**High Level Design Document For**

**ASAPI v2.5**

|  |
| --- |
|  |
|  |

**Table of Contents**

[1. Title 5](#_Toc386981433)

[2. Introduction 5](#_Toc386981434)

[3. Purpose 6](#_Toc386981435)

[3.1. Target Audience 6](#_Toc386981436)

[3.2. Next Step 6](#_Toc386981437)

[4. Traceability 7](#_Toc386981438)

[4.1. Technical Mapping 7](#_Toc386981439)

[4.2. Functional Mapping 7](#_Toc386981440)

[4.3. Non-Functional Mapping 7](#_Toc386981441)

[5. Design Considerations 7](#_Toc386981442)

[5.1. Assumptions 7](#_Toc386981443)

[5.2. Limitations & Constraints 7](#_Toc386981444)

[6. System Architecture 7](#_Toc386981445)

[6.1. Logical Architecture 7](#_Toc386981446)

[6.2. Recommended Code Deployment 9](#_Toc386981447)

[6.2.1. Impact Analysis 10](#_Toc386981448)

[~~6.2.2.~~ ~~Context Diagram~~ 10](#_Toc386981449)

[6.3. User Interface 10](#_Toc386981450)

[6.4. Physical Architecture 10](#_Toc386981451)

[6.4.1. System Diagram 10](#_Toc386981452)

[6.4.2. Hardware elements 10](#_Toc386981453)

[6.4.3. Software elements 11](#_Toc386981454)

[7. Alternative Solutions and criteria for selection 11](#_Toc386981455)

[7.1. Design alternate 1 – Limiting number of search records 11](#_Toc386981456)

[7.2. Design alternate 2 13](#_Toc386981457)

[7.3. Criteria for selection 13](#_Toc386981458)

[7.4. Solution selected with rationale 13](#_Toc386981459)

[8. System Details 13](#_Toc386981460)

[8.1. Interrelationship and interfaces between the components 13](#_Toc386981461)

[9. Data Model 13](#_Toc386981462)

[9.1.1. Database Design 14](#_Toc386981463)

[9.2. Application/System Integration - Request/Response Signatures 14](#_Toc386981464)

[9.2.1. ASAPI v2.5 & later Request Signature 14](#_Toc386981465)

[9.2.2. ASAPI Response XML 18](#_Toc386981466)

[9.2.3. ASAPI Response Codes v2.5 20](#_Toc386981467)

[9.2.4. System Configuration Tokens 22](#_Toc386981468)

[10. Re-usable Components 23](#_Toc386981469)

[11. High level Security Design Assessment 23](#_Toc386981470)

[11.1. Sensitive Data 23](#_Toc386981471)

[11.2. Internal Interface 24](#_Toc386981472)

[11.3. External Interface 24](#_Toc386981473)

[12. Non Functional Requirements 24](#_Toc386981474)

[12.1. Automated Testing 24](#_Toc386981475)

[12.2. Performance Benchmarking 24](#_Toc386981476)

[12.3. Logging 25](#_Toc386981477)

[12.4. BI 25](#_Toc386981478)

[12.5. NOC Dashboard 25](#_Toc386981479)

[12.6. Availability 25](#_Toc386981480)

[13. Future Enhancements 26](#_Toc386981481)

[Revision History 26](#_Toc386981482)

# Title

Project Code: **PRJ201**

Project Name: **ASAPI v2.1**

System Name: **COMMON APIs**

Component Name: **COMMON APIs**

# Introduction

The initial Mind Tree version of advance search API (hence forth called as version 1.0) has be written to lookup resident details using combination of solar index search. This option is primarily used in Admin Portal when resident EID, ENO, REFID or UID is not known. Exposing this capability of search using partial data fields related to resident and enrolment related data fields help find a lost EID or UID for the resident.

