GPKI Certificate Login Service

Summary

The administrative electronic signature certificate is electronic information issued to administrative institutions, assistant institutions, public institutions, banks and users to confirm and prove that the administrative electronic signature is genuine. The electronic signature is a means of guaranteeing the reliability of the electronic documents. The administrative electronic signature is a certificate (GPKI) for government employees and administrative affairs managers and provides services in connection with the public certificate (NPKI).

- The administrative electronic signature is layered into RootCA, RA and LRA, and each approves and manages the permissions of its sub institutions. In order to keep an effective administrative electronic signature certification system, roles are managed per layer.
- The administrative electronic signature certificate is issued as institution use and as personal use and should follow the issuance process based on the purpose of use.

Function Flow

Function name	***Function flow***
Certificate login	Select certificate→ Enter password → Request to the certification institute → ***Check permissions*** → ***Set up sessions*** → ***create log- in log*** → ***Set up menu per permission*** → Load screen per permission

- If the e- government standard framework is not used, refer to the Certificate login of the element technology. (functions are provided to prevent depending on the framework)

Prerequisites

In order to use the GPKI certificate login service provided in this project, you have to receive the administrative electronic signature certificate. This is issued by Government Certification Management Authority. The following is the procedure of issuance.

- Issuance process

① Download and fill out the administrative electronic signature application form .
② Submit the form to the RA or LRA.
③ The RA or LRA manager check the identification of the applicant.
④ The manager emails a issuance guide to identified applicants.
⑤ User certificate is issued.

For details, visit Administrative Electronic Signature Government Certification Management Authority (http://www.gpki.go.kr).

Description

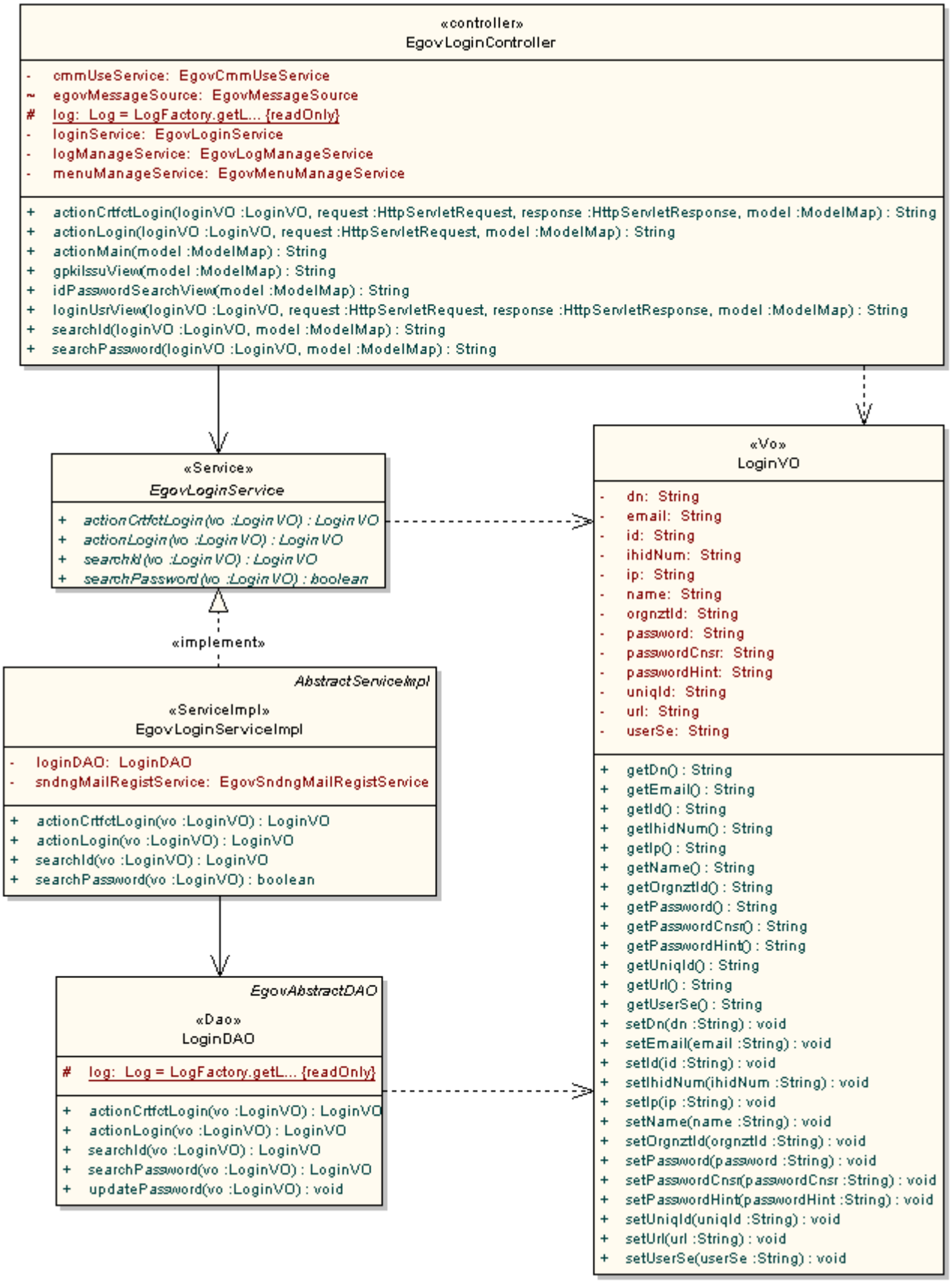
The GPKI certificate API is a management tool to create an application that contains or provides a security service based on the password theory. Even a developer who is not knowledgeable in passwords can develop a program that provides security service, by using the standard security API module based on the security

