ROYAL CYBER
Modernized e-business solutions


# Single Sign-On in PHP & HATS Applications using Hashed Cookies

Royal Cyber Inc.
Modernized e-business solutions

## Overview

This white paper focuses on how to achieve Single Sign-On functionality between PHP-WordPress based applications and the HATS applications using hashed cookies. The use case scenario is:

- Customer has a PHP-WordPress based website where the users perform Login operation using LDAP credentials. From there they are provided with a link to HATS application.

The goal is when the user clicks on the provided HATS application link; he/she should automatically be signed-in into HATS without providing any additional credentials or any other information. This can be achieved using hashed cookies method.

## Part 1 - What are Cookies and What is Hashing

In this part a brief overview of Cookies will be given. Furthermore, the basic concept of Hashing will also be explained.

## Part 2 - Steps to Achieve SSO in IBM Rational HATS

In this part, the basic steps required to achieve Single Sign-on functionality are discussed.

## Part 3 - Cookies Management and Performing Login in IBM Rational HATS

In the final part cookie management for performing login into IBM Rational HATS is explained. The workflow to achieve this functionality is also demonstrated graphically.

## PART 1- WHAT ARE COOKIES AND WHAT IS HASHING

### Cookies

Cookies are small files which are stored on a user's computer. They are designed to hold a modest amount of data specific to a particular client and website, and can be accessed either by the web server or the client computer. This allows the server to deliver a page tailored to a particular user, or the page itself can contain some script which is aware of the data in the cookie and so is able to carry information from one visit to the website (or related site) to the next.

### Hashing

Hashing is a one way process of encoding message (or information) so that eavesdroppers or hackers cannot decode original message. In hashing scheme, the message or information is hashed using a hashing algorithm, turning it into un-decrypted cipher text. The hashing algorithms only allow encrypting text; they don't decrypt encrypted text, so any adversary that can see the cipher text should not be able to determine anything about the original message. An authorized party however is able to generate same hash value because they will know about hashing algorithm and format of value that is hashed. For our purpose in this article we are using Secure Hash Algorithm (SHA512).

## PART 2 - STEPS TO ACHIEVE SSO IN IBM RATIONAL HATS

### Set the Cookies

Given the particular scenario, customer will create/set two cookies when the user logs in to their portal, one cookie contains the username and the other contains hash code generated by SHA512 algorithm.

### Get the Cookies

When the user clicks on HATS application link, we get both the cookies by using JSP code and we reset the lifetime of the cookies by using JavaScript (as shown below).

```
//function to reset the time of cookie
   function resetCookie() {

           var now = new Date();
           var time = now.getTime();
           time += 1800 * 1000;
           now.setTime(time);
           document.cookie = 'VPCL_uid=' + "<%=newMyCookie%>" + '; expires=' + now.toGMTString() + "; path=/";
           document.cookie = 'VPCL_hash=' + "<%=encryptedCookie%>" + '; expires=' + now.toGMTString() + "; path=/";

       }

// function to redirect the page when cookie expired

  function cookieExpire(){

            window.location.assign("http://www.corona2.vpcl.on.ca");

  }
```

Figure 1 Jave Script Code

## PART 3 - COOKIES MANAGEMENT AND PERFORMING LOGIN IN IBM RATIONAL HATS (SSO)

By using SHA512 algorithm, we generate hash code by appending username (which we get from username cookie) with a salt value. After that, we compare the hash code that we are getting from hash code cookie and the new generated hash code on username. If these both hash codes are equal, it means the username is correct as per it hash code. From there we run a macro to get that user login into HATS application successfully. If the two hash codes don't match, page is be redirected to PHP application with error message.

```
// function to press enter on sign on
  function gotoMainMenu(){

            javascript:ms('[enter]','HATSForm');

  }
```

Figure 2 Macro / Event File Snap

The maximum time for the expiration of cookies is 30 minutes, so when user leaves the HATS application for 30 minutes without interaction, the cookies will be expired automatically and system will redirect the page to PHP application.

