Polyspace® Bug Finder™ Access™

Getting Started Guide

R2019b

MathWorks®

How to Contact MathWorks

Latest news:	`www.mathworks.com`
Sales and services:	`www.mathworks.com/sales_and_services`
User community:	`www.mathworks.com/matlabcentral`
Technical support:	`www.mathworks.com/support/contact_us`
Phone:	508-647-7000

The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

Polyspace® Bug Finder™ Access™ Getting Started Guide

Trademarks

Patents

Revision History

March 2019	Online Only	New for Version 2.0 (Release 2019a)
September 2019	Online Only	Revised for Version 2.1 (Release 2019b)

Contents

1 Install Polyspace Bug Finder Access

Polyspace Bug Finder Access Product Description **1-2**

System Requirements for Polyspace Access **1-3**

Storage Configuration **1-5**

Network Port Configuration **1-7**

Configure and Start the Cluster Operator **1-8**
Prerequisites **1-8**
Unzip Installation Image and Start COP **1-8**
Configure COP for HTTPS **1-10**

Configure Polyspace Access Services **1-13**
Prerequisites **1-13**
Default Installation **1-14**
Custom Installation **1-17**
Configure Polyspace Access Services with HTTPS **1-18**
Configure Lightweight Directory Access Protocol **1-20**

Configure the User Manager **1-26**

Configure the Database and ETL Services **1-29**

Configure the Web Server and Gateway **1-31**
Add JIRA Instance Configured with HTTPS **1-33**

License Management **1-35**

Register Polyspace Desktop User Interface **1-36**
Generate Client Keystore for Desktop Interface **1-38**

Start Polyspace Access and Upload Examples **1-40**
Configure Polyspace Access to Restart Automatically **1-41**
Upload Examples **1-41**
Open the Polyspace Access Web Interface **1-44**

Database Backup **1-45**
Database Backup **1-45**

Database Clean Up **1-47**
Perform Database Vacuuming **1-47**
Delete Outdated Projects **1-48**

Update or Uninstall Polyspace Access **1-51**

2 Get Started with Polyspace Bug Finder Access

Upload Results to Polyspace Access **2-2**
Upload Results from Polyspace Desktop Client **2-2**
Upload Results at Command Line **2-3**

Open or Export Results from Polyspace Access **2-5**
Open Polyspace Access Results in a Desktop Interface **2-5**
Export Polyspace Access Results to a TSV File **2-5**

Dashboard **2-7**

Review **2-16**

Manage Permissions and View Project Trends **2-19**
Create a Project Folder **2-19**
Manage Project Permissions **2-20**
View Project Trends **2-23**

Manage Results **2-25**

Migrate Results from Polyspace Metrics to Polyspace Access
. . . . **2-28**
Requirements for Migration **2-30**
Migration of Results **2-30**

Differences in SQO Between Polyspace Metrics and Polyspace Access **2-32**

Quick Start Guide for Polyspace Server and Access Products 2-34

Installation **2-35**

Setting Up Polyspace Analysis **2-36**

Install Polyspace Bug Finder Access

- “Polyspace Bug Finder Access Product Description” on page 1-2
- “System Requirements for Polyspace Access” on page 1-3
- “Storage Configuration” on page 1-5
- “Network Port Configuration” on page 1-7
- “Configure and Start the Cluster Operator” on page 1-8
- “Configure Polyspace Access Services” on page 1-13
- “Configure the User Manager” on page 1-26
- “Configure the Database and ETL Services” on page 1-29
- “Configure the Web Server and Gateway” on page 1-31
- “License Management” on page 1-35
- “Register Polyspace Desktop User Interface” on page 1-36
- “Start Polyspace Access and Upload Examples” on page 1-40
- “Database Backup” on page 1-45
- “Database Clean Up” on page 1-47
- “Update or Uninstall Polyspace Access” on page 1-51

Polyspace Bug Finder Access Product Description

Review code analysis results and monitor software quality metrics

Polyspace Bug Finder Access provides a web browser interface to Polyspace static code analysis results and quality metrics. It includes a central repository for analysis results that enables team-based collaboration. Results from Polyspace Bug Finder Server™ can be published to Polyspace Bug Finder Access for triage and resolution. With Polyspace Bug Finder Access you can create and assign tickets in defect-tracking systems such as Jira.

Polyspace Bug Finder Access dashboards display information that you can use to monitor software quality, project status, number of defects, and code metrics (such as lines of code, cyclomatic complexity, and recursion). The dashboards help you graphically track compliance to coding rules standards such as MISRA C:2012 and AUTOSAR C++14 and security standards such as CERT-C and ISO/IEC TS 17961.

System Requirements for Polyspace Access

- The installation of Polyspace Access components requires Docker version 1.10 or later.
- The installation of Polyspace Access has been validated with Docker Community Edition (CE) only.

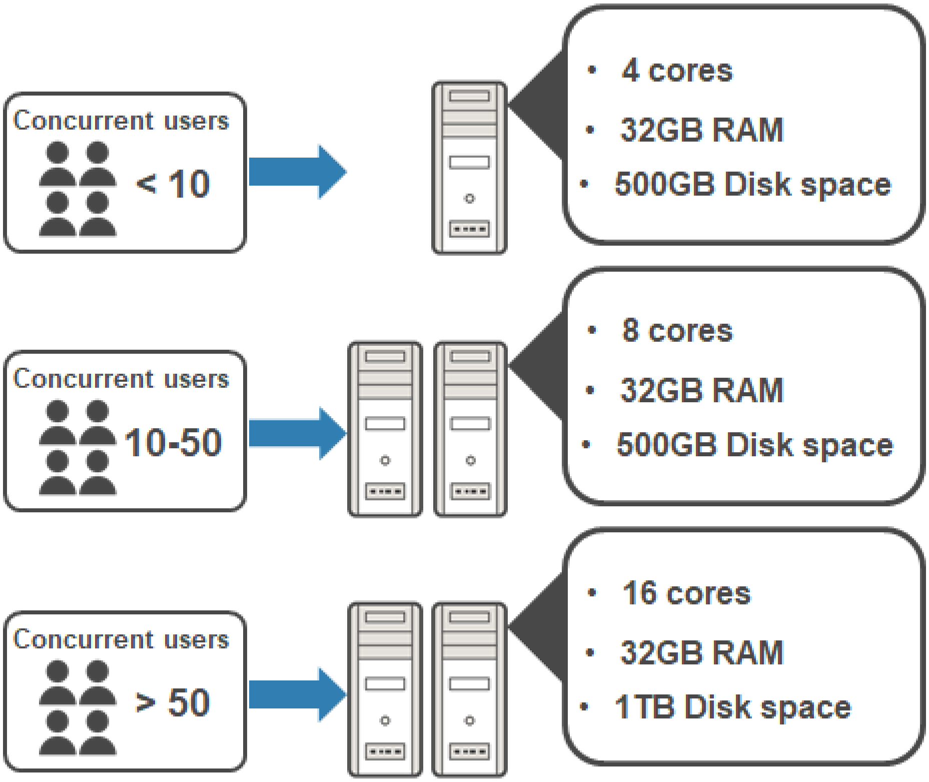
  To install Docker on your machine, click your platform and follow the installation instructions. The installation of Polyspace Access on a Linux system is recommended.

  On Windows® systems:

  - Your Windows 10 edition must include Hyper-V and you must enable virtualization in your BIOS.
  - During the installation, in the **Installing Docker for Windows** window, make sure that you clear the **Use Windows containers instead of Linux containers** box.
  - You cannot run Docker CE as a service on Windows 10. Logging off your machine shuts down Polyspace Access. The creation of a dedicated user account to start Docker CE is recommended.
  - Docker version 2.1.0.0 and later does not support Windows build 14393.
- The Polyspace Access installation process requires Internet access to build docker images.
- It is recommended to install Polyspace Access on physical machines. Installing Polyspace Access on virtual machines is possible, but this might affect the performance of Polyspace Access and technical support will be limited.
- The Cluster Operator does not support the Internet Explorer web browser.
- Docker CE is supported on Windows 10 and these Linux® distributions:

  - Ubuntu® 14.04 LTS, 16.04 LTS, and 17.04
  - Debian® 7.7, 8.0, and Stretch
  - Maintained version of CentOS 7
  - Fedora™ 24 and 25

  A full list of supported platforms for Docker is available here.
- Different minimum hardware configurations are recommended, depending on how many people in your organization use Polyspace Access simultaneously. This diagram lists the minimum recommended configurations per machine.



- Data transfers between the server and client machines require a high speed network connection. A gigabit network connection is recommended.

See Also

More About

- "Install Polyspace Access"

Storage Configuration

- To ensure optimal data storage performance, use physical drives instead of networked storage solutions.
- The database is stored under `/var/lib/docker/volumes`. Make sure that you have adequate disk space for the Database mount point. On Windows systems, this folder is on the hard disk of the virtual machine running the Docker services. The default location for this virtual hard disk is `C:\Users\Public\Documents\Hyper-V\Virtual Hard Disks\MobyLinuxVM.vhdx`.
- It is a best practice to secure the database mount point with a RAID array and to back up the mount point regularly.
- Allocate this recommended amount of disk space for the mount points of the working directories of the Polyspace Access processes:

  - **Temporary upload directory**: 3 GB

    Uploaded files are stored in this directory while they are transferred to the web server mount point.
  - **Upload directory**: 3 GB

    Once the transfer to the web server mount point is complete, files are moved to this directory. The path to this directory must be the same for the extract-transform-load (ETL) and web server services. If the services are on different machines, the paths to this directory must point to the same hard drive.
  - **Storage directory**: 15 GB

    The ETL (import process) looks for files in the upload directory and stores them in the storage directory. Files that are successfully uploaded to the database are deleted. Files that fail to upload are sent to the invalid results directory.
  - **Working directory**: 3 GB

    The ETL (import process) uses this directory to process files from the storage directory. Files are treated in the order in which they are received. The data is prepared to be sent to the database.
  - **Invalid results directory**: 50 GB

    Files that fail to upload are stored in this directory. You can recover and analyze the files to determine why the upload failed. Back up this folder regularly. Set up a policy to determine the amount of time after which older data can be deleted.

- Make sure that all users have read and write permissions for the directories you specify under **ETL** and the **Temporary upload directory** in the COP settings.

See Also

More About

- “Install Polyspace Access”
- “Database Backup” on page 1-45

Network Port Configuration

When you configure the Polyspace Access services, you specify port numbers for some of the services. To avoid installation errors, and to ensure that the services are accessible, make sure that the ports that you specify are open. To check whether a port `portNumber` is open, use these commands:

- On Windows: `netstat -na | find "portNumber"`
- On Linux: `netstat -na | grep portNumber`

If the output of the command is not empty, the port is in use. Specify a different port or stop the service currently using the port.

This table lists the port numbers of the different services when you use the default configuration for the Polyspace Access installation.

Service	Default Port Number
Cluster Operator	8080 for HTTP configuration 8443 for HTTPS configuration
Database	5432
User Manager	5001
Web Server	9444
Gateway	9443

See Also

More About

- “Install Polyspace Access”
- “Configure Polyspace Access Services” on page 1-13

Configure and Start the Cluster Operator

The Cluster Operator (COP) is an agent that enables you to install, configure, and start the Docker containers for the different Polyspace Access services.

Prerequisites

Before configuring and starting the COP, make sure that:

- You have installed the required software and that your system meets the minimum hardware requirements. See "System Requirements for Polyspace Access" on page 1-3.
- You have enough data storage available. See "Storage Configuration" on page 1-5.
- Docker is running on your machine. At the command line, type:

  ```
  docker stats --no-stream
  ```

  If you get an error message, follow the instructions in this table. Otherwise, press **CTRL+C** to return to the command line.

UNIX/Linux	Windows
To start Docker, run the command `sudo systemctl start docker`. If `systemctl` is not available, use `service` instead. After you start Docker, you must be logged in as a member of the `docker` group to run Docker commands. To see a list of current members of this group, use the command `grep 'docker' /etc/group`	Go to the **Start** menu, type `docker`, and press **ENTER**. If you get a sign-in request, close the sign-in window. You do not have to sign in to start Docker. Once Docker has started, from the Windows taskbar, right-click the Docker icon in the system tray and open the settings. On the **Shared Drives** tab, select the drives you want to share with Docker. See the Docker documentation.

Unzip Installation Image and Start COP

The COP binary is included with the `polyspace-access-`*`VERSION`*`.zip` installation image for Polyspace Access. *`VERSION`* is the release version, for instance `R2019a`. After you download the installation image, unzip it to extract these files:

- `cop-docker-agent` and `cop-docker-agent.exe`.
- `polyspace-cop.tar`
- `polyspace-db.tar`
- `polyspace-etl.tar`
- `polyspace-gateway.tar`
- `polyspace-usermanager.tar`
- `polyspace-web-server.tar`

To start the COP binary, from the command line, navigate to the installation folder where you extracted the contents of the `zip` installation image. Once inside this folder, at the command-line, enter:

```
cop-docker-agent
```

If you configure the COP or Polyspace services with HTTPS, specify *`hostName`*, the fully qualified domain name (FQDN) of the machine where you run the COP:

```
cop-docker-agent --hostname hostName
```

For more information, see “Configure COP for HTTPS” on page 1-10 and “Configure Polyspace Access Services with HTTPS” on page 1-18. The command line outputs messages indicating that the agent is downloading image layers. After the download is complete, you see a message with information on how to connect to the agent:

```
Cluster Operator started. You can now connect to the Cluster Operator through
your web browser at http://localhost:8080/ using the initial password randomPass
```

`randomPass` is a randomly generated initial password. Copy this password. The command-line output shows the password only the first time you start COP. To reset the COP password, press **CTRL+C** to stop the COP binary if it is running and run this command:

```
cop-docker-agent --reset-password
```

To view the new password, restart the COP binary.

By default, the COP starts on port 8080. If this port is already in use, you get a `Permission denied` error message. To start the COP on a different port, use the flag `--port` and specify a different port number, for instance:

```
cop-docker-agent --port 9999
```

In Windows 10, if you get an error message about shared drives, make sure that you shared your drives with Docker. To see which drives are shared with Docker, right-click the Docker icon in the system tray, select settings, then click the **Shared Drives** tab. See the Docker documentation.

Once you start the COP, the agent creates a `settings.json` file and stores it in the same folder as the COP binary by default. Ensure that only the user who starts the COP has read/write permissions on this file.

Configure COP for HTTPS

To encrypt the data between the COP server and client machines, configure the COP with the HTTPS protocol. To complete the configuration, you must provide a certificate and the private key you used to generate the certificate as `PEM` files.

It is recommended to use a certificate issued by a certificate authority to configure HTTPS. If you do not want to use a certificate authority, you can configure HTTPS by using self-signed certificates. By default, the COP starts on port 8443 when you enable HTTPS.

Secure your COP private key by following best practices such as:

- Do not transfer the private key between machines. Instead, generate and store the private key on a local file system.
- Restrict read/write permissions. Grant access to the private key file only to the COP administrators.
- Rotate your private key and certificate regularly (annually) and audit which users have access to the private key file.

The configuration of HTTPS for the COP does not enable HTTPS for the Polyspace Access web interface.

Use Certificates Signed by a Certificate Authority

These steps illustrate how to configure SSL encryption on a Debian Linux system by using your organization's certificate authority and the `openssl` utility.

1. Create a certificate signing request. In the `CN` field (common name), specify *`hostName`*, the fully qualified domain name (FQDN) of the machine where you run the COP.

```
openssl req -new -newkey rsa:4096 -nodes -out myRequest.csr -keyout myKey.key \
-subj "/C=US/ST=/L=/O=/CN=hostName"
```

The command outputs a private key file `myKey.key` and the file `myRequest.csr`, which contains a public key and data that describes your server.

2. Submit `myRequest.csr` to your organization's certificate authority. The certificate authority uses the file to generate a signed server certificate. For instance, `cop_cert.cer`.
3. Start the COP by using the generated private key and signed certificate:

```
./cop-docker-agent --hostname hostName\
--https-certificate-file fullPathTo/cop_cert.cer \
--https-private-key-file fullPathTo/myKey.key \
--https-trusted-certificates-file /etc/ssl/certs/ca-certificates.crt
```

The *`hostName`* you specify in this command must match the *`hostName`* you specified in step 1. *`fullPathTo`* is the full file path. When you open the COP web interface, your browser considers the connection secure if the browser uses the certificate store that you specify for **Trusted certificates file:**.

Use Self-Signed Certificates

To configure HTTPS on a Debian Linux system by using a self-signed certificate that you generate with `openssl`, follow these steps:

1. Generate a certificate and private key as `PEM` files. In the `CN` field (common name), specify *`hostName`*, the fully qualified domain name (FQDN) of the machine where you run the COP.

```
openssl req -newkey rsa:2048 -new -nodes -x509 -days 365 -keyout private_key.pem \
-out certificate.pem -subj "/C=US/ST=/L=/O=/CN=hostName"
```

2. Start the COP by using the generated `certifcate.pem` and `private_key.pem` files.

```
./cop-docker-agent --hostname hostName\
--https-certificate-file fullPathTo/certificate.pem \
--https-private-key-file fullPathTo/private_key.pem \
--https-trusted-certificates-file fullPathTo/certificate.pem
```

The *`hostName`* you specify in this command must match the *`hostName`* you specified in step 1. *`fullPathTo`* is the full file path. If you use relative paths, you get an error message.

See Also

`cop-docker-agent`

More About

- “System Requirements for Polyspace Access” on page 1-3
- “Storage Configuration” on page 1-5
- “Configure Polyspace Access Services” on page 1-13

Configure Polyspace Access Services

To set up the Polyspace Access centralized database, web server, and user authentication, you install these services:

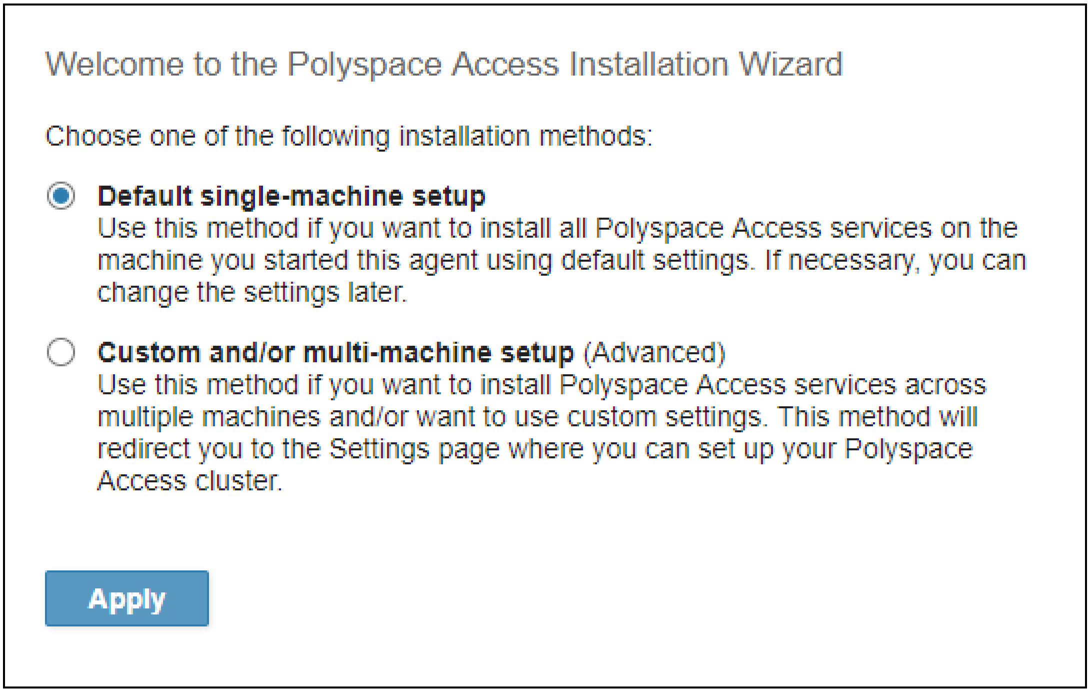
- **User Manager**: To authenticate user logins and issue signed JSON Web Tokens to authenticated users.
- **Database** and extract-transform-load (**ETL**): To upload and manage results on the database.
- **Web Server**: To provide a user interface that you can open in a web browser.
- **Gateway**: To handle all communications between clients and the other Polyspace Access services.

Prerequisites

Before you begin configuring the Polyspace Access services, make sure that you have started the cluster operator. See “Configure and Start the Cluster Operator” on page 1-8 (COP).

