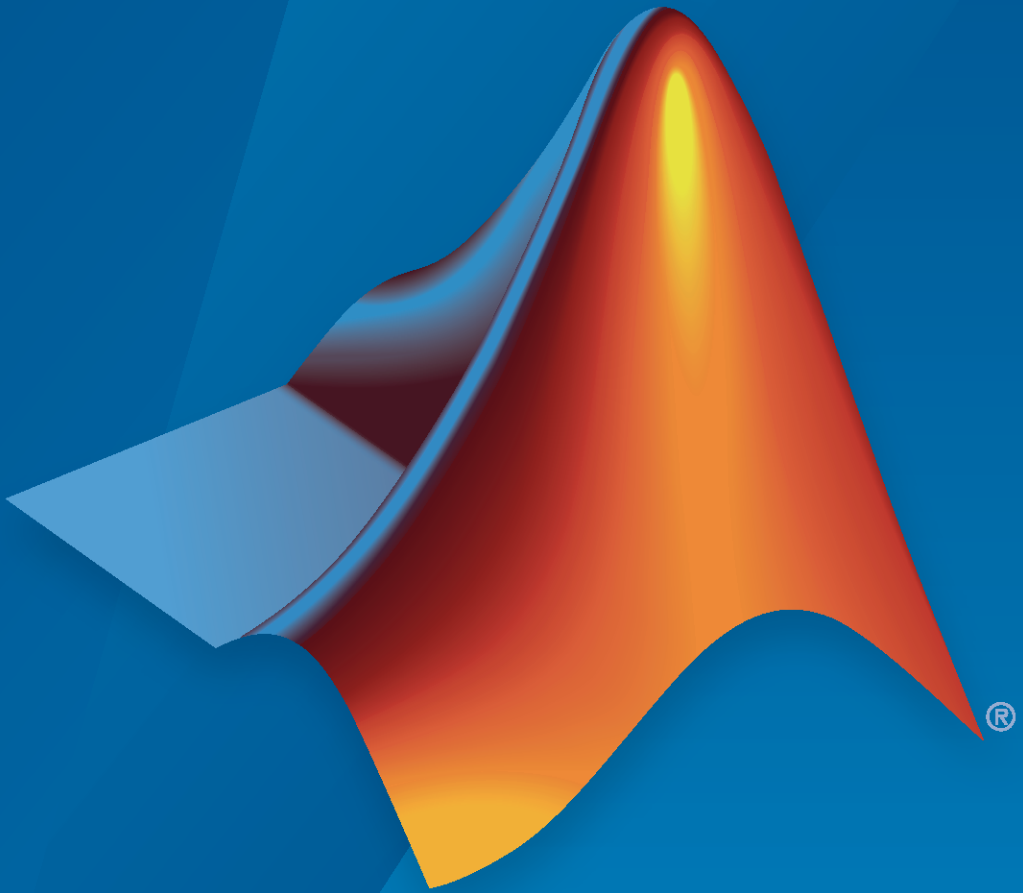
Polyspace® Bug Finder™ Access™

Getting Started Guide



R2021a

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)
March 2020	Online Only	Revised for Version 2.2 (Release 2020a)
September 2020	Online Only	Revised for Version 2.3 (Release 2020b)
March 2021	Online Only	Revised for Version 3.0 (Release 2021a)

Contents

1 Install Polyspace Bug Finder Access

2 Manage Polyspace Access License

3 Get Started with Polyspace Bug Finder Access

4 Install Polyspace as You Code

5 Get Started with Polyspace as You Code

Install Polyspace Bug Finder Access

Polyspace Bug Finder Access Product Description

Identify coding defects and review static analysis results

Polyspace Bug Finder Access provides a web browser interface for reviewing static code analysis results. It also provides Polyspace as you Code, a plug-in and analysis engine for performing static code analysis from within an integrated development environment (IDE) such as Visual Studio®, Visual Studio Code, or Eclipse™.

The web browser interface lets you review, assign, and resolve code analysis results produced by Polyspace Bug Finder Server™. The interface provides project dashboards displaying 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. You can also use the web browser interface to create and assign tickets in defect-tracking systems such as Jira and Redmine.

