Intellicus Single Sign-on

Version: 7.3

intellicus

Power To Understand Your Business

Dated: August 2015

Acknowledgements

Intellicus acknowledges using of third-party libraries to extend support to the functionalities that they provide.

For details, visit: http://www.intellicus.com/acknowledgements.htm

Contents

1 Introduction

Single sign-on refers to one time authentication performed by the host application. Users accessing Intellicus from within a host application are already authenticated. Intellicus does not perform authentication check for such users.

This means that User can access Intellicus without going through the Intellicus Login page. Host application would require passing user credentials of currently logged-in user (in host application) to Intellicus.

In addition to the user credentials, host application can also pass business parameters that could be used in the reports for data filtering based on the user context.

Single Sign-On is briefly referred as SSO.

Note: Single Sign-On is required only when Host application and Intellicus are running as two separate web applications on the same or different web servers.

Single Sign-On is not required when Intellicus is embedded inside the Host application.

Single Sign-On Requests

Host application's users can access Intellicus reporting features integrated in their application. Reporting features includes Report listing, Report execution, User preferences, Adhoc wizard etc.

These features can be accessed either inside an *Iframe* or in a new window.

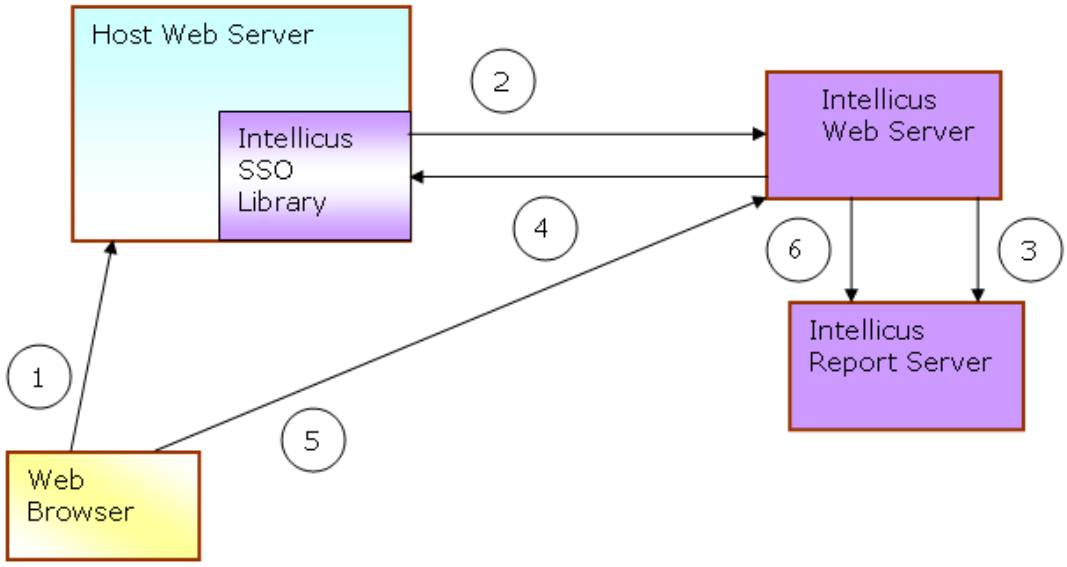
So end-users of host application can request all these reporting features and also pass business/request parameters to Intellicus.

Host application would require following a sequence of steps in order to achieve SSO.

In order to access reporting features of Intellicus, a user space should exist at Intellicus for each user of host application. User space at Intellicus can be created manually through Intellicus web portal. It can also be created dynamically using Intellicus APIs.

Note: Please refer Chapter 5 of this document for settings required for dynamic user creation.

Single Sign-On Flow:



Steps:

1. In Web browser, Host Application user requests for reporting.
2. Using Intellicus SSO Libraries, host application requests Token from Intellicus. With this request Host application sends user credentials and other business parameters.
3. Intellicus Web server send request to report server for User space Identification.
4. If user space exists at Intellicus, intellicus token is returned to host application web server.
5. From client browser, reporting request is sent to Intellicus web server with intellicus token and the relative URL for Intellicus HTTP API.
6. Intellicus receive the token and use it for user identification.
7. Intellicus creates user session and serve the reporting request to Host application user.

2 SSO for Host Application on Java Platform

Configurations Required

Configuring Host Application

In order to invoke methods at Intellicus end, the file intellicaSSO.jar needs to be placed in host application's library.

The jar will be provided with intellicus setup.

Path for jar file: <Install_Path>\APIs\SingleSignOn\Java

Note: For Intellicus version earlier than 4.1, this jar can be requested to Intellicus Support.

Host application needs to set Intellicus web application URL.

Configuring Intellicus

Intellicus application contains Integration.xml file for integration and dynamic user creation activities.

- In Integration.xml, set business parameters required to pass (If any)
- Set CREATE_USER for Dynamic User creation in Intellicus.

This xml file contains the information regarding integration like user role, dynamic category creation etc.

Path for Integration.xml:

<Intellicus_Install_path>\Jakarta\webapps\intellicus\WEB-INF

Note: Host application needs to give details in xml according to their requirements.

Corresponding to this integration xml file content, createUser() method in ReportControllerDetails.JSP at the Intellicus end should be defined.

Sample Integration xml and Controller jsp are provided with the Intellicus Setup.

Note: For Intellicus version earlier than 4.1, ReportController.jsp,ReportController.Detail.jsp,LaunchPortal.jsp can be requested to Intellicus Support and should be placed at <Intellicus_install_path>\Jakarta\webapps\intellicus.

Implementation for Single Sign-On Request

Implementation code can be written inside any JSP or servlet of host application.

1. Host application needs to add intellicaSSO.jar in their application.

Host application needs to import class Enums.java, SingleSignOn.java, SingleSignOnException.java, UserInfo.java.

import com.intellicus.integration.singlesignon.Enums;

import com.intellicus.integration.singlesignon.SingleSignOn;

import com.intellicus.integration.singlesignon.SingleSignOn Exception;

import com.intellicus.integration.singlesignon.UserInfo;

Make an object of SingleSignOn class for invoking the methods of this class.

Make an object of UserInfo class and set the user credentials using the setter methods provided by UserInfo class.

Constructor

```
public UserInfo()
```

Constructor

public UserInfo(String userId, String orgId)

Parameters:

userId : User Id of the logged in user

orgId : Organization id of the logged in user

Pass this userInfo object to SingleSignOn class using the setUserInfo (userInfo) method.

public void setUserInfo(UserInfo userInfo)

Parameters:

UserInfo: Object reference of UserInfo class.

Set the IntellicusUrl. Intellicus URL can be read from property file.

If host application wants to set some hidden parameters, then invoke the setHiddenParameter (String paramName, String paramValue) for each hidden parameter.

These parameters can be read from property/xml file.

Method:

This method puts the hidden parameter into the hashmap for hidden parameters.

public void setHiddenParameter(String paramName,String paramValue)

Parameters:

paramName: Name of the business parameter.

paramValue: Value of the business parameter.

Note: This method would be called before calling, the getIntellicusToken method. User can not change/update the parameters set through this method. If these parameters need to be changed,then host application need to request intellicusToken again in order to consider new value for these parameters.

Check for the Intellicus Token availability in session. If its not available in session then got to step 11 else go to step 12.

Call the getIntellicusToken() method of SingleSignOn class to get the token from Intellicus.

Method:

This method calls Intellicus API and passes the user credentials and other hidden/business/request parameters to Intellicus.

public String getIntellicusToken() throws SingleSignOnException

Returns:

TokenString: Received token from Intellicus

If host application gets the token from Intellicus, then it redirects the request to Intellicus redirectionAPI with token.

Host Application can set the name of redirectionAPI. Its default value is “LaunchPortal.jsp”.

Method:

This method sets the name of Intellicus jsp to which request is redirected.

public void setRedirectionAPI(String redirectionAPI)

Parameters:

redirectionAPI: Name of the jsp at Intellicus end to which host application wants to redirect the request after user authentication

If host application wants to set some other business parameters, then invoke the setBusinessParameter (String paramName, String paramValue) for each such parameter.

These parameters can be read from property. xml file.

Method:

This method puts the business parameter into the hashmap for request parameters.

public void setBusinessParameter(String paramName,String paramValue)

Parameters:

paramName: Name of the request parameter

paramValue: value of the request parameter.

Note: This method would be called before calling, the redirectToIntellicus method. Parameters set in this method can be updated without requesting new token.

After setting the name of the redirectionAPI, invoke the method for redirecting the request to Intellicus.

Method:

This method sets the name of Intellicus jsp to which request is redirected.

public void redirectToIntellicus(String onSuccess,String intellicusToken,HttpServletResponse response) throws IOException

Parameters:

onSuccess: Name of the requested Intellicus API.

intellicusToken: Token received from Intellicus after user authentication.

response: It is the HttpServletResponse object. It is used for redirecting a request.

If host application does not get the token from intellicus i.e. if user authentication fails at Intellicus end, then host application can show their error page or error message based on the status message returned by the Intellicus.

Setter method for each UserInfo attributes

1. Method to set the User id

public void setUserId (String userId)

Parameters:

userId: User id.

2. Method to set the User Password

public void setPassword (String password)

Parameters:

password: password for the user.

3. Method to set the User's Organization id

public void setOrgID (String orgID)

Parameters:

orgID: organization id for the user.

4. Method to set the User's status(ACTIVE / SUSPENDED)

public void setStatus (String status)

Parameters:

status: status of the user i.e. user is active or suspended.

5. Method to set the user to Super Admin("true"/"false")

public void setIsSuperAdmin (boolean isSuperAdmin)

Parameters:

isSuperAdmin: Whether user is Super admin or not.

6. Method to set the user to Admin("true"/"false")

public void setIsAdmin (boolean isAdmin)

Parameters:

isAdmin: Whether user is admin or not.

7. Method to set role Id's belonging to that user

public void setRoleIds (String roleIds)

Parameters:

roleIds: Role that should be assigned to this user.

8. Method to set role User's Description

public void setDescription (String description)

Parameters:

description: Any description or detail about that user.

9. Method to set session id

public void setSessionId (String sessionId)

Parameters:

sessionId: session id for the user.

