Software Risk Manager Documentation (v2025.9.4)

Contents

Software Risk Manager Install Guide

Software Risk Manager Install and Deployment Options

The full range of Software Risk Manager functionality is based on how it is deployed. This section provides an overview of what each deployment provides.

Software Risk Manager has four deployment options, which are described below.

Native Installer

The native installer (also referred to as the "stand alone" deployment) is available for both Windows and Linux. It provides an easy way to install the entire technology stack on a single server with an easy-to-use UI and command line wizard. This deployment is recommended for users who want to get up and running quickly and for simple maintenance and upgrades.

With the native installer, both the web application and database are running on the same system, which can cause contention. For high availability, the Kubernetes deployment is recommended. In addition, the native installer does not support running an external database, nor does it support horizontally scaling the running of tools. For more information, see Running the Native Installer.

Note: Tool Orchestration and Scan Farm modules are not supported with this deployment option.

Docker Compose Deployment

This deployment is recommended for users who want to leverage containers while having everything on a single server. (Note that the database can also be deployed on a seperate server). This deployment option allows for using a bundled database or an external one. For more information, see Installation Using Docker Compose.

Note: Tool Orchestration and Scan Farm modules are not supported with this deployment option.

Kubernetes Deployment

This deployment option is recommended for high availability and scalability. Kubernetes also supports Tool Orchestration and Scan Farm modules. For more information, see Installation Using Kubernetes.

Manual Installation

This deployment is reserved for advanced administrators who want total control of the deployment. It allows the user to install the web server and database and manage that infrastructure independently. The database can be deployed on the same machine or a separate one from the web application. This deployment method is not recommended due to its complexity and being prone to error.

Note: Tool Orchestration and Scan Farm modules are not supported with this deployment option.

Installation Requirements for the Native Installer

The Native Installer has the following requirements.

Software Requirements

Software Risk Manager is pre-packaged with most of its requirements. There are, however, certain prerequisites for installations that will be leveraging the .NET scanning support of Software Risk Manager.

Note: The bundled Dependency-Check periodically updates its database of vulnerabilities. If Software Risk Manager is installed in an environment without a connection to the internet, this update will not succeed. For more information, see the Internet Access section.

.NET Analysis

In order to run the bundled .NET tools supported by Software Risk Manager, the .NET Framework runtime is required for Windows and the Mono runtime is required for Linux. Additionally, Dependency-Check requires the .NET Core 8.0 SDK on all platforms.

Note: The .NET Core 8.0 SDK is preferred over the .NET Core 8.0 Runtime, as the Runtime may cause Dependency-Check .NET analyses to fail in some environments.

Windows Users

It is recommended that the latest version of .NET be installed.

Software Risk Manager is capable of running multiple .NET analysis tools on your codebase. FxCop is a supported tool and is developed and distributed by Microsoft. For Software Risk Manager to run FxCop on your behalf, you must install it separately. Software Risk Manager will then automatically discover its location and run it.

Software Risk Manager supports FxCop versions 10+ and will automatically discover the location of the latest version of FxCop installed on your machine. If you would like to provide a specific location, set the `fxcop.path` property in the Software Risk Manager configuration file (`codedx.props`). FxCop is a legacy analyzer that is no longer available through Microsoft. Starting with Visual Studio 2019 and .NET 5.0, FxCop is replaced with Microsoft Code Analysis (Roslyn) Analyzers.

Hardware Requirements

Hardware requirements vary, depending on how many Software Risk Manager projects will be active at the same time, how frequently analyses will be conducted, whether built-in tools are being used, the number of results from tools in use, how many concurrent users are expected to use the system, and what other system interactions might be setup. However, the following is the suggested hardware configuration requirements based on deployment size.

Table 1:

Deployment Size	CPU Cores	Memory	IOPs	Storage
Small This configuration is recommended for supporting up to 100 projects with limited use of built-in tools, up to 1,000 analyses per day, and up to 8 concurrent analyses.	4 cores	16 GB RAM	3,000	250 GB
Medium This configuration is recommended for supporting between 100 and 2,000 projects, up to 2,000 analyses per day, and up to 16 concurrent analyses.	8 cores	32 GB RAM	6,000	500 GB

Deployment Size	CPU Cores	Memory	IOPs	Storage
Large This configuration is recommended for supporting between 2,000 and 10,000 projects, up to 10,000 analyses per day, and up to 32 concurrent analyses.	16 cores	64 GB RAM	12,000	1 TB
Extra Large This configuration is recommended for supporting over 10,000 projects, more than 10,000 analyses per day, and up to 64 concurrent analyses.	32 cores	128 GB RAM	24,000	2 TB

Operating Systems Supported

The following operating systems are supported:

- Windows (10, 11, and Server 2016+)
- Linux (Ubuntu 18+, RHEL/CentOS 7+)

Installation Using the Native Installer

This section describes the Software Risk Manager native installer. For Docker or Kubernetes installation instructions, please see the codedx-docker and srm-k8s pages respectively.

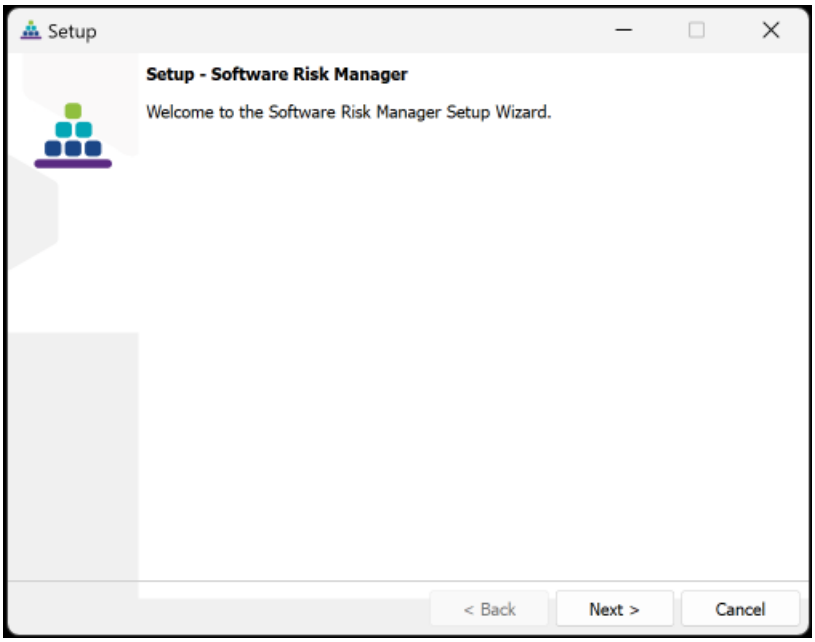
Note: A Kubernetes-based installation is necessary when using Tool Orchestration or Scan Service (needed to run Coverity and Black Duck scans).

The Software Risk Manager installer is self-contained as it includes all of the software necessary to run Software Risk Manager and is dependent upon its own Tomcat, Apache, and MariaDB services. The services automatically start whenever Software Risk Manager is restarted.

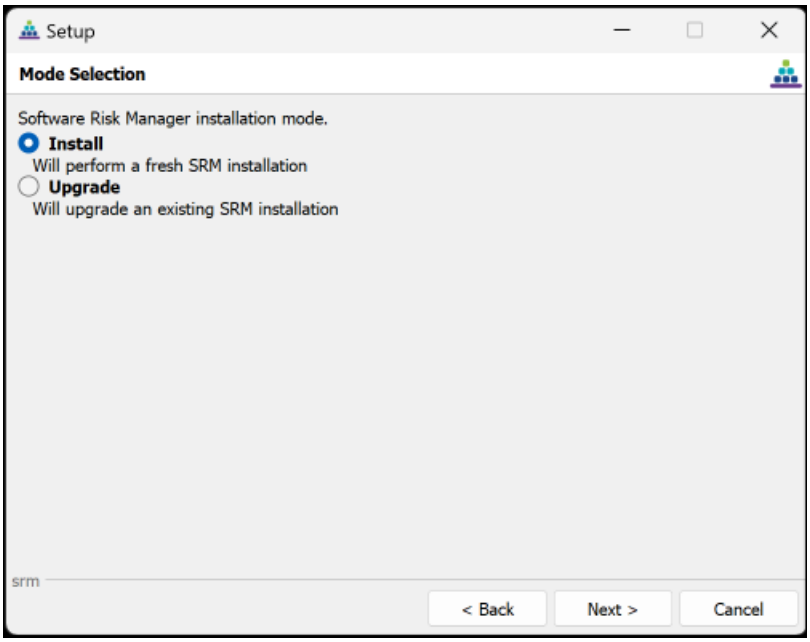
Note: While multiple Software Risk Manager installations are supported, it is discouraged to run them on the same machine. This will likely lead to resource contention, resulting in performance degradation in those installations.

Some of the Software Risk Manager dependencies listen for network activity (e.g., the web server to accept incoming requests). If a port conflict is detected, you will be prompted with a suggested alternate port. It is your choice to accept that port or change it.

When first starting the installer downloaded for your platform, you should see the following screen.



Click Next and select whether this is a fresh install or an upgrade.



Click Next to continue.

Linux Considerations

For headless Linux environments, the installer comes with an option to interact from the command line. To do so, run the installer with the `--mode text` option.

For example:

```
./srm-<version>-linux-x64-installer.run --mode text
```

where you substitute the installer version number for "version."

The installer behaves differently depending on whether it is executed by the root user (or with sudo privileges) or by a standard user. The default directories will change and the port numbers used for the installation will also be different (port 80 for HTTP access for instance when using the root user vs. port 8080 for non-root users). For production installations, we strongly recommend triggering the installer with root/sudo privileges.

By default, Software Risk Manager does not automatically start up on Linux systems. In order to have Software Risk Manager start and stop cleanly when the system boots and shuts down, several commands need to be executed depending on distribution.

Note: Running on start up is only supported for root installations and assumes commands are run with root privileges.

Run on Start Up for Debian-based Distributions

Copy `ctlscript.sh` to the `/etc/init.d` directory and name it `codedx`. `ctlscript.sh` can be found in the Software Risk Manager installation folder. By default this will be in `/opt/srm`.

```
cp installdir/ctlscript.sh /etc/init.d/codedx
```

Then make the script executable.

```
chmod +x /etc/init.d/codedx
```

Add the following to the top of the codedx file below `#!/bin/bash` to ensure that the script is compatible with systemd.

```
### BEGIN INIT INFO
# Provides:          codedx
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start SRM services at boot time
# Description:       Enable services required by SRM at startup.
### END INIT INFO
```

Enable the service to run by default

```
systemctl enable codedx
```

Following a reboot, Software Risk Manager should start automatically. To revert these changes, SRM can be removed from the startup services with the following commands:

```
cd /etc/init.d
systemctl disable codedx
```

Run on Start Up for RedHat-based Distributions

Copy `ctlscript.sh` to the `/etc/init.d` directory and name it `codedx`. `ctlscript.sh` can be found in the Software Risk Manager installation folder. By default this will be in `/opt/srm`.

```
cp installdir/ctlscript.sh /etc/init.d/codedx
```

Replace the `#!/bin/bash` at the top of the file with the following:

```
#!/bin/bash
#
# chkconfig: 2345 80 30
# description: SRM services
```

Then install the script as a service.

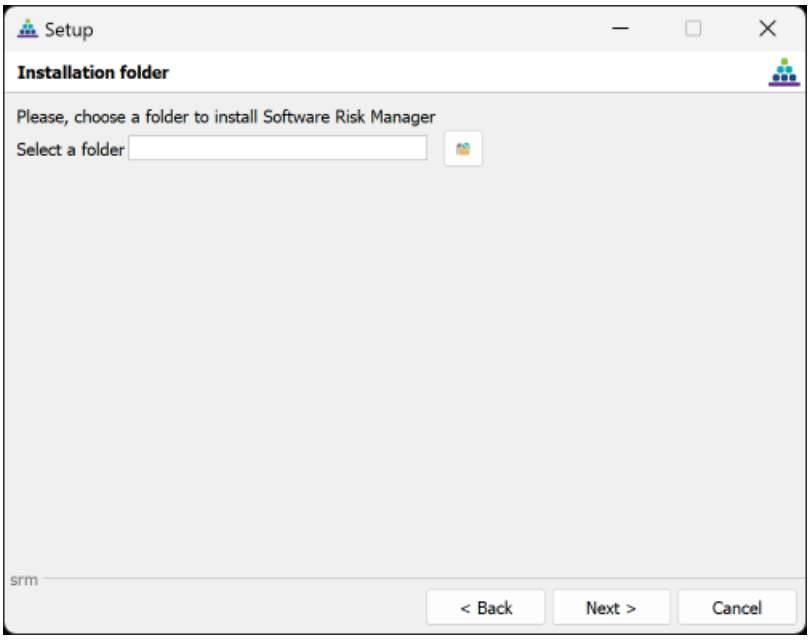
```
chkconfig --add codedx
```

After a reboot, Software Risk Manager will start up automatically. To revert these changes, Software Risk Manager may be removed from the startup services with the following command:

```
chkconfig --del codedx
```

Installation Directory

The Software Risk Manager installer places all its dependencies and program executables in the installation directory. The installer will suggest a location on your file system that sticks with common conventions for your adopted platform; however, you can choose to install Software Risk Manager to a different location if needed.

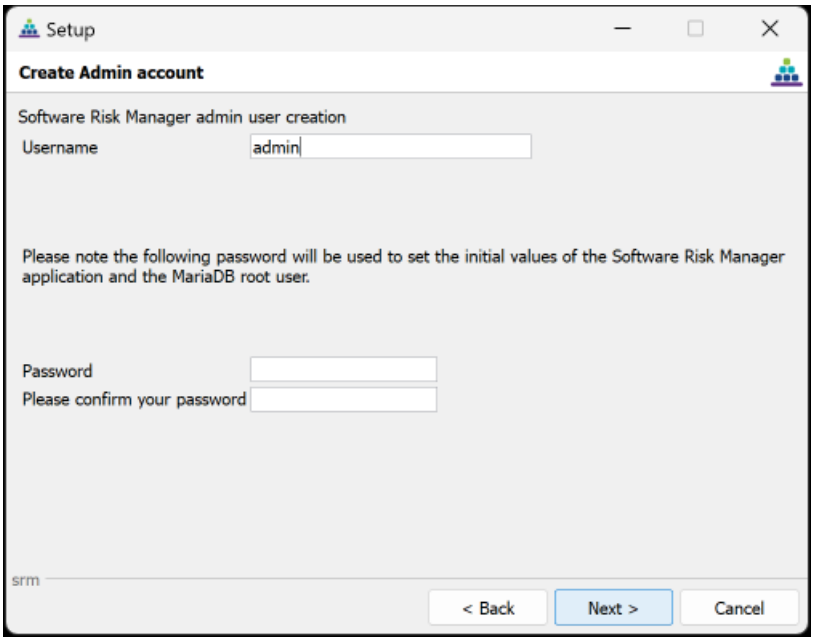


Click Next to continue.

Admin Account

Software Risk Manager has a built-in admin user to manage the system. The installer prompts you for the credentials you want to use for that account. Be sure to keep note of these credentials in a secure location as they are not retrievable once the admin user has been created. **A strong password is strongly recommended including 9 or more characters with combination of lowercase, uppercase, numbers and special characters.** When new Software Risk Manager releases are available, you will download and run the latest installer. Your installer admin credentials are required to complete the installation.

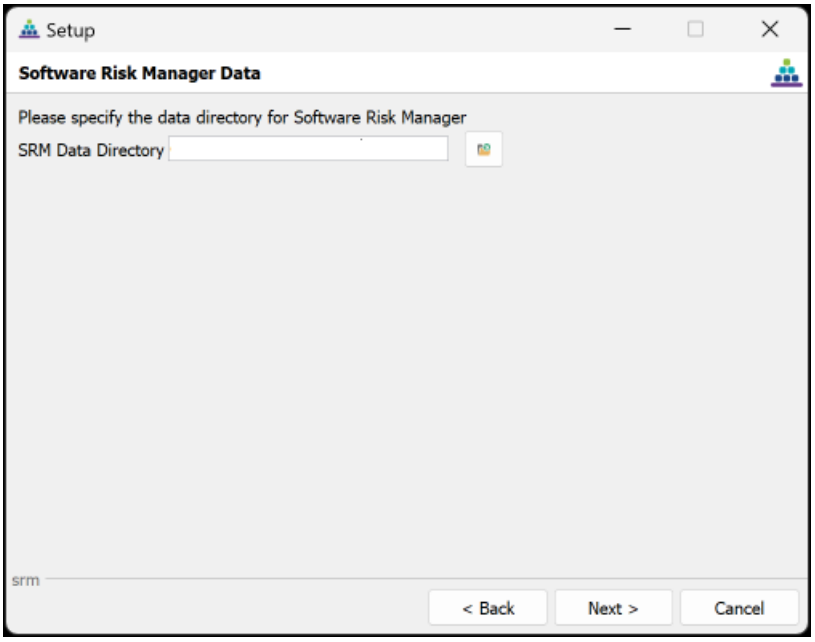
Note: The first time you use Software Risk Manager, you must log in using the installer admin credentials. After that, you can change the admin's password within Software Risk Manager; however, changing the password in Software Risk Manager does **not** change the admin's password for the installer.



Click Next to continue.

Data Directory

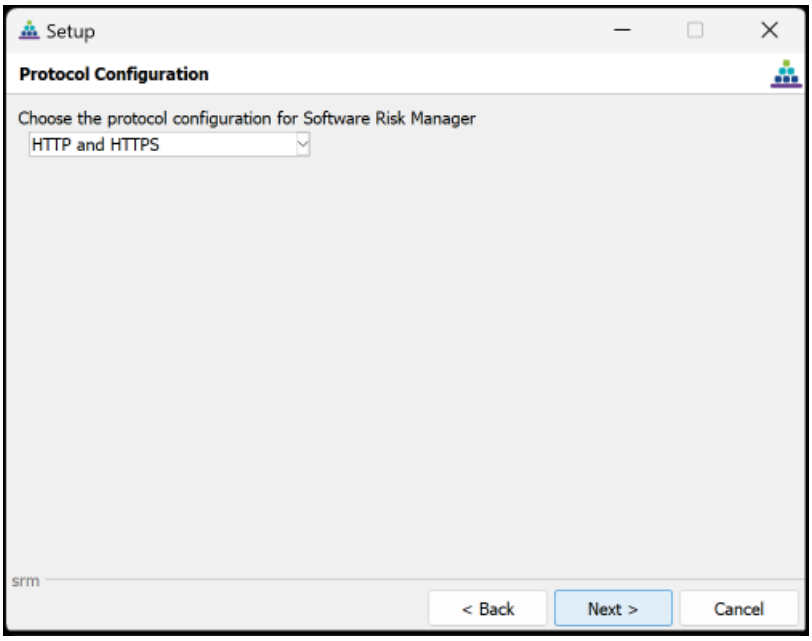
Software Risk Manager stores data in the data directory you configure during the installation. The default location is the typical place used by your adopted platform to place program data files. This can be changed, if needed, from the relevant installer screen. For production installations of Software Risk Manager, we strongly recommend that this data directory be part of your backup plan as it contains all data managed by Software Risk Manager.



Click Next to continue.

Protocol Configuration

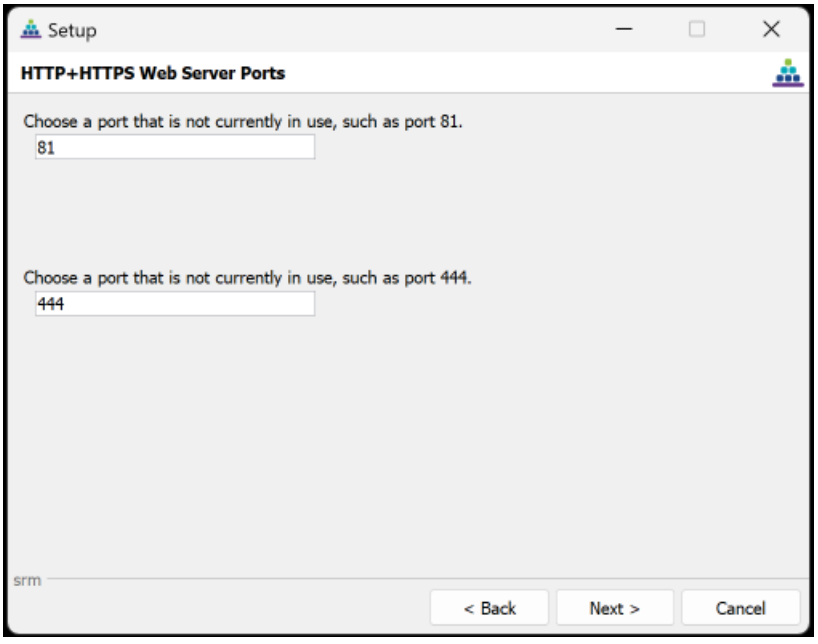
Choose the protocol configuration for Software Risk Manager. The default is HTTP and HTTPS.



Click Next to continue.

Web Server Ports

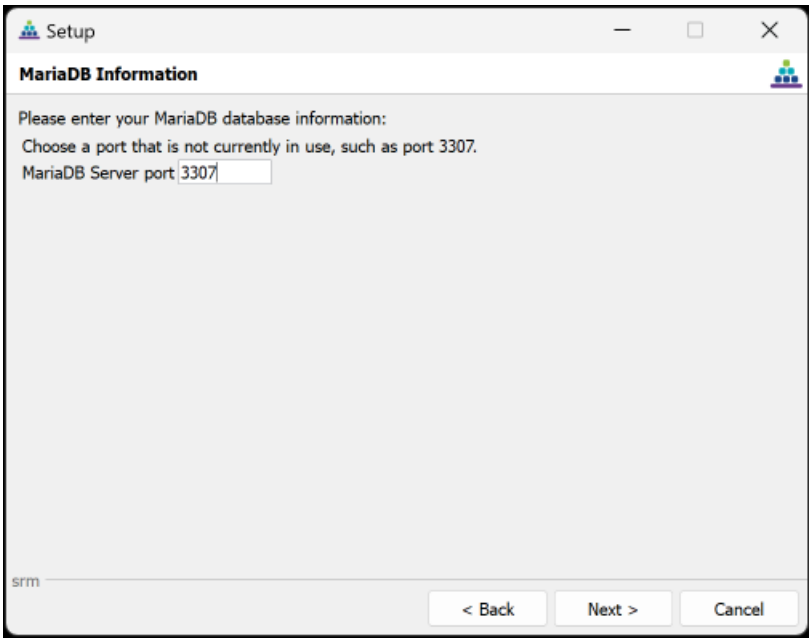
If you selected the default protocol (HTTP+HTTPS) on the previous screen, select which ports to use.