This existing v1.0 workflow shall be enhanced to make the lookup process secure and easily usable:

1. Allow querying against the resident database only if the query parameters are well defined so that the search result DO NOT exceed the policy limit of 19 records.
2. This policy (MAX\_SEARCH\_RESULTS\_ALLOWED\_POLICY) shall NOT be exposed out to the caller and will be internally controlled by the ASAPI as part of a constant variable.
3. ASAPI shall lookup against SOLR indexes first and apply the policy (MAX\_SEARCH\_RESULTS\_ALLOWED\_POLICY) to process the request or return an error code for retry.
4. Ensures 100’s of records are not searched!
5. Expose the existing feature to an API
6. The new Advanced Search API (ASAPI) can be easily integrated by resident’s self service portal, SMS, CRM applications used by Contact Center to help resident find the Lost UID/EID details
7. Additionally the existing logic can be re-designed better to make it more efficient and faster and improve the user experience.

# Purpose

# Target Audience

The relevance to this document is for the below teams, in the order precedence:

1. Architects
2. Developers
3. QA Team
4. Business Team

# Next Step

* Development team create Low Level Design Document (LLD)
* Develop the v2.5 service by re-factoring the existing code to make sure the code is modular and re-usable
* **The Low Level Design Document should create a detailed flow chart with various path based on request/response signatures, data flow and response codes defined in various sections of this document.**
* Implement all the non-functional features listed in this HLD

# Traceability

# Technical Mapping

<< BRD to Technical mapping >>

# Functional Mapping

<< BRD to functional mapping >>

# Non-Functional Mapping

# Design Considerations

## Assumptions

1. ASAPI v2.5 shall work seamlessly and enable admin portal to work more efficiently by limiting the search result output to a policy level controlled number of records.
2. Having this feature exposed as a REST end point shall allow other application to search resident data store in a controlled manner and enable services like finding out Lost EID/UID numbers.
3. Considering the sensitivity of the accessing the resident store, only the application hosted in CIDR shall have access to ASAPI rest end point
4. Applications integrating with ASAP shall make sure the resident information is shared only thorough secure channels like resident’s registered email or mobile number

## Limitations & Constraints

1. Nice to have features have been listed out in “Future Enhancement” section of this HLD.

# System Architecture

## Logical Architecture

Below is very high level code flow that explains the new workflow for v2.5 along with the existing v1.0 code base.

## Recommended Code Deployment

It is recommended that Advance Search API (ASAPI) shall be designed and developed as part of the same RPM deployment of CSAPI and Status Search API.

All the three APIs shall be developed in the same uidenrlsearchserver project scope. Maximum re-use of common validation classes and logic to be leveraged by this setup. This shall help improve the operational maintainability of the code in the long run.



## Impact Analysis

1. ASAPI service deployment should have no impact to the existing client applications using CSAPI.
2. ASAPI will always ensure true backward compatibility of its API versions with every new deployment of the service
3. It is the responsibility of each client application to migrate to the latest version to get additional features (as described in the API signatures) and take advantage of the newer API version.
4. Clients have to make code changes when migrating to ASAPI v2.5 as this latest version changes the response signature to provide additional information

## User Interface

***Not Applicable***

## Physical Architecture

To ensure high availability of the ASAPI service, the following deployment is highly recommended:

1. ASAPI service should be active in both the data center locations (BDC & GNDC)
2. ASAPI being read only service, always point to the read only replica’s of data bases –Mongo available in each data center site
3. All the ASAPI instances should be hosted behind a ***hardware load balancer*** (equivalent to an apache reverse proxy) for high availability at each individual site

## System Diagram

***Not Applicable. Deployment recommendations are defined above***

## Hardware elements

* Deployed on the existing hardware of CSAPI
* 2 CPU, 4 Core machine
* At-least 20GB of RAM

## Software elements

* Linux Operating systems
* Tomcat Application servers
* Apache Reverse Proxy (Software Load Balancer)
* Mongo Database servers
* BI Platform
* NOC Dashboard modules

# Alternative Solutions and criteria for selection

Considering UIDAI main business objective to service the residents/partners with identity, It becomes very important to ensure only the required information is shared to peripheral applications (e.g., resident portals, CRM, IVR, admin portals etc). If a resident has lost the EID/UID, this centralized ASAPI service shall help lookup against the resident store based on partial information search.

Any future changes with respect to schema related changes, sharding, indexing, securing of important columns from tampering or database technologies is all encapsulated by the ASAPI. All the ecosystem client’s implementation and roll out of these features are truly de-coupled.

