**COMPETITIVENESS AND INNOVATION FRAMEWORK PROGRAMME**

ICT Policy Support Programme (ICT PSP)

Towards pan-European recognition of electronic IDs (eIDs)

Project acronym: STORK

Project full title: Secure Identity Across Borders Linked

STORK Service Providers

Start Guide

|  |  |
| --- | --- |
| **Deliverable Id :** |  |
| **Deliverable Name :** |  |
| **Status :** |  |
| **Dissemination Level :** |  |
| **Due date of deliverable :** |  |
| **Actual submission date :** |  |
| **Work Package :** | **WP5.8** |
| **Organisation name of lead contractor for this deliverable :** |  |
| **Author(s):** |  |
| **Partner(s) contributing :** | **PT,ES** |

**Abstract:** Quick Guide to configure and deploy Portuguese Service Providers.

**Project co-funded by the European Community under the ICT Policy Support Programme**

© Copyright by the STORK-eID Consortium

History

|  |  |  |  |
| --- | --- | --- | --- |
| *Version* | *Date* | *Modification reason* | *Modified by* |
| 0.1 | 22/12/2011 | Initial Guide | Ricardo Ferreira |

Table of contents

[History 2](#_Toc312448744)

[Table of contents 3](#_Toc312448745)

[List of abbreviations 4](#_Toc312448746)

[Executive summary 5](#_Toc312448747)

[1 Introduction 6](#_Toc312448748)

[2 Before you start 7](#_Toc312448749)

[2.1 Keystore 7](#_Toc312448750)

[2.2 Server Configuration 7](#_Toc312448751)

[2.2.1 Tomcat 5 7](#_Toc312448752)

[2.2.2 Tomcat 6 7](#_Toc312448753)

[2.2.3 JBoss 5 8](#_Toc312448754)

[2.2.4 Glassfish V3 8](#_Toc312448755)

[3 Quick Start 9](#_Toc312448756)

[3.1 Configurations 9](#_Toc312448757)

[3.2 Quick Compilation/Deployment 9](#_Toc312448758)

[3.2.1 Tomcat Server Deployment 9](#_Toc312448759)

[4 Installing Demo SP 10](#_Toc312448760)

[4.1 Configuration files 10](#_Toc312448761)

[4.1.1 sp.properties 10](#_Toc312448762)

[4.1.2 SamlEngineSignConfig.xml 12](#_Toc312448763)

[4.2 Start Application 13](#_Toc312448764)

[5 STORK API 15](#_Toc312448765)

[5.1 JARs 15](#_Toc312448766)

[5.2 Examples of use 15](#_Toc312448767)

[5.2.1 Generating a STORK Authentication Requests 15](#_Toc312448768)

[5.2.2 Validating and Reading STORK Authentication Responses 16](#_Toc312448769)

[6 Set of Test Certificates 17](#_Toc312448770)

List of abbreviations

<Abbreviation> <Explanation>

STORK Secure idenTity acrOss boRders linKed

PEPS Pan European Proxy Server

SP Service Provider

Executive summary

This document is meant to provide a quick and, also, a more detailed information on how to configure, build and deploy an SP application for use the STORK network.

First of all we need to have an application server up and running, so, first there are provided some basic configurations of the server.

Then this document describes what you need to know about different configurations of the several projects, in order to successfully build the application.

So, after reading this document, you should be able to configure, build and deploy the application by yourself.

# Introduction

This document is divided in several parts.

First of all, it’s presented a set of steps to be followed in order to configure an Application Server (Tomcat, JBoss, etc) to serve as container for your application.

Then, a quick guide describes how to run the application in a few minutes.

After that, the configurations are explained in a more detailed way, in order for you to change its specific aspects.

Finally, there is a FAQ section, were we answer the more common doubts.

# Before you start

Before you begin make sure that you have **Tomcat 5.5+** installed on your machine, ensuring that it has the **OpenSAML** library. The project build files are in **Maven2** format, so you also need it installed. The project is built using the version **1.5+** of the **Java** SDK.

In the next section, instructions are provided on how to deploy the project on Tomcat 6 or JBoss 5 or GlassFish v3.

