System-to-System (S2S) Guide for eRA Web Services

Version No: 5.0

11/15/2024

Table of Contents

1. Introduction

The System-to-System (S2S) interface provides eRA Web Services as a means for external systems to interact with eRA systems to perform various business functions.

Information about eRA Web Services can be found at

- Grantor systems: https://inside.era.nih.gov/technical/web-services.htm
- Grantee system: https://era.nih.gov/applicants/system/system-webserv.htm

2. Target Audience

The target audience are internal users (federal staff in NIH and non-NIH agencies) and external users (grant and contract organizations staff) running systems that use eRA S2S Web Services provided for the respective grantor and grantee communities. This information is primarily interest of IT, Operations, and Security staff supporting these systems within their organizations.

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 an S2S interface with an eRA system.

3. Process Overview

All eRA Web Services are offered as standards-based web services and published via WSDL or WADL, allowing client stubs to be auto-generated. The services use the certificate-based authentication (also known as mutual TLS, or mTLS).

Appendix A provides an overview of the technical process that establishes the secure connection.

eRA requires S2S clients to use valid up-to-date certificates accessing eRA Web services, including:

1. Certificate must be issued by an authority recognized and supported by eRA (see Appendix C for the list of authorities accepted by eRA). *eRA doesn't accept self-signed certificates.*
2. All certificates in the chain must be version 3 (version 1 will no longer be accepted).
3. All certificates in the chain must be within their validity period and non-revoked.

eRA requires the calling application to provide full certificate chain. You can see Appendix B for instructions on how to view your certificate chain and how to build full certificate chain into your configuration file.

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

1. Procure a certificate from a supported certificate provider (Section 3.1)
2. Convert the certificate (Section 3.2)
3. View the certificate (Section 3.3)
4. Register the certificate (Section 3.4)
5. Configure the calling program (Section 3.5)

3.1. 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 they may already have. The user must keep track of the certificate

expiration date and other key information (see step 3.3 for the types of information that the user needs to record for each certificate).

eRA requires having separate certificates for production and non-production environments**. Users must procure and install a certificate for each production environment and a separate certificate for each non-production environment.**

eRA accepts certificates issued by the authorities listed in Appendix C.

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.

3.1.1. 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.

3.1.2. 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.

3.1.3. 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.

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

3.1.5. 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.

3.2. 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. eRA internally utilizes certificates in the formats PKCS#12 (.p12) and Java Key Store (.jks), and may provide technical assistance/verification for partners using the same formats.

NOTE: Different formats used by the certificate providers for their certificates are described in the Appendix D. Users should consult with this list to identify the format for their certificate and determine if conversion is required. Appendix B provides useful Open SSL commands for converting certificates and keys to various formats.

3.3. View the Certificate.

The user should *view* the certificate and key 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 an SSL utility like OpenSSL.

3.3.1. Using the browser. The following describes the steps for viewing using Microsoft Edge

a. Click the 3 dots in the top right corner.
b. Go to the "**Settings"** menu.
c. On the left side, select **"Privacy, search, and services"**.
d. Select "**Manage Certificates"** in the Security section**.**
e. Select (highlight) the certificate that you want to view from the list.
f. Select **View**, 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:
   - Issued to
   - CA which issued the certificate
   - Validity period

- Serial number
- Issuer details

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

- Version
- Serial Number
- Issuer
- Validity Period
- Public Key
- X09 extensions
- Signature
- Certificate

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

3.4. Register and Validate the Certificate

The certificate must be registered with eRA 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 are accomplished using the eRA's Account Management Module (AMM), formerly Account Management System (AMS). Certificate registration and validation are accomplished via the AMM'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 AMM "Create System Account" and "Manage System Account" processes and tasks to successfully register and validate a certificate.

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

3.4.1. Internal users

An 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 AMM and complete certificate registration and validation. The IC Coordinator accesses AMM (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 AMM is found in the AMM on line help at https://era.nih.gov/erahelp/AMS_NEW/ .

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 https://inside.era.nih.gov/technical/web-services.htm The request must be approved and submitted by the IC Coordinator. The eRA SD will process this request as follows:

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

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 above. In this case, the

request must be approved and submitted by the Federal Sponsor. The eRA Service Desk uses the same process as outlined above to process the request.

3.4.2. 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.

1. Login to eRA Commons - https://public.era.nih.gov/commons
2. Click on **Admin**
3. Click on **Accounts**
4. Click on **Account Management**
5. Follow the instructions for registering and validating a certificate. These are found on the online AMM help at https://era.nih.gov/erahelp/AMS_NEW/

3.5. 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.

3.5.1. 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.

3.5.2. 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.

3.5.3. 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.

4. 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 AMM 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 AMM have been granted for this certificate.
Other Exceptions	SOAP response containing a system exception	For help resolving this issue, please contact the eRA Service Desk with the date/time of the call, serial number, certificate authority, and full URL including the name of the Web Service.

APPENDIX A – Secure Sockets Layer Technical Background

An 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. OpenSSL binaries for all major Operating Systems are available from standard OS repositories or from the OpenSSL Wiki: https://wiki.openssl.org/index.php/Binaries

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

OpenSSL Command Types	Specific User Action	OpenSSL Command
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 DER	openssl x509 -outform der -in certificate.pem -out certificate.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	openssl x509 –text –in <cert name>

APPENDIX C – Certificate Authorities Currently Supported By eRA

Certificates Authority [Common Name]
COMODO RSA Certification Authority
COMODO RSA Code Signing CA
COMODO RSA Organization Validation Secure Server CA
DigiCert Assured ID CA G2
DigiCert Assured ID Client CA G2
DigiCert Assured ID Root G2
DigiCert Global G2 TLS RSA SHA256 2020 CA1
DigiCert Global Root G2
DigiCert SHA2 Assured ID CA
DigiCert SHA2 Assured ID Code Signing CA
DigiCert SHA2 Extended Validation Server CA
DigiCert SHA2 High Assurance Server CA
DigiCert TLS RSA SHA256 2020 CA1
Entrust Certification Authority L1K
Entrust OV TLS Issuing RSA CA 1
Entrust Root Certification Authority G2
Federal Common Policy CA
Federal Common Policy CA G2
GeoTrust Primary Certification Authority G3
Go Daddy Root Certificate Authority G2
Go Daddy Secure Certificate Authority G2
HHS FPKI Intermediate CA E1
HydrantID Server CA O1
IdenTrust Commercial Root CA 1
InCommon RSA Server CA
InCommon RSA Server CA 2
InCommon RSA Standard Assurance Client CA
RapidSSL Global TLS RSA4096 SHA256 2022 CA1
RapidSSL TLS RSA CA G1
Sectigo RSA Domain Validation Secure Server CA

SSL.com Root Certification Authority RSA
SSL.com RSA SSL subCA
SSL.com TLS RSA Root CA 2022
Thawte EV RSA CA 2018
Thawte RSA CA 2018
Thawte TLS RSA CA G1
USERTrust RSA Certification Authority
VeriSign Universal Root Certification Authority

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.