BaroPAM 가이드(Java)

목차

목차 0

1. BaroPAM 연동 API 1

1.1 연동 API 사용 전 준비사항 1

1.2 BaroPAM 연동 API 1

2. BaroPAM 적용 31

2.1 BaroPAM 적용 프로세스 31

2.2 BaroPAM 적용 화면 31

2.3 본인확인 적용 프로세스 32

2.4 본인확인 적용 화면 33

2.5 애플리케이션 로그인 34

3. About BaroPAM 39

1. BaroPAM 연동 API

1.1 연동 API 사용 전 준비사항

BaroPAM에서 사용하는 인증 코드인 **일회용 인증키**는 Java를 기반으로 작성되었기 때문에 반드시 최신 **JDK 6.x 이상**이 설치 되어 있어야 한다. 만약, 설치되어 있지 않으면 최신 JDK를 설치해야 한다.

JDK 설치 확인)

```
[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
```

JDK 설치 디렉토리 확인)

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

Java 버젼 확인)

```
[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)
```

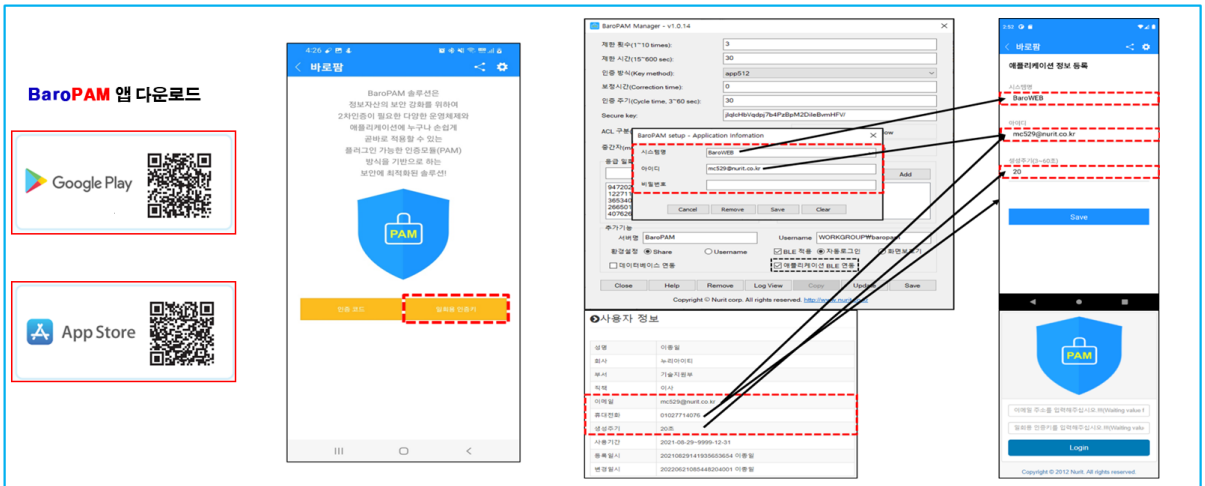
1.2 BaroPAM 연동 API

1. 로그인 화면

1) BaroPAM 로그인 화면 예시)

예) BaroPAM 앱 설치 및 정보 설정

정보자산에 로그인 시 Verification code에 입력할 **일회용 인증키**의 생성기인 BaroPAM 앱의 다운로드 (https://play.google.com/store/apps/details?id=com.baro.pam)는 구글의 "**Play 스토어**"나 Apple의 "**App 스토어**"에서 가능하며, 설치는 일반 앱의 설치와 동일하다.



BaroPAM 앱은 Android 6.0 (Marshmalliw) API 23, iOS 13.0 이상에서 사용 가능하며, 가로보기 모드를 지원하지 않는다.

BaroPAM 앱을 설치한 후 BaroPAM앱을 실행하여 메뉴 선택화면에서 "일회용 인증키" 버튼을 클릭하여 애플리케이션의 사용자 정보에 설정한 "인증주기, 아이디, 시스템명"을 BaroPAM 앱의 "애플리케이션 정보 등록" 화면에서 동일하게 입력해야 한다.

현상 : 안드로이폰 또는 아이폰의 날짜와 시간이 현재 시간과 차이가 발생하여 "일회용 인증키"가 맞지 않은 경우
원인 : 안드로이폰 또는 아이폰의 날짜와 시간을 네트워크에서 제공하는 시간을 사용하지 않아서 발생.
조치 : 안드로이폰인 경우는 폰의 "설정" -> "일반" -> "날짜 및 시간" -> "날짜 및 자동 설정"과 "시간대 자동 설정" -> "허용"
아이폰인 경우는 폰의 "설정" -> "날짜 및 시간" -> "자동으로 설정" -> "허용"

2) 인증키 검증 부분

애플리케이션에 로그인 시 비밀번호란에 입력한 **일회용 인증키**를 검증하는 API는 "**barokey.jar**"로 제공되며, NodeJS 웹서버의 node를 실행할 디렉토리에 "**barokey.jar**"를 위치시키거나 classpath에 "**barokey.jar**"가 존재하는 디렉토리를 포함해서 설정해 주면 된다.

```
[root] /home/tomcat/lib > ls -al
total 23100
drwxrwxrwx  2 root root     4096 May 22 13:29 .
drwxr-xr-x 13 root root     4096 Aug 14  2017 ..
-rw-r--r--  1 root root     2550 Apr 23  2022 .bash_history
-rwxrwxrwx  1 root root    15240 Sep 18  2014 annotations-api.jar
-rw-r--r--  1 root root    16864 May 22 15:10 barokey.jar
-rw-r--r--  1 root root    16730 Mar 14 14:08 barokey.jar.old
-rwxrwxrwx  1 root root    54565 Sep 18  2014 catalina-ant.jar
-rwxrwxrwx  1 root root   132132 Sep 18  2014 catalina-ha.jar
-rwxrwxrwx  1 root root   237521 Sep 18  2014 catalina-tribes.jar
-rwxrwxrwx  1 root root  1243752 Sep 18  2014 catalina.jar
...........
```

반드시 연동 API인 "**barokey.jar**"에서 사용하는 "**BAROPAM**" 환경변수를 "**.bash_profile**"에 설정해야 한다.

```
export BAROPAM=/home/tomcat/conf/.baro_nurit
```

일회용 인증키 검증 시 예외사항(Exception)이 발생할 경우 원인 파악을 손쉽게 할 수 있도록 표준 출력(Starndard ouput)으로 상세한 예외사항을 Stack trace를 콘솔에 제공한다.

```
try {

    ......
```

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

애플리케이션 로그인 시 사용자 폰번호(USER_PHONE), 일회용 인증키 생성주기(CYCLE_TIME), 로그인 최종 시간(LOGIN_TIME) 컬럼을 추가해야 한다.

