Single Sign-On

Published 2014-10-28 | (Compatible with SDK 2.5,3.5,4.5,5.0,5.1 and 2011,2012,2013,2014 models)

> Using Single Sign-On: basic communication approaches including registering, retrieving server information and reseting.

Contents

Single Sign-On (SSO), provided by the Application Manager, makes it possible for users to enter their account information only once. This means that users avoid the inconvenience of having to enter account information through the TV remote control whenever they use the application.

Application Manager provides the function by encrypting and saving the user's account information and sending it to the application through the SSO common module.

SSO operates as follows:
Generate a TV account
Create a TV account to use SSO. This is generated in the Application Manager (Settings->Samsung Smart TV-ID) and has a four-digit PIN number as the password.
Register Service Sites
Register the account information of service sites (for example: YouTube, Picasa) in the generated TV account. Note that the information is registered only when the entered account information is determined as valid. You can make the registration in Settings->Samsung Smart TV ID.
Sign-in to the TV account
Sign in to the TV account, created in Step 1, by pressing the red key on the initial screen. Users can register the account information on several service sites in a TV account. Sign-in enables the retrieval of account information on a service site from several applications.
Execute an application.
The Application Manager decides whether SSO is available when executing an application. If available, the account information is given.
Use account information on service sites.
In an application, the account information coexists with other information provided by the Application Manager. Thus, it is required to use the account information after parsing.

Sample application

> Note
>
> See: **Sample application**.

Registering Account Information on Service Sites

To register account information on a service site, you must first register the account information, and then confirm the validity of the account information.

To register account information on service sites:

1. Go to Settings->Samsung Smart TV ID. If you wish to display the names of the service sites, configure the config.xml file as shown below.
   <cpname>YourCompany</cpname>
   <login>y</login>
   Your Company item is now displayed in the list of service sites. If it is not, restart the TV. You may also need to reload the Application Manager.
2. Enter the account information on the service site, 'YourCompany'. However, the registration is completed only if there is a module confirming the validity of the account information entered. If the registration cannot be completed, you must generate the module.

   Validity Confirmation is to make sure that the entered account information is consistent with the ID and password which are available for sign in on the real service site. The information is valid, if the API can login to each service site.

   In Google, there is ClientLogin to judge validity. If sign in is successfully done and auth token is granted, the account information is regarded valid. Refer to the URL states below regarding ClientLogin.

   To confirm the validity of account information:
3. Enter the name of the JavaScript file whose config.xml file has the module. Write the name of JavaScript file which has the module to confirm account information validity in <cpauthjs> item. For example, if the file name is Auth11101000000.js , write Auth11101000000. The Application Manager executes the module including the file. File names must be unique for each application. Also, check the variables such as curWidget.id.
   <cpauthjs>Auth11101000000</cpauthjs>
4. Create a JavaScript file corresponding to the file name defined in the previous step. A module is written in this file. The file is embedded in the root of the directory containing index.html.
5. Draw up a module confirming the validity of the account information. This module is generated in the file created in the previous step; ensure that the format observes all regulations.
   Declare the object, whose name is the value entered in step 1.
   Add a 'checkAccount' method to the object.
   The checkAccount method confirms the validity of the account information on the service site, and delivers the results to the callback function transferred to the factor.

For example:

```
var Auth11101000000 = {}; // Object Generated

Auth11101000000.checkAccount = function (id, pw, fnCallback) { // checkAccount Method Added

    /**
     * The id and pw inputted from the application manager are transferred to the factor.
     * Check validity of id and pw, and then operate callback function.
     */
    // ...
    // Confirmation on Validity
    // ...
    // Deliver the results
    if (isValidAccount) {
        fnCallback ('TRUE');
    } else {
        fnCallback ('FALSE');
    }
}
```

The table below lists the values returned to the Callback function and their effect.

Return value	Description
'TRUE'	If the transferred ID and password are valid, return 'TRUE'. The Application Manager registers the ID and Password in the TV account.
'FALSE'	If the account information is invalid, return 'FALSE'. The Application Manager allows the user enter the account information again.
'ERROR'	If any error occurs while confirming validity, return 'ERROR'. After a while, the Application Manager allows the user to enter the account information again.
Others	Values other than these three values are processed as 'ERROR'.