In this document there will be used some variables which are going to be explained below:

* $TOMCAT\_HOME – Base directory of your tomcat server. (e.g. /home/user/apps/apache-tomcat-5.5.28)
* $JBOSS\_HOME – Base directory of your JBoss server. (e.g. /home/user/apps/jboss-5.1.0.GA)
* $SERVER\_CONFIG – JBoss server configuration name (e.g. default)

So, if you want to use the “default” configuration server, your full path will be /home/user/apps/jboss-5.1.0.GA/server/default

* $GLASSFISH\_HOME – Base directory of your Glasshfish server (e.g. /home/user/apps/glasshfishv3);

## Keystore

Copy your “certificates-demo-provider.jks” (the key store with your StorkDemoSP sign certificate and the public certificates of the Test and Production country environments) into a directory of your own choice, and make sure that the properties “keystorePath” on file STORKDemoSP \src\main\resources\eu \SamlEngineSignConfig.xml reflects the location of your SP “certificates-demo-provider.jks”;

For more information check 4.1.

## Server Configuration

### Tomcat 5

Extract the OpenSAML zip and copy the following libs to $TOMCAT\_HOME/common/endorsed

endorsed\xml-apis-2.9.1.jar

endorsed\resolver-2.9.1.jar

endorsed\serializer-2.9.1.jar

endorsed\xalan-2.7.1.jar

endorsed\xercesImpl-2.9.1.jar

### Tomcat 6

Create a folder named endorsed in $TOMCAT\_HOME.

Create a folder named shared in $TOMCAT\_HOME and a subfolder named lib in $TOMCAT\_HOME\shared

Edit the file $TOMCAT\_HOME\conf\catalina.properties and change the property shared.loader so that it reads:

shared.loader=${catalina.home}/shared/lib/\*.jar

Open the OpenSAML zip and copy the following libs to $TOMCAT\_HOME/endorsed

endorsed\xml-apis-2.9.1.jar

endorsed\resolver-2.9.1.jar

endorsed\serializer-2.9.1.jar

endorsed\xalan-2.7.1.jar

endorsed\xercesImpl-2.9.1.jar

### JBoss 5

Open the OpenSAML zip and copy the following libs to $JBOSS\_HOME/lib/endorsed

endorsed\xml-apis-2.9.1.jar

endorsed\resolver-2.9.1.jar

endorsed\serializer-2.9.1.jar

endorsed\xalan-2.7.1.jar

endorsed\xercesImpl-2.9.1.jar

### Glassfish V3

Open the OpenSAML zip and copy the following libs to $GLASSFISH\_HOME/ glassfish/lib/endorsed

endorsed\xml-apis-2.9.1.jar

endorsed\resolver-2.9.1.jar

endorsed\serializer-2.9.1.jar

endorsed\xalan-2.7.1.jar

endorsed\xercesImpl-2.9.1.jar

# Quick Start

This quick start allows you to set-up, compile and run the project in a few minutes, with only one instance of your Application Server. Remember that your server must be configured, as described in the previous chapter (see “Before you start”). Further information about configurations files, attributes, can be found on the next chapter.

## Configurations

1. Edit the file DemoSP\src\main\resources\sp.properties and change the following properties:

* **sp.return**=https://**insert.your.ip.here**/SP/ReturnPage
* **sp.pepsurl**=https://eu-id.teste.cartaodecidadao.gov.pt/PEPS/ServiceProvider (Quality environment)
* **sp.pepsurl**=https://eu-id.cartaodecidadao.gov.pt/PEPS/ServiceProvider (Production environment)

You now have a Service Provider configured to run on your IP address.

## Quick Compilation/Deployment

### Tomcat Server Deployment

You must compile, install and deploy the projects, in the following order.

1. In DemoSP folder type:

* mvn clean package
* copy target\DemoSP.war $TOMCAT\_HOME\webapps\SP.war

Finally, run your tomcat, start your browser and navigate to the following page: http://**your.ip.goes.here**:**portGoesHere**/SP/

# Installing Demo SP

## 

## Configuration files

Demo SP provides configuration files that can be tweaked. In this section we will explain what are those files are and what is each property.

