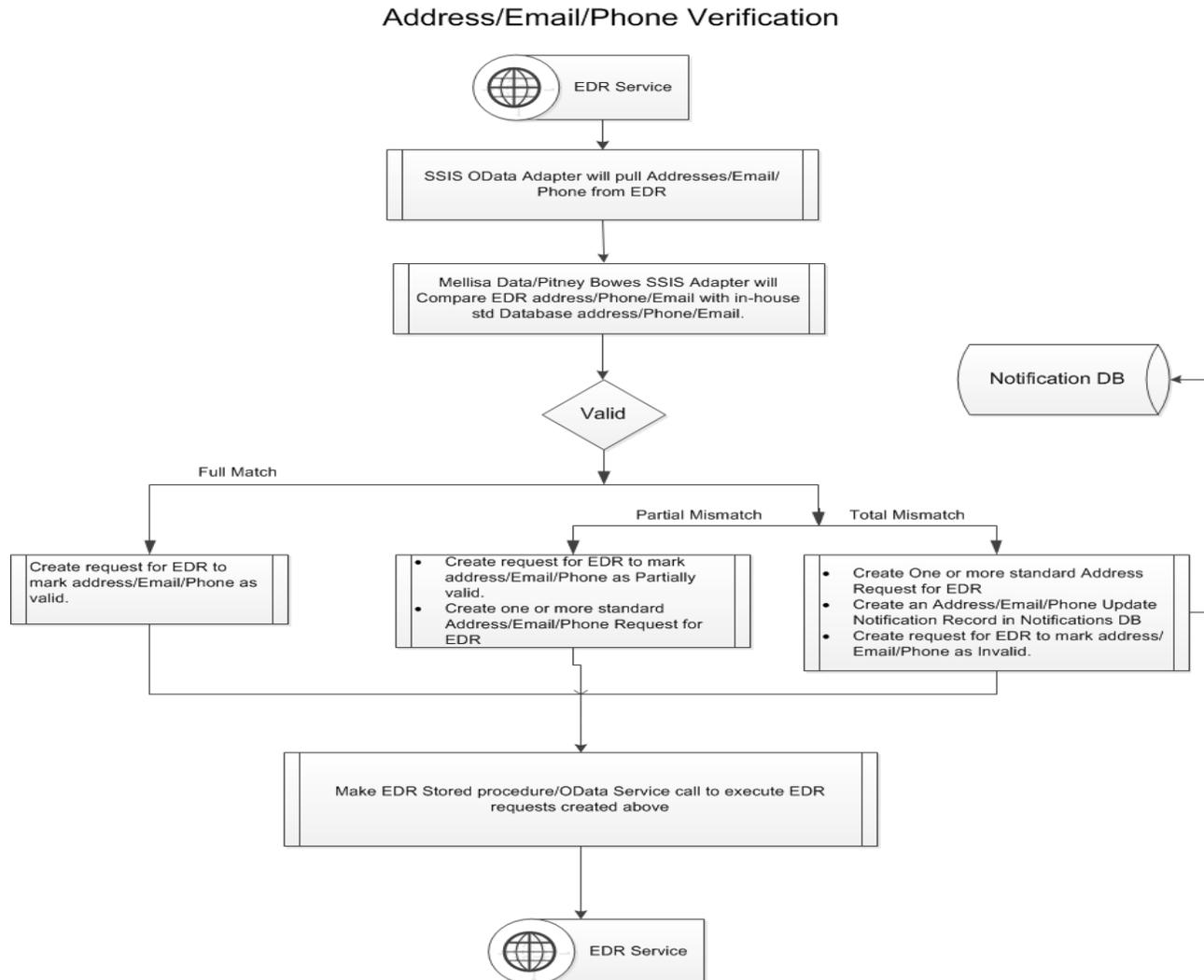


Figure 9 EDR Address/Email/Phone Verification Overview

Figure 9 above identifies a sub-project EDR address/email/phone verification within the EDR Validation Track. It identifies components that support this verification.