information. This function handles GPKI certificate login so that the efficiency of administration can be improved by sharing GPKI constructed by Government Certification Management Authority.

Related Sources

Types	***Target Soruce***	***Notes***
Controller	egovframework.com.uat.uia.web.EgovLoginController.java	Controller class for log- in
Service	egovframework.com.uat.uia.service.EgovLoginService.java	Service interface for log- in
Service	egovframework.com.sym.log.service.EgovLogManageService.java	Service interface for creation of log- in log
Service	egovframework.com.sym.mpm.service.EgovMenuManageService.java	Service interface for menu inquiry after log- in
ServiceImpl	egovframework.com.uat.uia.service.impl.EgovLoginServiceImpl.java	Service implementation class for log- in
VO	egovframework.com.uat.uia.service.LoginVO.java	VO class for log- in
DAO	egovframework.com.uat.uia.service.impl.LoginDAO.java	Data processing class for log- in
Query XML	resources/egovframework/sqlmap/com/uat/uia/EgovLoginUsr_SQL.xml	Query XML for log- in
JSP	WEB_INF/jsp/egovframework/cmm/uat/uia/EgovLoginUsr.jsp	Log- in page
js	/js/egovframework/cmm/uat/uia/EgovGpkiInstall.js	Javascript file for GPKI SecureWeb installation
js	/js/egovframework/cmm/uat/uia/EgovGpkiVariables.js	Javascript file that contains variables required for GPKI certification processing
js	/js/egovframework/cmm/uat/uia/EgovGpkiFunction.js	Javascript file for for GPKI certification processing

Class diagram



Related Table

Table name	***Table name(English)***	***Note***
Business user	COMTNEMPLYRINFO	Manage business user information
Log- in log	COMTNLOGINLOG	Manage login- in log
Menu	COMTNMENUINFO	Manage menu information

Related Settings

If you want to implement GPKI certificate log- in function, modify EgovLoginController.java as in the following.