Polyspace as you Code checks compliance with coding rule standards such as MISRA C®, MISRA® C++, JSF++, CERT® C, CERT C++, and custom naming conventions while you code. It enables you to identify critical defects and security vulnerabilities early in development and without leaving your IDE.

System Requirements for Polyspace Access

Required Software

- The installation of Polyspace Access components requires Docker version 1.10 or later.
- Polyspace Access is compatible with Docker Engine on Linux®.

  To install Docker Engine on your machine, click one of these Docker supported Linux platforms and follow the installation instructions. The installation of Polyspace Access on a Linux platform is recommended.

  - Ubuntu
  - Debian
  - CentOS
  - Fedora

  Docker Engine is also available on other Linux distributions.. For Docker on SUSE and Red Hat, see Docker EE on Linux distros.
- The configuration of some Polyspace Access services requires the `openssl` toolkit to generate public and private keys. This toolkit is also required to configure Polyspace Access with HTTPS.
- To connect Polyspace Access with Polyspace desktop user interfaces over HTTPS, you must use the Java Platform, Standard Edition Development Kit (JDK). You use the JDK to generate a Java Key Store file.
- Polyspace Access licenses are network named user (NNU) licenses that require a license manager. Polyspace uses the FlexNet® Publisher (FLEXlm®) license manager. See “Install License Manager” on page 2-5.

Windows Requirements

To install Polyspace Access inside Linux virtual machines on Windows Server 2016 and 2019, you must:

- Enable virtualization in your BIOS.
- Install and enable Hyper-V.
- Create a virtual switch for Hyper-V.

You must also enable nested virtualization if you run Hyper-V inside of a Hyper-V VM.

Hardware and Other Requirements

- The minimum hardware configuration that is recommended for up to 100 users of Polyspace Access is:

  - 4 cores
  - 32 GB of RAM
  - 500 GB of disk space

Typically, only a reasonable fraction of users are concurrently interacting actively with the Polyspace Access web interface. To configure Polyspace Access for more than 100 users, contact MathWorks technical support.

- Data transfers between the server and client machines require a high speed network connection. A gigabit network connection is recommended.
- The Polyspace Access Cluster Admin does not support the Internet Explorer web browser.
- It is recommended to install Polyspace Access on physical machines. The installation of Polyspace Access on a virtual machine might result in up to a 50% overhead during I/O operations.

See Also

More About

- “Install Polyspace Access”

Storage and Port Configuration

Storage Configuration

- To ensure optimal data storage performance, use physical drives instead of networked storage solutions.
- The database is stored under `*polyspaceAccessRoot*appdata/polyspace-access` where *`polyspaceAccessRoot`* is the folder where you unzip the Polyspace Access installation image. Make sure that you have adequate disk space for the Database mount point.
- 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**: 40 GB

    Uploaded files are stored in this folder while they are transferred to the web server mount point.
  - **Upload directory**: 10 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**: 20 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**: 10 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 **Polyspace Access ETL** and the **Temporary upload directory** in the **Cluster Admin** settings.

Network Port Configuration

When you configure Polyspace Access, you specify a port number that client machines use to communicate with the Polyspace Access services. To avoid installation errors, and to ensure that the services are accessible, make sure that the port that you specify is open. To check whether a port *`portNumber`* is open, use these commands:

<table>
<tr><td><b>Windows® PowerShell</b></td><td><code>netstat -na | find "<i>portNumber</i>"</code></td></tr>
<tr><td><b>Linux</b></td><td><code>netstat -na | grep <i>portNumber</i></code></td></tr>
</table>

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

Check the availability of these ports if you install Polyspace Access on a single node with a default configuration:

- 9443 — Polyspace Access default port.
- 27000 — Inbound port of license manager that is required to manage license checkouts.
- 3268 — Lightweight Directory Access Protocol (LDAP) server default port. This port is not required if you do not use your company LDAP.

Depending on your Polyspace Access configuration, you might need to check the availability of a port for your bug tracking tool.

See Also

More About

- “Install Polyspace Access”
- “Database Backup” on page 1-43
- “Configure and Start the Cluster Admin” on page 1-13

Create a Linux Virtual Machine by Using Hyper-V

You can install Polyspace Access on Windows Server® 2016 and 2019 by creating a virtual machine (VM) that runs a Linux distribution, and then installing Polyspace Access inside that VM.