Click Next to continue.

MariaDB Database Information

Enter a port for your MariaDB installation.



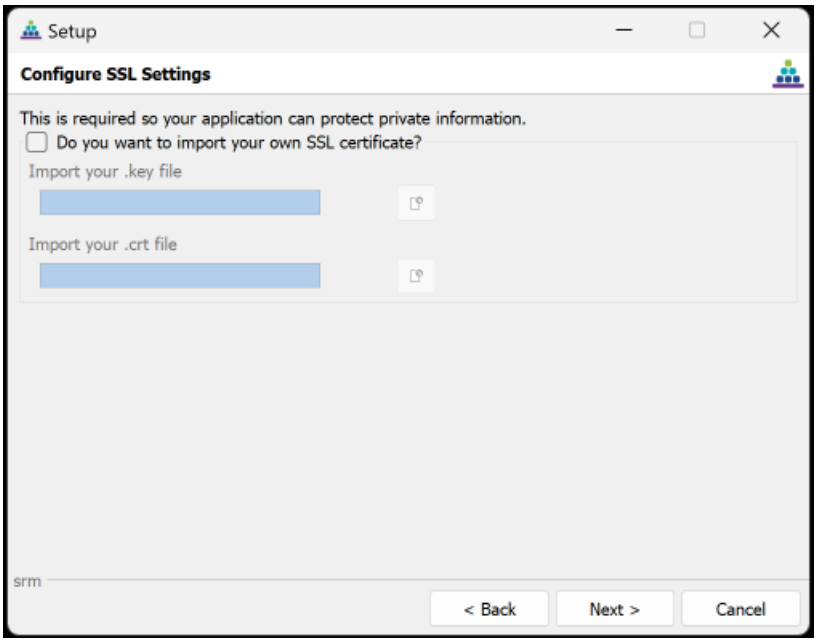
Click Next to continue.

SSL Configuration

What you do with the SSL configuration depends on your deployment scenario. By default, the installer will setup Software Risk Manager dependencies to offer both HTTP and HTTPS access to Software Risk Manager. This is sufficient for most evaluation scenarios. If, however, you are deploying Software Risk Manager for production usage, then we recommend that you use or obtain a valid SSL certificate for the host machine. If you don't use your own certificates, HTTPS is enabled by default with a self-signed certificate generated during installation. This will ensure secure communications to all clients accessing Software Risk Manager. Note that if you use the default HTTPS option with the self-signed certificate, most browsers will display a security warning when connecting, which will need to be accepted to proceed with access to the site.

If Software Risk Manager will be used in a networked environment, using HTTPS to connect is recommended. For example, once Software Risk Manager is installed, use `https://<hostname>/srm` instead of `http://<hostname>/srm` (substituting in your machine's hostname).

Click the "Next" button if you're using the Software Risk Manager self-signed certificate. Otherwise, use your own SSL certificate by getting the certificate and associated private key in PEM-encoded X.509 format. (Some issuers refer to this as “Apache format.”) Click "Do you want to import your own SSL certificate?" and browse to those files.



Replacing the SSL Certificate Post-Install

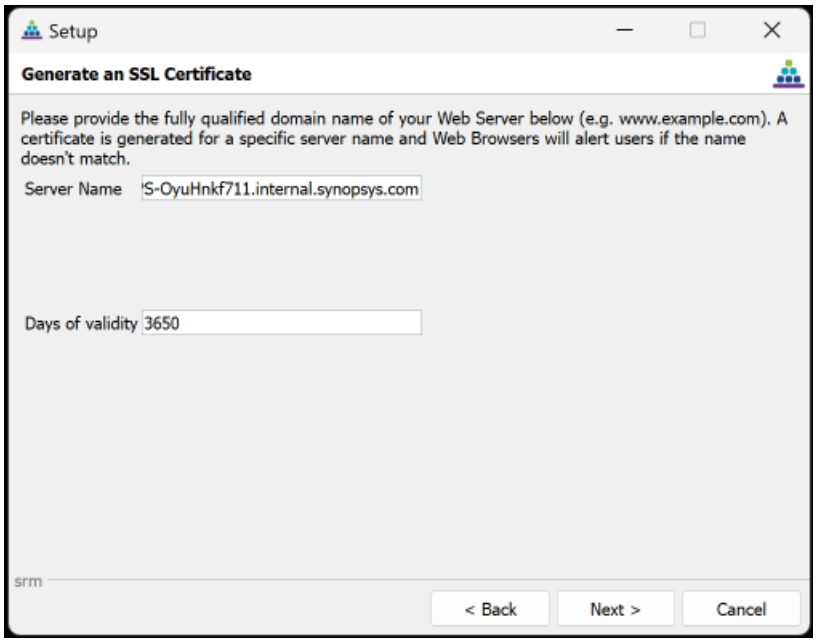
If you completed your installation using the Software Risk Manager self-signed SSL certificate (as shown above), you can replace this self-signed certificate with your own SSL certificate. You might do this if you skipped importing your own SSL certificate due to certificate failure and you now have a new SSL certificate.

To replace the SSL certificate after Software Risk Manager installation:

1. Get your SSL certificate in PEM-encoded X.509 format (name the file `server.crt`) and the private key for the certificate in PEM format (name the file `server.key`).
2. Copy the two files into the following directory: `<your-srm-install-directory>/apache2/conf/certs`.
3. Restart your Software Risk Manager Apache service. You can do this in the manager app or in the Services area on your control panel.

Generate an SSL Certificate

1. Enter the domain name for your Web Server to generate an SSL certificate.



2. Click Next to continue.

Trusting Certificates

If you're connecting with an application (such as Jira or Git) that is using a self-signed HTTPS certificate, you will need to add it to the Software Risk Manager Java trust store, as explained below.

Getting Software Risk Manager to Trust Your Third-party or Self-signed Certificate

Windows

1. Open a command prompt in administrator mode.
2. Change directory to the Software Risk Manager Java trust store: `cd <installation-directory>\Code Dx\java\bin` where the `<installation-directory>` default is `C:\Program Files\Software Risk Manager`.
3. Run `keytool -printcert -rfc -sslserver <server hostname:port> | keytool -importcert -keystore "../lib/security/cacerts" -storepass changeit -alias <name for your server> -noprompt`. Replace `<server hostname:port>` and `<name for your server>` with the appropriate information for your environment.
4. Restart the Software Risk Manager services.

Linux

1. Open a terminal window.
2. Change directory to the Software Risk Manager Java trust store: `cd <installation-directory>/java/bin` where the `<installation-directory>` defaults are `/opt/srm/` for Linux root and `/home/srm/` for Linux non-root.
3. Import the key to the Software Risk Manager Java installation's keystore:
   - [root] use `./keytool -printcert -rfc -sslserver <server hostname:port> | sudo ./keytool -importcert -keystore "../lib/security/cacerts" -storepass changeit -alias <name for your server> -noprompt`. Replace `<server`

hostname:port> and <name for your server> with the appropriate information for your environment.

- [non-root] use ./keytool -printcert -rfc -sslserver <server hostname:port> | ./keytool -importcert -keystore "../lib/security/cacerts" -storepass changeit -alias <name for your server> -noprompt. Replace <server hostname:port> and <name for your server> with the appropriate information for your environment.

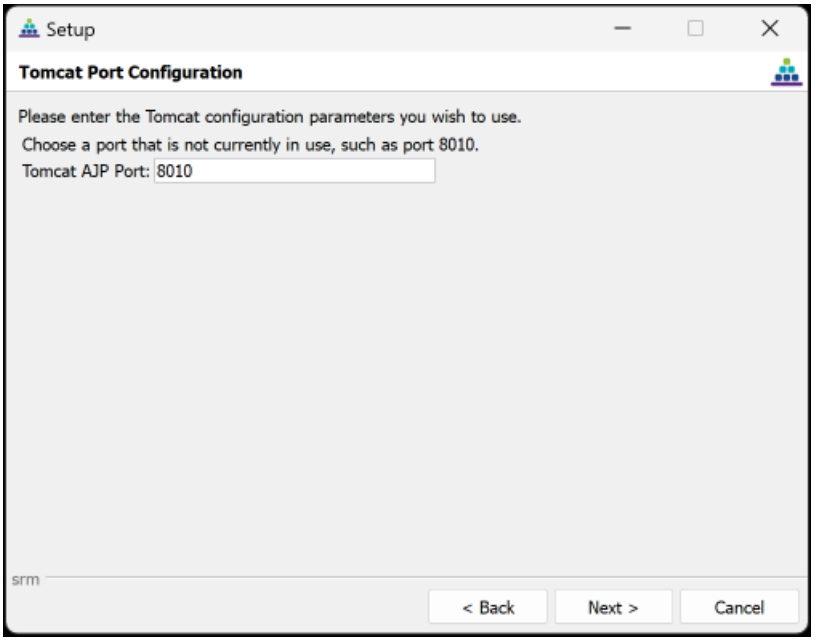
4. Restart the Software Risk Manager services.

Deleting a Certificate from the Trust Store

1. Open a Windows command prompt in administrator mode or a terminal in Linux.
2. Change the directory to the Software Risk Manager Java trust store: cd <installation-directory>/java/bin where the <installation-directory> defaults are C:\Program Files\Software Risk Manager on Windows; /opt/srm for Linux root; and /home/srm/ for Linux non-root. Replace <name for your server> with the appropriate information for your environment.
   - [Windows] use keytool -delete <name for your server> -keystore ../lib/security/cacerts.
   - [Linux root installation] use sudo ./keytool -delete <name for your server> -keystore ../lib/security/cacerts.
   - [Linux non-root installation] use ./keytool -delete <name for your server> -keystore ../lib/security/cacerts.
3. Restart the Software Risk Manager services.

Tomcat Port Configuration

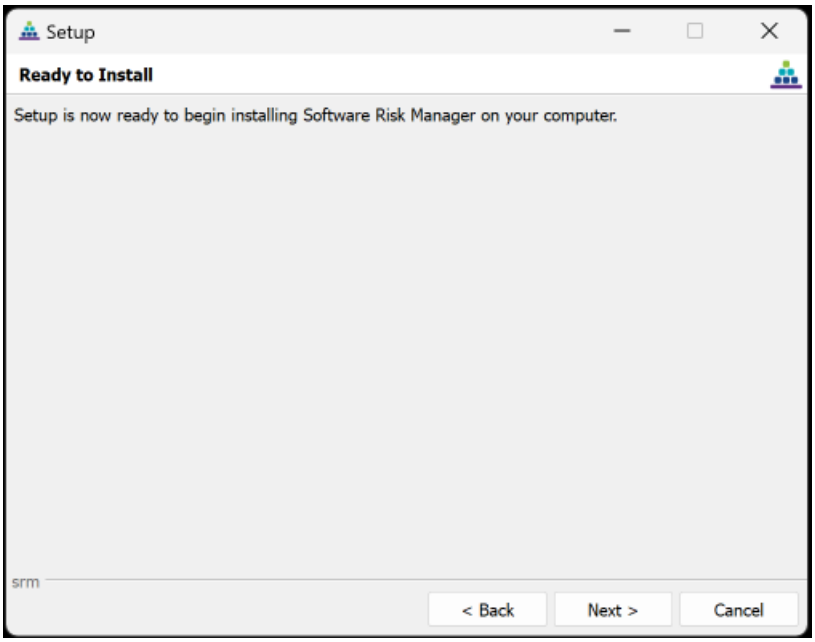
Enter the Tomcat configuration parameters you wish to use in the Tomcat AJP Port field.



Click Next to continue.

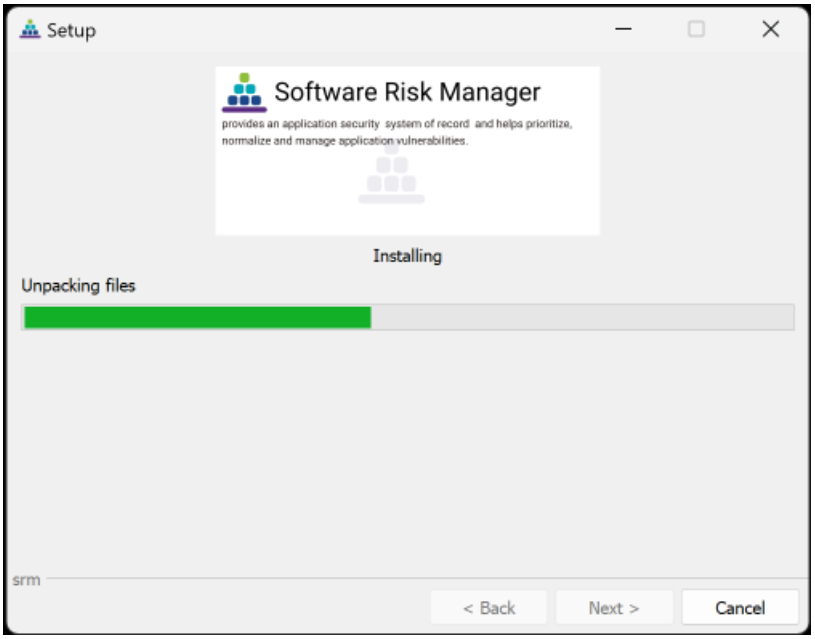
Time Expectations

Software Risk Manager is now ready to install. Use the Back button to correct any configuration options.

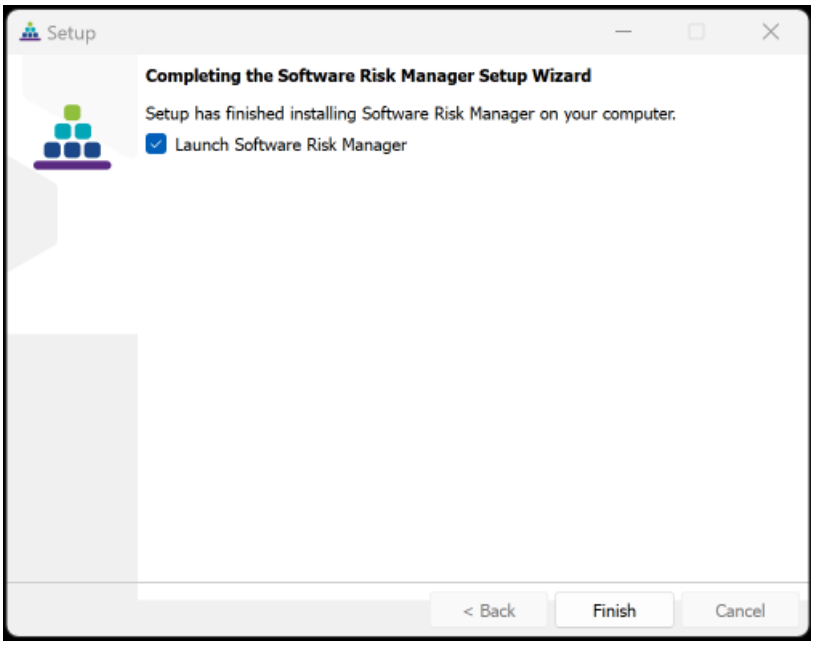


Click Next to begin the installation.

Installation may take several minutes depending on the hardware performance for the deployment machine.



When the installation is complete, the Setup Wizard completion screen will appear.

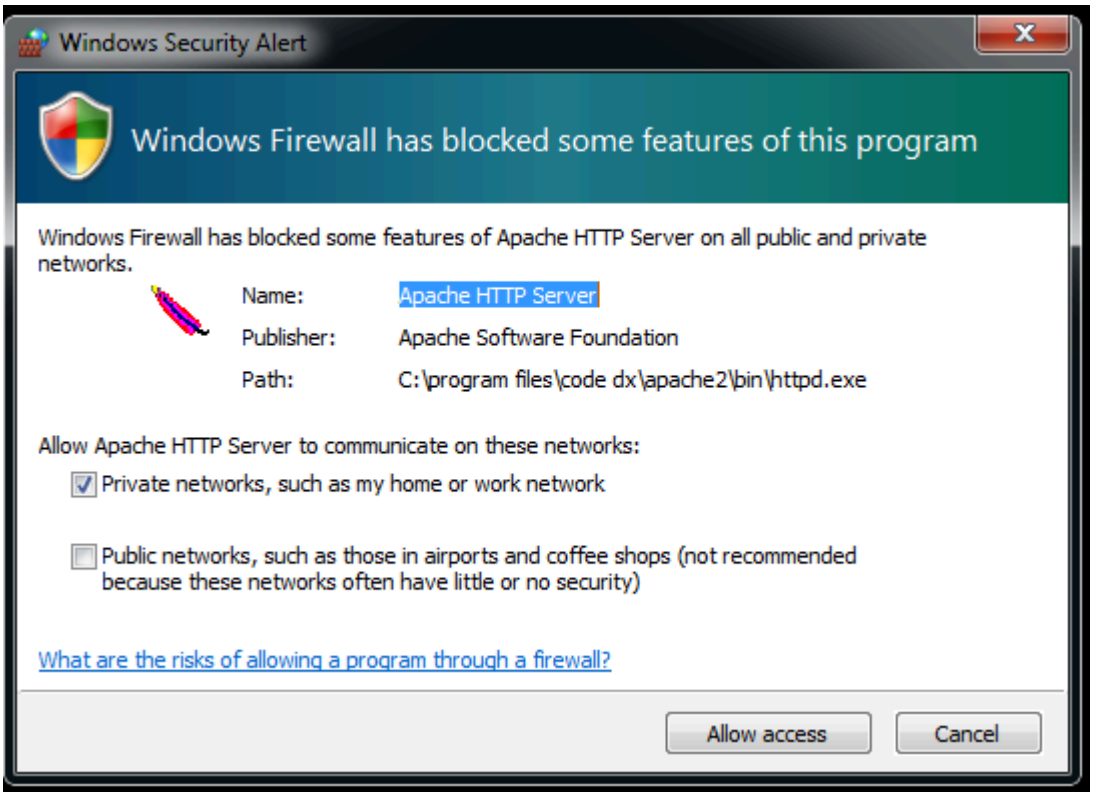


Select the Launch Software Risk Manager option to launch the applicaton after clicking the Finish button, then Click Finish.

Firewall

Software Risk Manager is a web application that requires a web server to house it. The Apache web server is provided and configured automatically with the installer. However, for most systems, this means that a firewall exception is necessary to allow incoming connections over the network so that users (including you) can access the installation from other machines. By default, the firewall ports that need to be opened up are 80 and 443, for HTTP and HTTPS respectively.

On Windows, you will be prompted with the following alert in order to grant the firewall exception.



Note: The first time you use Software Risk Manager, you must log in using the installer admin credentials. After that, you can change the admin's password within Software Risk Manager; however, changing the password in Software Risk Manager does **not** change the admin's password for the installer.

License File

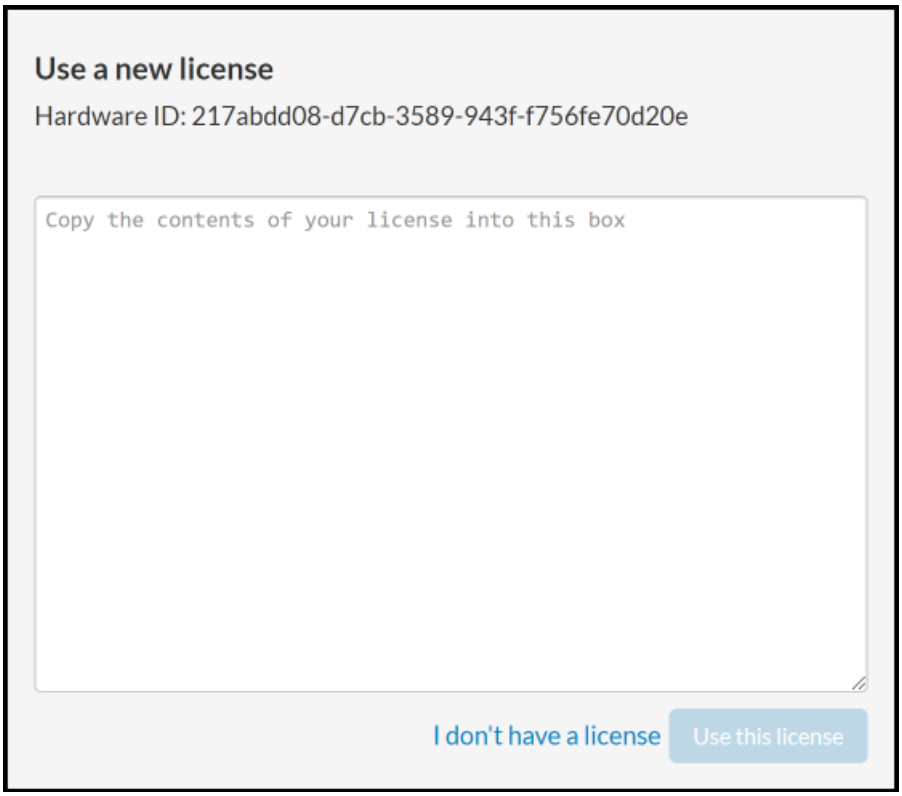
If you do not have a license or if your license has expired or is invalid, please request a license from our Sales Team (see *Requesting a License* below).

When you purchase Software Risk Manager, you will receive (usually via email) a new license (a blob of letters and numbers) that you will need to install.

Installing a New License

To install a new license, you must access the *License Management* page. If your license has expired or is invalid, the *License Management* page will automatically open. Otherwise, you can open the *License Management* page by clicking the license summary link at the top of the *Admin* page.