After you configure and start COP, open your web browser and go to URL specified in the command-line output when you started COP.

Log in with the initial password that you obtained when you started the COP agent. If this time is your first time logging in, follow the prompts. Then, on the installation wizard page, select an installation method.



Welcome to the Polyspace Access Installation Wizard

Choose one of the following installation methods:

● **Default single-machine setup**
Use this method if you want to install all Polyspace Access services on the machine you started this agent using default settings. If necessary, you can change the settings later.

○ **Custom and/or multi-machine setup** (Advanced)
Use this method if you want to install Polyspace Access services across multiple machines and/or want to use custom settings. This method will redirect you to the Settings page where you can set up your Polyspace Access cluster.

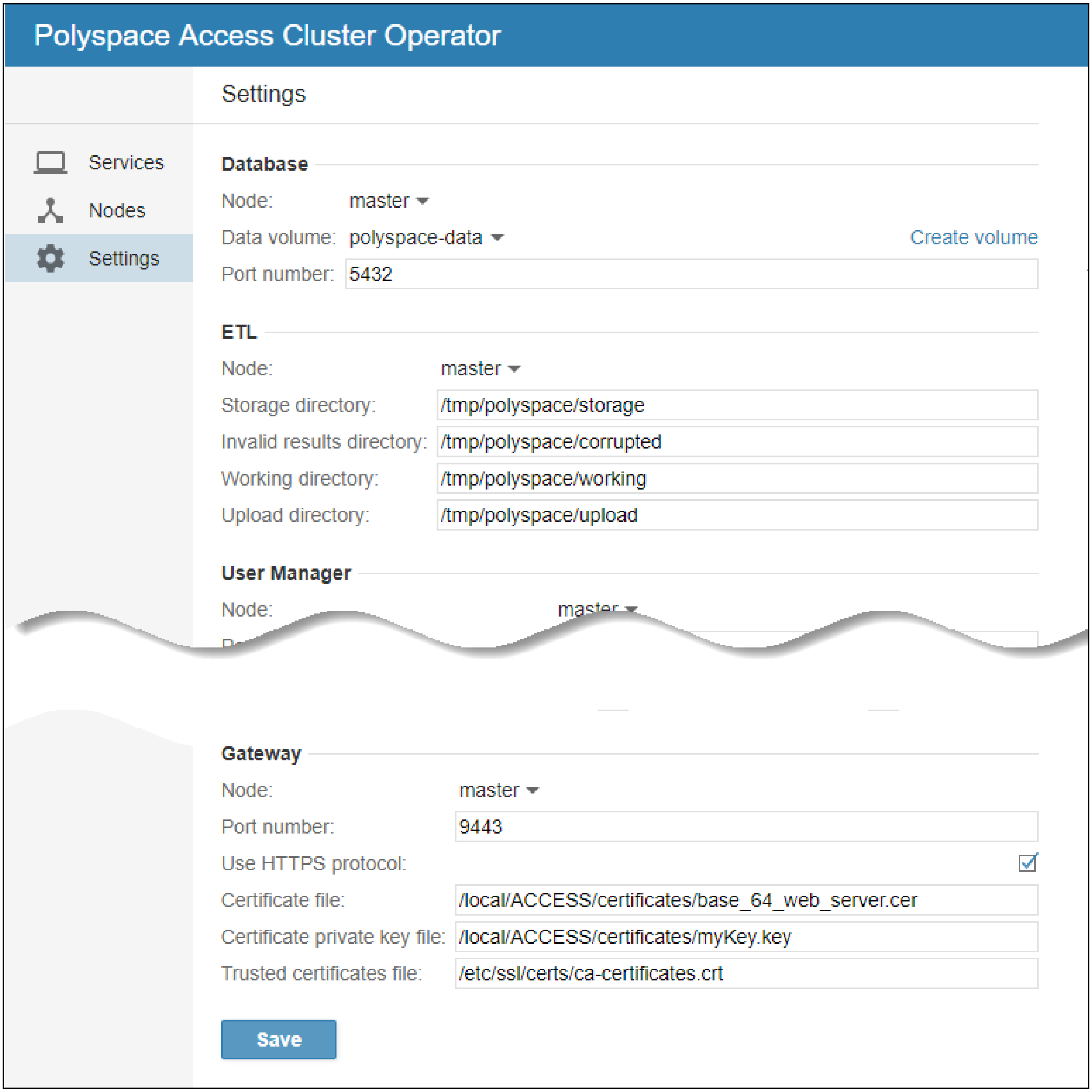
It is best practice to change your COP password after your first login. To set a new password, click the icon in the upper right corner of the web interface and select **Change password**. Share the COP password only with users who configure and manage the Polyspace Access services.

Note

1. Whenever you change the settings, click **Save**, then in the **Services** tab click **PROVISION** for the changes to take effect.
2. On Windows systems, all the file paths in the **Settings** must point to local drives.

Default Installation

The default installation for Polyspace Access uses a configuration that enables you to install and run all the services on the machine where COP is running. The default settings use the HTTPS protocol and the embedded Lightweight Directory Access Protocol (LDAP) for user authentication. See “Configure Polyspace Access Services with HTTPS” on page 1-18 and “Use the Polyspace Access Embedded LDAP” on page 1-22. Complete the configuration of the **User Manager** on page 1-26, **Web Server** on page 1-31, and **Gateway** on page 1-33 services before you start Polyspace Access.



Polyspace Access Cluster Operator

Settings

Database

Node: master

Data volume: polyspace-data

Port number: 5432

ETL

Node: master

Storage directory: /tmp/polyspace/storage

Invalid results directory: /tmp/polyspace/corrupted

Working directory: /tmp/polyspace/working

Upload directory: /tmp/polyspace/upload

User Manager

Node: master

Gateway

Node: master

Port number: 9443

Use HTTPS protocol: ☑

Certificate file: /local/ACCESS/certificates/base_64_web_server.cer

Certificate private key file: /local/ACCESS/certificates/myKey.key

Trusted certificates file: /etc/ssl/certs/ca-certificates.crt

Custom Installation

To install Polyspace Access services on more than one machine, or to specify different data volume, port numbers, and folder paths for the different services, select a custom installation on the installation wizard page. A custom installation on multiple machines enables you to run the **Database**, **ETL**, **User Manager**, **Web Server**,and **Gateway** services on different machines.

For a custom installation on a single machine, you must use the same path for the **Upload directory** of the **ETL** and **Web Server** services.

For an installation on multiple machines, for example two machines with host names `host1.company.com` and `host2.company.com`, follow these steps. The example assumes that COP is already running on `host1.company.com` on port `8083`.

1. Copy the `cop-docker-agent` binary, the `tar` files, and the `settings.json` file from the installation folder on `host1.company.com` to `host2.company.com`.
2. On `host2.company.com`, navigate to the folder where you copied the content from `host1.company.com` and run this command:

   ```
   cop-docker-agent --hostname host2.company.com --operator-host host1.company.com:8083 -node-id new_node
   ```

   The command creates a node `new_node` on `host2.company.com` and associates this node with the instance of COP running on `host1.company.com`.
3. In the web interface of the COP instance running on `host1.company.com` (`http://host1.company.com:8083`), the **Nodes** tab lists two nodes, `master` and `new_node`.

   In the **Settings** tab, set the **Node:** parameter of any service that you want to run on `host2.company.com` to `new_node`. If you want to continue running the service on `host1.company.com`, leave the **Node:** parameter set to `master`.

   If you run the **Database** service on `host2.company.com`, click **Create volume** to create a database mount point on `host2.company.com`. Set the **Data volume:** parameter to the volume that you created.

The paths of the **Upload directory** setting for the **ETL** and **Web Server** services must point to the same hard drive.

Complete the configuration of the **User Manager** on page 1-26, **Web Server** on page 1-31, and **Gateway** on page 1-33 services before you start Polyspace Access.

Configure Polyspace Access Services with HTTPS

The HTTPS protocol is enabled by default to allow the encryption of data transfers between Polyspace Access and client machines, and between the different Polyspace Access services.

When you configure the Polyspace Access services for HTTPS, you provide a certificate and the private key that you used to generate that certificate. It is recommended that you secure your Polyspace Access services private keys by following best practices such as:

- Do not transfer the private key between machines. Instead, generate and store the private key on a local file system.
- Restrict read/write permissions. Grant access to the private key file to only those Polyspace Access administrators who manage the services.
- Establish a policy to periodically check which users have access to the private key file.

To configure the HTTPS protocol for the **User Manager**, **Web Server**, and **Gateway** services, complete these fields.

- **Certificate file:** Full path to the signed server certificate `PEM` file. On Windows systems, the path must point to a local drive.
- **Certificate private key file:** Full path to the private key `PEM` file you used to generate the certificate file. On Windows systems, the path must point to a local drive.
- **Trusted certificates file:** Full path to the certificate store where you store trusted certificate authorities. For instance, on a Linux Debian distribution, `/etc/ssl/certs/ca-certificates.crt` . If you use self-signed certificates, use the same path you specify for **Certificate file:**.

If you do not want to use HTTPS, clear **Use HTTPS protocol** for the **User Manager**, **Web Server**, and **Gateway** services.

Use Certificates Signed by a Certificate Authority

These steps illustrate how to configure SSL encryption on a Debian Linux system by using your organization's certificate authority and the `openssl` utility.

1. Create a certificate signing request. In the `CN` field (common name), specify *`hostName`*, the fully qualified domain name (FQDN) of the machine where you run the Polyspace Access service you are configuring. This must match the host name you specified when you started the `cop-docker-agent` binary. If you run the **User**

Manager, **Web Server**, and **Gateway** services on different machines, you must request a certificate with the correct fully qualified domain name (FQDN) for each service.

```
openssl req -new -newkey rsa:4096 -nodes -out myRequest.csr -keyout myKey.key \
-subj "/C=US/ST=/L=/O=/CN=hostName"
```

The command outputs a private key file `myKey.key` and the file `myRequest.csr`, which contains a public key and data that describes your server.

2. Submit `myRequest.csr` to your organization's certificate authority. The certificate authority uses the file to generate a signed server certificate. For instance, `server_cert.cer`.
3. Complete the configuration using the private key file, the signed server certificate, and the certificate store where you store the certificate that identifies your certificate authority.

Certificate file:	Full path to the server certificate you obtained in step 2.
Certificate private key file:	Full path to the private key file you generated in step 1.
Trusted certificates file:	`/etc/ssl/certs/ca-certificates.crt`

When you open Polyspace Access in a web browser, it considers the connection secure if the browser uses the certificate store that you specify for **Trusted certificates file:**.

Use Self-Signed Certificates

Alternatively, use self-signed certificates. Because the identity of your server is not certified by a certificate authority, your browser might consider the connection to your server as untrusted. To generate a self-signed certificate with `openssl`, use this command:

```
openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:4096 -keyout self-key.pem \
-out self-cert.pem -subj "/C=US/ST=/L=/O=/CN=hostName"
```

In the `CN` field (common name), specify *`hostName`*, the fully qualified domain name (FQDN) of the machine where you run the Polyspace Access service you are configuring. This must match the host name you specified when you launched the `cop-docker-agent` binary. If you run the **User Manager**, **Web Server**, and **Gateway** services on different machines, you must generate a self-signed certificate with the correct fully qualified domain name (FQDN) for each service.

The command outputs files `self-key.pem` and `self-cert.pem`. Use these files to complete the configuration.

Certificate file:	`*fullPathTo*/self-cert.pem`
Certificate private key file:	`*fullPathTo*/self-key.pem`
Trusted certificates file:	`*fullPathTo*/self-cert.pem`

fullPathTo is the full file path.

Configure Lightweight Directory Access Protocol

Use Your Organization LDAP

Polyspace Access authenticates users by checking for valid user names and passwords against information from the LDAP server. To use the LDAP server of your organization, clear **Use embedded LDAP** in the COP **User Manager** settings. Contact your network administrator for the LDAP URL, base, and any other setting.

LDAP URL	You must enter the LDAP URL as `ldap://`*HOST*`:`*PORT*, where *HOST* is the LDAP host. If you have configured your LDAP server over SSL, enter the URL as `ldaps://`*HOST*`:`*PORT*. For additional LDAPS configuration steps, see “Configure the User Manager for LDAP over SLL” on page 1-21. Because communications between the LDAP server and clients are not encrypted, the configuration and use of LDAP over SSL (LDAPS) is recommended.
LDAP username:	User name of user with read permission to the LDAP server. Leave this field blank if your access to the LDAP server is not password-protected.

LDAP password:	Password of user with read permission to the LDAP server. Leave this field blank if your access to the LDAP server is not password-protected. The password is stored in the `settings.json` file. For added security, set restrictions on the read and write permissions for this file. By default this file is stored in the same folder as the COP binary.
LDAP base	You can retrieve this parameter by using an LDAP explorer tool. For instance, connect to your LDAP server with Apache Directory Studio and open the properties for your connection. In the **Browser Options**, click **Fetch Base DNs** to get the LDAP base.
LDAP search filter:	Use the search filter to retrieve a subset of users from the LDAP database. Polyspace Access loads this subset on start up instead of all users in your organization. Loading a smaller number of users for authentication improves the performance of Polyspace Access. Specify the search filter as *`attribute=value`*, for instance `CN=test*` matches all users that have a common name (CN) attribute that starts with "test". Use parentheses to combine multiple filter expression in an AND (&) or OR(|) clause. For instance `(|(CN=jdoe)(department=foo))` matches all users that have CN attribute "jdoe" or department attribute "foo". The default search filter is `objectClass=organizationalPerson`. For more information on search filters, see the LDAP filters.
Other LDAP settings	Leave these settings unchanged unless instructed otherwise by your LDAP administrator.

Configure the User Manager for LDAP over SLL

If you use a LDAP configured over SSL (LDAPS), add the LDAPS certificate to the **Trusted certificates file:** you specify in COP settings for the **User Manager** service.

For instance, on a Linux Debian distribution:

- If you configure the **User Manager** service for HTTPS with a certificate authority:
  - If the LDAPS certificate is already part of the certificate trust store file, and you point to `/etc/ssl/certs/ca-certificates.crt` in the COP settings, no action is required.
  - If the LDAPS certificate is not part of the certificate trust store file, concatenate `/etc/ssl/certs/ca-certificates.crt` and the LDAPS certificate and point to the resulting file in **Trusted certificates file:**. For example, for a LDAPS certificate file `ldaps_cert.pem`, use this command.

    ```
    cat /etc/ssl/certs/ca-certificates.crt \
       /local/ldaps/ssl/ldaps_cert.pem > \
        /local/access/ssl/trusted-certificates.pem
    ```

    Point to `/local/access/ssl/trusted-certificates.pem` for the **Trusted certificates file:** setting.
- If you configure the **User Manager** service using self-signed certificates:
  - Concatenate the certificate you use to configure HTTPS for the **User Manager** service with the LDAPS certificate, or with `/etc/ssl/certs/ca-certificates.crt` if the LDAPS certificate is in the certificates trust store. For example, for a LDAPS certificate file `ldaps_cert.pem`, and a self-signed certificate `self-cert.pem`, use this command.

    ```
    cat /local/access/self_cert.pem \
       /local/ldaps/ssl/ldaps_cert.pem > \
        /local/access/ssl/trusted-certificates.pem
    ```

    Point to `/local/access/ssl/trusted-certificates.pem` for the **Trusted certificates file:** setting.
- If you do not configure the **User Manager** service for HTTPS, you must still point to the LDAPS certificate in the **Trusted certificates file:** setting, or to `/etc/ssl/certs/ca-certificates.crt` if the LDAPS certificate is in the certificates trust store .

Use the Polyspace Access Embedded LDAP

If you cannot or choose not to use the LDAP server of your organization, you can authenticate user logins by using the Polyspace Access embedded LDAP. Select **Use embedded LDAP** in the COP settings. The embedded LDAP checks user logins against credentials that are stored in an `LDIF` file.

This example shows how to create an `LDIF` file and add new users.

1. Copy the template file to a text editor and save it on your system as `embedded_ldap.ldif`

   Template File

   ```
   dn: dc=customer,dc=mathworks,dc=com
   objectclass: top
   objectclass: domain
   objectclass: extensibleObject
   dc: customer


   ## BEGIN entry
   dn: uid=admin,dc=customer,dc=mathworks,dc=com
   objectclass: inetOrgPerson
   # 'common' name
   cn: admin
   # user id
   uid: admin
   mail: admin@invalid.com
   userPassword: pass
   displayName: admin
   # surname
   sn: admin
   ##END entry. Leave one empty line before next entry.
   ```

   The template defines a single user with user name `admin` and password `pass`.

2. To create a user, copy all the lines from `##BEGIN entry` to `##END entry...` and paste them below the last entry in the file.

3. Enter a new user name in the fields for **`dn: uid=`**, **`cn:`** , and **`uid:`** , and provide a new password. For instance, the entry for user `jsmith` with password `new_pass` is:

   ```

   ## BEGIN entry
   dn: uid=jsmith,dc=customer,dc=mathworks,dc=com
   objectclass: inetOrgPerson
   # 'common' name
   cn: jsmith
   # user id
   uid: jsmith
   mail: jsmith@invalid.com
   userPassword: new_pass
   displayName: John
   # surname
   sn: Smith
   ##END entry. Leave one empty line before next entry.
   ```

   You can optionally provide values for the `mail`, `displayName`, and `sn` fields.

4. Complete the configuration in COP.

LDIF file:	*fullPathTo*/embedded_ldap.ldif *fullPathTo* is the full file path. On Windows systems, the path must point to a local drive.
LDAP base	dc=customer,dc=mathworks,dc=com
Other LDAP settings	Leave these settings unchanged unless you are familiar with `LDIF` file templates.

The passwords you store in the `LDIF` file are not encrypted. Restrict read/write permissions for this file to only the administrator who edits the file.

Manage LDAP Users in Polyspace Access

When the **User Manager** service starts, Polyspace Access loads a list of authorized users to its database from your organization's LDAP database or, if you use the embedded LDAP, from the LDIF file that you specify. You can select users only from the list of authorized users when assigning analysis findings or managing permission for a project.

Polyspace Access does not update its database of authorized users if you remove a user from the LDAP database or from the LDIF file, even if you restart the **User Manager** service.

To update the list of authorized users in Polyspace Access, follow these steps. You must be logged in as a user with **Administrator** privileges. To set a user as **Administrator**, see “Configure the Web Server and Gateway” on page 1-31.

1. Restart the **User Manager** service to load the most recent list of authorized users. At the command line, enter:

   ```
   docker restart polyspace-usermanager
   ```

2. In a web browser, enter the URL that you use to “Open the Polyspace Access Web Interface” on page 1-44 and append `/users/list/removed` to the URL, for example `https://access-machine.company.com:9443/users/list/removed`.
3. Select the user names that you want to remove from the Polyspace Access database of authorized users and click **Confirm clean-up**. To select multiple users, hold down the **CTRL** key while you click.

See Also

More About

- “Start Polyspace Access and Upload Examples” on page 1-40

Configure the User Manager

User Manager

Setting	Description
Node:	Use the drop-down list to select the node on which your run this service. To create nodes and run the service on a different machine, see “Custom Installation” on page 1-17.
Port number:	See “Network Port Configuration” on page 1-7.
Use HTTPS protocol:	This option is enabled by default. See “Configure Polyspace Access Services with HTTPS” on page 1-18. Use the same HTTPS protocol settings for the **User Manager**, **Web Server**, and **Gateway** services.
Certificate file:	If you select **Use HTTPS protocol:**, these fields are required.
Certificate private key file:	
Trusted certificates file:	
Use embedded LDAP:	This option is enabled by default. See “Use the Polyspace Access Embedded LDAP” on page 1-22. If you disable this option, see “Use Your Organization LDAP” on page 1-20
LDIF file: or **LDAP URL:**	To configure these settings, see “Use the Polyspace Access Embedded LDAP” on page 1-22 if **Use embedded LDAP:** is enabled or “Use Your Organization LDAP” on page 1-20 otherwise.
LDAP base:	
LDAP search filter:	
Other LDAP settings	

Setting	Description
Authentication token expiration (sec):	Specify in seconds the period of validity of the signed JSON Web Tokens that the **User Manager** issues to authenticated users. This expiration time determines the lifetime of a session. Once you log in to Polyspace Access, your license is checked out and your session refreshes periodically to keep it from expiring. The session ends once you explicitly log out or close your web browser and your license is checked back in. If your browser crashes, your license stays checked out until the session expires. When you set the expiration time, consider: • If the expiration time is too short, frequent users will be prompted to log back in frequently. On large teams, the license server experiences a high volume of license checkins and checkouts. • If the expiration time is too long, the session time of less frequent users might be overestimated in the license logs.
Authentication issuer:	Service that issues the signed JSON Web Tokens.
Authentication private key file:	Specify the full path to the private key `PEM` file that the **User Manager** uses to sign JSON Web Tokens. On Windows systems, the paths must point to local drives. The **User Manager** service does not support password-protected private keys. You can generate a private key by using the `openssl` utility. For example: `openssl genrsa -out private.pem 2048` Restrict access to this private key to only those administrators who manage the **User Manager** service.