# Design alternate 1 – Limiting number of search records

SOLR is the UID’s search index platform to query against non ID (i.e EID, ENO, REFID, UID) attributes to find out a resident record based on other personal attributes like name, dob, email, mobile, enrolment agency info. However SOLR manages one record entry for every interaction that a resident would have had with UID.

Considering the core requirement of ASAPI is NOT to reveal too many records to the calling client and forces the calling applications to be specific with respect to quality of the search parameters such that lookup against SOLR results in records that are allowed as per the business policy.

Since SOLR stores all the interactions of a resident, it becomes apparent for the ASAPI to consider only one record that relates to a resident.

For e.g if application queries:

Name = Ram Charan; YOB: 1969; Gender:Male;

There is a possibility that a resident that matched above criteria may have changed addresses 10 times. Also, it is possible, there may be a totally 10 other different residents matching the same above criteria, who may have also have changed address or other attributes multiple times.

To take care of the above scenarios, as part of the ASAPI search it is mandatory to do the following:

1. Query SOLR with max 500 record limit to fetch all the records that matches the search query
2. For each record return from SOLR contains UID, EID and REFID values
3. Group the records based on UID and pick the record with the latest time stamp contained within EID
4. Apply the MAX\_SEARCH\_RESULTS\_ALLOWED\_POLICY on the output of the above unique record set
5. If the record count is greater than policy allowed limit, return 202 error
6. Mongo lookup for successful cases must be fetch only those details mentioned in the ASAPI response signature. Note, Photo is not required as part of ASAPI lookup.

# Design alternate 2

1. **If an UPDATE packet EID is successfully processed then make sure SOLR index is updated to with the latest EID value and any of the changed indexed columns.**
2. **If the update packet is rejected then that EID SHOULD NOT be indexed in SOLR**
3. **This will ensure for a given resident there is always only one SOLR entry and ASAPI do not have to do any filtering and consider multiple UID records in SOLR.**

**This above logic requires boarder change at SEDA end and may be considered as a long term change. This change also may have to be considered with respect to other business values of having indexed all packets in SOLR.**

# Criteria for selection

Design #1 is chosen for this current implementation of ASAPI v2.5, as this does not have any additional major changes at the system level.

After full impact analysis of Design #2, ASAPI implementation should automatically work as SOLR shall return only unique records.

# Solution selected with rationale

***Not Applicable***

# System Details

## Interrelationship and interfaces between the components

***The following classes are to be analyzed and enhanced to make the platform code is truly backward and extensible.***

1. MongoDBHandler

MongoDBHandler is the base class and this code is the gateway to access SOLR/Mongo across UID platform applications.

***getNumRecordsToProcess*** method allows to determine max number of records to read from Mongo or SOLR. This method returns lowest count as per request or collection or handler limit.

***searchFromSolr*** method allows to search from SOLR. If applyMaxSearchPolicy is enabled, then this method should pick only unique Ids before applying this policy against the return record count from SOLR.

1. EnrolmentMongoDBHandler

Should make sure maximum logic is reused from the base MongoDBHandler class.

1. MongoDBMigratorHandler

Should make sure maximum logic is reused from the base MongoDBHandler class.

1. EnrlSearchCriteria

applyMaxSearchPolicy attribute shall be enabled by default. This attribute shall ensure that only those internal application that do require to search without any max limit, like FI batch, DDC, MDD, CSAPI etc can explicitly disable this flag at the time of Mongo/Solr query creation.

1. DemographicSearchCriteria

extends EnrlSearchCriteria and all the calling application searching against Mongo/SOLR uses this.

**SOLR should always search with max read count set to handler limit of 500. This will enable to fetch all the records matching search criteria and there after filter the records based on UID by the calling code.**

**Reading against Mongo shall be based on the outcome of the SOLR output or the min value returned out of *getNumRecordsToProcess* method in MongoDBHandler.**

**TODO: should explore if SOLR query can be written to search only the latest updated record grouped by UIDs**

# Data Model

ASAPI v2.5 refers to SOLR index server and Mongo DB only.