The image below shows the state of the page when you first arrive.



The header area provides the current license status. In this example, the license is valid, but if the license has expired or is invalid, the header will show a message indicating what is wrong.

The form shown in the example above allows you to upload the license you received. Open your license key file using any ASCII text editor (such as Notepad), copy the license text, and paste it in the *Use a new license* form.

Click the *Use this license* button. If the license is valid, you will be sent to the Software Risk Manager home page; otherwise, the *License Management* page will just reload.

If you require further assistance, please contact us.

Uninstallation or Reinstallation

Software Risk Manager is distributed with an uninstaller, which will remove the files and system services that were setup for Software Risk Manager. You will be prompted with the choice of removing the data files used by Software Risk Manager. Be aware that removing data files is irreversible and will result in data deletion.

To reinstall Software Risk Manager, running the uninstaller first is recommended.

Upgrading

During the upgrade process, the installer may present you with a list of issues. These issues must be resolved to complete the process.

If you are performing an upgrade using the installer over SSH, it is recommended to use `screen`, `tmux`, or similar to ensure the upgrade process will continue if the SSH session is disconnected.

Clean File System

The upgrade process expects only certain files and folders to exist in the installation location. If you are presented with a list of files, do the following:

- Open the installation directory (e.g., `C:\Program Files\Software Risk Manager` or `/opt/srm/`).
- Remove the specified files or folders but leave the backup folder (`backup-<timestamp>`), the `upgrade.ini` file, and the `properties.ini` file.

Remove Services

On Windows, the upgrade process requires specific Software Risk Manager services. Other Software Risk Manager services should be removed. If you are presented with a list of services, you can remove them by doing the following:

- Open the “Command Prompt” application as an administrator.
- Remove the presented service:

```
> sc delete CodeDxMariaDB
[SC] DeleteService SUCCESS
```

Note: Replace CodeDxMariaDB with the service name that the installer displays.

Free Ports

The upgrade process requires certain ports to be available. If you are presented with a list of ports, you can free them using the procedures shown below.

Windows

1. Open the “Resource Monitor” application as an administrator.
2. Click the “Network” tab.
3. Click the “TCP Connections” bar to see a list of TCP connections established.
4. Look for one of the ports in the presented list.

To continue with the upgrade process, you have to make sure that the presented ports are available. If the process using the required ports is a version of Software Risk Manager, please follow the instructions in the "Remove Services" section. If it is not related to Software Risk Manager, make sure to find the conflicting program and ensure that the problem will not occur after installation.

Linux and OS X

List the processes listening to a TCP connection using terminal:

```
sudo lsof -iTCP:8009 -sTCP:LISTEN
COMMAND   PID    USER   FD   TYPE   DEVICE SIZE/OFF NODE NAME
java    81106  codedx   50u  IPv4 40576813      0t0  TCP *:8009 (LISTEN)
```

Note: Only use sudo if installed as root.

To continue with the upgrade process, you have to make sure that the presented ports are available. You should stop the process that spawned the service and ensure that the conflict will not occur after installation.

Restoring to a Previous Version

In your backup folder, there's a file called `restore-backup`, which is used to restore to a previous version of Software Risk Manager.

Windows

1. Navigate to the backup folder, which is in the installation folder that you provided during installation. The default folder is `C:\Program Files\Software Risk Manager\backup-<timestamp>\` where `<timestamp>` is the date and time the upgrade was performed.
2. Right-click on `restore-backup.bat` and choose "Run as administrator."

Linux Root Installation

1. Navigate to the backup folder that was created during the upgrade process. The default folder is `/opt/srm/backup-<timestamp>/` where `<timestamp>` is the date and time the upgrade was performed.
2. Open a terminal and run `sudo chmod +x restore-backup.sh`.
3. Run `sudo ./restore-backup.sh`.

Linux Non-Root Installation

1. Navigate to the backup folder that was created during the upgrade process. The default folder is `/home/srm/backup-<timestamp>/` where `<timestamp>` is the date and time the upgrade was performed.
2. Open a terminal and run `chmod +x restore-backup.sh`.
3. Run `./restore-backup.sh`.

Manage Software Risk Manager Services

The Software Risk Manager installer includes a graphical tool to manage services, which can be found in the Software Risk Manager installation folder. For Windows, the tool is called *manager-windows*; for Linux, it's *manager-linux-x64.run*.

Open the appropriate tool and go to the *Manage Servers* tab to view and change the status of the services. To start, stop, or restart individual services, highlight the service, then click the desired action (see the screenshot below). Use the buttons on the bottom to change the status of all the services.

Alternatively, you can use the Windows Service Manager or the ctlscript shell files for Linux to change the status of Software Risk Manager services. To start, stop, or restart any Software Risk Manager services using ctlscript, navigate to the Software Risk Manager installation folder and follow the examples below:

- Start individual services (you can replace "start" with "stop" or "restart"):
  - `./ctlscript.sh start tomcat`
  - `./ctlscript.sh start apache`
  - `./ctlscript.sh start mysql`
- Start, stop, or restart all services:
  - `./ctlscript.sh start`
  - `./ctlscript.sh stop`
  - `./ctlscript.sh restart`
- Obtain the current status of all services:

- `./ctlscript.sh status`

Installation Using Docker Compose

A Docker-based installation consists of two parts: the SRM web application Docker image and the database upon which it depends. You can provide your own MariaDB or MySQL database instance or use the provided MariaDB Docker image.

For more information about using Docker Compose, see the sections below:

Software Risk Manager Core Deployments

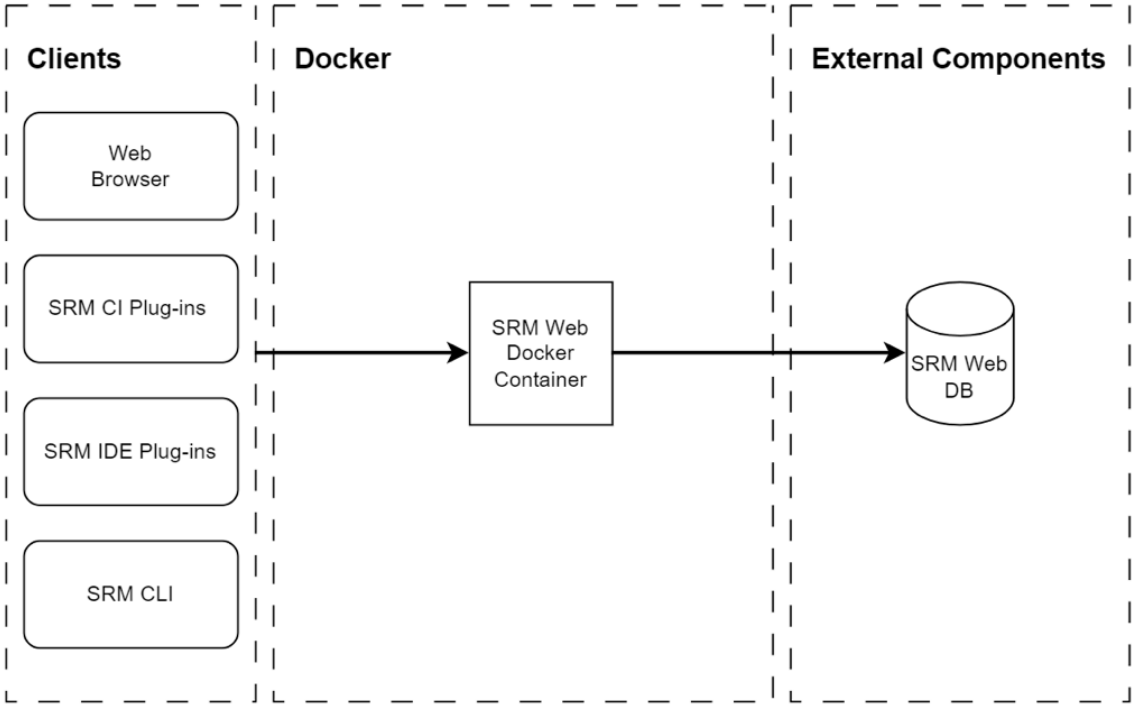
The footprint of your Software Risk Manager Docker-Compose deployment depends on whether or not you plan to use an external database.

The Software Risk Manager web application requires a MariaDB (version 10.6.x) or a MySQL (version 8.0.x) database instance. You can provide a database instance or use what's included in the default Compose file.

An external database can be a standalone instance or one managed for you by a cloud provider such as AWS or Azure.

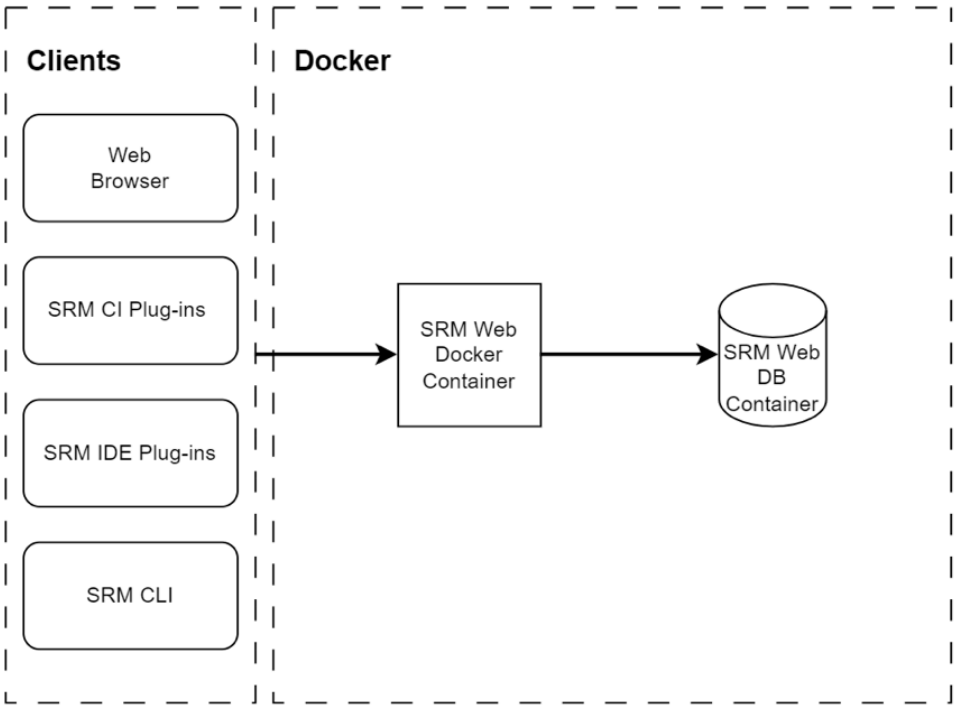
A deployment using an external database consists of one Software Risk Manager Docker container for the web application that depends on a Docker volume.

Figure 1: Core Deployment with an External Database



A deployment without an external database includes an additional Docker container for the Software Risk Manager Web database instance. The MariaDB database instance stores data using a Docker volume.

Figure 2: Core Deployment without an External Database



Docker Compose Requirements

Software Risk Manager deployment requires Docker Compose version 2.

System Size

Hardware requirements can vary based on a number of factors, including how many Software Risk Manager projects will be active at the same time, how frequently analyses will be conducted, whether built-in tools are being used, the number of results from tools in use, how many concurrent users are expected to use the system, and what other system interactions might be configured. Taking that into account, you can refer to the tables below for some general guidelines to help determine the size of your deployment.

Table 2:

Size	Total Projects	Daily Analyses	Concurrent Analyses
Small	1–100	1,000	8
Medium	100–2,000	2,000	16
Large	2,000–10,000	10,000	32
Extra Large	10,000+	10,000+	64

Core Feature Requirements

This section describes the web and web database requirements based on the system size.

Core Web Workload Requirements

Table 3:

Size	CPU Cores	Memory	IOPs	Storage
Small	4	16 GB	3,000	64 GB
Medium	8	32 GB	6,000	128 GB
Large	16	64 GB	12,000	256 GB
Extra Large	32	128 GB	32,000	512 GB

Core Web Database Workload Requirements

Table 4:

Size	CPU Cores	Memory	IOPs	Storage
Small	4	16 GB	3,000	192 GB
Medium	8	32 GB	6,000	384 GB
Large	16	64 GB	12,000	768 GB
Extra Large	32	128 GB	32,000	1536 GB

Core Persistent Storage Requirements

Table 5:

Volume	Feature	Description
Web AppData	Core	Required volume for web workload
DB Data	Core (when not using external database)	Database volume for database

Core Internet Access Requirements

Software Risk Manager uses internet access in the background for some activities, such as keeping tool data up-to-date and periodically checking for a new Software Risk Manager release.

Software Risk Manager does not require internet access; however, internet access is highly recommended to ensure full functionality.

To disable background internet access by Software Risk Manager, customize your Software Risk Manager deployment by setting `codedx.offline-mode = true`. The default is `false`. Note that this will not disable any internet access that may occur due to user action or configuration settings, such as Tool Connector, Git, or Issue Tracker configurations.

When internet access is enabled, Software Risk Manager will perform the following actions:

- **Update notifications.** Software Risk Manager will periodically check for newer versions and display an update notification when one is available. Requests for the latest version are sent to https://service.codedx.com/updates/latestVersionData.json.

- **Dependency-Check updates.** Dependency-Check will periodically download updates from the National Vulnerability Database, the Retire.js repository, or reach out to Maven Central while scanning Java dependencies (this aids in the dependency identification process, to cut down on both false positive and false negative results).
- **Offline mode.** If Software Risk Manager is in offline mode, this may lead to lower quality results when running Dependency-Check as a bundled tool.
- **Secure Code Warrior.** Unless noted elsewhere, Software Risk Manager will reach out to any URLs belonging to the `securecodewarrior.com` domain.

Configuration Tasks (Pre-work)

The following sections specify configuration tasks that might apply to your deployment. Pre-work tasks require updates to your Docker Compose file.

- If you are *not* using an external database, you will edit `docker-compose.yml`.
- If you are using an external database, you will edit `docker-compose-external-db.yml` instead.

For more information, see the following sections:

- Persistent Storage Pre-work
- External Web Database Pre-work
- Trust Certificates Pre-work
- HTTPS Pre-work

Persistent Storage Pre-work

Software Risk Manager depends on one or more Docker volumes. When using selinux, you must append `:Z` to volumes listed in your Docker Compose file, including the default volumes and any extra volumes added during configuration tasks.

External Web Database Pre-work

Software Risk Manager includes a MariaDB database that requires no configuration on your part, so you can skip this section if you do not plan to use an external database instance.

Your MariaDB/MySQL database must be on port 3306.

If you prefer an external database, the web workload supports MariaDB version 10.6.x and MySQL version 8.0.x. Complete the configuration tasks shown below before installing Software Risk Manager with an external web database.

Your MariaDB/MySQL database must include the following variable configuration:

- `optimizer_search_depth=0`
- `character_set_server=utf8mb4`
- `collation_server=utf8mb4_general_ci`
- `lower_case_table_names=1`
- `log_bin_trust_function_creators=1`

The `log_bin_trust_function_creators` parameter is required when using replication (sometimes enabled by default).

If you are using a database instance hosted by AWS, Azure, or GCP, refer to the following:

- How to create a new AWS parameter group

- How to configure Azure Database for MySQL parameters
- How to configure GCP Cloud SQL MySQL database flags.

Refer to the Web Database Workload Requirements section for database instance configuration details.

To create the database catalog and Software Risk Manager database user:

Note: If you have to reinstall Software Risk Manager and delete your Software Risk Manager data, you must repeat steps 2 and 3 below after deleting your Software Risk Manager Docker volume.

1. Create a database user.
   You can customize the following statement to create a user named "srm." Remove `REQUIRE SSL` when not using TLS (your database instance may require security transport).

   ```
   CREATE USER 'srm'@'%' IDENTIFIED BY 'enter-a-password-here' REQUIRE SSL;
   ```

2. Create a database catalog.
   The following statement creates a catalog named srmdb:

   ```
   CREATE DATABASE srmdb;
   ```

3. Grant required privileges on the database catalog to the database user you created.
   The following statements grant permissions to the srm database user.

   ```
   GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, CREATE TEMPORARY TABLES,
   ALTER, REFERENCES, INDEX, DROP, TRIGGER ON srmdb.* to 'srm'@'%'; FLUSH
   PRIVILEGES;
   ```

If your database configuration requires Software Risk Manager to trust a certificate (e.g., the Amazon RDS root certificate), follow the Trust Certificates instructions to trust your database certificate and update the volumes section of your `docker-compose-external-db.yml` file.

Trust Certificates Pre-work

Your Software Risk Manager instance can trust self-signed certificates or certificates issued by certificate authorities that are not trusted by default. Obtain a copy of the cacerts file from a Java 11 distribution, which will include the keytool program that you will need to run the following command:

```
keytool -import -trustcacerts -keystore ./cacerts -file /path/to/cert -alias cert-name
```

Note: The default password for a Java cacerts file is `changeit`.

You can mount your cacerts file by adding a line to the volumes list in the codedx-tomcat section:

```
    codedx-tomcat:
        ...
        volumes:
            - codedx-appdata:/opt/codedx
            - /path/to/cacerts:/opt/java/openjdk/lib/security/cacerts
        ...
```

Note: Append `:Z` to the extra volume mount when using selinux.

HTTPS Pre-work

The Tomcat container can support HTTPS. For example, generate a self-signed certificate with `openssl` or obtain a real certificate from a certificate authority:

```
openssl req -new -newkey rsa:4096 -days 3650 -nodes -x509 -subj "/C=US/ST=New York/L=Northport/
O=Software Risk Manager/CN=localhost" -keyout ./ssl.key -out ./ssl.crt
```

The server.xml file contains a configuration that supports HTTPS using **Tomcat's SSL/TLS capability** .

This template can be mounted over the existing `server.xml` in the Docker image. The SSL certificate and private key must also be mounted.

Update the codedx-tomcat section in your Docker Compose file (either `docker-compose.yml` or `docker-compose-external-db.yml`) with SSL and `server.xml` volume mounts, switching ports from `8080:8080` to `8443:8443`. See what follows for Docker Compose file content using port 8443 with extra volume mounts for `server.xml`, `ssl.key`, and `ssl.crt`.

```
    codedx-tomcat:
        ...
        volumes:
            - codedx-appdata:/opt/codedx
            - /path/to/ssl.crt:/usr/local/tomcat/conf/ssl.crt
            - /path/to/ssl.key:/usr/local/tomcat/conf/ssl.key
            - /path/to/server.xml:/usr/local/tomcat/conf/server.xml
        ports:
            - 8443:8443
        ...
```

Note: Append `:Z` to the extra volume mount when using selinux.

Software Risk Manager Installation

This section details how to start a functional instance of Software Risk Manager using Docker Compose. Installation instructions will vary based on whether or not you are using an external database. Complete the required pre-work configuration tasks before continuing with either the no-external-database or external-database installation path.

- Installation Prerequisites
- Volume Naming
- Installation without an External Database
- Installation with an External Database

Installation Prerequisites

Before installing Software Risk Manager, you must install the following:

1. Install Docker using **these instructions**.
2. Install **docker-compose**.

Volume Naming

When creating named volumes, Docker Compose will prepend the project name, which is the current directory name by default, to the volume name. In other words, if you have a Docker Compose install under the folder `srm-docker` and another under `srm-docker-2`, their volume names will be distinct and contain different data. Without specifying the `-p` option, the following two named volumes would exist:

- `srm-docker_codedx-appdata-volume`
- `srm-docker-2_codedx-appdata-volume`

Named volumes are created when doing `docker-compose up`, so if you want to override the default naming, you need to specify a project name the first time you execute the following command:

```
docker-compose -p srm -f ./docker-compose.yml up
```

Installation without an External Database

The `docker-compose.yml` file is used to install Software Risk Manager without an external database.

To install Software Risk Manager without an external database:

1. Select a password for the MariaDB root user, one that does not use single quote characters, and edit your `docker-compose.yml` file by specifying the password for both the `MARIADB_ROOT_PASSWORD` and `DB_PASSWORD` parameters.
2. Select a password for the Software Risk Manager admin user and edit your `docker-compose.yml` file by specifying the password for the `SUPERUSER_PASSWORD` parameter.
3. Run `docker-compose -f ./docker-compose.yml up -d` to start the SRM Docker containers.
4. Run `docker-compose -f ./docker-compose.yml logs -f` to view log data.
   When the message "The Server is now ready!" appears in the console, you can navigate to either http://hostname:8080/srm or https://hostname:8443/srm (depending on your HTTPS configuration) to log into your Software Risk Manager instance.

- To stop, run `docker-compose -f ./docker-compose.yml stop`.
- To remove the Docker containers automatically created, run `docker-compose -f ./docker-compose.yml down`.

Note: If you want to migrate data from an existing Software Risk Manager system, refer to these instructions.

Installation with an External Database

To install Software Risk Manager with an external database:

1. Select a password for the Software Risk Manager admin user and edit your `docker-compose-external-db.yml` file by specifying the password for the `SUPERUSER_PASSWORD` parameter.
2. Edit the `docker-compose-external-db.yml` file by appending `&useSSL=true&requireSSL=true` to the `DB_URL` parameter.
3. Edit the `docker-compose-external-db.yml` file as follows:
   a. Enter the database username for the `DB_USER` parameter.
   b. Enter the database password for the `DB_PASSWORD` parameter.
   c. Update the `DB_URL` parameter by specifying your database instance hostname.
   d. Update the `DB_URL` parameter by specifying your Software Risk Manager database name (e.g., "srmdb").
   The following is an example of a `DB_URL` parameter value using hostname `db-hostname` and database name `srmdb`:

```
"jdbc:mysql://db-hostname/srmdb?
sessionVariables=sql_mode='STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION'"
```

   If you followed the steps required for secure transport, your connection string will look like the following:

```
"jdbc:mysql://db-hostname/srmdb?
sessionVariables=sql_mode='STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION'&useSS
```

4. Follow the HTTPS instructions if your deployment requires TLS/SSL.
5. Start Software Risk Manager with the following command:

```
docker-compose -f ./docker-compose-external-db.yml up -d
```

You can run the following commands to view Software Risk Manager log data (assumes an `srm-docker-codedx-tomcat-1` container name):

```
docker exec srm-docker-codedx-tomcat-1 tail -f /opt/codedx/log-files/codedx.log
docker logs srm-docker-codedx-tomcat-1
```

Note: If you want to migrate data from an existing Software Risk Manager system, refer to these instructions.

