WEB SERVICES
CERTIFICATE GUIDE

1. Purpose

The purpose of this document is to provide information to internal and external users who want to access an eRA Web Service using the certificate based authentication and authorization technique. Internal users are federal staff in NIH and participating non-NIH federal agencies. External users are grant and contract organization (university or organization) staff that want to receive or are already receiving funding from participating funding agency. The information is designed to assist internal and external users in acquiring, converting, testing and registering the required certificates and keys.

The System-to-System (S2S) interface provides eRA Web Services as a means for external systems to interact with eRA systems (such as iEdison) to perform various business functions. eRA uses the Secure Sockets Layer (SSL) protocol to establish a highly secure connection between the client application and the eRA Web Service. Appendix A provides an overview of the technical process that establishes the secure connection.

.

> **NOTE**: eRA recommends that the user work closely with their IT department in implementing the S2S Web Services using the certificate technique. There are technical challenges with this process that often require the support of staff with the required skills and experience. In addition, many internal and external organizations have IT policies and processes related to procuring and using certificates that impact how the user accomplishes establishing a S2S interface with an eRA system.

The URL to access a specific eRA Web Service can be retrieved from the technical user guide of the service. The technical guides are found at http://inside.era.nih.gov/Technical/oltp.cfm#1. All eRA Web Services are offered as standards-based web services and published via WSDL or WADL, allowing client stubs to be auto-generated.

2. Process Overview

The following steps are required for a client application to be configured to access eRA Web Services and use a SSL connection:

1. Procure a certificate from a supported certificate provider (Section 2.1)
2. Convert the certificate (Section 2.2)
3. View the certificate (Section 2.3)
4. Register the certificate (Section 2.4)
5. Configure the calling program (Section 2.5)

> **NOTE**: eRA recommends that the user install a utility that will facilitate acquiring, viewing, converting, managing, and troubleshooting certificates. A number of these utilities are available from the certificate providers and other sources. eRA uses the OpenSSL utility to support its internal certificate management operations. As an example of the functionality of these utilities, Appendix B provides an overview of OpenSSL and describes the most common OpenSSL functions and commands.

2.1 Procure a Certificate from a Supported Certificate Provider

The user intending to access eRA Web Services is responsible for contacting a certificate provider and procuring a certificate, or reusing an existing certificate it may already have. The user must keep track of the certificate expiration date and other key information (see step 2.3 for the types of information that the user needs to record for each certificate).

Considerations for procuring a certificate:

- eRA supports certificates used to secure client-to-server information.
- eRA does **NOT support wildcard or self-signed certificates**.
- eRA does **NOT support any certificates issued by HHS.**
- As part of its security policy, eRA requires having a separate certificate for production and non-production environments**. Consequently, users must procure and install a certificate for each production environment and a separate certificate for each non-production environment.**

Currently, eRA supports certificates from the following certificate providers:

- Comodo
- DigiCert
- Entrust
- GoDaddy
- VeriSign
- Thawte
- GeoTrust

> **NOTE**: A complete list of the certificates that eRA accepts is found in Appendix C. The user should review this list to identify the certificate provider and certificate that it wants to acquire and use.

In general, certificate procurement involves the 5 activities outlined below. Details vary by certificate provider consequently the user should use the *how to procure a certificate* information that is normally provided by the selected certificate provider.

2.1.1 Generate Certificate Signing Request (CSR). The CSR must be generated from the application that will call the eRA Web Service. The private key file (.key) is generated at this time and is stored on the client system.

2.1.2 Submit the completed CSR to the certificate provider. The CSR is uploaded into the certificate provider's system. There is a wait time from when the CSR is submitted to when the certificate is made available. The wait time varies by certificate provider.

2.1.3 Pay for the certificate. Payment for certificates is required. Payment arrangements (e.g., use of PO or credit card) and due dates (e.g., when the CSR is submitted, when it is made available for download) vary by certificate provider.

2.1.4 Certificate provider approves CSR and generates certificate. The certificate is uploaded to the certificate provider's web site when it has been approved. The user monitors the certificate provider's website for the completed and approved certificate.

2.1.5 Download the certificate. The user logs into the certificate provider's website, locates the provided certificate and downloads it to the client application where the CSR was generated.