- Modify the log- in screen processing.

```
public String loginUsrView(@ModelAttribute("loginVO") LoginVO loginVO,
    HttpServletRequest request,
    HttpServletResponse response,
    ModelMap model)
    throws Exception {
        // # When processing GPKI certification
        GPKIHttpServletResponse gpkiresponse = null;
        GPKIHttpServletRequest gpkirequest = null;

        try{

                gpkiresponse=new GPKIHttpServletResponse(response);
                gpkirequest= new GPKIHttpServletRequest(request);
                gpkiresponse.setRequest(gpkirequest);
                model.addAttribute("challenge", gpkiresponse.getChallenge());
                return "cmm/uat/uia/EgovLoginUsr";

        }catch(Exception e){
                return "cmm/egovError";
        }
}
```

- Modify the certificate log- processing as in the following.

```
public String actionCrtfctLogin(@ModelAttribute("loginVO") LoginVO loginVO,
    HttpServletRequest request,
    HttpServletResponse response,
    ModelMap model)
    throws Exception {
        // # When processing GPKI certification
        // connection IP
        String userIp = EgovClntInfo.getClntIP(request);

        // 1. GPKI certification
        GPKIHttpServletResponse gpkiresponse = null;
        GPKIHttpServletRequest gpkirequest = null;
        String dn = "";
        try{
                gpkiresponse = new GPKIHttpServletResponse(response);
                gpkirequest = new GPKIHttpServletRequest(request);
                gpkiresponse.setRequest(gpkirequest);
                X509Certificate cert = null;

                byte[] signData = null;
                byte[] privatekey_random = null;
                String signType = "";
                String queryString = "";

                cert = gpkirequest.getSignerCert();
                dn = cert.getSubjectDN();

                java.math.BigInteger b = cert.getSerialNumber();
```

```
                b.toString();
                int message_type =   gpkirequest.getRequestMessageType();
                if( message_type == gpkirequest.ENCRYPTED_SIGNDATA ||
                     message_type == gpkirequest.LOGIN_ENVELOP_SIGN_DATA ||
                     message_type == gpkirequest.ENVELOP_SIGNDATA ||
                     message_type == gpkirequest.SIGNED_DATA){
                     signData = gpkirequest.getSignedData();
                     if(privatekey_random != null) {
                           privatekey_random     = gpkirequest.getSignerRValue();
                     }
                     signType = gpkirequest.getSignType();
                }
                queryString = gpkirequest.getQueryString();
          }catch(Exception e){
                return "cmm/egovError";
          }

          // 2. In the business user table, use dn value to request user ID and PW
          // certify them in the form of generic login
          if (dn != null && !dn.equals("")) {

                loginVO.setDn(dn);
                LoginVO resultVO = loginService.actionCrtfctLogin(loginVO);
                if (resultVO != null && resultVO.getId() != null &&
                     !resultVO.getId().equals("")) {
                     // 3. spring security connection
                     return "redirect:/j_spring_security_check?j_username=" +
                               resultVO.getUserSe() + resultVO.getId() +
                                "&j_password=" + resultVO.getUniqId();
                } else {
                     model.addAttribute("message",
                                         egovMessageSource.getMessage("fail.common.login"));
                     return "cmm/uat/uia/EgovLoginUsr";
                }
          } else {
                model.addAttribute("message",
                                    egovMessageSource.getMessage("fail.common.login"));
                return "cmm/uat/uia/EgovLoginUsr";
          }
}
```

Environmental settings

The following items and their configurations are required to use GPKI certification function.