This Design will utilize SSIS (SQL Server Integration Services) as a tool to implement address/email/phone cleansing. It will leverage SSIS OData adapter to get address/email/phone data from EDR Service. The SSIS package will be running from a SQL agent scheduler on a configurable amount of time.

Once Data is in the SSIS dataflow task, it will use vendor SSIS custom component to verify the data against the standard database hosted within AOC premises. A stored Procedure will be used to mark a given address as invalid, verified or partially verified. The procedure will additionally generate an error message, if applicable. A complete match is marked as verified. In case of a partially cleansed address, a standard address record is created in EDR. For an invalid address, one or more recommended addresses (provided by data validation service) is enclosed in the notification to source system.

Source system will verify the recommended address and will update it in the source system. A CMS to EDR integration process will create this address in EDR to close the loop. This step is not part of the proposed Address Cleansing Process, but is added in this document to give a complete picture.

Why Use Stored Procedure for EDR Address/Email/Phone Status Update

PROS:

- Stored procedures are saved in compiled format in the database, making them faster for any TSQL (Transact-SQL) operation, as compared with parameterized queries.
- Stored procedures provide a level of abstraction, so as to have minimal impact in case of underlying schema changes.
- Stored procedure is one of the fastest available patterns to accomplish a given data change in SQL Server. SSIS is built on SQL Server stack, and it has access to internal SQL Server API's.
- No EDR data reads are executed using these stored procedures, so the security layer built in EDR Service remains intact.
- This approach is easier to maintain, due to isolation of SSIS package and EDR DB schemas.

- These procedures will be written and maintained by the Data Validation team. Therefore, negligible/no effort would be required from the EDR Core team. The EDR DBA would need to provide EXEC permission to SSIS user account for these stored procedures.
- This approach will help EDR service by redirecting some traffic to stored procedures.

CONS:

- EDR Core agreement would need to be attained to allow access to their database using stored procedure.
- Although no data reads would be made, a loophole will nonetheless be created for bypassing the security model to access the EDR database built-in EDR service.
- The current version of SSIS services does not have any in-built adapter to post data to OData Service.
- Another third-party, open-source, or custom adapter is too slow to handle EDR traffic.
- Utilization of script component or other .NET component running within SSIS process to make OData calls is an option, but this will compromise processing time.