Warning The use of Polyspace Access inside a VM might result in up to a 50% overhead during I/O operations compared to using Polyspace Access on a physical machine.

Prerequisites

Before you create the VM:

- Make sure that Hyper-V is enabled on your machine.

  Open Windows PowerShell™ by pressing the Windows+X keys and clicking **Windows PowerShell (Admin)**.

  In the PowerShell command prompt, enter:

  ```
  (Get-WindowsOptionalFeature -featurename Microsoft-hyper-v -online).state
  ```

  If the command does not return `Enabled`, enter:

  ```
  Install-WindowsFeature -Name Hyper-V -IncludeManagementTools -Restart
  ```

  The command enables Hyper-V and restarts your machine.

  Open the Hyper-V Manager by pressing the Windows key and typing HyperV, then click **Action > Connect to Server** and select **Local computer**.
- Make sure that an external virtual switch has been created in Hyper-V.

  In a PowerShell command prompt, enter:

  ```
  Get-VMSwitch | where SwitchType -eq 'External'
  ```

  If the command does not return anything, follow these instructions to create an external virtual switch. Running this command might require administrator privileges.
- Download an ISO image for a Linux distribution that is supported by Docker, for instance Ubuntu Server. For a list of Linux distributions that are available for Docker Engine or Docker Engine Enterprise(EE), see supported platforms for Docker Engine and Docker EE on Linux distros.
- Download and install the network license manager. See “Install License Manager” on page 2-5.

Create a Virtual Machine

To create a virtual machine, open the Hyper-V manager. In the **Actions** pane, click **New > Virtual Machine**.

Follow the prompts in the **New Virtual Machine Wizard** window.

- For the **Specify Generation** step, select **Generation 2**.
- For the **Assign Memory** step, allocate enough memory to meet the requirements on page 1-3 for Polyspace Access. The recommended minimum memory is 32 GB.

- For the **Configure Networking** step, select the switch that corresponds to the external connection type.
- For the **Connect Virtual Hard Disk** step, the size of the virtual hard drive must meet the requirements on page 1-3 of the Polyspace Access database. The recommended minimum disk size is 500 GB.
- For the **Installation Options** step, select **Install an operating system from a bootable image file**, and provide the path to the Linux ISO image that you downloaded.

After you click **Finish** and the wizard closes, right-click the newly created VM in the **Virtual Machines** pane and click **Settings**. In the settings window, click **Security** in the left pane, select **Enable Secure Boot** and, choose **Microsoft UEFI Certificate Authority** from the **Template** drop-down. Secure boot helps preventing the loading utility of the operating system from running unauthorized code at boot time. For a list of Linux distributions that Microsoft supports for secure boot, see Supported Linux and FreeBSD virtual machines for Hyper-V on Windows.

Start and Configure the Virtual Machine

To start the virtual machine (VM), in the Hyper-V manager, right-click the VM name in the **Virtual Machines** pane, and then click **Connect**. If this is the first time that you are starting the VM, follow the prompts to install the Linux distribution you specified in the **Installation Options** step when you created the VM.

During this installation process, you specify a host name for the Linux machine and a user name and password to log into the Linux machine. Enter this password when you use the `sudo` command in later configuration steps.

After you install the Linux distribution, restart the VM and open a Linux command-line terminal.

- Install the Docker engine. For installation instructions, see the Docker documentation ,for instance Get Docker Engine - Community for Ubuntu.

  Once you install the Docker engine, add the current user to the `docker` group. Only users that are in the `docker` group can run Docker commands. In the terminal, enter:

  ```
  sudo usermod -aG docker $USER
  ```
- Install the `openssl` utility. The utility allows you to generate public/private key pairs to configure the **User Manager** service and to generate the necessary certificates if you enable HTTPS for Polyspace Access. For instance, on Ubuntu, enter this command:

  ```
  sudo apt install openssl
  ```

  If `openssl` is already installed, this command has no effect.
- Install the `openssh-server` server and make sure that port 22 is enabled in the firewall configuration. You can then remote into the Linux machine by using SSH or securely transfer files to the Linux machine. For instance, on Ubuntu, enter these commands:

  ```
  sudo apt install openssh-server
  sudo ufw allow 22
  ```

  If `openssh-server` is already installed, the install command has no effect. Once you complete this step, you can use a command such as `scp` to securely transfer files between your Windows Server 2016 machine and the Linux VM.