10. Method to set Security Descriptor

public void setSecurityDescriptor (String securityDescriptor)

Parameters:

securityDescriptor: any specific information about the user.

11. Method to set customer Id

public void setCustomerId (String customerId)

Parameters:

customerId: customer id for the user.

12. Method to set location

public void setLocation (String location)

Parameters:

location: location for the user.

13. Method to set locale

public void setLocale (String locale)

Parameters:

locale: locale for the user.

14. Method to set database name

public void setDBName (String dbName)

Parameters:

dbName: Database name for the user.

15. Method to set TimeStamp

public void setTimeStamp(long longTimeStamp)

Parameters:

longTimeStamp: timestamp for the user.

16. Method to set System Privileges

public void setSystemPrivileges(String systemPrivileges)

Parameters:

systemPrivileges: system privileges for a user.

17. Method to set blank password

public void setBlankPassword(boolean blankPassword)

Parameters:

blankPassword: it is true or false.

Note: Please refer IntellicusSSOEnduserRequest.java for end user request sample code.
Path: <Install_Path>\SampleCodes\SingleSignOn\Java

Note: Please refer IntellicusSSOLogout.java for logout sample code.
Path: <Install_Path>\SampleCodes\SingleSignOn\Java

Optional Settings

Controller API

This is the main controller for the integration of Intellicus with a host application. It reads information from Integration.xml and sets hidden parameters (like domain ID and workspace ID) at Intellicus end.

Default name of this API is: ReportController.jsp and ReportControllerDetail.jsp.

If required, name of this API can be changed.

To call this API using different name than default names, you need to use method given below.

Method:

This method sets the name of Intellicus JSP to which request is redirected.

public void setIntellicusControllerAPI (String intellicusControllerAPI)

Parameters:

intellicusControllerAPI: Name of the jsp at Intellicus which performs controlling activities for Intellicus.

This controller API is placed at:

<Intellicus_Install_path>\Jakarta\webapps\intellicus

Redirection API

It is the API available at Intellicus end to which request is redirected to from host application to Intellicus.

Its default name is: LaunchPortal.jsp

If required, name of this API can be changed.

To call this API using different name than default names, you need to use method given below.

Method:

This method sets the name of Intellicus JSP to which request is redirected.

public void setRedirectionAPI(String redirectionAPI)

Parameters:

redirectionAPI: Name of the jsp at Intellicus end to which host application wants to redirect the request after user authentication.

This redirection API is placed at:

<Intellicus_Install_path>\Jakarta\webapps\intellicus

lbMode

Intellicus web application can be running on multiple web servers, so in such scenario a load balancer feature is used to decide which web server should serve the reporting request from Host application.

Host application need to specify whether reporting request is sent to load balancer or to a particular web server (in case if there is single web server for Intellicus application).

lbMode: This variable specifies whether to take reporting request to Load balancer or to particular Intellicus web server.

Its default value is: false.

It means no load balancer is in picture.

Method:

This method sets the lbMode as true or false.

public void setLbMode(boolean lbMode)

Parameters:

lbMode: boolean value.

lbRelativePath

This variable specifies the relative path for Load balancer. It is accessed only when lbMode is true.

Its default value is: /LoadBlancerServlet

Method:

This method sets the lbMode as true or false.

public void setLbRelativePath(String lbRelativePath)

Parameters:

lbRelativePath: String for relative path.

intellicusExternalURL

Intellicus application would be accessed by Host web server (Internal IP) to get the IntellicusToken.

But an end-user can access the host application from some outer network. As host application need to redirect the request for Intellicus HTTP APIs from browser, an External IP for Intellicus web application needs to be specified.

intellicusExternalURL: This variable specifies the external URL for Intellicus web application.

Method: This method is to set the External URL for Intellicus.

public void setIntellicusExternalURL(String intellicusExternalURL)

Parameters:

intellicusExternalURL: String for external URL.

Implementation for Logout

On logout from Host application, session for the user is invalidated and is redirected to home page of Host application. Now new user can login through same window.

Because host application and Intellicus web application are running on different web servers, so if host application user logs out from that application, it does not destroy the session in Intellicus for that user.

In order to destroy a session in Intellicus corresponding to a Host application end user, host application need to invoke logout method of Intellicus as well.

So logout needs implementation for both host application as well as Intellicus.

Note: If on logout, host appliaction is closing the current window, then there is no need of invoking the logout action at Intellicus. A new user will login through new window, so new session will be created for that user.

Implementation code can be written inside any JSP or servlet of Host application.

1. Host application need to add intellicaSSO.jar in their application.
2. Host application need to import class Enums.java , SingleSignOn.java, SingleSignOnException.java ,UserInfo.java.

import com.intellicus.integration.singlesignon.Enums;

import com.intellicus.integration.singlesignon.SingleSignOn;

3. Make an object of SingleSignOn class for invoking the methods of this class.
4. Set the IntellicusUrl. Intellicus URL can be read from property file.
5. Call the method logoutFromIntellicus of SingleSignOn.

Method:

This method calls Intellicus API and passes the user credentials to Intellicus.

public void logoutFromIntellicus (HttpServletResponse response)

Parameters:

response: It is the HttpServletResponse object. It is used for redirecting a request.

Sample logout code for Host application:

SingleSignOnLogout.jsp: It contains the steps mentioned above.

index.jsp: Home page of Host application.

frm1: Iframe on Host application screen in which Intellicus reporting feature are called.

```
function fnLogOut()
{
            var logout=confirm('Do you wan to logout?');
            if(logout == false)
            return;
            document.getElementById("frm1").src="SingleSignOnLogout.jsp";
            Form1.action="index.jsp";
```

```
            Form1.target="_self";
            Form1.submit();
}
```

Sample Code for Single Sign-On request:

```
try
{
HttpSession session=request.getSession(true);
  String intellicusToken=null;
            SingleSignOn singleSignOn=new SingleSignOn();

            //Set user credentials for user to be activated/deleted/suspended by
Admin user.
            // OR set the credentials for logged-in user for End -user requests.
            //user password is not required, if the authentication mode for
organization is "Host Application."
            //These credentials can be fetched from the cre
             String hostAppUserid=(String)session.getAttribute("userId");
             String hostAppOrgId=(String)session.getAttribute("orgId");

             UserInfo userInfo=new UserInfo();
             //set the credentials for logged-in user.
            userInfo.setUserId(hostAppUserid);
            userInfo.setOrgID(hostAppOrgId);
            singleSignOn.setUserInfo(userInfo);

            // Set user credentials for admin user.
            // Admin user credentials are required if some request for admin
activity is raised.
            // Admin activities are like User Management, Database connection
creation/modification etc.
            // These can be read from any property file or from
repository/database.
            // Set user credentials for admin user.
            // Admin user credentials are required if some request for admin
activity is raised.
            // Admin activities are like User Management, Database connection
creation/modification etc.
            // These can be read from any property file or from
repository/database.
            String intellicusAdminUserId="Admin";   //This value can be read from
any property file or database.
            String intellicusAdminOrgId="Intellica";//This value can be read from
any property file or database.
            String intellicusAdminPassword="Admin"; //This value can be read from
any property file or database.

 UserInfo adminUserInfo=new UserInfo();
            adminUserInfo.setUserId(intellicusAdminUserId);
            adminUserInfo.setOrgID(intellicusAdminOrgId);
            adminUserInfo.setPassword(intellicusAdminPassword);
```

```
            SingleSignOn.setAdminUserInfo(adminUserInfo);

            // Set the path for Intellicus Web application
            // This can be read from any property file or from
repository/database.

            singleSignOn.setIntellicusURL("http://localhost/intellicus");

            //    Set the business parameters/hidden parameters that need to be
passed to Intellicus
            //This can be read from any property file or from repository/database.
            // for spaces give %20 or +

            singleSignOn.setHiddenParameter("p_CompanyOID","Ultra+Sports+2");

            // get the url for requested Intellicus API like
            // Report listing /Dashboards/User preferences/Query Object list etc.
            String onSuccess=request.getParameter("onSuccess");

            // Check for the availability of Intellicus token in session.
            // If it is not found in session, it means user is first time  giving
request to intellicus.
            // So Call the Intellicus methods to get the Token from Intellicus.
            // This token is sent by Host Application for the further interaction
with intellicus.

            // If token is found in session, then it means, user has already taken
token from intellicus.
            // So,no need to get the token again from Intellicus. User can use the
same token which he has.
             singleSignOn.setIntellicusExternalURL("http://localhost/intellicus");
                        if(session.getAttribute("intellicusToken")==null)//if
token not found in session
            {
                  //if user is not available at Intellicus end,
                  // it will create the user dynamically and assign the role to
that user.

            singleSignOn.setHiddenParameter("USER_ROLES","Admin");

                  //call getIntellicusToken().
                  // this method returns a intellicus token string, if user
authentication is done successfully.

            intellicusToken=singleSignOn.getIntellicusToken();

                  //if user is authenticated by Intellicus, then only call the
Intellicus redirectionAPI
                  //else show the error status message
                  if(singleSignOn.isUserAuthenticated())
                  {
                        session.setAttribute("intellicusToken",
intellicusToken);
                        singleSignOn.setBusinessParameter("ABC","1");
```

```
                        singleSignOn.redirectToIntellicus(onSuccess,
intellicusToken, response);

            }
                  else // if user authentication fails at Intellicus end
                  {
                        PrintWriter out=response.getWriter();

            if(Enums.ResponseMessages.AUTHENTICATION_FAILED.equalsIgnoreCase(

            singleSignOn.getUserAuthenticatedMessage()))
            {
                  out.println("Invalid Login name or Password");
            }
            else
if(Enums.ResponseMessagse.COULD_NOT_CONNECT_TO_REPORT_SERVER.equalsIgnoreCase(

            singleSignOn.getUserAuthenticatedMessage()))

            {
            out.println("Report Server is Down");
            }
            else if(Enums.ResponseMessages.REPOSITORY_DB_IS_DOWN.equalsIgnoreCase(

            singleSignOn.getUserAuthenticatedMessage()))
            {
                  out.println("Repository Database Connection is Down");
            }
            else
            {
                  out.println(singleSignOn.getUserAuthenticatedMessage());
            }
}
}
else// if token found in session
{
singleSignOn.setBusinessParameter("ABC","2");
singleSignOn.redirectToIntellicus(onSuccess, intellicusToken, response);
            }

}
            catch(SingleSignOnException e)// if connection for the intellicusURL
can not be opened.Reason can be
//Intellicus url is wrong or Report Server is down.
{
            PrintWriter out=response.getWriter();
            out.println("Intellicus Web Application Not Available ");
}
            catch(Exception e)
            {
            PrintWriter out=response.getWriter();
            out.println("Intellicus Web Application Not Available ");
            }
```

3 SSO for Host Application on .Net Platform

Configurations Required

Configuring Host Application

In order to invoke methods at Intellicus end, the file intellicaSSO.dll needs to be placed in host application's library.