Alternative Design

Address/Email/Phone Verification Alternative Design

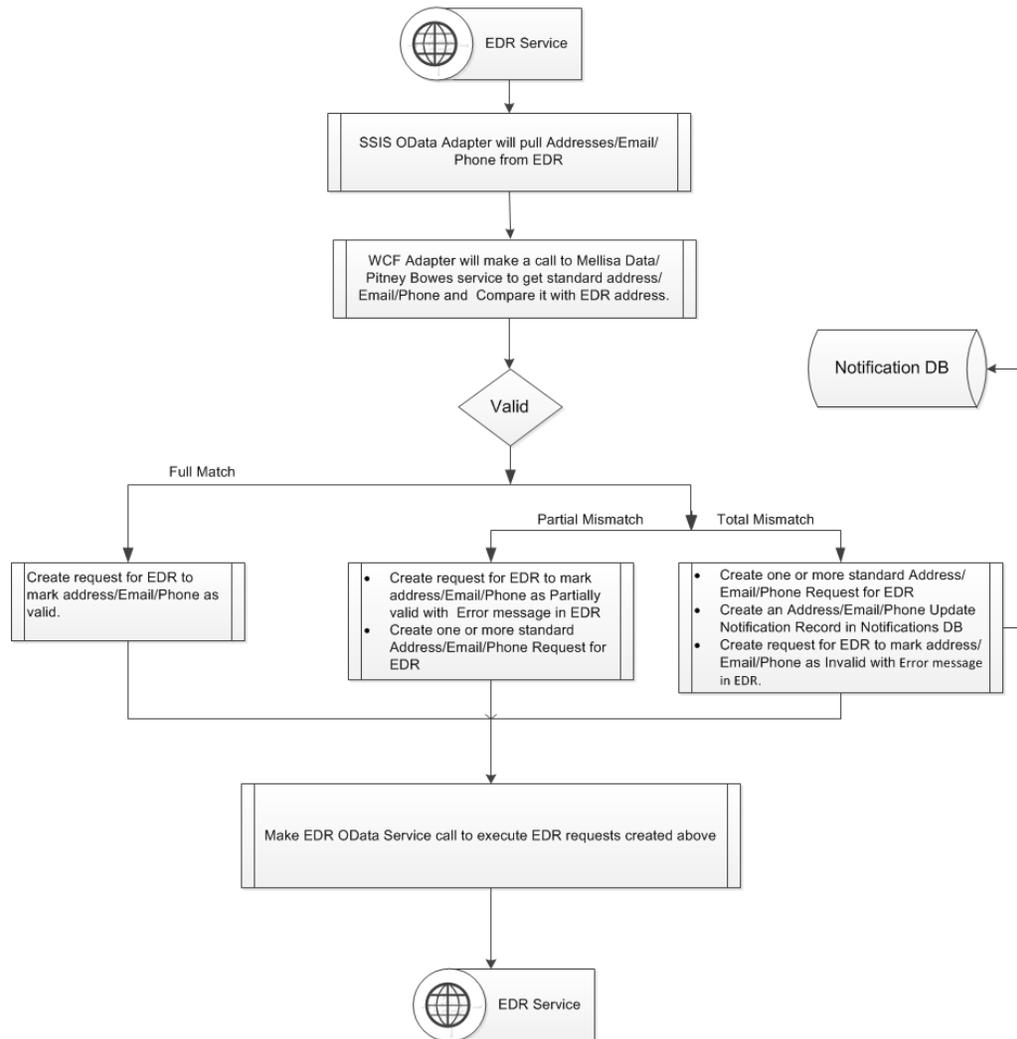


Figure 10 EDR Address Verification alternative Design

Figure 10 above identifies a sub-project EDR address/email/phone verification alternative design within the EDR Validation Track. It identifies components that support address/email/phone verification.

This design will utilize SSIS as a tool to implement address cleansing. It will leverage SSIS OData adapter to get Address Data from EDR Service. SSIS package will be running from a SQL agent scheduler on a configurable frequency.

Once data is in the SSIS dataflow task, it will make a WCF (Windows Communication Foundation) call to vendor from the SSIS component to verify it against the standard-address database hosted on vendor premises.

An OData service call to EDR will be used to mark a given address as invalid, verified or partially verified, as well as to record an error message, if any. A complete match is marked as verified. In case of a partially cleansed address, a standard address record is created in EDR. For an invalid address, a recommended address (if provided by data validation service) is enclosed in the notification for source system.

PROS:

- In this design, address verification calls are made to vendor service; hence, the overhead of maintaining a standard-address database within AOC premises is eliminated.
- This design would be easier to maintain, due to isolation of the SSIS package and direct calls to vendor service.

CONS:

- This approach will restrict address-verification service availability to vendor-service availability, thereby introducing the need to synchronize with their downtimes.
- This design is making web service calls from SSIS, which can create performance issues with increased load.
- Web service calls to vendor service and OData calls to EDR can prove to be a performance damper and can make it difficult to meet validation requirements for the initial load.

Recommended Design

- This design (Design 1) is more efficient in terms of completing the given task and can be implemented in less time, due to minimum custom coding.
- This design uses an in-house database provided by the address-verification vendor. Address/email/phone cleansing task is completed within AOC firewalls, thus addressing issues related to theft or misuse of person data.

- This design is independent of 24x7 internet availability requirement, thus making it more stable and robust.
- SSIS is selected as a tool because it is built on SQL Server stack and has the best support for native SQL Server adapters.
- Validation is one of the features built on top of EDR, so allowing direct access to DB using Stored Procedure can help in reducing the load on EDR service.
- This design would be easier to maintain in the long run.

Exception Handling and Logging

All exception/error records will be updated with Address Service Verification Error status in EDR along with error message.

At package level, in-built logging features of SSIS will be used to capture audit-log information.