Check GPKI API install file

For the function of GPKI certificate login, visit Government Certification Management Authority (http://www.gpki.go.kr) and apply for and receive the standard security API suitable for the system. The standard security API in the server is for IBM AIX, you cannot use it in Windows and other UNIX systems

Components of Standard API

Classification	***format***	***File name/folder***	***Explanation***
Standard API Native module	Library	libgpkiapi64.a	For IBM AIX (government)
Standard API	Library	libgpkiapi64_jni.a	For IBM AIX (government)

Native module			
Standard API Native module	Library	libibmldap64n.a	For IBM AIX (private)
Environment file(conf)	Environment file	gpkiapi.conf	Include information required for certificate verification
Test program (sample)	code	/java	Cert.java, Cms.java, Crypto.java, Ivs.java, Main.java, Tsa.java, Util.java (source code)
Test program (sample)	Executable file	/class	/Sample (data required to run the test program)Cert.class, Cms.class, Crypto.class, Ivs.class, Main.class, Tsa.class, Util.class (test program)
Standard API	Jar file	libgpkiapi_jni.jar	Standard Security API

Set up class and library paths

export GPKI_HOME=/product/jeus/egovProps/libgpkiapi
export CLASSPATH=$GPKI_HOME/libgpkiapi_jni.jar:$CLASSPATH
export LIBPATH=/product/jeus/egovProps/libgpkiapi/gpkiapi
export PATH=$PATH:/product/jeus/egovProps/libgpkiapi/gpkiapi

In order to use the standard security API (libgpkiapi_jni.jar) for Java, the jar file should exist in the class path and JNI file path called by the standard security API should be specified. The JNI file is connected with the standard security API and LDAP libraries and the paths of these two library should be also specified.

Location of Certificate for Testing

/product/jeus/egovProps/gpkisecureweb/certs/SVR1311000011_env.cer
/product/jeus/egovProps/gpkisecureweb/certs/SVR1311000011_env.key
/product/jeus/egovProps/gpkisecureweb/certs/NPKIRootCA1.der
/product/jeus/egovProps/gpkisecureweb/certs/GPKIRootCA1.der

Configure the profile file

dsjdf.properties

#[Log- related]
logger.driver=com.dsjdf.jdf.DefaultLoggerWriter

#[Directory where logs are left]
#Log directory 의 Absolute Path
logger.dir=/product/jeus/egovProps/gpkisecureweb/log

#[Log level]
logger.sys.trace=false
logger.err.trace=true
logger.warn.trace=false
logger.info.trace=true
logger.debug.trace=false
logger.autoflush=true

#[Project config file or Server config file]
pbf.propertiesFile=/product/jeus/egovProps/gpkisecureweb/conf/gpkisecureweb.properties

This file is for DSJDF environment configuration and used when calling from DSJDF's Config. In order to operate this file, you have to use the java - D option to specify the absolute path of the file in com.dsjdf.jdf.config.file, when running the application using DSJDF. Or just place it in the root folder of Web

Application Server. You can configure logger.dir=/product/jeus/egovProps/gpkisecureweb/log to stack the GPKI certificate related log files.

gpkisecureweb.properties

```
#============================================================================
# This file configures GPKISecureWeb environment for Servlet and JSP.
# JavaBean directory in WAS is used as Root and should be located in /conf/gpkisecureweb.properties
# If not, GPKISecureWEBConfigException occurs.
# The following is the installation location for this file
#     => Within in Webserver/WAS: [GPKISecureWeb installation directory]/conf/gpkisecureweb.properties
#============================================================================
=
#GPKI server certificate location //If changed, restart WAS.
#Absolute Path of GPKI server certificate
GPKISecureWeb.CertFilePathName = /product/jeus/egovProps/gpkisecureweb/certs/SVR1311000011_env.cer
GPKISecureWeb.PrivateKeyFilePathName =
/product/jeus/egovProps/gpkisecureweb/certs/SVR1311000011_env.key
GPKISecureWeb.PrivateKeyPasswd = q6e6qysp

#GPKI API path configuration
GPKISecureWeb.gpkiapi.ConfFilePath=/product/jeus/egovProps/gpkisecureweb/conf
GPKISecureWeb.CheckChallenge = yes

# Number of ROOTCA certificates
GPKISecureWeb.TrustedROOTCACert.count=2

# ROOTCA certificate location
# ROOTCA certificate should be updated before the term of validity has expired.
# Double click ROOTCA certificate in Windows to check
# the term of validity on the screen.
# ROOTCA certificate can be obtained by using LDAP browser.
# For GPKI, in ldap://152.99.56.86:389, CN=Root CA,OU=GPKI,O=Government of Korea,C=KR,
# For NPKI, in ldap://dir.signkorea.com:389, CN=KISA RootCA 1,OU=Korea Certification Authority
Central,O=KISA,C=KR
# Download from LDAP's cACertificate;binary entry.
# ROOTCA certificate locations as many as ROOTCA certificates should be created.
GPKISecureWeb.TrustedROOTCACert.FilePathName.1 =
/product/jeus/egovProps/gpkisecureweb/certs/NPKIRootCA1.der
GPKISecureWeb.TrustedROOTCACert.FilePathName.2 =
/product/jeus/egovProps/gpkisecureweb/certs/GPKIRootCA1.der
```