This dll will be provided with intellicus setup.

Path for dll file: <Install_Path>\APIs\SingleSignOn\DotNet

Note: For Intellicus version earlier than 4.1, this file can be requested to Intellicus Support.

Host application needs to set Intellicus web application URL.

Configuring Intellicus

Intellicus application contains Integration.xml file for integration and dynamic user creation activities.

- In Integration.xml, set business parameters required to pass (If any)
- Set CREATE_USER for Dynamic User creation in Intellicus.

This xml file contains the information regarding integration like user role, dynamic category creation etc.

Path for Integration.xml:

<Intellicus_Install_path>\Jakarta\webapps\intellicus\WEB-INF

Note: Host application needs to give details in xml according to their requirements.

Corresponding to this integration xml file content, createUser() method in ReportControllerDetails.JSP at the Intellicus end should be defined.

Sample Integration xml and Controller jsp are provided with the Intellicus Setup.

Note: For Intellicus version earlier than 4.1, ReportController.jsp,ReportController.Detail.jsp,LaunchPortal.jsp can be requested to Intellicus Support and should be placed at <Intellicus_install_path>\Jakarta\webapps\intellicus.

Implementation for Single Sign-On Request

Implementation code can be written inside any aspx of Host application.

1. Host application needs to add intellicaSSO.dll in their application.
2. Host application needs to import namespace Intellicus.Integration.SingleSignOn using Intellicus.Integration.SingleSignOn;

Make an object of SingleSignOn class for invoking the methods of this class.

Make an object of UserInfo class and set the user credentials using the setter methods provided by UserInfo class.

Constructor

```
public UserInfo()
```

Constructor

public UserInfo(String userId, String orgId)

Parameters:

userId : User Id of the logged in user

orgId : Organization id of the logged in user

Set this userInfo object to UserInfo property of SingleSignOn class.

singleSignOn.UserInfo = userInfo;

Set the IntellicusUrl. Intellicus url can be read from property file.

If host application wants to set some hidden parameters, then invoke the setHiddenParameter (String paramName, String paramValue) for each hidden parameter.

These parameters can be read from property/xml file.

Method:

This method puts the hidden parameter into the hashmap for hidden parameters.

public void setHiddenParameter(String paramName,String paramValue)

Parameters:

paramName: Name of the business parameter.

paramValue: Value of the business parameter.

Note: This method would be called before calling, the getIntellicusToken method. User can not change/update the parameters set through this method. If these parameters need to be changed,then host application need to request intellicusToken again in order to consider new value for these parameters.

Check for the Intellicus Token availability in session. If it's not available in session then got to step 11 else go to step 12.

Call the getIntellicusToken () method of SingleSignOn class to get the token from Intellicus.

Method:

This method calls Intellicus API and passes the user credentials and other hidden parameters to Intellicus. It throws SingleSignOnException.

public String getIntellicusToken()

Returns:

TokenString: Received token from Intellicus

If host application gets the token from Intellicus, then it redirects the request to Intellicus redirectionAPI with token.

Host Application can set the name of intellicusRedirectionAPI. Its default value is “LaunchPortal.jsp”.

Property:

This property sets the name of Intellicus JSP to which request is redirected.

```
        public String IntellicusRedirectionAPI
        {
    get { return intellicusRedirectionAPI; }
    set { intellicusRedirectionAPI = value; }
  }
```

IntellicusRedirectionAPI: Name of the jsp at Intellicus end to which host application wants to redirect the request after user authentication

If host application wants to set some other business parameters, then invoke the setBusinessParameter (String paramName, String paramValue) for each such parameter.

These parameters can be read from property, xml file.

Method:

This method puts the business parameter into the hashmap for request parameters.

public void setBusinessParameter(String paramName,String paramValue)

Parameters:

paramName: Name of the request parameter

paramValue: value of the request parameter.

Note: This method would be called before calling, the redirectToIntellicus method. Parameters set in this method can be updated without requesting new token.

After setting the name of the redirectionAPI, invoke the method for redirecting the request to Intellicus.

Method:

This method sets the name of Intellicus jsp to which request is redirected. It throws SingleSignOnException.

public void redirectToIntellicus(String onSuccess,String intellicusToken)

Parameters:

onSuccess: Name of the requested Intellicus API.

intellicusToken: Token received from Intellicus after user authentication.

response: It is the HttpServletResponse object. It is used for redirecting a request.

If host application does not get the token from intellicus i.e. if user authentication fails at Intellicus end, then host application can show their error page or error message based on the status message returned by the Intellicus.

If logout action is invoked at Host application end, then logout action should also be invoked at the Intellicus end.

Call the method logoutFromIntellicus() method of SingleSignOn class to invalidate the session at Intellicus end.

Method:

This method puts the business parameter into the hashmap for business parameters.

public void logoutFromIntellicus()

Getter/Setter property for each UserInfo attributes

1. Get/Set the User id

public string UserId

{

    get { return userId; }

```
 set { userId = value; }
}
```

2. Get/Set the User Password.

```
public string Password
{
        get { return password; }
    set { password = value; }
}
```

3. Get/Set the User's Organization id

```
public string Organization
{
        get { return orgID; }
    set { orgID = value; }
}
```

4. Get/Set the User's status (ACTIVE / SUSPENDED).

```
public string Status
{
    get { return status; }
 set { status = value; }
}
```

5. Get/Set the user to Super Admin ("true"/"false").

```
public bool IsSuperAdmin
{
    get { return isSuperAdmin; }
 set { isSuperAdmin = value; }
}
```

6. Get/Set the user to Admin("true"/"false").

```
public bool IsAdmin
{
```

```
            get { return isAdmin; }
    set { isAdmin = value; }
}
```

7. Get/Set role Id's belonging to that user.

```
public string RoleIds
{
 get { return roleIds; }
 set { roleIds = value; }
}
```

8. Get/Set role User's Description.

```
public string Description
{
    get { return description; }
 set { description = value; }
}
```

9. Get/Set the session id.

```
public string SessionId
{
            get { return sessionId; }
    set { sessionId = value; }
}
```

10. Get/Set Security Descriptor.

```
public string SecurityDescriptor
{
 get { return securityDescriptor; }
 set { securityDescriptor = value; }
}
```

11. Get/Set customer Id.

```
public string CustomerId
{
        get { return customerId; }
    set { customerId = value; }
}
```

12. Get/Set location.

```
public string Location
{
        get { return location; }
 set { location = value; }
}
```

13. Get/Set locale.

```
public string Locale
{
        get { return locale; }
        set { locale = value; }
}
```

14. Get/Set database name.

```
public string DBName
{
    get { return dbName; }
    set { dbName = value; }
}
```

15. Get/Set TimeStamp.

```
public long TimeStamp
{
        get { return longTimeStamp; }
```

```
    set { longTimeStamp = value; }
  }
```

Note: Please refer IntellicusSSOEnduserRequest.aspx for end user request sample code.
Path: <Install_Path>\SampleCodes\SingleSignOn\DotNet

Note: Please refer IntellicusSSOLogout.aspx for logout sample code.
Path: <Install_Path>\SampleCodes\SingleSignOn\DotNet

Optional Settings

Controller API

Integrating with Intellicus, Controller API at Intellicus end can be given any suitable name.

So in order to call the Intellicus controller API, host application can set the name of controller API by setter property provided by intellicaSSO.dll

Its default name is: ReportController.jsp and ReportControllerDetail.jsp

Property:

This property gets/sets the name of Intellicus jsp to which request is redirected.

```
public String IntellicusControllerAPI
{
    get { return intellicusControllerAPI; }
    set { intellicusControllerAPI = value; }
}
```

IntellicusControllerAPI: Name of the jsp at Intellicus which performs controlling activities for Intellicus.

This controller API is placed at:

<Intellicus_Install_path>\Jakarta\webapps\intellicus

Redirection API

It is the API available at Intellicus end to which request is redirected to from Host application to Intellicus.

Its default name is: LaunchPortal.jsp

Property:

This property gets/sets the name of Intellicus jsp to which request is redirected.

```
public String IntellicusRedirectionAPI
{
    get { return intellicusRedirectionAPI; }
    set { intellicusRedirectionAPI = value; }
}
```

IntellicusRedirectionAPI: Name of the jsp at Intellicus end to which host application wants to redirect the request after user authentication

This redirection API is placed at:

<Intellicus_Install_path>\Jakarta\webapps\intellicus

lbMode

This variable specifies, whether to take reporting request to Load balancer or to specify Intellicus web server.

Its default value is: false

Property:

This property gets/sets the lbMode as true or false.

```
public bool LbMode
{
            get { return lbMode; }
    set { lbMode = value; }
}
```

lbRelativePath

This variable specifies the relative path for Load balancer. It is accessed only when lbMode is true.

Its default value is : /LoadBlancerServlet

Property:

This property gets/sets the lbRelativePath.

```
public String LbRelativePath
{
 get { return lbRelativePath; }
 set { lbRelativePath = value; }
}
```

LbRelativePath: String for relative path

intellicusExternalURL

Intellicus application would be accessed by Host web server (Internal IP) for getting the IntellicusToken.

But an End User can Access the Host application from some outer network. As host application need to redirect the request for Intellicus HTTP APIs from browser, an External IP for Intellicus web application need to be specified.

intellicusExternalURL**:** This variable specifies the external URL for Intellicus web application.