Setting	Description
Authentication public key file:	Specify the full path to the public key `PEM` file that the **User Manager** uses to sign JSON Web Tokens. You generate this file from the private key file that you specify for **Authentication private key file:**. On Windows systems, the paths must point to local drives. You can generate a public key pair from private key `private.pem` by using the `openssl` utility. For example: `openssl rsa -in private.pem -outform PEM -pubout -out public.pem`

See Also

More About

- “Configure the Database and ETL Services” on page 1-29
- “Configure the Web Server and Gateway” on page 1-31
- “Register Polyspace Desktop User Interface” on page 1-36
- “Start Polyspace Access and Upload Examples” on page 1-40

Configure the Database and ETL Services

Database

Setting	Description
Node:	Use the drop-down list to select the node on which your run this service. To create nodes and run the service on a different machine, see "Custom Installation" on page 1-17.
Data volume:	Select an existing data volume or create one for the database mount point. Deleting the **Database** service and uninstalling Polyspace Access does not erase the results that you uploaded to the database from the data volume. To delete a data volume and its content, click **Nodes** in the left pane, click the appropriate node ID, select the volume, and then click **DELETE**.
Port number:	See "Network Port Configuration" on page 1-7.

ETL

Setting	Description
Node:	Use the drop-down list to select the node on which your run this service. To create nodes and run the service on a different machine, see "Custom Installation" on page 1-17.
Storage directory:	Specify the full path to folders with adequate write permissions. On Windows systems, the paths must point to local drives. See "Storage Configuration" on page 1-5. The **Upload directory:** path must be the same for the **Web Server** and **ETL** services if they are running on the same machine. If the services are running on different machines, the paths must point to the same hard drive.
Invalid results directory:	
Working directory:	
Upload directory:	

See Also

More About

- "Configure the User Manager" on page 1-26
- "Configure the Web Server and Gateway" on page 1-31
- "Register Polyspace Desktop User Interface" on page 1-36
- "Start Polyspace Access and Upload Examples" on page 1-40

Configure the Web Server and Gateway

Web Server

Setting	Description
Node:	Use the drop-down list to select the node on which your run this service. To create nodes and run the service on a different machine, see "Custom Installation" on page 1-17.
Port number:	See "Network Port Configuration" on page 1-7.
Use HTTPS protocol:	This option is enabled by default. See "Configure Polyspace Access Services with HTTPS" on page 1-18. Use the same HTTPS protocol settings for the **User Manager**, **Web Server**, and **Gateway** services.
Certificate file:	If you select **Use HTTPS protocol:**, these fields are required.
Certificate private key file:	
Trusted certificates file:	
Upload directory:	Specify the full path to folders with adequate write permissions. On Windows systems, the paths must point to local drives. See "Storage Configuration" on page 1-5. The **Upload directory:** path must be the same for the **Web Server** and **ETL** services if they are running on the same machine. If the services are running on different machines, the paths must point to the same hard drive.
Temporary upload directory:	
License file:	Specify the full path to the `network.lic` license file that you obtained with Polyspace Access. On Windows systems, the paths must point to local drives. See "License Management" on page 1-35

Setting	Description
Jira URL:	If you use the Jira software as part of your development process, enter the URL of the JIRA instance for your organization, for example `https://jira.mycompany.com`. This setting enables the creation of tickets in JIRA from the Polyspace Access interface. If your JIRA instance is configured with HTTPS, see "Add JIRA Instance Configured with HTTPS" on page 1-33.
Authorization administrators:	Enter a comma-separated list of user names to set those users as administrators. A user with administrator privileges can: • manage project permissions on page 2-20 for all projects in the Polyspace Access database. • Remove or add project owners. • "Manage LDAP Users in Polyspace Access" on page 1-24 To remove an administrator: 1 Remove the user name from the list, save your changes, then provision and restart Polyspace Access. 2 After the restart, a Polyspace Access administrator must unassign the user from all top-level folders in the **PROJECT EXPLORER** in the web interface by using the context menu. The administrator can also perform this task at the command line by using the `-unset-role` flag with the `polyspace-access` binary. For more information, see `polyspace-access -unset-role -h`.

Gateway

Setting	Description
Node:	Use the drop-down list to select the node on which your run this service. To create nodes and run the service on a different machine, see “Custom Installation” on page 1-17.
Port number:	See “Network Port Configuration” on page 1-7. This number is the port number that you specify in the URL you use to open the Polyspace Access web interface.
Use HTTPS protocol:	This option is enabled by default. See “Configure Polyspace Access Services with HTTPS” on page 1-18. Use the same HTTPS protocol settings for the **User Manager**, **Web Server**, and **Gateway** services.
Certificate file:	If you select **Use HTTPS protocol:**, these fields are required.
Certificate private key file:	
Trusted certificates file:	

Add JIRA Instance Configured with HTTPS

If your JIRA instance is configured with HTTPS, add the JIRA certificate to the **Trusted certificates file:** you specify in COP settings for the **Web Server** service.

For instance, on a Linux Debian distribution:

- If you configure the **Web Server** service for HTTPS with a certificate authority:
  - If the JIRA certificate is already part of the certificate trust store file, and you point to `/etc/ssl/certs/ca-certificates.crt` in the COP settings, no action is required.
  - If the JIRA certificate is not part of the certificate trust store file, concatenate `/etc/ssl/certs/ca-certificates.crt` and the JIRA certificate and point to the resulting file in **Trusted certificates file:**. For example, for a JIRA certificate file `jira_cert.pem`, use this command.

```
cat /etc/ssl/certs/ca-certificates.crt \
   /local/jira/ssl/jira_cert.pem > \
    /local/access/ssl/trusted-certificates.pem
```

Point to `/local/access/ssl/trusted-certificates.pem` for the **Trusted certificates file:** setting.

- If you configure the **Web Server** service using self-signed certificates:

  - Concatenate the certificate you use to configure HTTPS for the **Web Server** service with the JIRA certificate, or with `/etc/ssl/certs/ca-certificates.crt` if the JIRA certificate is in the certificates trust store. For example, for a JIRA certificate file `jira_cert.pem`, and a self-signed certificate `self-cert.pem`, use this command.

    ```
    cat /local/access/self_cert.pem \
       /local/jira/ssl/jira_cert.pem > \
        /local/access/ssl/trusted-certificates.pem
    ```

    Point to `/local/access/ssl/trusted-certificates.pem` for the **Trusted certificates file:** setting..

- If you do not configure the **Web Server** service for HTTPS, you must still point to the JIRA certificate in the **Trusted certificates file:** setting, or to `/etc/ssl/certs/ca-certificates.crt` if the JIRA certificate is in the certificates trust store .

See Also

More About

- “Configure the User Manager” on page 1-26
- “Configure the Database and ETL Services” on page 1-29
- “Register Polyspace Desktop User Interface” on page 1-36
- “Start Polyspace Access and Upload Examples” on page 1-40

License Management

To configure and manage Polyspace Access licenses, you must install the FlexNet® Publisher license manager program. You use the MathWorks® installer to perform this installation. See License Manager Installation and Management to install the license manager and configure the licenses.

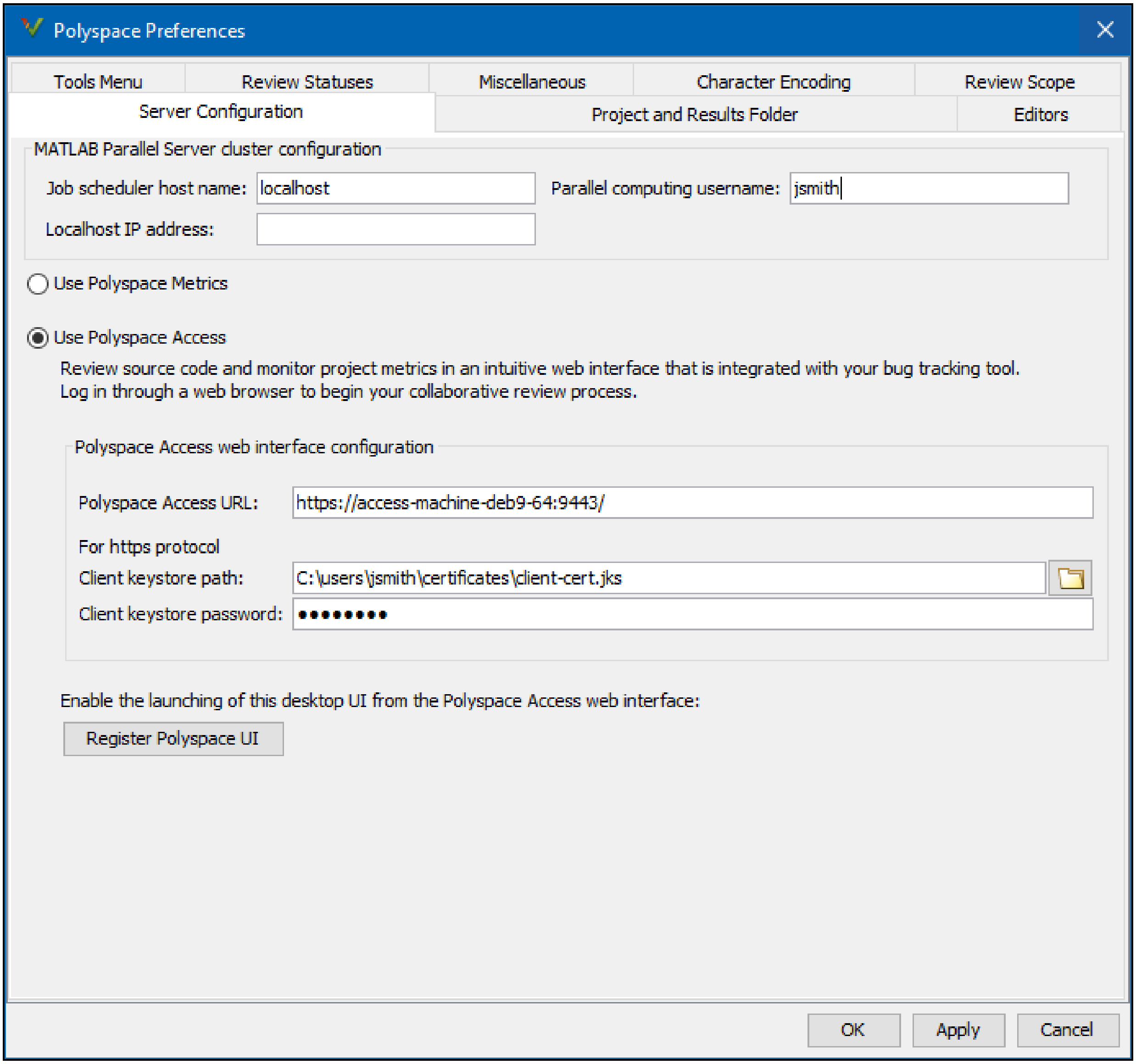
Each licensed Polyspace Access user can log in to up to 5 concurrent sessions. These actions define a session:

- Log in to the Polyspace Access web interface.
- View results stored on Polyspace Access from the desktop interface.
- Upload results to Polyspace Access from the desktop or command-line interface.
- Perform an operation from the command-line interface that requires a Polyspace Access login.
- Generate a report for results stored on Polyspace Access.

To review or generate reports for results that were generated with Polyspace Code Prover or Polyspace Ada products, and that are stored on Polyspace Access, you need a Polyspace Code Prover™ Access license.

Register Polyspace Desktop User Interface

To enable interaction between a Polyspace desktop user interface and Polyspace Access, start the desktop interface and go to **Tools** > **Preferences**.



MATLAB Parallel Server cluster configuration

Job scheduler host name: localhost

Parallel computing username: jsmith

Localhost IP address:

○ Use Polyspace Metrics

● Use Polyspace Access

Review source code and monitor project metrics in an intuitive web interface that is integrated with your bug tracking tool.
Log in through a web browser to begin your collaborative review process.

Polyspace Access web interface configuration

Polyspace Access URL: https://access-machine-deb9-64:9443/

For https protocol

Client keystore path: C:\users\jsmith\certificates\client-cert.jks

Client keystore password: ●●●●●●●●●

Enable the launching of this desktop UI from the Polyspace Access web interface:

In the **Server Configuration** tab, complete these fields:

- **Polyspace Access URL**: `https://`*`hostName`*`:`*`port`*, where *`hostName`* is the fully qualified domain name (FQDN) of the machine on which the **Gateway** service is running. *`port`* is the **Port number** value specified in the **Gateway** COP settings.

  If you did not select **Use HTTPS protocol** in the **Gateway** COP settings, replace `https` with `http`.
- **Client keystore path**: path to the key store where you imported the **Gateway** signed certificate. See “Generate Client Keystore for Desktop Interface” on page 1-38.

  If you did not select **Use HTTPS protocol** in the **Gateway** COP settings, leave this field blank.
- **Client keystore password**: The password associated with the key store file.

  If you did not select **Use HTTPS protocol** in the **Gateway** COP settings, leave this field blank.

To associate your Polyspace desktop interface with Polyspace Access, click **Register Polyspace UI**, click **OK**, and then close and restart the desktop interface for the changes to take effect. From the Polyspace Access web interface, you can now start the desktop interface and view currently opened results.

Once you restart the desktop interface, select **Access** to:

- Open the Polyspace Access web interface.
- Open analysis results from the Polyspace Access database.
- Upload analysis results to the Polyspace Access database.

Note In Linux, the desktop interface must already be open before you can view results currently open in Polyspace Access.

Generate Client Keystore for Desktop Interface

If you configure the Polyspace Access **Gateway** service to use the HTTPS protocol, you must generate a Java® Key Store (JKS) file to enable communications between the desktop interface and Polyspace Access. You import the signed certificate that you used to configure HTTPS for the **Gateway** service to the JKS file. See “Configure the Web Server and Gateway” on page 1-31.

To generate the `jks` file, use the `keytool` key and certificate management utility. To use `keytool` you must have Java Platform, Standard Edition Development Kit (JDK) installed on your machine. `keytool` is available from the Java installation folder, for instance:

- `C:\Program Files\Java\jdk1.8.0_181\bin\keytool.exe` on Windows.
- `/usr/bin/keytool` or `%JAVA_HOME%/bin/keytool` on UNIX/Linux.

For example, if you used signed certificate `gateway-cert.cer` to configure HTTPS for the **Gateway** service, generate the corresponding `JKS` file by using this command:

```
keytool -import -trustcacerts -alias cert -file gateway-cert.cer -keystore client-cert.jks -storepass password
```

The command outputs file `client-cert.jks`. The password associated with this key store file is `password`.

See Also

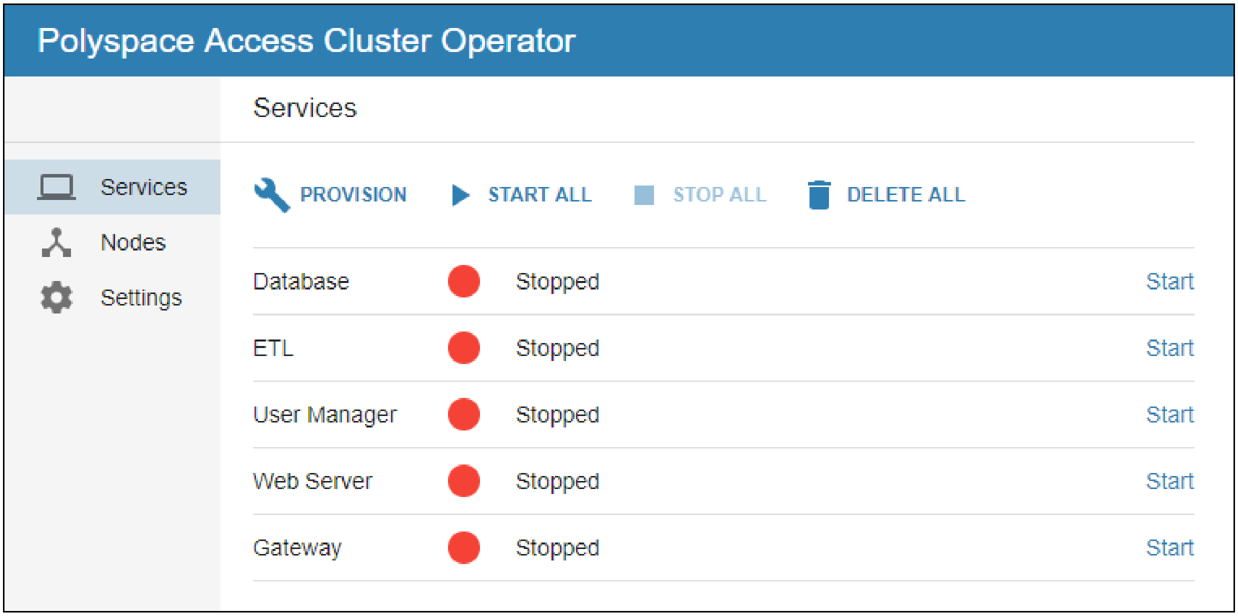
More About

- "Upload Results from the Desktop Interface" on page 1-42

Start Polyspace Access and Upload Examples

To complete the installation, start the Polyspace Access services and upload some examples to the Polyspace Access database. Before you begin, make sure that you have configured the Polyspace Access services. See “Configure Polyspace Access Services” on page 1-13.

After you enter the settings parameters, click **Save**. From the left pane, go to the **Services** view and click **PROVISION**. COP loads and installs Docker images for the **Database**, **ETL**, **User Manager**, **Web Server**, and **Gateway** services. If the installation is successful, all the services are stopped and show a red indicator.



Database	🟥	Stopped	Start
ETL	🟥	Stopped	Start
User Manager	🟥	Stopped	Start
Web Server	🟥	Stopped	Start
Gateway	🟥	Stopped	Start

To start the services, click **START ALL**. The indicator turns green when a service starts. The **Web Server** might take a few moments to start even after the indicator turns green.

Once you complete the installation, you can stop the COP agent at the command line by pressing **Ctrl+C**.

Note If one of the services starts and stops after a short time, try restarting the service and, at the command line, enter:

```
docker logs -f polyspace-service
```

`service` corresponds to `db` (database), `etl`, `usermanager`, `web-server`, or `gateway`. Use the output log to try to identify the cause of the stopped service.

Configure Polyspace Access to Restart Automatically

Update the Docker restart policy to configure Polyspace Access for automatic restarts after an unexpected system shutdown or a reboot.

To update the Docker restart policy, at the command line, enter:

```
docker update --restart always polyspace-gateway polyspace-web-server polyspace-etl polyspace-db polyspace-usermanager
```

The Docker daemon will always attempt to restart the specified Polyspace Access containers when they stop. If you stop a container manually, the container restarts only when the Docker daemon restarts or when you restart the container manually.

To prevent a restart loop, the restart policy applies only to containers that have started successfully. For more details, see the Docker documentation.

Upload Examples

Upload Results from the Command line

To upload the examples provided with your Polyspace Bug Finder Server or Polyspace Code Prover Server installation, from the command line, go to the *`polyspaceroot`*`\polyspace` and run these commands:

```
bin\polyspace-access -host hostname -port port^
-upload examples\cxx\Bug_Finder_Example\Module_1\BF_Result

bin\polyspace-access -host hostname -port port^
-upload examples\cxx\Bug_Finder_Example_Cpp\Module_1\BF_Result

bin\polyspace-access -host hostname -port port^
-upload examples\cxx\Code_Prover_Example\Module_1\CP_Result

bin\polyspace-access -host hostname -port port^
-upload examples\cxx\Code_Prover_Example_Cpp\Module_1\CP_Result
```

`polyspaceroot` is the path to your R2019a Polyspace installation. *`hostname`* is the fully qualified domain name (FQDN) of the machine hosting the **Gateway** service. *`port`* is the **Port number** value specified for the **Gateway** setting in COP. For more information on uploading results from the command line, see the documentation for Polyspace Bug Finder Server or Polyspace Code Prover Server.