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

로그인 최종시간을 추가하는 이유는 일회용 인증키 생성 주기 내에 동일한 로그인-ID에 사용자 한명만 로그인 가능하게 제한하여 재사용 및 중간자 공격(Man-in-the-middle attack)에 대비하기 위함이며, 로그인이 성공한 경우에 로그인 최종시간을 Update해야 한다.

애플리케이션에 로그인 시 입력한 비밀번호인 일회용 인증키를 검증하는 프로그램에 다음과 같은 코드를 삽입하면 된다.

① 로그인-ID를 사용하는 경우

```
...
import com.barokey.barokey;
...
로그인-ID 유효성 확인하여 성공인 경우만 인증키 검증 모듈을 호출
...
boolean bauth_key = barokey.verifyKEYL(String login_id, String phone_no, String cycle_time, auth_key);
boolean bauth_key = barokey.verifyKEYL(String login_id, String phone_no, String cycle_time, String key_method, String auth_key);

if (bauth_key == true) {
    // 검증 성공
} else {
    // 검증 실패
}
...
```

파라미터	설명	비고
login_id	로그인 화면의 로그인-ID 항목에 입력한 ID를 설정.	
phone_no	사용자별 스마트 폰 번호를 숫자만 설정.	
cycle_time	사용자별로 지정한 일회용 인증키의 생성 주기(3~60초)를 설정.	
key_method	일회용 인증키의 인증 방식(app1, app256, app384, app512: 앱)을 설정.	
auth_key	로그인 화면의 비밀번호에 입력한 일회용 인증키를 설정.	

만약, 사용자별로 스마트 폰 번호 및 개인별로 지정한 일회용 인증키의 생성 주기가 일회용 인증키의 생성 기와 다른 경우 일회용 인증키가 달라서 검증에 실패할 수 있다. 반드시 정보를 일치시켜야 한다.

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"); %>
<%
/*--------------------------------------------------------------------------*/
/* 변수선언 및 초기화.                                                      */
/*--------------------------------------------------------------------------*/
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 = "";                                // 로그인최종시간
boolean bauth_key  = false;                             // 검증결과
/*--------------------------------------------------------------------------*/
/* Request에서 데이터를 얻어옴(사용자정보).                                 */
/*--------------------------------------------------------------------------*/
String  auth_key   = clib.strDefault(request.getParameter("auth_key"  ), "" );
String  user_email = clib.strDefault(request.getParameter("user_email"), "" );
String  auto_login = clib.strDefault(request.getParameter("auto_login"), "N");
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
              + "&auto_login="         + auto_login
              ;
logger.info(param);
/*--------------------------------------------------------------------------*/
/* Begin.                                                                   */
/*--------------------------------------------------------------------------*/
try {
    /*----------------------------------------------------------------------*/
    /* 사용자정보 조회.                                                     */
    /*----------------------------------------------------------------------*/
    bdto = udao.reads(user_email);
    /*----------------------------------------------------------------------*/
    /* 사용자정보가 존재한 경우.                                            */
    /*----------------------------------------------------------------------*/
    if (bdto != null) {
        /*------------------------------------------------------------------*/
        /* 사용하지 않는 사용자인 경우.                                     */
        /*------------------------------------------------------------------*/
        if ("N".equals(bdto.getUse_yn())) {
            HTML = "USE_ERR";
        /*------------------------------------------------------------------*/
```

```
        /* 사용하는 사용자인 경우.                                          */
        /*------------------------------------------------------------------*/
        } else {
            /*------------------------------------------------------------------*/
            /* 로그인 최종 시간 Edit.                                          */
            /*------------------------------------------------------------------*/
            login_time = Long.toString(barokey.get_logintime(bdto.getCycle_time()));
            /*------------------------------------------------------------------*/
            /* 로그인 최종시간이 생성주기 보다 큰 경우.                        */
            /*------------------------------------------------------------------*/
            if (Long.parseLong(login_time) > Long.parseLong(bdto.getLogin_time())) {
                /*------------------------------------------------------------------*/
                /* 인증키 검증.                                                    */
                /*------------------------------------------------------------------*/
                bauth_key       =       barokey.verifyKEYL(user_email,       bdto.getUser_phone(),
bdto.getCycle_time(), auth_key);
                /*------------------------------------------------------------------*/
                /* 인증키 검증(성공).                                              */
                /*------------------------------------------------------------------*/
                if (bauth_key == true) {
                    /*------------------------------------------------------------------*/
                    /* 쿠키 생성.                                                      */
                    /*------------------------------------------------------------------*/
                    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  ());
                    /*------------------------------------------------------------------*/
                    /* 쿠키 생성.                                                      */
                    /*------------------------------------------------------------------*/
                    if ("Y".equals(auto_login)) {
                        Date now = new Date();
                        String timestamp = now.toString();
                        Cookie cookie = new Cookie ("SES_USER_NO", bdto.getUser_no());
                        //cookie.setMaxAge(7 * 24 * 60 * 60);
                        cookie.setMaxAge(60*60*24*365); // 쿠키 유지기간-1년
                        cookie.setPath("/");
                        response.addCookie(cookie);
                    }
                    /*------------------------------------------------------------------*/
                    /* 로그인 최종 시간을 Update.                                      */
                    /*------------------------------------------------------------------*/
                    udao.updateLoginTime(user_email, login_time);
                    HTML = bdto.getUser_no();
                /*------------------------------------------------------------------*/
                /* 인증키 검증(실패).                                              */
                /*------------------------------------------------------------------*/
                } else {
                    HTML = "KEY_ERR";
```