Method:

This method is to set the External URL for Intellicus.

```
public String IntellicusExternalURL
{
  get { return intellicusExternalURL; }
   set { intellicusExternalURL = value; }
}
```

Parameters:

intellicusExternalURL: String for external URL.

Sample Code for Single Sign-On request:

```
String intellicusToken = null;
String hostAppUserid = null;
String hostAppOrgId = null;
SingleSignOn singleSignOn = new SingleSignOn();
try
{

#region Creating UserInfo

     //Set the credentials for logged-in user for End -user
     //requests.
     // user password is not required, if the authentication mode
     //for organization is "Host Application."
     //These credentials can be fetched from the data structure
     //maintained for the selected user.

        if (Session["userId"] != null)
          hostAppUserid = Session["userId"].ToString();
        if (Session["orgId"] != null)
          hostAppOrgId = Session["orgId"].ToString();

     //set the credentials for the user to be
     // activated/deleted/suspended/modified

        UserInfo userInfo =new UserInfo();
        userInfo.UserId = hostAppUserid;
        userInfo.OrgID =  hostAppOrgId;
        singleSignOn.UserInfo = userInfo

#endregion

#region Create AdminInfo

                //Set user credentials for admin user.
     //Admin user credentials are required if some request for
     //admin activity is raised.
     //Admin activities are like User Management, Database
     //connection creation/modification etc.
     //These can be read from any property file or from
     //repository/database.

   //This value can be read from any prperty file or database.
     String intellicusAdminUserId = "Admin";
 //This value can be read from any prperty file or database.
 String intellicusAdminOrgId = "Intellica”;
 //This value can be read from any prperty file or database.
     String intellicusAdminPassword = "Admin";

                UserInfo adminUserInfo =new UserInfo();
     adminUserInfo.UserId = intellicusAdminUserId;
     adminUserInfo.OrgID = intellicusAdminOrgId;
     adminUserInfo.Password = intellicusAdminPassword;
```

```
     SingleSignOn.AdminUserInfo = adminUserInfo;

#endregion

#region Get/Set Intellicus Path and Parameters

   //Set the path for Intellicus Web application
   //This can be read from any property file or from
   //repository/database.
    singleSignOn.IntellicusURL= "http://192.168.33.165/intellicusvss";

   //Set the business parameters/hidden parameters that need to
   //be passed to Intellicus
   //This can be read from any property file or from
   //repository/database.
   //These parameters should be mentioned in Integration xml.
   //parameter name mentioned here should be same as parameter
   //name mentioned in Integration xml.

singleSignOn.setHiddenParameter("p_CompanyOID","Ultra Sports 5");
singleSignOn.setHiddenParameter("prmCategoryName",  "cat1");
singleSignOn. setHiddenParameter ("REPORT_CONN_NAME", "ReportDB");

   //get the url for requested Intellicus API like
   //Report listing /Dashboards/User preferences/Query Object
   //list etc.

String onSuccess = "./core/CategoryList.jsp";

#endregion



#region Act Based on Token Availability

  //Check for the availability of Intellicus token in session.
  //If it is not found in session, it means user is first time
  //giving request to intellicus.
  //So Call the Intellicus methods to get the Token from
  //Intellicus.
  //This token is sent by Host Application for the further
  //interaction with intellicus.
  //If token is found in session,then it means,user has already
  //taken token from intellicus.
  //So,no need to get the token again from Intellicus.User can
  //use the same token which he has.

  //if token not found in Session
  if (Session["intellicusToken"] == null)
  {
    //if user is not available at Intellicus end,
    // it will create the user dynamically and assign the
    //role to that user.
    // these roles should have entry in Integration xml.
```

```
    singleSignOn.setHiddenParameter("USER_ROLES", "Admin");
    //this method returns a intellicus token string ,if
    //user authentication is done successfully.
    intellicusToken = singleSignOn.getIntellicusToken();
    //if user is authenticated by Intellicus
    //then only call the Intellicus redirectionAPI
    //else show the error status message
    if (singleSignOn.IsUserAuthenticated)
    {
     Session["intellicusToken"] = intellicusToken;
     singleSignOn.redirectToIntellicus(onSuccess,
     intellicusToken);
    }
    else
    {
    string userAuthMsg = singleSignOn.UserAuthenticationMessage;
      if(userAuthMsg.Equals(
             Enums.ResponseMessages.AUTHENTICATION_FAILED))
      {
        Response.Write("Invalid Login name or Password
        OR Invalid Host web server IP");
      }
      else if(userAuthMsg.Equals(
      Enums.ResponseMessages.COULD_NOT_CONNECT_TO_REPORT_SERVER))
      {
         Response.Write("Report Server is Down");
      }
       else if(userAuthMsg.Equals(
              Enums.ResponseMessages.REPOSITORY_DB_IS_DOWN))
       {
           Response.Write("Repository Database Connection
           is Down");
       }
       else
       {
           Response.Write(userAuthMsg);
       }
   }

 }
 else// if token found in session
 {

singleSignOn.redirectToIntellicus(onSuccess,intellicusToken);
 }
#endregion

}
catch(SingleSignOnException ex)
{
    Response.Write(ex.Message);
}
catch (Exception exc)
{
    Response.Write(exc.Message);;
```

```
}
```

4 SSO for Host Application on PHP Platform

Configurations Required

Configuring Host Application

In order to invoke methods at Intellicus end, the file intellicaSSO.php needs to be placed in host application's library.

This file will be provided with intellicus setup.

Path for the file: <Install_Path>\APIs\SingleSignOn\PHP

Note: For Intellicus version earlier than 4.1, this php can be requested to Intellicus Support.

Host application needs to set Intellicus web application URL.

Configuring Intellicus

Intellicus application contains Integration.xml file for integration and dynamic user creation activities.

- In Integration.xml, set business parameters required to pass (If any)
- Set CREATE_USER for Dynamic User creation in Intellicus.

This xml file contains the information regarding integration like user role, dynamic category creation etc.

Path for Integration.xml:

<Intellicus_Install_path>\Jakarta\webapps\intellicus\WEB-INF

Note: Host application needs to give details in xml according to their requirements.

Corresponding to this integration xml file content, createUser() method in ReportControllerDetails.JSP at the Intellicus end should be defined.

Sample Integration xml and Controller jsp are provided with the Intellicus Setup.

Note: For Intellicus version earlier than 4.1, ReportController.jsp,ReportController.Detail.jsp,LaunchPortal.jsp can be requested to Intellicus Support and should be placed at <Intellicus_install_path>\Jakarta\webapps\intellicus.

Implementation for Single Sign-On Request

Implementation code can be written inside any php or html file of Host application.

1. Host application needs to add intellicaSSO.php in their application.

Host application file needs to include file intellicaSSO.php.

include_once("intellicaSSO.php");

Make an object of SingleSignOn class for invoking the methods of this class.

Make an object of UserInfo class and set the user credentials using the setter methods provided by UserInfo class.

Pass this userInfo object to SingleSignOn class using the setUserInfo (userInfo) method.

public function setUserInfo($userInfo)

Parameters:

UserInfo: Object reference of UserInfo class.

Set the IntellicusUrl. Intellicus url can be read from property file.

If host application wants to set some hidden parameters, then invoke the setHiddenParameter ($paramName, $paramValue) for each hidden parameter.

These parameters can be read from property/xml file.

Method:

This method puts the hidden parameter into the array for hidden parameters.

public function setHiddenParameter($paramName,$paramValue)

Parameters:

paramName: Name of the business parameter

paramValue: Value of the business parameter.

Note: This method would be called before calling, the getIntellicusToken method. User can not change/update the parameters set through this method. If these parameters need to be changed, then host application need to request intellicusToken again in order to consider new value for these parameters.

Check for the Intellicus Token availability in session. If its not available in session then got to step 11 else go to step 12.

Call the getIntellicusToken() method of SingleSignOn class to get the token from Intellicus.

Method:

This method calls Intellicus API and passes the user credentials and other hidden/business/request parameters to Intellicus.

public function getIntellicusToken()

Returns:

TokenString: Received token from Intellicus

If host application gets the token from Intellicus, then it redirects the request to Intellicus redirectionAPI with token.

Host Application can set the name of redirectionAPI. Its default value is “LaunchPortal.jsp”.

Method:

This method sets the name of Intellicus jsp to which request is redirected.

public function setRedirectionAPI($redirectionAPI)

Parameters:

redirectionAPI: Name of the jsp at Intellicus end to which host application wants to redirect the request after user authentication

If host application wants to set some other business parameters, then invoke the setBusinessParameter ($ paramName, $paramValue) for each such parameter.

These parameters can be read from property, xml file.

Method:

This method puts the business parameter into the array for request parameters.

public function setBusinessParameter($paramName,$paramValue)

Parameters:

paramName: Name of the request parameter.

paramValue: value of the request parameter.

Note: This method should be called before calling, the redirectToIntellicus method. Parameters set in this method can be updated without requesting new token.

After setting the name of the redirectionAPI, invoke the method for redirecting the request to Intellicus.

Method:

This method sets the name of Intellicus jsp to which request is redirected.

public function redirectToIntellicus($onSuccess,$ intellicusToken)

Parameters:

onSuccess: Name of the requested Intellicus API.

intellicusToken: Token received from Intellicus after user authentication.

If host application does not get the token from intellicus i.e. if user authentication fails at Intellicus end, then host application can show their error page or error message based on the status message returned by the Intellicus.

Setter method for each UserInfo attributes

1. Method to set the User id

public function setUserId ($userId)

Parameters:

userId: User id.

2. Method to set the User Password

public function setPassword ($password)

Parameters:

password: password for the user.

3. Method to set the User's Organization id

public function setOrgID ($orgID)

Parameters:

orgID: organization id for the user.

4. Method to set session id

public function setSessionId ($sessionId)

Parameters:

sessionId: session id for the user.

5. Method to set Security Descriptor

public function setSecurityDescriptor ($securityDescriptor)

Parameters:

securityDescriptor: any specific information about the user.

6. Method to set customer Id

public function setCustomerId ($customerId)

Parameters:

customerId: customer id for the user.

7. Method to set location

public function setLocation ($location)

Parameters:

location: location for the user.