Customizing Software Risk Manager

You can customize Software Risk Manager's behavior by specifying certain configuration properties.

Custom Props

Software Risk Manager features can be customized through the configuration file `codedx.props` which, by default, is located in your Tomcat container at `/opt/codedx`. (A full list of configuration parameters and how to change them can be found in the section on using the native installer.)

For example, to automatically sign a user out after 15 minutes of inactivity (20 minutes by default), set `session.lifetime` to `15 minutes`.

To set the sign-out property in a Docker Compose install:

1. With Software Risk Manager up and running, copy its `codedx.props` file to your current working directory using the following `docker cp` command, replacing the `srm-docker-codedx-tomcat-1` Docker container name as necessary (you can list running Docker containers with `docker ps`):

   ```
   docker cp srm-docker-codedx-tomcat-1:/opt/codedx/codedx.props .
   ```

2. Edit your local `codedx.props` file by appending the following line:

   ```
   session.lifetime = 15 minutes
   ```

3. Copy `codedx.props` back to its original location, replacing the `srm-docker-codedx-tomcat-1` Docker container name as necessary:

   ```
   docker cp codedx.props srm-docker-codedx-tomcat-1:/opt/codedx/codedx.props
   ```

4. Restart the SRM web container by running the Docker Compose `down` and `up` commands that you use for your deployment.

Custom Context Path

By default, Software Risk Manager is accessible at `/srm`. For backward compatibility, requests to `/codedx/api` and `/codedx/x` will be rewritten to `/srm/api` and `/srm/x` respectively, and requests to `/codedx` will be redirected to `/srm`.

You can change the Software Risk Manager context path by setting the SRM_WEBAPP_NAME environment variable in your Docker Compose file. The following example changes the default context path from `/srm` to `/mysrm`:

```
    codedx-tomcat:
        image: ...
        environment:
            DB_URL: ...
            SRM_WEBAPP_NAME: "mysrm"
```

With this configuration, you can access Software Risk Manager at `hostname/mysrm` after restarting Software Risk Manager. URL rewrites and redirects from `/codedx` become disabled when using a custom context path.

Backup and Restore

This section explains how to back up and restore Software Risk Manager using `backup.ps1` and `restore.ps1` in the `scripts` directory. Both scripts include a `-p` project name parameter that you should specify if you used a project name with your Docker Compose `up` command.

The backup script creates a new volume named codedx-backups where each backup gets stored by name. You can either specify a name or let the backup script generate a name for you formatted as `backup-{date}-{time}`.

Note: Be cautious of commands such as `docker volume prune`. The volume storing Software Risk Manager backups is not attached to a container and would be deleted.

For more information, see the following sections:

- Backup and Restore Prerequisites
- Creating a Backup
- Backup Retention
- Restoring a Backup

Back Up and Restore Prerequisites

The Software Risk Manager backup and restore scripts depend on PowerShell Core, which can be installed on macOS, Linux, and Windows.

When running the backup and restore scripts, make sure you're either in a PowerShell Core terminal environment or the command begins with `pwsh` so it's run by PowerShell Core.

Windows Prerequisites

Ensure you can run PowerShell Core scripts on Windows by switching your PowerShell Execution Policy to RemoteSigned (recommended) or Unrestricted. You must run the `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned` command from an elevated/administrator Command Prompt.

Creating a Backup

Backup instructions will vary based on whether or not you are using an external database. Use the procedures shown below to create a backup that will include the volume data for your deployment under the name `my-srm-backup`.

Backup without an External Database

To create a backup without an external database:

1. Stop your containers using the Docker Compose `down` command for your deployment.

   ```
   docker-compose -f /path/to/your-docker-compose-file down
   ```

2. Open a terminal window and change the directory to your `srm-docker` GitHub repository.

   ```
   cd /path/to/srm-docker
   ```

3. Run the `backup.ps1` script by specifying a backup name (e.g., `my-srm-backup`) and the path to your Docker Compose file.

```
pwsh ./scripts/backup.ps1 -BackupName my-srm-backup -ComposeConfigPath /path/to/your-docker-compose-file
```

Note: Specify a project name using the `-p` parameter, if necessary.

4. Verify that you see the following message indicating a successful backup:

```
Successfully created backup <backup-name>
```

Backup with an External Database

To create a backup with an external database:

1. Stop your containers using the Docker Compose `down` command for your deployment.

```
docker-compose -f /path/to/your-docker-compose-file down
```

2. Open a terminal window and change the directory to your `srm-docker` GitHub repository.

```
cd /path/to/srm-docker
```

3. Run the `backup.ps1` script by specifying a backup name (e.g., `my-srm-backup`), the path to your Docker Compose file, and the name of your web volume (you can list volumes with the `docker volume ls` command).

```
pwsh ./scripts/backup.ps1 -BackupName my-srm-backup -AppDataVol srm-docker_codedx-appdata-ex-db-volume -ComposeConfigPath /path/to/your-docker-compose-file
```

Note: Specify a project name using the `-p` parameter, if necessary.

4. Verify that you see this message indicating a successful backup:

```
Successfully created backup <backup-name>
```

5. Back up your external database.

Backup Retention

Backups will be automatically removed by the backup script if they meet one of the following criteria:

- **Older than 30 days.** The 30-day default can be changed with the `-Retain` backup script parameter, using `-Retain 0` to ignore backup age, `-Retain 5` for 5 days, and `-Retain 10:00` for 10 hours of retention.
- **Exceeds maximum backup count.** By default, a maximum of 10 backups will be stored at a time. This can be configured with the `-MaximumBackups` backup script parameter.

For more advanced usages of the backup script, such as setting the names of your Tomcat and DB containers if they are not the default, see the help info via the following command run from the srm-docker directory:

```
pwsh -Command get-help .\scripts\backup.ps1 -full
```

Restoring a Backup

You can use the included `restore.ps1` script in your `srm-docker/scripts` folder to restore a backup created by `backup.ps1`. This will restore Software Risk Manager data from a provided backup name that was either specified or generated when creating the backup.

To restore a backup:

1. Stop your containers using the Docker Compose `down` command for your deployment to avoid unexpected restore behavior.
2. Run the following command to restore a backup by entering its name from the list of backups:

   ```
   pwsh ./scripts/restore.ps1
   ```

   Note: You can use the `-BackupName` parameter to skip the backup list by specifying a specific backup.
3. Verify that you see the following message indicating a successful restore:

   ```
   Successfully restored backup <backup-name>
   ```
4. If you are using an external database, restore the related database backup now.

The restore command assumes no defaults have been changed about the Software Risk Manager Docker Compose environment. If defaults were modified or for more advanced usages of the restore script, see the help information provided by running the following command from the `srm-docker` directory:

```
pwsh -Command get-help .\scripts\restore.ps1 -full
```

Upgrading Software Risk Manager

This section details how to upgrade Software Risk Manager to the latest version.

Note: It is recommended that you create a backup of your Software Risk Manager Docker Compose environment before upgrading to the latest Software Risk Manager version.

Remove your Software Risk Manager container(s) before starting the upgrade. (Do not use the `-v` switch with the `down` command because it will delete your data volumes.)

```
docker-compose -f /path/to/your-docker-compose-file down
```

The recommended upgrade method is to pull the latest changes from GitHub. The Docker Compose file is updated with each Software Risk Manager release to reference the latest Docker image versions. Edits to local files, such as your Docker Compose file, may block your pull from GitHub, so use the following commands to stash your changes and reapply them after your pull:

```
cd /path/to/srm-docker
git stash save before-upgrade
git pull
git stash apply
```

If you see a "Merge conflict" message when you run `git stash apply`, edit the flagged file to resolve the conflict, then run `git add /path/to/file` to mark the file as merged.

If you do not want to use Git, you can alternatively download the latest ZIP file from GitHub. (If you use the ZIP to replace an existing folder, you must reapply any local edits, such as the changes you made to your Docker Compose file.)

Once you have the latest changes, run your Docker Compose `up` command to start Software Risk Manager:

```
docker-compose -f /path/to/your-docker-compose-file up
```

Running Software Risk Manager After Upgrade

After successfully upgrading, you can run the following command to see the effects of the upgrade:

```
docker-compose -f /path/to/your-docker-compose-file up
```

Pulling the latest files via Git ensures that you update your original files with the latest versions. Running your Docker Compose commands with the upgraded files from the same directory avoids unexpected behavior due to how docker-compose sets project names.

Migrating from the Software Risk Manager Native Installer to Docker Compose

Refer to the sections below to migrate your data from a system created by the Software Risk Manager Installer to your Docker Compose instance. The version you are running with Docker Compose must equal the version number of the system whose data you want to migrate. If necessary, upgrade your systems to a matching version.

Prerequisites

The Software Risk Manager migration script depends on PowerShell Core, which can be installed on macOS, Linux, and Windows. However, the target system you're migrating data to should have successfully gone through the installation process.

Windows Prerequisites

Ensure you can run PowerShell Core scripts on Windows by switching your PowerShell Execution Policy to `RemoteSigned` (recommended) or `Unrestricted`. You must run the `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned` command from an elevated/administrator Command Prompt.

Migration Paths

For more information on migration, see the following:

- Migration Without an External Database
- Migration With an External Database

Migration Without an External Database

The following steps cover migrating data from a Software Risk Manager system installed with the native installer to an existing Docker Compose deployment.

To migrate data from a Software Risk Manager system installed using the native installer:

1. Verify that your Software Risk Manager deployed with Docker Compose is running.
2. Verify that your Software Risk Manager deployed with the native installer is running.
3. Verify that the version numbers of both systems match.
4. Log on to your source Software Risk Manager server whose data you want to migrate.
5. Run `mysqldump` to create a backup file. You can run the following command to create a `dump-srm.sql` file after specifying the parameters that work for your database.

   ```
   mysqldump --host=127.0.0.1 --port=3306 --user=root -p codedx -r dump-srm.sql
   ```

   Note: The above command uses a database named `codedx`. Older versions of Software Risk Manager may use a database named `bitnami_codedx`.

6. You may encounter an issue running SRM if the database dump file makes use of `DEFINER`. Run the following command with `dump-srm.sql` replaced with the path of your database dump file. This will overwrite the existing database dump file with the definers removed.

```
(Get-Content "dump-srm.sql") -replace '\sDEFINER=`[^`]*`@`[^`]*`',' | Out-File dump-srm.sql
```

7. Locate the directory path for your Software Risk Manager AppData directory (e.g., /path/to/codedx_data/codedx_appdata). The AppData directory contains your analysis-files and log-files directories.
8. Copy your database dump file and Software Risk Manager AppData directory to the system running Software Risk Manager with Docker Compose.
9. Return to the system running Software Risk Manager with Docker Compose, change directory to this repository (srm-docker folder), and run the `migrate-data.ps1` script with the following commands. When prompted, enter the path to your dump-srm.sql file, your Software Risk Manager AppData directory, and specify the password for the Docker Compose root database user.

```
cd /path/to/srm-docker
pwsh ./admin/migrate-data.ps1
```

Note: The above command will use the default values for the script parameters `-tomcatContainerName` (srm-docker-codedx-tomcat-1), `-dbContainerName` (srm-docker-codedx-db-1), and `dbName` (codedx). You can find your Docker container names by running `docker ps`.

Your script output should look similar to the following:

```
pwsh ./admin/migrate-data.ps1
Enter the path to your mysqldump file: /path/to/dump-srm.sql
VERBOSE: Checking database dump file path...
Enter the path to your Software Risk Manager AppData folder: /path/to/codedx
VERBOSE: Checking appdata path...
VERBOSE: Checking appdata/analysis-files path...
Enter the password for the docker MariaDB root user: **********
VERBOSE: Checking PATH prerequisites...
VERBOSE: Checking running containers...
VERBOSE: Dropping database named codedx...
VERBOSE: Creating database named codedx...
VERBOSE: Creating temporary directory...
VERBOSE: Copying database dump file to container...
VERBOSE: Importing database dump file (may take a while)...
VERBOSE: Deleting database dump file...
VERBOSE: Deleting directories...
VERBOSE: Deleting directory /opt/codedx/analysis-files...
VERBOSE: Deleting directory /opt/codedx/keystore...
VERBOSE: Deleting directory /opt/codedx/mltriage-files...
VERBOSE: Copying directories...
VERBOSE: Copying directory /path/to/codedx/analysis-files to /opt/codedx/analysis-files...
VERBOSE: Copying directory /path/to/codedx/mltriage-files to /opt/codedx/mltriage-files...
VERBOSE: Restarting Software Risk Manager...
Restarting srm-docker-codedx-tomcat-1 ... done
Restarting srm-docker-codedx-db-1     ... done
Done
```

Migration With an External Database

The following steps cover migrating data from a Software Risk Manager system installed with the native installer to an existing Docker Compose deployment.

To migrate data from a Software Risk Manager system installed using the native installer:

1. Verify that your Software Risk Manager deployed with Docker Compose is running and using the docker-compose-external-db.yml file. The external database details will need to be added to this file. From the root of the project, the up command should look as follows:

```
docker-compose -f docker-compose-external-db.yml up
```

2. Verify that your Software Risk Manager deployed with the native installer is running.

3. Verify that the version numbers of both systems match.
4. Log on to your source Software Risk Manager server whose data you want to migrate.
5. Run `mysqldump` to create a backup file. You can run the following command to create a dump-srm.sql file after specifying the parameters that work for your database.

```
mysqldump --host=127.0.0.1 --port=3306 --user=root -p codedx -r dump-srm.sql
```

   Note: The above command uses a database named codedx. Older versions of Software Risk Manager may use a database named bitnami_codedx.

6. You may encounter an issue running SRM if the database dump file makes use of DEFINER. Run the following command with dump-srm.sql replaced with the path of your database dump file. This will overwrite the existing database dump file with the definers removed.

```
(Get-Content "dump-srm.sql") -replace '\sDEFINER=`[^`]*`@`[^`]*`',' ' | Out-File dump-srm.sql
```

7. Return to the system running Software Risk Manager with Docker Compose, change directory to this repository (srm-docker folder), and run the migrate-data.ps1 script with the following commands. The script will guide you through the rest of the procedure.

```
cd /path/to/srm-docker
pwsh ./admin/migrate-data.ps1 -externalDatabase
```

   Note: The above command will use the default values for the script parameters -tomcatContainerName (srm-docker-codedx-tomcat-1) and dbName (codedx). You can find your Docker container names by running docker ps.

Your script output should look similar to the following:

```
pwsh ./admin/migrate-data.ps1 -externalDatabase
Enter the path to your Software Risk Manager AppData folder: /path/to/codedx
VERBOSE: Checking appdata path...
VERBOSE: Checking appdata/analysis-files path...
VERBOSE: Checking PATH prerequisites...
VERBOSE: Checking running containers...

Since your SRM deployment uses an external database (one that you maintain on your own
that is not installed or updated by the SRM deployment script), you must restore the dump-srm.sql
 file you
created in Step 5 of the migration instructions.

You can restore mysqldump files using a command that looks like this:

mysql -uroot -p srmdb < dump-srm.sql

Note: Replace 'root' and 'srmdb' as necessary.

VERBOSE: Deleting directories...
VERBOSE: Deleting directory /opt/codedx/analysis-files...
VERBOSE: Deleting directory /opt/codedx/keystore...
VERBOSE: Deleting directory /opt/codedx/mltriage-files...
VERBOSE: Copying directories...
VERBOSE: Copying directory /path/to/codedx/analysis-files to /opt/codedx/analysis-files...
VERBOSE: Copying directory /path/to/codedx/mltriage-files to /opt/codedx/mltriage-files...
VERBOSE: Restarting Software Risk Manager...
Restarting srm-docker-codedx-tomcat-1 ... done
Done
```

Uninstall

You can remove Software Risk Manager and the related Docker volume(s) by running the following command:

```
docker-compose -f /path/to/your-docker-compose-file down -v
```

Installation Using Kubernetes

Kubernetes deployment is required for high-availability and the ability to use Scan Farm and Tool Orchestration.

For complete deployment instructions, see Deploying Software Risk Manager on Kubernetes.

Manual Installation

This section provides information on installing SRM manually. Note that while manual installation provides expanded configuration options, it also requires a high level of technical knowledge and experience with the third-party tools and software required to deploy SRM.

Stack

The supported stack for SRM is as follows:

- Java 11
- Apache Tomcat 8.5.x or 9.0.x
- MariaDB 10.6.x or MySQL 8.0.x

Note: Other configurations are not guaranteed to work.

Prerequisites

Nothing specific beyond the stack should be required to install SRM on Linux or macOS.

On Windows, the VC15 runtime is required. For more information, see https://support.microsoft.com/en-us/help/2999226/update-for-universal-c-runtime-in-windows.

Setup

Start by extracting the SRM zip file. The three files of interest are as follows:

- `srm`
- `codedx.props`
- `logback.xml`

AppData

SRM stores a variety of files, including configuration files, temporary files created during analyses, log files, and copies of files uploaded by the user. The root location for these files is the SRM `AppData` directory. You will need to create a directory to serve as the `AppData` directory (which will be referred to in these instructions as SRM_APPDATA). Make sure that the user running Tomcat has read-and-write privileges to SRM_APPDATA.

Move both `codedx.props` and `logback.xml` into SRM_APPDATA.

The user that Tomcat will be running as will need to have full access to the appdata folder. Since sensitive data may be stored here as well, it is recommended that access to this folder be restricted via disk permissions as well as access restrictions to the machine SRM will be running on.

Database

Create a new, empty database for SRM to use and create a user for SRM with `SELECT, INSERT, UPDATE, DELETE, CREATE, CREATE TEMPORARY TABLES, ALTER, REFERENCES, INDEX, DROP, TRIGGER` permissions on the database.

For example,

```
CREATE USER 'srm'@'%' IDENTIFIED BY 'enter-a-password-here';
CREATE DATABASE srm;
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, CREATE TEMPORARY TABLES, ALTER, REFERENCES, INDEX,
 DROP, TRIGGER ON codedx.* to 'srm'@'%';
FLUSH PRIVILEGES;
```

Edit the `codedx.props` file to reflect your new database and the credentials required to connect to it. For example,

```
swa.db.url = jdbc:mysql://localhost/srm
swa.db.user = srm_username
swa.db.password = srm_user_password
```

Required changes:

- Make sure `sql_mode` doesn't contain `ONLY_FULL_GROUP_BY` or `PAD_CHAR_TO_FULL_LENGTH`. Newer versions of MySQL (as of v5.7) use a `sql_mode` that includes `ONLY_FULL_GROUP_BY`. MariaDB includes neither setting by default, as of this writing. Make sure if using MySQL, `sql_mode` doesn't include `NO_AUTO_CREATE_USER` as it's no longer supported in MySQL 8. To view the current configuration of this, you may run the query `select @@sql_mode`. For example, the defaults for MySQL 8 with the `ONLY_FULL_GROUP_BY` option removed would be `sql_mode=STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,N`
- `optimizer_search_depth=0`
- `character_set_server=utf8mb4`
- `collation_server=utf8mb4_general_ci`
- `log_bin_trust_function_creators=1`

Tomcat

Drop the `srm.war` file into Tomcat's `webapps` directory. (You can also deploy it via Tomcat's manager interface if you prefer).

You also need to add a property to let the SRM app know the value of `SRM_APPDATA`. You will need to set a value for the Java property `codedx.appdata`. You will also need to add `-Dcodedx.appdata=SRM_APPDATA` to the list of `CATALINA_OPTS` values. (Where that's done is system-dependent: see the Tomcat Documentation for details.) Alternatively, you can define a `SRM_APPDATA` environment variable with this value, if you'd prefer that to configuring `CATALINA_OPTS`.

Additionally, you will need to ensure that the JVM will use `UTF-8` as its file encoding. Doing so prevents issues with Japanese user input like comments and manual results. Add `-Dfile.encoding=UTF-8` to the list of `CATALINA_OPTS` values.

Now when you start Tomcat, SRM will be able to access the `codedx.props` file.

Configuration

To make Tomcat stop reporting about cache issues, edit `conf/context.xml` and add `<Resources cachingAllowed="true" cacheMaxSize="128000" />` inside the `<Context>`.

It is also recommended to edit `conf/server.xml` and raise the value of `Connector connectionUploadTimeout`, if applicable. This is set to 3600000 by default when using the SRM installer. We also recommend setting `disableUploadTimeout` to `false`. In the same file, `connectionTimeout` on the `Connector` is by default `20000` for an installation of SRM.

Further configuration may be performed per your deployment and requirements (such as configuring AJP and placing an instance of Apache HTTPD in front of Tomcat or changing the logging configuration).

Starting SRM

For version 2023.12.5 or later, if SRM is being deployed behind a proxy that is using HTTPS, it is recommended to set the prop `auth.cookie.secure` to `true`. This will allow the secure flag on the session cookie to be set and passed along through HTTPS via the proxy.

Once everything is configured, start the database and Tomcat (make sure the database is running before you start Tomcat). Open a web browser and point it at SRM (e.g., `http://localhost/srm/`). You will be presented with an installation screen allowing you to set the super admin username and password and initialize SRM.

You may need to add a `codedx.internal-url` setting to your `codedx.props` file. This should contain a URL that SRM can access itself at, for example `http://localhost/srm/` or `https://localhost/srm`. This will depend on how exactly SRM is deployed.

Software Risk Manager Configuration

While the Software Risk Manager installer automates the configuration steps needed to match the system and user selections made during the installation process, certain advanced options are configurable from the Software Risk Manager configuration files. This section describes the configuration options for Software Risk Manager in the event that changes to the default setup are required.