Risks

- This design relies on the assumption that EDR will allow creation of a few stored procedures in their database and provide execute permission to SSIS user account.
- EDR OData Service calls with custom code can have performance bottlenecks.
- Quality and success of this design will depend on how frequently Address Verification Database is updated to the most current version.

Benefits

The recommended solution is built using SSIS, which is one of the best available tools from SQL Server to SQL Server data conversion. It is a multi-threaded environment, and most fitting, considering AOC load requirements. It reduces development time and, hence, maintenance overhead is limited.

Scoring & Identify Associations

The purpose of the Identity Matching component is to uniquely identify individuals across the various court systems and jurisdictions. This will enable judicial officers and other court staff to review a complete and accurate case history for an individual appearing in their court.

Business and data subject matter experts (SMEs) must identify and enter matching rules for person identities. The rules and matching thresholds are entered into DQS through the DQS client. Using the DQS Matching component in SSIS, a transformation job executes which reads identity data that has yet to be matched, as well as the DQS matching rules. Based on the rules, the DQS Matching component places matched records in a target location that is later read by another transformation that will insert the matches into the EDR.

Scope

The scope of high-level design is to obtain person-record score and to identify person associations among persons stored in EDR. The context section that follows provides limited insight into how interactions with this component may proceed, but efforts which fulfill the specific requirements may differ from what is presented here.