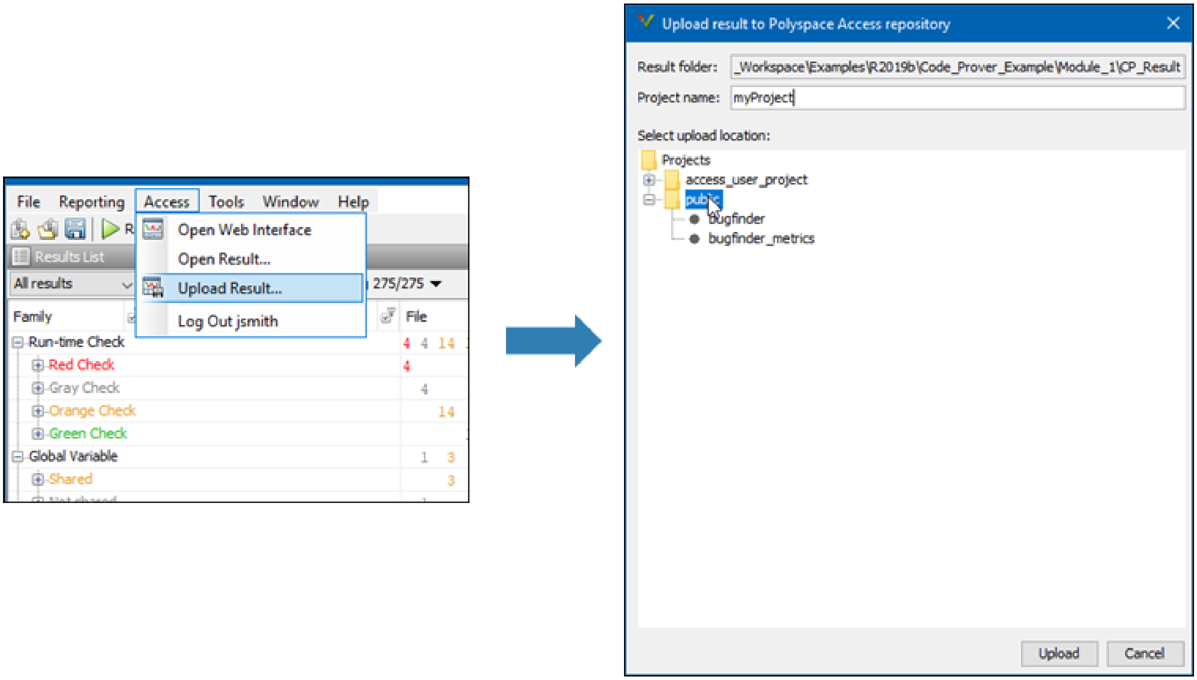
After each command, you are prompted to enter your login and password. Enter the credentials that you use to log into Polyspace Access.

You cannot use the command line to upload results from a Polyspace Desktop product analysis to the Polyspace Access database.

Upload Results from the Desktop Interface

To upload the demo examples provided with your Polyspace Bug Finder or Polyspace Code Prover:

1. Open an example in the desktop interface and select the results in the **Project Browser** pane or switch to the **Results List** pane.
2. From the menu, click **Access** > **Upload Results**. If you are prompted to log in, use your Polyspace Access credentials.
3. In the **Upload results to Polyspace Access repository** window, click a folder to select an upload location, then click **Upload**. You can optionally rename the project.



Result folder: _Workspace\Examples\R2019b\Code_Prover_Example\Module_1\CP_Result

Project name: myProject

You can also upload to the Polyspace Access database by selecting a result in the **Project Browser** pane and using the context menu.

You must configure the desktop interface to communicate with Polyspace Access. See “Register Polyspace Desktop User Interface” on page 1-36.

After you upload results to Polyspace Access:

- If you open a local copy of the results in the Desktop interface, you cannot make changes to the **Status**, **Severity**, or comment fields.
- To make changes to the **Status**, **Severity**, or comment fields, open the results from Polyspace Access by going to **Access** > **Open Results**.

  Once you save the changes you make to these fields in the desktop interface, the changes are reflected in the Polyspace Access web interface.

Open the Polyspace Access Web Interface

You can now open Polyspace Access from any machine connected to the server hosting the Polyspace **Gateway** service. If you have enabled the HTTPS protocol, from your web browser, go to `https://`*`hostName`*`:`*`port`*

`hostName` is the fully qualified domain name (FQDN) of the web server host. *`port`* is the **Port number** value specified in the COP settings for the **Gateway** service. For instance, `https://access-machine.company.com:9443`.

If you did not select **Use HTTPS protocol**, replace `https` with `http` in the address bar of your browser.

See Also

More About

- “Review Results in Polyspace Bug Finder Access”

Database Backup

There are two recommended methods to create a backup of your Polyspace Access database. Both methods rely on PostgreSQL utilities. The first method creates a dump of your database, or database cluster, and enables you to recover the database from when the dump was created. The second method uses write ahead logs (WAL) generate by PostgreSQL. It allows incremental backups and recovery.

Based on your database size and frequency of use, establish a policy for how often you perform a backup and which backup method is the most adequate.

Database Backup

To back up a particular database, use the `pg_dump` utility. The utility generates a list of SQL commands that allow you to reconstruct your database. To back up all the databases in your database cluster, use `pg_dumpall`. The backup operations require superuser privileges. The privileges are set through postgre and are separate from the user privileges on your system. To ensure that your backup does not contain partial or corrupted data, stop the **ETL** and **Web Server** services before starting the backup operation.

For example, to back up database `my_db` from the **Database** service running on machine `host1` on port `1234`, use this command:

```
pg_dump -U postgres -h host1 -p 1234 my_db > db_backup
```

The `-U` specifies superuser `postgres`. The command outputs a text file `db_backup`.

You can then create a database `new_db` by using template `template0` and restore from `db_backup`.

```
createdb -U postgres -h host1 -p 1234 -T template0 new_db
psql -U postgres -h host1 -p 1234 new_db < db_backup
```

To back up all the databases of the **Database** service, use `pg_dumpall`. Use pipes to load your backup directly to a different server:

```
pg_dumpall -U postgres -h host1 -p 1234 | \
psql -U postgres -h host2 -p 5678 postgres
```

Before you execute this command, the **Database** service on `host2` at port `5678` must exist and the service must be running. For more examples on creating database dumps, see SQL Dump.

Note Using `pg_dumb` or `pg_dumpall` with large databases might generate files that exceed the maximum file size limit on some operating systems and can be time consuming.

Alternatively, you can rely on WAL files to perform incremental backups and recoveries of your database. The WAL records all changes made to the database. The system usually stores only a few WAL files and recycles older files.

By creating a base backup and storing all subsequent WAL files, you can restore your database by replaying the WAL sequence up to any point between when you made your base backup and the present. For an example of how to configure an incremental backup, see Continuous Archiving and Point-in-Time Recovery (PITR).

See Also

More About

- “Storage Configuration” on page 1-5
- “Manage Polyspace Bug Finder Access Software”

Database Clean Up

To optimize the performance of the Polyspace Access database, perform regular database clean up operations such as vacuuming and the deletion of old or obsolete projects. It is recommended that you back up on page 1-45 your database before you perform a clean up operation.

Perform Database Vacuuming

When a row is updated or deleted in a database table, it is not physically removed from the table because other database transactions might still use the old version of the row. To reclaim the disk space of old rows that are no longer used by any database transaction, use the PostgreSQL `vacuumdb` command. Vacuuming the database regularly prevents your database disk space from growing too large or fragmented.

Before you perform a vacuum operation, ensure that no users are connected to Polyspace Access then stop the **Web Server** and **ETL** services. To stop the services, go to the **Services** tab on the cluster operator (COP) web UI and click **Stop** next to the service you want to stop. You can also stop the services by opening a terminal on the server hosting these services and entering:

```
docker stop polyspace-etl
docker stop polyspace-web-server
```

To vacuum your Polyspace Access database, open a terminal on the server hosting your database and enter:

```
docker exec polyspace-db vacuumdb -U postgres --no-password --host hostName --port port prs_data
```

hostName is the host name of the machine hosting the **Database** service. *port* is the **Port number** value specified for the **Database** setting in the cluster operator.

You can also run the `vacuumdb` command and use the `--analyze` option to update the PostgreSQL server statistics. Open a terminal on the server hosting your database and enter:

```
docker exec polyspace-db vacuumdb -U postgres --no-password --host hostName --port port --analyze prs_data
```

Accurate server statistics help prevent degradations in the performance of the database.

To minimize the size of your database tables and return unused space to the operating system, run `vacuumdb` by using the `--full` option. Open a terminal on the server hosting your database and enter:

```
docker exec polyspace-db vacuumdb -U postgres --no-password --host hostName --port port --full prs_data
```

This operation can take a long time and writes a new version of the table that does not have any empty spaces. When you perform a full vacuum, no other database process can run in parallel. The database is not accessible during a full vacuum.

Establish a policy for how often you want to perform a regular and a full vacuum. For instance perform a regular vacuum weekly.

After you complete the vacuum operation, restart the **Web Server** and **ETL** services from the **Services** tab of the COP web UI, or in a terminal on the server hosting the services by entering:

```
docker start polyspace-etl
docker start polyspace-web-server
```

It might take a few moments after you restart the **Web Server** service before you can open Polyspace Access in your web browser.

Delete Outdated Projects

When users delete a project from the **PROJECT EXPLORER** of the Polyspace Access web interface, the project moves to the **ProjectsWaitingForDeletion** folder. The project, including all the runs that you uploaded to the project, remains stored in the database until you explicitly delete it. The **ProjectsWaitingForDeletion** folder is visible only to Polyspace Access users that have the role of **Administrator**.

Define a policy for how often you delete older results from the database. You can automate this operation by using a script. You can delete results even if the results are not in the **ProjectsWaitingForDeletion** folder.

To automate the removal of old project runs or entire projects from your database, use a script that you save as a `.pscauto` file. You run the script by moving the `.pscauto` file to the **Storage directory** of the **ETL** service. Only a user with write privileges on the **Storage directory** can perform this operation.

- To delete project runs from a project but not the project itself, use the `clean_project` command. Specify the project path and one of these options.

  - `clean_project` *`projectPath`* `DATE YYYY-MM-DD`

    Results uploaded before the given date are deleted.
  - `clean_project` *`projectPath`* `MAXRUNS NNN`.

    `NNN` is an integer. If you upload more than `NNN` runs to a project, only the `NNN` most recent runs are not deleted. To delete all the project runs, use `MAXRUNS 0`.
  - `clean_project` *`projectPath`* `AGE D`

    `D` is the number of days. To remove recently uploaded results, use this option. All results runs that are older than `D` days are deleted.
- To completely delete a project from the Polyspace Access database, use the `delete_project` command and specify the project path:

  ```
  delete_project projectPath
  ```

`projectPath` is the full project path in the Polyspace Access **PROJECT EXPLORER**. To get a project path, use the context menu in the **PROJECT EXPLORER** or, at the command line, use the `-list-project` flag with the `polyspace-access` binary. For more information, see `polyspace-access -h -list-project`. If the project path contains spaces or special characters, enclose the project path in double quotes.

For example, to perform a one-time cleanup of project `public/foo` and remove all results uploaded before a specific date, save this command to a `.pscauto` file, for instance `cleanup.pscauto`:

```
clean_project public/foo DATE 2018-09-01
```

Move the file to the **Storage directory** of the **ETL** service:

```
mv cleanup.pscauto /tmp/polyspace/storage
```

All analysis runs uploaded to project `foo` prior to September 1, 2018 are deleted from the database.

You can also perform a cleanup on a specific project every time you upload a run to that project. To keep only the 20 most recent runs every time you upload a result to `myProject/example (Bug Finder)`, save these commands to your `.pscauto` file:

```
assign_to_project "myProject/example (Bug Finder)"
clean_project $currentProjectName MAXRUNS 20
```

To delete `myProject/example (Bug Finder)` entirely from the database, save this command to the `.pscauto` file.

```
delete_project "myProject/example (Bug Finder)"
```

Note You cannot recover the data that you delete by using the `.pscauto` script unless you have a backup copy of the data.

See Also

More About

- “Storage Configuration” on page 1-5
- “Manage Polyspace Bug Finder Access Software”

Update or Uninstall Polyspace Access

If you want to remove or update Polyspace Access, follow these steps to uninstall Polyspace Access. Before you apply any updates, uninstall your current version of Polyspace Access . Before you uninstall the software, back up the Polyspace Access database. See “Database Backup” on page 1-45.

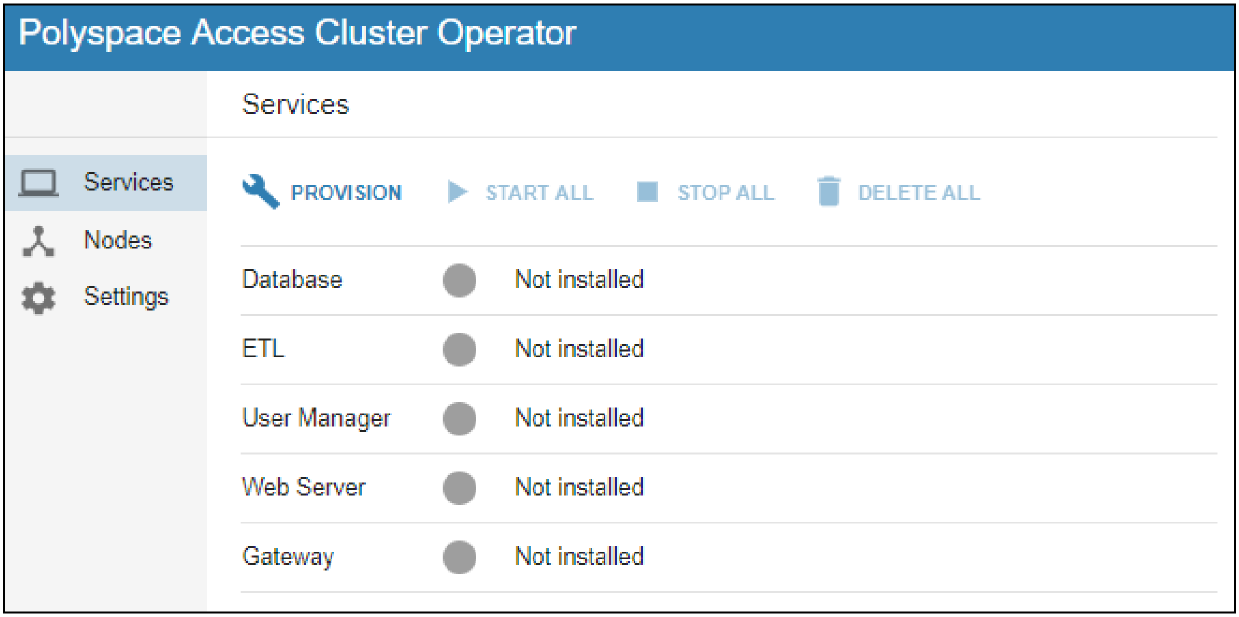
1 Inform Polyspace Access users of the upcoming update or uninstallation.

2 Verify that the COP agent is running with the command

```
docker stats --no-stream
```

If `polyspace-cop` is not listed under the `NAME` column in the command output, start the COP binary `cop-docker-agent`.

3 Open the COP web interface, go to the **Services** tab, and click **DELETE ALL**. After you delete a service, the service indicator turns gray and you see the text **Not installed** next to the indicator.



Database	●	Not installed
ETL	●	Not installed
User Manager	●	Not installed
Web Server	●	Not installed
Gateway	●	Not installed

Deleting the **Database** service and uninstalling Polyspace Access does not erase the results that you uploaded to the database from the data volume.

To delete a data volume and its content, click **Nodes** in the left pane, click the appropriate node ID, select the volume, and then click **DELETE**.

4 Stop the COP agent from the command line window by pressing **CTRL+C**.

5 Go to the folder where you unzipped the installation image for Polyspace Access and delete the `cop-docker-agent` binaries and TAR files. If you are updating Polyspace Access, do not delete these files until you complete the update.

To reuse your current Polyspace Access services configuration after you update, make a backup copy of the `settings.json` file.

To complete an update, download and unzip the new installation image, then go to the folder where you unzipped the new image and “Configure and Start the Cluster Operator” on page 1-8.

- If you reuse an existing configuration for the Polyspace Access services, copy your backup `settings.json` file into the same folder as the COP binary before you start the COP.
- To apply new configuration settings for the Polyspace Access services, see “Configure Polyspace Access Services” on page 1-13.

See Also

`cop-docker-agent`

More About

- “Configure and Start the Cluster Operator” on page 1-8
- “Configure Polyspace Access Services” on page 1-13

Get Started with Polyspace Bug Finder Access

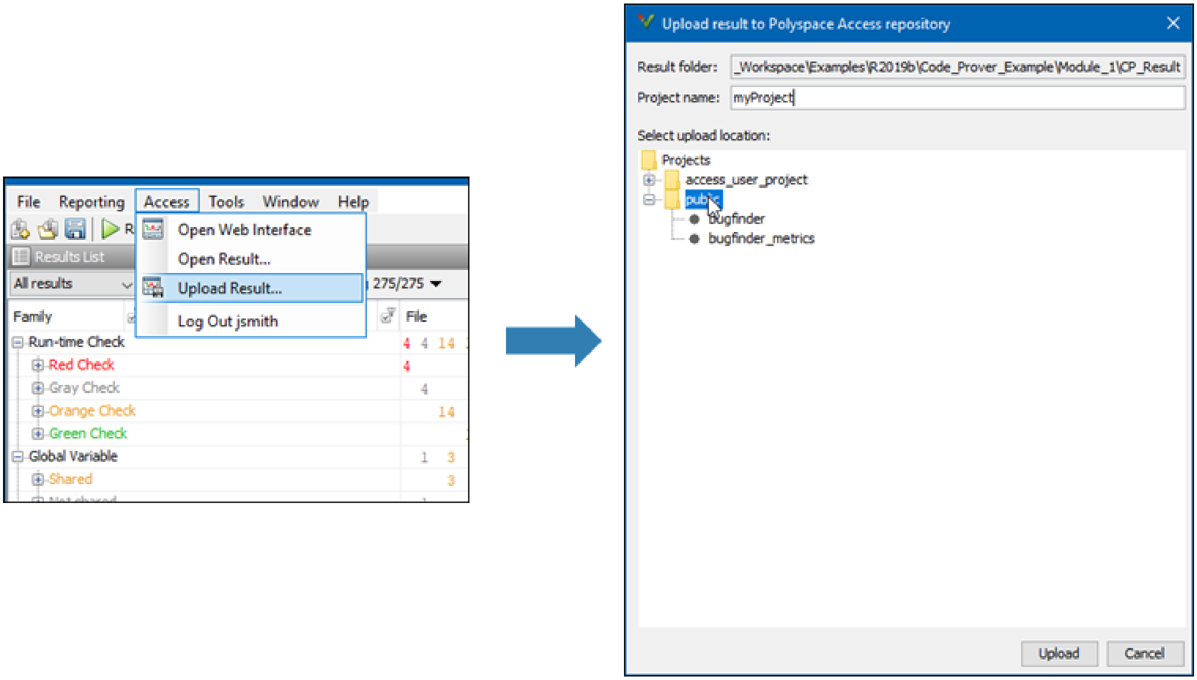
Upload Results to Polyspace Access

Polyspace Access offers a centralized database where you can store Polyspace analysis results for sharing and collaborative reviews. After you upload results, open the Polyspace Access web interface to view statistics about the quality of your code and to triage and review individual results.

Upload Results from Polyspace Desktop Client

Before you upload results, you must configure the Polyspace desktop client to communicate with Polyspace Access. See “Register Polyspace Desktop User Interface” on page 1-36.

To upload analysis results to the Polyspace Access database from the Polyspace desktop client, select a set of results in the **Project Browser** pane or open the results in the **Results List** pane. Go to **Access > Upload Results** and follow the prompts. If you get a login request, use your Polyspace Access login credentials.



Result folder: _Workspace\Examples\R2019b\Code_Prover_Example\Module_1\CP_Result

Project name: myProject

You can also upload results to Polyspace Access by selecting a result in the **Project Browser** pane and using the context menu.

After you upload results to Polyspace Access, if you open a local copy of the results in the desktop interface, you cannot make changes to the **Status**, **Severity**, or comment fields. To make changes to the **Status**, **Severity**, or comment fields, open the results from Polyspace Access by going to **Access** > **Open Results**.

