

**Century Salesforce Gateway**

Single Sign-On Integration

**Legal Disclaimer**

This document contains unpublished, confidential and proprietary information of Excelacom Inc. that is protected by copyright and other intellectual property laws. Excelacom Inc. software and its related documentation are subject to and made available only pursuant to the terms of a license agreement with Excelacom Inc. and may be used or copied only in accordance with the terms of that agreement.

Any use, copying, reproduction, alteration, transmission, or translation of these materials, in whole or in part, in any form or by any means, is strictly prohibited without the prior written permission of Excelacom Inc.

If this material is provided with software licensed by Excelacom Inc. the information herein is provided subject to the terms of the warranty provided with the product license. If this material is not provided with licensed software, the information herein is provided "as is" without warranty of any kind. In either case, there are no other warranties, either expresses or implied, including, but not limited to the implied warranties of merchantability, fitness for a particular purpose, non-infringement, or quality. In no event shall Excelacom Inc. or any of its affiliates be liable for any direct or indirect, special, incidental, consequential, punitive, or exemplary damages arising out of the use or inability to use the materials and/or information contained herein. Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to you.

Excelacom Inc. takes reasonable measures to ensure the quality of the data and other information produced by Excelacom Inc. that is made available herein. However, these materials may contain technical inaccuracies or typographical errors, and are not guaranteed to be error-free. Information may be changed or updated without notice. Excelacom Inc. has no obligation to update the information in this material based on changes to its products or services or those of third parties. Excelacom Inc. may also make improvements or changes to the products or services described in this information at any time without notice.

**Table of Contents**

[1. Objective 4](#_Toc420956439)

[2. Revision History 4](#_Toc420956440)

[3. Scope 4](#_Toc420956441)

[4. High Level Architecture 4](#_Toc420956442)

[5. Detailed Approach 5](#_Toc420956443)

[6. Sequence Flow 6](#_Toc420956444)

[7. Process Flow Diagram 7](#_Toc420956445)

[8. Technical Overview 8](#_Toc420956446)

[8.1 Display Salesforce Main Screen by Delegated Authentication 8](#_Toc420956447)

[8.1.1 Approach. 8](#_Toc420956448)

[8.1.2 Implementation Changes 8](#_Toc420956449)

[9. Acronyms and Glossary 13](#_Toc420956450)

# Objective

This document provide the approach for Single Sign-On Integration for Century with Salesforce by Delegated Authentication in Salesforce.

# Revision History

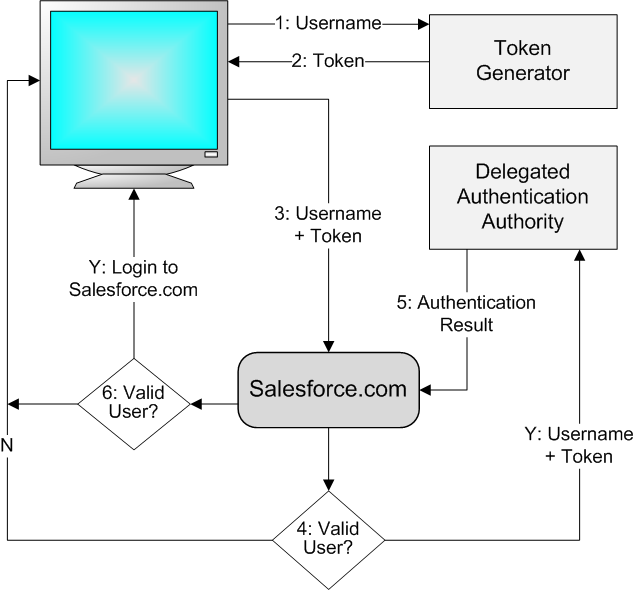
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Revision History | | | | | |
| Version | **Date** | **Reason for Change** | **Prepared By** | **Reviewed By** | **Approved By** |
| 0.1 | 1/6/2015 | Single Sign-On Integration | VijaiKumar | TamilKumaran |  |
|  |  |  |  |  |  |

# Scope

The Scope of these document which includes

* Show the Century Login Screen
* Generate the Token and Perform the Verification for Token.
* Authenticate the Logged In User from Salesforce
* Redirect to Salesforce Main Screen.