Person Scoring and Identifying Associations

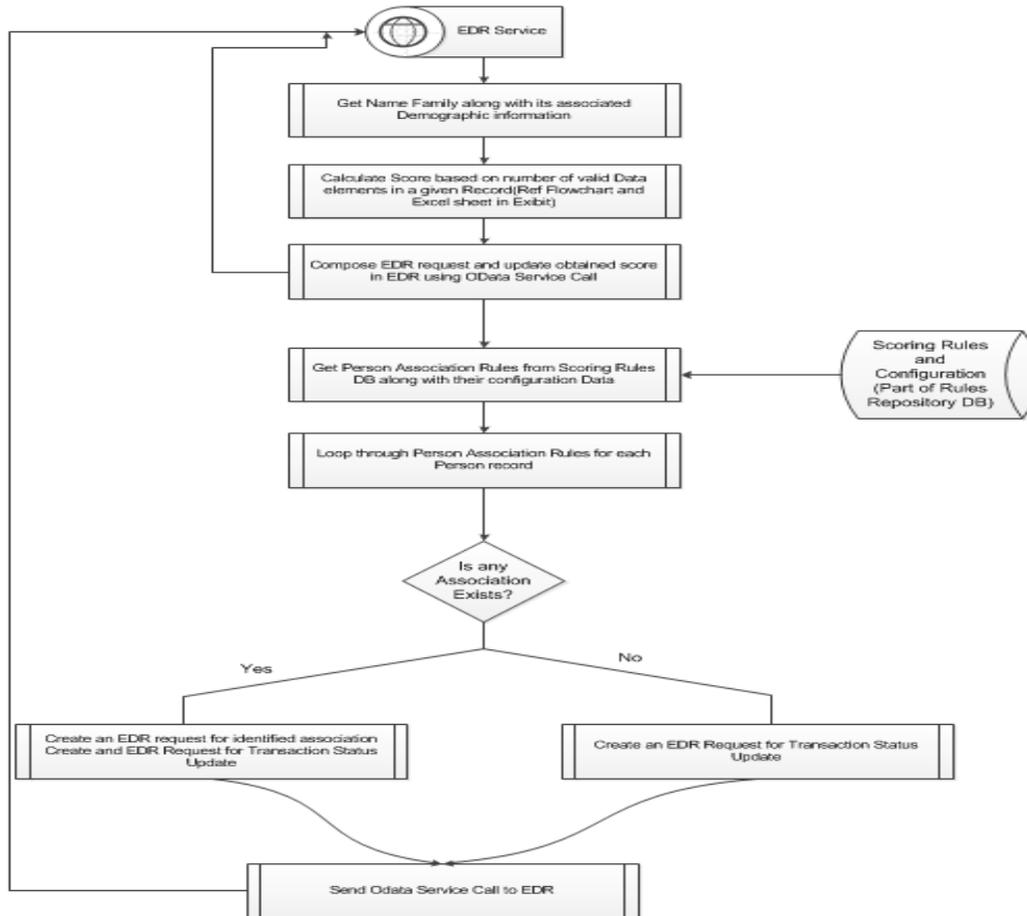


Figure 11 EDR Person Scoring and Identifying Associations

Figure 11 above identifies a sub-project Identity Management within the EDR Validation Track. It identifies components that support person scoring and person associations. This Design will utilize Windows Service and a set of .NET executables developed using C# as a tool to implement Identity Management. It will leverage multi-threading and asynchronous calls to OData service for parallel processing.

The tool will fetch person records as a JSON object from EDR using OData Service Call. JSON object will be loaded into a person object which will be used in calculating person-record score (an indicator of person-information completeness). Once the score is determined, an OData call will be sent to EDR to update the score.

In the next step, the tool will fetch person-association rules and corresponding configuration values from the DB, then loops through all applicable rules to determine associations. OData calls are made based on a specific query for a given rule and inference is reached as to whether a given person has any associations in EDR or not. If an association is identified, a new record is created in the EDR Associations entity with corresponding keys, using an OData Service call.

In the last stage, the record is marked as “processing complete” for association identification.