```
                }
            /*-------------------------------------------------------------*/
            /* 로그인 최종시간이 생성주기 보다 작거나 같은 경우.           */
            /*-------------------------------------------------------------*/
            } else {
                HTML = "KEY_ERR";
            }
        }
        /*-------------------------------------------------------------*/
        /* Log정보를 insert.                                           */
        /*-------------------------------------------------------------*/
        bdto.setIp_addr(ip_addr);
        ldao.create(bdto);
    /*-------------------------------------------------------------*/
    /* 사용자정보가 존재하지 않는 경우.                            */
    /*-------------------------------------------------------------*/
    } else {
        HTML = "ID_ERR";
    }
    json.put("HTML", HTML);
    out.println(json);
/*-------------------------------------------------------------*/
/* 예외사항 처리(Exception).                                   */
/*-------------------------------------------------------------*/
} catch(Exception e) {
    logger.info("Exception = [" + e + "]");
    e.printStackTrace();
/*-------------------------------------------------------------*/
/* Finally.                                                    */
/*-------------------------------------------------------------*/
} finally {
    logger.info("(result_user.jsp)Ending.....................");
}
%>
```

② Secure key를 사용하는 경우

```
...
import com.barokey.barokey;
...
로그인-ID 유효성 확인하여 성공인 경우만 인증키 검증 모듈을 호출
...
boolean bauth_key = barokey.verifyKEYP(String secure_key, String cycle_time, String tkey);
boolean bauth_key = barokey.verifyKEYP(String secure_key, String cycle_time, String key_method,
String auth_key);

if (bauth_key == true) {
    // 검증 성공
} else {
    // 검증 실패
}
...
```

파라미터	설명	비고
secure_key	벤더에서 제공한 Secure key를 설정.	
cycle_time	개인별로 지정한 **일회용 인증키**의 생성 주기(3~60초)를 설정. 만약, 개인별로 지정한 **일회용 인증키**의 생성 주기가 **일회용 인증키**의 생성기와 다른 경우 **일회용 인증키**가 달라서 검증에 실패할 수 있다. 반드시 정보를 일치시켜야 한다.	
key_method	**일회용 인증키**의 인증 방식(app1, app256, app384, app512: 앱)을 설정.	
auth_key	로그인 화면의 비밀번호에 입력한 **일회용 인증키**를 설정.	

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"); %>
<%
/*-------------------------------------------------------------------------*/
/* 변수선언 및 초기화.                                                     */
/*-------------------------------------------------------------------------*/
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 = "";                                // 로그인최종시간
boolean bauth_key  = false;                             // 검증결과
/*-------------------------------------------------------------------------*/
/* Request에서 데이터를 얻어옴(사용자정보).                                */
/*-------------------------------------------------------------------------*/
String  auth_key   = clib.strDefault(request.getParameter("auth_key"  ), "" );
String  user_email = clib.strDefault(request.getParameter("user_email"), "" );
String  auto_login = clib.strDefault(request.getParameter("auto_login"), "N");
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
              + "&auto_login="         + auto_login
              ;
logger.info(param);
/*-------------------------------------------------------------------------*/
/* Begin.                                                                  */
/*-------------------------------------------------------------------------*/
try {
```

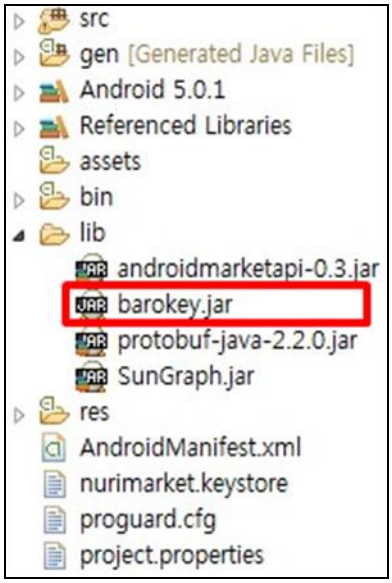
```
    /*-------------------------------------------------------------------*/
    /* 사용자정보 조회.                                                  */
    /*-------------------------------------------------------------------*/
    bdto = udao.reads(user_email);
    /*-------------------------------------------------------------------*/
    /* 사용자정보가 존재한 경우.                                         */
    /*-------------------------------------------------------------------*/
    if (bdto != null) {
        /*---------------------------------------------------------------*/
        /* 사용하지 않는 사용자인 경우.                                  */
        /*---------------------------------------------------------------*/
        if ("N".equals(bdto.getUse_yn())) {
            HTML = "USE_ERR";
        /*---------------------------------------------------------------*/
        /* 사용하는 사용자인 경우.                                       */
        /*---------------------------------------------------------------*/
        } else {
            /*-----------------------------------------------------------*/
            /* 로그인 최종 시간 Edit.                                    */
            /*-----------------------------------------------------------*/
            login_time = Long.toString(barokey.get_logintime(bdto.getCycle_time()));
            /*-----------------------------------------------------------*/
            /* 로그인 최종시간이 생성주기 보다 큰 경우.                  */
            /*-----------------------------------------------------------*/
            if (Long.parseLong(login_time) > Long.parseLong(bdto.getLogin_time())) {
                /*-------------------------------------------------------*/
                /* 인증키 검증.                                          */
                /*-------------------------------------------------------*/
                bauth_key    =    barokey.verifyKEYP(bdto.getSecure_key(),    bdto.getCycle_time(),
auth_key);
                /*-------------------------------------------------------*/
                /* 인증키 검증(성공).                                    */
                /*-------------------------------------------------------*/
                if (bauth_key == true) {
                    /*---------------------------------------------------*/
                    /* 쿠키 생성.                                        */
                    /*---------------------------------------------------*/
                    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  ());
                    /*---------------------------------------------------*/
                    /* 쿠키 생성.                                        */
                    /*---------------------------------------------------*/
                    if ("Y".equals(auto_login)) {
                        Date now = new Date();
                        String timestamp = now.toString();
                        Cookie cookie = new Cookie ("SES_USER_NO", bdto.getUser_no());
                        //cookie.setMaxAge(7 * 24 * 60 * 60);
```

```
                    cookie.setMaxAge(60*60*24*365); // 쿠키 유지기간-1년
                    cookie.setPath("/");
                    response.addCookie(cookie);
                }
                /*-------------------------------------------------------*/
                /* 로그인 최종 시간을 Update.                            */
                /*-------------------------------------------------------*/
                udao.updateLoginTime(user_email, login_time);
                HTML = bdto.getUser_no();
            /*-----------------------------------------------------------*/
            /* 인증키 검증(실패).                                        */
            /*-----------------------------------------------------------*/
            } else {
                HTML = "KEY_ERR";
            }
        /*---------------------------------------------------------------*/
        /* 로그인 최종시간이 생성주기 보다 작거나 같은 경우.             */
        /*---------------------------------------------------------------*/
        } else {
            HTML = "KEY_ERR";
        }
    }
    /*-------------------------------------------------------------------*/
    /* Log정보를 insert.                                                 */
    /*-------------------------------------------------------------------*/
    bdto.setIp_addr(ip_addr);
    ldao.create(bdto);
/*-----------------------------------------------------------------------*/
/* 사용자정보가 존재하지 않는 경우.                                      */
/*-----------------------------------------------------------------------*/
} else {
    HTML = "ID_ERR";
}
json.put("HTML", HTML);
out.println(json);
/*-------------------------------------------------------------------------*/
/* 예외사항 처리(Exception).                                               */
/*-------------------------------------------------------------------------*/
} catch(Exception e) {
    logger.info("Exception = [" + e + "]");
    e.printStackTrace();
/*-------------------------------------------------------------------------*/
/* Finally.                                                                */
/*-------------------------------------------------------------------------*/
} finally {
    logger.info("(result_user.jsp)Ending.....................");
}
%>
```

