BaroPAM integration API(Java-en)

Nurit Co., Ltd.

Lee Jongil

January 20, 2024

1. Preparation before using the integration API

The one-time authentication key, the authentication code used by BaroPAM, is written based on Java, so the latest JDK 6.x or higher must be installed. If it is not installed, you need to install the latest JDK.

Verify JDK installation)

```
[root]# rpm -qa | grep java
java-1.4.2-gcj-compat-devel-1.4.2.0-40jpp.115
java-1.7.0-openjdk-javadoc-1.7.0.131-2.6.9.0.el5_11
java-1.4.2-gcj-compat-1.4.2.0-40jpp.115
java-1.4.2-gcj-compat-javadoc-1.4.2.0-40jpp.115
bsh-javadoc-1.3.0-9jpp.1
tzdata-java-2016j-1.el5
java-1.6.0-openjdk-devel-1.6.0.41-1.13.13.1.el5_11
java-1.7.0-openjdk-src-1.7.0.131-2.6.9.0.el5_11
java-1.4.2-gcj-compat-src-1.4.2.0-40jpp.115
java-1.7.0-openjdk-1.7.0.131-2.6.9.0.el5_11
java-1.7.0-openjdk-demo-1.7.0.131-2.6.9.0.el5_11
java-1.4.2-gcj-compat-devel-1.4.2.0-40jpp.115
xmlrpc-javadoc-2.0.1-3jpp.1
gcc-java-4.1.2-55.el5
java-1.6.0-openjdk-1.6.0.41-1.13.13.1.el5_11
java-1.7.0-openjdk-devel-1.7.0.131-2.6.9.0.el5_11
```

Check the JDK installation directory)

```
[root]# env | grep JAVA_HOME
JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.121.x86_64
```

Java version check)

```
[root]# java -version
java version "1.7.0_121"
OpenJDK Runtime Environment (rhel-2.6.8.1.el5_11-x86_64 u121-b00)
OpenJDK 64-Bit Server VM (build 24.121-b00, mixed mode)
```

2. BaroPAM integration API

2.1 Login screen

1) BaroPAM login screen example)

2) Authentication key verification part

The API for verifying the one-time authentication key entered in the password field when logging in to the application is provided as "**barokey.jar**".

You can locate "**barokey.jar**" in the lib directory of WAS (Web Application Server) or set the class-path to include the directory where "**barokey.jar**" exists.

```
[root] /home/tomcat/webapps/ROOT/WEB-INF/lib > ls -al
합계 4908
drwxr-xr-x 2 root root    4096  5 월  8 11:25 .
drwxr-xr-x 5 root root    4096  5 월  9 15:12 ..
-rw------- 1 root root     116  3 월 13  2015 .bash_history
-rw-r--r-- 1 root root   26074  6 월 20 20:49 barokey.jar
-rw-r--r-- 1 root root   57779  5 월 24  2011 commons-fileupload-1.2.1.jar
```

```
-rw-r--r-- 1 root root   109043  5 월 24  2011 commons-io-1.4.jar
-rw-r--r-- 1 root root    60841  5 월 24  2011 commons-logging-1.1.1.jar
-rw-r--r-- 1 root root    26520  5 월 24  2011 commons-logging-adapters-1.1.1.jar
-rw-r--r-- 1 root root    56404  5 월 24  2011 cos.jar
-rw-r--r-- 1 root root   313898  5 월 24  2011 dom4j-1.6.1-goldkeby.jar
-rw-r--r-- 1 root root    19679  6 월  4  2014 gcm-server.jar
-rw-r--r-- 1 root root   341207 10 월  6  2008 j2ssh-common-0.2.9.jar
-rw-r--r-- 1 root root   355141 10 월  6  2008 j2ssh-core-0.2.9.jar
-rw-r--r-- 1 root root   110582 10 월  6  2008 j2ssh-dameon-0.2.9.jar
-rw-r--r-- 1 root root   456805  5 월  6  2016 j2ssh-maverick-1.5.5.jar
-rw-r--r-- 1 root root   258160 10 월 27  2011 jai_codec.jar
-rw-r--r-- 1 root root  1900631 10 월 27  2011 jai_core.jar
-rw-r--r-- 1 root root   464938  3 월  6  2012 jimiproclasses-sabisung.jar
-rw-r--r-- 1 root root    23737  6 월  4  2014 json-simple-1.1.1.jar
-rw-r--r-- 1 root root    30202  5 월 24  2011 json.jar
-rw-r--r-- 1 root root   312603  1 월 25  2012 twitter4j-core-2.2.5.jar
```

When an exception occurs when verifying a one-time authentication key, a stack trace is provided to the console with a detailed exception to the standard output (Standard output) so that the cause can be easily identified.

```
try {

    ......

} catch (final Exception e) {
    e.printStackTrace();
} finally {
}
```

When logging into an application, you must add the user phone number (USER_PHONE), one-time authentication key generation cycle (CYCLE_TIME), and login final time (LOGIN_TIME) columns.

```
USER_PHONE          VARCHAR2(50)  NOT NULL ,
CYCLE_TIME          VARCHAR2(2)   DEFAULT '30' ,
LOGIN_TIME          VARCHAR2(10) DEFAULT '0' ,
```

The reason for adding the login final time is to prevent reuse and man-in-the-middle attacks by limiting the ability of only one user to log in with the same login-ID within the one-time authentication key generation cycle. If the login is successful, In this case, the last login time must be updated.

Insert the following code into a program that verifies the one-time authentication key, which is the password entered when logging in to the application.

① When to use login-id

```
...
import com.barokey.barokey;
...
Login-ID validation is checked and authentication key verification module is called only
in case of success.
...
boolean bauth_key = barokey.verifyKEYL(String login_id, String phone_no, String cycle_time,
tkey);
boolean bauth_key = barokey.verifyKEYL(String login_id, String phone_no, String cycle_time,
String key_method, String tkey);

if (bauth_key == true) {
    // Verification success
} else {
    // Verification failure
}
...
```

Parameter	Description	Etc
login_id	Set the ID entered in the Login-ID field of the login screen.	
phone_no	Set smartphone numbers for each user only by numbers.	
cycle_time	Set the generation cycle (3~60 seconds) of one-time authentication key specified for each user.	
key_method	Set the one-time authentication key authentication method (app1, app256, app384, app512: app).	
tkey	Set the one-time authentication key entered in the password on the login screen.	

If the generation period of the smart phone number for each user and the one-time authentication key designated for each individual is different from the generator of the one-time authentication key, verification may fail because the one-time authentication key is different. You must match the information.