2.2 Convert the Certificate.

The certificate that is delivered by the certificate provider is not necessarily in a format that can be used by the client system to call the eRA web service. Currently, eRA has tested the following two formats **only** and confirmed that they work in the eRA environment:

- **PKCS#12 (.p12) for certificates that are intended to be used with a web browser**
- **Java Key Store (.jks) for certificates that are intended to be used for testing (e.g., functionality testing with SOAP UI**).

> **NOTE**: The different formats used by the certificate providers for their certificates are described in Appendix D. Users should consult with this list to identify the format for their certificate and determine if conversion is required.

The PEM format is the most common format used by certificate providers to issue a certificate. The following outlines the steps that are necessary to convert a PEM format to PKCS#12 and then to Java Key Store.

2.2.1 Ensure that the user is in the certificate directory where the user wants to convert the .pem file
2.2.2 Combine the certificate and key files into one .pem file
2.2.3 Convert the .pem file to a .p12 file
2.2.4 Convert the .p12 file to a .jks file (if necessary). If the .jks file is needed, the user must use the Java Runtime Environment (JRE) key tool utility to convert the .p12 file.

Users are reminded that they must convert all certificates and keys for production and non-production environments. Certificate and key conversion needs to be accomplished only once. It is to be noted that certificate conversions do fail; it is recommended that the user simply re-try the conversion. Appendix B provides the Open SSL commands for converting certificates and keys to various formats.

2.3 View the Certificate.

The user should *view* the certificate and key in order to identify and collect key information on the certificate and key for registration, control and troubleshooting purposes. The user has two main ways of accomplishing this: using the browser in which the certificate and key have been installed or using a SSL utility like OpenSSL.

2.3.1 Using the browser. The following describes the steps for viewing using IE.

a. Go to the **Tools** menu
b. Select **Internet Options**
c. Select **Certificates**
d. Select (highlight) the certificate that you want to view from the drop down list
e. Select **View**
f. The **General, Details and Certification Path tabs** for the selected certificate are displayed
g. Scroll through the tabs to identify and record key information such as:
    a. Issued to
    b. CA who issued the certificate
    c. Validity period

d. Serial number
e. Issuer details

2.3.2 Using the Open SSL utility. The "view" commands vary by utility. Appendix B contains an OpenSSL "view" prompt as an example of this option. This results in the following information being displayed for the selected certificate:

a. Version
b. Serial Number
c. Issuer
d. Validity Period
e. Public Key
f. X09 extensions
g. Signature
h. Certificate

The user should record the key information – in particular version, serial number, issuer, and validity period.

2.4 Register and Validate the Certificate

The certificate must be registered with eRA in order for the user to access eRA Web Services. The registration process is used to grant the users the required roles within the eRA system. The certificate is also validated (i.e.; can operate successfully with the eRA system) as part of the registration process. The certificate registration and validation process is accomplished using the eRA's Account Management System (AMS). Certificate registration and validation is accomplished via the AMS's "Create System Account" or "Manage System Account" modules. Individuals who have the required roles to register and validate certificates must know and follow the AMS "Create System Account" and "Manage System Account" processes and tasks in order to successfully register and validate a certificate.

NOTE: At this time, iEdison Agency/ERL and State Department users cannot create system accounts.

Detailed information on how to use AMS is found in the AMS on line help at https://era.nih.gov/erahelp/AMS_NEW/ .

2.4.1 **Internal users**.

2.4.1.1 The internal user should contact their IMPAC II IC Coordinator and work with the IC Coordinator to register the certificate. The IC Coordinator is the only one who is authorized to use AMS and complete certificate registration and validation. The IC Coordinator accesses AMS (https://apps.era.nih.gov/ams/), performs a search and clicks the appropriate **Manage** button in the hit list or clicks **Create New Account** button. Detailed information on how to use AMS is found in the AMS on line help at https://era.nih.gov/erahelp/AMS_NEW/ .

2.4.1.2 In addition, **and only if the internal user wants to access a non-production Web Service**, the internal user works with their IC Coordinator to submit a *Request for Web Services Account* to the eRA Service Desk (SD) at Helpdesk@od.nih.gov. It is to be noted that the user will be limited to certain role(s) as outlined in the technical user guide for each web service found at http://inside.era.nih.gov/Technical/oltp.cfm#1. The request must be approved and submitted by the IC

Coordinator. The eRA SD will process this request as follows:

i. Review the request form and work with the user to resolve any questions or issues
ii. Approve or disapprove the request. If it is disapproved, the internal user will be informed of the reasons why the request was disapproved.
iii. If approved, the eRA Service Desk will create the Web Service accounts and inform the internal user and IC Coordinator.

