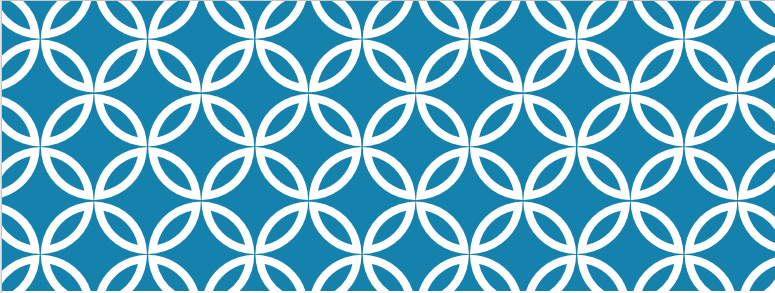
a



MOSIP

Modular Open Source Identity Platform

Version 0.1 | 24 Sep 2018

Registration Context Management - LLD

Copyright Information

This document is the exclusive property of \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; the recipient agrees that they may not copy, transmit, use or disclose the confidential and proprietary information in this document by any means without the expressed and written consent of Mindtree. By accepting a copy, the recipient agrees to adhere to these conditions to the confidentiality of \_\_\_\_\_\_\_\_\_\_\_\_\_ practices and procedures; and to use these documents solely for responding to \_\_\_\_\_\_\_\_\_\_\_\_\_\_ operations methodology.

Revision History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver** | **Change Description** | **Sections** | **Date** | **Author** | **Reviewer** |
| 0.1 | First Draft | All | 20-Sep-18 | Omsaieswar Mulakaluri | Karthik R |
| 0.2 | Second Draft | Pending Section | 21-Sep-2018 | Omsaieswar | Karthik R |
| 0.3 | Third Draft | Table update | 22-Sep-2018 | Omsaieswar | Karthik R |

References

|  |  |  |  |
| --- | --- | --- | --- |
| No | Document Name | Ver. | Location |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Glossary

|  |  |  |
| --- | --- | --- |
| **Terminology** | **Definition** | **Remarks** |
| EO | Enrollment Officer |  |
| ES | Enrollment Supervisor |  |
|  |  |  |

Table of Contents

[Copyright Information 2](#_Toc525795930)

[Revision History 2](#_Toc525795931)

[References 2](#_Toc525795932)

[Glossary 2](#_Toc525795933)

[Table of Contents 3](#_Toc525795934)

[Part A: Background 5](#_Toc525795935)

[1 Introduction 5](#_Toc525795936)

[1.1 Context 5](#_Toc525795937)

[1.2 Purpose of this document 5](#_Toc525795938)

[2 Scope 5](#_Toc525795939)

[2.1 Functional Scope 5](#_Toc525795940)

[2.2 Non Functional Scope 6](#_Toc525795941)

[2.3 Assumption 7](#_Toc525795942)

[2.4 Out of Scope 7](#_Toc525795943)

[3 Technical Approach 7](#_Toc525795944)

[3.1 Design Detail 7](#_Toc525795945)

[3.2 Configuration: 10](#_Toc525795946)

[3.3 Class Diagram {to be updated} 10](#_Toc525795947)

[3.4 Sequence Diagram {to be updated} 10](#_Toc525795948)

[4 Success / Error Code 10](#_Toc525795949)

[5 Audit Information 11](#_Toc525795950)

[6 Dependency Modules 11](#_Toc525795951)

[7 Database - Tables 12](#_Toc525795952)

[8 User Story References 12](#_Toc525795953)

[9 Pending Items / FAQ 12](#_Toc525795954)

Part A: Background

# Introduction

## Context

MOSIP is developed as an open source framework project. The java standard design principles will be followed to design the component.

## Purpose of this document

This document provides the low level technical design approach of a particular functionality in MOSIP Platform. It details out the in depth technical area of a particular scope.

# Scope

## Functional Scope

The Registration client application should create a session and manage it during the span of the enrollment and other process. The entire functionality of this application should work only within the scope of a particular session. The session should be managed for a specific interval [eg: 5 mins] and if the user is ideal and does not perform any operation during this period then alter should be displayed and signed off from the application. While signing off, the entire data inside the object should be cleared off from the memory.

This scope is only valid for ‘Registration Client’ application and not for any other application / product in MOSIP platform.

The detailed requirement of the each context is mentioned below:

* Application Context:
* When the application is launched the related context would be generated and the detail would be stored into that.
* This context would be available for the entire span of the application until it is closed.
* This context object shouldn’t be closed or cleared off when users are login or logoff from the application.
* While creating and closing the context the respective detail should be audited.
* Session Context [LogIn]:
  + Once the user successfully logged in, the session context should be created and can hold required information during the session valid period.
  + User and Enrollment Center related information should also be stored into this object and that will be used all the way through the enrollment process when the session is in valid state.
  + Session Context Details:
    - Session id, Login time, Time of interval, Ideal time, User Context has to be created once the user logged in successfully.
  + User Context Details:
    - User id, User name, Enrolment center, Roles, Authorization context details should be present.
  + The Session context and the intern objects will be cleared off when user clicks ‘logoff’ or if the session time expired.
  + Session Create:
    - The user session should be created when the user successfully logged into the application.
  + Session Clear:
    - The user session should be cleared off when the user log out or inactive state for a particular period.