예)

```
<%@ page contentType="text/html; charset=UTF-8" language="java" pageEncoding="UTF-8" %>
<%@include file="./inc_common.jsp" %>
<%@include file="./inc_session.jsp" %>
<% request.setCharacterEncoding("utf-8"); %>
<% response.setContentType("text/html; charset=utf-8"); %>
<%
/*-------------------------------------------------------------------------*/
/* Variable declaration and initialization.                                */
/*-------------------------------------------------------------------------*/
int     ii = 0, jj = 0, kk = 0, ll = 0;                 // Index

CommonLib clib = new CommonLib();                       // CommonLib
LogDAO    ldao = new LogDAO();                          // LogDAO
UserDAO   udao = new UserDAO();                         // UserDAO
BaroDTO   bdto = new BaroDTO();                         // BaroDTO

String  HTML       = "";                                // HTML
String  msg        = "";                                // msg
```

```
String  value      = "";                                    // value
String  result     = "";                                    // Result
String  login_time = "";                                    // Last login time
boolean bauth_key  = false;                                 // Verification result
/*-------------------------------------------------------------------------*/
/* Get data from Request (user information).                               */
/*-------------------------------------------------------------------------*/
String  auth_key   = clib.strDefault(request.getParameter("auth_key"  ), "" );
String  user_email = clib.strDefault(request.getParameter("user_email"), "" );
String     ip_addr          =   clib.strDefault(request.getParameter("ip_addr"        ),
request.getRemoteAddr());

logger.info("(result_user.jsp)Starting.....................");
String  param = request.getServerName() + request.getRequestURI()
              + "?remote_addr="         + request.getRemoteAddr()
              + "&ip_addr="             + ip_addr
              + "&user_email="          + user_email
              + "&auth_key="            + auth_key
              ;
logger.info(param);
/*-------------------------------------------------------------------------*/
/* Begin.                                                                  */
/*-------------------------------------------------------------------------*/
try {
    /*---------------------------------------------------------------------*/
    /* User information inquiry.                                           */
    /*---------------------------------------------------------------------*/
    bdto = udao.reads(user_email);
    /*---------------------------------------------------------------------*/
    /* If user information exists.                                         */
    /*---------------------------------------------------------------------*/
    if (bdto != null) {
        /*-----------------------------------------------------------------*/
        /* If you are not using it.                                        */
        /*-----------------------------------------------------------------*/
        if ("N".equals(bdto.getUse_yn())) {
            HTML = "USE_ERR";
        /*-----------------------------------------------------------------*/
        /* If you are a user.                                               */
        /*-----------------------------------------------------------------*/
        } else {
            /*-------------------------------------------------------------*/
            /* Login last time Edit.                                        */
            /*-------------------------------------------------------------*/
            login_time = Long.toString(barokey.get_logintime(bdto.getCycle_time()));
            /*-------------------------------------------------------------*/
            /* When the last login time is greater than the creation cycle.  */
            /*-------------------------------------------------------------*/
            if (Long.parseLong(login_time) > Long.parseLong(bdto.getLogin_time())) {
                /*---------------------------------------------------------*/
                /* Authentication key verification.                        */
                /*---------------------------------------------------------*/
                bauth_key     =     barokey.verifyKEYL(user_email,     bdto.getUser_phone(),
bdto.getCycle_time(), auth_key);
                /*---------------------------------------------------------*/
```

```
                /* Authentication key verification (success).                */
                /*-----------------------------------------------------------*/
                if (bauth_key == true) {
                    /*-------------------------------------------------------*/
                    /* Cookie creation.                                      */
                    /*-------------------------------------------------------*/
                    session = request.getSession(true);
                    request.getSession().setAttribute("SES_USER_NO"     ,  bdto.getUser_no
());
                    request.getSession().setAttribute("SES_USER_NAME"  ,  bdto.getUser_name
());
                    request.getSession().setAttribute("SES_USER_DEPT"  ,  bdto.getUser_dept
());
                    request.getSession().setAttribute("SES_USER_TITLE",
bdto.getUser_title());
                    request.getSession().setAttribute("SES_USER_EMAIL",
bdto.getUser_email());
                    request.getSession().setAttribute("SES_USER_PHONE",
bdto.getUser_phone());
                    request.getSession().setAttribute("SES_ADMIN_YN"    ,  bdto.getAdmin_yn
());
                    /*-------------------------------------------------------*/
                    /* Update login last time.                               */
                    /*-------------------------------------------------------*/
                    udao.updateLoginTime(user_email, login_time);
                    HTML = bdto.getUser_no();
                /*-----------------------------------------------------------*/
                /* Authentication key verification (failure).                */
                /*-----------------------------------------------------------*/
                } else {
                    HTML = "KEY_ERR";
                }
            /*---------------------------------------------------------------*/
            /* When the login final time is less than or equal to the creation cycle. */
            /*---------------------------------------------------------------*/
            } else {
                HTML = "KEY_ERR";
            }
        }
        /*-------------------------------------------------------------------*/
        /* Insert log information.                                           */
        /*-------------------------------------------------------------------*/
        bdto.setIp_addr(ip_addr);
        ldao.create(bdto);
    /*-----------------------------------------------------------------------*/
    /* When user information does not exist.                                 */
    /*-----------------------------------------------------------------------*/
    } else {
        HTML = "ID_ERR";
    }
    json.put("HTML", HTML);
    out.println(json);
/*---------------------------------------------------------------------------*/
/* Handling exceptions.                                                      */
/*---------------------------------------------------------------------------*/
```

```
} catch(Exception e) {
    logger.info("Exception = [" + e + "]");
    e.printStackTrace();
/*-------------------------------------------------------------------------*/
/* Finally.                                                                */
/*-------------------------------------------------------------------------*/
} finally {
    logger.info("(result_user.jsp)Ending.....................");
}
%>
```

② When using a secure key

```
...
import com.barokey.barokey;
...
Login-ID validation is checked and authentication key verification module is called only in case of success.
...
boolean bauth_key = barokey.verifyKEYP(String secure_key, String cycle_time, tkey);
boolean bauth_key = barokey.verifyKEYP(String secure_key, String cycle_time, String key_method, String tkey);

if (bauth_key == true) {
    // Verification success
} else {
    // Verification failure
}
...
```

Parameter	Description	Etc
secure_key	Set the secure key provided by the vendor.	
cycle_time	Set the generation cycle (3~60 seconds) of one-time authentication key specified for each user.	
key_method	Set the one-time authentication key authentication method (app1, app256, app384, app512: app).	
tkey	Set the one-time authentication key entered in the password on the login screen.	

예)

```
<%@ page contentType="text/html; charset=UTF-8" language="java" pageEncoding="UTF-8" %>
<%@include file="./inc_common.jsp" %>
<%@include file="./inc_session.jsp" %>
<% request.setCharacterEncoding("utf-8"); %>
<% response.setContentType("text/html; charset=utf-8"); %>
<%
/*-------------------------------------------------------------------------*/
/* Variable declaration and initialization.                                */
/*-------------------------------------------------------------------------*/
int     ii = 0, jj = 0, kk = 0, ll = 0;                 // Index

CommonLib clib = new CommonLib();                       // CommonLib
LogDAO    ldao = new LogDAO();                          // LogDAO
```

```
UserDAO   udao = new UserDAO();                          // UserDAO
BaroDTO   bdto = new BaroDTO();                          // BaroDTO

String  HTML       = "";                                 // HTML
String  msg        = "";                                 // msg
String  value      = "";                                 // value
String  result     = "";                                 // Result
String  login_time = "";                                 // Last login time
boolean bauth_key  = false;                              // Verification result
/*-------------------------------------------------------------------------*/
/* Get data from Request (user information).                               */
/*-------------------------------------------------------------------------*/
String  auth_key   = clib.strDefault(request.getParameter("auth_key"  ), "" );
String  user_email = clib.strDefault(request.getParameter("user_email"), "" );
String     ip_addr           =   clib.strDefault(request.getParameter("ip_addr"        ),
request.getRemoteAddr());

logger.info("(result_user.jsp)Starting.....................");
String  param = request.getServerName() + request.getRequestURI()
              + "?remote_addr="         + request.getRemoteAddr()
              + "&ip_addr="             + ip_addr
              + "&user_email="          + user_email
              + "&auth_key="            + auth_key
              ;
logger.info(param);
/*-------------------------------------------------------------------------*/
/* Begin.                                                                  */
/*-------------------------------------------------------------------------*/
try {
    /*---------------------------------------------------------------------*/
    /* User information inquiry.                                           */
    /*---------------------------------------------------------------------*/
    bdto = udao.reads(user_email);
    /*---------------------------------------------------------------------*/
    /* If user information exists.                                         */
    /*---------------------------------------------------------------------*/
    if (bdto != null) {
        /*-----------------------------------------------------------------*/
        /* If you are not using it.                                        */
        /*-----------------------------------------------------------------*/
        if ("N".equals(bdto.getUse_yn())) {
            HTML = "USE_ERR";
        /*-----------------------------------------------------------------*/
        /* If you are a user.                                               */
        /*-----------------------------------------------------------------*/
        } else {
            /*-------------------------------------------------------------*/
            /* Login last time Edit.                                        */
            /*-------------------------------------------------------------*/
            login_time = Long.toString(barokey.get_logintime(bdto.getCycle_time()));
            /*-------------------------------------------------------------*/
            /* When the last login time is greater than the creation cycle.  */
            /*-------------------------------------------------------------*/
            if (Long.parseLong(login_time) > Long.parseLong(bdto.getLogin_time())) {
                /*---------------------------------------------------------*/
```