2. 안드로이드 폰인 경우

1) 인증키 생성기 부분

애플리케이션에 로그인 시 비밀번호란에 입력할 **일회용 인증키**를 생성하는 API는 "**barokey.jar**"로 제공되며, Eclipse나 Android studio를 사용하는 경우 libs 디렉토리에 "**barokey.jar**"를 위치시켜야 한다.

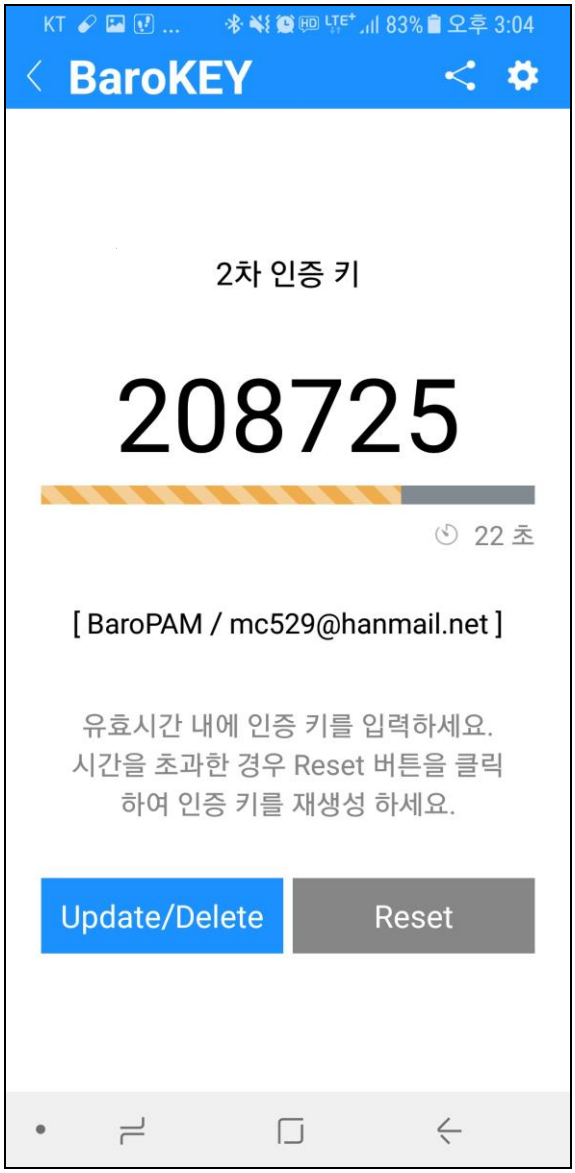


애플리케이션에 로그인 시 입력할 비밀번호인 **일회용 인증키**를 생성하는 프로그램에 다음과 같은 코드를 삽입하면 된다.

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

파라미터	설명	비고
login_id	로그인 화면의 로그인-ID 항목에 입력한 ID를 설정.	
phone_no	TelephonyManager 클래스를 이용해서 앱 내부에서 얻어낸 스마트 폰 번호를 설정.	
cycle_time	개인별로 지정한 **일회용 인증키**의 생성 주기(3~60초)를 설정. 만약, 개인별로 지정한 **일회용 인증키**의 생성 주기가 다른 경우 **일회용 인증키**가 다르게 생성될 수 있다.	