## Database Design

***Not Applicable***

# Application/System Integration - Request/Response Signatures

# ASAPI v2.5 & later Request Signature

The Rest call is differentiated between versions from the existence of version tag (e.g **2.5 etc)** in the URI:

***Example URI***: *http://<commonsearchapi\_server\_ip>:8080/enrlsearch/adv/****2.5***

|  |  |  |
| --- | --- | --- |
| XSD Type | Version | XSD File |
| adv-search-request-v2.5 | 2.5 |  |
| adv-search-response-v2.5 | 2.5 |  |
| searchapi-common-types-v2.5 | 2.5 |  |

| **Option** | **Supported API Version** | **Default Value** | **Description** |
| --- | --- | --- | --- |
| commonsearch/adv/ | 2.5+ |  | Rest endpoint uri |
| appID | 2.5 + | DEFAULT | The **appID** value should map to the pre-defined appIds configured for the ASAPI.   * This value will be used for auditing.   This value will be used in future for any additional filtering of status type to be returned in response. For e.g certain applications would want to service only active residents. |
| txnId | 2.5+ |  | 50 alphanumeric char (with ‘-‘) set by calling client to identify a transaction. Can be set as UUID value format. |
| userId | 2.5+ |  | **Optional**  Max 12 character userId value for auditing.  Application like CRM can decide to provide this information for auditing. |
| Ver | 2.5+ |  | This part of rest URI and payload as well. Determines the rest end point version to call. |
| Lk | 2.5+ |  | **Optional** attribute. This will be used for future use to control api usage by calling client application. |
| Ts | 2.5+ |  | Timestamp set by the calling client in YYYY-MM-DDThh:mm:ss (Time zone is always assumed to be in IST) |
| maxSearchResults | 2.5+ | 5 | Max records to look up against SOLR |
| parameters | 2.5+ |  | Detailed search params to look up against. The detail SOLR fields with operators is listed below. |

**Additional Notes (IMPORTANT):**

1. Advance Search API v2.5 is enabled to gracefully handle only one type of search in a request.

**Below 23 solar indexed attributes can be used to search through ASAPI**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Attribute** | **API Search Key Name** | **Search Type** | **Compare operator to use**  *eq => equals, lt => less than,*  *gt => greater than, nu => null, nn => not null* |
| **1** | *Aadhaar Number* | **uid** | **numeric** | **Eq** |
| **2** | *Enrolment Number* | **enrolmentNumber** | **string** | **Eq** |
| **3** | *Enrolment Date* | **enrolmentDate** | **date** | **eq or lt or gt** |
| **4** | *Resident Name* | **fullName** | **string** | **Eq** |
| **5** | *Resident Address* | **fullAddress** | **string** | **Eq** |
| **6** | *Agency Name* | **agencyName** | **string** | **Eq** |
| **7** | *Registrar Name* | **registrarName** | **string** | **eq** |
| **8** | *State Name* | **stateName** | **string** | **eq** |
| **9** | *District Name* | **districtName** | **string** | **eq** |
| **10** | *Pin Code* | **pincode** | **string** | **eq** |
| **11** | *VTC Name* | **vtcName** | **string** | **eq** |
| **12** | *Operator Name* | **operatorName** | **string** | **eq** |
| **13** | *Supervisor Name* | **supervisorName** | **string** | **eq** |
| **14** | *Introducer Name* | **introducerName** | **string** | **eq** |
| **15** | *Head Of Family* | **hofName** | **string** | **eq** |
| **16** | *Guardian Name* | **relName** | **string** | **eq** |
| **17** | *Client Machine Id* | **clientMachineId** | **string** | **eq** |
| **18** | *Client Version* | **clientVersion** | **string** | **eq** |
| **19** | *Location* | **enrlLocation** | **string** | **eq** |
| **20** | *Year of birth* | **dob** | **date** | **eq or lt or gt** |
| **21** | *Email Id* | **email** | **string** | **eq or nu or nn** |
| **22** | *Mobile Number* | **mobile** | **string** | **eq or nu or nn** |
| **23** | *Gender* | **gender** | **string** | **eq** |

# ASAPI Response XML

The below table describes the ASAPI v2.5 response tags.