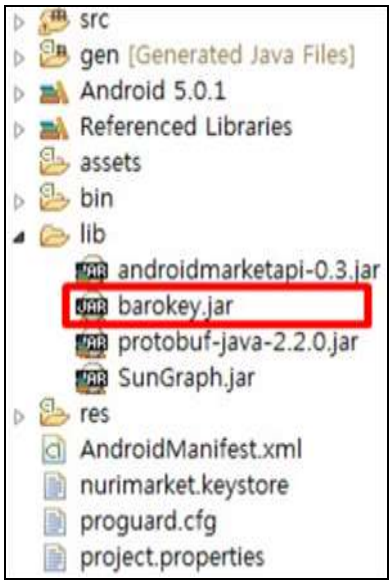
```
                /* Authentication key verification.                          */
                /*-----------------------------------------------------------*/
                bauth_key = barokey.verifyKEYP(bdto.getSecure_key(), bdto.getCycle_time(),
auth_key);
                /*-----------------------------------------------------------*/
                /* Authentication key verification (success).                */
                /*-----------------------------------------------------------*/
                if (bauth_key == true) {
                    /*-------------------------------------------------------*/
                    /* Cookie creation.                                      */
                    /*-------------------------------------------------------*/
                    session = request.getSession(true);
                    request.getSession().setAttribute("SES_USER_NO"     , bdto.getUser_no
());
                    request.getSession().setAttribute("SES_USER_NAME"  , bdto.getUser_name
());
                    request.getSession().setAttribute("SES_USER_DEPT"  , bdto.getUser_dept
());
                    request.getSession().setAttribute("SES_USER_TITLE",
bdto.getUser_title());
                    request.getSession().setAttribute("SES_USER_EMAIL",
bdto.getUser_email());
                    request.getSession().setAttribute("SES_USER_PHONE",
bdto.getUser_phone());
                    request.getSession().setAttribute("SES_ADMIN_YN"    , bdto.getAdmin_yn
());
                    /*-------------------------------------------------------*/
                    /* Update login last time.                               */
                    /*-------------------------------------------------------*/
                    udao.updateLoginTime(user_email, login_time);
                    HTML = bdto.getUser_no();
                /*-----------------------------------------------------------*/
                /* Authentication key verification (failure).                */
                /*-----------------------------------------------------------*/
                } else {
                    HTML = "KEY_ERR";
                }
            /*---------------------------------------------------------------*/
            /* When the login final time is less than or equal to the creation cycle. */
            /*---------------------------------------------------------------*/
            } else {
                HTML = "KEY_ERR";
            }
        }
        /*-------------------------------------------------------------------*/
        /* Insert log information.                                           */
        /*-------------------------------------------------------------------*/
        bdto.setIp_addr(ip_addr);
        ldao.create(bdto);
    /*-----------------------------------------------------------------------*/
    /* When user information does not exist.                                 */
    /*-----------------------------------------------------------------------*/
    } else {
        HTML = "ID_ERR";
    }
```

```
    json.put("HTML", HTML);
    out.println(json);
/*-------------------------------------------------------------------------*/
/* Handling exceptions.                                                    */
/*-------------------------------------------------------------------------*/
} catch(Exception e) {
    logger.info("Exception = [" + e + "]");
    e.printStackTrace();
/*-------------------------------------------------------------------------*/
/* Finally.                                                                */
/*-------------------------------------------------------------------------*/
} finally {
    logger.info("(result_user.jsp)Ending.....................");
}
%>
```

2.2 For Android phone

1) Authentication key generator part

The API that creates the one-time authentication key to be entered in the password field when logging in to the application is provided as "**barokey.jar**", and when using Eclipse or Android studio, "**barokey.jar**" must be located in the libs directory.



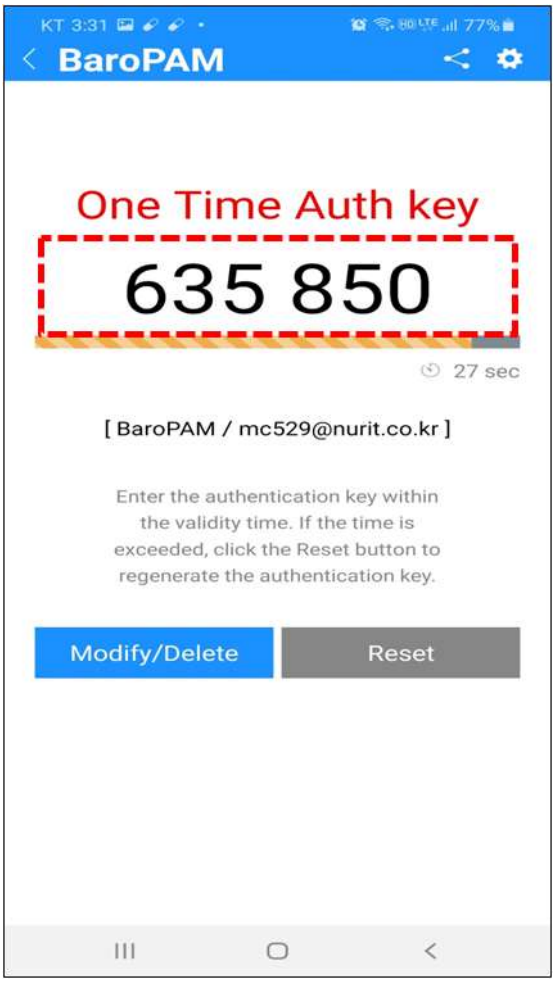
Insert the following code into a program that creates a one-time authentication key that is a password to enter when logging in to the application.

```
...
import com.barokey.barokey;
...
String tkey = barokey.generateKEYL(String login_id, String phone_no, String cycle_time);
```

```
...
```

Parameter	Description	Etc
login_id	Set the ID entered in the Login-ID field of the login screen.	
phone_no	Use the TelephonyManager class to set the smart phone number obtained from inside the app.	
cycle_time	Set the generation cycle (3~60 seconds) of the one-time authentication key specified for each individual. If the generation period of the one-time authentication key designated for each individual is different, the one-time authentication key may be generated differently.	

Screen example)



Screen Layout Example)

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@color/bg_body_default"
```

```
    android:orientation="vertical">

    <include
        android:id="@+id/inc_header"
        layout="@layout/inc_header"
        android:layout_width="fill_parent"
        android:layout_height="@dimen/head_height" />

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginLeft="@dimen/body_margin_right_default"
        android:layout_marginRight="@dimen/body_margin_right_default"
        android:layout_marginTop="@dimen/head_height">

        <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:id="@+id/body_frame"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="81dip"
                android:padding="10dp"
                android:text="@string/tv_key_vc"
                android:textColor="@color/text_body_default"
                android:textSize="20dip" />

            <TextView
                android:id="@+id/tv_auth_key"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="150dip"
                android:background="@android:color/transparent"
                android:ems="10"
                android:gravity="center"
                android:imeOptions="actionGo"
                android:inputType="text"
                android:maxLength="8"
                android:nextFocusDown="@+id/btn_login"
                android:singleLine="true"
                android:text=""
                android:textAppearance="?android:attr/textAppearanceLarge"
                android:textColor="@color/text_body_default"
                android:textSize="65dip" />

            <TextView
                android:layout_width="fill_parent"
                android:layout_height="1dip"
                android:layout_gravity="center_horizontal"
                android:layout_marginLeft="50dip"
                android:layout_marginRight="50dip"
```

```
        android:layout_marginTop="230dip"
        android:background="@color/line_text_under"
        android:visibility="invisible" />

    <com.beardedhen.androidbootstrap.BootstrapProgressBar
        android:id="@+id/progressBar"
        android:layout_width="fill_parent"
        android:layout_height="12dip"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="240dip"
        app:animated="true"
        app:bootstrapBrand="warning"
        app:bootstrapProgress="100"
        app:striped="true" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="260dip"
        android:orientation="horizontal">

        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1" />

        <ImageView
            android:layout_width="15dip"
            android:layout_height="15dip"
            android:layout_gravity="center_vertical|right"
            android:background="@drawable/ico_countdown" />

        <TextView
            android:id="@+id/tv_remainTime"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="right|center_vertical"
            android:paddingLeft="10dip"
            android:textColor="@color/text_body_guide"
            android:textSize="17dip" />

    </LinearLayout>

    <TextView
        android:id="@+id/tv_system_nm"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="315dip"
        android:text=""
        android:textColor="@color/text_body_default"
        android:textSize="18dip" />

    <TextView
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="380dip"
        android:text="@string/tv_key_msg_1"
        android:textColor="@color/text_body_guide"
        android:textSize="18dip" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="405dip"
        android:text="@string/tv_key_msg_2"
        android:textColor="@color/text_body_guide"
        android:textSize="18dip" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="430dip"
        android:text="@string/tv_key_msg_3"
        android:textColor="@color/text_body_guide"
        android:textSize="18dip" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="490dip"
        android:orientation="horizontal">

        <Button
            android:id="@+id/btn_update"
            android:layout_width="fill_parent"
            android:layout_height="@dimen/btn_height_default"
            android:layout_weight="1"
            android:background="@drawable/btn_default_drawable"
            android:text="@string/btn_upd_del"
            android:textColor="@color/white"
            android:textSize="20dip" />

        <TextView
            android:layout_width="6dip"
            android:layout_height="1dip"
            android:layout_gravity="center_horizontal"
            android:background="@android:color/transparent" />

        <Button
            android:id="@+id/btn_reset"
            android:layout_width="fill_parent"
            android:layout_height="@dimen/btn_height_default"
            android:layout_weight="1"
            android:background="@drawable/btn_default_drawable"
            android:enabled="false"
            android:text="@string/btn_reset"
```

```
                    android:textColor="@color/white"
                    android:textSize="20dip" />

            </LinearLayout>

        </FrameLayout>

    </ScrollView>

</FrameLayout>
```

Program example)