8. Method to set locale

public function setLocaleForIntellicus ($locale)

Parameters:

locale: locale for the user.

9. Method to set database name

public function setDBName ($dbName)

Parameters:

dbName: Database name for the user.

10. Method to set TimeStamp

public function setTimeStamp($longTimeStamp)

Parameters:

longTimeStamp: timestamp for the user.

11. Method to set the User's status(ACTIVE / SUSPENDED)

public function setStatus ($status)

Parameters:

status: status of the user i.e. user is active or suspended.

12. Method to set the user to Super Admin("true"/"false")

public function setIsSuperAdmin ($isSuperAdmin)

Parameters:

isSuperAdmin: Whether user is Super admin or not.

13. Method to set the user to Admin("true"/"false")

public function setIsAdmin ($isAdmin)

Parameters:

isAdmin: Whether user is admin or not.

14. Method to set role Id's belonging to that user

public function setRoleIds ($roleIds)

Parameters:

roleIds: Role that should be assigned to this user.

15. Method to set User's Description

public function setDescription ($description)

16. Method to set System Privileges

public function setSystemPrivileges($systemPrivileges)

Parameters:

systemPrivileges: system privileges for a user.

17. Method to set blank password

public function setBlankPassword($blankPassword)

Parameters:

blankPassword: it is true or false.

Note: Please refer IntellicusSSOEnduserRequest.php for end user request sample code.
Path: <Install_Path>\SampleCodes\SingleSignOn\PHP

Note: Please refer IntellicusSSOLogout.php for logout sample code.
Path: <Install_Path>\SampleCodes\SingleSignOn\PHP

Optional Settings

Controller API

This is the main controller for the integration of Intellicus with a host application. It reads information from Integration.xml and sets hidden parameters (like domain ID and workspace ID) at Intellicus end.

Default name of this api is: ReportController.jsp and ReportControllerDetail.jsp

If required, name of this API can be changed.

To call this API using different name than default names, you need to use method given below.

Method:

This method sets the name of Intellicus jsp to which request is redirected.

public function setIntellicusControllerAPI ($intellicusControllerAPI)

Parameters:

intellicusControllerAPI: Name of the jsp at Intellicus which performs controlling activities for Intellicus.

This controller API is placed at:

<Intellicus_Install_path>\Jakarta\webapps\intellicus

Redirection API

It is the API available at Intellicus end to which request is redirected to from host application to Intellicus.

Its default name is: LaunchPortal.jsp

If required, name of this API can be changed.

To call this API using different name than default names, you need to use method given below.

Method:

This method sets the name of Intellicus jsp to which request is redirected.

public function setRedirectionAPI($redirectionAPI)

Parameters:

redirectionAPI: Name of the jsp at Intellicus end to which host application wants to redirect the request after user authentication

This redirection API is placed at:

<Intellicus_Install_path>\Jakarta\webapps\intellicus

lbMode

Intellicus web application can be running on multiple web servers, so in such scenario a load balancer feature is used to decide which web server should serve the reporting request from Host application.

Host application need to specify whether reporting request is sent to load balancer or to a particular web server (in case if there is single web server for Intellicus application).

lbMode: This variable specifies whether to take reporting request to Load balancer or to particular Intellicus web server.

Its default value is: false

It means no load balancer is in picture.

Method:

This method sets the lbMode as true or false.

public function setLbMode($lbMode)

Parameters:

lbMode: boolean value.

lbRelativePath

This variable specifies the relative path for Load balancer. It is accessed only when lbMode is true.

Its default value is: /LoadBlancerServlet .

Method:

This method sets the lbMode as true or false.

public function setLbRelativePath($lbRelativePath)

Parameters:

lbRelativePath: String for relative path.

intellicusExternalURL

Intellicus application would be accessed by Host web server (Internal IP) for getting the IntellicusToken.

But an End User can Access the Host application from some outer network. As host application need to redirect the request for Intellicus HTTP APIs from browser, an External IP for Intellicus web application need to be specified.

intellicusExternalURL: This variable specifies the external URL for Intellicus web application.

Method:

This method is to set the External URL for Intellicus.

public function setIntellicusExternalURL($intellicusExternalURL)

Parameters:

intellicusExternalURL: String for external URL.

Implementation for Logout

On logout from host application, session for the user is invalidated and user is redirected to home page of Host application. Now new user can login through same window.

As Host application and Intellicus web application are running on different web servers, so if Host application user logout from that application, it does not destroy the session in Intellicus for that user.

In order to destroy a session in Intellicus corresponding to a Host application end user, Host application need to invoke logout method of Intellicus as well.

So logout needs implementation for both host application as well as Intellicus.

Note: If on logout, Host appliaction is closing the current window, then there is no need of invoking the logout action at Intellicus. A new user will login through new window, so new session will be created for that user.

Implementation code can be written inside any php or html file of Host application.

1. Host application need to add intellicaSSO.php in their application.
2. Host application needs to include file intellicaSSO.php.

include_once("intellicaSSO.php");

3. Make an object of SingleSignOn class for invoking the methods of this class.

4. Set the IntellicusUrl. Intellicus url can be read from property file.

5. call the method logoutFromIntellicus of SingleSignOn.

Method:

public function logoutFromIntellicus()

Sample Code for Single Sign-On request:

include_once("intellicaSSO.php");

```
try
{

            $sso=new SingleSignOn();

            // Set the path for Intellicus Web application
            // This can be read from any property file or from
repository/database.
            // In lbMode , give the URl for LB application,
            // else give the URL for Intellicus Web application.
            $sso->setIntellicusURL("http://192.168.33.165/intellicus");
            //$sso->setLbMode(TRUE);

            // Set the credentials for logged-in user for End -user requests.
            // user password is not required, if the authentication mode for
organization is "Host Application."
            // This user must exist at Intellicus.
            $userinfo=new UserInfo;
            $userinfo->setUserId("a");
            $userinfo->setOrgID("ab");
            /*$userinfo->setUserId((string)$_SESSION['userId']);
            $userinfo->setOrgID((string)$_SESSION['orgId']);*/
            $sso->setUserInfo($userinfo);

            //      Set user credentials for admin user.
            // Admin user credentials are required if some request for admin
activity is raised.
            // Admin activities are like User Management, Database connection
creation/modification etc.
            // These can be read from any property file or from
repository/database.
            $adminUserinfo=new UserInfo;
            $adminUserinfo->setUserId("Admin");
            $adminUserinfo->setPassword("Admin");
            $adminUserinfo->setOrgID("Intellica");
            //$sso->setAdminUserInfo($adminUserinfo);
            SingleSignOn::setAdminUserInfo($adminUserinfo);
```

```
            //Set the business parameters/hidden parameters that need to be passed
to Intellicus
            //This can be read from any property file or from repository/database.
            $sso->setHiddenParameter("p_CompanyOID","Company_xyz");

            session_start();

            // get the url for requested Intellicus API like
            // Report listing /Dashboards/User preferences/Query Object list etc.
            //$onSuccess="./core/ReportListForCategory.jsp?REPORT_TYPE=ADHOC";
            $onSuccess="./core/CategoryList.jsp?";

            // Check for the availability of Intellicus token in session.
            // If it is not found in session, it means user is first time  giving
request to intellicus.
            // So Call the Intellicus methods to get the Token from Intellicus.
            // This token is sent by Host Application for the further interaction
with intellicus.

            // If token is found in session,then it means,user has already  taken
token from intellicus.
            // So,no need to get the token again from Intellicus.User can use the
same token which he has.
            if($_SESSION['intellicusToken']==null)
            {

                  //if user is not available at Intellicus end,
                  // it will create the user dynamically and assign the role to
that user.
                  // these roles should have entry in Integration xml.
                  $sso->setHiddenParameter("USER_ROLES","Admin");

                  //call getIntellicusToken().
                  // this method returns a intellicus token string ,if user
authentication is done successfully.
                  $token=$sso->getIntellicusToken();

                  //if user is authenticated by Intellicus,then only call the
Intellicus redirectionAPI
                  //else show the error status message
                  if(trim($sso->isUserAuthenticated())=="TRUE")
                  {
                        $_SESSION['intellicusToken']=$token;
                        $sso->setBusinessParameter("ABC","1");
                        $sso->redirectToIntellicus($onSuccess,$token);
                  }
                  else // if user authentication fails at Intellicus end
                  {
                        print_r($sso->getUserAuthenticationMessage());
                  }
            }
            else // if token found in session
            {
                  $sso->setBusinessParameter("ABC","2");
```

```
                    $sso->redirectToIntellicus($onSuccess,$token);
            }

}
catch(SingleSignOnException $e)// if connection for the intellicusURL can not be
opened.Reason can be
//Intellicus url is wrong or Report Server is down.
{
            print_r("Intellicus Web Application Not Available");
}
catch(Exception $e)
{
            print_r("Intellicus Web Application Not Available");
}
```

5 Sample XML file for Integration (Integration.xml)

Integration xml file contains the various integration requirement details.

Integration requirement may include User management, Host web server IP authentication, Admin user credentials, etc.

Host application would require making changes in the ReportController.jsp and ReportControllerDetail.jsp corresponding to details mentioned in this integration xml.

These two JSPs are placed at:

<Intellicus_Install_path>\Jakarta\webapps\intellicus

Note: If any changes are made in xml, then Intellicus Web server need to be restarted.