2.4.1.3 Furthermore, **and only if the internal user is in OD and wants to access a production Web Service,** the internal user works with a Federal Sponsor to submit a *Request for Web Services Account* as outlined in 2.4.1.2. In this case, the request must be approved and submitted by the Federal Sponsor. The eRA Service Desk uses the same process as outlined in 2.4.1.2 to process the request.

2.4.2 **iEdison users.** Only staff that have the **EXTRAMURAL_TTO_ADMIN** role in iEdison have the necessary privileges. Consequently iEdison users must work with the staff that have this role to register and validate a certificate.

i. Login to iEdison - https://public.era.nih.gov/iedison
ii. Click on **Main Menu**
iii. Scroll to **Account Administration**
iv. Select one of the following:
    1. **Create an iEdison Account**
    2. **Search for an iEdison Account to Modify**
v. Follow the instructions for registering and validating a certificate. These are found on the online AMS help at https://era.nih.gov/erahelp/AMS_NEW/

2.4.3 **Commons users.** Only staff that have the **SO** role in Commons have the necessary privileges. Consequently Commons users must work with the staff that have this role to register and validate a certificate.

i. Login to eRA Commons - https://public.era.nih.gov/commons
ii. Click on **Admin**
iii. Click on **Accounts**
iv. Click on **Account Management**
v. Follow the instructions for registering and validating a certificate. These are found on the online AMS help at https://era.nih.gov/erahelp/AMS_NEW/

2.5 Configure the Calling Program

The user needs to install the private key, properly configure the calling program, verify that the roles have been established through the registration process, and verify that the calling program can call the appropriate eRA Web Service. The following provides additional information on these steps.

2.5.1 The private key will need to be properly installed on the local machine that will be calling the eRA Web Service. How this is done will vary by platform, but note that for Windows-based machines, the key is only available for user accounts in the Administrators group and for the user who installed the client certificate. Therefore, access must be granted to the certificate and key for the user account that is used to run the calling program.

2.5.2 The calling program will need to be properly configured. Details regarding this will vary based on the technology used in the calling program. The proper configuration of the calling program is the

responsibility of the user. It is strongly recommended that the user work with their IT organization to properly accomplish this. The eRA SD cannot provide support to users in configuring the calling program.

2.5.3 Verify the registration and access to the eRA Web Service. The client system calls the selected eRA Web Service. If the call is successful, then the roles have been established and the calling program is properly configured. If the call is not successful, the user should work with their IT department to troubleshoot any problems.

3 Troubleshooting Service Issues

The following table describes certain error messages that the user may see the possible cause, and what action the user should take to resolve the issue.

Type of Issue	Error Message	Possible Cause/Solution
Connectivity	UnknownHostException	The full URL for the Web Service is incorrect. (Verify the URL that is being called.)
	SSLPeerUnverifiedException	The certificate has not been obtained or configured properly in the calling program.
Mapping	SOAP response containing a HTTP 500: "Mapping doesn't exist for certificate"	Verify that the serial number and authority in AMS match the certificate on the caller's server.
Authorization	SOAP response containing a HTTP 500: "User <XYZ> is not authorized to access this operation"	Verify that the roles in AMS have been granted for this certificate.
Other Exceptions	SOAP response containing a system exception	For help resolving this issue, please contact the eRA Servicedesk with the date/time of the call, serial number, certificate authority, and the name of the Web Service.

APPENDIX A – Secure Sockets Layer Technical Background

A SSL connection is set up by a handshake. The handshake consists of 3 main phases - Hello, Certificate Exchange and Key Exchange.

1**. Hello** - The handshake begins with the client sending a ClientHello message. This contains all the information the server needs in order to connect to the client via SSL, The server responds with a ServerHello, which contains similar information required by the client.

2. **Certificate Exchange** - Now that contact has been established, the server has to prove its identity to the client. This is achieved using its SSL certificate. A SSL certificate contains various pieces of data, including the name of the owner, the property (e.g. domain) it is attached to, the certificate's public key, the digital signature and information about the certificate's validity dates. The client checks that it is verified and trusted by one of several Certificate Authorities (CAs) that it also implicitly trusts.

3. **Key Exchange** - The encryption of the actual message data exchanged by the client and server will be done using a symmetric algorithm, agreed during the Hello phase. A symmetric algorithm uses a single key for both encryption and decryption, in contrast to asymmetric algorithms that require a public/private key pair. Both parties need to agree on this single, symmetric key, a process that is accomplished securely using asymmetric encryption and the server's public/private keys.