화면 예시)



화면 Layout 예시)

```
<?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>
```

프로그램 예시)

```
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();
                }
            }
        }
    }
}
```

3. 아이폰인 경우

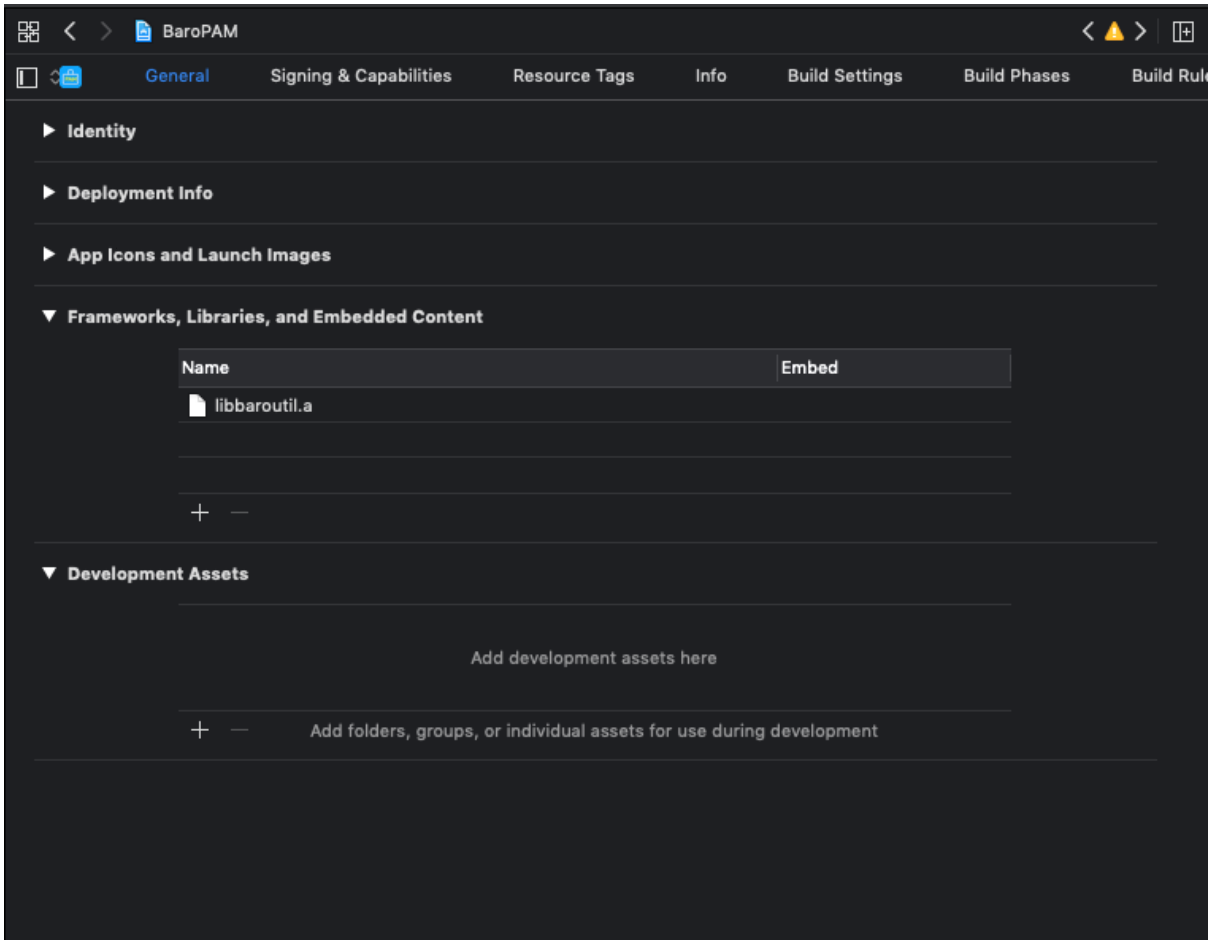
1) 인증키 생성기 부분

애플리케이션에 로그인 시 비밀번호란에 입력할 **일회용 인증키**를 생성하는 API는 "**libbaroutil.a**"로 제공되며, 이 파일에는 barokey, barocrypt, base64 관련 라이브러리를 포함하는 NSObject Interface용 라이브러리 파일이다.

라이브러리 파일은 두가지 종류로 제공한다. XCode의 iPhone simulator 용과, iPhone용 두가지를 필요에 따라 libbaroutil.a로 변경하여 사용한다.

- libbaroutil.a.iphone : iPhone용
- libbaroutil.a.simul : iPhone simulator 용

이 파일은 아래와 같이 XCode의 프로젝트 설정시에 등록하여 사용한다.



BaroPAM 관련된 API는 다음과 같다. 함수는 C 함수 Interface로 되어 있어, 입력값의 자료형은 C 함수 스타일로 표기한다. 사용 예의 소스는 iOS swift 5.0 이상으로 작성된 코드이다.

generateKEYL 함수

애플리케이션에 로그인/인증 시 사용하는 **일회용 인증키**를 생성하는 함수이다.

입력변수	const char *login_id	로그인 화면의 로그인-ID 항목에 입력한 ID를 설정해야 한다.
	const char *phone_no	사용자 스마트 폰번호다. **안드로이드용 앱에서와는 달리, iOS에서 사용자의 스마트 폰번호를 얻어 오지 않고 서버의 인증 모듈에서 사용할 사용자의 스마트 폰번호를 앱에서 직접 등록하여 관리하며, 등록된 스마트 폰번호를 선택하여 사용한다.**
	const char *cycle_time	개인별로 지정한 **일회용 인증키**의 생성 주기(3~60초)와 반드시 일치해야 한다. 만약, 개인별로 지정한 **일회용 인증키**의 생성 주기가 다른 경우 **일회용 인증키**가 다르게 생성될 수 있다.
	const char *key_method	**일회용 인증키**의 인증 방식(app1, app256, app384, app512: 앱)으로 "**app512**"를 설정한다.
리턴 값	**일회용 인증키**	생성한 **일회용 인증키**를 반환한다.

swift 5.0 이상에서의 사용예시)

```
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)
        }
    })
}
```

화면 예시)

화면 Layout 예시)

Storyboard를 의미한다. 각 파라메터에 대한 의미는 developer.apple.com을 참고한다.

```
<!—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-Tl-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-Tl-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 입력하세요. 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-2l6">
                <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="_tf0TP" 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>
```