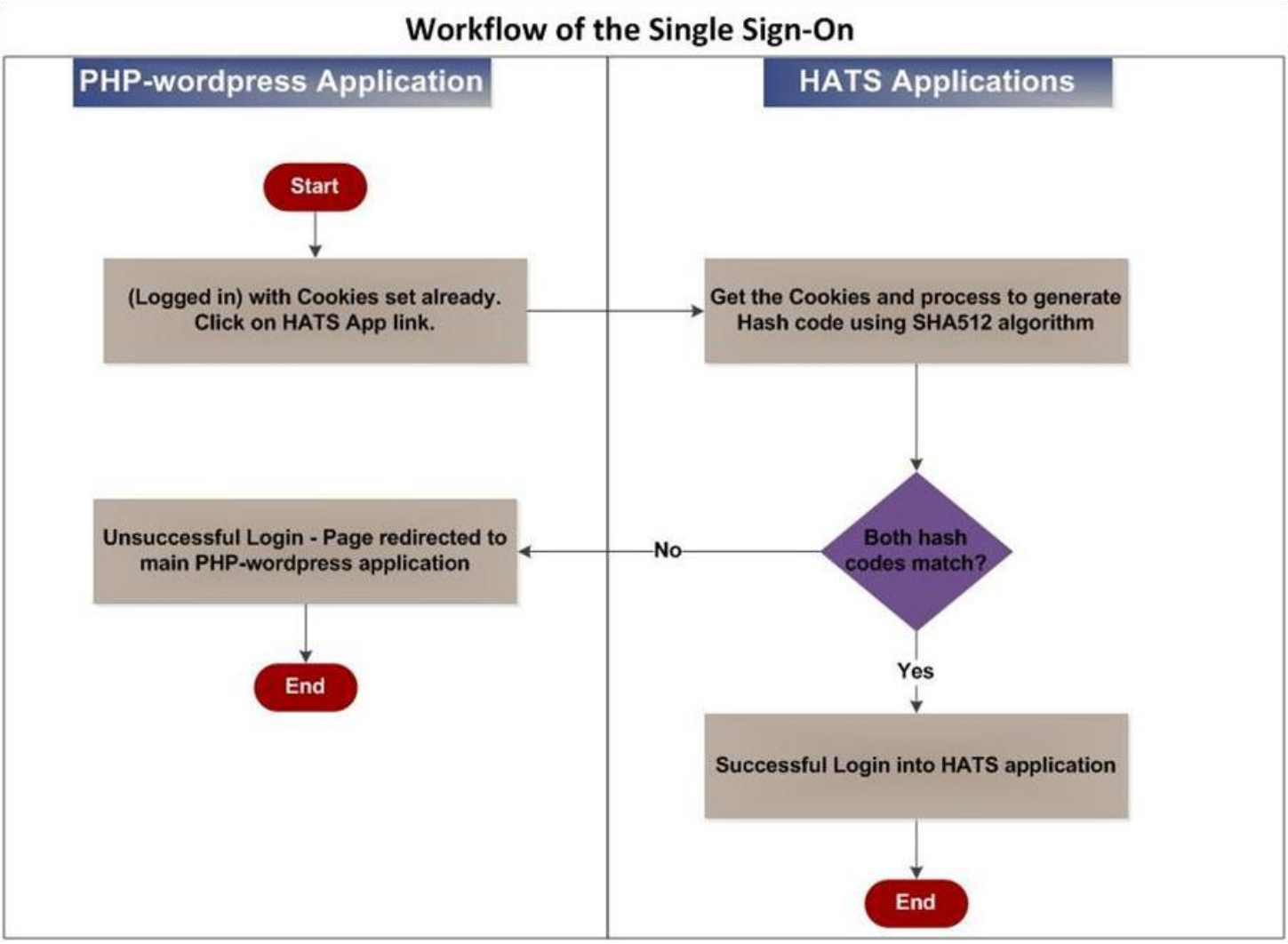


Figure 3 Single Sign-On Workflow