What is to determine in the above is to the location of #GPKI server certificate location and #GPKI API path configuration. The certificate is owned per server and there two files (cer and key files) as configured in the above. The next to be seen is the certificate password. There should be gpki.conf and gpkiapi.conf in Gpkiapi.confFilePath. gpkiapi.conf is provided by the standard API and gpki.conf exists in the directory where GPKIClientActiveX is installed.

Script configuration

EgovGpkiVariables.js

```
// Base64 Encoding information of server certificate
var ServerCert =
"MIID5jCCAs6gAwIBAgIQSYexPQLd1LW4+HKjW0HZIDANBgkqhkiG9w0BAQUFADBQMQswCQYDVQQGEwJLUjEc
MBoGA
1UEChMTR292ZXJubWVudCBvZiBLb3JlYTENMAsGA1UECxMER1BLSTEUMBIGA1UEAxMLQ0ExMzEwMDAwMDE
wHhcNMDkwMjAzMDI1MTQxWhcN
```

MTEwNTAzMDI1MTQxWjBdMQswCQYDVQQGEwJLUjEcMBoGA1UECgwTR292ZXJubWVudCBvZiBLb3JlYTEYMB
YGA1UECwwPR3JvdXAgb2YgU2V
...";

In EgovGpkiVariables.js, ServerCert insert Base64 encoding data for the certificate to be put into the server. This data should be changed for each new certificate. The following shows how to extract data.

Base64Encode.java

```
import com.gpki.gpkiapi.cert.X509Certificate;
import com.gpki.gpkiapi.util.*;

public String Base64Encode() {

    X509Certificate x509Cert = null;
    byte[ ] cert = null;
    String base64cert = null;
    try {
        x509Cert = Disk.readCert("/product/jeus/egovProps/gpkisecureweb/certs/SVR..._env.cer");
        cert = x509Cert.getCert();
        Base64 base64 = new Base64();
        base64cert = base64.encode(cert);
        System.out.println("The data converted to Base64: " + base64cert);

    } catch (GpkiApiException e) {
        e.printStackTrace();
    }
}
```

Use SVR1311000011_env.cer certificate put into the server to output the encoded data. Put this into ServerCert value in EgovGpkiVariables.js.

Screen and execution manual

GPKI certificate login

Action	*URL*	*Controller method*	*QueryID*
Login screen	/uat/uia/egovLoginUsr.do	loginUsrView	
Certificate login	/uat/uia/actionCrtfctLogin.action	actionCrtfctLogin	loginDAO.actionCrtfctLogin

GPKI certificate login uses the certificate and password to check the certificate and, if normal, extract the dn value.
Use the dn value to retrieve the user information (ID and password) in the business user table.
Use the retrieved user information to create login log and call Spring Security to carry out permission certification and session configuration.

Select the storage type: specify the location where the certificate is stored.
Enter password: enter the password that suits the certificate.
Certificate login: use the certificate and password to create the encrypted data to be sent to the server and carry out certification.

Reference

N/A