VISAP-Authorization

Vision and high-level flow

Table of Contents

[Revision History 2](#_Toc412535027)

[1 Introduction 3](#_Toc412535028)

[1.1 Objective 3](#_Toc412535029)

[1.2 Purpose 3](#_Toc412535030)

[1.3 Scope 3](#_Toc412535031)

[2 Executive Summary 3](#_Toc412535032)

[2.1 Key Decisions 3](#_Toc412535033)

[2.2 Core Services 3](#_Toc412535034)

[2.3 High level Workflow 3](#_Toc412535035)

[3 Technical Design and Constraints 7](#_Toc412535037)

[3.1 Design Decisions and Technical Constraints 7](#_Toc412535038)

[4 Features and Services 10](#_Toc412535039)

[4.1 List of Services 10](#_Toc412535040)

[5 Architecture 13](#_Toc412535042)

[5.1 Architecture Diagram 14](#_Toc412535043)

[6 Risks and Next Steps 15](#_Toc412535047)

[6.1 Risks 15](#_Toc412535048)

[6.2 Next Steps 15](#_Toc412535049)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| 1.0 | 11-Mar-2015 | Bharathi | Reviewed by Tejaswini |

# Introduction

This document provides high-level details for Authentication and Authorization

This document outlines the Design prospective Authentication.

## Objective

The objective of his document is to provide

1. Design details of the Authentication
2. Any Other system Integration with ViSAP, document need to explain about the mechanism of Authentication and authorization.
3. Defining the Roles
4. Mapping between the roles and users
5. Privilege for the roles.

## Purpose

The purpose of this document is to have the research work of deciding about the

Implementation structure serving the above mentioned objective.

The scope of this document is -

1. To provide high-level Implementation details and flow.
2. To provide an overview how Authentication is taken care with external systems.
3. To provide details of database
4. To provide a workflow to reach Home page of the ViSAP

## Key Decisions

SARAS will be the consumer for ViSAP

## Core Services

ViSAP Authentication Service.

## High level Workflow

**Existing Implementation:**

**Current Implementation requires username to be passed from client script**

**GET**

* **Passed as header parameter**

**POST**

* **Passed as request parameter**

Username should not be passed as parameter from the Client side

**Assumptions**

1. SSOURL -will be Saras.com
2. Token will be excepted from the External SSOURL
3. Authentication Module starts from ViSAP.Init method of Visap.js
4. Taking SARAS as External client

**Sequence Diagram:**



# Technical Design and Constraints

This chapter focuses on the design constraints and how the ViSAP service will be used within the system.

## Design Decisions and Technical Constraints

**Authentication**

ViSAP Script init will have Authenticate Mode saying whether it is Widget /Embedded

**Case1: Authentication**

Mode:Embedded

1. An ajax call to the external service to get the Token
2. With that Token ViSAP service is called for Authenticating the Token
3. Giving the Token to external service username is got
4. Username is encrypted and along with the expiry, its added to Visap HttpCookie which the
5. Client is going to use as well as service whenever username is required.

SARAS.com

External Service

Token

ViSAP Service

Username/UserId

GetSSOToken()

Cookie

ValidateSSOToken(Token)

**ViSAP**

Visap.init()

**GetSSOToken()**

Method called from Script(Visap.js)

* If the mode of Authehtication is Embedded then
* Call GetSSOToken an ajax call to SARAS service
* Returns the Token
* Call ValidaeSSOToken(Token) method from ViSAP Service

**ValidateSSOToken(Token)** method in service will validate the Token

Steps to validate the token

* Call the External service to verify the Token with the ReturnURL
* Once the token is verified username will be sent back
* Check for the username
* Encrypt the username with expiry
* Add to Http cookie

**Case2: Authentication**

Mode:Widget

* External system access ViSAP Application
* From Saras.com ViSAP Init’s method is called
* From ViSAP Init Authenticate method is called from service.