Understanding the AppData Directory

Software Risk Manager needs a place to store a variety of files, such as the analysis inputs it receives that the source code that it uses to display in the *Finding Details* page, log files, and configuration files. These files are placed in the Software Risk Manager appdata directory. This directory also contains important information from the ongoing Software Risk Manager usage activity; therefore, it is recommended that this directory be in a stable location that is periodically backed up.

The location of the appdata directory is set during installation. By default, the appdata directory is located in the following locations:

- For Windows: `C:\ProgramData\Software Risk Manager\codedx_appdata\`
- For Linux (installed as root): `/var/opt/srm/`
- For Linux (installed as non-root: `/home/srm_data/codedx_appdata/`
- For Docker: `/opt/codedx/`

Enabling Intelligent Orchestration

Software Risk Manager provides an Intelligent Orchestration integration that allows the application of an IO Pre-Scan policy to a Software Risk Manager analysis. (For more information on Intelligent Orchestration and its use, see Intelligent Orchestration in the Software Risk Manager User Guide.)

Note: Intelligent Orchestration requires a separate IO license and may not be available in all implementations.

To enable IO functionality, add the following configuration settings to the codedx.props file:

- `io.enabled = true` - sets whether or not to enable IO [default: false]
- `io.url = io-url` - sets the base URL for your IO server [example: https://io.codedx.com]
- `io.api-key = io-token` - sets the token used by Software Risk Manager to access the IO API [example: ABCD...===]

(For more information on property settings and the `codedx.props` file, see Configuration Files.)

Configuration Files

Log Configuration File

Software Risk Manager uses Logback for logging. A sample `logback.xml` file is provided in the appdata directory. For more information about the logging configuration, consult the Logback manual.

Automatic Log Deletion

For fresh installations of Software Risk Manager, the Logback configuration file will specify a default retention period of seven days. To change the default, replace the value of `maxHistory` in the `logback.xml` file with the desired number of days to retain a log file. Log files outside of this period will be deleted.

Existing installations may achieve this same behavior by adding the following within the `<rollingPolicy>` block in the `logback.xml` configuration file.

```
<!-- keep 7 days worth of history -->
<maxHistory>7</maxHistory>
```

After making any changes, it is recommended that you delete the log files—except `codedx.log`—from the `log-files` directory to avoid unexpected behavior. Problems may occur because existing installations will likely exceed the maximum number of log files Logback will delete at a time.

Note: With this configuration change, eligible log files are deleted with each rollover period (which is daily for the default Software Risk Manager Logback configuration). If you also want log files to be deleted when the Software Risk Manager server is started, add the code shown below just after `maxHistory`.

```
<cleanHistoryOnStart>true</cleanHistoryOnStart>
```

You must restart the Tomcat server after you've completed your revisions. This can be done by launching the Software Risk Manager application's Manager Tool, clicking the Manage Servers tab, selecting Tomcat server, and then clicking Restart.

Software Risk Manager Properties File

The most important configuration file is `codedx.props`, also referred to in this guide as the "props" file. Located in the appdata directory, the props file configuration determines a variety of settings, including the database connection information, the analysis behavior, and Active Directory integration.

The props file is formatted as a .properties file, using key-value pairs to set configuration fields.

When changing this file, remember to delete the comment designation (#) at the beginning of the line.

You must restart the Tomcat server after you've completed any revisions. This can be done by launching the Manager Tool, clicking the Manage Servers tab, selecting Tomcat server, and then clicking Restart.

Database Connection Config

Software Risk Manager requires a MariaDB database for storage. The MariaDB database is automatically installed and configured during installation. The following properties are used to configure Software Risk Manager database connections and are set automatically during installation. **Please do not modify these settings** unless there is a strong need to setup an alternate database for use with Software Risk Manager. However, doing so might cause problems when installing Software Risk Manager upgrades.

- `swa.db.url` - The JDBC URL of the database Software Risk Manager will be communicating with
- `swa.db.user` - The username that will be used to access the database
- `swa.db.password` - The password that will be used to access the database

For instance, to configure Software Risk Manager to communicate with a MariaDB database running on the same machine as the Software Risk Manager server with a username of "database_username" and password of "database_password," use the following configuration:

```
swa.db.url = jdbc:mysql://localhost/codedx
swa.db.user = database_username
swa.db.password = database_password
```

Internet Access

Software Risk Manager uses internet access in the background for some activities, such as keeping tool data up-to-date and periodically checking for a new Software Risk Manager release.

Software Risk Manager does not require internet access; however, to insure full functionality, internet access is highly recommended.

To disable background internet access by Software Risk Manager, add `codedx.offline-mode = true` in your properties file (`codedx.props`). The default is `false`. Note that this will not disable any internet access that may occur as a result of user action or configuration settings, such as *Tool Connector*, *Git*, or *Issue Tracker* configurations.

When internet access is enabled, Software Risk Manager will perform the following actions:

- Update notifications - Software Risk Manager will periodically check for newer versions and display an update notification when one is available.
- Dependency-Check updates - Dependency-Check will periodically download updates from the National Vulnerability Database, the Retire.js repository, or reach out to Maven Central while scanning Java dependencies (this aids in the dependency identification process, to cut down on both false positive and false negative results). If Software Risk Manager is in offline mode, this may lead to lower quality results when running Dependency-Check as a bundled tool.
- Secure Code Warrior - Unless noted elsewhere, Software Risk Manager will reach out to any URLs belonging to the `securecodewarrior.com` domain.

Dependency-Check External Access

The base paths below are external resources that Dependency-Check may attempt to access during analyses or updates. If Software Risk Manager is not running in offline mode, ensure that all of the following paths are accessible to allow normal operation:

- https://jeremylong.github.io/DependencyCheck/current.txt
- https://nvd.nist.gov/feeds/json/cve/1.1/nvdcve-1.1-modified.json.gz
- https://nvd.nist.gov/feeds/json/cve/1.1/nvdcve-1.1-%d.json.gz (where %d is a year)
- https://static.nvd.nist.gov/feeds/xml/cpe/dictionary/official-cpe-dictionary_v2.3.xml.gz

- https://repository.sonatype.org/service/local/
- https://search.maven.org/solrsearch/select
- https://search.maven.org/remotecontent?filepath=
- https://repo1.maven.org/maven2/
- https://ossindex.sonatype.org
- https://registry.npmjs.org/-/npm/v1/security/audits
- https://raw.githubusercontent.com/Retirejs/retire.js/master/repository/jsrepository.json

Note: The first resource (https://jeremylong.github.io/DependencyCheck/current.txt) is not necessary for proper operation; however, Dependency-Check will occasionally attempt to access it to check the latest release version number.

Software Risk Manager External Access

To see the latest Software Risk Manager version: https://service.codedx.com/updates/latestVersionData.json

Secure Code Warrior External Access

For Software Risk Manager Secure Code Warrior integration, Software Risk Manager will attempt to reach out to a number of URLs that belong to the `securecodewarrior.com` domain. If Software Risk Manager is not running in offline mode and Secure Code Warrior functionality is enabled, ensure that the domain `securecodewarrior.com` (and all subdomains) are accessible to allow normal operation.

External URL

In most cases, Software Risk Manager can infer its own external URL based on the requests sent to it from users. For example, you might access Software Risk Manager from `https://localhost/codedx` or `https://codedx.mycompany.com`. This information is necessary for some subprocesses of Software Risk Manager that need to communicate back to Software Risk Manager over HTTP, or to provide accurate links back to Software Risk Manager when generating content—Issue Trackers, for example.

In some cases, the accurate "external URL" may not be available through inspection of user requests. When this happens, the value must be provided from your *codedx.props* file via the `codedx.external-url` property.

```
codedx.external-url = https://codedx.mycompany.com
```

Additionally, there are some cases where you may want internal services (running on the same machine as Software Risk Manager) to access Software Risk Manager from a different URL compared to the external URL. By default, the external URL will be used in this scenario (unless overridden by setting `codedx.internal-url`):

```
codedx.internal-url = https://localhost:8443/codedx/
```

Active Directory/LDAP Configuration

Software Risk Manager allows you to create and deactivate new users that are only known to the Software Risk Manager system. You may, however, want to let users use the same credentials as they do for your organization. To facilitate this, you may provide an LDAP or Active Directory configuration in the properties file (`codedx.props`). The available settings are as follows:

- `auth.ldap.url = ldap://127.0.0.1:389/` – sets the LDAP URL to connect to; this setting is *required* for LDAP usage.
- `auth.ldap.systemUsername = sAMAccountName=admin,ou=users,dc=codedx,dc=com` – sets the system username (full account DN) that is used when connecting to the LDAP server for authorization queries; this setting is not required if the LDAP server accepts anonymous queries.
- `auth.ldap.systemPassword = secret` – sets the password corresponding with the system username (above) that is used when connecting to the LDAP server for authorization queries; this setting is not required if the LDAP server accepts anonymous queries.
- `auth.ldap.authenticationMechanism = simple` – sets LDAP authentication mechanism for authorization queries; accepted values are *none* (anonymous) and *simple* [default: none].
- `auth.ldap.userSearchTemplate = sAMAccountName={0},ou=users,dc=codedx,dc=com` – controls the template to generate the user search query for authorization; more information can be found here [default: `<direct user input>`].
- `auth.ldap.useAsDnTemplate = false` – whether to use the provided search template(s) as DN templates instead [default: false].
- `auth.ldap.useStartTLS = false` – enable or disable the use of StartTLS after establishing an LDAP connection [default: false].
- `auth.ldap.legacyMode = false` – enable or disable the legacy Code Dx LDAP implementation [default: false].
- `auth.ldap.user.groupAttribute = memberOf` – set the name of the LDAP attribute on a user that lists their groups [default: memberOf].
- `auth.ldap.group.nameAttribute = cn` – set the name of the LDAP attribute on a group that indicates their readable name (leave blank for pass-through) [default: cn].
- `auth.ldap.user.displayNameAttribute = cn` – set the comma-separated list of LDAP attributes names on a user indicating their display name; see the relevantsections below for syntax [default: cn, userPrincipalName, sAMAccountName].
- `auth.ldap.sync.autoUpdate = false` – set whether or not to automatically schedule LDAP group membership updates [default: false].
- `auth.ldap.sync.refreshInterval = 5 minutes` – set the refresh interval for automatic LDAP group membership updates in readable text [default: "5 minutes"].
- `auth.ldap.searchSubTree = true` – sets whether or not to search sub-trees for users with the provided userSearchTemplate(s) [default: true].

Note: LDAP property names were changed in Code Dx version 3.5.4. The previous names may still be used.

You may be able to get by with only setting the `auth.ldap.url` property, depending on the complexity of your LDAP setup. It may be necessary to use a fully-qualified username (e.g., `john.doe@example.com` rather than just `john.doe`) when adding the user to Software Risk Manager.

The `systemUsername`, `systemPassword`, and `authenticationMechanism` properties don't need to be set for a secure LDAP configuration, due to the fact that they are not used during user authentication, but for querying user information that will be used during their authentication. Read the section below for more details.

User Search Template

The user DN is the fully qualified LDAP identifier for the user when logging in. In some LDAP systems, the user DN may require an attribute that the actual user isn't aware of, such as an integer ID. Rather than

asking the user for this hidden attribute, the user provides a different unique identifier (such as their account name) and Software Risk Manager will insert that into the given query to find a matching user with that attribute. The text they enter as their username is used to replace the `{0}` part of the template.

In the example above (`sAMAccountName={0},ou=users,dc=codedx,dc=com`), when the user attempts to log in, a query will be run on `sAMAccountName=test,ou=users,dc=codedx,dc=com` to retrieve the user DN, which may give something like `uid=12345,ou=users,dc=codedx,dc=com`. This user DN is then used to bind an LDAP connection using the discovered DN and the user's given password.

Note: In order to make this query, Software Risk Manager needs to be able to authenticate against the LDAP server without the user's credentials. If the LDAP server supports anonymous queries, this won't be an issue. If the LDAP server requires authentication, assign `auth.ldap.authenticationMechanism = simple` and assign `auth.ldap.systemUsername`/`auth.ldap.systemPassword` to the credentials to use when making these queries. The `systemUsername` should be the full, bindable DN for the user.

If no template is provided, the username entered by the user will be directly used, which may not map to what your LDAP server is expecting. When specifying a user search template, the `{0}` placeholder must be present. If it is missing, an error will be logged at Software Risk Manager startup, and LDAP authentication will be disabled until the template is corrected. After correcting the LDAP authentication, restart the Software Risk Manager Tomcat server.

Using as a DN Template

If performing a user search before binding is untenable in your environment, you can set Software Risk Manager to treat the search templates as direct DN templates instead by assigning `auth.ldap.useAsDnTemplate = true`. This means that each authentication performs only one bind, using the resolved DN and the provided user password.

While this may provide slightly better performance, it is mostly useful when anonymous binds are unsupported and there is no system user account available to perform a search.

Note: Enabling this option will disable LDAP connection pooling, so a new connection will be made and bound for each authentication attempt. Also note that this option cannot be enabled when using multiple templates.

Multiple Templates

Multiple search templates can be used by adding multiple, uniquely named properties, starting with `auth.ldap.userSearchTemplate`. For example,

```
auth.ldap.userSearchTemplate-A = sAMAccountName={0},ou=users,dc=codedx,dc=com
auth.ldap.userSearchTemplate.B = cn={0},ou=customers,dc=codedx,dc=com
```

With the above properties, both of those search templates will be used to resolve a user DN when signing in. The property names can be anything, as long as they start with `auth.ldap.userSearchTemplate` and are unique. If `auth.ldap.userSearchTemplate-A` was used twice in the above example, the second assignment would overwrite the first.

Note: Make sure that any LDAP user added to Software Risk Manager can be resolved by only *one* of the registered search templates.

Sign-in with Multiple Names

When multiple search templates are provided, it is possible for the same LDAP user to sign in using different names. Software Risk Manager resolves each LDAP user to their DN and stores that as their identity, allowing the user to sign into the same Software Risk Manager account regardless of the name used.

This DN resolution occurs when a user signs in and not when a user is manually registered through the User Admin page. For their first sign-in, they must use the name that has been registered. This causes their DN to be properly resolved and stored, allowing sign-in under different names.

User Merging

In rare cases, it is possible for two LDAP user entries to exist in Software Risk Manager that refer to the same LDAP user. If two separate accounts resolve to the same DN during authentication, Software Risk Manager will automatically merge the users. This merge creates a new user with the same name that is being authenticated, reassigns all content from matching users to the new user, deletes those previous users, and then authenticates with that new user. This invalidates any active sessions using either of the two previous LDAP users.

Note: Any user merging will be accompanied by an entry in the Software Risk Manager server logs, as well as the Visual Log, with appropriate details.

Display Names

The LDAP attribute used for display names can be set with `auth.ldap.user.displayNameAttribute`, which is a comma-separated list of attribute names. Each attribute is checked in order, and the first received value is used. If unassigned, Software Risk Manager will attempt to use their CN, UPN, and SAN, in that order.

Display names are automatically updated when a user signs in. Changes to the `displayNameAttribute` property won't affect users until they sign in again.

Single-value LDAP attributes will be used as-is, if available. Multi-value attributes will have their first entry selected by default. You can choose a specific index in the list, if needed, by using the syntax `myAttribute{n}`, where n is the 0-based index in the attribute list. For example, you can select the second value in a list while falling back to another attribute with the following:

```
auth.ldap.user.displayNameAttribute = myAttribute{1}, someOtherAttribute
```

You could also prioritize a second value while falling back to the first with:

```
auth.ldap.user.displayNameAttribute = myAttribute{1}, myAttribute{0}
```

Note: `myAttribute{0}` and `myAttribute` are equivalent; they both select the first value available from the attribute.

LDAP Group Mapping

LDAP groups for your users can be mapped to Software Risk Manager groups within the application. This requires connection to an LDAP server that provides a user attribute listing its groups, such as Active Directory and its `memberOf` attribute.

This group mapping requires a valid `groupAttribute` and `nameAttribute` to be assigned in your `codedx.props` file. The defaults are suitable for use with an Active Directory server and can be left unassigned if using Active Directory.

The `auth.ldap.user.groupAttribute` is the name of the attribute on your *user objects* that lists the groups that the user is a member of. This is typically a multi-value attribute containing the full DN paths for a user's groups.

The `auth.ldap.group.nameAttribute` is the name of the attribute on your *group objects* that determine how you refer to those groups in Software Risk Manager. When defining what a Software Risk Manager group will map to, you will use the values provided by that attribute. This is used to pull a readable name from the `groupAttribute` of any authenticated users.

SAML Configuration

Software Risk Manager can also be configured to authenticate against a SAML 2.0 IdP (Identity Provider). This can be done by using the settings below.

Note: SAML authentication isn't currently supported for Software Risk Manager plugins.

- `auth.saml2.identityProviderMetadataPath` – [required] Sets the file path to the XML metadata for your SAML IdP (file must exist on the Software Risk Manager host).
- `auth.saml2.keystorePath` [default: internal Software Risk Manager path] – Sets the signing key for SAML requests by Software Risk Manager; must be JKS format with the appropriate cert, private key, and an assigned password for both the JKS key store and private key; will be auto-generated at an internal Software Risk Manager location if unassigned.
- `auth.saml2.keystorePassword` – [required] Sets the password used to open the assigned JKS file.
- `auth.saml2.privateKeyPassword` – [required] Sets the password to use with the private key stored in the assigned JKS file.
- `auth.saml2.uniqueNameAttribute` [default: user's Name Id] – Sets the specific attribute to use as the unique username ID; the value of this attribute is used as the user's login ID and display name, unless the `auth.saml2.displayNameAttribute` setting (see below) is defined.
- `auth.saml2.displayNameAttribute` [default: value of `auth.saml2.uniqueNameAttribute`] – Sets the specific attribute to use as the user's display name; if not set, the value of `auth.saml2.uniqueNameAttribute` (default: user's Name Id) is used instead.
- `auth.saml2.namePolicyFormat` default: from IdP XML – Sets the accepted/required format of the Name Id.
- `auth.saml2.entityId` default: https://codedx.com – Sets the entity ID used to identify Software Risk Manager as a service provider when registering with your IdP.
- `auth.saml2.responseBindingType` [default: `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST`] – Sets the Assertion Consumer Service (ACS) binding type for Software Risk Manager.
- `auth.saml2.authnRequestBindingType` [default: from IdP XML] – Sets the binding type to use when Software Risk Manager submits Authn requests to your IdP (must be supported by your IdP's SSO service endpoint).
- `auth.saml2.maximumAuthenticationLifetime` [default: 3600] – Sets the maximum allowed age (in seconds) of the `AuthnInstant` specified in an Authn Response from the IdP. This is an additional (but not entirely necessary) validation step performed by Software Risk Manager. If rolling session lifetimes are in place from your IdP, this may be disabled by setting to `-1` (which sets the max lifetime to 10 years).
- `auth.saml2.maxSkewSeconds` [default: 120] – Sets the maximum allowable difference in UTC time between the Software Risk Manager server and an IdP server (loosens the window of maximumAuthenticationLifetime).
- `auth.saml2.passive` [default: `false`] – Sets whether or not to use passive authentication against your IdP.
- `auth.saml2.forceAuth` [default: `false`] – Sets whether or not to force re-authentication with your IdP when the user attempts to start a new Software Risk Manager session (this re-authentication may start a new session and invalidate existing SAML sessions with other applications).
- `auth.saml2.requireSignedAssertions` [default: `false`] – Sets whether or not Software Risk Manager should require signed assertions from your IdP (regardless of whether the message as a whole was signed).
- `auth.saml2.requireSignedResponses` [default: `false`] – Sets whether or not Software Risk Manager should require signatures on all responses from your IdP.

- `auth.saml2.signAuthenticationRequests` [default: `false`] – Sets whether or not Software Risk Manager should sign authentication requests to your IdP.
- `auth.saml2.useNameQualifier` [default: `false`] – Sets whether or not Software Risk Manager should include a name qualifier in Authn requests to your IdP.
- `auth.autoExternalRedirect` [default: `true`] – Sets whether or not to automatically redirect unauthenticated users to your IdP login portal.
- `ui.auth.samlLabel` [default: SAML] – Sets the label for SAML-related operations within the Software Risk Manager interface; this will be displayed when signing in to Software Risk Manager or when creating new user.

Supported Profiles

Software Risk Manager adheres only to the Web SSO Profile. (Software Risk Manager also does not make use of RelayState.) Other profiles, including Single Logout (SLO), are unsupported.

Basic Configuration

Upon successful configuration and after navigating to the Software Risk Manager Login page, your `codedx.log` file should contain the message "Successfully configured SAML authentication." Otherwise, it will contain "SAML authentication was either incomplete or missing, skipping." A minimal configuration looks like the following:

```
auth.saml2.identityProviderMetadataPath = C:/idp-metadata.xml
auth.saml2.keystorePassword = jkspassword
auth.saml2.privateKeyPassword = pkpassword
```

The IdP XML should have a root element of `<*:EntitiesDescriptor>`. with `<*:EntityDescriptor>` elements inside. The recommended location for this file is in the Software Risk Manager appdata folder, but this is not in a requirement. Ensure that Software Risk Manager has the necessary permissions to read the XML file.

The keystore is used by Software Risk Manager to sign requests to your IdP. This keystore will be created automatically unless `keystorePath` is assigned and a file exists at the assigned path. The passwords for the keystore and its private key must be set manually through the `keystorePassword` and `privateKeyPassword` properties. The assigned passwords will be used to create the keystore if necessary. The default path for the keystore is in Software Risk Manager appdata directory, at `.../codedx_appdata/keystore/codedx-saml.jks`. If a custom keystore is provided, this folder is the recommended location. Ensure that Software Risk Manager has the necessary permissions to read the keystore file.