Below is sample of Integration xml:

```
<INTEGRATION_DETAILS CREATE_USER="true" CREATE_ORG="true" CREATE_MAPPING="true"
UPDATE_CONN_AR="true" PERMISSION_TO_CONN="ROLE" UPDATE_USER="true" >

            <HOST_WEBSERVER_DETAIL IP="">
            </HOST_WEBSERVER_DETAIL>

            <ADMIN_USER>
            <ID USERID="" PASSWORD="" ORGID=""/>
            </ADMIN_USER>

            <REPORTING_ROLES UPDATE_ORG_ROLES="true" UPDATE_USER_ROLES="false">
                  <REPORTING_ROLE NAME="Basic">
                        <ENTITY_TYPES>
                              <ENTITY_TYPE TYPE="CAT">
                                    <ENTITY  CREATION="Static" ID="4F9245A7-
D639-4F99-604D-F32641B77725"  ACCESS_LEVEL="2" ACCESSRIGHT="0,2,4,6,8,10,12"
></ENTITY>
                                    <ENTITY  CREATION="Dynamic"
ID="&lt;%prmCategoryName%&gt;"  ACCESS_LEVEL="2" ACCESSRIGHT="0,2,4,6,8,10,12"
PARAM="prmCategoryName"></ENTITY>
                              </ENTITY_TYPE>
                        </ENTITY_TYPES >
                  </REPORTING_ROLE>
```

```
                    <REPORTING_ROLE NAME="Admin">
                          <ENTITY_TYPES >
                                <ENTITY_TYPE TYPE="CAT">
                                      <ENTITY CREATION="Static" ID="Setup"
ACCESS_LEVEL="2" ACCESSRIGHT="0,2,4,6,8,10,12" ></ENTITY>
                                      <ENTITY CREATION="Dynamic"
ID="Finance_&lt;%prmCategoryName%&gt;"  ACCESS_LEVEL="2"
ACCESSRIGHT="0,2,4,6,8,10,12" PARAM="prmCategoryName"></ENTITY>
                                      <ENTITY CREATION="Dynamic"
ID="HR_&lt;%prmCategoryName%&gt;"  ACCESS_LEVEL="2" ACCESSRIGHT="0,2,4,6,8,10,12"
PARAM="prmCategoryName"></ENTITY>
                                </ENTITY_TYPE>
                          </ENTITY_TYPES>
                    </REPORTING_ROLE>

              </REPORTING_ROLES>

</INTEGRATION_DETAILS>
```

Details of xml:

- <INTEGRATION_DETAILS> tag:

CREATE_USER:

Possible values:

true: Check for existence of User at run time. If User does not exist, create the User at Intellicus.

false: Do not check for existence of User at run time.

If CREATE_USER is true then only further attributes like CREATE_ORG, CREATE_MAPPING will be considered.

No other tag/attribute in xml is dependent on CREATE_USER attribute.

Dynamic user creation is supported only for “Host Application” and “Call Back” authentication mode.

CREATE_ORG:

This attribute is read only when CREATE_USER is true.

Dynamic organization creation is supported only for “**Host Application**” authentication mode.

Possible values:

true: Check for existence of Organization at run time. If Organization does not exist, create the Organization at Intellicus.

false: Do not check for existence of Organization at run time.

CREATE_MAPPING:

This attribute is read only when CREATE_USER is true.

Possible values:

true: Check for existence for user mapping at run time if user already exists. If user mapping does not exist, create the mapping of at Intellicus.

False: Do not check for existence for user mapping at run time if user already exists.

UPDATE_CONN_AR:

Possible value: true/false

true: Allow modification of access control for any DBCONNECTION Object at Intellicus.

false: Do not allow modification of access control for any DBCONNECTION Object at Intellicus.

PERMISSION_TO_CONN:

This attribute is read only when UPDATE_CONN_AR is true.

It is for providing access control of any DBCONNECTION to any ORG/ROLE/USER.

Possible values: ORG/ ROLE/USER.

ORG: Grant access control of DBCONNECTION Object on ORG. ORG can be existing/new.

ROLE: Grant access control of DBCONNECTION Object on ROLE level. ROLE can be existing/new.

USER: Grant access control of DBCONNECTION Object on USER level. USER can be existing/new.

UPDATE_USER:

This is applicable for both new user (user mapping is added for existing user) as well as existing user

Possible value: true/false

true: Allow modification of existing user's attributes.

false: Do not allow modification of existing user's attributes

<HOST_WEBSERVER_DETAIL> tag:

IP:

List of IP addresses separated by ',' on which Host application is running.

If host application is running on multiple web servers, then IP address for each web server should be mentioned, separated by “,”.

If this authentication is not required, then leave the IP attribute of the <HOST_WEBSERVER_DETAIL> tag empty.

Note: If both Host application and Intellicus are running on the same web server, then give the IP address of localhost i.e. 127.0.0.1

- <ADMIN_USER> tag:

  Credentials of admin user having system privileges of super admin can be entered in this xml under the tag <ADMIN_USER>.

- <REPORTING_ROLES> tag:

  All the reporting roles should be defined in this xml under this tag. Only those roles that are defined under this tag are created at Intellicus.

UPDATE_ORG_ROLES:

Possible values:

true: check for the existence of the roles at run time to match with xml if org already exists. If any role does not exist, then add that role to the organization.

false: Do not check for the existence of the roles at run time to match with xml if org already exists.

UPDATE_USER_ROLES:

Possible values:

true: check for the role assigned to user at run time to match with xml , if user already exists.

false: Do not check for the role assigned to user at run time to match with xml, if user already exists.

Each reporting role has following properties:

Name: Name of the role

Access right of Role are:

1. ENTITY_TYPE: It can be "CAT" for category or "REPORT" for report.
2. ENTITY on which access need to be provided.
3. Attributes of ENTITY if ENTITY_TYPE is CAT are:

CREATION: If the entity will be created dynamically or manually.

Dynamic: For dynamic creation.

Static: For manual creation.

ID: Id of entity.

ACCESS_LEVEL:

Deny Access: 0.

Full Access: 1.

Partial Access: 2.

ACCESSRIGHT: The possible values for entity type category are

View reports: 0.

View reports secured: 1

Save reports: 2

Save reports secured:3

Export reports: 4.

Export reports secured: 5

Print reports: 6

Print reports secured: 7

Print reports at server: 8

Print reports at server secured: 9

Schedule reports: 10

Publish layouts: 11

Publish outputs: 12

PARAM: If any parameter is associated with that entity. This is applicable to only Dynamic creation entities.

6 Admin Activities Performed through SSO

Admin activities deal with the action performed by Admin user of Intellicus.

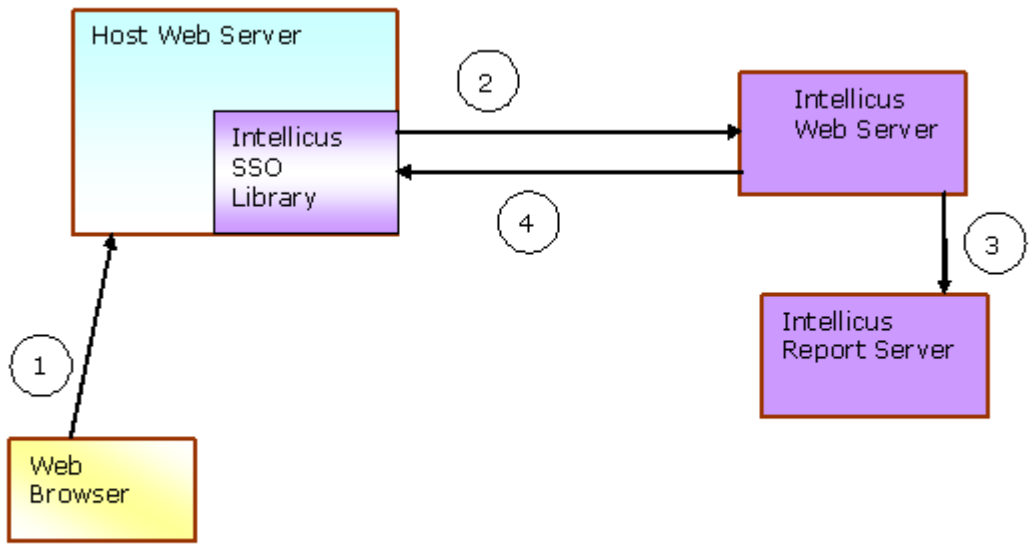
It includes User Management at Intellicus.

User management takes care of following activities:

User create/activate/delete/suspend operation initiated by Admin User.

For admin activities in Intellicus, host application need to send admin user credentials to Intellicus. This admin user should exist in Intellicus and must have super administrator system privilege.

Integration Flow



Steps:

1. In Host Application, user sends request for admin activity at Intellicus. With this request host application sends user credentials, Intellicus admin user credentials and appropriate action code.
2. Using Intellicus SSO Libraries, host application send request for Intellicus Controller API.
3. Intellicus web server send request to report server for admin activity.

A status message is sent back to the host application both in case of activity requested is performed successfully or failed.

Host Application on Java Platform

Implementation code can be written inside any JSP or servlet of Host application.

1. Host application need to add intellicaSSO.jar in their application.
2. Host application need to import class Enums.java, SingleSignOn.java, SingleSignOnException.java, UserInfo.java.

import com.intellicus.integration.singlesignon.Enums;

import com.intellicus.integration.singlesignon.SingleSignOn;

import com.intellicus.integration.singlesignon.SingleSignOn Exception;

import com.intellicus.integration.singlesignon.UserInfo;

3. Make an object of SingleSignOn class for invoking the methods of this class.
4. Make an object of UserInfo class and set the user credentials using the setter methods provided by UserInfo class.

Constructor

public UserInfo()

Constructor

public UserInfo (String userId,String password,String orgId)

Parameters:

userId: User Id of the logged in user

password: Password of the logged in user.

orgId: Organization id of the logged in user

Constructor

public UserInfo (String userId, String orgId)

Parameters:

userId: User Id of the logged in user.

orgId: Organization id of the logged in user.

5. Pass this userInfo object to SingleSignOn class using the setUserInfo (userInfo) method.

public void setUserInfo(UserInfo userInfo)

Parameters:

- UserInfo: Object reference of UserInfo class.

6. Make an object of UserInfo class for Admin user info. Set the user credentials for Admin user in Intellicus in this object.
7. Set the IntellicusUrl.
8. Intellicus url can be read from property file or some database or repository.
9. Set the action code. Action code is required to specify which action admin user want to perform in Intellicus.

Action code can be:

ACTIVATE_USER

DELETE_USER

SUSPEND_USER

CREATE_USER

MODIFY_USER

S.No	Admin Activity	Action Code	Enum Provided by Intellicus

1.	Create a user: Admin can create a user at intellicus end.	CREATE_USER	Enums.ActionCodes.CREATE_USER
2.	Suspend a user: Admin can suspend an active user.	SUSPEND_USER	Enums.ActionCodes.SUSPEND_USER
3.	Activate a user: Admin can activate a suspended user.	ACTIVATE_USER	Enums.ActionCodes.ACTIVATE_USER
4.	Delete a user: Admin can delete an active or suspended user.	DELETE_USER	Enums.ActionCodes.DELETE_USER
5.	Modify a user: Admin can modify a user.	MODIFY_USER	Enums.ActionCodes.MODIFY_USER

10. Set the user role. User role specifies which type of role admin wants to assign to newly created user.
11. Call the callIntellicusControllerAPI() method of SingleSignOn class to send the request to Intellicus for Admin Activity.