Once you save the changes you make to these fields in the desktop interface, the changes are reflected in the Polyspace Access web interface.

Upload Results at Command Line

You can upload results from the from the command line only if they are generated with Polyspace Bug Finder Server or Polyspace Code Prover Server.

To upload analysis results to Polyspace Access from the DOS or UNIX command line, use the `polyspace-access` binary. In the command, specify the path of the folder under which the `.psbf`, `.pscp`, or `.rte` results file is stored. For instance, to upload Polyspace Bug Finder results stored in the file `BF_results\ps_results.psbf`, use this command:

```
polyspace-access -host hostName -port port -upload BF_results
```

The command prompts you for your Polyspace Access login credentials, then uploads the results to the public folder of the Polyspace Access database. *`hostName`* is the fully qualified host name of the machine where you run the **Gateway** service. *`port`* is the **Port number** value specified for the **Gateway** service in the settings of the cluster operator. Depending on your configuration, you might also have to specify the `-protocol` option in the command. See “Configure Polyspace Access Services” on page 1-13.

For additional information on `polyspace-access`, see the documentation for Polyspace Bug Finder or Polyspace Bug Finder Server.

See Also

More About

- “Register Polyspace Desktop User Interface” on page 1-36
- “Interpret Results”
- “Manage Results”

Open or Export Results from Polyspace Access

Polyspace Access offers a centralized database where you can store Polyspace analysis results for sharing with your team and collaborative reviews. After you upload analysis results to the database, you can view the results in your web browser or you can open the results from any Polyspace desktop interface that is configured for Polyspace Access. You can also export a list of results to a tab-separated value (TSV) file for further processing, such as applying custom filters and pass/fail criteria.

Open Polyspace Access Results in a Desktop Interface

Before you open Polyspace Access results in a desktop interface, you must configure the Polyspace desktop interface to communicate with Polyspace Access. See “Register Polyspace Desktop User Interface” on page 1-36.

To open results stored in the Polyspace Access database, go to **Access** > **Open Result** in the desktop interface, and follow the prompts. If you get a login request, use your Polyspace Access login credentials.

You can also open the desktop interface from the Polyspace Access web interface. On the toolstrip, click **Open in Desktop**. The desktop interface opens and shows the analysis results currently displayed in the Polyspace Access web interface.

Note In Linux, the desktop interface must already be open before you can view analysis results currently open in Polyspace Access.

Once you open results in the Polyspace desktop interface, changes you make to the **Status**, **Severity**, or comments fields are reflected in Polyspace Access after you save those changes.

After you upload analysis results to Polyspace Access, if you open a local copy of these results in the desktop interface, you cannot edit the **Status**, **Severity**, or comments fields.

Export Polyspace Access Results to a TSV File

You can export Polyspace Access results to a tab-separated values (TSV) file only from the command line by using the `polyspace-access` binary. When you export results, you generate a TSV file that lists results with most of the same results attributes as the

“Results List”. Each listed result also includes a URL through which you can open the result in Polyspace Access. To filter the list of results you export, see the `polyspace-access` export options.

For example, to export all coding rules with status `Unreviewed` from project **myProject** stored in the **public** folder on Polyspace Access, open a command prompt terminal and enter:

```
polyspace-access -host hostName -port port ^
-export public/myProject -coding-rules -review-status Unreviewed ^
-output coding_rules.tsv
```

The command prompts you for your Polyspace Access login credentials, and then outputs file `coding_rules.tsv`.

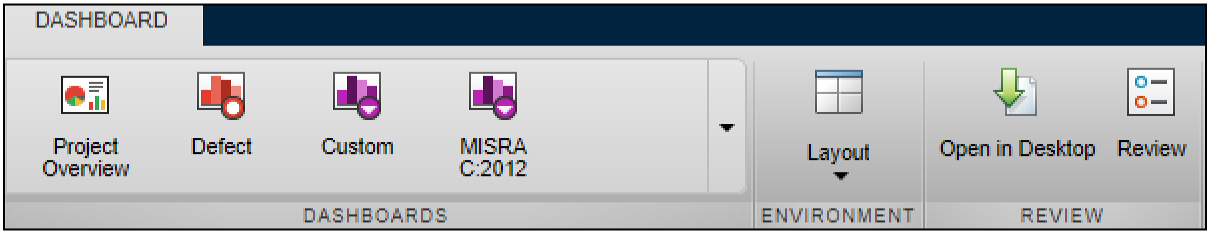
`hostName` is the fully qualified host name of the machine where you run the **Gateway** service. *`port`* is the **Port number** value specified for the **Gateway** service in the settings of the cluster operator. Depending on your configuration, you might also have to specify the `-protocol` option in the command. See “Configure Polyspace Access Services” on page 1-13.

For additional information on `polyspace-access`, see the documentation for Polyspace Bug Finder or Polyspace Bug Finder Server.

Dashboard

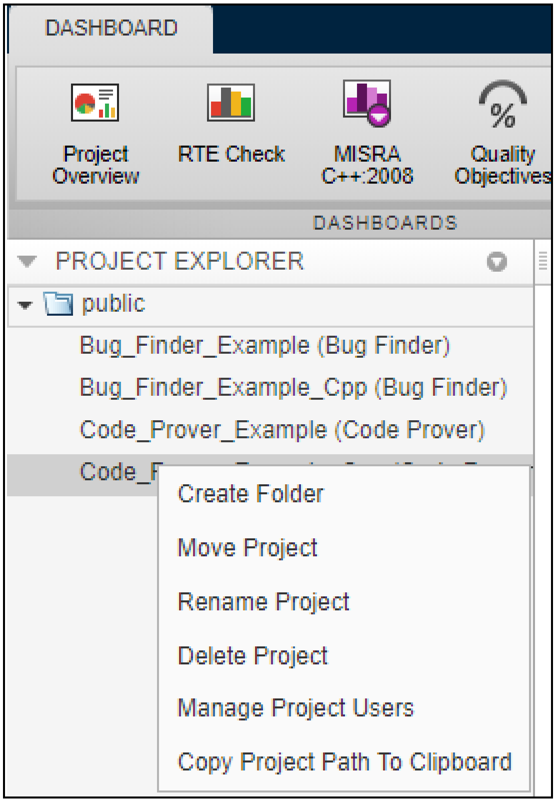
The **DASHBOARD** perspective provides an overview of the analysis results in graphical format, with clickable fields that enable you drill down into your findings by file, project, or category.

DASHBOARD toolstrip



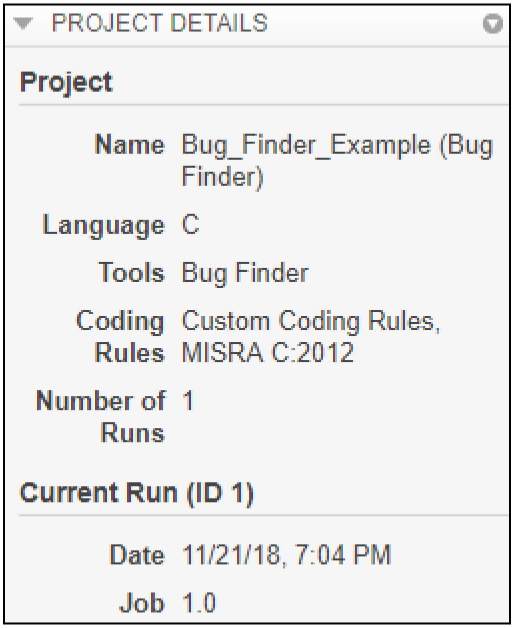
- Click a button in the **DASHBOARDS** section of the toolstrip to open the corresponding dashboard for the selected folder or project. Except for **Project Overview** and **Quality Objectives**, each dashboard shows information for a single family of findings.
- The **Open in Desktop** and **Review** buttons in the toolstrip are not available when you select a folder in the **PROJECT EXPLORER** pane.

PROJECT EXPLORER pane



- View all projects and folders for which you are an **Administrator**, **Owner** or **Contributor**. All users are contributors to the **Public** folder
- To manage folders, projects, or user permissions, use the context menu.
- The dashboards on the right display information for the selected folder or project.

PROJECT DETAILS pane



Project

Name: Bug_Finder_Example (Bug Finder)

Language: C

Tools: Bug Finder

Coding Rules: Custom Coding Rules, MISRA C:2012

Number of Runs: 1

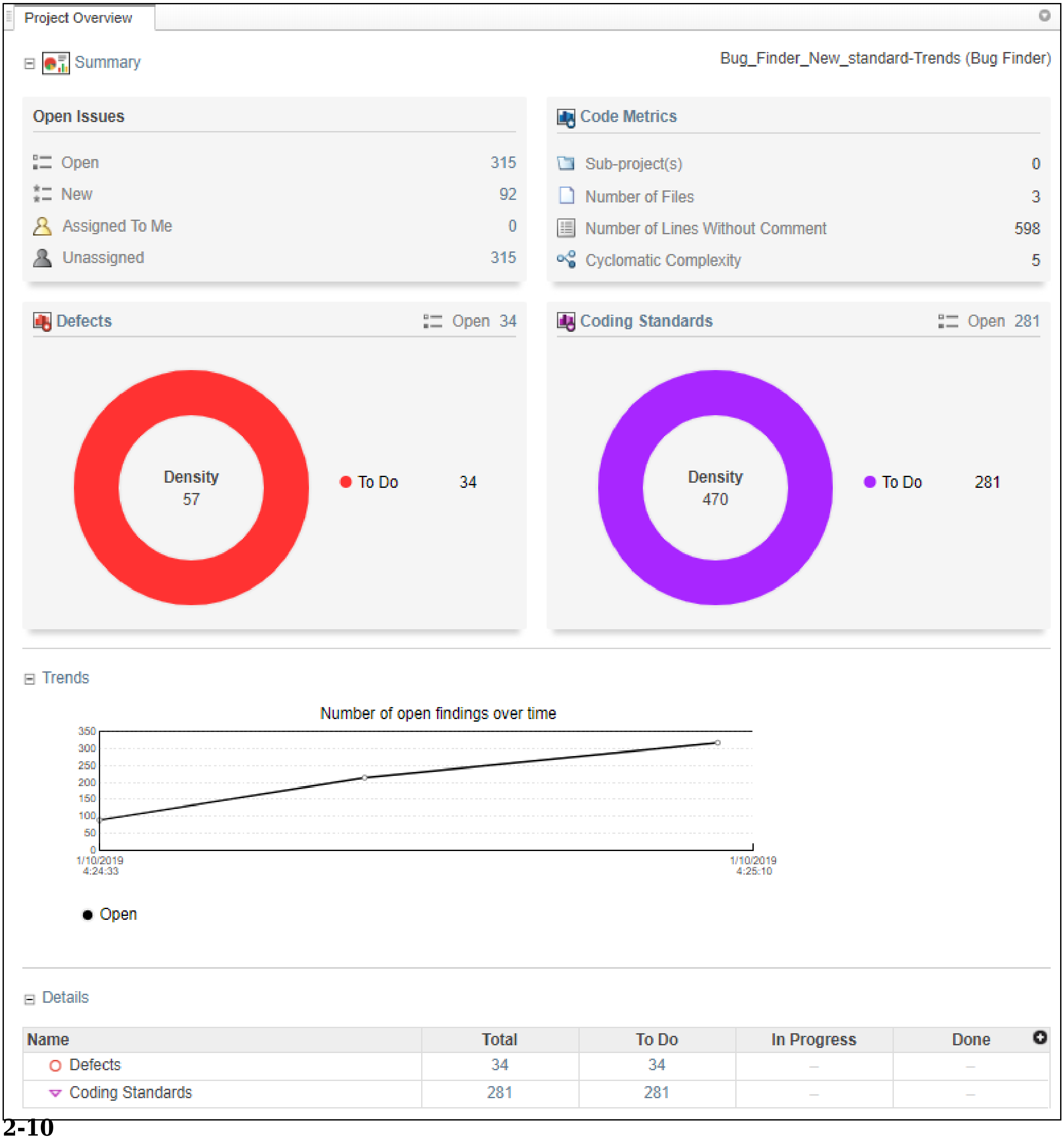
Current Run (ID 1)

Date: 11/21/18, 7:04 PM

Job: 1.0

- View additional details about the selected folder or project. You can view information about which language and coding standards were enabled in the analysis configuration.

Project Overview dashboard



Project Overview

Summary

Bug_Finder_New_standard-Trends (Bug Finder)

Open Issues	
Open	315
New	92
Assigned To Me	0
Unassigned	315

Code Metrics	
Sub-project(s)	0
Number of Files	3
Number of Lines Without Comment	598
Cyclomatic Complexity	5

Defects — Open 34

Density 57

● To Do 34

Coding Standards — Open 281

Density 470

● To Do 281

Trends

Number of open findings over time

Details

Name	Total	To Do	In Progress	Done
○ Defects	34	34	–	–
▽ Coding Standards	281	281	–	–

This dashboard gives you a snapshot of all the findings available for the selected folder or project. If you select a folder that includes multiple projects, the dashboard displays an aggregate of results for all the projects. The dashboard contains three collapsible sections:

- **Summary**

  Displays cards with information about open issues, code metrics, quality objectives (when available), and the different families of findings. Click the card title to open its corresponding dashboard. Click the number of findings in a card (when applicable) to open a list filtered to this set of findings.

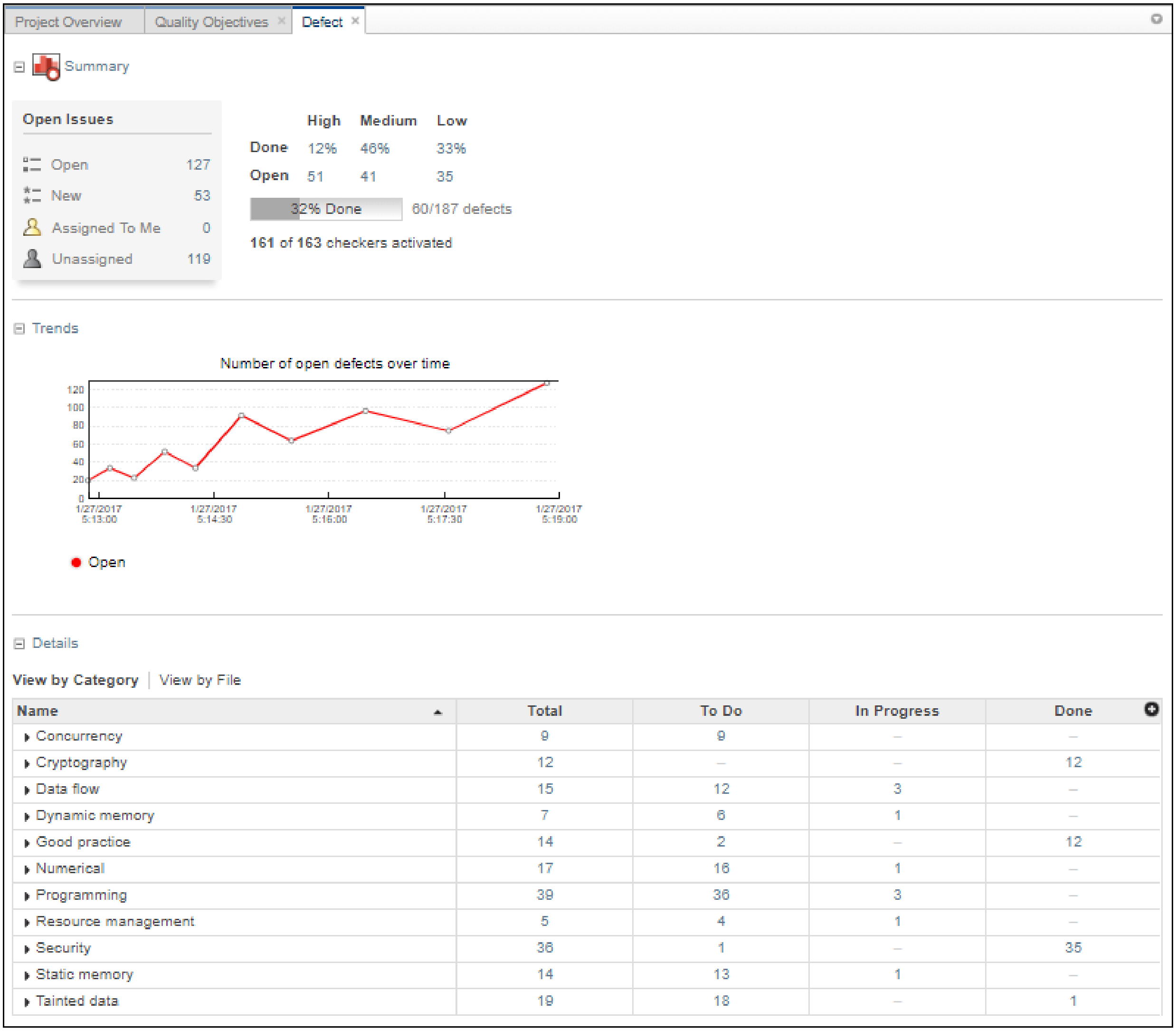
  The **Run-time Check** card shows a distribution of findings as red, orange, gray, and green. The card also shows the selectivity, the number of green checks as a percentage of all detected run-time checks.

  Defects and **Coding Rules** cards show a distribution of findings as to do, in progress, and done. The card also shows the density, the number of defects or coding standard violation per one thousand lines of code without comments. To view the density you must enable **Code Metrics** in your analysis configuration.
- **Trends**

  Displays a trend chard of the number of open findings over time as you upload additional runs for a project.
- **Details**

  Displays a table with a row containing the number of total, to do, in progress, and done for each type of analysis finding. Click the number of findings in a row (when applicable) to open a list filtered to this set of findings.

RTE Check, Defects and Coding Rules dashboards



Open Issues	
Open	127
New	53
Assigned To Me	0
Unassigned	119

	High	Medium	Low
Done	12%	46%	33%
Open	51	41	35

32% Done 60/187 defects

161 of 163 checkers activated

Number of open defects over time

View by Category | View by File

Name	Total	To Do	In Progress	Done
Concurrency	9	9	–	–
Cryptography	12	–	–	12
Data flow	15	12	3	–
Dynamic memory	7	6	1	–
Good practice	14	2	–	12
Numerical	17	16	1	–
Programming	39	36	3	–
Resource management	5	4	1	–
Security	36	1	–	35
Static memory	14	13	1	–
Tainted data	19	18	–	1

These dashboards give you a more in-depth overview for a family of findings. If you select a folder that includes multiple projects, the dashboards display an aggregate of results for all the projects. The dashboards contain three collapsible sections:

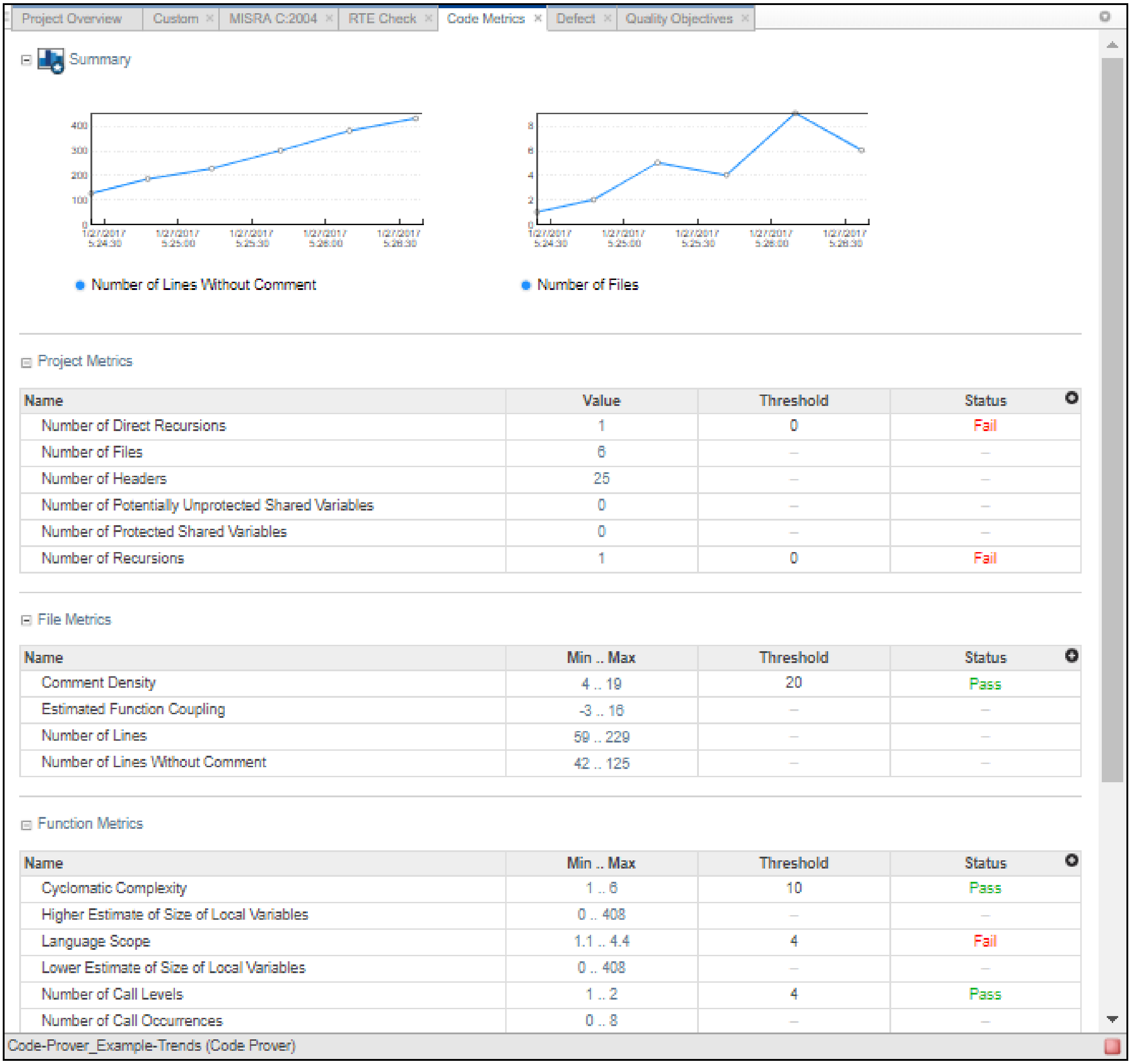
- **Summary**

  Displays a table with information about open issues and the progress in addressing these issues. Click the number of findings in the card (when applicable) to open a list filtered to this set of findings.
- **Trends**

  Displays a trend chard of the number of open findings over time as you upload additional runs for a project.
- **Details**

  Displays a table that allows you to drill down into the findings by category or by file. If you select a folder that contains multiple projects, you get a categorization by project instead of by file.Click the number of findings in a row (when applicable) to open a list filtered to this set of findings.