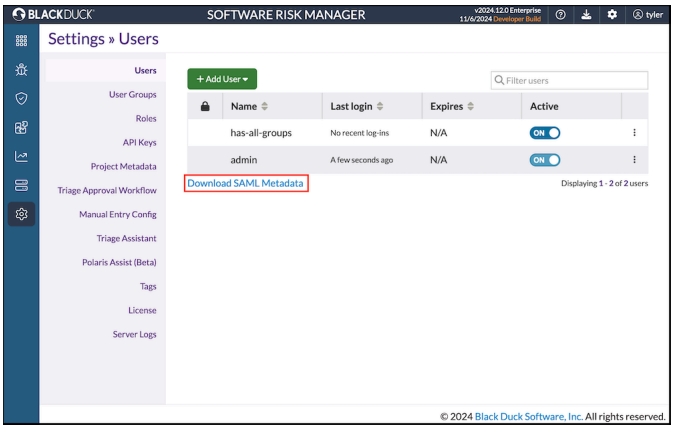
Software Risk Manager has no inherent requirements for the IdP it authenticates against. Requests from Software Risk Manager can be configured as necessary using the properties listed above to meet the needs of your IdP.

Navigating to Software Risk Manager will redirect you to your IdP Login Portal, which you may sign into but will not provide access to Software Risk Manager. You must first add your SAML account as a user recognized by Software Risk Manager. See the *Software Risk Manager User Guide* for more information. By default, Software Risk Manager will redirect you to your IdP, but you can sign in using the Software Risk Manager *Login* page by navigating to `/login?local`. Here you can enter the default Software Risk Manager admin and password to sign in and add yourself and others as SAML-authenticated users.

If you get a generic error page after signing in with SAML, you may need to explicitly assign the Software Risk Manager base path. This should match what you have registered with your IdP. See the section on Multiple Hosts below for more information.

SP Metadata

The full XML Metadata for Software Risk Manager as an SP can be downloaded from the Users page if SAML has been configured (see figure below). Notable details are provided below.



The Software Risk Manager Assertion Consumer Service (ACS) endpoint is at `/login/callback/saml`, relative to your Software Risk Manager URL. For example, if hosting at `https://localhost/srm`, the ACS will be `https://localhost/srm/login/callback/saml`.

The metadata XML will list the ACS binding type as `HTTP-POST`, regardless of the binding type set in `codedx.props`. This can be ignored. Software Risk Manager will use the binding type specified in `codedx.props`. You can change the binding type in that XML if necessary.

Callback URL, Proxies, and Multiple Hosts

Software Risk Manager can be accessed through a variety of hosts (such as machine name, FQDN, IP address, reverse proxy), but only one will be recognized by Software Risk Manager as its ACS. The Software Risk Manager ACS is built dynamically when the first request is made, and it is based on the first URL requested from Software Risk Manager. If you access Software Risk Manager through `https://192.168.1.10/srm`, the ACS URL will be at `https://192.168.1.10/srm/login/callback/saml` and Software Risk Manager will only respond to SAML requests to that endpoint. SP metadata will also be based on this URL.

In this case, SAML authentication will fail unless your IdP has the Software Risk Manager ACS at the `https://192.168.1.10/...` endpoint. The downloaded SP metadata will have your SAML IdP redirect using the Software Risk Manager IP, when you may want it to point to a reverse proxy instead.

You can assign the `auth.hostBasePath` property in your `codedx.props` file to force a consistent ACS URL. This affects the endpoint that Software Risk Manager will listen to, along with the SP metadata that it generates. If Software Risk Manager is accessible at `https://myhost/srm`, you can add the following:

```
auth.hostBasePath = https://myhost/srm
```

This will force a consistent ACS URL to `https://myhost/srm`. (**Note:** This only affects authentication; it will not change the base path of Software Risk Manager itself.)

Single Log-Out and Ensuring Active Sessions

Software Risk Manager currently does not support any of the capabilities specified in the SAML Single Logout (SLO) Profile.

If a user signs out from their SAML session, this logout will not automatically propagate to Software Risk Manager.

To prevent the use of expired SAML sessions, we strongly recommend configuring Session Lifetimes within Software Risk Manager to ensure the SAML session is validated frequently.

Automatic SAML Authentication

Once SAML is configured within Software Risk Manager, all login requests will be redirected to your IdP authentication portal. Authenticated users will be redirected back to Software Risk Manager, where they can view the *Projects* page if they are registered with Software Risk Manager. Otherwise, they will be redirected to the Software Risk Manager *Login* page, where they can log in as a local user or retry authentication with your IdP. Software Risk Manager does not perform any single sign-out; signing in with a different SAML user will require signing out through your IdP portal and re-authenticating with the new user information.

You can disable the automatic redirect to your IdP by setting `auth.autoExternalRedirect = false` in your `codedx.props`.

Multiple Software Risk Manager Deployments with the Same IdP

Authenticating multiple Software Risk Manager instances against the same IdP requires setting a unique Service Provider Entity ID for each deployment. Assign the `auth.saml.entityId` property to different values for each deployment, and register each new Entity ID with your IdP using the appropriate ACS for the associated deployment.

For example, consider the following: Two Software Risk Manager deployments are hosted at `https://foo.com/srm` and `https://bar.com/srm`. Modify `auth.saml.entityId` for both; in this case they are assigned to `https://codedx-foo.com` and `https://codedx-bar.com`, respectively. Software Risk Manager places no restrictions on Entity IDs; you can assign them as appropriate for your case.

Restart both Software Risk Manager deployments and register them with your IdP as follows:

- Register entity `https://codedx-foo.com` with its ACS `https://foo.com/srm/login/callback/saml`
- Register entity `https://codedx-bar.com` with its ACS `https://bar.com/srm/login/callback/saml`

Both deployments will authenticate against your SAML provider. Note that users registered with one Software Risk Manager deployment will not be registered with another, even if they authenticate against the same IdP. Accessing both deployments with SAML user "John Doe" will require registering them with each Software Risk Manager deployment individually.

While multiple Software Risk Manager installations are supported, it is discouraged to run them on the same machine. This will likely lead to resource contention, resulting in performance degradation in those installations.

Forcing Local Sign-in

Software Risk Manager does not validate that the configured IdP is available before attempting to authenticate. If your SAML IdP becomes unavailable or you need to sign in through a local account, there are two options:

1. Set `auth.autoExternalRedirect = false` in your `codedx.props`. Unauthenticated users will be redirected to the Software Risk Manager *Login* page instead of your IdP portal, where they can sign in with a local account or attempt to sign in through your IdP portal through a link below the sign in form.
2. Manually navigate to `/login?local`, relative to your Software Risk Manager hosting path. This will force the Software Risk Manager *Login* page to display regardless of the value of `auth.autoExternalRedirect`.

Note: You may not be able to sign in as a SAML user while your IdP is offline, even if you had been previously authenticated.

IdP-Specific SAML Instructions

Identity Providers with known special considerations are discussed below. Presence (or non-presence) of a particular IdP does not reflect which IdPs are "supported" or "preferred"; the list of IdPs and known issues may not be comprehensive.

Microsoft Entra ID (formerly Azure AD)

Software Risk Manager can use Microsoft Entra ID for SAML authentication by registering Software Risk Manager as an "Enterprise application." To do this, click "Create your own application," enter a name, and then select "Non-gallery" from the available options.

Once created, open the newly created application, select "Single sign-on," and then select "SAML."

Only two properties must be specified: the app's Identifier and Reply URL. The Identifier should be `https://codedx.com`, unless `auth.saml2.entityId` was changed to a different value in the properties file. The Reply URL should be the Software Risk Manager ACS URL as described in the SP Metadata section.

Note: Software Risk Manager does not support Relay State or Single Logout. Sign-on URL may be blank or set to the `/srm/login` page.

After editing "Basic SAML Configuration" as described, save your changes and confirm that the Identifier field was set with the expected value.

There are two locations to confirm: Within the "Single sign-on" page for your Enterprise registration, and the "Expose an API" page in the associated App registration.

The Identifier from the "Enterprise registration -> Single sign-on" page should match the "Application ID URI" from the "App registration -> Expose an API" page; both should match the entity ID for Software Risk Manager.

Important Properties Settings

Configuration of Software Risk Manager for Entra ID only requires the basic mandatory settings: `auth.saml2.identityProviderMetadataPath`, `auth.saml2.keystorePassword`, and `auth.saml2.privateKeyPassword`.

The `maximumAuthenticationLifetime` must also be configured properly for a reliable integration. The section below discusses this in more detail.

Session Lifetime Considerations

Integrations with Entra ID must pay attention to the maximum lifetime of a session. At the time of writing, Entra ID's default lifetime is a "rolling window" of 90 days. Typically the 90 days would be converted to seconds and used as the value for the Software Risk Manager property `auth.saml2.maximumAuthenticationLifetime`.

Synchronizing this setting between Software Risk Manager and Entra ID (or any IdP) is important: If Software Risk Manager receives an authentication response containing a session older than the maximum allowed lifetime, the response will be rejected with an error message.

If a policy with a rolling window is applied, there is effectively no single "maximum" age Software Risk Manager can expect. In this case, we recommend disabling the sliding window within Entra ID, or disabling the Software Risk Manager SAML session lifetime limit by setting `auth.saml2.maximumAuthenticationLifetime` to `-1`.

The original 90-day value (`7776000` seconds) may be used regardless of the rolling window. In this case, users attempting to authenticate with Software Risk Manager must manually start a new session with their IdP once their session exceeds that age.

Behavior in Entra ID may be viewed or customized with the "Conditional Access" policies available.

SAML Group Mapping

Software Risk Manager can detect SAML group information during authentication, which can then be used to auto-create SAML users and enable mapping between SAML and SRM groups to automatically add/remove users from SRM groups.

- The `auth.saml2.groupMappingMethod` option in `codedx.props` specifies which method to use. For more information, see the Single SAML Attribute and Multiple SAML Attributes sections below.
- The `auth.saml2.groupNameMatcher` property lets you define a regular expression with a required capture group to extract group names from SAML attributes. Only the part of the attribute matched by the capture group is used as the group name. If the pattern is invalid or does not include a capture group, authentication will fail.
  This is particularly useful when your group names are formatted like LDAP DNs. For example: if your group name is

  ```
  cn=TESTGROUP1,ou=groups,o=org
  ```

  and you want to authenticate based on whether the user is part of the `TESTGROUP1` group, then you may set

  ```
  auth.saml2.groupNameMatcher = cn=(\w+),ou=groups,o=org
  ```

  to successfully authenticate such users.

Software Risk Manager can read group information for a SAML user through either of the following methods:

- **Method 1:** A single SAML attribute containing a list of user groups (`string-list`)
- **Method 2:** Multiple SAML attributes, each indicating membership of one user group (`multi-attr`)

Selecting the appropriate mapping method will depend on your SAML IdP and how it is configured to provide user group information. (These methods cannot be used together at the same time.)

Note: SRM does not support the Assertion Query/Request SAML profile and can only access user SAML groups during authentication. Enabling Session Lifetimes is recommended to ensure users are regularly signed back in and updated (see Session Expiration).

Method 1: Single SAML Attribute

For single SAML attribute mapping, add the following to the properties file: `auth.saml2.groupMappingMethod = string-list`. This method also requires setting the `auth.saml2.groupAttribute` property to the name of the SAML attribute containing the list of user groups (this attribute name is case-sensitive).

This attribute will be interpreted as a comma-separated string, where each entry is a group that the user is a member of.

This method supports both single-value and multi-value attributes. For multi-value attributes, each value is interpreted as an additional comma-separated list and is combined with the others for the final list of user groups.

Method 2: Multiple SAML Attributes

For multiple SAML attribute mapping, add the following to the properties file: `auth.saml2.groupMappingMethod = multi-attr`. This method also requires adding at least one

prop, beginning with `auth.saml2.groupAttributePattern`. (The prop `groupAttributePattern` acts as a template for extracting groups from a user's list of attributes.)

The value of `groupAttributePattern` should match the format of the relevant attribute names and use `{group}` as a placeholder for the part containing the group name. For example, given an attribute named `is-secops-group`, you would use the pattern `is-{group}-group`.

You can handle multiple patterns by adding additional `groupAttributePattern` props, such as in the following example:

```
auth.saml2.groupAttributePattern = is-{group}-group
auth.saml2.groupAttributePattern-other = member-of-{group}
```

Note: A pattern value must contain exactly one occurrence of the text `{group}`.)

All patterns described by props starting with `auth.saml2.groupAttributePattern` will be used together to detect the list of user groups. (This is unaffected by whether one pattern detects a match or all patterns have a match—if any pattern matches an attribute name, it will be recognized as a group name, regardless of whether other patterns match that attribute.)

With `groupMappingMethod = multi-attr` and `groupAttributePattern*` set, a user will be considered a member of any group that matches the specified pattern.

You can also have Software Risk Manager check the value of the attribute using `auth.saml2.membershipAttributeValues` and/or `auth.saml2.nonMembershipGroupAttributeValues`. These options can be set to comma-separated lists of values that are used to determine membership/non-membership:

- If `membershipAttributeValues` is set, a user will only be considered a member of a group if the attribute contains one of the values assigned to the prop. Any other value will be interpreted as “non-membership.”
- If `nonMembershipAttributeValues` is set, a user will be considered as not a member of the group if the attribute contains one of the assigned values. If not set, or if the attribute value does not match, group membership is determined by `membershipAttributeValues` behavior.

Account Lockout

Software Risk Manager will automatically lock user accounts that are attempting to log in too many times. When a user's account is locked, they can no longer log in until the account is unlocked. An account will either unlock after the set lockout duration has elapsed, the password is reset, or when it was manually unlocked by an admin. Admins may view locked accounts and unlock them on the users page.

Note that SRM only locks local user accounts. Other authentication schemes such as LDAP, SSO, etc., are not subject to these rules and may have rules of their own.

All of these parameters for lockout may be customized via the following props:

- `auth.lockout.enabled` – [default: true] Determines if the account lockout feature is enabled or not. Note that setting this to `false` will not unlock already locked accounts
- `auth.lockout.timespan` – [default: 15 minutes] The time period in which the log in attempts must occur
- `auth.lockout.attempts` – [default: 5] The number of failed attempts that must occur before the account is locked
- `auth.lockout.duration` – [default: 1 hour] How long the account will be locked for. After the lockout duration, a user can attempt to log in again. This value may be set to `0` if an indefinite lockout is desired

Password Policy

Software Risk Manager can be configured to have new/edited local user passwords meet certain requirements.

The defaults for SRM's password requirements are listed below; however, requirements can be specified by setting the appropriate properties:

- `local-password-policy.minimum-length = 12` - Require a minimum length [default: 12].
- `local-password-policy.maximum-length = 1048576` - Require a maximum length [default: 1048576].
- `local-password-policy.contains-lowercase = false` - Require a lowercase character [default: true].
- `local-password-policy.contains-uppercase = false` - Require an uppercase character [default: true].
- `local-password-policy.contains-number = false` - Require a number [default: true].
- `local-password-policy.contains-special-character = false` - Require a special character (e.g. $! # %) [default: true].
- `local-password-policy.common-check.enabled = true` - Require passwords to be distinct from an internal set of known compromised passwords [default: true].
- `local-password-policy.unique-password-check.enabled = true` - Require passwords to be distinct from the last *n* previously used passwords, where *n* is configurable [default: true].
  - `local-password-policy.unique-password-check.num-to-check = 10` - The last *n* previous passwords to prohibit a user from setting as their password [default: 10].
- `local-password-policy.reset-on-first-login = true` - Require that users set a new password when logging in for the first time [default: true].
- `local-password-policy.reset-after-admin-sets-password = true` - Require that users set a new password when an admin resets their password [default: true].
- `local-password-policy.max-password-age = 12 months` - Require that users set a new password after a set amount of time since it was last reset. This can be disabled by setting a value of 0 [default: 12 months].

Header-Based Authentication

Software Risk Manager allows you to authenticate users via a request header. Requests to the Software Risk Manager server which provide the configured header with a username as its value will be treated as "logged in" as that user. You can optionally restrict the IP addresses from which such requests can originate. For example, if you put Software Risk Manager behind a proxy server that manages its own authentication and adds the header only to authenticated requests, you would specify the proxy server's IP address as the only "allowed" IP address.

The name of the request header and the set of allowed IP addresses are configured by the following keys in the `codedx.props` file:

```
codedx.header-authentication.header = My-Magic-Authentication-Header
codedx.header-authentication.allowed-ips = 172.1.1.1, 127.0.0.1, 0:0:0:0:0:0:0:1
```

There is no default value for `header` key. If omitted, header-based authentication will be disabled. If the `allowed-ips` key is not specified, all IP addresses will be considered "allowed." This will also cause a warning in your log, as it is not recommended to configure a `header` without an `allowed-ips` whitelist.

Once configured, header-based authentication and username/password logins will be mutually-exclusive; you cannot log in via header while already logged in via username/password, and vice versa. There is no "log out" functionality for a header-based session. Instead, you need to stop sending the header with your requests.

Header-based authentication may be used to log in as *any* enabled user that has been added to Software Risk Manager. This includes the super-admin user, all local users, and all LDAP users. Because of this, it is highly recommended to provide an `allowed-ips` whitelist and hide the Software Risk Manager server behind a proxy that can manage the header.

External User Auto-Registration

Software Risk Manager supports auto-registration of users that sign in through external methods such as LDAP and SAML. This can include automatic creation, enabling, and disabling of users based on their properties. This is particularly useful when managing an installation using third-party authentication, where there may be many users in your organization that need to be maintained.

Note: User auto-registration respects the user limit of your license; if a new user attempts to sign in without any seats available, they will receive an error message regarding license limitations.

Auto-Registration with LDAP

Auto-registration can be configured in your `codedx.props` file with the following settings:

- `auth.auto-create.ldap.enabled` – set whether to auto-create LDAP users that don't exist yet in Software Risk Manager [default: false]. (This setting was previously exposed as `auth.auto-create.enabled` which will be deprecated in an upcoming release.)
- `auth.auto-create.ldap.group-names = foo, bar` – a comma-separated list of group names, where an LDAP user must be a member of at least one of those groups to allow auto-creation [default: none].
- `auth.auto-create.ldap.auto-toggle-enabled` – set whether to automatically enable/disable LDAP registered users based on their membership in the provided LDAP group names (no effect if `group-names` is unassigned or empty) [default: false].

When using auto-registration with LDAP, we recommend using both the `group-names` and `auto-toggle-enabled` options. Without `group-names`, any LDAP user can sign in to Software Risk Manager. Without `auto-toggle-enabled`, a user leaving one of those required groups would still be allowed to sign in to Software Risk Manager.

Note: LDAP group membership checks are only available for LDAP providers that maintain an attribute on users listing their memberships (e.g., Active Directory and its `memberOf` attribute).

Auto-registration with LDAP requires a valid LDAP configuration in your `codedx.props` file, and Software Risk Manager must have permission to read the necessary LDAP objects and attributes to check for group membership. This depends on your LDAP server configuration, but typically requires that you've assigned a `systemUser` and `systemPassword` and set `authenticationMechanism = simple` in your `codedx.props` file.

Note: Automatic enable/disable of users occurs when logging in. Changing the user's LDAP group membership will not sign the user out or disable them. If necessary, you can disable or delete the user manually. Configuring session durations in Software Risk Manager may also be useful.

Also note that LDAP group names in the `group-names` property does not need to match any LDAP group mappings in your Software Risk Manager user groups, though there may be some overlap.

Auto-Registration with SAML

Auto-registration can be configured in your `codedx.props` file with the following settings:

- `auth.auto-create.saml.enabled` – set whether or not to automatically create SRM users when a SAML user signs in (optionally gated by `group-names`) [default: false].
- `auth.auto-create.saml.group-names` – a comma-separated list of group names used as a requirement on auto-create and auto-toggle behavior [default: none].
- `auth.auto-create.saml.auto-toggle-enabled` – set whether or not to automatically enable/disable SRM SAML users depending on their group membership (requires `group-names` be set) [default: false].

When using auto-registration with SAML, using both the `group-names` and `auto-toggle-enabled` options is recommended.

Auto-registration with SAML requires a valid SAML and SAML Groups configuration in your codedx.props file.

Cookie Config

For SRM 2023.12.5 and later, the prop `auth.cookie.secure` is available and gives control for setting the secure flag on the session cookie used by SRM. Native HTTP installs will have this prop set to `false` and HTTPS installs will have this set to `true`. If not using a native install, and HTTPS is being used between the client and SRM, this attribute will need to be manually set to `true` in the props file.

- `auth.cookie.secure` - [default: false] When SRM is communicating over HTTPS, the secure flag on the session cookie will always be set regardless of the value of this prop. When SRM is communicating over HTTP, the value of this prop controls if the session cookie has this flag set. Where `true` means the secure flag is set.

Session Expiration

You can set user sessions to expire after a period of inactivity using the settings below:

- `session.lifetime` - sets the duration of inactivity required before automatically signing out a user; if set to 0, sessions will never expire [default: 20 minutes]
- `session.timeout-notice` - sets the point at which the user will be notified when their session will expire [default: 2 minutes]

Both of these properties use a readable duration format, that is, `1 hour`, `1 hour and 30 minutes`, `3 days`, and so on.

Enabling session expiration requires assignment of `session.lifetime`, but `session.timeout-notice` is optional. If left unassigned, the timeout notice will be the shortest of either two minutes or 25% of the session expiration period.

If session expiration is enabled, the user's session will be closed if it either times out or if the user closes their browser.

Note: If combining session expiration with SAML authentication, make sure to set `auth.saml2.forceAuth = true` so that the user always has to re-authenticate with your IdP.