For example, if you use user name `accessUser` to log into the Linux VM with host name `access-vm-lnx`, you can transfer file `myFile.txt` by entering this command from the Windows Server machine:

```
scp pathTO\myFile.txt accessUser@access-vm-lnx:~
```

The command copies the file to folder `/home/accessUser` on the Linux VM.

`pathTO` is the path to `myFile.txt`.

- Once you complete the previous configuration steps, restart the VM.

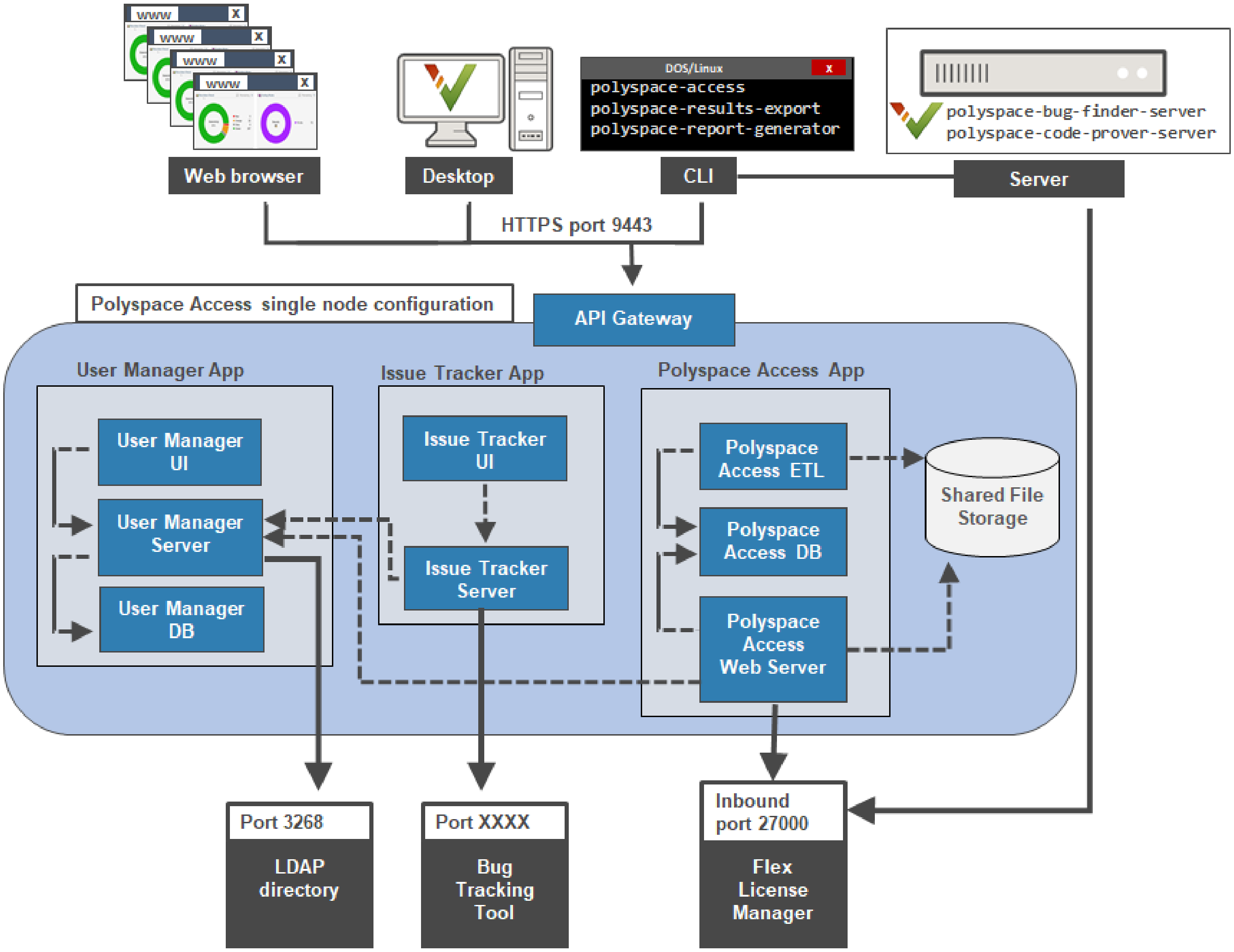
To install Polyspace Access, see “Install Polyspace Access” and “Manage Polyspace Access License”.