## sp.properties

The sp.properties provides the main Service Provider configurations.

|  |  |
| --- | --- |
| **Key** | **Description** |
| sp.name | Service Provider name (e.g. Portal do Cidadão) |
| sp.sector | Sector for this service provider |
| sp.application | Application for this service provider |
| sp.country | PT |
| sp.environment | Connect environment of this DemoSP (TEST / PROD) |
| sp.qaalevel | QAA level of this Service Provider |
| sp.return | URL used when the Stork Country Service finishes the process |

Avaliable attributes for this SP:

|  |  |
| --- | --- |
| **Key** | **Description** |
| attribute.number | Number of attributes to be requested |
| attributeX.name | Name of the attribute X |
| attributeX.mandatory | If this attribute is not retrieved with a value, there is no authentication. |
| attributeX.value | Optional parameter that will set the value for special attributes such as isAgeOver |

Filtered countries for this SP:

|  |  |
| --- | --- |
| **Key** | **Description** |
| discountry.number | The number of countries that this SP doesn’t want to communicate |
| discountryX | The name of the country X |

Now, you must to choose between two options:

1.- Automatic control of countries: A XML file provided by your Production STORK Node is used.

2.- Manual control of countries. Then you need to configure each country in this sp.properties file.

|  |  |
| --- | --- |
| **Key** | **Description** |
| sp.versioncontrol | If “true” we are in the first option. And if is “false”, we are in the Manual Control of countries |

OPTION 1) sp.versioncontrol=true

|  |  |
| --- | --- |
| **Key** | **Description** |
| versioninfofile | Local path where is located the downloaded XML file with the country info. |

OPTION 2) sp.versioncontrol=false

|  |  |
| --- | --- |
| sp.pepsurl | URL of your country STORK Node. |

Avaliable countries for this SP:

|  |  |
| --- | --- |
| **Key** | **Description** |
| country.number | The number of possible countries that communicate with this SP |
| countryX.name | The name of the Country X |

## SamlEngineSignConfig.xml

This file is used to configure the Sign and Validation module for the SAML tokens used to send Authentication Request and receive Authentication Responses.

|  |  |
| --- | --- |
| **Key** | **Description** |
| keystorePath | Path were the JKS file is located |
| keyStorePassword | Password of the JKS |
| keyPassword | Password of the private certificate |
| issuer | Issuer of the private certificate |
| serialNumber | Serial Number of the certificate |
| keystoreType | Type of the keystore (JKS) |

In the keystore you should have a private certificate (to sign your SAML Authentication Requests). You must also have the public certificate of the Portuguese node (PEPS), so you can validate its Authentication Responses.

## Start Application

After performing the steps above, start your Application Server, open your browser and navigate to http://your.ip.addres/SP/ (if your server is listening at another port feel free to change the URL). You should be viewing the following page:



Figure 1 – Demo Service Provider Home Page

Once selected the Country’s electronic Certificate desired (Citizen-Country), you are redirected to the STORK network and interact with Portuguese node (PEPS) and with the Citizen-Country STORK Server.

At last, you will be redirected to http://your.ip.addres/SP/ReturnPage, and view the following page:

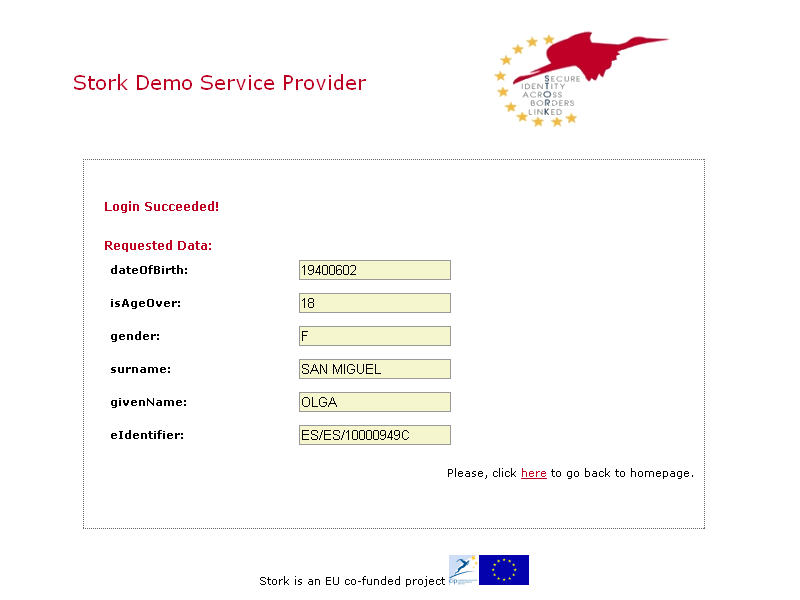


Figure 2 – Demo Service Provider Return Page

# STORK API

## JARs

Portuguese API is composed by two JARs:

* + pt-saml-engine-1.0.jar: is the one for validate and create SAML tokens.
  + spt-commons-1.0jar: is used for manage the data extracted from tokens.