**Authenticate()**

Steps in above method

* Call the SSOURL with ReturnURL in querystring

For Example

SSOUrl=http://Saras.com

ReturnUrl=http://Visap.com

* SSOToken is excepted as part of querystring
* SSOToken is again sent to SSOUrl

[http://Saras.com/ReturnUrl=Visap.com?SSOToken=”xyz](http://Saras.com/ReturnUrl=Visap.com?SSOToken=)”

ReturnUrl

Visap Service

Saras Service

SSOToken

SSOToken

Username

* Username is encrypted and Expiry is set
* Set to ViSAPCookie(HttpCookie) username
* All other services will read username from Cookie
* Will be sent back to scripts
* Delete the Cookie based on Expiry

**Expiry:**

* Decrypt the cookie
* Check for the Time
* If time is currenttime –cookietime Expirytime(added while adding to cookie)
* If it is less then 20 mins
* Username is Invalid
* Authentication process is done again

# Features and Services

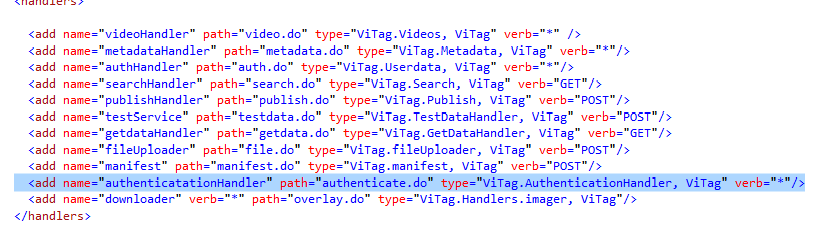
## List of Services

# Implementation schema

Add new service as shown below

# 

Configure the service in web.config file



Call the ViSAP Authentication service from Visap.Init

(function (ns) {

//#region Public

ns.init = function (args, Mode) {

if(authenticate.mode==”Embedded”)

{

//**var ssoToken=GetSSOToken();**

**// ValidateSSOToken(ssoToken);**

}

else

{

// **Authenticate();**

}

}

Below methods will be in authenticate namespace in ViSAP.js

**GetSSOToken**: function ( fnPreSend, fnSuccess, fnError) {

$.ajax({

url: “ssourl”,

type: "POST",

success: fnSuccess,

error: fnError,

});

},

**ValidateSSOToken**: function ( token,fnPreSend, fnSuccess, fnError) {

$.ajax({

url: “authenticate.do?ssoToken=”+token,

type: "POST",

success: fnSuccess,

error: fnError,

});

},

**Authenticate**: function (fnPreSend, fnSuccess, fnError) {

$.ajax({

url: “authenticate.do”,

type: "POST",

success: fnSuccess,

error: fnError,

});

},

# Implementation schema in AUTHENTICATION service

Service authentication.cs

public class AuthenticationHandler : IHttpHandler

{

//use Authentication class and its members

}

Class in service layer

namespace Excel.Visap.Security

{

public class Authentication

{

public void Authenticate()

{

//check if cookie is present

//else GetSSOToken

}

private string GetSSOToken()

{

//call to SSOUrl with ReturnUrl pointing to Visap

//actually coding need to know what would be this

//saras will provide ssourl

//expected would be token

}

private string ValidateSSOToken(Token)

{

//call SSOURL sending SSOToken as Querystring parameter

//and returnurl pointing to visap.com

//based on token username is returned from SARAS

}

private string EncryptUserName(string username)

{

//encrypt username

// Encrypt = “username” | “12/03/2015 02:00:00”

//Add 20 minutes to current time

}

private void SetCookie(string username)

{

//add encrypted value to Httpcookie with expiry

}

private bool CheckIfCookieisPresent()

{

//check in the current context where cookie is present

}

private bool CheckforCookieExpiration()

{

// Verify time – current – encryptime <= 20 mins

}

private void DecryptUsername()

{

//if(Cookie has username)

{

//Split (“|”)

If(CheckforCookieExpiration())

{

//Then username is valid use them

}

Else{

Remove.cookie()

Authenticate()

}

}

}

private void RemoveCookie()

{

//remove the cookie after checking for the expiration

}

}

# Architecture

This section describes a high level architecture of the proposed product.

## Architecture Diagram

**Single sign on Solution**

**Widget:**

**Authentication service of SARAS**

**ViSAP Service**

SSOURL

Token (ReturnUrl)

Token(ReturnUrl UserName

Token(ReturnUrl

Cookie Authenticate

**SARAS App**

**ViSAP App**

**SARA**

**Embeded:**

SARAS.com

External Service

Token

ViSAP Service

Username/UserId

GetSSOToken()

Cookie

ValidateSSOToken(Token)

**ViSAP**

ViSAP Init()

## Risks

No Risks

## Next Steps

Exploring with SARAS for Service

Authrorization –Mapping user and roles