```
package com.baro.otp.info;

import android.Manifest;
import android.annotation.SuppressLint;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.os.Vibrator;
import android.support.v4.app.ActivityCompat;
import android.telephony.TelephonyManager;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.TextView;

import com.baro.common.base.BaseActivity;
import com.baro.common.base.BaseInterface;
import com.baro.common.setting.SettingACT;
import com.baro.common.util.Util;
import com.baro.pam.R;
import com.barokey.barokey;
import com.beardedhen.androidbootstrap.BootstrapProgressBar;

import java.util.Date;

public class OTPCreateACT extends BaseActivity implements BaseInterface, OnClickListener {
    //public class OTPCreateACT extends AppCompatActivity implements BaseInterface,
OnClickListener {
    private Button   btn_setting, btn_share, btn_close, btn_reset, btn_update;
    private TextView tv_auth_key;
    private TextView tv_remainTime;
    private BootstrapProgressBar progressBar;
    private TextView tv_system_nm;
    private String    intent_reg_dt = "", intent_system_nm = "", intent_login_id = "",
intent_cycle_time = "";
```

```
    private String   PhoneNumber = "", SerialNumber = "", AndroID = "", MacAddr = "";

    private long     createdMillis, remainingSec;

    private static final int MESSAGE_REFRESH_REMAINING_SECOND = 101;
    private static final int SENDMESSAGE_INTERVAL = 250;

    private     String[]     permission_list     =     {     Manifest.permission.INTERNET,
Manifest.permission.ACCESS_WIFI_STATE,           Manifest.permission.ACCESS_NETWORK_STATE,
Manifest.permission.READ_EXTERNAL_STORAGE,     Manifest.permission.WRITE_EXTERNAL_STORAGE,
Manifest.permission.READ_PHONE_STATE, Manifest.permission.CALL_PHONE };

    @Override
    public void onCreate(Bundle savedInstanceState) {
        try {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.act_otpcreate);
            checkPermission();

            drawView();
            getIntentData();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
        }
    }

    @Override
    public void onPause() {
        super.onPause();

        if (null != m_handlerProc) {
            m_handlerProc.removeMessages(MESSAGE_REFRESH_REMAINING_SECOND);
        }
    }

    @Override
    public void onResume() {
        super.onResume();

        if (null != m_handlerProc) {
            m_handlerProc.sendEmptyMessageDelayed(MESSAGE_REFRESH_REMAINING_SECOND,
SENDMESSAGE_INTERVAL);
        }
    }

    @SuppressLint("HardwareIds")
    @Override
    public void drawView() {
        try {
            vibe = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);

            findViewById(R.id.body_frame).setOnClickListener(new OnClickListener() {
                public void onClick(View v) {
                    InputMethodManager          imm          =          (InputMethodManager)
```

```
getSystemService(Context.INPUT_METHOD_SERVICE);
                    imm.hideSoftInputFromWindow(v.getWindowToken(), 0);
                }
            });

            tv_system_nm = (TextView) findViewById(R.id.tv_system_nm);
            tv_system_nm.setOnClickListener(this);

            tv_auth_key = (TextView) findViewById(R.id.tv_auth_key);
            tv_auth_key.setFocusable(true);
            tv_auth_key.setClickable(false);

            progressBar = (BootstrapProgressBar) findViewById(R.id.progressBar);

            tv_remainTime = (TextView) findViewById(R.id.tv_remainTime);

            btn_setting = (Button) findViewById(R.id.btn_setting);
            btn_setting.setOnClickListener(this);

            btn_share = (Button) findViewById(R.id.btn_share);
            btn_share.setOnClickListener(this);

            ((Button) findViewById(R.id.btn_go_back)).setOnClickListener(this);

            btn_close = (Button) findViewById(R.id.btn_close);
            btn_close.setOnClickListener(this);

            btn_update = (Button) findViewById(R.id.btn_update);
            btn_update.setOnClickListener(this);

            btn_reset = (Button) findViewById(R.id.btn_reset);
            btn_reset.setOnClickListener(this);

            TelephonyManager systemService = (TelephonyManager)
getSystemService(Context.TELEPHONY_SERVICE);
            assert systemService != null;
            PhoneNumber = systemService.getLine1Number();
            PhoneNumber = PhoneNumber.substring(PhoneNumber.length() - 10,
PhoneNumber.length());
            PhoneNumber = "0" + PhoneNumber;

        } catch (SecurityException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
        }
    }

    public void getIntentData() {
        try {
            Intent intent = getIntent();
            getDefaultIntent(intent);

            if (intent.getStringExtra("reg_dt") != null) {
```

```
                intent_reg_dt = intent.getStringExtra("reg_dt").trim();
            }
            if (intent.getStringExtra("system_nm") != null) {
                intent_system_nm = intent.getStringExtra("system_nm");
            }
            if (intent.getStringExtra("login_id") != null) {
                intent_login_id = intent.getStringExtra("login_id").trim();
            }
            if (intent.getStringExtra("cycle_time") != null) {
                intent_cycle_time = intent.getStringExtra("cycle_time").trim();
            }
            if ("".equals(intent_system_nm.trim())) {
                tv_system_nm.setText("[ " + intent_login_id + " ]");
            } else if (!"".equals(intent_system_nm) && (!"".equals(intent_login_id))) {
                tv_system_nm.setText("[ " + intent_system_nm + " / " + intent_login_id +
" ]");
            }
            if     (!"".equals(intent_login_id)     &&     !"".equals(PhoneNumber)     &&
(!"".equals(intent_cycle_time))) {
                onAuthKey();
            } else {
                finish();
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
        }
    }

    @Override
    public void onClick(View v) {
        try {
            switch (v.getId()) {
                case R.id.btn_setting: // Setting
                     Intent intent = new Intent(this, SettingACT.class);
                     intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
                     startActivity(intent);
                     //finish();
                     break;

                case R.id.btn_share:
                     intent = new Intent(Intent.ACTION_SEND);
                     intent.addCategory(Intent.CATEGORY_DEFAULT);
                     intent.putExtra(Intent.EXTRA_TEXT , getString(R.string.app_share));
                     intent.putExtra(Intent.EXTRA_TITLE, getString(R.string.app_name ));
                     intent.setType("text/plain");
                     startActivity(Intent.createChooser(intent,
getString(R.string.share_text)));
                     //finish();
                     break;

                case R.id.btn_go_back: // go back
                     finish();
                     break;
```

```
                case R.id.btn_close: // Close
                     moveTaskToBack(true);
                     finish();
                     android.os.Process.killProcess(android.os.Process.myPid());
                     break;

                case R.id.btn_update: // Update
                     intent = new Intent(OTPCreateACT.this, OTPUpdateACT.class);
                     intent.putExtra("reg_dt"    , intent_reg_dt    );
                     intent.putExtra("system_nm" , intent_system_nm );
                     intent.putExtra("login_id"  , intent_login_id  );
                     intent.putExtra("cycle_time", intent_cycle_time);
                     startActivity(intent);
                     finish();
                     break;

                case R.id.btn_reset: // Reset
                     if   (!"".equals(intent_login_id)   &&   !"".equals(PhoneNumber)   &&
(!"".equals(intent_cycle_time))) {
                         onAuthKey();
                     } else {
                         finish();
                     }
                     break;
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
        }
    }

    public void onAuthKey() {
        try {
            tv_auth_key.setText("");
            createdMillis = estimateCreatedMillis(intent_cycle_time);
            tv_auth_key.setText(barokey.generateKEYL(intent_login_id,          PhoneNumber,
intent_cycle_time));
            m_handlerProc.sendEmptyMessageDelayed(MESSAGE_REFRESH_REMAINING_SECOND,
SENDMESSAGE_INTERVAL);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
        }
    }

    private final Handler m_handlerProc = new Handler() {
        @Override
        public void handleMessage(Message message) {
            switch (message.what) {
                case MESSAGE_REFRESH_REMAINING_SECOND:
                    try {
                        long    cycleMillis       = (Long.parseLong(intent_cycle_time) *
1000L);
                        long                                    remainingMillis          =
estimateRemainingMillis(intent_cycle_time, createdMillis);
```

```
                        long   remainingSecond = remainingMillis != 0 ? (remainingMillis /
1000L) : 0;

                        if (0 < remainingMillis) {

m_handlerProc.sendEmptyMessageDelayed(MESSAGE_REFRESH_REMAINING_SECOND,
SENDMESSAGE_INTERVAL);

                            btn_reset.setEnabled(false);
                        } else {

m_handlerProc.removeMessages(MESSAGE_REFRESH_REMAINING_SECOND);

                            btn_reset.setEnabled(true);
                        }
                        tv_remainTime.setText(remainingSecond      +      "      "      +
getString(R.string.remain_time_suffix));