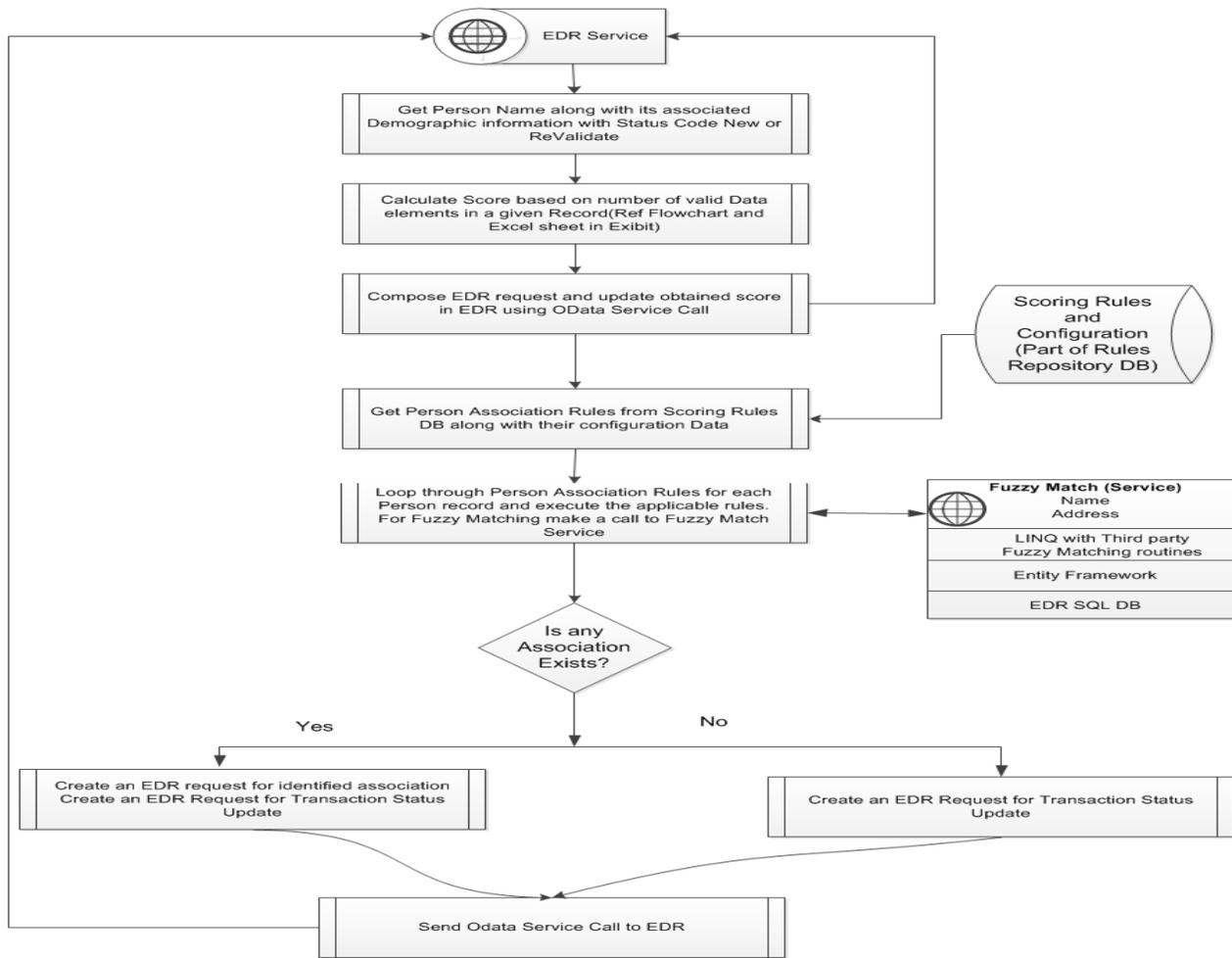


Figure 12 EDR Person Scoring and Identifying Associations Alternative Design

Figure 12 above identifies sub-project Identity Management within the EDR Validation Track. It identifies components that support person scoring and person associations. This Design will utilize Windows Service, WCF Service and a set of .NET executables developed using C# as a tool to implement Identity Management. It will leverage multi-threading and asynchronous calls to OData service for parallel processing.

The tool will fetch person records where address verification is completed as a JSON object from EDR using OData Service Call. JSON object will be loaded into a person object, which will be used in calculating person record score. Once the score is determined, an OData Call will be sent to EDR to update the score.

In the next step, the tool will fetch person-association rules and corresponding configuration values (if any) from DB. The process will loop through all applicable rules to determine associations. OData calls are made based on a specific query for a given rule, and inference is reached as to whether a given person has any associations in EDR or not.

In this design, we will leverage a third-party fuzzy-name/address matching routine for fuzzy-match requirements. Fuzzy Match Service will leverage LINQ and Entity Framework along with third-party or open-source fuzzy-match logic. If an association is identified, a new record is created in the EDR Associations entity with corresponding keys, using an OData Service call.

In the last step, the record is marked as “processing complete” for association identification. Multi-threading/multi-tasking and asynchronous calls are used whenever they result in reducing average record-processing time.

Why this Design

- This design divides the whole process into individual components responsible for specific functionalities. The components can be developed and deployed independently, making the solution more robust and easier to develop and maintain.
- This design is proposed once all options to buy or rent a COTS product are exhausted and a decision is made for In-house development.
- This design is easier to maintain in the long run.

Recommended Design

- This design (Design 2 Alternative Design) is better structured because it moves fuzzy-search logic to a separate service which is designed and fine-tuned for a given task.
- This design uses a third-party/open-source utility for fuzzy matches, which has the advantage of having been tested over a period of time and covering all possible combinations of match.

Exception Handling and Logging

All exception/error records will be updated with EDR validation error status in EDR, along with the corresponding error message.

This application will use Log4NET for logging with the same template as is used in other applications.

Risks

- Quality and success of this design will depend on using a good combination of C# programming features like multi-tasking, asynchronous service calls, and optimized code. Fuzzy-records matching is a resource-intensive activity and needs to be designed and implemented using best practices.
- Implementing fuzzy-match logic using OData Service can be a serious performance bottleneck.
- Use of OData Service for all EDR communication will be a serious problem area.
- EDR is a highly normalized database and, as a result, the data-validation process will be heavy on select queries. Validation process performance will be directly proportional to EDR performance.

Benefits

This solution is custom built for the unique needs of AOC and is therefore more flexible in implementing what AOC has learned over the years. This Solution is scalable and multi-threaded and is consequently the best fit for AOC volume requirements.