MSA Java API design
(For Microsoft)

Author:	Kula Shi
Distribution:	Public
Version:	1.3

Revision Chart

Revision	Version	Comments	Date	Author	Reviewer
01	1.0	Initial	2015-06-25	Kula Shi	Scar Ai
02	1.1	Change class to service.ms package as MS required	2015-07-10	Kula Shi	Torres
03	1.2	1. Change java APIs follows latest MSA document 2. Add annotations. 3. Remove signin and signoff as Torres said they are not provided for java applications.	2015-07-22	Kula Shi	
04	1.3	Add static keyword for getInstance method of class MSAManager	2015-08-17	Kula Shi	

Content

Revision Chart 2
1 *Java MSA APIs* 3
1.1 MASManager.java 3
1.2 MSAListener.java 4
1.3 MSAUserInfo.java 5

1 Java MSA APIs

Java MSA (Microsoft Accounts) APIs design follows platform MSA API document (MS7701_MSA_Interface_v0.2.docx). Here three classes are introduced to fulfil MSA requirement.

1.1 MASManager.java

This class is similar as MSA/SSO module of platform side. It mainly uses for Java applications get token, refresh token and get user information when authrization is required on runtime. But it does not really communicate with MSA/SSO Authorization server, instead, it more like a wrapper of platform MSA manager module for Java applications. And base on this, Java applications can access resource server like native applications.
This class is designed as singleton pattern. All Java applications only need one object instance to coordinate actions across the MSA/SSO.

Package:

```
    com.ms.service.msa;
```

Class:

```
    public final  class MSAManager extends java.lang.Object;
```

Fields:

```
    public static final int MSA_OK = 0;
    public static final int MSA_FAILED = -1;
    public static final int MSA_TOKEN = -2;
    public static final int MSA_NO_TOKEN = -3;

    /* pre-defined MSA application types */
    public static final int MSA_APP_TYPE_CONTACT_SYNC = 0;
    public static final int MSA_APP_TYPE_CAMERA_ROLL = 1;
    public static final int MSA_APP_TYPE_EMAIL = 2;
    public static final int MSA_APP_TYPE_SKYPE = 3;
```

Methods:

```
    /**
     * Generate singleton instance of MSAManager.
     * @return singleton instance of MSAManager.
    */
    public static synchronized MSAManager getInstance();

    /**
     * Get MSA token tontent. This API is not  blocked, the  token content will be
    *  updated by the registed listener.
     * @param login, true means show UI, otherwise don't show UI.
     * @param appType, the MSA applicatoin type.
     * @param listener, MSA listener will used to get token content once registed.
     * @return MSA_OK means success
     *              MSA_FAILED means get token failure.
     *              MSA_TOKEN means user does not login.
```

```
 */
public int getToken(boolean  login,int appType, MSAListener listener);

/**
 * Refresh MSA token tontent. It’s similar to getToken() API.
* @param appType, the MSA applicatoin type.
* @param listener, MSA listener will used to get updated token content once registed.
 * @return MSA_OK means success
 *              MSA_FAILED means get token failure.
 *              MSA_TOKEN means user does not login.
 */
public int refreshToken(int appType, MSAListener listener);

/**
 * Get user information. This API return an instance of MSAUserInfo.
* @return an instance of MSAUserInfo.
 */
public MSAUserInfo getUserInfo();
```

1.2 MSAListener.java

This class is used to listen the callback of token update, including get token and refresh token. So when one Java application calls getToken or refreshToken, it must give a listener to get the token update. Otherwise, the token content will be ingored. It’s similar to callback function of platform side.
Application MUST implement this interface once it needs to get token content by calling ***MSAManager.getToken()*** or ***MSAManager.refreshToken()***.

Package:

com.ms.service.msa;

Class:

*public interface **MSAListener**;*

Fields:

None

Methods:

```
/**
 * After Java applications get/refresh token, token content will be updated by this API
* if they register listener.
* @param result, token update result,
*              MSAManager.MSA_OK means success.
*              MSAManager.MSA_FAILED means get/refresh token failure.
*              MSAManager.MSA_NO_TOKEN means user needs to login.
*@param buf, the token content.
*@param bufLen, size of token content in bytes.
 */
public void MSATokenUpdate(int result, byte[] buf, int bufLen);
```

1.3 MSAUserInfo.java

This class is used to store the user information once application call ***getUserInfo()*** from ***MSAManager***. The class only can be instanced by ***MSAManager,*** Java application can read user data by using APIs in this class.

Package:

com.ms.service.msa;

Class:

*public final class **MSAUserInfo** extends java.lang.Object;*

Fields:

None

Methods:

*public String **getId**();*
*public String **getName**();*
*public String **getFirstName**();*
*public String **getLastName**();*
*public String **getLink**();*
*public String **getGender**();*
*public String **getLocale**();*
*public String **getUpdatedTime**();*
*public String **getClientId**();*