Code Metrics dashboard



Project Metrics

Name	Value	Threshold	Status
Number of Direct Recursions	1	0	Fail
Number of Files	6	–	–
Number of Headers	25	–	–
Number of Potentially Unprotected Shared Variables	0	–	–
Number of Protected Shared Variables	0	–	–
Number of Recursions	1	0	Fail

File Metrics

Name	Min .. Max	Threshold	Status
Comment Density	4 .. 19	20	Pass
Estimated Function Coupling	-3 .. 16	–	–
Number of Lines	59 .. 229	–	–
Number of Lines Without Comment	42 .. 125	–	–

Function Metrics

Name	Min .. Max	Threshold	Status
Cyclomatic Complexity	1 .. 6	10	Pass
Higher Estimate of Size of Local Variables	0 .. 408	–	–
Language Scope	1.1 .. 4.4	4	Fail
Lower Estimate of Size of Local Variables	0 .. 408	–	–
Number of Call Levels	1 .. 2	4	Pass
Number of Call Occurrences	0 .. 8	–	–

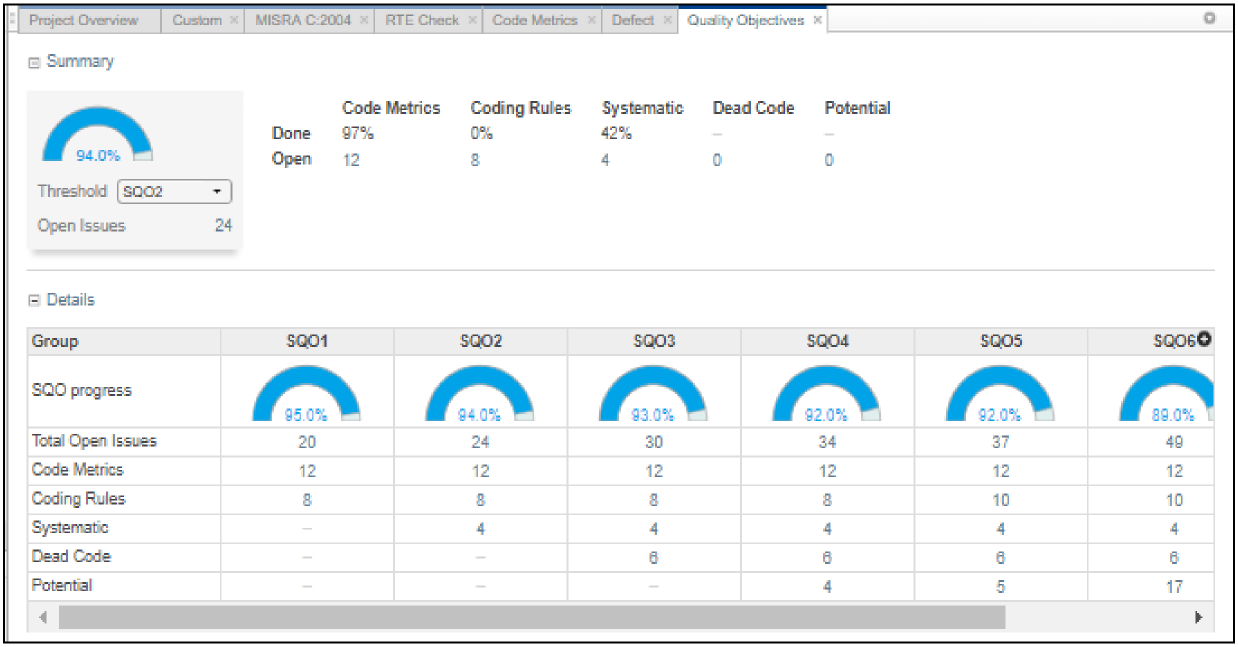
This dashboard contains four collapsible sections.

- **Summary**

  Displays trends charts of the number of lines without comments and the number of files for the selected folder or project
- **Project Metrics**, **File Metrics**, and **Function Metrics**

  These sections display tables with rows containing the value or range of a metric, along with its threshold and pass/fail status when applicable. Click the number of findings in a row (when applicable) to open a list filtered to this set of findings.

Quality Objectives dashboard



Threshold: SQO2

Open Issues: 24

	Code Metrics	Coding Rules	Systematic	Dead Code	Potential
Done	97%	0%	42%	–	–
Open	12	8	4	0	0

Group	SQO1	SQO2	SQO3	SQO4	SQO5	SQO6
SQO progress	95.0%	94.0%	93.0%	92.0%	92.0%	89.0%
Total Open Issues	20	24	30	34	37	49
Code Metrics	12	12	12	12	12	12
Coding Rules	8	8	8	8	10	10
Systematic	–	4	4	4	4	4
Dead Code	–	–	6	6	6	6
Potential	–	–	–	4	5	17

This dashboard displays a summary of the quality of your code against the threshold selected from the dropdown menu. The dashboard also shows a table with details of code quality for all quality objective thresholds.

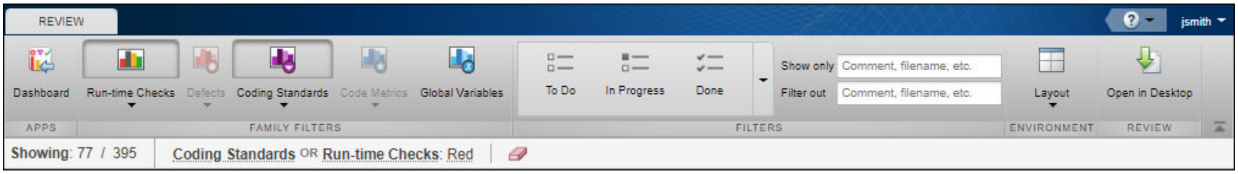
Review

The **REVIEW** perspective provides you with an environment that enables you to:

- Filter and investigate individual findings in your code.
- Add a review status, severity or comment to findings.
- Assign an owner to a finding and create a ticket in your bug tracking tool to track the issue.

Note The **REVIEW** perspective is only available for analysis results generated with a Polyspace product version R2019a or later.

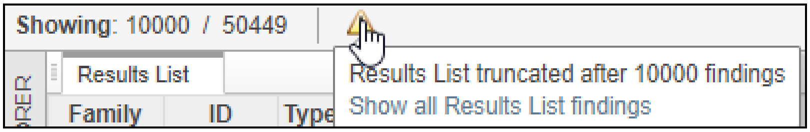
REVIEW toolstrip



Showing: 77 / 395 | Coding Standards OR Run-time Checks: Red

- Click a button in the **FAMILY FILTERS** section of the toolstrip to see the corresponding family of findings, or a subset of those findings. The filters bar underneath shows how many findings are displayed out of the total findings, along with which filters are currently applied.

  If the **Results List** exceeds 10000 findings, Polyspace Access truncates the list and displays this icon ⚠ in the filters bar. To show all findings, see the contextual help of the icon.

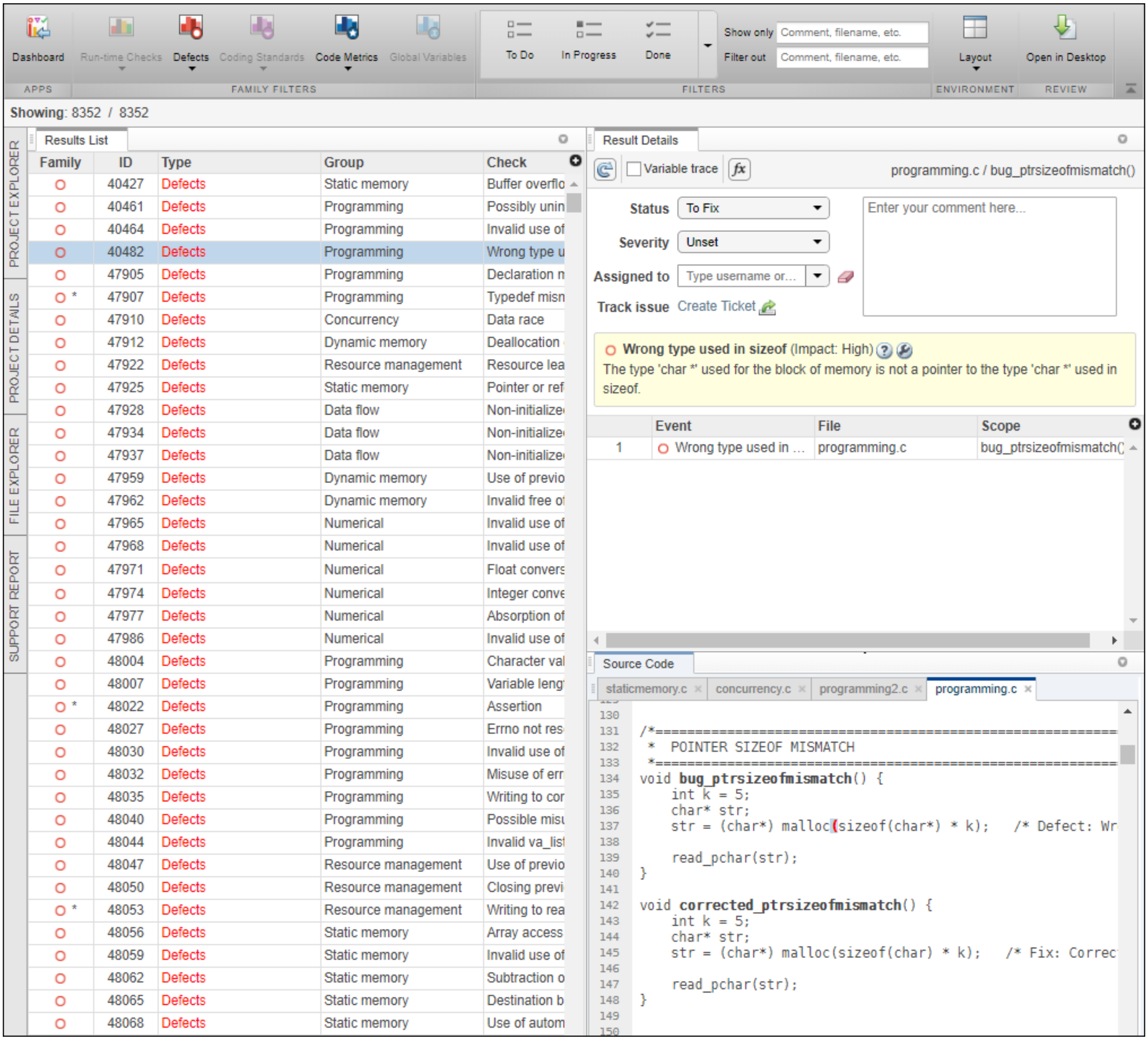
  

  Showing: 10000 / 50449

  Results List truncated after 10000 findings
  Show all Results List findings

  The 10000 findings limit is preset and cannot be changed.
- The buttons in the **FILTERS** section of the toolstrip are global. They apply to all families of findings.

Default view: Results List, Results Details, and Source Code



Show only: Comment, filename, etc.

Filter out: Comment, filename, etc.

Showing: 8352 / 8352

Family	ID	Type	Group	Check
○	40427	Defects	Static memory	Buffer overflo
○	40461	Defects	Programming	Possibly unin
○	40464	Defects	Programming	Invalid use of
○	40482	Defects	Programming	Wrong type u
○	47905	Defects	Programming	Declaration n
○ *	47907	Defects	Programming	Typedef misn
○	47910	Defects	Concurrency	Data race
○	47912	Defects	Dynamic memory	Deallocation
○	47922	Defects	Resource management	Resource lea
○	47925	Defects	Static memory	Pointer or ref
○	47928	Defects	Data flow	Non-initialize
○	47934	Defects	Data flow	Non-initialize
○	47937	Defects	Data flow	Non-initialize
○	47959	Defects	Dynamic memory	Use of previo
○	47962	Defects	Dynamic memory	Invalid free of
○	47965	Defects	Numerical	Invalid use of
○	47968	Defects	Numerical	Invalid use of
○	47971	Defects	Numerical	Float convers
○	47974	Defects	Numerical	Integer conve
○	47977	Defects	Numerical	Absorption of
○	47986	Defects	Numerical	Invalid use of
○	48004	Defects	Programming	Character val
○	48007	Defects	Programming	Variable leng
○ *	48022	Defects	Programming	Assertion
○	48027	Defects	Programming	Errno not res
○	48030	Defects	Programming	Invalid use of
○	48032	Defects	Programming	Misuse of err
○	48035	Defects	Programming	Writing to cor
○	48040	Defects	Programming	Possible misu
○	48044	Defects	Programming	Invalid va_list
○	48047	Defects	Resource management	Use of previo
○	48050	Defects	Resource management	Closing previ
○ *	48053	Defects	Resource management	Writing to rea
○	48056	Defects	Static memory	Array access
○	48059	Defects	Static memory	Invalid use of
○	48062	Defects	Static memory	Subtraction o
○	48065	Defects	Static memory	Destination b
○	48068	Defects	Static memory	Use of autom

□ Variable trace

programming.c / bug_ptrsizeofmismatch()

Status: To Fix

Severity: Unset

Assigned to: Type username or...

Track issue: Create Ticket

Enter your comment here...

○ **Wrong type used in sizeof** (Impact: High)
The type 'char *' used for the block of memory is not a pointer to the type 'char *' used in sizeof.

	Event	File	Scope
1	○ Wrong type used in ...	programming.c	bug_ptrsizeofmismatch()

staticmemory.c × | concurrency.c × | programming2.c × | programming.c ×

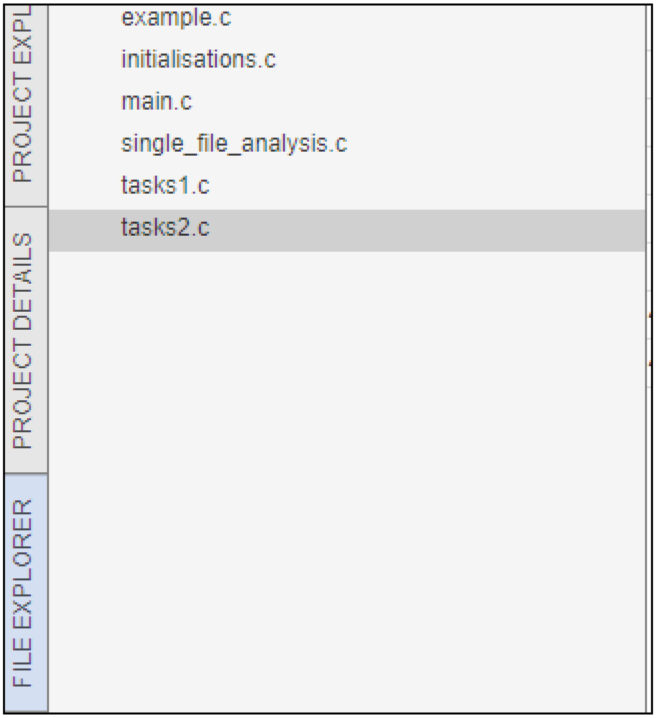
```

/*=============================================================
 *  POINTER SIZEOF MISMATCH
 *=============================================================
void bug_ptrsizeofmismatch() {
    int k = 5;
    char* str;
    str = (char*) malloc(sizeof(char*) * k);   /* Defect: Wr

    read_pchar(str);
}

void corrected_ptrsizeofmismatch() {
    int k = 5;
    char* str;
    str = (char*) malloc(sizeof(char) * k);   /* Fix: Correc

    read_pchar(str);
}

```

In the default layout, you see the **Results List**, **Results Details**, and **Source Code** panes.

- Click a finding in the **Results List** to see its location in the **Source Code** pane. Additional information about the finding is available in the **Results Details** pane. To open contextual help for the finding, in the **Results Details** click ?. When available, click the 🔧 icon to see fix suggestions for the defect.
- Click a column heading in the **Results List** to sort findings according to that heading.
- Right-click a cell in the **Results List** to show only/exclude findings based on the content of that cell.

To open additional panes, use **Layout** > **Show/Hide View**.

FILE EXPLORER pane



example.c
initialisations.c
main.c
single_file_analysis.c
tasks1.c
tasks2.c

Use the file explorer to show findings by file in the **Results List** pane.

Manage Permissions and View Project Trends

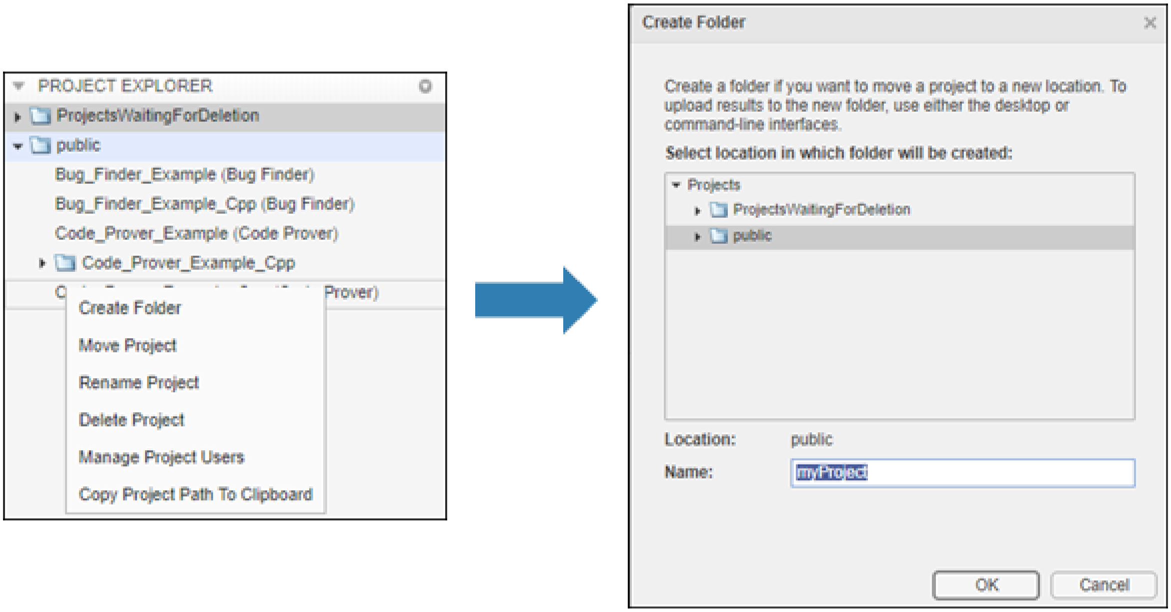
Before you start reviewing the overall quality of a project and investigating findings in your code, create project folders and set permissions to allow or restrict team members access to your projects.