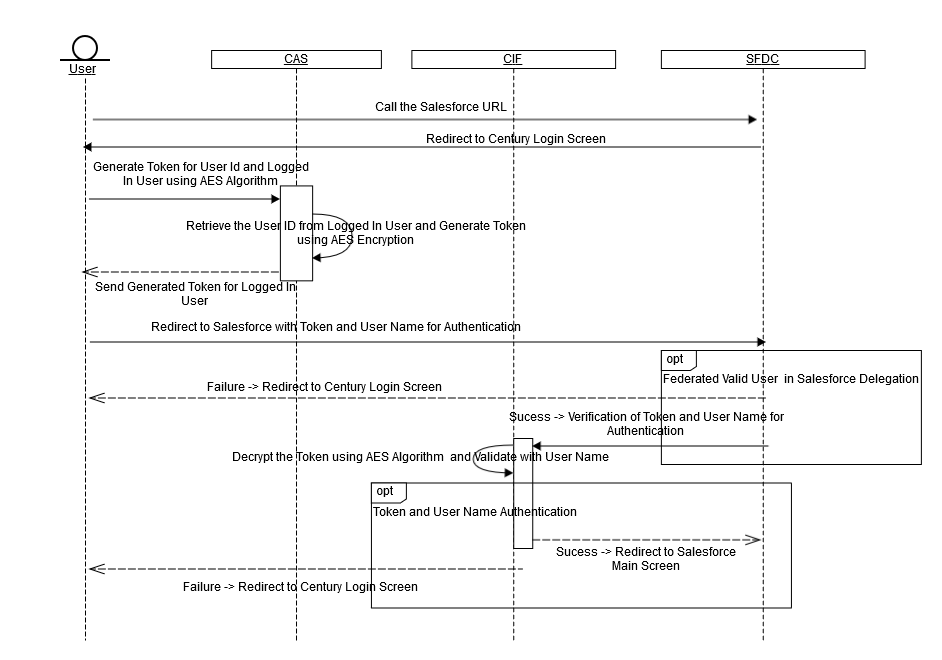
# High Level Architecture



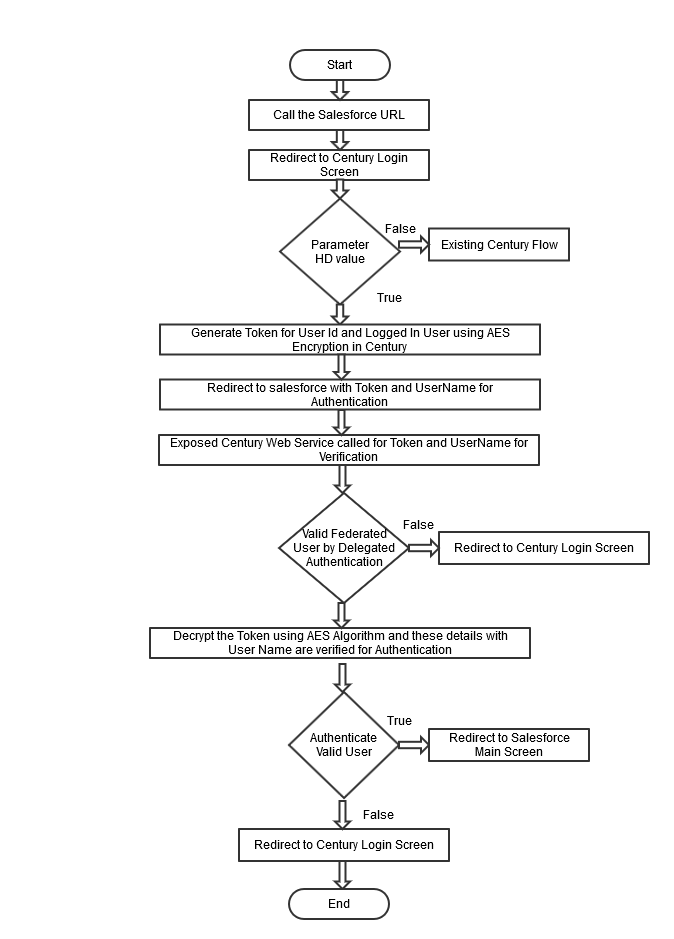
# Detailed Approach

* Call the Salesforce URL and redirect to Century Login Screen.
* Logged in with Century CAS Login Screen which have the parameter “hd=true”. These Login UI Screen will be changed based on our user design.
* If parameter “hd=false”, then existing Century Flow logic will be executed.
* If parameter “hd=true”, then Salesforce Redirection Flow logic will be executed.
* If “hd=true”, then Token will be generated in century and send to salesforce for verification.
* Token will be generated with the combination of **User Id** (Unique Sequence Number for each user – eg. 11230) and  **Logged User Id** (eg.custpmauto) by using **AES Algorithm** for Encryption.
* User Name and Generated Token in century will be send for salesforce page redirection.
* Read the Username and Token and it send these information to Century Exposed Web Service by Delegated Configuration provided in salesforce.
* Exposed Web Service will be configured in Salesforce Single Sign-On Setting.
* Logged User Id will be configured in the Federated ID in salesforce.
* Exposed Web service will be decrypt the Token using **AES Algorithm** and these details with User Name will be verified for authentication.
* If “authenticate=true”, then it redirect to salesforce Main Screen.
* If “authenticate=false”, then it redirect to Century-Salesforce Login Screen.