                        if (0 != cycleMillis) {
                            progressBar.setProgress((int)   (((float)   remainingMillis   /
(float) cycleMillis) * 100.0F));
                        }
                    } catch (Exception e) {
                        e.printStackTrace();
                    } finally {
                    }
                    break;
                }
            }
        };

        public long estimateCreatedMillis(String cycleSecondString) {
            long  remainingMillis  =  (barokey.getRemainingTime(cycleSecondString)  *  1000L)  -
200;
            long cycleMillis     = (Long.parseLong(cycleSecondString) * 1000L);
            long currentMillis   = (new Date()).getTime();
            long elapsedMillis   = cycleMillis   - remainingMillis;
            long createdMillis   = currentMillis - elapsedMillis;

            return createdMillis;
        }

        public long estimateRemainingMillis(String cycleSecondString, long createdTime) {
            long cycleMillis     = (Long.parseLong(cycleSecondString) * 1000L);
            long currentMillis   = (new Date()).getTime();
            long elapsedMillis   = currentMillis - createdTime;

            long remainingMillis = barokey.getRemainingTime(cycleSecondString) * 1000L;
            remainingMillis      = cycleMillis     >  elapsedMillis ? remainingMillis : 0;
            remainingMillis      = remainingMillis >= cycleMillis ? 0 : remainingMillis;

            return remainingMillis;
        }

        public void checkPermission() {
```

```
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M)
            return;

        for(String permission : permission_list) {
            int permssionCheck = checkCallingOrSelfPermission(permission);

            if (permssionCheck == PackageManager.PERMISSION_DENIED) {
                ActivityCompat.requestPermissions(this, permission_list, 0);
            }
        }
    }

    public void onRequestPermissionsResult(int requestCode, String[] permissions, int[]
grantResults) {
        if (requestCode == 0) {
            for(int ii = 0; ii < grantResults.length; ii++) {
                if (grantResults[ii] != PackageManager.PERMISSION_GRANTED) {
                    Util.MsgToast(OTPCreateACT.this, getString(R.string.msg_security_set),
0);
                    finish();
                }
            }
        }
    }
}
```

2.3 For iPhone

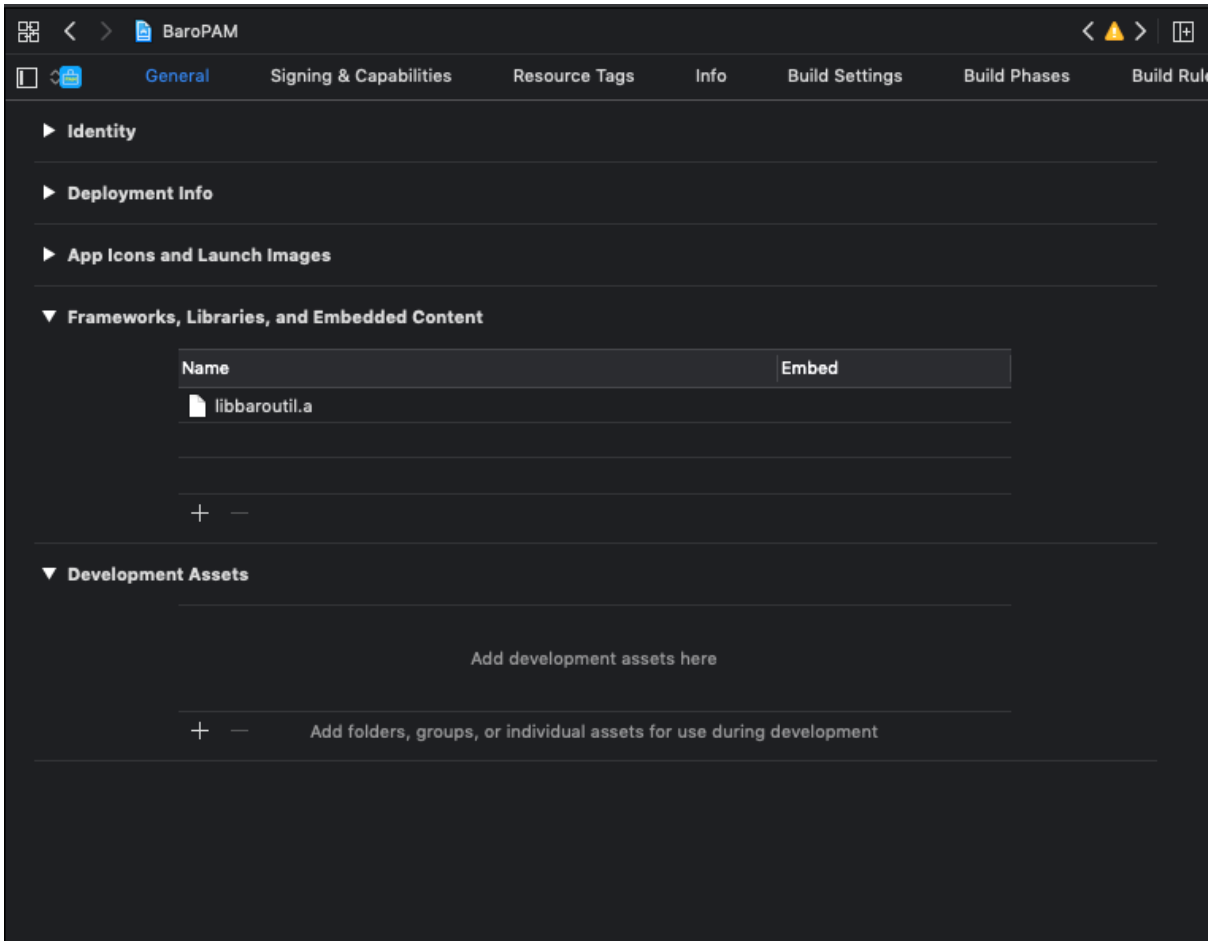
1) Authentication key generator part

The API for creating a one-time authentication key to be entered in the password field when logging in to the application is provided as "**libbaroutil.a**", and this file is a library file for NSObject Interface that includes libraries related to barokey, barocrypt, and base64.

Library files are provided in two types. XCode's iPhone simulator and iPhone use are changed to **libbaroutil.a** as needed.

- libbaroutil.a.iphone : for iPhone
- libbaroutil.a.simul : for iPhone simulator

This file is registered and used when setting the XCode project as follows.



BaroPAM related API is as follows. The function is composed of C function interface, so the data type of the input value is expressed in C function style. The source of the usage example is the code written with iOS swift 5.0 or higher.

generateKEYL function

This is a function that creates a one-time authentication key used when logging in/authentication to an application.

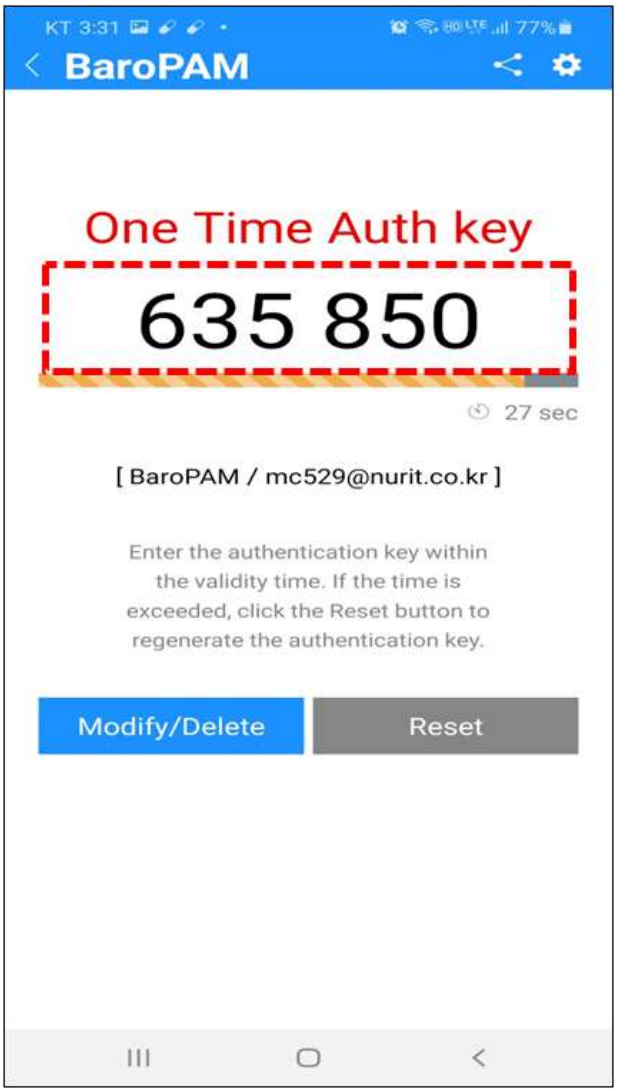
Input variable	const char *login_id	Set the ID entered in the Login-ID field of the login screen.
	const char *phone_no	This is the user's smartphone number. **Unlike the Android app, the user's smartphone number to be used in the server's authentication module is directly registered and managed in the app, without obtaining the user's smartphone number from iOS, and the registered smartphone number is selected and used.**
	const char *cycle_time	It must match the generation cycle (3~60 seconds) of the **one-time authentication key** designated for each individual. If the generation period of the one-time authentication key designated for each individual is different, the one-time authentication key may be generated differently.
	const char *key_method	Set "**app512**" as the authentication method of the one-time authentication key (app1, app256, app384, **app512**: app).

Return value	One-time auth key	Returns the generated one-time authentication key.

Example of use in swift 5.0 or higher)