## Examples of use

## Generating a STORK Authentication Requests

//Creating STORK Authentication Request --------------------------------

STORKAuthnRequest authnRequest = **new** STORKAuthnRequest();

//Filling Authentication Request fields --------------------------------

authnRequest.setIssuer(..);

authnRequest.setDestination(..);

authnRequest.setProviderName(..);

authnRequest.setQaa(Integer.*parseInt*(..));

authnRequest.setAssertionConsumerServiceURL(*..*);

authnRequest.setSpSector(*..*);

authnRequest.setSpInstitution(*..*);

authnRequest.setSpApplication(..);

authnRequest.setSpCountry(*..*);

authnRequest.setSPID(*..*);

authnRequest.setCitizenCountryCode(..); //Selected Citizen Country

//Loading Stork attributes to request

IPersonalAttributeList attributelist = **new** PersonalAttributeList();

PersonalAttribute attribute1 = **new** PersonalAttribute();

attribute1.setName(*..*); //Attribute name

attribute1.setIsRequired(**true**); //Attribute mandatory

attributelist.add(attribute1);

// The same for more attributes

authnRequest.setPersonalAttributeList(attributelist);

//Getting STORKSAMLEngine object --------------------------------------

STORKSAMLEngine engine = STORKSAMLEngine.*getInstance*(“SP”);

authnRequest = engine.generateSTORKAuthnRequest(authnRequest);

**byte**[] token = authnRequest.getTokenSaml();

SAMLRequest = PEPSUtil.*encodeSAMLToken*(token);

Then, this SAMLRequest must be placed into a HTML form:

<form name="redirectForm" method="post" action="stork.country.server.URL”>

<input hidden name="SAMLRequest" value="above-SAMLRequest”/>

<input name="country" value="selectedCitizenCountry"/>

</form>

## Validating and Reading STORK Authentication Responses

First, we receive a formulary with the field “SAMLResponse”.

Then:

//Decodes incoming SAML Response

byte[] decSamlToken = PEPSUtil.decodeSAMLToken(SAMLResponse);

//Get SAMLEngine instance

STORKSAMLEngine engine = STORKSAMLEngine.getInstance(“SP”);

//Validating SAML Authentication response

STORKAuthnResponse authnResponse = null;

authnResponse = engine.validateSTORKAuthnResponse(decSamlToken, (String)request.getRemoteHost());

//Reading Authentication Response

if(!authnResponse.isFail()){

//Get attributes

IPersonalAttributeList attributeList=authnResponse.getPersonalAttributeList();

setAttrList(new ArrayList<PersonalAttribute>( attributeList.values()));

Iterator<String> list = attrList.iterator();

while ( attrList.hasNext() ){

PersonalAttribute attribute=attrList.next();

System.out.println( "Attribute: "+ attribute.name);

System.out.println( "Value: "+ attribute.value.get(0));

System.out.println( "Status: "+ attribute.status);

}

}else{

System.out.println ("Saml Response is fail:" + authnResponse.getMessage());

}

# Set of Test Certificates

A set of certificates is distributed with the DemoSP application package.

Its purpose is to serve for testing the authentication of Citizens with correspondent STORK Country Server.