Prepare Your Installation

Polyspace Access provides a web browser interface so that you can review Polyspace analysis results that are hosted on a centralized database. When you install Polyspace Access, you install these apps:

- **User Manager App**: Authenticates user logins against your company Lightweight Directory Access Protocol (LDAP) or against a custom internal database of users. The app issues signed JSON Web Tokens to authenticated users and provides a user interface to manage the custom database of users.
- **Issue Tracker App**: Manages the communication between Polyspace Access and your bug tracking tool (BTT) software and provides a user interface to create BTT tickets.
- **Polyspace Access App**: Manages results uploads to the Polyspace Access database and results exports from that database. Provides a user interface to review results.

Each app contains services that are deployed inside docker containers. A separate **Gateway** service handles communications between Polyspace Access and client machines.



Before you begin your installation, decide whether to use the HTTPS protocol and how you to configure user authentication and bug tracking tool integration.

Prerequisites

- Install the required software and make sure that your system meets the minimum hardware requirements. See “System Requirements for Polyspace Access” on page 1-3.
- Verify that you have enough data storage available and that the ports that you use are available and not blocked by your firewall. See “Storage and Port Configuration” on page 1-5.
- Make sure that the license manager is installed and running, and that the license manager options file includes the users to whom you grant right-to-use privileges for Polyspace Access. See “Manage Polyspace Access License”.
- To avoid any potential issues with license file operation, consider upgrading the network license manager software whenever you upgrade Polyspace Access. See “Update Network License Manager Software”.
- Create and configure a Linux virtual machine (VM) if you are installing Polyspace Access inside a Linux VM on Windows Server 2016 or 2019. See “Create a Linux Virtual Machine by Using Hyper-V” on page 1-7.
- If you enable HTTPS to encrypt communications between Polyspace Access and client machines, obtain an SSL private key and a signed certificate from a certificate authority or use self-signed certificates. See “Choose Between HTTP and HTTPS Configuration for Polyspace Access” on page 1-14
- If you configure the Polyspace Access to use the HTTPS protocol, you must generate a Java® Key Store (`JKS`) file to enable communications between Polyspace Access and the Polyspace desktop interface, or the `polyspace-results-export` and `polyspace-report-generator` binaries. See “Generate a Client Keystore” on page 1-41.

User Manager Prerequisites

- The **User Manager** configuration requires the generation of an SSL private key file. See “Configure User Manager” on page 1-20. This private key must be different from the private key that you use for the HTTPS configurations.
- If you use your company LDAP to authenticate user logins, contact your LDAP administrator to:
  - Obtain the LDAP URL and LDAP base that your organization uses.
  - Obtain LDAP login credentials if access to the LDAP server is password-protected.
  - Discuss an LDAP search filter that enables you to retrieve specific subsets of users from the LDAP database. See LDAP filters.
- If you use LDAP configured over SSL (LDAPS), you must add the LDAP SSL certificate to the certificate trust store file that you use for Polyspace Access. See “Configure the User Manager for LDAP over SSL” on page 1-24.

Issue Tracker Prerequisites

- The **Issue Tracker** configuration requires the URL that you use to connect to your BTT interface.
- If you use the Jira software with the OAuth authentication method, you must first create an application link in Jira. See the first step on this page.

- If you use the Redmine BTT, contact your Redmine administrator to obtain the Redmine API key.
- If you use a BTT configured by using HTTPS, you must add the BTT SSL certificate to the certificate trust store file that you use for Polyspace Access. See “Add BTT Instance Configured by Using HTTPS” on page 1-32.

Configure and Start the Cluster Admin

The Cluster Admin 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 Cluster Admin, make sure that:

- You have downloaded the Polyspace Access installation image. To download the installation image:

  1. Go to the MathWorks® download page and click the **Download Rxxxxy** button. You may be required to log in to your MathWorks account to complete this step.
  2. On the following page, select the Polyspace Access link under Additional Rxxxxy Product Downloads.