|  |  |  |  |
| --- | --- | --- | --- |
| Parent Tag | Tag | Comments | V2.5 |
| *AdvSearchResp* |  | Root element | Y |
| *appID* |  | appID from the request | Y |
| *txnId* | txnID (attribute) | *txnID from the request* | N |
| *Result* | result (attribute) | *y or n* | N |
| *err* | err (attribute) | *Only if result is n* | N |
| respCode |  | New UUID for each request set by the server | Y |
| *ts* | *ts (attribute)* | *Server response time stamp - YYYY-MM-DDThh:mm:ss (timezone default to IST UTC +5.30)* | Y |
| *record.uid* | *Uid* |  | Y |
| *record.eid* | *eid* |  | Y |
| *record.name* | *Name* |  | Y |
| *record.gender* | *Gender* |  | Y |
| *record.yob* | *yob* |  | Y |
| *record.emailID* | *Emailed* |  | Y |
| *records.mobileNO* | *mobileNO* |  | Y |
| *records.communicationAddress* | *communicationAddress* |  | Y |

# ASAPI Response Codes v2.5

ASAPI v2.5 provides detailed error code responses to the calling clients:

|  |  |  |
| --- | --- | --- |
| Error Code | Reason | Comments  (for internal developer understanding/guidance) |
| 201 | No Records Found | No records found. Refine the query and try again |
| 202 | Too Many Records Found | More than allowed limit of records found. Please refine the query and try again |
| 300 | Invalid Request | Request is invalid, Please check the request signature/options and try again. |
| 301 | Invalid txnId | txnId is invalid. Please check and try again |
| 305 | Invalid maxSearchResult | maxSearchResult is not valid. Please check and try again. maxSearchResult should always be less than the MAX\_SEARCH\_RESULTS\_ALLOWED\_POLICY configured as part of PlatformConstants. |
| 306 | Invalid userId | userId is invalid. Check the length and try again. |
| 500 | Technical Exception | Check the logs for detailed technical errors |
| 501 | Internal Data Error | One or more of the mandatory database columns are not available to process the request. Please contact support team. |

***Sample Request***

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

    xmlns:p="[http://www.uidai.gov.in/server/model/adv-search-request/2.5](http://www.uidai.gov.in/server/model/adv-search-request/2.0)"

     xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>">

<p:AdvSearch appID=”CRM” txnID=”ASA20140503112010 userID=”h110071890” ver=”2.5” ts=”2014-05-03T11:20:10”

   <p:parameters>

<p:EntityIdentifier>       <!-- optional tag -- >                                       <p:uniqueId>1</p:uniqueId>

                                </p:EntityIdentifier>

                                <p:selectFields></p:selectFields>

                                <p:parameters>

                                                <p:field>fullName</p:field>

                                                <p:operator>eq</p:operator>

                                                 <p:value>Jyothi</p:value>

                                </p:parameters>

                                <p:detailsFlag>true</p:detailsFlag>

                </p:parameters>

</p:AdvSearch>

**Sample Response**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<p: AdvSearchResp appID=”CRM” txnID=”ASA20140503112010 result=”y” respCode=”asfg3-234ssfsd-23sarrr” ts=”2014-05-03T11:20:10”

<record uid=*"223456789012"* eid=*”1234000001000120140503121212”* communicationAddress=*"12th Main, HSR Layout, Bangalore 560102"* emailId=*"ram@yahoo.com"* gender=*"M"* mobileNo=*"9900990099"* name=*"Ram Charan"* yob=*"1980"* />

<record uid=*"323456789012"* eid=*”2234000001000120140503121212”* communicationAddress=*"20th Main, Palace Road, Delhi 100102"* emailId=*"t1@yahoo.com"* gender=*"M"* mobileNo=*"8800990099"* name=*"Vikas Gupta"* yob=*"1972"* />

</p: AdvSearchResp>

# System Configuration Tokens

For Advance Search API v2.5 and later to consider a new application id, there is no code changes required but expand the ***ADVANCESEARCH.API.RESIDENT.UID.CLIENT.LIST*** token with the new appId and then create a new token with the above format for the new appId.

#CLIENT.LIST helps define the list of client’s apps (ALL IN UPPERCASE) using ADVANCE SEARCH API services.

#Each of the below APPID MUST HAVE statuses TOKENS defined.