Sessions can also be invalidated if SRM is configured to only allow one user session at a time with the following setting:

- `srm.limit-sessions.enabled` - [default: false] Controls if user sessions should be limited to one at a time

EULA Acceptance

Normally after the initial installation of Software Risk Manager, the end-user license agreement (EULA) will be presented to the first admin user to log in. You can configure Software Risk Manager to skip this step by accepting the EULA via the props file:

```
codedx.eula-accepted = true
```

Data Retention Configuration

To help manage data retention, Software Risk Manager offers the following settings to enable the purging of old data:

- `findings.purge-after-days` [default: unspecified] - if specified, sets the number of days to retain *Findings* marked as *Gone*. For example, `findings.purge-after-days = 90`.
- `results.purge-after-days` [default: unspecified] - if specified, sets the number of days to retain archived *Results*. For example, `results.purge-after-days = 7`.
- `visual-log.purge-after-days` [default: 30] - if specified, sets the number of days to retain *Visual Log* entries. For example, `visual-log.purge-after-days = 45`. Setting the value to -1 will cause all visual log entries to be retained.
- `analysis.logs.cleanup-mode` [default: expired] - sets what strategy is used to clean up analysis logs. Valid values are:
  - `expired` - delete analysis logs older than `analysis.logs.days-to-keep`.
  - `archived` - delete logs for analyses where all of its inputs are archived.
  - `expired-or-archived` - delete logs for analyses that are either older than `analysis.logs.days-to-keep`, or have all of their inputs archived.
  - `expired-and-archived` - delete logs for analyses that are both older than `analysis.logs.days-to-keep`, and have all of their inputs archived.
  - `disabled` - analysis logs are not deleted.
- `analysis.logs.days-to-keep` [default: 7] - sets the number of days to retain analysis logs if the `analysis.logs.cleanup-mode` is set to `expired`, `expired-or-archived`, or `expired-and-archived`.
- `analysis.logs.cleanup-time` [default: 00:00] - sets the time of day that analysis log clean up is ran. A value for this property should be a 24-hour time in the form `HH:MM`. For example `01:00` for 1 in the morning, or `13:00` for 1 in the afternoon.

Findings

When enabled, findings that are marked as *Gone* and have not been modified for more than the specified number of days will be removed. For a finding to be eligible for removal, it must be marked as *Gone* and not have been modified recently on *all* Branches that it is present on. If the finding has been modified recently or has a status other than *Gone* on any branch, then the finding will be kept on all branches.

Results

When enabled, results coming from *Analysis Inputs* that have been archived and are older than the specified number of days will be removed. However, results that are associated with findings that are marked as *Gone* will not be removed. This is done to prevent a scenario where a *Gone* finding has no associated results (since all results associated with a *Gone* finding will be archived). Any findings removed per the data retention configuration above do not count toward this condition. Results will be removed on any branch where these conditions are met, regardless of their status in other branches.

Visual Log

When enabled, *Visual Log* entries that are older than the specified number of days will be removed.

Git Related Configuration

Software Risk Manager allows you to configure each project to automatically use source from a git repository as input for each analysis. When configuring a connection to a git repository, Software Risk Manager will, by default, disallow the usage of “local” URLs (i.e., URLs that point to a file in the Software Risk Manager file system). This is enforced as a security measure to prevent system information exposure via the validation user interface. Although it is strongly recommended that this setting be left disabled, in the exceptional cases where it is necessary to use local git repositories, set the `git.config.allow-local-urls` property to `true`.

When configuring a git repository connection, Software Risk Manager uses a request timeout of 60 seconds by default. This timeout can be changed by setting a value for the `git.config.timeout` property. For example, a value of `2 minutes` changes the timeout to 2 minutes.

Job Configuration

Software Risk Manager performs most long-running tasks, including background cleanup tasks, on a job system. Controls are in place to limit the number of such tasks running concurrently to ensure the system isn't overloaded. The resource limits listed below may be adjusted. Higher numbers translate to more available resources of that type. The minimum value allowed is 1000, and the default is 2000. Values for this setting are unitless; they are relative measures of the power of your Software Risk Manager server. Certain jobs are more CPU-intensive, while others are more database-intensive. Each running job uses a certain amount of the "limit," and this is the mechanism for limiting job concurrency.

These values should be adjusted in relation to the default value. For example, if you believe your server is twice as powerful as an average server, set these settings to 4000 (double the default) to increase the number of concurrent jobs that may be run.

- `swa.jobs.cpu-limit` determines the maximum available CPU
- `swa.jobs.memory-limit` determines the maximum available memory
- `swa.jobs.database-limit` determines the maximum available database I/O
- `swa.jobs.disk-limit` determines the maximum available disk I/O

In addition, certain jobs (e.g., generating reports for a project) may produce outputs. The `swa.jobs.expiration` (default: `60`) setting specifies how long the job results will be available (in minutes).

Analysis Behavior

Various settings allow you to affect Software Risk Manager behavior regarding the analyses it conducts. Changing any of the analysis behavior properties can be done at any time after the initial installation; however, you will still need to restart the Tomcat server to reload the properties.

- `analysis.concurrent-analysis-limit` (default: 2) - the maximum number of analyses to run concurrently. Note that this does not override the *Job Configuration* settings, but merely sets another limit.
- `storage.keep-raw-inputs` (default: `true`) - when this setting is true, Software Risk Manager will keep copies of all files uploaded for analysis. While these inputs are not required by Software Risk Manager after the analysis is complete, keeping them for archival purposes will allow them to be downloaded from the *Show Inputs* list. If storage space is an issue, setting this to `false` will prevent Software Risk Manager from storing the raw inputs.
- `storage.keep-archived-inputs` (default: `false`) - when this setting is true, Software Risk Manager will keep all uploaded files, whether they are archived or not. Leaving this setting `false` will cause Software Risk Manager to delete stored copies of data upon archival. Note that this setting is dependent on the value of `storage.keep-raw-inputs`, which must also be `true` in order to keep the archived data.
- `swa.tools.keep-all-logs` (default: `false`) - this setting determines whether to keep all the log files for the tools that Software Risk Manager runs. If false, only the logs from failures are kept.
- `swa.upload.maxfilesize` (default: `2048`) - this setting controls the maximum file upload size allowed (in megabytes) for a single file.
- `swa.upload.maxuploadsize` (default: `2048`) - this setting controls the maximum size of all uploaded files (in megabytes) for a single analysis.
- `codedx.analysis.hybrid-enabled-by-default` (default: `false`) - this setting determines the default value for the "Enable hybrid analysis" setting in *Analysis Config*. Since hybrid analysis requires some relatively time-consuming steps, this setting is `false` by default.
- `codedx.analysis.auto-archive-enabled-by-default` (default: `true`) - this setting determines the default value for the "Archive findings" setting in *Analysis Config*. Most projects will only upload a scan from a tool if it represents a new state of the project, but an organization may want to be able to upload many tool result files in sequence without having them interfere with each other. Such organizations should set this setting to `false`.
- `codedx.analysis.auto-archive.excluded-tools` (default: none) - this setting allows excluding outputs from certain tools from the auto archival process. This setting is a comma-separated list of the names of the tools to exempt from auto archival.
- `ingestion.skip-code-metrics` (default: false) - if set to true, code metrics will not be gathered during analysis.
- `finding.reopen-gone` (default: true) - this setting determines the default value for the "Allow gone findings to be reopened" setting in *Analysis Config*.
- `finding.reopen-resolved` (default: false) - this setting determines the default value for the "Reopen resolved findings when updated" setting in *Analysis Config*.
- `storage.keep-failed-analysis-inputs` (default: false) - this setting determines whether to keep files associated with failed analyses.
- `analysis-prep.idle-lifetime` (default: 30 minutes) - this setting controls how long an analysis prep must be inactive before it expires and its prep id becomes invalid. Note that if uploading to an analysis prep **via the Software Risk Manager API**, the analysis prep is still considered inactive until the upload finishes. It is possible for the analysis prep to expire before the upload finishes. If you're encountering an error from the API stating the prep doesn't exist, it is recommended to increase the default of this prop. A value for this property should follow the form of `{number}{unit}`, for example, `50m` for 50 minutes or `1h` for 1 hour.

- `srm.correlation.default-component-correlation-mode` (default: `vulnerability,identifier,type`) - this setting configures what the default component correlation mode will be for newly created projects in SRM. Valid values are
  - `vulnerability,name&version,type`
  - `vulnerability,identifier,type`
  - `identifier,type`
  - `name&version,type`
  - `vulnerability,type`
- `code-metrics.keep-all-logs` (default: false) - this setting determines whether to keep all the log files for the code metrics tools that Software Risk Manager runs. If false, only the logs from failures are kept.

Bundled Tools

Software Risk Manager bundles various tools that run independently during the analysis process. Each of these tools requires a memory budget during its own analysis. The memory requirements vary based on the sizes of the codebases the analyzers are checking. The memory budget for each of these tools is configurable in the properties file; each of the following settings specify the number of megabytes allotted to their respective tools. In general, the static analyzers will require more memory in order to analyze larger projects.

- `java.tools.maxmemory` (default: `1024` (1GB)) determines the maximum heap size for java-based tools.
- `java.tools.maxmemorypercentage` (default: <none>) determines maximum heap size for java-based tools as a percentage of total system memory.
- `ruby.tools.maxmemory` (default: `1024` (1GB)) determines the maximum heap size for Ruby-based tools, which are run with Java via JRuby.
- `ruby.tools.maxmemorypercentage` (default: <none>) determines the maximum heap size for Ruby-based tools as a percentage of total system memory, which are run with Java via JRuby.
- `php.tools.maxmemory` (default: `1024` (1GB)) determines the maximum heap size for PHP tools.

Additionally, these bundled tools may be disabled entirely, if necessary, by setting `bundled-tools.disable` to `true` (default: `false`). When this flag is set, no bundled tools will be available on the new analysis page nor will they be run.

Report Configuration Templates

Several report configuration options accept templates for their values. These templates may contain plain text and various substitutions, surrounded by `{{` and `}}`. To include a `{` `}` or `\`, prefix them with a backslash (e.g., {).

The available substitutions are as follows:

- `{{report.name}}` or `{{report.type}}` - the type of report, e.g., 'PDF.'
- `{{report.title}}` - the title of the report, e.g., 'Assessment Report' or 'HIPAA Report.'
- `{{project.id}}` and `{{project.name}}` - the ID or name of the project the report was generated for.
- `{{user.name}}` - the display name of the user who generated the report.

- `{{date:pattern}}` - the current date/time formatted with the given pattern; details on acceptable patterns may be found here.

Software Risk Manager users may also include project metadata values as well:

- `{{meta:foo}}` or `{{metadata:foo}}` - provides the contents of the metadata field named "foo" for the project (or a blank string if that value is not specified).
- `{{meta.id:bar}}` or `{{metadata.id:bar}}` - provides the ID of the chosen metadata value for the field named "bar" (or a blank string if one was not given).

The following modifiers may be applied to any value:

- `ltruncate:length` - will take up to the last `length` characters (truncated from the left of the string).
- `truncate:length` or `rtruncate:length` - will take up to the first `length` characters (truncated from the right of the string).
- `labbreviate:length` - will take up to the last `length` characters, inserting an ellipsis on the left if the string was truncated.
- `mabbreviate:length` or `cabbreviate:length` - will take up to `length` characters from the string, inserting an ellipsis in the center of the string, if necessary.
- `abbreviate:length` or `rabbreviate:length` - will take up to the first `length` characters, inserting an ellipsis on the right if the string was truncated.

For example, `{{project.name|abbreviate:8}}` will limit the length of the project name to the first 5 characters followed by an ellipse. A project name of "My Project" would be truncated to "My Pr..." for display purposes.

Report Filename

The template used to generate filenames for reports may be customized. This can be done by setting the `report.filename-template` property. See the previous section for details on valid values.

The default template is `{{project.name}} ({{date:dd MMM YYYY}})` - which will produce filenames such as "My Project (27 Mar 2017).xml."

Custom Logo

Users can add their company logo to a PDF report through a prop setting in `codedx.props`. The custom logo will appear on the cover of the PDF report. You can place the logo in the same folder as the properties file or specify an absolute path. For best results, the specified image should be at least 432 pixels wide and/or 144 pixels tall.

For an image file named "logo.png" placed in the same folder as the properties file, the setting in `codedx.props` would be as follows:

```
report.pdf.custom-logo = logo.png
```

Finding Search

On the *Findings Page*, the *Search* area lets you filter findings based on metadata fields from their results, specifically, "Evidence" on the *Details Page*. The fields presented in the *Search* dropdown are the intersection of the set of fields present in that project and the set of fields present in a "for display" whitelist. The "for display" whitelist can be augmented via configuration:

- `additional-values.extra-for-display` [default: N/A] - a comma-separated list of result metadata fields that you want to appear in the *Search* dropdown. Leading and trailing spaces around

each entry in the list will be trimmed. For example, `Field 1, Field 2 , Field 3` will be interpreted as a 3-item list; `Field 1`, `Field 2`, and `Field 3`.

Note: When searching based on a result metadata field, you must enter the full case-sensitive value of the metadata to get a match. For example, searching by `Field 1` for `abc` will not match a `Field 1` with a value of `abc123`or `ABC`.

Tool Configuration

Props Settings

Software Risk Manager has tool-specific settings for some tools, which control how Software Risk Manager will behave in response to data from those tools.

Veracode

When accessing Veracode via a Tool Connector, it is possible to load "Callstack" information for each Veracode Flaw. (Veracode Callstack information is interpreted as Software Risk Manager Data Flow information for the corresponding results.) Doing so will improve the decision-making process of agentless hybrid correlation, but this comes at a steep performance cost (roughly 1-2 seconds per flaw).

- `veracode.callstack-load-enabled` enables loading of the optional "Callstack" information when set to `true`. Defaults to `false`.

NowSecure

NowSecure uses `lab-api` as the default subdomain for the lab api and `app` as the default subdomain for the user interface. Software Risk Manager allows user-defined values for these subdomains in the props file.

- `nowsecure.lab-api-subdomain` defines the subdomain for the lab api. Defaults to `lab-api`.
- `nowsecure.lab-ui-subdomain` defines the subdomain for the lab user interface. Defaults to `app`.

Black Duck

When automatically importing projects with parent mapping from Black Duck, Software Risk Manager omits the top-level Black Duck project group.

- `blackduck.project-enumeration.include-top-level-group` enables including the top-level Black Duck project group in SRM parent project mapping when set to `true`. Defaults to `false`.

Clang-Tidy

By default, the Clang-Tidy reader parses a certain portion of the input file for format detection. The relevant properties can be changed according to preferences in the props file:

- `clang-tidy.format-detection.max-line-length` sets the maximum number of characters to read per line of the input file. Defaults to `8192` characters.
- `clang-tidy.format-detection.num-lines` sets the number of lines to read from the input file. Defaults to `1500` lines.
- `clang-tidy.format-detection.patience` sets the number of lines to scan before running into a relevant line for format detection. Defaults to `15` lines.

Clippy

Clippy is a tool for linting Rust code, typically through the Cargo package manager. Software Risk Manager can run Clippy on Rust code, if available.

When Software Risk Manager is properly configured to use Clippy, it will be offered as a bundled tool when analyzing a ZIP containing at least one `Cargo.toml` file. During analysis, Clippy will be invoked for each `Cargo.toml` file discovered in the ZIP.

In order for Software Risk Manager to run Clippy:

- The standard Rustup installer must be used and installed in a manner accessible to Software Risk Manager.
- Clippy must be installed as a Cargo component. Clippy is typically installed as part of Rustup installation, but it can also be installed manually, if needed.
- Software Risk Manager must be able to discover the installation paths.
- Any additional dependencies needed for building your Rust projects must be pre-installed on the Software Risk Manager machine.

The procedure varies by platform as described in the sections below.

Clippy on Linux

These steps should be followed only after Software Risk Manager has been installed.

Note: Rustup must be installed for the user which runs Software Risk Manager.

- *For non-root installations,* this will be the user that was logged in when Software Risk Manager was installed. The Rustup installer can be run by that user and installed as usual.
- *For root installations,* this will be the `tomcat` user created automatically by the Software Risk Manager installer. The Rustup installer must be run through a `tomcat` user session. For example, on Ubuntu, this may be done with `sudo -u tomcat bash` to start a new shell as the `tomcat` user, followed by the Rustup install command.

This should provide a `.cargo` folder in the user's home directory that contains the `bin/cargo` executable required by Software Risk Manager.

Restart Software Risk Manager and Clippy should be offered as a bundled tool during analysis of relevant ZIP files.

Clippy on Windows

These steps can be followed before or after Software Risk Manager has been installed.

On Windows, Software Risk Manager runs through the `Local Service` user, which does not have a home directory. Instead, these steps will be followed:

- Install Rustup for at least one user on the Software Risk Manager machine.
- Give the `Local Service` user permission to read, write, and execute content in the `.cargo` and `.rustup` folders in the user's home directory.
- Update `codedx.props` to provide the path to the `.cargo` and `.rustup` folders.

Log in to the Software Risk Manager machine and run the standard Rustup installer as that user. Once installation has completed, navigate to the user's home directory (Win+R, type `%USERPROFILE%`, press enter) and find the `.cargo` and `.rustup` folders.

For each folder, do the following:

1. Right-click and select Properties.
2. Go to the Security tab.
3. Click the "Edit" button.
4. Click the "Add" button.

5. In the textbox at the bottom, enter the text "Local Service."
6. Click "OK" to close the "Select User or Groups" modal. The "Local Service" user should now appear in the list of users.
7. Select the user and enable all permissions except "Full control" and "Special permissions."
8. Click "OK" to close the "Permissions for Folder" window.
9. Click "OK" to close the "Folder Properties" window.

With these changes, Software Risk Manager will have access to the `.cargo` and `.rustup` folders.

Next, open the Software Risk Manager Properties file and add these entries:

```
cargo.path = C:/Users/<username>/.cargo
rustup.path = C:/Users/<username>/.rustup
```

... where `<username>` will be replaced with the name of the user that installed Rustup.

Restart Software Risk Manager, and Clippy should be offered as a bundled tool during analysis of relevant ZIP files.

Coverity

By default, a finding's severity in Software Risk Manager is based on the Coverity issue's severity. "Dismissed" and "Absent Dismissed" issues are also excluded by default from the findings. The relevant properties can be changed according to preferences in the props file:

- `coverity.finding-severity-from-cvss-score` determines whether to use the CVSS Score to configure the finding severity. Defaults to `false`.
- `coverity.include-dismissed-issues` determines whether to include Coverity issues with "Dismissed" status in the list of findings. Defaults to `false`.
- `coverity.include-absent-dismissed-issues` determines whether to include Coverity issues with "Absent Dismissed" status in the list of findings. Defaults to `false`.

When automatically importing Coverity projects and versions from the integrations page, SRM will attempt to parse the branch name from the stream name. Coverity stream names must be globally unique, and so a common pattern is to use the project name, followed by a separator and the branch name. For example, a project `WebGoat` with branch `main` might have a stream named `WebGoat-main`. By default, SRM will attempt to determine the branch from the stream name by checking for the project name, followed by a separator of either `-`, `_`, or a space. Additional stream name parsing behavior can be defined with the following props setting:

- `coverity.stream-name-regexes.#` defines additional regular expressions to be used for Coverity stream name parsing. # must be replaced with 0 for the first regex pattern to be defined. If additional regex patterns are required, each additional one must increment # by one.

The first regular expression to match the stream name in ascending order of their indexes will be used to determine the branch name. The branch name will be extracted from the first capture group of the matching regex. If other groups are required, they must be defined as non-capturing groups. Additionally, if the stream name format includes the project name, `{{projectName}}` may be used in place of the project name. When evaluating the regular expressions, SRM will replace any instances of `{{projectName}}` with each project name's literal.

For example, consider a project `WebGoat` with streams `WebGoat - main` and `Webgoat (develop)`, where `main` and `develop` are the respective branches. The first stream's format may be defined with the regex `{{projectName}} - (.+)`. The second stream's format may be defined with the regex `{{projectName}} \((.+)\)`. Both regexes may be defined in the props file by assigning one to

the property `coverity.stream-name-regexes.0`, and the other to `coverity.stream-name-regexes.1`.

Dynatrace

When ingesting Dynatrace data into Software Risk Manager using the tool connector, it limits the number of entities and related attacks that are included in each result, since these lists can get overwhelmingly large. By default, the tool connector also specifies a port for ActiveGate environments, and a page size for the number of vulnerabilities or attack results to include per page. The relevant properties can be changed according to preferences in the props file:

- `dynatrace.active-gate-environment.port` sets the default port number to use for ActiveGate type Dynatrace environments. Defaults to `9999`.
- `dynatrace.vulnerabilities.page-size` sets the number of vulnerability results to include per page. Should be between 1 and 500. Defaults to `100`.
- `dynatrace.max-num-entities` sets the maximum number of entities that are included in vulnerability results. Defaults to `10`.
- `dynatrace.max-num-attacks` sets the maximum number of related attacks that are included in vulnerability results. Defaults to `10`.
- `dynatrace.attacks.page-size` sets the number of attack results to include per page. Should be between 1 and 500. Defaults to `100`.

SD Elements

Regulations Sections data from SD Elements can be pulled in by SRM based on a configuration setting defined in `codedx.props`. To pull in this data, set the property `sdelements.fetch-regulation-sections` to `true`. (The default setting is `false`.)

Note: The default is set to `false` due to a known issue with SD Elements that requesting Regulations Sections data causes errors.

Configuration Files

For some of the bundled tools, Software Risk Manager provides the ability to define a configuration file, either system-wide or on a per-project basis. Within the Software Risk Manager appdata directory, locate the `tool-data` directory (or create it if it isn't present). To define a configuration file for a tool, create a directory with that tool's name (as specified below). A system-wide configuration should be placed in that directory, or, for a per-project config, create a sub-directory named with the given project's ID, and then place the configuration file in that sub-directory.

Scalastyle

Software Risk Manager supports user-defined `config.xml` scalastyle config files. Place the file within the `tool-data/scalastyle` directory or within a project-specific subdirectory. Files should follow the format defined by scalastyle.

Software Risk Manager also supports a user-defined `scalastyle.props` config file for configuring the JVM environment in which the scalastyle tool runs. Currently, the only supported property is `file.encoding`, which will be passed to the JVM via the `-Dfile.encoding` environment variable. This allows you to run scalastyle on projects that don't use the standard encoding.

Checkstyle

Software Risk Manager supports user-defined `config.xml` Checkstyle configuration files. Place the file within the `tool-data/checkstyle` directory or within a project-specific subdirectory. Files should follow the format defined by Checkstyle.

Cppcheck

Software Risk Manager supports a user-defined `cppcheck.conf` config file. Create the file in the `tool-data/cppcheck` directory or within a project-specific subdirectory. Within that file, you can define a value for the `useThreads` property (e.g., `useThreads=4` to request that Cppcheck use four threads). Also, Cppcheck will run with the `--inline-suppr` option enabled by default, allowing you to suppress errors from within your source code. This behavior can be disabled by setting `inlineSuppression=false`. See the Cppcheck Manual for more details on these settings.