  Rxxxxy corresponds to a release number, for instance R2021a.
- Docker is running on your machine. At the command line, type:

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

  If you get an error message, 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
  ```

  To add the current user to the `docker` group, use the command:

  ```
  sudo usermod -aG docker $USER
  ```

Unzip Installation Image and Start Cluster Admin Agent

The Cluster Admin `admin-docker-agent` binary is included with the `polyspace-access-`*`VERSION`*`.zip` installation image for Polyspace Access. *`VERSION`* is the release version, for instance R2021a. After you download the installation image, unzip it to extract these files and folders:

```
admin-docker-agent*
admin-docker-agent.exe*
admin.tar
appdata/
download/
gateway.tar
issuetracker-server-main.tar
issuetracker.tar
issuetracker-ui-main.tar
lm/
polyspace-access-db-main.tar
polyspace-access-etl-main.tar
polyspace-access.tar
polyspace-access-web-server-main.tar
products/
usermanager-db-main.tar
usermanager-server-main.tar
```

```
usermanager.tar
usermanager-ui-main.tar
VERSION
```

To start the `admin-docker-agent` 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:

```
admin-docker-agent
```

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:

```
time="2020-07-10T14:23:11Z" level=info msg="Cluster Admin started. You can now connect to the Cluster Admin through
your web browser at http://localhost:9443/admin 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 **Cluster Admin**.

By default, the **Cluster Admin** uses the HTTP protocol and starts with host name localhost and port 9443. To configure the **Cluster Admin** with HTTPS, see “Choose Between HTTP and HTTPS Configuration for Polyspace Access” on page 1-14. If the port is already in use, you get `Permission denied` error message. Use the flag `--port` to specify a different port number, for instance:

```
admin-docker-agent --port 9999
```

To reset the password, press **CTRL+C** to stop the `admin-docker-agent` binary and enter this command:

```
admin-docker-agent --reset-password
```

To view the new password, restart the binary.

The **Cluster Admin** agent creates a `settings.json` file the first time it starts, and stores this file in the same folder as the `admin-docker-agent` binary by default. Ensure that only the user who starts the `admin-docker-agent` has read/write permissions on the `settings.json` file.

Choose Between HTTP and HTTPS Configuration for Polyspace Access

HTTP Configuration

By default, the **Cluster Admin** uses the HTTP protocol. When you start the `admin-docker-agent` binary, you do not need to specify any additional flags.

HTTPS Configuration

To encrypt the data between Polyspace Access and client machines, configure the **Cluster Admin** with the HTTPS protocol. To complete the configuration, provide an SSL certificate and the private key that you used to generate the certificate as `PEM` files.

Do not reuse the private key file that you use for the **Authentication private key file** in the **User Manager** configuration.

It is recommended that you 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.

Secure your 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 **Cluster Admin** 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 **Cluster Admin** enables HTTPS for the API Gateway service. This service handles all communications between the other Polyspace Access services and client machines.

The SSL certificate, private key, and CA files that you provide when you start the `admin-docker-agent` binary are reused in the **Nodes** settings, unless the node is already configured with a different set of files. When you select **Enable SSL** for a node, you enable HTTPS for all the services installed on that node. Enabling SSL in the **Nodes** settings affects communications only between the Polyspace Access services, and between those services and the bug tracking tool and LDAP servers. To view and make changes to the **Nodes** settings, click **Configure Nodes** on the **Cluster Dashboard**.

By default, all services are installed on the same node and the services ports are not exposed. You do not need to enable HTTPS for the **User Manager**, **Issue Tracker**, and **Polyspace Access** services unless you install these services on different nodes, or you start the `admin-docker-agent` binary with option `--force-exposing-ports`.

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 `admin-docker-agent` binary.

   ```
   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, `admin_cert.cer`.
3. Start `admin-docker-agent` and use the generated private key and signed certificate. Specify the FQDN *`hostName`* and the full path to the certificate trust store file `ca-certificates.crt` :

   ```
   ./admin-docker-agent --hostname hostName\
   --ssl-cert-file fullPathTo/admin_cert.cer \
   --ssl-key-file fullPathTo/myKey.key \
   --ssl-ca-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 **Cluster Admin** web interface, your browser considers the connection secure if the browser uses the certificate trust store that you specify for `--ssl-ca-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 `admin-docker-agent` binary.

