

从第三方系统中单点登录Smartbi

1、概要说明

在多个系统集成时，为了方便用户使用和统一管理，一般都会要求多个系统实现单点登录。单点登录一般有如下几个步骤：

1. 用户信息同步。用户信息同步有多种方式，可参考 [与第三方系统用户集成](#)。
2. 第三方系统通过链接等方式打开 Smartbi 并传递用户名等信息完成登录，后文详述。
3. Smartbi 实现第三方的登录验证方法。具体请参考“[与第三方系统用户集成](#)”中的“[使用第三方系统的用户验证](#)”章节。

文档目录：

- 1、概要说明
- 2、第三方系统单点登录时的几种方法
 - 方法一：使用Smartbi浏览器端SDK登录
 - 方法二：使用LoginToken扩展包
 - 方法三：使用Smartbi服务端SDK登录
 - 方法四：在页面中使用Form登录
 - 方法五：CAS单点登录
- 3、注意事项
- 4、单点登录方式对比

相关文档：

- [与第三方系统资源集成](#)
- [使用第三方系统进行用户管理](#)

2、第三方系统单点登录时的几种方法

可以通过以下五种方法之一，在第三方系统中单点登录到 Smartbi 系统。请根据实际需要，选择其中一种登录方法。

• 方法一：使用Smartbi浏览器端SDK登录

在第三方系统中编写 JavaScript 代码调用 Smartbi 的 [浏览器端SDK](#) 方法进行登录。具体实现请参考：[使用浏览器端SDK单点登录到Smartbi](#)。

• 方法二：使用LoginToken扩展包

Smartbi 服务器加载LoginToken扩展包（[点击此处下载](#)），并在第三方系统中编写Java代码，调用Smartbi的服务器端SDK方法进行登录并生成登录Token，实现单点登录。

```
// URL
// SmartbiSDK60——

//
ClientConnector conn = new ClientConnector("http://localhost:8080/smartbi");
boolean open = conn.open("admin", "manager"); // tokennull
if(open){
    InvokeResult result = conn.remoteInvoke("LoginTokenModule", "generateLoginToken", new Object[]
{"demo"}); //
    System.out.println(result.getResult()); //result.getResult()demotoken
    conn.close();
}else{
    //admin
    System.out.println("");
}

// iframeSmartbihttp://localhost:8080/smartbi/vision/index.jsp?user=demo&password=<>smartbi
smartbiURLURL
```

加载这个扩展包后，还需要修改 Smartbi config.jsp 配置界面上的“用户管理”段的信息。

1. 将“加密类型”设置为：CLASS
2. 将“登录验证类”设置为：smartbi.logintoken.LoginTokenAuthntication

← → ↻ ⚠ 不安全 | 10.10.202.17:18600/smartbi/vision/config.jsp

SMARTBI 大数据分析 修改登录密码

配置文件 smartbi-config.xml 存放位置:

License文件存放位置:

☒ 文件路径: [在官网获取license文件](#)

☐ 服务器:

上传License文件:

用户管理

显示操作权限管理界面: ☒ 是 ☐ 否 初始值(否)

加密类型:

登录验证类:

ad.properties文件存放位置:

同步通知类(多个时用分号分隔):

第三方用户同步类:

显示修改个人密码的入口: ☒ 是 ☐ 否 初始值(是)



注意：若登录不成功，请检查。

1、扩展包已正常加载？

2、登录验证类填写正确？

3、生成token的代码正确（必须要通过拥有系统管理员角色的用户来登录，然后再生成其他用户的登录token，不能给自己生成token并登录，admin不走token验证）。

4、并且此logintoken.ext还实现普通用户单独访问smartbi的时候使用用户原本的密码访问，集成的时候可以使用用户原本的密码也可以使用一次性密码，建议是集成的时候使用一次性密码集成访问。

• 方法三：使用Smartbi服务端SDK登录

在第三方系统中编写 JSP 页面调用 Smartbi 的服务器端 SDK 进行登录。此方式适用于“在第三方系统中打开Smartbi资源”、“在第三方系统中新建Smartbi资源”和“在第三方中打开功能模块”，而不适用于从第三方系统链接转到Smartbi。具体操作步骤如下：

1. 参考开发接口中的[服务端SDK方法](#) 的使用步骤，把相关的 jar 包加载到您的 Java 项目的 classpath 中。
2. 编写登录代码，调用 SDK 中的相应方法。
3. 可以将得到的smartbiCookie 作为参数传入openresource.jsp等页面，必须转码后再传递，例如[http://server:8080/smartbi/vision/openresource.jsp?resid=<资源ID>&smartbiCookie=<%=java.net.URLEncoder.encode\(conn.getCookie\(\), "UTF-8"\)%>](http://server:8080/smartbi/vision/openresource.jsp?resid=<资源ID>&smartbiCookie=<%=java.net.URLEncoder.encode(conn.getCookie(),)

```

<%@ page contentType="text/html; charset=GBK" import="java.util.*,
    smartbi.sdk.ClientConnector,
    smartbi.sdk.service.user.UserManagerService" %>

<%
    request.setCharacterEncoding("GBK");
    String user = "admin";
    String password = "manager";
    String smartbiURL = "http://localhost:18080/smartbi";
    ClientConnector conn = new ClientConnector(smartbiURL);
    // smartbi
    boolean ret = conn.open(user, password);
    if (!ret) {
        %>
        <script>
            alert("!");
        </script>
        <%
    } else {
        %>
        <script>
            alert("!");
            // smartbicookiesmartbiCookiecookie
            var smartbiCookie = "<%=conn.getCookie()%>";
        </script>
        <%
    }
%>