프로그램 예시)

```
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
```

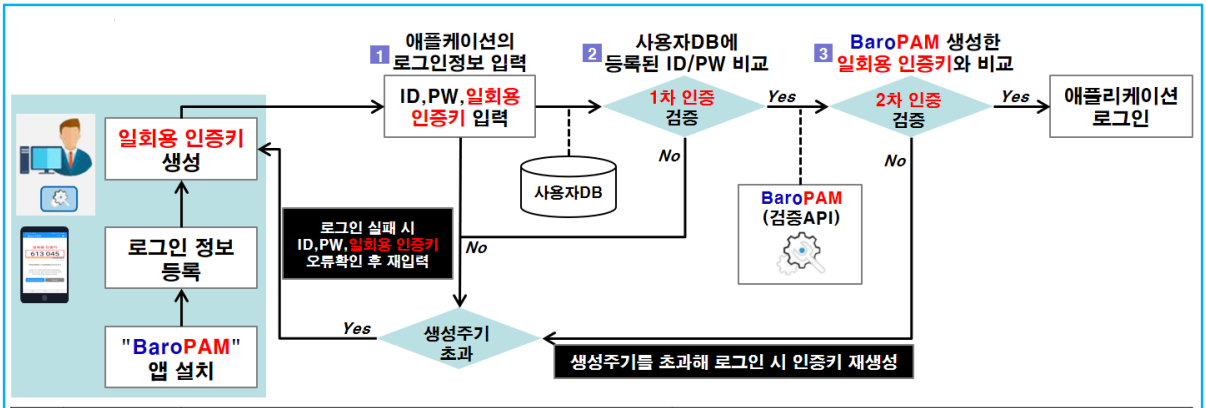
```
        print("--------> W(otp.REG_DT), W(otp.LOGIN_ID), W(otp.SYSTEM_NM), W(otp.CYCLE_TIME)")
    }

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

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

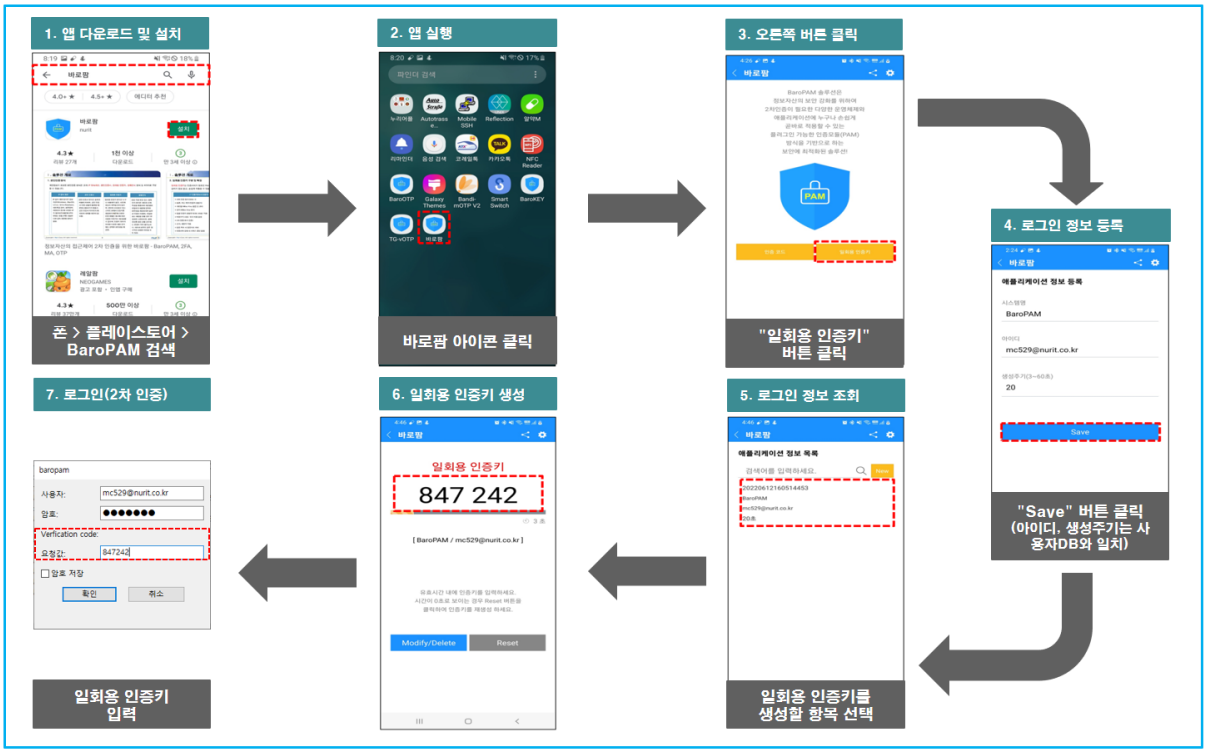
2. BaroPAM 적용

2.1 BaroPAM 적용 프로세스



No	구분	내용	설명
1	2차 인증 앱 (BaroPAM)	BaroPAM 앱 다운로드 및 설치	-안드로이드폰: 구글 플레이스토어 접속 및 "바로팜" 앱 다운로드 -아이폰: 애플 앱스토어 접속 및 "BaroPAMs" 앱 다운로드
2		BaroPAM 앱 실행	
3		"일회용 인증키" 버튼 선택	-실행된 앱 하단 오른쪽 버튼 선택
4		애플리케이션의 로그인 정보 등록	-시스템명: 애플리케이션 명칭(임의 등록 가능) -아이디: 애플리케이션의 로그인 -ID -생성주기: 3~60초 사이 입력
5		저장 후 목록 조회 시 나오는 항목 클릭, 일회용 인증키 생성	-아이폰 사용자는 본인 확인 절차를 거치지 않은 경우 본인 인증 절차를 진행
6		애플리케이션에 접속해 ID, PW, 일회용 인증키를 입력해 로그인	-아이디, 생성주기, 일회용 인증키를 이용해 2차 인증 검증

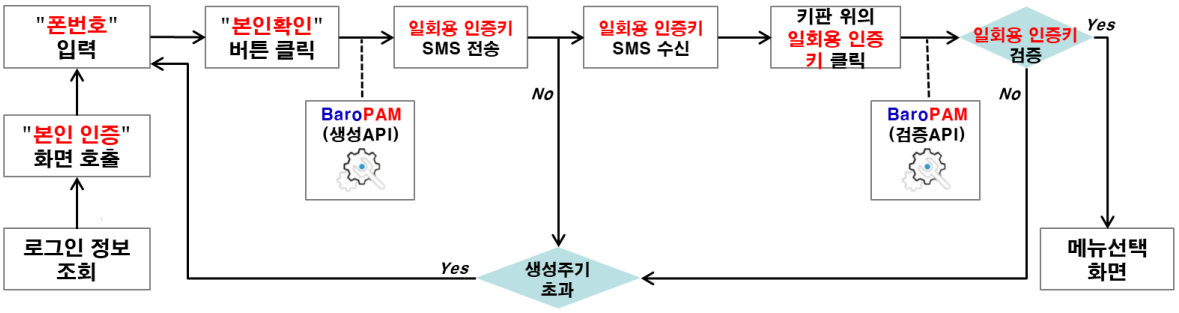
2.2 BaroPAM 적용 화면



2.3 본인확인 적용 프로세스

아이폰(iPhone)의 기기정보를 얻지 못해서 **2차 인증키(일회용 인증키)**를 생성하기 위해서 로그인 정보 항목을 선택했을 때 "**일회용 인증키**" 생성 화면으로 이동하지 않은 경우가 발생할 수 있다.

또한, 타인의 폰번호를 부정으로 사용하지 못하도록 하기 위해서 별도의 본인확인 기능을 적용할 필요가 있는데, "**BaroPAM**" 앱에서는 자체 알고리즘을 적용하여 자체적으로 본인확인 절차를 실행하고 있다.

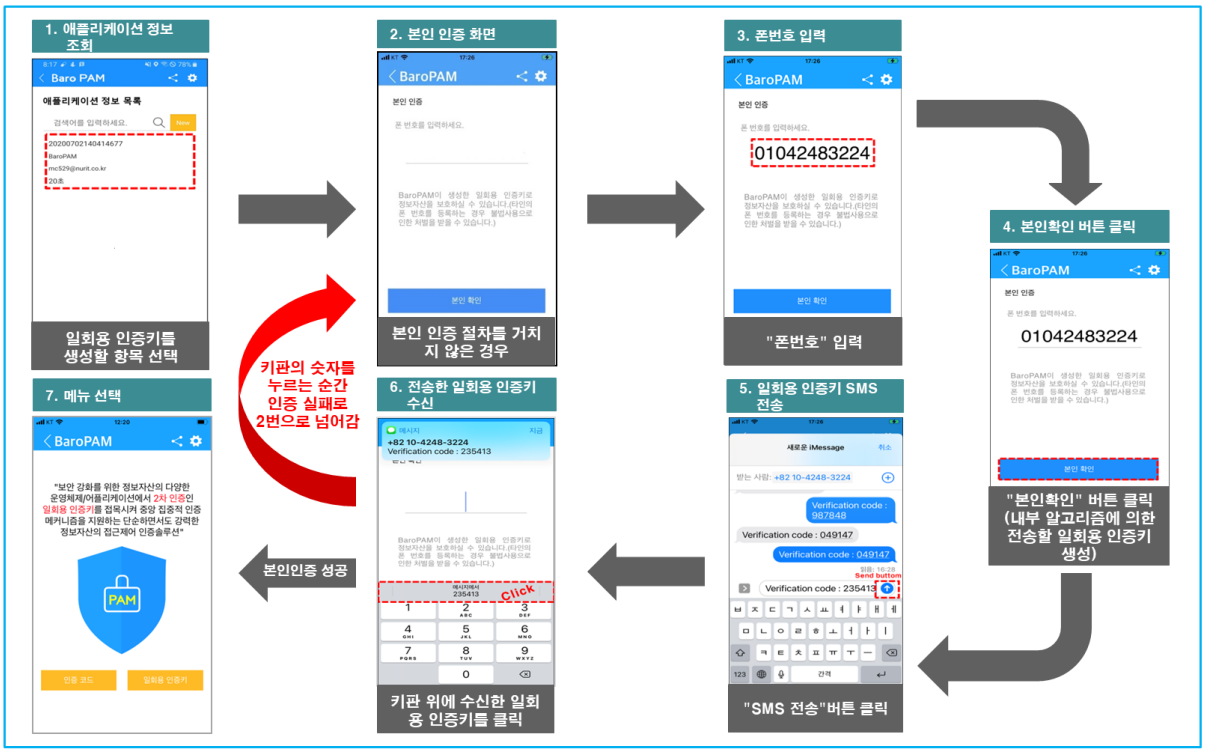


No	구분	내용	설명
1	아이폰 본인확인	로그인 정보 목록에서 일회용 인증키를 생성할 항목 클릭	-본인확인 절차를 거치지 않은 경우 본인인증 절차를 진행
2		"본인인증" 화면 호출	
3		"폰번호" 입력	-본인인증할 폰번호를 입력
4		"본인확인" 버튼 클릭	-내부 알고리즘에 의하여 "Verification code"인 일회용 인증키를 생성 -생성한 일회용 인증키를 전송하기 위한 SMS 연동 화면 호출