```
private func makeOtpInfo() {
    let loginid = _otp?.LOGIN_ID ?? "mc529@hanmail.net"
    let tel = _otp?.PHONE_NO ?? "01027714076"
    let time = (_otp?.CYCLE_TIME ?? "30")!
    let otpnum = generateKEYL(loginid, tel, time, "app512")
    _otpInfo.text = "[ W(_otp?.SYSTEM_NM ?? "")/W(_otp?.LOGIN_ID ?? "") ]"
    let otpnumStr = String(cString: otpnum!)
    let start = otpnumStr.index(otpnumStr.startIndex, offsetBy: 0)
    let end = otpnumStr.index(otpnumStr.startIndex, offsetBy: 3)
    let start2 = otpnumStr.index(otpnumStr.startIndex, offsetBy: 3)
    let end2 = otpnumStr.index(otpnumStr.startIndex, offsetBy: 6)

    _tfOTP.text = otpnumStr[start..<end] + " " + otpnumStr[start2..<end2]

    var step = 0
    self._progress.progress = 0
    self._btnReset.isEnabled = false
    self._btnReset.backgroundColor = uicolorFromHex(rgbValue: 0xA0AAB4)
    let remain = getRemainingTime(_otp?.CYCLE_TIME ?? "30")
    let cycle_time = Int(self._otp!.CYCLE_TIME)

    _timer = Timer.scheduledTimer(withTimeInterval: 1.0, repeats: true, block: { _timer in
        let change: Float = Float(Double(remain - step - 1) / Double(cycle_time!))
        print("---- W(remain),W(change),W(step)")

        self._progress.progress = change
        step += 1
        self._remainTime.text = String(remain - step) + " " + "TIME".localized
        if step == remain {
            self._timer?.invalidate()
            self._btnReset.isEnabled = true
            self._btnReset.backgroundColor = uicolorFromHex(rgbValue: 0x1B90FF)
        }
    })
}
```

Screen example)



Screen Layout Example)

It means Storyboard. For the meaning of each parameter, refer to developer.apple.com.

```
<!--Create View Controller-->
  <scene sceneID="xJv-bd-Ejb">
    <objects>
      <viewController storyboardIdentifier="CreateOTP" id="BPh-TI-Gd5"
customClass="OTPCreateViewController" customModule="BaroPAM" customModuleProvider="target"
sceneMemberID="viewController">
        <layoutGuides>
          <viewControllerLayoutGuide type="top" id="TF9-Et-51n"/>
          <viewControllerLayoutGuide type="bottom" id="rXs-zr-mnc"/>
        </layoutGuides>
        <view key="view" contentMode="scaleToFill" id="Dbl-ks-whW">
          <rect key="frame" x="0.0" y="0.0" width="375" height="812"/>
          <autoresizingMask key="autoresizingMask" widthSizable="YES"
heightSizable="YES"/>
          <subviews>
            <textView clipsSubviews="YES" multipleTouchEnabled="YES"
contentMode="scaleToFill" fixedFrame="YES" text="일회용 인증키" textAlignment="center"
```

```
translatesAutoresizingMaskIntoConstraints="NO" id="00T-0a-9fL">
            <rect key="frame" x="0.0" y="125" width="375" height="40"/>
            <autoresizingMask          key="autoresizingMask"          widthSizable="YES"
flexibleMaxY="YES"/>
            <color key="textColor" white="0.0" alpha="1" colorSpace="calibratedWhite"/>
            <fontDescription      key="fontDescription"      name="SpoqaHanSans-Regular"
family="SpoqaHanSans" pointSize="17"/>
            <textInputTraits key="textInputTraits" autocapitalizationType="sentences"/>
          </textView>
          <textField    opaque="NO"    clipsSubviews="YES"    contentMode="scaleToFill"
fixedFrame="YES"    contentHorizontalAlignment="left"    contentVerticalAlignment="center"
text="12345678"                textAlignment="center"                minimumFontSize="17"
translatesAutoresizingMaskIntoConstraints="NO" id="y9V-i0-Xec">
            <rect key="frame" x="19" y="204" width="336" height="52"/>
            <autoresizingMask          key="autoresizingMask"          widthSizable="YES"
flexibleMaxY="YES"/>
            <color          key="backgroundColor"          white="1"          alpha="1"
colorSpace="calibratedWhite"/>
            <fontDescription      key="fontDescription"      name="SpoqaHanSans-Regular"
family="SpoqaHanSans" pointSize="50"/>
            <textInputTraits key="textInputTraits"/>
          </textField>
          <button      opaque="NO"      contentMode="scaleToFill"      fixedFrame="YES"
contentHorizontalAlignment="center"                        contentVerticalAlignment="center"
buttonType="roundedRect"                                    lineBreakMode="middleTruncation"
translatesAutoresizingMaskIntoConstraints="NO" id="wn5-JQ-qp2">
            <rect key="frame" x="23" y="683" width="160" height="43"/>
            <autoresizingMask          key="autoresizingMask"          widthSizable="YES"
flexibleMaxX="YES" flexibleMinY="YES"/>
            <fontDescription      key="fontDescription"      name="SpoqaHanSans-Regular"
family="SpoqaHanSans" pointSize="17"/>
            <state key="normal" title="Update/Delete">
              <color key="titleColor" white="1" alpha="1" colorSpace="calibratedWhite"/>
            </state>
            <connections>
              <action             selector="onEdit:"             destination="BPh-Tl-Gd5"
eventType="touchUpInside" id="Lq0-gt-fdh"/>
              <action              selector="onOk:"              destination="BYZ-38-t0r"
eventType="touchUpInside" id="ya1-b8-A5Q"/>
            </connections>
          </button>
          <button      opaque="NO"      contentMode="scaleToFill"      fixedFrame="YES"
contentHorizontalAlignment="center"                        contentVerticalAlignment="center"
buttonType="roundedRect"                                    lineBreakMode="middleTruncation"
translatesAutoresizingMaskIntoConstraints="NO" id="phw-d7-Zsz">
            <rect key="frame" x="199" y="683" width="160" height="43"/>
            <autoresizingMask          key="autoresizingMask"          flexibleMinX="YES"
widthSizable="YES" flexibleMinY="YES"/>
            <fontDescription      key="fontDescription"      name="SpoqaHanSans-Regular"
family="SpoqaHanSans" pointSize="17"/>
            <state key="normal" title="Reset">
              <color key="titleColor" white="1" alpha="1" colorSpace="calibratedWhite"/>
            </state>
            <connections>
              <action           selector="onReset:"           destination="BPh-Tl-Gd5"
```