**ADVANCESEARCH.API.RESIDENT.UID.CLIENT.LIST=DEFAULT, EKYC, EAADHAAR**

# Re-usable Components

As number of client applications using ASAPI grows, it becomes a challenge to expect all the clients to adapt to a new version on day-1.

It becomes mandatory for ASAPI to support multiple versions at any given point in time, so that the clients are not impacted. This begs a question how to structure the code in ASAPI so that maximum code re-use is done without duplicating common functionality and logic across various versions.

Starting from version 2.5, the code should be structured to have an abstract base class that holds common code and any unique logic that varies between versions will only be implemented in the respective version handlers.

The rest end point must implement either using reflections or delegate patterns to invoke the right version of the API.

Finally, ASAPI (Advance Search API), CSAPI (Common Search API) and SSAPI (Status Search API) shall all be hosted together in the same ***enrlsearchserver*** rpm. Every effort has to be made to call out the common utility and helper classes to the common package that can be re-used across all the three APIs.

*For e.g Validation of EIDs, ENOs, UID and REFID logic can be common re-usable method.*

# High level Security Design Assessment

## Sensitive Data

Considering the sensitivity of the information processed by ASAPI, the service shall be accessible only within the CIDR premises and is not directly exposed to the outside clients. The service is hosted behind the reverse proxy in DMZ location.

**IP Address from the http request header must always be logged across:**

* + 1. **Application Service log file**
    2. **NOC**
    3. **BI events**

**This will enable additional real time and offline monitoring of API usage**

## Internal Interface

***This component is available only for internal applications.***

## External Interface

***Not Applicable***

# Non Functional Requirements

# Automated Testing

Considering the Advanced Search API is restful end point with an xsd contract that is very similar to the existing advance search feature of admin portal, the ASAPI can be tested by simulating the admin portal jsp page using test HTTP Client and compare the html response with the new ASAPI v2.5 XML response.

The automated tool shall be developed with the following objectives:

1. Identify various search parameter combination with sample data
2. Easily configure the ASAPI environment to point to for generating the xml response.
3. Create an Test HTML client as part the same tool to get the admin portal html response
4. For each of the test request, compare the HTML and XML outputs and report any differences into a csv file.
5. This will be done on very similar lines of CSAPI automated tool

# Performance Benchmarking

ASAPI is designed to be stateless and should handle at least 30 M requests over a 10 hour window, on similar lines of CSAPI.

The proposal is to co-host the CSAPI and ASAPI into a same war file and hosts these two APIs on the same tomcat server.

If the number of advance search requests becomes large in number, the applications can be horizontally scalable. Also the existing flow mandates to go either to Mongo or SOLR.

# Logging

ASAPI v2.5 shall handle all the exception scenarios more gracefully and ensure adequate information is available with the appropriate log level.

Appropriate error codes will be enumerated as part of the contract to easily debug the issue.

To ensure resident privacy, No personal information related to resident will be logged. Any sensitive information logged will be hashed with SHA1. AdvSearchException handler should be leveraged to pass on the exceptions with appropriate messages bubble out the restful end point to log the appropriate message. This shall help operations for easy troubleshooting.

Publishing to NOC dashboard with real time updates related to each request server response time, response status code. NOC module internally caches this in memory and publishes to rest end point which will persist the info to the NOC database.

# BI

BI Event has to be published with the following elements:

1. *txnID (Unique ID from request)*
2. *appID*
3. *userId*
4. ***ip-address (from the http request header)***
5. *respCode*
6. *result (y or n)*
7. *err*
8. *request-ts*
9. *response-ts*
10. *response-time*

# NOC Dashboard

NOC event shall be published to the NOC rest end point.

However, consumption of this information is not in scope of this project.

# Availability

ASAPI is highly available service and is hosted between BDC & GNDC locations. Each location hosts multiple ASAPI instances behind a load balancer and reverse proxy.

# Future Enhancements

* + None

# Revision History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **VERSION NO.** | **DATE** | **AUTHOR** | **Description for Change** | **REVIEWED BY** | **APPROVED BY** |
| 1.0 | 05-05-2014 | Vijay Vujjini | HLD with high level implementation details to the class/method level for ASAPI v2.5 |  |  |
|  |  |  |  |  |  |