```



如果 Smartbi 是集群部署模式，则不能采用本方法进行单点登录。

原因是ClientConnector对象是在jsp页面的java代码块中初始化的，其发起对象是第三方系统服务器，而集成资源是浏览器发起的，浏览器在使用该cookie时，不能保证跟服务器一样访问到同一个节点，在没有访问到同一个节点时，其cookie是无效的。

• 方法四：在页面中使用Form登录

在第三方系统中编写 JS 脚本实现登录，假设目标登录页面为 <http://localhost:18080/smartbi/vision/index.jsp>，登录脚本如下所示。

```

<form method="POST" id="formLogin" onsubmit="go();return false">
    <p><input type="text" name="user" size="20" value="admin"></p>
    <p><input type="text" name="password" size="20" value="manager"></p>
    <p><input type="text" name="surl" size="20" value="manager"></p>
    <p><input type="text" name="furl" size="20" value="manager"></p>
    <p><input type="submit" value="" name="B1"></p>
</form>
function go() {
    formLogin.action = "http://localhost:18080/smartbi/vision/login.jsp";
    formLogin.user.value = "admin";
    formLogin.password.value = "manager";
    formLogin.surl.value = " "; // http://
    formLogin.furl.value = " "; //
    formLogin.submit();
    return true;
}

```

前端登录后，需要有相应的注销代码

```
<form method="POST" id="formLogout" onsubmit="go();return false">
    <p><input type="text" name="surl" size="20" value="manager"></p>
    <p><input type="text" name="furl" size="20" value="manager"></p>
    <p><input type="submit" value="" name="B1"></p>
</form>
function go() {
    formLogout.action = "http://localhost:18080/smartbi/vision/logout.jsp";
    formLogout.surl.value = " "; // smartbiindex.jsp
    formLogout.furl.value = " "; // smartbiindex.jsp
    formLogout.submit();
    return true;
}
```

• 方法五：CAS单点登录

如果已有CAS服务器，要求进行CAS统一认证，可进行cas单点登录。参考“CAS单点登录”文档。

3、注意事项

- 如果不通过单点登录，那么在集成 Smartbi 资源时必须传入user、password 参数。
- 如果采用 Smartbi 的服务器端SDK方法进行单点登录，在集成资源时除了可以传递参数user、password之外，还可以单独传递参数smartbiCookie，此参数为从服务器端获取的cookie。获取此值的方法，见下面红色方框内代码。

```
18 %>
19 <script>
20     alert("登录成功!");
21     // 获取smartbi的cookie，在集成页面中除了传递参数user、password之外，还可以传递smartbiCookie值
22     var smartbiCookie = "<%=conn.getCookie()%>";
23 </script>
24 <%
```

- 如果Smartbi是集群部署模式，不能采用“方法三”进行单点登录，原因是ClientConnector对象是在jsp页面的java代码块中初始化的，其发起对象是第三方系统服务器，而集成资源是浏览器发起的，浏览器在使用该cookie时，不能保证跟服务器一样访问到同一个节点，在没有访问到同一个节点时，其cookie是无效的。

4、单点登录方式对比

上述提供的几种单点登录方式对比如下：

单点登录名称	方案简述	优点	缺点	适用系统
LoginToken扩展包登录	在第三方系统中编写Java代码，调用Smartbi的服务器端SDK方法进行登录并获取登录Token再使用Token登录。	token只能使用一次且60秒内有效，不会暴露用户的真实密码，支持集群，安全性高。	需要加载扩展包并修改登录验证类。	安全性要求高的系统。
页面中使用Form登录	在页面中提交form表单传递用户密码参数登录，并传递参数告知smartbi登录成功和失败跳转的地址。	集成方式简单。	用户名和密码暴露在前端，安全性差。	安全性要求不高的系统。
浏览器端SDK登录	通过JSLoader加载bof.usermanager.UserService的js方法登录。	简单方便，通用性高。	集成方式较麻烦，用户名和密码暴露在前端，安全性差，需要解决跨域问题。	安全性要求不高的系统。
服务器端SDK登录	在第三方系统中编写 JSP 页面调用 Smartbi 的服务器端 SDK获取Cookie进行登录。	不会暴露用户的真实密码，安全性高。	不支持集群部署，不支持从第三方系统链接转到Smartbi系统首页进行登录。	安全性要求高的系统，只适用于在第三方系统中打开和新建Smartbi资源以及在第三方中打开功能模块进行登录。
CAS单点登录	将Smartbi接入到统一登录认证平台进行登录。	安全性高，多系统集成时登录方便。	需要加载扩展包，需要修改扩展包配置文件和登录验证类。	用户已有CAS服务器的情况。