The client generates a random key and encrypts it using an algorithm also agreed upon during the Hello phase, and the server's public key (found on its SSL certificate). It sends this encrypted key to the server, where it is decrypted using the server's private key, and the interesting parts of the handshake are complete. HTTP requests and responses can now be sent by forming a plaintext message and then encrypting and sending it. The other party is the only one who knows how to decrypt this message.

APPENDIX B – OpenSSL

OpenSSL is an open-source implementation of the SSL and TLS protocols. The core library, written in the C programming language, implements basic cryptographic functions and provides various utility functions. Wrappers allowing the use of the OpenSSL library in a variety of computer languages are available.

OpenSSL Command Types	Specific User Action	OpenSSL Command
General - The user needs to generate CSRs, Certificates, and Private Keys and do other miscellaneous tasks.	Generate a new private key and Certificate Signing Request (CSR)	openssl req -out CSR.csr -new -newkey rsa:2048 -nodes -keyout privateKey.key
	Generate a CSR for an existing private key	openssl req -out CSR.csr -key privateKey.key -new
	Generate a CSR based on an existing certificate	openssl x509 -x509toreq -in certificate.crt -out CSR.csr -signkey privateKey.key
	Remove a passphrase from a private key	openssl rsa -in privateKey.pem -out newPrivateKey.pem
Checking -The user needs to check the information within a Certificate, CSR or Private Key.	Check a CSR	openssl req -text -noout -verify -in CSR.csr
	Check a private key	openssl rsa -in privateKey.key -check
	Check a certificate	openssl x509 -in certificate.crt -text -noout
	Check a PKCS#12 file (.pfx or .p12)	openssl pkcs12 -info -in keyStore.p12
Debugging - If the user is receiving an error that the private key doesn't match the certificate or that a certificate has been installed to a site that is not trusted	Check an MD5 hash of the public key to ensure that it matches with what is in a CSR or private key	openssl x509 -noout -modulus -in certificate.crt | openssl md5 openssl rsa -noout -modulus -in privateKey.key | openssl md5 openssl req -noout -modulus -in CSR.csr | openssl md5
	Check an SSL connection. All the certificates (including Intermediates) should be displayed	openssl s_client -connect www.paypal.com:443
Converting Certificate File Format – The user needs to convert certificates and keys to different formats to make them compatible with specific types of servers or software.	Convert a DER file (.crt .cer .der) to PEM	openssl x509 -inform der -in certificate.cer -out certificate.pem
	Convert a PEM file to	openssl x509 -outform der -in certificate.pem -out certificate.der

OpenSSL Command Types	Specific User Action	OpenSSL Command
	DER	
	Convert a PKCS#12 file (.pfx .p12) containing a private key and certificates to PEM	openssl pkcs12 -in keyStore.pfx -out keyStore.pem -nodes
	Convert a PEM certificate file and a private key to PKCS#12 (.pfx .p12)	openssl pkcs12 -export -out certificate.pfx -inkey privateKey.key -in certificate.crt -certfile CACert.crt
Viewing – the user needs to view certificate information	View the selected certificate	X509 –text –in <cert name>

APPENDIX C – CERTIFICATES CURRENTLY SUPPORTED BY eRA

Intermediate Certificate Name	Expiration Date
AddTrust External CA Root, AddTrust AB	May 30, 2020
Betrusted Production SSL CA A1, Betrusted US Inc	Dec 9, 2020
COMODO High-Assurance Secure Server CA, COMODO CA Limited	May 30, 2020
Cybertrust Public Issuing CA 1, Cybertrust Inc	Jul 11, 2017
DigiCert High Assurance CA-3, DigiCert Inc	Apr 2, 2022
DigiCert High Assurance CA-3, DigiCert Inc	Apr 3, 2022
DigiCert High Assurance EV Root CA, Digital Inc	Nov 9, 2031
DigiCert Secure Server CA, DigiCert Inc	Mar 8, 2023
DigiCert SHA2 Assured ID CA, DigiCert Inc	Nov 5, 2028
DigiCert SHA2 High Assurance Server CA, DigiCert Inc	Oct 22, 2028
DigiCert SHA2 Secure Server CA, DigiCert Inc	Mar 8, 2023
DST ACES Device CA A4, Digital Signature Trust	Nov 20, 2017
Entrust	Dec 15th, 2020
Entrust Certification Authority - L1C, Entrust, Inc	Dec 10, 2019
Entrust Certification Authority - L1G, Entrust, Inc	Nov 21, 2023
Entrust Certification Authority - L1K, Entrust, Inc	Oct 23, 2024
Entrust.net Certification Authority (2048), Entrust.net	Mar 23, 2019
Entrust.net Secure Server Certification Authority, Entrust.net	May 25th, 2019
Equifax	Aug 22, 2018
Federal Common Policy CA, U.S. Government	Dec 1, 2030
GeoTrust DV SSL CA, GeoTrust Inc.	Feb 25, 2020