Create a Project Folder

To facilitate the review process, create folders in Polyspace Access to group related results.

Create Folder from the Polyspace Access Interface

From the **PROJECT EXPLORER** in the **DASHBOARD** perspective, select any existing folder or project and click **Create Folder** in the context menu. In the **Create Folder** window, click an existing folder to create a subfolder. To create a folder at the top of the **PROJECT EXPLORER** hierarchy, click **Projects**.


PROJECT EXPLORER
ProjectsWaitingForDeletion
public
Bug_Finder_Example (Bug Finder)
Bug_Finder_Example_Cpp (Bug Finder)
Code_Prover_Example (Code Prover)
Code_Prover_Example_Cpp
Prover)
Create Folder
Move Project
Rename Project
Delete Project
Manage Project Users
Copy Project Path To Clipboard
Create Folder
Projects
ProjectsWaitingForDeletion
public
OK
Cancel

Create a folder if you want to move a project to a new location. To upload results to the new folder, use either the desktop or command-line interfaces.

Select location in which folder will be created:

Location: public

Name: myProject

Create Project Folder at Command Line

To create a folder in Polyspace Access from the DOS or UNIX command lines, use the `polyspace-access` binary. This binary is available under the *`polyspaceroot`*`/polyspace/bin` folder with a Polyspace Bug Finder or a Polyspace Bug Finder Server installation. *`polyspaceroot`* is the Polyspace product installation folder, for example `C:\Program Files\Polyspace Server\2019a`.

For instance, to create `myProject` under the node `myRelease`, use this command:

```
polyspace-access -host hostName -create-project myRelease/myProject
```

`hostName` is the host name of the machine where the Polyspace Access **Gateway** service is running. You specify this host name in the URL of the Polyspace Access web interface. Depending on your configuration, you might also need to specify the `-port` and `-protocol` options in the migration command.

For more information on `polyspace-access`, see the Polyspace Bug Finder Server or Polyspace Code Prover Server documentation.

Manage Project Permissions

To set permissions for folders or projects in Polyspace Access, assign user roles . These permissions are the permissions that correspond to each role.

Role	Permission
Owner	Move, rename, or delete specified folders or projects and review their content. Assign roles **Contributor** or **Forbidden**. You cannot move a folder or project to a new location if a folder or project with the same name already exists at that location.
Contributor	Review content of specified folder or project. See the roles of other users in the project.
Forbidden	No access specified folder or project. Set this role to restrict the access of to a project inside a folder that is accessible to the users.

Administrator or **Owner** roles can allow or restrict the access of other team members to a project or folder. You are the owner of folders that you create and of project results that you upload.

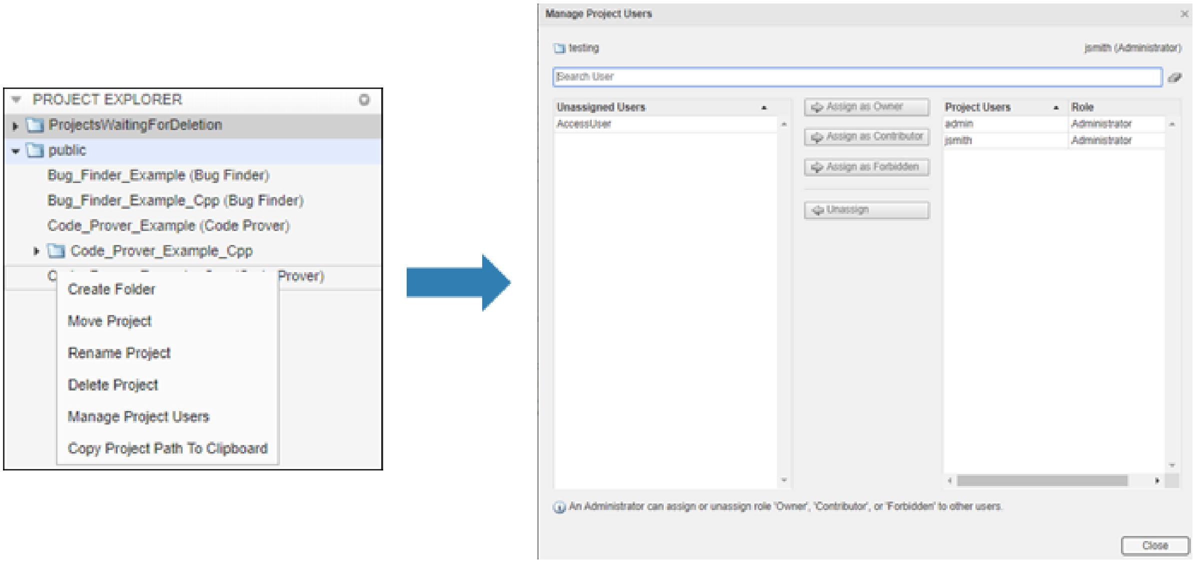
Only administrators can assign owners to a project. To set a user as **Administrator**, see "Configure the Web Server and Gateway" on page 1-31.

The permissions that you set on a folder apply to all projects in that folder. For instance, if user `jdoe` has **Contributor** privileges for folder `myRelease`, `jdoe` is a contributor for all projects under `myRelease`. You can set additional permissions for each project under `myRelease`.

Project results uploaded to the **public** folder are accessible to all team members.

Manage Permissions in Polyspace Access Web Interface

From the **PROJECT EXPLORER** in the **DASHBOARD** perspective, select any existing folder or project and click **Manage Project Users** in the context menu. You can search for a user, assign a role to a user with no role, or change the role of a user.



Unassigned Users
AccessUser

Project Users	Role
admin	Administrator
jsmith	Administrator

An Administrator can assign or unassign role 'Owner', 'Contributor', or 'Forbidden' to other users.

Manage Permissions at Command Line

To manage access to uploaded results from the DOS or UNIX command lines, use the `polyspace-access` binary. This binary is available under the *`polyspaceroot`*`/polyspace/bin` folder with a Polyspace Bug Finder or a Polyspace Bug Finder Server installation. *`polyspaceroot`* is the Polyspace product installation folder, for example `C:\Program Files\Polyspace Server\2019a`.

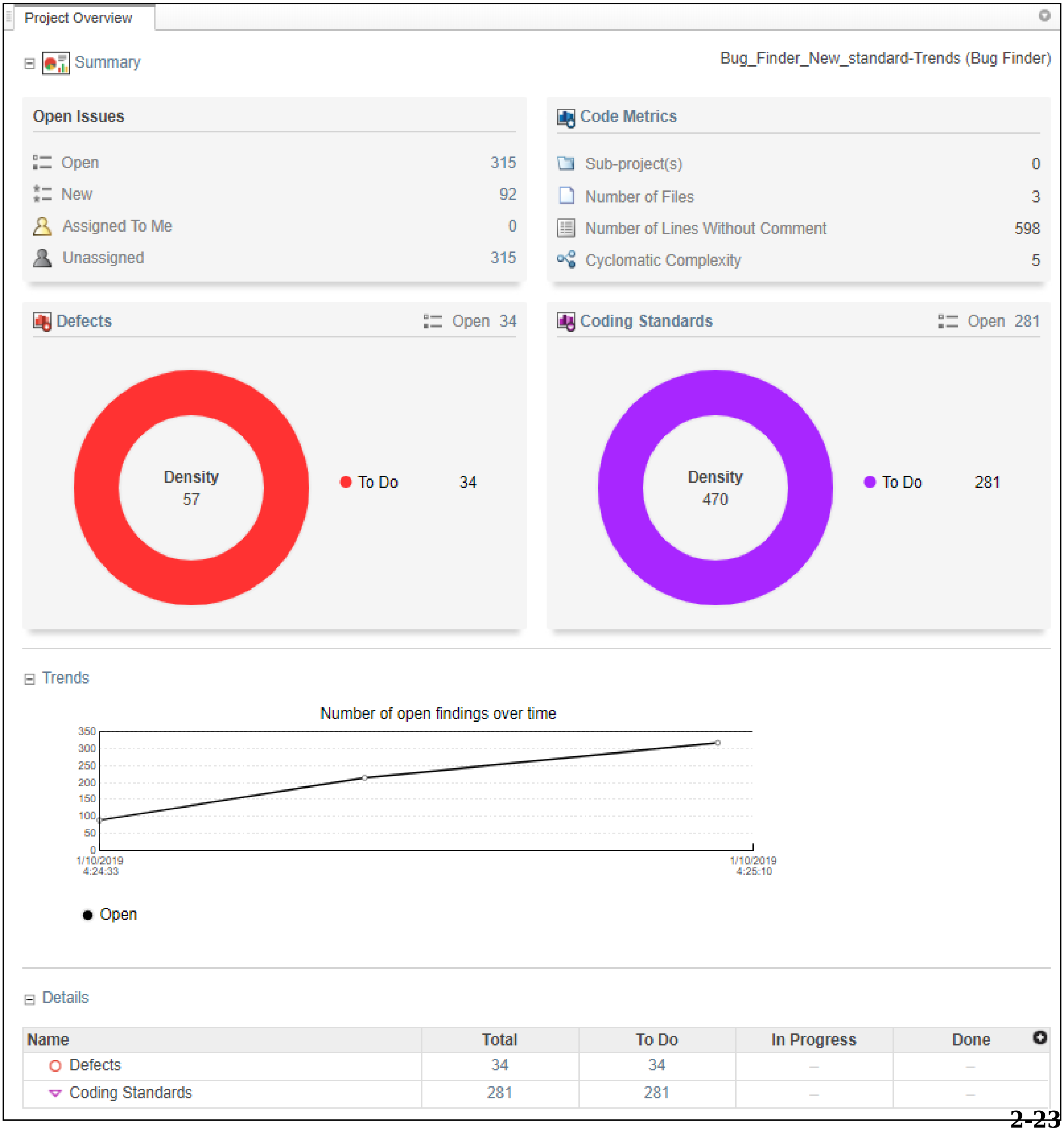
For instance to assign `jsmith` as **Contributor** for project `myProject`, use this command:

```
polyspace-access -host hostName ^
-set-role contributor -user jsmith ^
-project-path myProject
```

`hostName` is the host name of the machine where the Polyspace Access **Gateway** service is running. You specify this host name in the URL of the Polyspace Access web interface. Depending on your configuration, you might also need to specify the `-port` and `-protocol` options in the migration command.

For more information on `polyspace-access`, see the Polyspace Bug Finder Server or Polyspace Code Prover Server documentation.

View Project Trends



Project Overview

Summary

Bug_Finder_New_standard-Trends (Bug Finder)

Open Issues

Open	315
New	92
Assigned To Me	0
Unassigned	315

Code Metrics

Sub-project(s)	0
Number of Files	3
Number of Lines Without Comment	598
Cyclomatic Complexity	5

Defects Open 34

Density 57

● To Do 34

Coding Standards Open 281

Density 470

● To Do 281

Trends

Number of open findings over time

Details

Name	Total	To Do	In Progress	Done
○ Defects	34	34	–	–
▽ Coding Standards	281	281	–	–

In the **DASHBOARD** perspective, select the project that you want to investigate from the **PROJECT BROWSER**.

If you select a folder, the project overview displays an aggregate of all the project results in that folder.

In the **Project Overview** dashboard, you see a summary of **Open Issues**, including the number of **New** results since the previous analysis run and the number of results that are **Unassigned**.

Other cards provide statistics for each family of findings. The **Run-time Checks** card shows the **Selectivity**, that is, the percentage of all findings that are green. When you enable the calculation of code metrics in your analysis, the **Defects** and **Coding Standards** cards show the **Density**, the number of findings per thousand lines of code without comments.

In the **Details** section, you see the review progress for each family of results. The results are classified as:

- **To Do**, with a status of `Unreviewed`.
- **In Progress**, with a status of `To fix`, `To investigate`, or `Other`.
- **Done**, with a status of `Justified`, `No action planned`, or `Not a defect`.

If the number of open issues increases, open additional dashboards by using the buttons in the **DASHBOARDS** section of the toolstrip. Each button opens a dashboard for a family of findings, for instance **Defects**. To determine the root cause of the increase, Use the information on these dashboards. Once you determine the set of findings that you want your team to focus on, open the **REVIEW** perspective to start managing the results. See “Manage Results” on page 2-25.

See Also

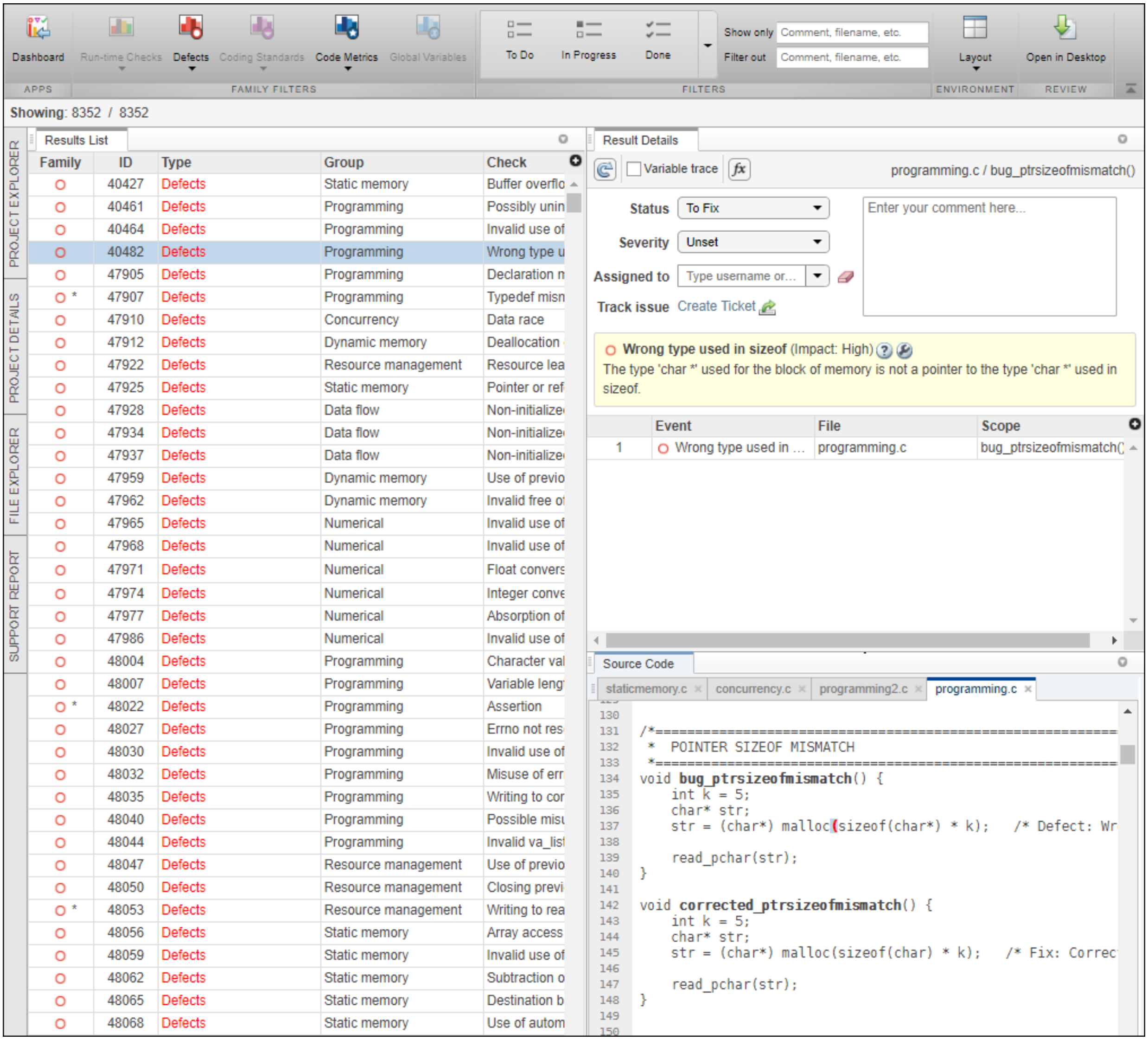
More About

- “Upload Results to Polyspace Access” on page 2-2

Manage Results

After you identify the results that you want to review, use the **REVIEW** perspective to manage these results. See “Manage Permissions and View Project Trends” on page 2-19.

If you open the **REVIEW** perspective from the **DASHBOARD** perspective, you see the **Results List** filtered down to the set of results you selected in a dashboard. If you open the **REVIEW** perspective by clicking a finding URL, you see only that finding in the **Results List**.



Showing: 8352 / 8352

Family	ID	Type	Group	Check
○	40427	Defects	Static memory	Buffer overflo
○	40461	Defects	Programming	Possibly unin
○	40464	Defects	Programming	Invalid use of
○	40482	Defects	Programming	Wrong type u
○	47905	Defects	Programming	Declaration n
○ *	47907	Defects	Programming	Typedef misn
○	47910	Defects	Concurrency	Data race
○	47912	Defects	Dynamic memory	Deallocation
○	47922	Defects	Resource management	Resource lea
○	47925	Defects	Static memory	Pointer or ref
○	47928	Defects	Data flow	Non-initialize
○	47934	Defects	Data flow	Non-initialize
○	47937	Defects	Data flow	Non-initialize
○	47959	Defects	Dynamic memory	Use of previo
○	47962	Defects	Dynamic memory	Invalid free o
○	47965	Defects	Numerical	Invalid use of
○	47968	Defects	Numerical	Invalid use of
○	47971	Defects	Numerical	Float convers
○	47974	Defects	Numerical	Integer conve
○	47977	Defects	Numerical	Absorption of
○	47986	Defects	Numerical	Invalid use of
○	48004	Defects	Programming	Character val
○	48007	Defects	Programming	Variable leng
○ *	48022	Defects	Programming	Assertion
○	48027	Defects	Programming	Errno not res
○	48030	Defects	Programming	Invalid use of
○	48032	Defects	Programming	Misuse of err
○	48035	Defects	Programming	Writing to cor
○	48040	Defects	Programming	Possible misu
○	48044	Defects	Programming	Invalid va_list
○	48047	Defects	Resource management	Use of previo
○	48050	Defects	Resource management	Closing previ
○ *	48053	Defects	Resource management	Writing to rea
○	48056	Defects	Static memory	Array access
○	48059	Defects	Static memory	Invalid use of
○	48062	Defects	Static memory	Subtraction o
○	48065	Defects	Static memory	Destination b
○	48068	Defects	Static memory	Use of autom

programming.c / bug_ptrsizeofmismatch()

Status: To Fix

Severity: Unset

Assigned to: Type username or...

Track issue: Create Ticket

Enter your comment here...

○ **Wrong type used in sizeof** (Impact: High)

The type 'char *' used for the block of memory is not a pointer to the type 'char *' used in sizeof.

	Event	File	Scope
1	○ Wrong type used in ...	programming.c	bug_ptrsizeofmismatch()

```
/*===========================================================
 *  POINTER SIZEOF MISMATCH
 *===========================================================
void bug_ptrsizeofmismatch() {
    int k = 5;
    char* str;
    str = (char*) malloc(sizeof(char*) * k);    /* Defect: Wr

    read_pchar(str);
}

void corrected_ptrsizeofmismatch() {
    int k = 5;
    char* str;
    str = (char*) malloc(sizeof(char) * k);   /* Fix: Correc

    read_pchar(str);
}
```

Apply additional filters to the **Results List** by using the toolstrip, or select a finding and use the context menu. To decide how to address each finding that you review, use the **Results Details** and **Source Code** panes. To open additional panes such as the **Call Hierarchy**, see **Layout > Show/Hide View** in the toolstrip. Once you decide how to address the finding, set or update the **Status**, **Severity**, **Assigned to**, or comment fields in the **Results Details** pane.

To create a JIRA issue and keep track of the workflow that addresses a finding from an existing JIRA project, click **Create Ticket** in the **Results Details** pane. Creating a JIRA issue is available only if Polyspace Access is configured to create JIRA issues. The ticket entry is populated with details of the finding and a link to Polyspace Access. See “Track Issue in Bug Tracking Tool”.