# Sequence Flow



# Process Flow Diagram



# Technical Overview

## Display Salesforce Main Screen by Delegated Authentication

* Redirect to Salesforce Main Screen from Century by Delegated Authentication

### Approach.

* Call the Salesforce URL and redirect to Century Login Screen.
* Parameter are passed as “**hd=true** “by calling salesforce.jsp page directly from browser.
* Redirected to default CAS login page.
* Create token with User ID and Logged User ID using AES Algorithm Encryption and send to sales force with username and Token.
* Verified the token using AES Algorithm Decryption and details with username from Salesforce.
* Authenticated is true and redirect to salesforce Main Screen.
* Authenticated is false and redirect to Century Login Screen.

### Implementation Changes

#### Salesforce.jsp [CAS- cas-server-webapp]

* Initiate call to display the CAS Login Screen

**Code Changes are below**

|  |
| --- |
| <%@ page language="java" session="false" %>  <%  final String queryString = request.getQueryString();  //request.setAttribue("message","user.success.mailsent");  String url = request.getContextPath() + "/login" + (queryString != null ? "?" + queryString : "");  if(url.indexOf("?")!= -1)  {  url += "&hd=true";  }  else  {  url += "?hd=true";  }  response.sendRedirect(response.encodeURL(url));%> |

#### InitialFlowSetupAction.java [CAS-cas-server-core]

* Setting Salesforce redirected URL
* Based on the HD parameter, redirect the URL to Salesforce.

**Code Changes are below**

|  |
| --- |
| // Added for External –CRM Integration – Starts  private String salesRedirecturl;  /\*\*  \* @return the salesRedirecturl  \*/  public String getSalesRedirecturl() {  return salesRedirecturl;  }  /\*\*  \* @param salesRedirecturl the salesRedirecturl to set  \*/  public void setSalesRedirecturl(String salesRedirecturl) {  this.salesRedirecturl = salesRedirecturl;  }  if(Boolean.parseBoolean(request.getParameter("hd")))  {  context.getFlowScope().put("hd", true);  request.getSession().setAttribute("baseLoginUrl", getSalesRedirecturl()+"/login.jsp");  redirecturl = request.getContextPath() + "/PostTokenResultForm.jsp";  }  // Added for External –CRM Integration – Ends |

#### UserController.java [CAS-cas-server-login]

* Getting UserID based on the userloginID.
* Generate the token for the combination of userID with userLoginID.

**Code Changes are below**

|  |
| --- |
| // Added for External –CRM Integration – Starts  String salesUserIDToken = getUserImpl().getSalesUserIDToken(userBean.getUserLoginId());    session.setAttribute("salesToken",salesUserIDToken);  // Added for External –CRM Integration – Ends |

#### User.java [CAS-cas-server-login]

* Declare the User Login ID.

**Code Changes are below**

|  |
| --- |
| // Added for External –CRM Integration – Starts  /\*\*  \* Gets the user ID as Token.  \* @param UserLoginID  \* @return userIDToken  \*/  public String getSalesUserIDToken(final String szUserLoginID);  // Added for External –CRM Integration – Ends |

#### UserImpl.java [CAS-cas-server-login]

* Return the Generated Token using the User Login Id.

**Code Changes are below**

|  |
| --- |
| // Added for External –CRM Integration – Starts    /\*\*  \* Gets the user ID as Token.  \* @param UserLoginID  \* @return userIDToken  \*/  public final String getSalesUserIDToken(final String szUserLoginID) {  return getUserServiceImpl().getSalesUserIDToken(szUserLoginID);  }  // Added for External –CRM Integration – Ends |

#### UserService.java [CAS-cas-server-login]

* Get the User ID Token.

**Code Changes are below**

|  |
| --- |
| // Added for External –CRM Integration – Starts  /\*\*  \* Gets the user ID as Token.  \* @param UserLoginID  \* @return userIDToken  \*/  public String getSalesUserIDToken(final String szUserLoginID);  // Added for External –CRM Integration – Ends |

#### UserServiceImpl.java [CAS-cas-server-login]

* Return the Generated Token using the User Login Id.