Intermediate Certificate Name	Expiration Date
GeoTrust Global CA, GeoTrust Inc	Aug 21, 2018
GeoTrust Global CA, GeoTrust Inc.	May 21, 2022
Go Daddy Secure Certificate Authority - G2	May 3, 2031
Go Daddy Secure Certification Authority, GoDaddy.com, Inc	Nov 15th, 2026
Go Daddy Secure Certification Authority, GoDaddy.com, Inc	Nov 15th, 2026
GTE CyberTrust Global Root, GTE Corporation	Aug 13, 2018
http://www.valicert.com/, ValiCert, Inc	Jun 25th, 2019
InCommon RSA Server CA	Oct 5, 2024
InCommon Server CA, Internet2	May 30, 2020
Thawte DV SSL CA, Thawte, Inc.	Feb 17, 2020
Thawte Premium Server CA, Thawte Consulting cc	Jan 1, 2021
thawte Primary Root CA, thawte, Inc	Dec 30, 2020
thawte Primary Root CA, thawte, Inc	Jul 16, 2036
Thawte SSL CA, Thawte, Inc	Feb 7, 2020
The Go Daddy Group, Inc	Jun 29, 2024
VeriSign Class 3 Extended Validation SSL CA, Verisign, Inc	Nov 7, 2016
VeriSIgn Class 3 Public Primary Certificatin Authority - G5, VeriSign, Inc	Jul 16, 2036
VeriSign Class 3 Secure Server CA - G2, VeriSign, Inc	Mar 24, 2019
VeriSign Class 3 Secure Server CA - G3, VeriSign, Inc	Feb 7, 2020
VeriSign Trust Network	Oct 24, 2016
VeriSign, Inc	Aug 1, 2028
AddTrust External CA Root, AddTrust AB	May 30, 2020

Intermediate Certificate Name	Expiration Date
Betrusted Production SSL CA A1, Betrusted US Inc	Dec 9, 2020

APPENDIX D – CERTIFICATE FORMATS

Certificate Format	File Extension	Description	Comment
PEM	.pem, .crt, .cer, .key.	They are Base64 encoded ASCII files and contain "-----BEGIN CERTIFICATE-----" and "-----END CERTIFICATE-----" statements. Server certificates, intermediate certificates, and private keys can all be put into the PEM format.	Apache and other similar servers use PEM format certificates. Several PEM certificates, and even the private key, can be included in one file, one below the other, but most platforms, such as Apache, expect the certificates and private key to be in separate files.
DER	.crt .cer .der	DER format is simply a **binary** form of a certificate instead of the ASCII PEM format.	It sometimes has a file extension of **.der** but it often has a file extension of **.cer** so the only way to tell the difference between a DER .cer file and a PEM .cer file is to open it in a text editor and look for the BEGIN/END statements.
PKCS#12	.pfx .p12	The PKCS#12 or PFX format is a binary format for storing the server certificate, any intermediate certificates, and the private key in one encryptable file (password protected).	PFX files usually have extensions such as .pfx and .p12. PFX files are typically used on Windows machines to import and export certificates and private keys. **The .p12 format recognized by most software and applications used at eRA**
PKCS#7/P7B	.p7b .p7c.	The PKCS#7 or P7B format is usually stored in Base64 ASCII format and has a file extension of .p7b or .p7c. P7B certificates contain "-----BEGIN PKCS7-----" and "-----END PKCS7-----" statements	Several platforms support P7B files including Microsoft Windows and Java Tomcat.
JKS	.jks	Java's keystore implementation It has 2 flavors, key store & trust store.	Key store is where private key is maintained and needs to be protected. Trust store is the store for public key / other trusted party. Technically it does not need to be protected.