5		"Verification code" SMS 전송	-일회용 인증키를 확인한 후 "SMS 전송" 버튼을 클릭
6		"Verification code" SMS 수신	-본인 폰으로 전송한 일회용 인증키가 키판 위에 표시 -폰번호가 틀린 경우 타인 폰으로 일회용 인증키가 전송됨
7		키판 위의 "Verification code" 클릭	-내부 알고리즘에 의하여 일회용 인증키를 검증 -검증에 성공하면 메뉴선택 화면, 실패하면 본인인증 화면이 호출

2.4 본인확인 적용 화면

아이폰(iPhone)의 기기정보를 얻지 못해서 **2차 인증키(일회용 인증키)**를 생성하기 위해서 로그인 정보 항목을 선택했을 때 "**일회용 인증키**" 생성 화면으로 이동하지 않은 경우가 발생할 수 있다.

또한, 타인의 폰번호를 부정으로 사용하지 못하도록 하기 위해서 별도의 본인확인 기능을 적용할 필요가 있는데, "BaroPAM" 앱에서는 자체 알고리즘을 적용하여 자체적으로 본인확인 절차를 실행하고 있다.



참고) SMS로 전송한 OTA key가 수신은 되었는데 키판 위에 표시되지 않거나 SMS로 전송한 OTA key가 수신되지 않은 경우

아이폰의 "**암호 자동 완성 기능**"이 설정되지 않아서 발생한다. "BaroPAM" 앱을 설치한 후 iOS12 부터는 더욱 편리한 암호 자동 완성 기능을 반드시 설정해야 한다. (아이폰의 "**설정**" -> "**암호**" -> "**암호 자동 완성**" -> "**허용**")

2.5 애플리케이션 로그인

애플리케이션 로그인 시 지문인식 OTP카드와 로그인-ID를 혼용하여 사용할 경우에는 다음과 같이 사용자 정보 테이블에 사용자가 **일회용 인증키**를 생성할 폰번호(USER_PHONE, 지문인식 OTP카드를 사용하는 경우 폰번호에 지문인식 OTP카드에 부여한 Secure key를 입력해야 함), **일회용 인증키** 생성주기(CYCLE_TIME), 로그인 최종시간(LOGIN_TIME)컬럼을 추가해야 한다.

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

로그인 최종시간을 추가하는 이유는 **일회용 인증키** 생성 주기 내에 동일한 로그인-ID에 사용자 한명만 로그인 가능하게 제한하여 재사용 및 중간자 공격(Man-in-the-middle attack)에 대비하기 위함이며, 로그인이 성공한 경우에 로그인 최종시간을 Update해야 한다.

만약, 지문인식 OTP카드를 사용하지 않고 로그인-ID를 사용하고 싶은 경우에는 폰번호에 사용자가 **일회용 인증키**를 생성할 폰번호를 입력해서 사용하면 된다.

먼저, 애플리케이션의 로그인 화면과 관련된 "**시스템명, 로그인-ID, 인증주기**"을 "BaroPAM" 앱의 "**로그인 정보 등록**" 화면에서 동일하게 입력해야 한다.