You can also choose to define the `platform` property. This affects Cppcheck's configuration for platform-specific types and sizes. By default, Cppcheck will choose the platform your Software Risk Manager server is running on. Available platform options are as follows:

- `unix32` 32 bit Unix variant
- `unix64` 64 bit Unix variant
- `win32A` 32 bit Windows ASCII character encoding
- `win32W` 32 bit Windows UNICODE character encoding
- `win64` 64 bit Windows
- `avr8` 8 bit AVR microcontrollers
- `native` Type sizes of host system are assumed, but no further assumptions
- `unspecified` Unknown type sizes

You can also use the `libraries` property to specify a list of library configurations for Cppcheck to use. For example, setting `libraries = [gtk, posix, qt]` will enable Cppcheck's library configurations for gtk, posix, and qt. The available configurations are avr, bento4, boost, bsd, cairo, cppcheck-lib, cppunit, daca, dpdk, embedded_sql, emscripten, ginac, gnu, googletest, gtk, icu, kde, libcerror, libcurl, libsigc++, lua, mfc, microsoft_atl, microsoft_sal, microsoft_unittest, motif, nspr, ntl, opencv2, opengl, openmp, openssl, pcre, posix, python, qt, ruby, sdl, sfml, sqlite3, tinyxml2, vcl, windows, wxsqlite3, wxsvg, wxwidgets, and zlib.

JSHint

Software Risk Manager supports a user-defined `jshint.conf` config file. Create the file in the `tool-data/jshint` directory or within a project-specific subdirectory. Within that file, you can define a value for the `scan-html` property, which tells JSHint to additionally scan `.htm` and `.html` files. This property is enabled by default (i.e., `scan-html=true`).

You can also define a value for the `extract` property, which sets the value for the `--extract` flag from JSHint. This flag controls how JSHint extracts HTML from files. Available extract options are as follows:

- `auto` JSHint will attempt to extract javascript only if the file looks like it is an HTML file. This is the default value (i.e., `extract=auto`).
- `always` JSHint will always attempt to extract javascript. Note that if this value is set, JSHint will scan all files as HTML. This means that javascript source files will not be scanned normally. Instead, JSHint will search them for HTML to extract and anything else in the file will be ignored.
- `never` JSHint will never attempt to extract javascript. Note that if this value is set, JSHint will not be able to scan HTML. `scan-html` must be set to `false`, otherwise JSHint will try to scan HTML files as javascript source and will report an error for each one.

PHPMD

Software Risk Manager supports user-defined `config.xml` PHPMD ruleset files. Place the file within the `tool-data/phpmd` directory or within a project-specific subdirectory. Files should follow the format defined by PHPMD.

PHP_CodeSniffer

Software Risk Manager supports user-defined `config.xml` PHP_CodeSniffer ruleset files. Place the file within the `tool-data/phpcodesniffer` directory or within a project-specific subdirectory. Files should follow the format defined by PHP_CodeSniffer.

PMD

In Software Risk Manager's config file in the `tool-data/pmd` directory or within a project-specific subdirectory, you can define a value for the `debug-logging` property (e.g., `debug-logging=true` to enable verbose PMD logs). You can also define a value for the `encoding` property (e.g., `encoding=UTF-8` to set the character encoding PMD expects source to be in. Acceptable values are any standard `java.nio.charset.Charset` character set). By default, these are `debug-logging=false` and `encoding=UTF-8`.

ESLint

By default, Software Risk Manager will pass `--ignore-pattern **/node_modules/*`.

Configuration

Software Risk Manager supports user-defined `.eslintrc` and `.eslintignore` ESLint config files. Place the file(s) within the `tool-data/eslint` directory or within a project-specific subdirectory. Files should follow the format defined by ESLint. You can use the `js`, `yml`, `json`, or `eslintrc` formats.

In the event that there are multiple config files present in different formats, Software Risk Manager will follow the same priority that ESLint uses. ESLint will also use any `.eslintrc` config files uploaded in the project as defined by the rules explained here.

Additionally, Software Risk Manager will search for an `.eslintignore` file in the root directory of the uploaded zip file. The `.eslintignore` file in the zip will take precedence over the `.eslintignore` file in the `tool-data/eslint` or project-specific directories. If you use a custom config, be sure to adjust your tool configuration. By default, only the recommended ESLint rules are enabled.

Software Risk Manager also supports user-defined "appdata-dir.dita"he default `eslintrc.json` file. For users wanting to add minor adjustments to the default configuration, a `eslint-config-extra.js` can be placed within the `tool-data/eslint` directory or within a project-specific subdirectory (a directory whose name is the numeric `id` of the Software Risk Manager project), as with the custom `.eslintrc` and `.eslintignore` files mentioned above. When running ESLint, SRM will search for this "extra" config file and, if present, will include it as an extension of its default ESLint config file.

The contents of the `eslint-config-extra.js` file should be in the form `module.exports = { ... }`, where the `...` is replaced by your custom configuration as defined in the ESLint configuration guide.

Important notes about using a custom `.eslintignore` file

- If an `.eslintignore` file is neither found in the uploaded zip file nor provided in the `tool-data/eslint` directory, Software Risk Manager will pass `--ignore-pattern **/node_modules/*` as a command-line argument to ESLint.
- If you want to provide an `.eslintignore` file, it is a good idea to include the `**/node_modules/*` pattern in that file; linting the contents of your third-party dependencies is not recommended, due to the fact that in some cases this can cause ESLint to fail.

Several ESLint plugins are made available when running as a bundled tool in Software Risk Manager:

- eslint-plugin-security: enabled by default
- eslint-plugin-scanjs-rules: disabled by default
- eslint-plugin-no-unsanitized: enabled by default

- eslint-plugin-xss: enabled by default
- eslint-plugin-html: enabled by default
- eslint-plugin-react: enabled by default
- eslint-plugin-react-hooks: enabled by default
- eslint-plugin-n: enabled by default; replaces eslint-plugin-node
- eslint-plugin-node: disabled by default
- eslint-plugin-import: enabled by default
- eslint-plugin-flowtype: enabled by default
- eslint-plugin-jsx-a11y: disabled by default in Software Risk Manager Tool Configuration
- @stylistic/eslint-plugin-js: disabled by default
- @babel/eslint-parser: default Software Risk Manager parser

Any of these plugins can be used in or excluded from your user-defined configs. All plugins use their default or recommended rules and settings, except for no-location-href-assign from `eslint-plugin-xss`. For this rule, Software Risk Manager uses `encodeURIComponent` for the `escapeFunc` option instead of the default `escape` function, which has been deprecated. More information about how to configure these plugins can be found on their respective npmjs or GitHub pages.

By default, Software Risk Manager will run ESLint on `.js`, `.html`, and `.htm` files. If you want to change this behavior, you can specify a comma-separated list of file extensions with the `eslint.extensions` setting in your `codedx.props` file. You do not need to include the `.` with each file extension (`eslint.extensions = js,html,htm`). Note that if you decide not to use the html plugin in your custom configs, you must use this setting to specify that you want ESLint to only run on `.js` files. Otherwise, it will try to scan any `.html` and `.htm` files it finds as javascript, which will cause ESLint parsing errors.

Custom Plugins and ESLint Installations

Software Risk Manager allows you to install third-party ESLint plugins, shareable configs, and parsers. To do so, first make sure you have `npm` installed and accessible from the command line by installing Node.js. Then, open a command line and navigate to the `tool-data/eslint` directory or a project-specific subdirectory. From there, you can install any plugins, shareable configs, and parsers you wish to use by running the command `npm install ... --save`. You do not need to install any of the plugins that are included with Software Risk Manager by default.

If you want to use your own ESLint environment, you can specify the path to it in your `codedx.props` file with the `eslint.path` setting. Software Risk Manager expects that the provided directory will point to the folder containing ESLint's `bin` folder. Software Risk Manager will also try to find any of the supported ESLint config files in this directory. Note that if you use the global or project config files as described above, they will take precedence over the ones in this folder. Also, when using an external installation, Software Risk Manager will ignore any custom plugin folders in the `tool-data/eslint` directory.

ZPA

Software Risk Manager supports user-defined `forms-metadata.json` ZPA Oracle Forms metadata files. Place the file within the `tool-data/zpa` directory or within a project-specific subdirectory. Files should follow the format defined by ZPA.

STIG

By default, the STIG reader only ingests vulnerabilities with `Open` and `Not Reviewed` status. The following properties can be configured in the props file to also ingest other statuses if required:

- `stig.status.include-not-a-finding` determines whether to ingest vulnerabilities with `Not A Finding` status. Defaults to `false`.
- `stig.status.include-not-applicable` determines whether to ingest vulnerabilities with `Not Applicable` status. Defaults to `false`.

IriusRisk

When ingesting IriusRisk data into Software Risk Manager using the tool connector, either based on Threats or Countermeasures, it ingests custom fields data and puts it into the result contextual description. The tool connector specifies a page size which can be used to configure the number of custom fields that are included per page of the API response:

- `iriusrisk.custom-fields-page-size` sets the number of custom fields to include per page. Defaults to `20`.

Also, when ingesting IriusRisk data based on Countermeasures specifically, the tool connector includes the `Implementations` tab data in the contextual description. This data is represented in Base 64 encoded form in the API response, which is then decoded by SRM to get the raw data to put into the result contextual description. The relevant properties to set the decoding scheme and charset used for this can be changed according to preferences in the props file:

- `iriusrisk.countermeasure-implementation-decoder` sets the specific decoding scheme to use for the implementations data decoding process. Acceptable values include `Default`, `URL` and `MIME`. Defaults to using all three schemes until decoding is successful - using `Default`, `URL` and `MIME`, in that order.
- `iriusrisk.countermeasure-implementation-charset` sets the specific charset to use for the implementations data decoding process. Any canonical name or alias of a charset is an acceptable value. Defaults to using `UTF-8`.

JVM System Properties

You can set custom system properties for the JVM process that hosts Software Risk Manager. To do so, use the prefix `codedx.jvmprops.` on a line in your `codedx.props` file. For example, to set the `http.proxyHost` system property to `1.2.3.4`, add the following line:

```
codedx.jvmprops.http.proxyHost = 1.2.3.4
```

Note that specifying a setting this way will overwrite existing settings in the JVM. For example, it is possible to overwrite the `user.home` property, which may be used by logic within Software Risk Manager. Use care that you don't overwrite an important value.

For a non-exhaustive list of system properties to be aware of, see https://docs.oracle.com/javase/tutorial/essential/environment/sysprop.html.

Proxies

Some features, like JIRA Integration and Tool Connectors, reach out to third-party programs via HTTP(S). If your Software Risk Manager server is running behind a proxy, you can configure Software Risk Manager to use that proxy for communications with these programs by setting the appropriate properties:

- `proxy.host` - The hostname, or address, of the proxy server, e.g., `1.2.3.4` or `myproxy.mydomain.com`.
- `proxy.port` - The port number of the proxy server (default: 80).
- `proxy.username` - The username for authentication, if necessary.
- `proxy.password` - The password for authentication, if necessary.

- `proxy.nonProxyHosts` - A `|`-separated list of hosts that should be accessed without going through the proxy. (default: `localhost|127.*|[::1]`)

A typical proxy configuration in your `codedx.props` file would resemble the following:

```
proxy.host = 123.234.156.178
proxy.port = 3128
proxy.username = myproxyuser
proxy.password = myproxypassword
```

Visual Log Configuration

By default, the visual log will not record `successful-login` events. To enable this, set the `auth.logging.recordSuccess = true`.

Note: Changing this property will not retroactively reveal successful logins that occurred previously, as this setting determines whether to *record* successful logins, not whether to *show* them.

Tool Orchestration Configuration

When Software Risk Manager is deployed on a Kubernetes cluster, you can run orchestrated analyses using the Software Risk Manager Tool Orchestration. If you have a Software Risk Manager license that includes Tool Orchestration, enable the feature when you run the Helm Prep Wizard.

Analysis Cleanup

By default, Software Risk Manager will remove old analyses and related storage from the tool service automatically. You can customize or disable this behavior with the following options in the `codedx.props` file:

- `tws.storage.cleanup.enabled = true` - sets whether to enable automatic analysis cleanup [default: true].
- `tws.storage.cleanup.max-total-analyses = 100` - sets the maximum number of orchestrated analyses to keep; further analyses will cause the oldest to be deleted [default: 100].
- `tws.storage.cleanup.max-analyses-per-project = 5` - sets the maximum number of orchestrated analyses to keep *per-project*; further analyses will cause the oldest in the project to be deleted [default: 5].

Note: Both successful and failed analyses count toward the "total" being tracked.

Issue Tracker Configuration

To support issue tracker integrations, Software Risk Manager will make requests to the issue tracker server periodically in order to create, update, and read issues. Due to the volume of requests that can be made in quick succession, issue tracker servers may begin to deny requests to prevent the server from being overloaded.

The following properties will affect how Software Risk Manager makes requests.

- `azure.auto-create-delay` [default: 50] - sets the delay (in ms) between subsequent work item creation requests made during an auto create job.
- `gitlab.auto-create-delay` [default: 60] - sets the delay (in ms) between subsequent requests made during auto create jobs and bulk update jobs.
- `jira.auto-create-delay` [default: 50 - sets the delay (in ms) between subsequent issue creation requests made during an auto create job.

- `servicenow.request-delay` [default: 750] - sets the delay (in ms) between subsequent requests made during auto create jobs and bulk update jobs.

Machine Learning Triage Server Memory Usage

The Software Risk Manager machine learning aided triage functionality is handled by a separate process, the *Machine Learning Triage Server*, which Software Risk Manager spawns and manages. Training a new prediction model is computationally intensive and can use large amounts of memory. There are times where training performance can be improved with the availability of more memory. The upper bound on the amount of memory that the *Machine Learning Triage Server* JVM can use during training and making predictions can be configured by setting the following properties setting:

`mltriage.service.maxheap` [default: none] - sets the max heap size (in megabytes) of the *Machine Learning Triage Server* JVMs.

Automatic Updating of the Software Risk Manager Prediction Model

Software Risk Manager can automatically update your prediction model. This feature can be configured or turned off completely by setting the following props settings in your `codedx.props` file:

- `mltriage.enable-periodic-retrain` [default: true] - enables Software Risk Manager to periodically train a prediction model.
- `mltriage.model-train-time` [default: 01:45:00] - the time at which Software Risk Manager will check if the threshold set in mltriage.retrain-threshold has been met, and if it has, train a prediction model.

Secure Code Warrior

Software Risk Manager integrates with Secure Code Warrior to provide developers with resources to help facilitate the learning of secure coding practices. This feature can be disabled by setting the following props setting in your `codedx.props` file:

- `codedx.scw.enabled` [default: true if offline mode is off, false otherwise] - enables the Software Risk Manager Secure Code Warrior integration and its relevant features.

Software Risk Manager Scoring Calculations

In analysing risk, Software Risk Manager calculates a "code score" (Critical, High, Medium, and Low) by averaging a "custom code score" and a "component score." This is done through a configurable function in the form `f(severity, count)` for certain metrics (shown below). However, this formula can be customized if needed. (For more information on Risk Scoring, see the *Risk Score* section in the *User Guide*.)

The metrics used for this calculation are as follows:

- `componentFindingVolume` - in which the count represents the number of component findings for a given severity.
- `customCodeFindingVolume` - in which the count represents the number of non-component findings for a given severity.
- `customCodeFindingVariety` - in which the count represents the number of finding types for a given severity.

These metrics are used to calculate a "penalty," which is removed from a top score of 100 down to a minimum score of 0. The "component score" uses the first of the metrics listed above; "custom code score" uses the sum of the other two.

The `f` function can be configured by name; for example, the *base config path* for the `componentFindingVolume` formula would be `dashboard.score.componentFindingVolume`.

Once the function is configured, there are three suffixes that control the formula:

- `.formulaType`
  - If set to `log`, the formula is `(severity, count) => log_<logBase>(count) * criticalWeight / (critical - severity)^2`
  - If set to `linear`, the formula is `(severity, count) => count * criticalWeight / (critical - severity)^2`
    (The `customCodeFindingVolume` metric uses the `log` formula by default, while the rest use `linear`.)
- `.criticalWeight` represents how much weight a Critical severity holds in the formula. "High" will have 1/2 the weight; "Medium," 1/4. (The default is 3.0.)
- `.logBase` is used as the base number for the log function in the `log` formulaType; for example, log-base-2 or log-base-10. (The default is 2.0.)

Here is an example of a default configuration for the `componentFindingVolume` metric:

```
dashboard.score.componentFindingVolume.formulaType = log
dashboard.score.componentFindingVolume.criticalWeight = 3.0
dashboard.score.componentFindingVolume.logBase = 2.0
```

SMTP Configuration

To support policy email notifications, a valid SMTP configuration is required. The following properties are used to configure SMTP:

- `smtp.sender` - [required] the address the email is being sent from.
- `smtp.host` - [required] the SMTP server being used to send email.
- `smtp.user` - the username for the SMTP server, if authentication is required.
- `smtp.password` - the password for the SMTP server, if authentication is required.
- `smtp.oauthToken` - the OAuth 2.0 access token for the SMTP server, if authentication is required.
- `smtp.port` - [default: 25] the port of the SMTP server.
- `smtp.auth` - [default: false] true/false, if authentication is being used.
- `smtp.tls` - [default: false] true/false, if TLS is being used.
- `smtp.additionalProps` - a list of additional SMTP properties to configure the SMTP server. These are expected as comma-separated `key -> value` pairs. For example: `smtp.additionalProps = mail.smtp.ssl.enable -> true, mail.smtp.ssl.trust -> *` You can refer to the full list of available SMTP properties here.

Host Normalization

Host normalization can be configured via the following props:

- `host-normalization.use-srm-tracking` – [default: false] Whether or not SRM should use native host tracking and ignore externally reported host tracking methods.

Webhook Configuration

Webhooks can be configured via the following props:

- `webhook.finding-chunk-size` – [default: 100] The maximum number of findings in a payload per request.

Polaris Assist

Polaris Assist can be configured using the properties listed below. For more information, see AI Insight Using Polaris Assist in the *User Guide*.

Properties Settings

The following properties can be set in the Software Risk Manager properties file:

- `assist.model-id [default: 'gpt-4o']` - The ID of the LLM that will be requested by SRM.
- `assist.code-context.max-lines [default: 100]` - The maximum number of lines of code to include as context for requests to the LLM (including the line range indicated by the finding and any surrounding lines).
- `assist.code-context.max-chars [default: 2048]` - The maximum number of characters of code to include as context for requests to the LLM.
  Lines of code will be excluded entirely if they would be truncated by this limit. For assessments on minified files that consist of a single line, this typically prevents code from being included at all.
- `assist.max-response-tokens [default: 100]` - The number of response tokens to request from the LLM for each assessment.
- `assist.max-file-size [default: 500000]` - The maximum file size (in bytes) of a source file that may be read to build the code-context for requests to the LLM.
  Source files which exceed this size will not be used when making requests to the LLM, regardless of `assist.code-context` settings.

Model ID, LLM API

SRM supports connection to Azure OpenAI APIs, making requests to `{Azure OpenAI URL}/openai/deployments/{model-id}/completions` with an `api-key` header.

Polaris Assist has been tested and validated with base models of GPT-3.5 Turbo, GPT-4, and GPT-4o. We do not recommend fine tuning or the use of other models.

Azure OpenAI

For installations connecting directly to Azure OpenAI, `model-id` will be the "Deployment name" of the model deployed with Azure OpenAI Studio. SRM does not use batch APIs, and the model's deployment type in Azure OpenAI Studio must *not* be "Batch."

Customizing the Policy Email Template

Software Risk Manager allows notifications to be sent by email when policy status changes. (To enable email notifications, see Configuring Email Notifications in the *SRM User Guide*.) There are three standard email templates, which can be customized as needed.

To customize a policy email template:

1. Navigate to the AppData folder and locate the template files.
   There are three generic email templates, and each one is based on a policy violation status:
   - `error-builtin-template.json`
     This file is used to notify the user that a policy has been violated.
   - `success-builtin-template.json`

This file is used to notify the user that the policy violation has been resolved.

- `warning-builtin-template.json`
  This file is used to notify the user that a policy has a violation risk.

(For more information on the SRM AppData directory, see Understanding the AppData Directory.)

2. Open the file you want to customize using any text editor.
3. Make changes to the text as needed.
   Changes should be limited to the actual text; variables or references, such as `\"{project-name}\"`, should not be altered.
4. Save the file using the form `*-custom-template.json`.
   Say, for example, you want to customize the "error" email template. You would locate the file `error-builtin-template.json`, edit the file, then save it as `error-custom-template.json`.

Configuring Automatically Deactivating Users

SRM can automatically deactivate inactive users. The following props can be configured:

- `maintenance.deactivate-inactive-users.enabled` - [default: false] set this to true to have SRM automatically deactivate inactive users.
- `maintenance.deactivate-inactive-users.scheduling-strategy` - [default: daily] defines when the job will be ran. Allowed values are `daily` and `interval.`
- `maintenance.deactivate-inactive-users.run-time` - [default: 00:00:00] the exact time when this job will run. Only used when `maintenance.deactivate-inactive-users.scheduling-strategy` is set to `daily`.
- `maintenance.deactivate-inactive-users.interval` - [default: 24 hours] the interval of time in which this job run. Only used when `maintenance.deactivate-inactive-users.scheduling-strategy` is set to `interval`.
- `maintenance.deactivate-inactive-users.after-days` - [default: 30] the number of days a user can be inactive before this job will automatically deactivate them. A user is considered active if they have either logged in or have been activated within this number of days.

Project Configuration

The following properties can be used to configure Software Risk Manager project behavior:

- `project.default-branch-name` [default: 'main'] - The name to use for the default branch when creating new projects.
- `project.scan-coverage.include-parent-projects` [default: false] - Decides whether or not to include child projects which are also parent projects in the scan coverage calculation.