## Non Functional Scope

* Security :
  + The individual resident detail **shouldn’t be** stored in the session object.
  + Session and user context details to be cleared off when the session becomes invalid.
  + Session should be validated for each delegating calls to the controller.
  + If the user is ideal for some time then system should display the alert for user action and if they don’t perform any action then should automatically signed off and clear everything [all objects] from memory.
* Audit :
* Audit should happen for during applicationcontext creation, session creation and clear off process along with user context detail.
* PI information should not be logged off
  + Each state of the context creation should be stored into the DB for audit purpose.
  + Important detail of the logged in user (password/OTP/Fingerprint) data should not be audited.
* Thread:
  + The background additional thread created when user login to the application to be cleared off when user close the application or log off from the application or the system log off automatically due to the idle time of the user .
* Cache :
  + Data cached are to be cleared off when the user logged off.
* Exception :
  + Any exception occurred during the context creation the same will be reported to the user with the user understandable exception.
* Data History :
  + Maintain the Login time, logout time and other high level info in the database table.
* Configuration:
  + Session time out
  + Time interval for user alert during user ideal time.

## Assumption

* The provided context in this document are used only in the Registration client application.

## Out of Scope

* Using of the provided Context components [Application, Session, User] across the MOSIP platform.

# Technical Approach

## Design Detail

In this section the detailed technical approach of the below contexts are explained:

|  |  |
| --- | --- |
| **Classes** | **Pattern and detail** |
| Application Context | Singleton pattern |
| Session Context | Singleton pattern |
| User Context | As an inner class of Session Context and should not be accessible outside the Session context class. |

**Application Context:**

The detailed technical process for application context creation when user opened the application.

* Create an ‘**ApplicationContext’** business object using singleton pattern and load the required application related information into the respective context.
  + Load the [configuration](#_Configuration:_1) file using
    - **Sample :-**
      * <context:property-placeholder location="{file:database.properties or classpath:db.properties}" />
      * <property name="driverClassName" value="${sqlserver.jdbc.driverClassName}"
  + Shouldn’t have provision to modify the loaded configuration data.
  + Provide getter and setter method to load the application related information.
  + The loaded data should be available during the entire life span of the application.

Create private attributes with the following names to hold the application related information.

* + Application id from configuration file [‘Registration’]
  + Application UUID.
  + Created datetime
  + No. of enrollment created.
  + Client IP address.
  + Mac ID.
  + Connected Device IDs.
  + Load the entire application configuration using spring configuration.
* When the object is initialized, invoke kernel provided ‘AuditManager’ component to insert the audit information into the respective table during context creation and exit.

**Session Context/ User Context:**

The following technical process would be followed when user logging into the Registration application.

* Create a Java component as ‘**Session Context’** with **singleton pattern** method.
* The application Session and User context would be created when the user login to the application.
* **Session context** will have the details of id, User context, Login time, Refreshed Date Time, Timeout Interval and Ideal Time.
* Unique id generated for the session using uuid.
* User level details will be maintained in user context.
* Login time has user logged in time.
* Refreshed Date Time has to be updated for subsequent user calls.
* Timeout interval and Ideal time has to be loaded from config files.
* **User context** is part of session context, it contains the user detail – user id, name, Roles, *EnrollmentCenterDetailDTO*, *AuthorizationDTO*.
* This object should be created inside the ***Session context as inner class*** and ***outside classes should not have*** permission to create it.
  + Id, Name, Role denotes user id, user name and user role.
  + EnrollmentCenterDetailDTO will have the details of geo location, location code, Center registration code, Center code, Center type code.
* This object creation and destroy will be taken care by the SessionContext component.
* While creating and clear off object the detail should be audited.
* Map the data from UserDTO to this context object, which was fetched from DB during login process.
* Functions to be added in the Session Context to support the business functionality.

|  |  |
| --- | --- |
| createSession() | Should be created with current datetime and initiate the map and other attributes. While creating the session object, the detail should be audited. |
| isValid() | Compare the session refresh time and current time. If the time difference more than the session valid time then return false and invalidate the session object. |
| invalidate() | Invalidate the entire session object along with the user context object. While clear off the session object, the detail should be audited. |
| getParam(key) | Get the value stored into the Session map object. |
| getAllParams() | Get all values from Session map object. |
| getUserContext() | Return user context object. |
| addParams(May) | Add the current map value to the existing Map, created inside the Session object. |
| setParam(Map) | Overwrite the existing map with the new Map. |
| addParam(key, value) | Add value into the Session map object. |
| getInstance() | The session object should be created by validating the respective instance object. |
| prepareUserContext(UserDTO) | Load the User detail from database and pass the respective DTO object to this method, which intern fetch the UserContext object from the session and load the respective user detail. If the usercontext already available then it won’t overwrite the value. |