```
eventType="touchUpInside" id="2is-dP-y2P"/>
            </connections>
          </button>
          <view                contentMode="scaleToFill"                fixedFrame="YES"
translatesAutoresizingMaskIntoConstraints="NO" id="KTy-6U-Omm">
            <rect key="frame" x="0.0" y="0.0" width="375" height="70"/>
            <autoresizingMask          key="autoresizingMask"          widthSizable="YES"
flexibleMaxY="YES"/>
            <subviews>
              <button     opaque="NO"     contentMode="scaleToFill"     fixedFrame="YES"
contentHorizontalAlignment="center"                        contentVerticalAlignment="center"
lineBreakMode="middleTruncation"  translatesAutoresizingMaskIntoConstraints="NO"  id="5ZR-
gQ-4P5">
                <rect key="frame" x="283" y="34" width="31" height="31"/>
                <autoresizingMask       key="autoresizingMask"       flexibleMinX="YES"
flexibleMaxY="YES"/>
                <inset key="imageEdgeInsets" minX="3" minY="3" maxX="3" maxY="3"/>
                <state key="normal" image="btn_share.png"/>
                <connections>
                  <action          selector="onShare:"          destination="BPh-TI-Gd5"
eventType="touchUpInside" id="rVd-IW-j3A"/>
                </connections>
              </button>
              <button     opaque="NO"     contentMode="scaleToFill"     fixedFrame="YES"
contentHorizontalAlignment="center"                        contentVerticalAlignment="center"
lineBreakMode="middleTruncation"  translatesAutoresizingMaskIntoConstraints="NO"  id="uD6-
U2-2w3">
                <rect key="frame" x="322" y="34" width="33" height="32"/>
                <autoresizingMask       key="autoresizingMask"       flexibleMinX="YES"
flexibleMaxY="YES"/>
                <inset key="imageEdgeInsets" minX="3" minY="3" maxX="3" maxY="3"/>
                <state key="normal" image="btn_setting.png"/>
                <connections>
                  <action         selector="onSetting:"         destination="BPh-TI-Gd5"
eventType="touchUpInside" id="Qhc-bj-CHe"/>
                </connections>
              </button>
              <imageView    userInteractionEnabled="NO"    contentMode="scaleAspectFit"
horizontalHuggingPriority="251"       verticalHuggingPriority="251"       fixedFrame="YES"
image="btn_prev.png" translatesAutoresizingMaskIntoConstraints="NO" id="cZQ-Jb-luv">
                <rect key="frame" x="19" y="35" width="31" height="31"/>
                <autoresizingMask       key="autoresizingMask"       flexibleMaxX="YES"
heightSizable="YES"/>
              </imageView>
              <imageView    userInteractionEnabled="NO"    contentMode="scaleAspectFit"
horizontalHuggingPriority="251"       verticalHuggingPriority="251"       fixedFrame="YES"
image="logo_barootp.png" translatesAutoresizingMaskIntoConstraints="NO" id="vWu-o6-6az">
                <rect key="frame" x="115" y="38" width="145" height="25"/>
                <autoresizingMask       key="autoresizingMask"       flexibleMaxX="YES"
flexibleMaxY="YES"/>
              </imageView>
            </subviews>
            <color            key="backgroundColor"            red="0.10588235294117647"
green="0.56470588235294117" blue="1" alpha="1" colorSpace="calibratedRGB"/>
          </view>
```

```
            <textView clipsSubviews="YES" multipleTouchEnabled="YES"
contentMode="scaleToFill" fixedFrame="YES" editable="NO" text="유효시간 내에 인증키를 입력
하세요. 시간을 초과한 경우 Reset 버튼을 클릭하여 인증키를 재생성 하세요."
textAlignment="natural" selectable="NO" translatesAutoresizingMaskIntoConstraints="NO"
id="s4z-fe-3rj">
              <rect key="frame" x="49" y="585" width="276" height="112"/>
              <autoresizingMask key="autoresizingMask" widthSizable="YES"
flexibleMinY="YES"/>
              <color key="textColor" red="0.33333333333333331" green="0.33333333333333331"
blue="0.33333333333333331" alpha="1" colorSpace="calibratedRGB"/>
              <fontDescription key="fontDescription" name="SpoqaHanSans-Regular"
family="SpoqaHanSans" pointSize="17"/>
              <textInputTraits key="textInputTraits" autocapitalizationType="sentences"/>
            </textView>
            <progressView opaque="NO" contentMode="scaleToFill"
verticalHuggingPriority="750" fixedFrame="YES" progress="0.5"
translatesAutoresizingMaskIntoConstraints="NO" id="eFk-qb-ugh">
              <rect key="frame" x="52" y="274" width="270" height="2"/>
              <autoresizingMask key="autoresizingMask" widthSizable="YES"/>
            </progressView>
            <imageView userInteractionEnabled="NO" contentMode="scaleToFill"
horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES"
image="ico_countdown.png" translatesAutoresizingMaskIntoConstraints="NO" id="UC7-dN-2I6">
              <rect key="frame" x="250" y="284" width="15" height="15"/>
              <autoresizingMask key="autoresizingMask" flexibleMaxX="YES"
flexibleMaxY="YES"/>
            </imageView>
            <label opaque="NO" userInteractionEnabled="NO" contentMode="left"
horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" text="0"
textAlignment="natural" lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines"
adjustsFontSizeToFit="NO" translatesAutoresizingMaskIntoConstraints="NO" id="c11-3a-nD8">
              <rect key="frame" x="270" y="281" width="52" height="21"/>
              <autoresizingMask key="autoresizingMask" flexibleMaxX="YES"
flexibleMaxY="YES"/>
              <fontDescription key="fontDescription" name="SpoqaHanSans-Regular"
family="SpoqaHanSans" pointSize="17"/>
              <color key="textColor" red="0.33333333333333331" green="0.33333333333333331"
blue="0.33333333333333331" alpha="1" colorSpace="calibratedRGB"/>
              <nil key="highlightedColor"/>
            </label>
            <label opaque="NO" userInteractionEnabled="NO" contentMode="left"
horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES"
text="[emplus/david.kscho@empluses.com]" textAlignment="center"
lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines"
adjustsFontSizeToFit="NO" translatesAutoresizingMaskIntoConstraints="NO" id="FZ0-er-yGs">
              <rect key="frame" x="23" y="318" width="332" height="30"/>
              <autoresizingMask key="autoresizingMask" widthSizable="YES"
flexibleMaxY="YES"/>
              <fontDescription key="fontDescription" name="SpoqaHanSans-Regular"
family="SpoqaHanSans" pointSize="17"/>
              <nil key="highlightedColor"/>
            </label>
          </subviews>
          <color key="backgroundColor" white="1" alpha="1" colorSpace="calibratedWhite"/>
        </view>
```

```
        <connections>
          <outlet property="_backView" destination="cZQ-Jb-luv" id="hti-Le-Rra"/>
          <outlet property="_btnReset" destination="phw-d7-Zsz" id="hVD-Q9-8Xq"/>
          <outlet property="_btnUpdate" destination="wn5-JQ-qp2" id="o6G-9e-g0S"/>
          <outlet property="_otpInfo" destination="FZ0-er-yGs" id="d1r-2i-KX2"/>
          <outlet property="_progress" destination="eFk-qb-ugh" id="csW-nT-cyw"/>
          <outlet property="_remainTime" destination="c11-3a-nD8" id="b6H-g5-IXA"/>
          <outlet property="_tfOTP" destination="y9V-i0-Xec" id="loX-6A-goi"/>
        </connections>
      </viewController>
      <placeholder         placeholderIdentifier="IBFirstResponder"         id="GRs-8z-hxZ"
userLabel="First Responder" sceneMemberID="firstResponder"/>
    </objects>
    <point key="canvasLocation" x="2948" y="440"/>
  </scene>
```

Program example)

```
import UIKit

class OTPCreateViewController : UIViewController {
    @IBOutlet weak var _progress: UIProgressView!
    @IBOutlet weak var _remainTime: UILabel!
    @IBOutlet weak var _backView: UIImageView!
    @IBOutlet weak var _otpInfo: UILabel!
    @IBOutlet weak var _tfOTP: UITextField!
    @IBOutlet weak var _btnUpdate: UIButton!
    @IBOutlet weak var _btnReset: UIButton!

    @IBAction func onClose(_ sender: Any) {
        exit(0)
    }

    var _timer: Timer?
    var _otp: OTPEntity? = nil

    override func viewDidLoad() {
        super.viewDidLoad()
        //chagneBackground()
        initControls()
        makeTappedView()
        makeOtpInfo()
    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)
        if (_otp?.IS_DELETE == 1) {
            _otp?.IS_DELETE = 0
            dismiss(animated: false, completion: nil)
        }
    }
    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
    }
```

```
    private func initControls()  {
        _btnUpdate.backgroundColor = uicolorFromHex(rgbValue: 0x1B90FF)
        _btnReset.backgroundColor = uicolorFromHex(rgbValue: 0x1B90FF)

        super.modalPresentationStyle = .fullScreen
    }

    private func chagneBackground() {
        // MAIN View Background Change
        let background = UIImageView(frame: UIScreen.main.bounds)
        background.image = UIImage(named: "bg_sub.png")
        self.view.insertSubview(background, at: 0)
    }

    private func makeTappedView() {
        let tap = UITapGestureRecognizer(target: self, action:
#selector(OTPCreateViewController.backTapped))
        _backView.isUserInteractionEnabled = true
        _backView.addGestureRecognizer(tap)
    }

    private func makeOtpInfo() {
        let loginid = _otp?.LOGIN_ID ?? "mc529@hanmail.net"
        let tel = _otp?.PHONE_NO ?? "01027714076"
        let time = (_otp?.CYCLE_TIME ?? "30")!
        let otpnum = generateKEYL(loginid, tel, time, "app512")
        _otpInfo.text = "[ W(_otp?.SYSTEM_NM ?? "")/W(_otp?.LOGIN_ID ?? "") ]"
        let otpnumStr = String(cString: otpnum!)
        let start = otpnumStr.index(otpnumStr.startIndex, offsetBy: 0)
        let end = otpnumStr.index(otpnumStr.startIndex, offsetBy: 3)
        let start2 = otpnumStr.index(otpnumStr.startIndex, offsetBy: 3)
        let end2 = otpnumStr.index(otpnumStr.startIndex, offsetBy: 6)