|  |
| --- |
| **Certificate (Quality)** |
| -----BEGIN CERTIFICATE-----  MIIFjTCCBHWgAwIBAgIEQm3h+zANBgkqhkiG9w0BAQUFADA+MQswCQYDVQQGEwJw  dDEVMBMGA1UEChMMTVVMVElDRVJULUNBMRgwFgYDVQQDEw9NVUxUSUNFUlQtQ0Eg  MDIwHhcNMTEwNzIxMTU0NjMxWhcNMTIwODE3MTU0MTE3WjCBwTELMAkGA1UEBhMC  UFQxFTATBgNVBAoTDE1VTFRJQ0VSVC1DQTEWMBQGA1UECxMNQ0VSVElQT1IgLSBS  QTESMBAGA1UECxMJQ29ycG9yYXRlMTMwMQYDVQQLEypBZ2VuY2lhIHBhcmEgYSBN  b2Rlcm5pemFjYW8gQWRtaW5pc3RyYXRpdmExGDAWBgNVBAsTD1dlYiBBcHBsaWNh  dGlvbjEgMB4GA1UEAxMXUEVQUyBQb3J0dWd1ZXMgZGUgVGVzdGUwgZ8wDQYJKoZI  hvcNAQEBBQADgY0AMIGJAoGBALo91gnq+SQj8yPx8ssFEKuPvAfagO8f+EagEs+u  XJhLx41GpFZesMuolxf86n3TdxJHcLSXI224HqZu3BtXExUiD1LCAvtGCjzOr6Rg  oySwhIQrgMEsKRRpkQN0jQHIMze11EXqVAJ2+MDX9V4cABuIEd9LOOl0PcQmc7m8  jcKXAgMBAAGjggKRMIICjTALBgNVHQ8EBAMCA/gwOAYIKwYBBQUHAQEELDAqMCgG  CCsGAQUFBzABhhxodHRwOi8vb2NzcC5tdWx0aWNlcnQuY29tL2NhMIHgBgNVHSAE  gdgwgdUwTQYJKwYBBAGwPAoCMEAwPgYIKwYBBQUHAgEWMmh0dHA6Ly93d3cubXVs  dGljZXJ0LmNvbS9jcHMvbXVsdGljZXJ0LWNhLWNwcy5odG1sMIGDBgsrBgEEAbA8  CgKIBjB0MHIGCCsGAQUFBwICMGYeZABoAHQAdABwADoALwAvAHcAdwB3AC4AbQB1  AGwAdABpAGMAZQByAHQALgBjAG8AbQAvAGMAcAAvAG0AdQBsAHQAaQBjAGUAcgB0  AC0AYwBhAC0AMQAwADMAMAAuAGgAdABtAGwwEQYJYIZIAYb4QgEBBAQDAgSwMIIB  AQYDVR0fBIH5MIH2MIGaoIGXoIGUhi9odHRwOi8vd3d3Lm11bHRpY2VydC5jb20v  Y2EvbXVsdGljZXJ0LWNhLTAyLmNybIZhbGRhcDovL2xkYXAubXVsdGljZXJ0LmNv  bS9jbj1NVUxUSUNFUlQtQ0ElMjAwMixvPU1VTFRJQ0VSVC1DQSxjPVBUP2NlcnRp  ZmljYXRlUmV2b2NhdGlvbkxpc3Q/YmFzZTBXoFWgU6RRME8xCzAJBgNVBAYTAnB0  MRUwEwYDVQQKEwxNVUxUSUNFUlQtQ0ExGDAWBgNVBAMTD01VTFRJQ0VSVC1DQSAw  MjEPMA0GA1UEAxMGQ1JMMjI4MB8GA1UdIwQYMBaAFB3DuYilGL5gpyymY8pmKvwM  J8G9MB0GA1UdDgQWBBRH/+uES4Jsr1UV5WeSoN3v1vUaPDAJBgNVHRMEAjAAMA0G  CSqGSIb3DQEBBQUAA4IBAQAOFAxM2U6HyZgWl5h6UB1MUUL4j9VTQQOs6nw4hm22  QK3SF+DPL6oXS1j+RKDHYNlpAfQ5r5ObcaxhEkaXOUZJ4q/3z1qScMVaZ1fjU0FB  hRyAUE2qfiHp/0Ql4V2IrQqcBZ+mEQD5DFwNgx/UDr22lO0idjHnmxRed83/Mrm0  03v+2eAujlsE9NfayP8oo9HkYNh5KvFjveCpUNv4IW18xEJLNDFd3dUEeb9UO+Bv  eEkrxmo6k/k7usuRUfGrXBaFuxcL71l3lFD4k66CB3m7atcbohmbiAYhfHnLegpR  EVKVolR6O3ljt3ou+Y79oI4U7bhn0U256R9hoobnX9Un  -----END CERTIFICATE----- |