**Session Validation**

This should be implemented during the following cases.

|  |  |
| --- | --- |
| **Scenario** | **Technical approach** |
| User Moving across the screen. | While moving across the controller object, super.validateSession() to invoked to validate the session object.  BaseController.validateSession() – this function will validate the session against the SessionContext object. If session not valid then route the control to the logout page. |
| When user performs some operation [Mouse event, key event] in the screen. | Implement the EventManager in BaseController and handle() method to capture all the events happening at ‘Stage’ level component in the JavaFX component hierarchy. When any events happening in the screens, as per the JavaFX component design, it would notify this EventManager. Inside the EventManager .handle() method refresh the session ‘refreshTime’ attribute with the current time. So the ‘*SessionValidateSchedularUtil*’ will validate session based on the ‘refreshTime’. |
| System ideal for some time. | *‘SessionValidateSchedularUtil*’ – Implement this component to validate the user ideal time using SessionContext.‘refreshTime’ against the current time. This is a thread based utility [java.util.TimerTask] class which runs in the configured interval to validate the session. This would invoke the callback() function to validate he session in a particular interval. It validates, before user breaches the threshold time [which little earlier than actual session timeout] and display the alert. |

**Capture stage based event and refresh the Session:**

**public** **class** BaseController {

**private** Stage stage = **new** Stage();

**protected** Stage getStage() {

EventHandler<Event> event = **new** EventHandler<Event>() {

@Override

**public** **void** handle(Event event) {

SchedulerUtil.*startTime* = System.*currentTimeMillis*();

}

};

stage.addEventHandler(EventType.***ROOT***, event);

**return** stage;

}

}

## Configuration:

Load the configuration from property file {mosip\_registration.properties} during application startup.

* + APP\_ID > {MOSIP\_REGISTRATION} application id.
  + APP\_SES\_TIME\_OUT > Session time out
  + APP\_SES\_TIME\_OUT\_UNIT > {minute, seconds}
  + SES\_REM\_INT\_TIME > Time interval for displaying user alert during user ideal time.

## Class Diagram

**Create and Invalidate context:**

[**https://github.com/mosip/mosip/blob/DEV/design/registration/\_images/\_class\_diagram/registration-contextmanage-classDiagram.png**](https://github.com/mosip/mosip/blob/DEV/design/registration/_images/_class_diagram/registration-contextmanage-classDiagram.png)

## Sequence Diagram

**Create and Invalidate context:**

[**https://github.com/mosip/mosip/blob/DEV/design/registration/\_images/\_sequence\_diagram/registration-contextmanage-sequenceDiagram.png**](https://github.com/mosip/mosip/blob/DEV/design/registration/_images/_sequence_diagram/registration-contextmanage-sequenceDiagram.png)

# Success / Error Code

While creating the session and exit from the session the respective Session and User context would be created and cleared off.

|  |  |  |
| --- | --- | --- |
| **Code** | **Type** | **Message** |
| REG-LGO-001 | Success | Session successfully created. |
| REG-LGO-002 | Success | Session successfully cleared off. |
| REG-LGO-003 | Error | Unable to initialize the session, due to database connection issue. |
|  |  |  |

# Audit Information

While Creating or Closing Application Context, the following parameter should be inserted into the Audit table. ->

1. Application id [UUID]
2. Client machine IP address.
3. Client id.
4. Station id.
5. Agent id.
6. Enrollment center id.
7. Application opened date time
8. Application closed date time.

While Creating or Closing Session Context, the following parameter should also be inserted Audit table. ->

1. Session id [UUID]
2. User id
3. Client machine IP address.
4. Client id.
5. Station id.
6. Agent id.
7. Enrollment center id.
8. Session created date time
9. Session expiry date time.
10. Module id [as ‘Session’]

# Dependency Modules

|  |  |  |
| --- | --- | --- |
| **Component Name** | **Module Name** | **Description** |
| Audit Manager | Kernel | To audit the process while creating the packet. |
| Exception Manager | Kernel | To prepare the user defined exception and render to the user. |
| Log | Kernel | To log the process. |
|  |  |  |

# Database - Tables

* Audit Table.

# User Story References

|  |  |  |
| --- | --- | --- |
| **Sprint** | **User Story No.** | **Reference Link** |
| **3** | MOS-553 | <https://mosipid.atlassian.net/browse/MOS-553> |
|  |  |  |

# Pending Items / FAQ

* Need to check the components from Kernel application.
* Need to check with BA, whether any time out required at Application level.
* Need to check with BA, whether any time out required at each enrollment level.