        _tfOTP.text = otpnumStr[start..<end] + " " + otpnumStr[start2..<end2]

        var step = 0
        self._progress.progress = 0
        self._btnReset.isEnabled = false
        self._btnReset.backgroundColor = uicolorFromHex(rgbValue: 0xA0AAB4)
        let remain = getRemainingTime(_otp?.CYCLE_TIME ?? "30")
        let cycle_time = Int(self._otp!.CYCLE_TIME)

        _timer = Timer.scheduledTimer(withTimeInterval: 1.0, repeats: true, block:
{ _timer in
            let change: Float = Float(Double(remain - step - 1) / Double(cycle_time!))
            print("---- W(remain),W(change),W(step)")

            self._progress.progress = change
            step += 1
            self._remainTime.text = String(remain - step) + " " + "TIME".localized
            if step == remain {
                self._timer?.invalidate()
                self._btnReset.isEnabled = true
                self._btnReset.backgroundColor = uicolorFromHex(rgbValue: 0x1B90FF)
```

```
            }
        })
    }

    @objc func backTapped(tabGestureRecg: UITapGestureRecognizer) {
        dismiss(animated: false, completion: nil) //
    }

    @IBAction func onEdit(_ sender: Any) {
        switchScreen("SystemOTP",          {          _          =          ($0          as!
OTPInfoSaveViewController).changeMode(.EDIT).setOtp(_otp!).setParent(self) })
    }

    @IBAction func onReset(_ sender: Any) {
        makeOtpInfo()
    }

    func setOtp(_ otp: OTPEntity) {
        _otp = otp
        print("-------->      W(otp.REG_DT),      W(otp.LOGIN_ID),      W(otp.SYSTEM_NM),
W(otp.CYCLE_TIME)")
    }

    @IBAction func onSetting(_ sender: Any) {
        switchScreen("Settings")
    }

    @IBAction func onShare(_ sender: Any) {
    }
}
```

3. BaroPAM integration example

When logging into an application, you must add the phone number (USER_PHONE), one-time authentication key generation cycle (CYCLE_TIME), and login final time (LOGIN_TIME) columns for the user to generate a one-time authentication key to the user information table.

```
USER_PHONE          VARCHAR2(50)  NOT NULL ,
CYCLE_TIME          VARCHAR2(2)   DEFAULT '30' ,
LOGIN_TIME          VARCHAR2(10) DEFAULT '0' ,
```

The reason for adding the login final time is to prevent reuse and **man-in-the-middle attacks** by limiting the ability to log in only one user with the same login-ID within the one-time authentication key generation cycle.

First, the "**system name, login-ID, and authentication cycle**" related to the application's login screen must be entered identically in the "**Login Information Registration**" screen of the

"BaroPAM" app.


```
<%@ page contentType="text/html; charset=UTF-8" language="java" pageEncoding="UTF-8" %>
<%@include file="./inc_common.jsp" %>
<%@include file="./inc_session.jsp" %>
<% request.setCharacterEncoding("utf-8"); %>
<% response.setContentType("text/html; charset=utf-8"); %>
<%
/*-------------------------------------------------------------------------*/
/* Variable declaration and initialization.                                */
/*-------------------------------------------------------------------------*/
int     ii = 0, jj = 0, kk = 0, ll = 0;                  // Index

CommonLib clib = new CommonLib();                        // CommonLib
LogDAO    ldao = new LogDAO();                           // LogDAO
UserDAO   udao = new UserDAO();                          // UserDAO
BaroDTO   bdto = new BaroDTO();                          // BaroDTO

String  HTML       = "";                                 // HTML
String  msg        = "";                                 // msg
String  value      = "";                                 // value
String  result     = "";                                 // Result
String  login_time = "";                                 // Last login time
boolean bauth_key  = false;                              // Verification result
/*-------------------------------------------------------------------------*/
/* Get data from Request.                                                  */
/*-------------------------------------------------------------------------*/
String  user_id  = clib.strDefault(request.getParameter("user_id" ), "" );
String  user_pw  = clib.strDefault(request.getParameter("user_pw" ), "" );
String  auth_key = clib.strDefault(request.getParameter("auth_key"), "" );
String       ip_addr       =    clib.strDefault(request.getParameter("ip_addr"          ),
request.getRemoteAddr());

logger.info("(result_login.jsp)Starting.....................");
String  param = request.getServerName() + request.getRequestURI()
              + "?remote_addr="        + request.getRemoteAddr()
              + "&user_id="            + user_id
```

```
                + "&user_pw="            + user_pw
                + "&auth_key="           + auth_key
               + "&ip_addr="             + ip_addr
                ;
logger.info(param);
/*---------------------------------------------------------------------*/
/* Begin.                                                              */
/*---------------------------------------------------------------------*/
try {
    /*-----------------------------------------------------------------*/
    /* User information inquiry.                                       */
    /*-----------------------------------------------------------------*/
    bdto = udao.read(user_id);
    /*-----------------------------------------------------------------*/
    /* If user information exists.                                     */
    /*-----------------------------------------------------------------*/
    if (bdto != null) {
        /*-------------------------------------------------------------*/
        /* If the password is the same.                                */
        /*-------------------------------------------------------------*/
        if (user_pw.equals(bdto.getUser_pw())) {
            /*---------------------------------------------------------*/
            /* Login last time Edit.                                   */
            /*---------------------------------------------------------*/
            login_time = Long.toString(barokey.get_logintime(bdto.getCycle_time()));
            /*---------------------------------------------------------*/
            /* When the last login time is greater than the creation cycle.  */
            /*---------------------------------------------------------*/
            if (Long.parseLong(login_time) > Long.parseLong(bdto.getLogin_time())) {
                /*-----------------------------------------------------*/
                /* Authentication key verificat.                       */
                /*-----------------------------------------------------*/
                bauth_key = barokey.verifyKEYP(bdto.getSecure_key(), bdto.getCycle_time(),
auth_key);
           /*-------------------------------------------------------------*/
           /* When the login final time is less than or equal to the creation cycle. */
           /*-------------------------------------------------------------*/
            } else {
                bauth_key = false;
}
            /*---------------------------------------------------------*/
            /* Authentication key verification (success).              */
            /*---------------------------------------------------------*/
            if (bauth_key == true) {
                /*-----------------------------------------------------*/
                /* Cookie creation.                                    */
                /*-----------------------------------------------------*/
                session = request.getSession(true);
                request.getSession().setAttribute("SES_USER_NO"   , bdto.getUser_no   ());
                request.getSession().setAttribute("SES_USER_NAME" , bdto.getUser_name ());
                request.getSession().setAttribute("SES_USER_DEPT" , bdto.getUser_dept ());
                request.getSession().setAttribute("SES_USER_TITLE", bdto.getUser_title());
                request.getSession().setAttribute("SES_USER_EMAIL", bdto.getUser_email());
                request.getSession().setAttribute("SES_USER_PHONE", bdto.getUser_phone());
                request.getSession().setAttribute("SES_ADMIN_YN"  , bdto.getAdmin_yn  ());
```

```
                /*-------------------------------------------------------------*/
                /* Update login last time.                                     */
                /*-------------------------------------------------------------*/
                udao.updateLoginTime(user_id, login_time);
                HTML = bdto.getUser_no();
            /*-----------------------------------------------------------------*/
            /* Authentication key verification (failure).                      */
            /*-----------------------------------------------------------------*/
            } else {
                HTML = "KEY_ERR";
            }
        /*---------------------------------------------------------------------*/
        /* If the passwords are not the same.                                  */
        /*---------------------------------------------------------------------*/
        } else {
            HTML = "PW_ERR";
        }
        /*---------------------------------------------------------------------*/
        /* Insert log information.                                             */
        /*---------------------------------------------------------------------*/
        bdto.setIp_addr(ip_addr);
        ldao.create(bdto);
    /*-------------------------------------------------------------------------*/
    /* User information does not exist.                                        */
    /*-------------------------------------------------------------------------*/
    } else {
        HTML = "ID_ERR";
    }
    json.put("HTML", HTML);
    out.println(json);
/*-----------------------------------------------------------------------------*/
/* Handling excepti.                                                           */
/*-----------------------------------------------------------------------------*/
} catch(Exception e) {
    logger.info("Exception = [" + e + "]");
    e.printStackTrace();
/*-----------------------------------------------------------------------------*/
/* Finally.                                                                    */
/*-----------------------------------------------------------------------------*/
} finally {
    logger.info("(result_login.jsp)Ending.....................");
}
%>
```