|  |
| --- |
| **Certificate (Production)** |
| -----BEGIN CERTIFICATE-----  MIIFojCCBIqgAwIBAgIEQmx+HTANBgkqhkiG9w0BAQUFADA+MQswCQYDVQQGEwJw  dDEVMBMGA1UEChMMTVVMVElDRVJULUNBMRgwFgYDVQQDEw9NVUxUSUNFUlQtQ0Eg  MDIwHhcNMTAwNTE3MTAyMjMyWhcNMTMwNTE3MTAyMzM4WjCB1jELMAkGA1UEBhMC  UFQxFTATBgNVBAoTDE1VTFRJQ0VSVC1DQTEWMBQGA1UECxMNQ0VSVElQT1IgLSBS  QTESMBAGA1UECxMJQ29ycG9yYXRlMT4wPAYDVQQLEzVBTUEgLSBBZ2VuY2lhIHBh  cmEgYSBNb2Rlcm5pemFjYW8gQWRtaW5pc3RyYXRpdmEgSS5QLjEYMBYGA1UECxMP  V2ViIEFwcGxpY2F0aW9uMSowKAYDVQQDEyFzYW1sLmV1LWlkLmNhcnRhb2RlY2lk  YWRhby5nb3YucHQwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMe2B9O1xCJp  CmT2/AuypD1q9kbwge1Y0VjY5FOkhYPfki/XuuFpEdUa7KrurbcoDuuAmgjIxCIn  v8vYAK5axY8hlPg9fp+vtRlmo1it5Y9IGY2mMvtN6OwoBzJOqKJypNexyAgIIR/u  PqhVZjQAwGkTe1JrcDswKOKGbv21M1+pAgMBAAGjggKRMIICjTALBgNVHQ8EBAMC  A/gwOAYIKwYBBQUHAQEELDAqMCgGCCsGAQUFBzABgRxodHRwOi8vb2NzcC5tdWx0  aWNlcnQuY29tL2NhMIHgBgNVHSAEgdgwgdUwTQYJKwYBBAGwPAoCMEAwPgYIKwYB  BQUHAgEWMmh0dHA6Ly93d3cubXVsdGljZXJ0LmNvbS9jcHMvbXVsdGljZXJ0LWNh  LWNwcy5odG1sMIGDBgsrBgEEAbA8CgKIBjB0MHIGCCsGAQUFBwICMGYeZABoAHQA  dABwADoALwAvAHcAdwB3AC4AbQB1AGwAdABpAGMAZQByAHQALgBjAG8AbQAvAGMA  cAAvAG0AdQBsAHQAaQBjAGUAcgB0AC0AYwBhAC0AMQAwADMAMAAuAGgAdABtAGww  EQYJYIZIAYb4QgEBBAQDAgSwMIIBAQYDVR0fBIH5MIH2MIGaoIGXoIGUhi9odHRw  Oi8vd3d3Lm11bHRpY2VydC5jb20vY2EvbXVsdGljZXJ0LWNhLTAyLmNybIZhbGRh  cDovL2xkYXAubXVsdGljZXJ0LmNvbS9jbj1NVUxUSUNFUlQtQ0ElMjAwMixvPU1V  TFRJQ0VSVC1DQSxjPVBUP2NlcnRpZmljYXRlUmV2b2NhdGlvbkxpc3Q/YmFzZTBX  oFWgU6RRME8xCzAJBgNVBAYTAnB0MRUwEwYDVQQKEwxNVUxUSUNFUlQtQ0ExGDAW  BgNVBAMTD01VTFRJQ0VSVC1DQSAwMjEPMA0GA1UEAxMGQ1JMMTczMB8GA1UdIwQY  MBaAFB3DuYilGL5gpyymY8pmKvwMJ8G9MB0GA1UdDgQWBBT+DvK0cR8Qa3uUWWYV  rUfVGZeUTDAJBgNVHRMEAjAAMA0GCSqGSIb3DQEBBQUAA4IBAQBiXYB/Nst7hDnV  RS9D6VjifN1F+JaxtwSLZBoxkij2mi/1kXRugKjkpo6e8Kwb24Wv7G+/ZAFjm3zN  WK9v0ziR192l+4lWke8wRVwHW4Ecsp3nOwOxCiCYkX4uVPDZQT5+cPeNYJbOwYyd  4jbHTPrPT7T2CmtgdqOIu2Dc+1aHyg9ZnhCGgwEwDbvq+grUr9RcHqmWqfdR3Eou  TvLugaM54N4Bur8rolFatHzETbKjvXfWzpHoTTFEekyHgQXWdnmVny8JajBFUmE5  TkONB+V+Jj/R2YPfF++9tRKwc4ifNeduWzSD6ohx+OFimdx2gKHIdkkAMfK09z1M  vz83eaDr  -----END CERTIFICATE----- |