Method:

This method calls Intellicus API and passes the user credentials to Intellicus.

public String callIntellicusControllerAPI() throws IOException

Returns:

String: Received Status message from Intellicus

S.No	Activity Status at Intellicus	Status Message	Enum Provided by Intellicus
1.	'user suspend' operation requested by admin is completed.	USER SUSPEND SUCCEEDED	Enums. ResponseMessages.USER_SUSPEND_ SUCCEEDED

S.No	Activity Status at Intellicus	Status Message	Enum Provided by Intellicus
2.	'user activate' operation requested by admin is completed.	USER ACTIVATION SUCCEEDED	Enums.ResponseMessages .USER_ACTIVATION_SUCCEEDED
3.	‘user suspend' operation requested by admin for an ‘already suspended’ user.	USER ALREADY SUSPENDED	Enums.ResponseMessages .USER_ALREADY_SUSPENDED
4.	'user activate' operation requested by admin for an ‘already active’ user.	USER ALREADY ACTIVATED	Enums.ResponseMessages .USER_ALREADY_ACTIVATED
5.	'user delete' operation requested by admin is completed.	USER DELETION SUCCEEDED	Enums.ResponseMessages .USER_DELETION_SUCCEEDED
6.	user to be activated/suspended/deleted does not exist at Intellicus.	USER DOES NOT EXIST	Enums.ResponseMessages .USER_DOES_NOT_EXIST
7.	Report Server is down.	COULD NOT CONNECT TO REPORT SERVER	Enums.ResponseMessages .COULD_NOT_CONNECT_TO_REPORT_SERVER
8.	User identification failed at Intellicus end.	AUTHENTICATION FAILED	Enums.ResponseMessages .AUTHENTICATION_FAILED
9.	Intellicus Repository database is down.	REPOSITORY DB IS DOWN	Enums.ResponseMessages .REPOSITORY_DB_IS_DOWN
10.	An unknown exception occurs at Intellicus.	REPORTING NOT AVAILABLE	Enums.ResponseMessages .REPORTING_NOT_AVAILABLE

S.No	Activity Status at Intellicus	Status Message	Enum Provided by Intellicus
11.	'user create' operation requested by admin is completed.	USER CREATION SUCCEEDED	Enums.ResponseMessages .USER_CREATION_SUCCEEDED
12.	'user create' operation requested by admin was for an 'already existing' user.	USER ALREADY EXIST	Enums.ResponseMessages .USER_ALREADY_EXIST
13.	'user modify'	USER MODIFICATION SUCCEEDED	Enums.ResponseMessages. USER_MODIFICATION_SUCCEEDED

Note: Please refer IntellicusSSOAdminActivity.java for sample code.
Path: <Install_Path>\SampleCodes\SingleSignOn\Java

Sample Code:

```
try
{
            String actionCode=request.getParameter("ACTION_CODE");

SingleSignOn singleSignOn=new SingleSignOn();
            //Set user credentials for user to be activated/deleted/suspended by
Admin user.
            // OR set the credentials for logged-in user for End -user requests.
            //user password is not required, if the authentication mode for
organization is "Host Application."
            //These credentials can be fetched from the data structure maintained
for the selected user.
String hostAppUserid="userId";
String hostAppOrgId="orgId";

            UserInfo userInfo=new UserInfo();

            // Set user credentials for admin user.
            // Admin user credentials are required if some request for admin
activity is raised.
            // Admin activities are like User Management, Database connection
creation/modification etc.
            // These can be read from any property file or from
repository/database.
String intellicusAdminUserId="Admin";   //This value can be read from any prperty
file or database.
```

```
String intellicusAdminOrgId="Intellica";//This value can be read from any prperty
file or database.
String intellicusAdminPassword="Admin"; //This value can be read from any prperty
file or database.

UserInfo adminUserInfo=new UserInfo();
             adminUserInfo.setUserId(intellicusAdminUserId);
             adminUserInfo.setOrgID(intellicusAdminOrgId);
             adminUserInfo.setPassword(intellicusAdminPassword);

             SingleSignOn.setAdminUserInfo(adminUserInfo);

             // Set the path for Intellicus Web application
             // This can be read from any property file or from
repository/database.

             singleSignOn.setIntellicusURL("http://localhost/intellicus");

//           This is for admin activities.
// Admin activity here deals with User Management at Intellicus.

if(actionCode!=null && actionCode.equals("ACTIVATE_USER"))
{
             singleSignOn.setActionCode(Enums.ActionCodes.ACTIVATE_USER);
              }
              else if(actionCode!=null && actionCode.equals("DELETE_USER"))
                {

             singleSignOn.setActionCode(Enums.ActionCodes.DELETE_USER);

              }
              else if(actionCode!=null && actionCode.equals("SUSPEND_USER"))
               {
                    singleSignOn.setActionCode(Enums.ActionCodes.SUSPEND_USER);

}
else if(actionCode!=null && actionCode.equals("CREATE_USER"))
               {
                    singleSignOn.setActionCode(Enums.ActionCodes.CREATE_USER);

}

             //Set the credentials for  User to be suspended
             userInfo.setUserId(hostAppUserid);
             userInfo.setOrgID(hostAppOrgId);
             singleSignOn.setUserInfo(userInfo);

             String statusMsg=singleSignOn.callIntellicusControllerAPI();

PrintWriter out=response.getWriter();
out.println(statusMsg);
}
```

```
catch(SingleSignOnException e)// if connection for the intellicusURL can not be
opened. Reason can be
                          //Intellicus url is wrong or Report Server is down.
{
e.printStackTrace();
             PrintWriter out=response.getWriter();
             out.println("Intellicus Web Application Not Available ");
             }
             catch(Exception e)
             {
                    PrintWriter out=response.getWriter();
                    out.println("Intellicus Web Application Not Available ");
              }
```

Host Application on .Net Platform

Implementation code can be written inside any ASPX of Host application.

1. Host application need to add intellicaSSO.dll in their application.
2. Host application need to import namespace Intellicus.Integration.SingleSignOn.
   using Intellicus.Integration.SingleSignOn;
3. Make an object of SingleSignOn class for invoking the methods of this class.
4. Make an object of UserInfo class and set the user credentials using the setter methods provided by UserInfo class.

Constructor

public UserInfo()

Constructor

public UserInfo (String userId,String password,String orgId)

Parameters:

userId : User Id of the logged in user

password: Password of the logged in user.

orgId : Organization id of the logged in user

Constructor

public UserInfo (String userId,String password,String orgId)

Parameters:

userId : User Id of the logged in user

orgId : Organization id of the logged in user

1. Set this userInfo object to UserInfo property of SingleSignOn class.
   singleSignOn.UserInfo = userInfo;
2. Make an object of UserInfo class for Admin user info. Set the user credentials for Admin user in Intellicus in this object.
3. Set the IntellicusUrl.
   Intellicus URL can be read from property file or some database or repository.
4. Set the action code. Action code is required to specify which action admin user want to perform in Intellicus.

Action code can be:

ACTIVATE_USER

DELETE_USER

SUSPEND_USER

CREATE_USER

S.No	Admin Activity	Action Code	Enum Provided by Intellicus
1.	Create a user: Admin can create a user at intellicus end.	CREATE_USER	Enums.ActionCodes.CREATE_USER
2.	Suspend a user: Admin can suspend an active user.	SUSPEND_USER	Enums.ActionCodes.SUSPEND_USER
3.	Activate a user: Admin can activate a suspended user.	ACTIVATE_USER	Enums.ActionCodes.ACTIVATE_USER
4.	Delete a user: Admin can delete an active or suspended user.	DELETE_USER	Enums.ActionCodes.DELETE_USER

5. Set the user role. User role specifies which type of role admin wants to assign to newly created user.
6. Call the callIntellicusControllerAPI () method of SingleSignOn class to send the request to Intellicus for Admin Activity.

Method:

This method calls Intellicus API and passes the user credentials to Intellicus. It throws SingleSignOnException.

public String callIntellicusControllerAPI()

Returns:

String: Received Status message from Intellicus

S.No	Activity Status at Intellicus	Status Message	Enum Provided by Intellicus
1.	'user suspend' operation requested by admin is completed.	USER SUSPEND SUCCEEDED	Enums. ResponseMessages.USER_SUSPEND_ SUCCEEDED
2.	'user activate' operation requested by admin is completed.	USER ACTIVATION SUCCEEDED	Enums.ResponseMessages .USER_ACTIVATION_SUCCEEDED
3.	‘user suspend' operation requested by admin for an ‘already suspended’ user.	USER ALREADY SUSPENDED	Enums.ResponseMessages .USER_ALREADY_SUSPENDED
4.	'user activate' operation requested by admin for an ‘already active’ user.	USER ALREADY ACTIVATED	Enums.ResponseMessages .USER_ALREADY_ACTIVATED
5.	'user delete' operation requested by admin is completed.	USER DELETION SUCCEEDED	Enums.ResponseMessages .USER_DELETION_SUCCEEDED

S.No	Activity Status at Intellicus	Status Message	Enum Provided by Intellicus
6.	user to be activated/suspended/deleted does not exist at Intellicus.	USER DOES NOT EXIST	Enums.ResponseMessages .USER_DOES_NOT_EXIST
7.	Report Server is down.	COULD NOT CONNECT TO REPORT SERVER	Enums.ResponseMessages .COULD_NOT_CONNECT_TO_REPORT_SERVER
8.	User identification failed at Intellicus end.	AUTHENTICATION FAILED	Enums.ResponseMessages .AUTHENTICATION_FAILED
9.	Intellicus Repository database is down.	REPOSITORY DB IS DOWN	Enums.ResponseMessages .REPOSITORY_DB_IS_DOWN
10.	An unknown exception occurs at Intellicus.	REPORTING NOT AVAILABLE	Enums.ResponseMessages .REPORTING_NOT_AVAILABLE
11.	'user create' operation requested by admin is completed.	USER CREATION SUCCEEDED	Enums.ResponseMessages .USER_CREATION_SUCCEEDED
12.	'user create' operation requested by admin was for an ‘already existing’ user.	USER ALREADY EXIST	Enums.ResponseMessages .USER_ALREADY_EXIST