Downloading Service Site Account Information

To get service site account information:

1. Configure the config.xml file as follows:
   ```
   <cpname>YourCompany</cpname>
   <login>y</login>
   ```
   This makes the application available for login with the service site as 'YourCompany'.
2. Login to the TV account. The account information registered in the account is available when the sign-in process to TV account is completed.
3. Execute the application. Register the account information on CP(YourCompany) in the designated TV account. Generally, several service sites are registered in a TV account; on the other hand, an application can get one service site relevant to the designated CP in step 1.
4. Parse the ID and password. The ID and password are transferred to the browser global variable window.location.search, once all the conditions are met. This variable includes circumstances information, and it must be parsed. For more information on window.location.search, see Retrieving Additional Information.

Reset function

The reset function executes initial modules when applications are deleted or initialized in Application Manager.

The Application Manager brings information of the installed application in config.xml file when deleting or initializing. Check if the application needs a module which initializes through the information in config.xml file. If the application has the module that initializes, bring information on the module, and execute it in accordance with rules and try initializing.

> Note

> The Application Manager does not obtain any return value but operates the module to initialize.

Resetting in config.xml

To reset the code in config.xml, add an xml element as shown below.

```
<deleteJS>MyReset</deleteJS>
```

> Note
>
> Without the deleteJS element, the application does not operate the initialization module.

Resetting Code in the Module

To reset the code in the module:

The file name should be the same as class name.

No factor.

The name of function is “reset”.

```
var MyReset= {
    ...
};

MyReset.reset = function() {
    ...
    ...
    alert("Reset Complete!");

}
```

XHR communication

The XHR object is the main tool used by Ajax applications for browser-server communication. This section gives an example of XHR communication in Smart TV applications.

The example below is an HTML code that includes the XHRExample.js file, and executes the XHRExample.onload() function when the 'load' event occurs.

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <title>XHR Example</title>
        <script type="text/javascript" src="$MANAGER_WIDGET/Common/API/TVKeyValue.js"></script>
        <script type="text/javascript" src="$MANAGER_WIDGET/Common/API/Widget.js"></script>
        <script type="text/javascript" src="XHRExample.js"></script>
    </head>
    <body onload="XHRExample.onload()">
        <div id="intro">XHR Example.</div>
    </body>
</html>
```

The JavaScript code is shown below. In the example, XHRExample is declared as an object and has the XHRObj variable in it. If the XHRObj variable already exists, XHRExample calls the destroy() method. If an XHR object is successfully created, set it and request the data. If the state of XHR changes to ‘4’, which means data reception is completed, the receiveData() function registered earlier in onreadystatechange is executed.

```
var XHRExample = {
        XHRObj : null
    },
    tvKey = new Common.API.TVKeyValue();
    widgetAPI = new Common.API.Widget();

XHRExample.onload = function () {
    widgetAPI.sendReadyEvent();
    var URL = "";       // Test URL here

    if (this.XHRObj != null) {
        this.XHRObj.destroy();
    }
    this.XHRObj = new XMLHttpRequest();

    if (this.XHRObj) {
        this.XHRObj.onreadystatechange = function () {
            if (XHRExample.XHRObj.readyState == 4) {
                XHRExample.recieveData();
            }
        }
        this.XHRObj.open("GET", URL, true);
        this.XHRObj.send(null);
    }
}

XHRExample.recieveData = function () {
    alert(this.XHRObj.responseText);
}
```

In the above example, note that:

The variable of XHRObj is in the XHRExample object.

The destroy() method is called before an object is assigned to XHRObj.

You may run out of memory if you continue to create and use XHR objects. Therefore, delete objects you used in the past and assign new ones when you use XHR. To free up objects used in the past, maintain an object pointer. For this, memorize and manage the variable dealing with XHR objects.

To remove an XHR object from memory, use its destroy() method. This is the method that the browser provides to allow using the Managing memory efficiently.

XHRObject.destroy	
Frees up the XHR object from the memory.	
Syntax	XHRObject.destroy()
Parameter	None
Return Value	None
Remarks	This function, which is executed to optimize the memory usage, is provided only by the browser for Samsung TV users.

Example	`if (this.XHRObj != null)`
	`  this.XHRObj.destroy();`
	`this.XHRObj = new XMLHttpRequest();`

In the example above, two common modules provided by the Application Manager are included. For more information, see Common Modules.