**Code Changes are below**

|  |
| --- |
| /\* Added for External –CRM Integration – Starts \*/  /\*\*  \* Gets the user ID as Token.  \* @param UserLoginID  \* @return userIDToken  \*/  public final String getSalesUserIDToken(final String szUserLoginID) {  return getUserDAOImpl().getSalesUserIDToken(szUserLoginID);  }  /\* Added for External –CRM Integration – Ends \*/ |

#### UserDAO.java [CAS-cas-server-login]

* Declare the User Id Token.

**Code Changes are below**

|  |
| --- |
| /\* Added for External –CRM Integration – Starts \*/    /\*\*  \* Gets the user ID as Token.  \* @param UserLoginID  \* @return userIDToken  \*/  public String getSalesUserIDToken(final String szUserLoginID);  /\* Added for External –CRM Integration – Ends \*/ |

#### UserDAOImpl.java[CAS-cas-server-login]

* + Get the Token using encryption for userloginID.

**Code Changes are below**

|  |
| --- |
| /\* Added for External –CRM Integration – Starts \*/    /\*\*  \* Gets the user ID as Token.  \* @param UserLoginID  \* @return userIDToken  \*/  public final String getSalesUserIDToken(final String szUserLoginID) {    String userID = getuserID(szUserLoginID);  System.out.println("szUserLoginID :::: " +szUserLoginID);  System.out.println("userID :::: "+userID);  String enct = getEncryptionImpl().encryptString(szUserLoginID+"-"+userID);    System.out.println("enct :::: " +enct);    return enct;  }    /\*\*  /\* Added for External –CRM Integration – Ends \*/ |

#### cas-servlet.xml [CAS- cas-server-webapp]

* Set the property value for salesforce redirection URL.

**Code Changes are below**

|  |
| --- |
| <!-- Added for External –CRM Integration – Starts - ->    <property name="salesRedirecturl" value="${salesRedirecturl}"/>    <!-- Added for External –CRM Integration – Ends - -> |

#### PostTokenResultForm.jsp [CAS- cas-server-webapp]

* Submit to sales force with username and generated token to salesforce redirection.

**Code Changes are below**

|  |
| --- |
| // Added for External –CRM Integration – Starts  <%@ page contentType="text/html; charset=UTF-8" %>  <html>  <head>  <s:include value="%{getText('app.includes.head')}" />  <style>  #tokenInfo th, #tokenInfo td{  #tokenInfo th, #tokenInfo td{  padding-bottom: 10pt;  }  #tokenInfo th{  font-weight: bold;  text-align: right;  padding-right: 10px;  }  </style>  <SCRIPT type="text/javascript">  window.name = "tokenMain";  function init(){  document.getElementById("delegatePoster").submit(); }  </SCRIPT></head>  <body onload="init();"> <p> redirecting.... </p>  <form action="${baseLoginUrl}" method="POST" id="delegatePoster" style="display: none;">  <table id="tokenInfo">  <tr><th>&nbsp;</th><td><strong style="font-size: 12pt;">POST Login Request</strong></td></tr>  <tr><th></th><td><strong>${baseLoginUrl}</strong></td></tr>  <tr><th><s:text name="label.delauth.username"/></th><td><input type="text" name="un" value="${accuarate\_name}" size="40"/></td></tr>  <tr><th><s:text name="label.delauth.tokenPassword"/></th><td><input type="text" name="pw" value="${salesToken}" size="40"/></td></tr>  <tr><th></th><td><input type="submit" value="<s:text name="label.login"/>"/></td></tr>  </table> </form> </body></html>  // Added for External –CRM Integration – Ends |

#### Salesforce Configuration Setting

* Exposed the Century Web service will be configured in Single Sign-on Configuration Setting.
* Federated ID will be configured with Century User Logged ID.

#### SalesAuthentication [Hotdeployment].

* Newly Added service and deployed in service Mix Hot deployment
* Web service for retrieve the Username and token from Salesforce.
* Authenticated the Token and User Name for sales force Redirection.



#### sales-sso-authentication [ServiceMix].

* Newly added service and deployed in Service Mix War file.
* Decrypt the token and check the Logged User Id and Token.
* Authenticated the Token and User Name for sales force Redirection.



# Acronyms and Glossary

|  |  |
| --- | --- |
| Term/Acronym | Definition |
| CM | Customer Manager |
| CSO | Century Service Orchestrator |
| UI | User Interface |