Note: Please refer IntellicusSSOAdminActivity.aspx for sample code.
Path: <Install_Path>\SampleCodes\SingleSignOn\DotNet

Sample Code:

```
try
{
    String actionCode = "SUSPEND_USER";
```

```
    SingleSignOn singleSignOn = new SingleSignOn();

#region Creating UserInfo
//Set user credentials for user to be
//activated/deleted/suspended by Admin user.
//user password is not required, if the authentication mode
//for organization is "Host Application."
//These credentials can be fetched from the data structure
//maintained for the selected user.
    String hostAppUserid = "a";
    String hostAppOrgId = "k31";

//set the credentials for the user to be
//activated/deleted/suspended
    UserInfo userInfo =new UserInfo();
    userInfo.UserId = hostAppUserid;
    userInfo.OrgID = hostAppOrgId;
    singleSignOn.UserInfo = userInfo;

#endregion

#region Create AdminInfo

    // Set user credentials for admin user.
// Admin user credentials are required if some request for
//admin activity is raised.
// this admin user should be present at Intellicus with
//Superadmin system privileges.
// Admin activities are like User Management, Database
//connection creation/modification etc.
// These can be read from any property file or from
//repository/database.

//This value can be read from any prperty file or database.
String intellicusAdminUserId = "Admin";
    //This value can be read from any prperty file or database.
String intellicusAdminOrgId = "Intellica”;
//This value can be read from any prperty file or database.
    String intellicusAdminPassword = "Admin";

    UserInfo adminUserInfo =new UserInfo();
    adminUserInfo.UserId = intellicusAdminUserId;
    adminUserInfo.OrgID = intellicusAdminOrgId;
    adminUserInfo.Password = intellicusAdminPassword;

    SingleSignOn.AdminUserInfo = adminUserInfo;
#endregion

#region Set Intellicus Path and ActionCode
    // Set the path for Intellicus Web application
// This can be read from any property file or from
//repository/database.
singleSignOn.IntellicusURL="http://192.168.33.165/intellicusvss";
```

```
  if (actionCode != null)
  {
    if (actionCode.Equals("ACTIVATE_USER"))
       singleSignOn.ActionCode = Enums.ActionCodes.ACTIVATE_USER;
    if (actionCode.Equals("DELETE_USER"))
       singleSignOn.ActionCode = Enums.ActionCodes.DELETE_USER;
    if (actionCode.Equals("SUSPEND_USER"))
       singleSignOn.ActionCode = Enums.ActionCodes.SUSPEND_USER;
  }

#endregion

#region Get Status Message
   //call the Intellicus controller API to activate/delet/suspend
   //a user at Intellicus end.
   // It will return a status message both in case of operation
   //success or failure.
   String statusMsg = singleSignOn.callIntellicusControllerAPI();

   Response.Write(statusMsg);
   Response.Write("<br>");
   Response.Write(userInfo.Locale);
   Response.Write("<br>");
   Response.Write(adminUserInfo.Locale);

#endregion

}
catch (SingleSignOnException ex)
{
   Response.Write(ex.Message);
}
catch (Exception exc)
{
   Response.Write(exc.Message);;
}
```

Host Application on PHP Platform

Implementation code can be written inside any php or html files of Host application.

1. Host application need to add intellicaSSO.php in their application.
2. Host application file needs to include file intellicaSSO.php.
   include_once("intellicaSSO.php");
3. Make an object of SingleSignOn class for invoking the methods of this class.
4. Make an object of UserInfo class and set the user credentials using the setter methods provided by UserInfo class.
5. Pass this userInfo object to SingleSignOn class using the setUserInfo (userInfo) method.

public function setUserInfo($userInfo)

Parameters:

- UserInfo : Object reference of UserInfo class.

1. Make an object of UserInfo class for Admin user info. Set the user credentials for Admin user in Intellicus in this object.
2. Set the IntellicusUrl.
3. Intellicus URL can be read from property file or some database or repository.
4. Set the action code. Action code is required to specify which action admin user want to perform in Intellicus.

Action code can be:

ACTIVATE_USER

DELETE_USER

SUSPEND_USER

CREATE_USER

MODIFY_USER

S.No	Admin Activity	Action Code	Enum Provided by EnumsActionCodes class in intellicaSSO.php
1.	Create a user: Admin can create a user at intellicus end.	CREATE_USER	CREATE_USER

S.No	Admin Activity	Action Code	Enum Provided by EnumsActionCodes class in intellicaSSO.php
2.	Suspend a user: Admin can suspend an active user.	SUSPEND_USER	SUSPEND_USER
3.	Activate a user: Admin can activate a suspended user.	ACTIVATE_USER	ACTIVATE_USER
4.	Delete a user: Admin can delete an active or suspended user.	DELETE_USER	DELETE_USER
5.	Modify a user: Admin can modify the attributes of any user.	MODIFY_USER	MODIFY_USER

5. Set the user role. User role specifies which type of role admin wants to assign to newly created user.
6. Call the callIntellicusControllerAPI () method of SingleSignOn class to send the request to Intellicus for Admin Activity.

Method:

This method calls Intellicus API and passes the user credentials to Intellicus.

public function callIntellicusControllerAPI()

Returns:

String: Received Status message from Intellicus

S.No	Activity Status at Intellicus	Status Message	Enum Provided by Enums ResponseMessages class of intellicaSSO.php
1.	'user suspend' operation requested by admin is completed.	USER SUSPEND SUCCEEDED	USER_SUSPEND_SUCCEEDED

S.No	Activity Status at Intellicus	Status Message	Enum Provided by Enums ResponseMessages class of intellicaSSO.php
2.	'user activate' operation requested by admin is completed.	USER ACTIVATION SUCCEEDED	USER_ACTIVATION_SUCCEEDED
3.	‘user suspend' operation requested by admin for an ‘already suspended’ user.	USER ALREADY SUSPENDED	USER_ALREADY_SUSPENDED
4.	'user activate' operation requested by admin for an ‘already active’ user.	USER ALREADY ACTIVATED	USER_ALREADY_ACTIVATED
5.	'user delete' operation requested by admin is completed.	USER DELETION SUCCEEDED	USER_DELETION_SUCCEEDED
6.	User to be activated / suspended / deleted does not exist at Intellicus.	USER DOES NOT EXIST	USER_DOES_NOT_EXIST
7.	Report Server is down.	COULD NOT CONNECT TO REPORT SERVER	COULD_NOT_CONNECT_TO_REPORT_SERVER
8.	User identification failed at Intellicus end.	AUTHENTICATION FAILED	AUTHENTICATION_FAILED
9.	Intellicus Repository database is down.	REPOSITORY DB IS DOWN	REPOSITORY_DB_IS_DOWN
10.	An unknown exception occurs at Intellicus.	REPORTING NOT AVAILABLE	REPORTING_NOT_AVAILABLE

S.No	Activity Status at Intellicus	Status Message	Enum Provided by Enums ResponseMessages class of intellicaSSO.php
11.	'user create' operation requested by admin is completed.	USER CREATION SUCCEEDED	USER_CREATION_SUCCEEDED
12.	'user create' operation requested by admin was for an 'already existing' user.	USER ALREADY EXIST	USER_ALREADY_EXIST
13.	'user modify'	USER MODIFICATION SUCCEEDED	USER_MODIFICATION_SUCCEEDED

Note: Please refer IntellicusSSOAdminActivity.php for sample code.
Path: <Install_Path>\SampleCodes\SingleSignOn\PHP

Sample Code:

```
include_once("intellicaSSO.php");
try
{
            $sso=new SingleSignOn();

            // Set the path for Intellicus Web application
            // This can be read from any property file or from
repository/database.
            // In lbMode, give the URL for LB application,
            // else give the URL for Intellicus Web application.
            $sso->setIntellicusURL("http://192.168.33.165/intellicus");

            //Set user credentials for user to be activated/deleted/suspended by
Admin user.
            //user password is not required, if the authentication mode for
organization is "Host Application."
            //These credentials can be fetched from the data structure maintained
for the selected user.
            $userinfo=new UserInfo;
            $userinfo->setUserId("k1");
            $userinfo->setOrgID("ab");
            $sso->setUserInfo($userinfo);

            // Set user credentials for admin user.
```

```
            // Admin user credentials are required if some request for admin
activity is raised.
            // this admin user should be present at Intellicus with Superadmin
system privileges.
            // Admin activities are like User Management, Database connection
creation/modification etc.
            // These can be read from any property file or from
repository/database.
            $adminUserinfo=new UserInfo;
            $adminUserinfo->setUserId("Admin");
            $adminUserinfo->setPassword("Admin");
            $adminUserinfo->setOrgID("Intellica");
            $sso->setAdminUserInfo($adminUserinfo);

            //     This is for admin activities.
            // Admin activity here deals with User Management at Intellicus.
            $actionCode=$_REQUEST["ACTION_CODE"];

            if($actionCode=="ACTIVATE_USER")
            {
                  $sso->setActionCode(EnumsActionCodes::$ACTIVATE_USER);
            }
            elseif($actionCode=="DELETE_USER")
            {
                  $sso->setActionCode(EnumsActionCodes::$DELETE_USER);
            }
            elseif($actionCode=="SUSPEND_USER")
            {
                  $sso->setActionCode(EnumsActionCodes::$SUSPEND_USER);
            }
            elseif($actionCode=="CREATE_USER")
            {
                  $sso->setActionCode(EnumsActionCodes::$CREATE_USER);
            }

            echo $sso->callIntellicusControllerAPI();

}
catch(SingleSignOnException $e)
{
            print_r("Intellicus Web Application Not Available");
}
catch(Exception $e)
{           print_r("Intellicus Web Application Not Available");
}
```