See Also

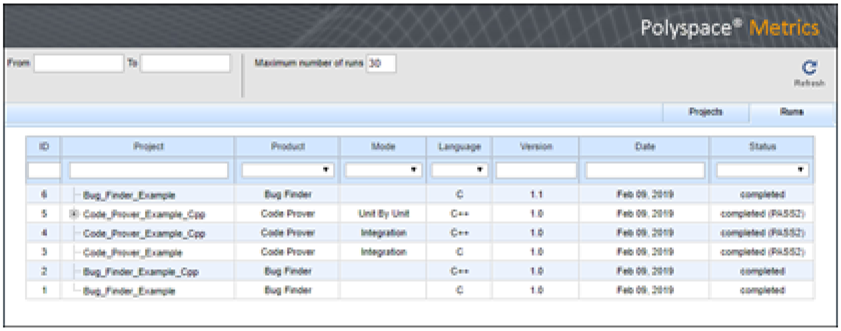
More About

- “Interpret Results”
- “Manage Results”

Migrate Results from Polyspace Metrics to Polyspace Access

If you use Polyspace Metrics to store results and monitor the quality of your source code, you can transfer those results to Polyspace Access.

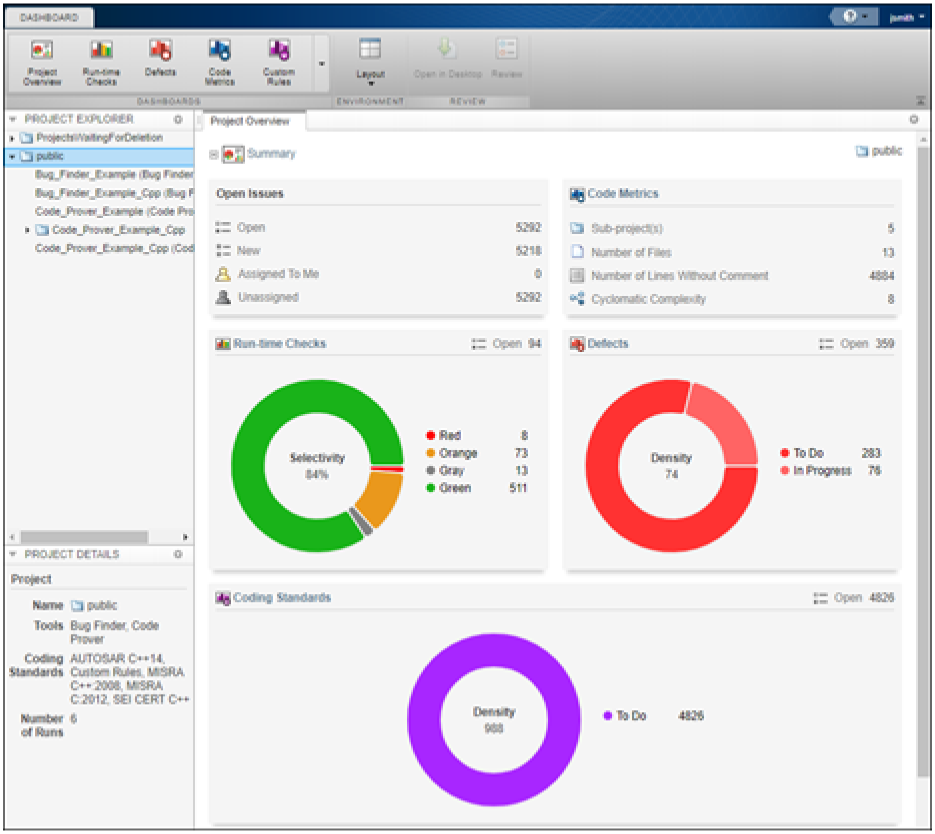
The Polyspace Access **DASHBOARD** perspective offers a web interface with navigation between projects and categories of results. From the **Project Overview** dashboard, view aggregated statistics for all your projects or drill down to view results details by category or file. For each family of findings, open an additional dashboard to see detailed. After you narrow down the set of findings that you want to address, open them in the **REVIEW** perspective to start reviewing individual results. You can also review results from Polyspace Access by opening them in the Polyspace desktop interface. You do not need to download a local copy of Polyspace Access results to view those results in the desktop interface. The edits that you make to the results are saved directly in Polyspace Access and enable you to perform collaborative reviews.



From To Maximum number of runs 30

ID	Project	Product	Mode	Language	Version	Date	Status
6	Bug_Finder_Example	Bug Finder		C	1.1	Feb 09, 2019	completed
5	Code_Prover_Example_Cpp	Code Prover	Unit By Unit	C++	1.0	Feb 09, 2019	completed (PASS2)
4	Code_Prover_Example_Cpp	Code Prover	Integration	C++	1.0	Feb 09, 2019	completed (PASS2)
3	Code_Prover_Example	Code Prover	Integration	C	1.0	Feb 09, 2019	completed (PASS2)
2	Bug_Finder_Example_Cpp	Bug Finder		C++	1.0	Feb 09, 2019	completed
1	Bug_Finder_Example	Bug Finder		C	1.0	Feb 09, 2019	completed

```
polyspace-access -generate-migration-commands
polyspace-access -migrate
```



ProjectsWaitingForDeletion
public
Bug_Finder_Example (Bug Finder
Bug_Finder_Example_Cpp (Bug F
Code_Prover_Example (Code Pro
Code_Prover_Example_Cpp
Code_Prover_Example_Cpp (Cod

Open Issues

Open	5292
New	5218
Assigned To Me	0
Unassigned	5292

Code Metrics

Sub-project(s)	5
Number of Files	13
Number of Lines Without Comment	4884
Cyclomatic Complexity	8

Run-time Checks Open 94

● Red	8
● Orange	73
● Gray	13
● Green	511

Defects Open 359

● To Do	283
● In Progress	76

Coding Standards Open 4826

● To Do	4826

PROJECT DETAILS

Project

Name public

Tools Bug Finder, Code Prover

Coding Standards AUTOSAR C++14, Custom Rules, MISRA C++:2008, MISRA C:2012, SEI CERT C++

Number of Runs 6

Requirements for Migration

The transfer of results from the Metrics repository to the Polyspace Access database requires the `polyspace-access` binary. This binary is available under the *`polyspaceroot`*`/polyspace/bin` folder with a Polyspace Bug Finder or a Polyspace Bug Finder Server installation. *`polyspaceroot`* is the Polyspace product installation folder, for instance `C:\Program Files\Polyspace Server\2019a`.

For more information on `polyspace-access`, see the Polyspace Bug Finder Server or Polyspace Code Prover Server documentation.

Migration of Results

To migrate results from Polyspace Metrics to Polyspace Access, follow these steps. You must be logged in to your Metrics server to complete this operation.

1. Identify the Metrics results repository location. The Polyspace Metrics results are stored in the `results-repository` folder at that location.

   To view the path to this location, from the desktop interface, go to **Tools > Metrics Server Settings**. Or, at the command line, run the command `psqueue-check-config`.

   By default, results are stored under `C:\Users\`*`username`*`\AppData\Roaming\Polyspace_RLDatas\results-repository` on Windows and `/home/`*`username`*`/.polyspace/results-repository` on Linux. *`username`* is your computer login user name.
2. Generate migration scripts.

   Once you identify the folder of the repository from which you want to transfer results, define a migration strategy. You can choose to transfer all your projects or you can narrow down the scope of the transfer to a specific set of projects. Specify a set of projects with the options listed in this table.

Option	Description
`-max-project-runs` *`int`*	Number of most recent analysis runs you want to migrate for each project. For instance, to migrate only the last two analysis runs of a project, specify 2.

Option	Description
`-project-date-after` *`YYYY[-MM[-DD]]`*	Only migrate results that were uploaded to Polyspace Metrics on or after the specified date.
`-product` *`productName`*	Product used to analyze and produce project findings, specified as `bug-finder`, `code-prover`, or `polyspace-ada`.
`-analysis-mode` *`mode`*	Analysis mode used to generate project findings, specified as `integration` or `unit-by-unit`.

For example, to transfer only Polyspace Bug Finder analysis results that you uploaded to Polyspace Metrics on or after June 2017, use this command:

```
polyspace-access -generate-migration-commands ^
C:\Users\username\AppData\Roaming\Polyspace_RLDatas\results-repository ^
-output-folder-path C:\Polyspace_Workspace\Migrate^
-project-date-after 2017-06^
-product bug-finder
```

The command outputs a migration script file for each project stored in `C:\Users\`*`username`*`\AppData\Roaming\Polyspace_RLDatas\results-repository` that matches the specified product and date. The migration scripts are stored under `C:\Polyspace_Workspace\Migrate`.

Before you continue, you can optionally open the migration scripts in a text editor and modify the `-project` or `-parent-project` parameters. The parameters correspond to the name of the project and the folder under which it is stored in Polyspace Access, respectively.

3 Migrate the projects.

After you generate the migration scripts, to transfer all the selected projects use those scripts with this migration command :

```
polyspace-access -host hostName ^
-migrate -option-file-path ^
C:\Polyspace_Workspace\Migrate
```

The command looks for migration scripts under `C:\Polyspace_Workspace\Migrate` and uploads the results to the Polyspace Access instance that you specify with *`hostName`*. Enter your Polyspace Access user name and password at the prompt.

`hostName` is the host name of the machine where the Polyspace Access **Gateway** service is running. You specify this host name in the URL of the Polyspace Access web interface. Depending on your configuration, you might also need to specify the `-port` and `-protocol` options in the migration command.

During the execution of a migration script, the command generates a temporary `STARTED` file. After each successful execution of a migration script, the command deletes the `STARTED` file and generates a corresponding `DONE` file in the same folder as the script. For example, the command generates `foo.started` during the execution of `foo.cmd`, and then `foo.done` once `foo.cmd` is done. Do not delete these `DONE` files until you have completed your migration from Metrics to Access.

Depending on the amount of data that you are transferring and on your network configuration, the migration might take a long time. You can interrupt the transfer, and then continue from where you left off at a later time. To stop the transfer, press **CTRL+C**. To restart the transfer:

a. Go to the folder where you store the migration scripts and open the `STARTED` file in a text editor. The file might be in a subfolder of the migration scripts folder.
b. Follow the instructions in the file, then reuse the same migration command that you used when you started the migration. The command skips projects that uploaded successfully.

If a project migration fails, go to the migration script folder. See the `FAILED` file for more information.

Differences in SQO Between Polyspace Metrics and Polyspace Access

After you migrate your projects from Polyspace Metrics to Polyspace Access, you might notice differences when you examine your code quality against “Software Quality Objectives” (Polyspace Code Prover Access) (SQO).

The difference is due to the way Polyspace Metrics and Polyspace Access calculate the thresholds for the quality objectives. Polyspace Metrics looks at individual files to determine whether your code achieves a given SQO threshold. For instance, if file `foo.c` does not achieve threshold `SQO2`, then the whole project does not achieve that threshold.

Polyspace Access looks at the whole project to determine whether your source code meets a given SQO threshold. Even if file `foo.c` does not achieve the threshold, the whole project can still meet the specified quality objective threshold.

See Also

More About

- “Register Polyspace Desktop User Interface” on page 1-36
- “Upload Results to Polyspace Access” on page 2-2

Quick Start Guide for Polyspace Server and Access Products

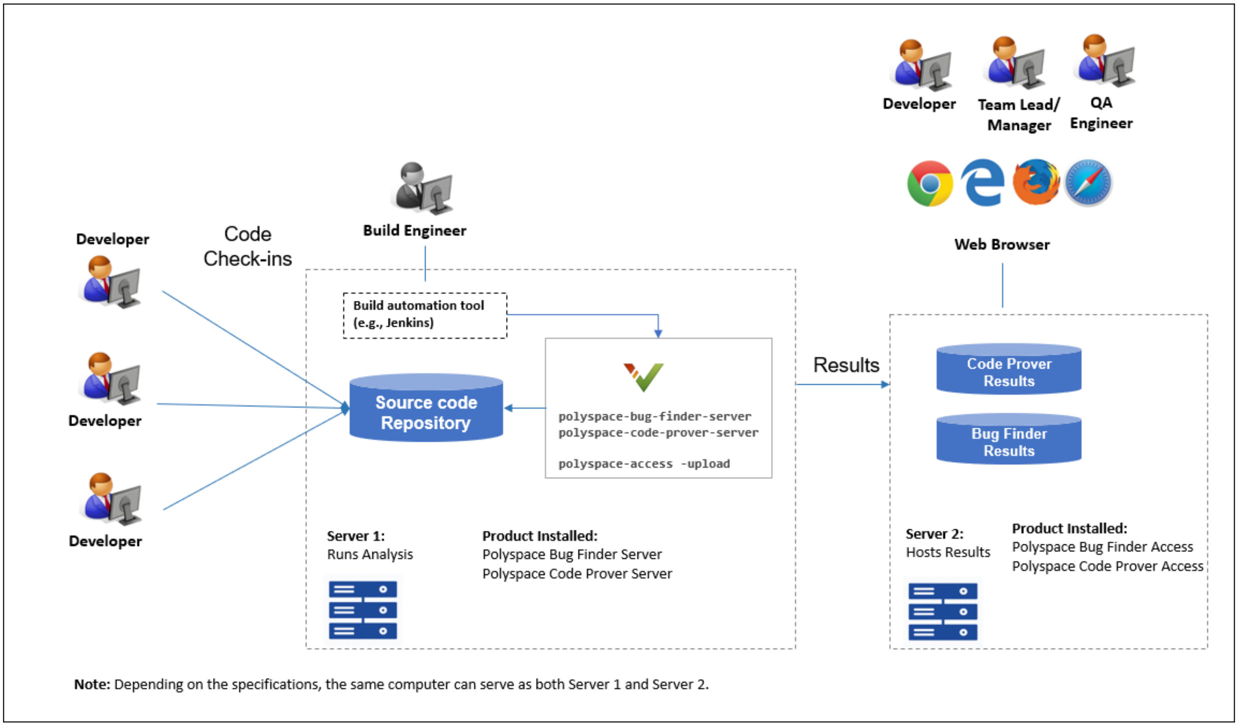
To avoid finding bugs late in the development process, run static analysis by using Polyspace products.

- **Polyspace Bug Finder** checks C/C++ code for bugs, coding standard violations, security vulnerabilities, and other issues.
- **Polyspace Code Prover** performs exhaustive checks for divide by zero, overflow, array access out of bounds, and other common types of run-time errors.

See also “Choose Between Polyspace Bug Finder and Polyspace Code Prover” (Polyspace Bug Finder Server).

If you run Polyspace checkers regularly as part of continuous integration, you can protect against regressions from new code check-ins. To run Polyspace on a server during continuous integration, use **Polyspace Bug Finder Server** and **Polyspace Code Prover Server**. To host the Polyspace analysis results, use **Polyspace Bug Finder Access** and **Polyspace Code Prover Access**.

A typical workflow looks like this figure.


Developer
Team Lead/ Manager
QA Engineer
Web Browser
Build Engineer
Developer
Code Check-ins
Developer
Developer
Build automation tool (e.g., Jenkins)
Source code Repository
polyspace-bug-finder-server
polyspace-code-prover-server
polyspace-access -upload
Results
Code Prover Results
Bug Finder Results
Server 1: Runs Analysis
Product Installed: Polyspace Bug Finder Server Polyspace Code Prover Server
Server 2: Hosts Results
Product Installed: Polyspace Bug Finder Access Polyspace Code Prover Access

Note: Depending on the specifications, the same computer can serve as both Server 1 and Server 2.

Installation

Prerequisites

Depending on the needs of your project, team or organization, you have decided to obtain a certain number of licenses of Polyspace Server and Polyspace Access products. This guide helps you to install individual instances of these products on a machine.

Install Polyspace Server

To install Polyspace Server products, download and run the MathWorks installer. Enter a license for the Polyspace Server products (or request a trial license). See also Request a Trial License. The Polyspace Server products are installed in a separate folder from other

MathWorks products. See also “Install Polyspace Server and Access Products” (Polyspace Bug Finder Server).

Install Polyspace Access

Before installing Polyspace Access, consider the number of users who will potentially review Polyspace results simultaneously. The system requirements depend on the number of simultaneous reviewers. See also “System Requirements for Polyspace Access” on page 1-3.

Polyspace Access consists of several services: a database to manage results, a user manager to handle user logins, a web server to show results, and a gateway to handle communications. The services are deployed in Docker containers. You can start the services from a common interface called the Cluster Operator.

To install Polyspace Access:

- Download the installer as a zip file.
- Unzip the file and start the Cluster Operator. From the Cluster Operator interface, start the various services. See “Install Polyspace Access”.

After installation, to see uploaded results, you and other reviewers can log in to:

```
https://hostName:portNumber/metrics/index.html
```

Setting Up Polyspace Analysis

Prerequisites

You or your IT department in your organization must install the required number of Polyspace Server and Polyspace Access instances. This guide helps you to set up a Polyspace analysis as part of continuous integration using a single instance of Polyspace Server and Polyspace Access.

To check that your Polyspace Server and Polyspace Access installations can communicate with each other, see “Check Polyspace Installation” (Polyspace Bug Finder Server).

Run Polyspace Server and Upload Results to Polyspace Access

You can run the Polyspace Server products at the command line of your operating system:

- To run the analysis, use the `polyspace-bug-finder-server` and `polyspace-code-prover-server` executables.
- To upload analysis results, use the `polyspace-access` executable. You can also use this executable to export the results from Polyspace Access as text files for archiving or email attachments.

You can run all Polyspace executables from the `polyspace/bin` subfolder of the Polyspace installation folder (for instance, `/usr/local/Polyspace Server/R2019a`). To start running Polyspace Server by using sample C source files and sample scripts, see:

- "Run Polyspace Bug Finder on Server and Upload Results to Web Interface" (Polyspace Bug Finder Server)
- "Send Email Notifications with Polyspace Bug Finder Results" (Polyspace Bug Finder Server)

You can also preconfigure the Polyspace analysis options from your build command (makefile), and then append a second options file with analysis specifications such as checkers. See "Prepare Scripts for Polyspace Analysis" (Polyspace Bug Finder Server).

If you have an installation of the Polyspace desktop products, you can prepare the analysis configuration in the user interface of the desktop products. You can then generate Polyspace options files to run during continuous integration. See "Configure Polyspace Analysis Options in User Interface and Generate Scripts" (Polyspace Bug Finder Server).

Include Polyspace Runs in Continuous Integration by Using Tools Such as Jenkins

Once you have working scripts to run a Polyspace analysis, you can run those scripts at predefined intervals using continuous integration tools such as Jenkins and Bamboo. In Jenkins, you can use a Polyspace plugin to point to your Polyspace installations and send email notifications to developers after the analysis, based on criteria such as new defects.

From within the Jenkins interface, search for and install the Polyspace plugin. For a quick start on using the Jenkins plugin and sample scripts, see the Polyspace plugin GitHub repository. For the full workflow with Jenkins, see "Sample Scripts for Polyspace Analysis with Jenkins" (Polyspace Bug Finder Server).

Create a Workflow for Result Reviewers

Depending on tools that you already use, you can set up a convenient workflow for result reviewers. For example:

Reviewers receive alerts for new results and log into Polyspace Access

- When new results are available, the continuous integration tool alerts a group of users. The email alert contains the Polyspace Access URL of the project where the results are uploaded.
- In the Polyspace Access interface, a reviewer can open this project URL, filter results based on files, and fix the issues or set a status for the results. See also:

  - “Filter and Sort Results”
  - “Address Polyspace Results Through Bug Fixes or Justifications”

Reviewers get customized email alerts with results in attachment

- Before upload to Polyspace Access, using the `-set-unassigned-findings` option of the `polyspace-access` executable, the continuous integration (CI) tool assigns owners to new analysis results based on file or component ownership or another criteria.
- After upload, using the `-export` option of the `polyspace-access` executable, the CI tool exports analysis results for each owner to a separate text file. The tool then sends the text file in an email attachment to the owner. The text file contains results with the corresponding URLs in the Polyspace Access interface.

  If you use Jenkins as your CI tool, the Polyspace plugin in Jenkins directly supports this workflow. See “Sample Scripts for Polyspace Analysis with Jenkins” (Polyspace Bug Finder Server).
- On receiving the email, the owner opens the attached text file, copies the URL of each result to their web browser and reviews the result.

Reviewers open tickets from bug tracking tools

- A reviewer, such as a quality engineer, reviews all new results and creates JIRA tickets for developers. See “Track Issue in Bug Tracking Tool”.
- Developers open each JIRA ticket and navigate to the corresponding Polyspace result in the Polyspace Access interface.