```
<%@ 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"); %>
<%
/*------------------------------------------------------------------------*/
/* 변수선언 및 초기화.                                                    */
/*------------------------------------------------------------------------*/
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 = "";                                 // 로그인최종시간
boolean bauth_key  = false;                              // 검증결과
/*-------------------------------------------------------------------------*/
/* Request에서 데이터를 얻어옴.                                            */
/*-------------------------------------------------------------------------*/
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 {
    /*---------------------------------------------------------------------*/
    /* 사용자정보 조회.                                                    */
    /*---------------------------------------------------------------------*/
    bdto = udao.read(user_id);
    /*---------------------------------------------------------------------*/
    /* 사용자 정보가 존재한 경우.                                          */
    /*---------------------------------------------------------------------*/
    if (bdto != null) {
        /*-----------------------------------------------------------------*/
        /* 비밀번호가 동일한 경우.                                         */
        /*-----------------------------------------------------------------*/
        if (user_pw.equals(bdto.getUser_pw())) {
            /*-------------------------------------------------------------*/
            /* 로그인 최종 시간 Edit.                                      */
            /*-------------------------------------------------------------*/
            login_time = Long.toString(barokey.get_logintime(bdto.getCycle_time()));
            /*-------------------------------------------------------------*/
            /* 로그인 최종시간이 생성주기 보다 큰 경우.                    */
            /*-------------------------------------------------------------*/
            if (Long.parseLong(login_time) > Long.parseLong(bdto.getLogin_time())) {
                /*---------------------------------------------------------*/
                /* 일회용 인증키 검증.                                     */
                /*---------------------------------------------------------*/
                bauth_key  = barokey.verifyKEYL(bdto.getUser_id(), bdto.getUser_phone().replace("-",
"") , bdto.getCycle_time(), auth_key);
```

```
            /*-------------------------------------------------------------*/
            /* 로그인 최종시간이 생성주기 보다 작거나 같은 경우.           */
            /*-------------------------------------------------------------*/
             } else {
                bauth_key = false;
             }
             /*-------------------------------------------------------------*/
             /* 일회용 인증키 검증(성공).                                   */
             /*-------------------------------------------------------------*/
             if (bauth_key == true) {
                /*-------------------------------------------------------------*/
                /* 쿠키 생성.                                                  */
                /*-------------------------------------------------------------*/
                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  ());
                /*-------------------------------------------------------------*/
                /* 로그인 최종 시간을 Update.                                  */
                /*-------------------------------------------------------------*/
                udao.updateLoginTime(user_id, login_time);
                HTML = bdto.getUser_no();
             /*-------------------------------------------------------------*/
             /* 일회용 인증키 검증(실패).                                   */
             /*-------------------------------------------------------------*/
             } else {
                HTML = "KEY_ERR";
             }
          /*-------------------------------------------------------------*/
          /* 비밀번호가 동일하지 않는 경우.                              */
          /*-------------------------------------------------------------*/
          } else {
             HTML = "PW_ERR";
          }
          /*-------------------------------------------------------------*/
          /* Log정보를 insert.                                           */
          /*-------------------------------------------------------------*/
          bdto.setIp_addr(ip_addr);
          ldao.create(bdto);
       /*-------------------------------------------------------------*/
       /* 사용자 정보가 존재하지 않는 경우.                           */
       /*-------------------------------------------------------------*/
       } else {
          HTML = "ID_ERR";
       }
       json.put("HTML", HTML);
       out.println(json);
/*-------------------------------------------------------------*/
/* 예외사항 처리(Exception).                                   */
```

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

3. About BaroPAM

Version 1.0 - Official Release - 2016.12.1
Copyright ⓒ Nurit corp. All rights reserved.
http://www.nurit.co.kr

제 조 사 : 주식회사 누리아이티
등록번호 : 258-87-00901
대표이사 : 이종일
대표전화 : 02-2665-0119(영업문의/기술지원)
이 메 일 : mc529@nurit.co.kr
주 소 : 서울시 강서구 마곡중앙2로 15, 913호(마곡동, 마곡테크노타워2)