   ```
   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 `admin-docker-agent` binary and use the generated `certificate.pem` and `private_key.pem` files. Specify the FQDN *`hostName`*.

   ```
   ./admin-docker-agent --hostname hostName\
   --ssl-cert-file fullPathTo/certificate.pem \
   --ssl-key-file fullPathTo/private_key.pem \
   --ssl-ca-file  fullPathTo/certificate.pem
   ```

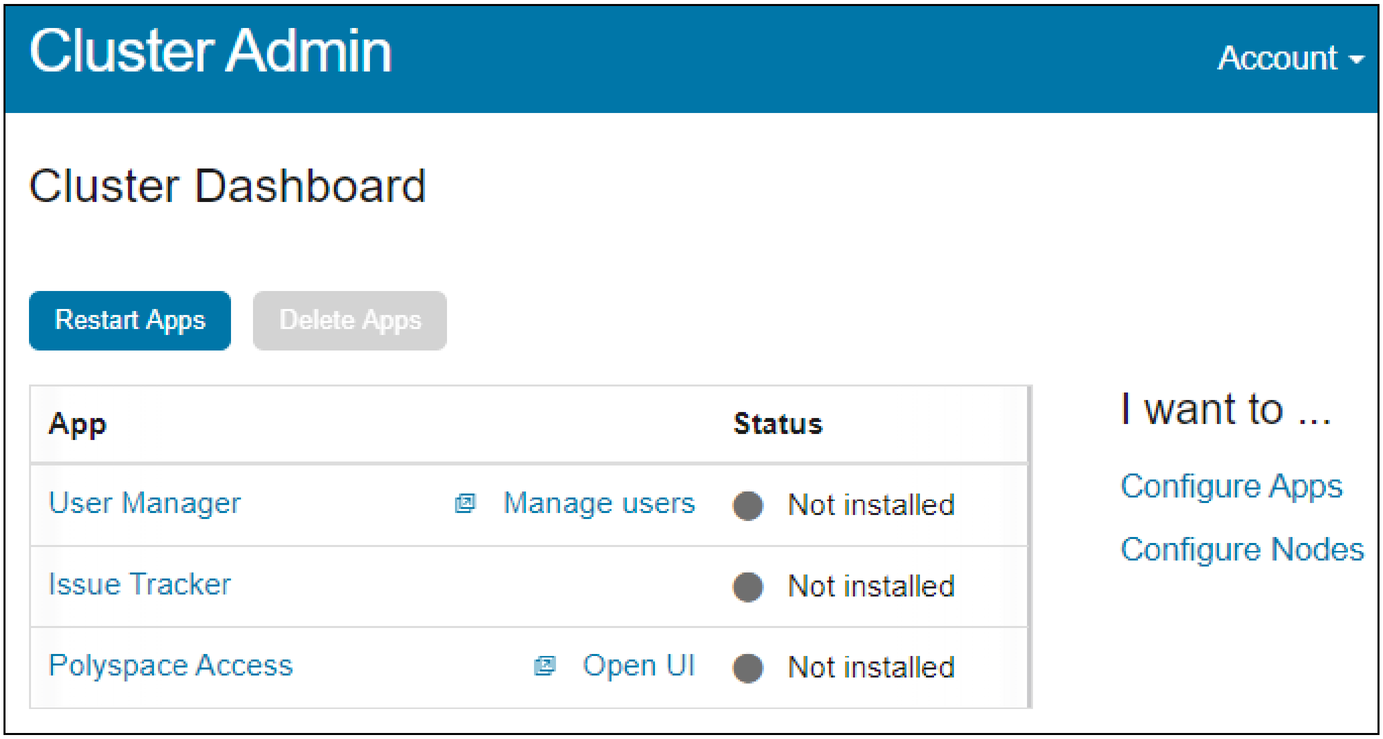
   The *`hostName`* you specify in this command must match the *`hostName`* you specified in step 1. The self-signed `certificate.pem` file is also used as the certificate trust store file. *`fullPathTo`* is the full file path. If you use relative paths, you get an error message.

When you open the **Cluster Admin** web interface, your browser shows a warning about the certificate being untrusted.

Open the Cluster Admin Interface

After you configure and start the **Cluster Admin**, open your web browser and go to URL specified in the command-line output when you started the `admin-docker-agent` binary.

Log in with the initial password that you obtained when you started the **Cluster Admin** agent. If this is your first time logging in, follow the prompts.

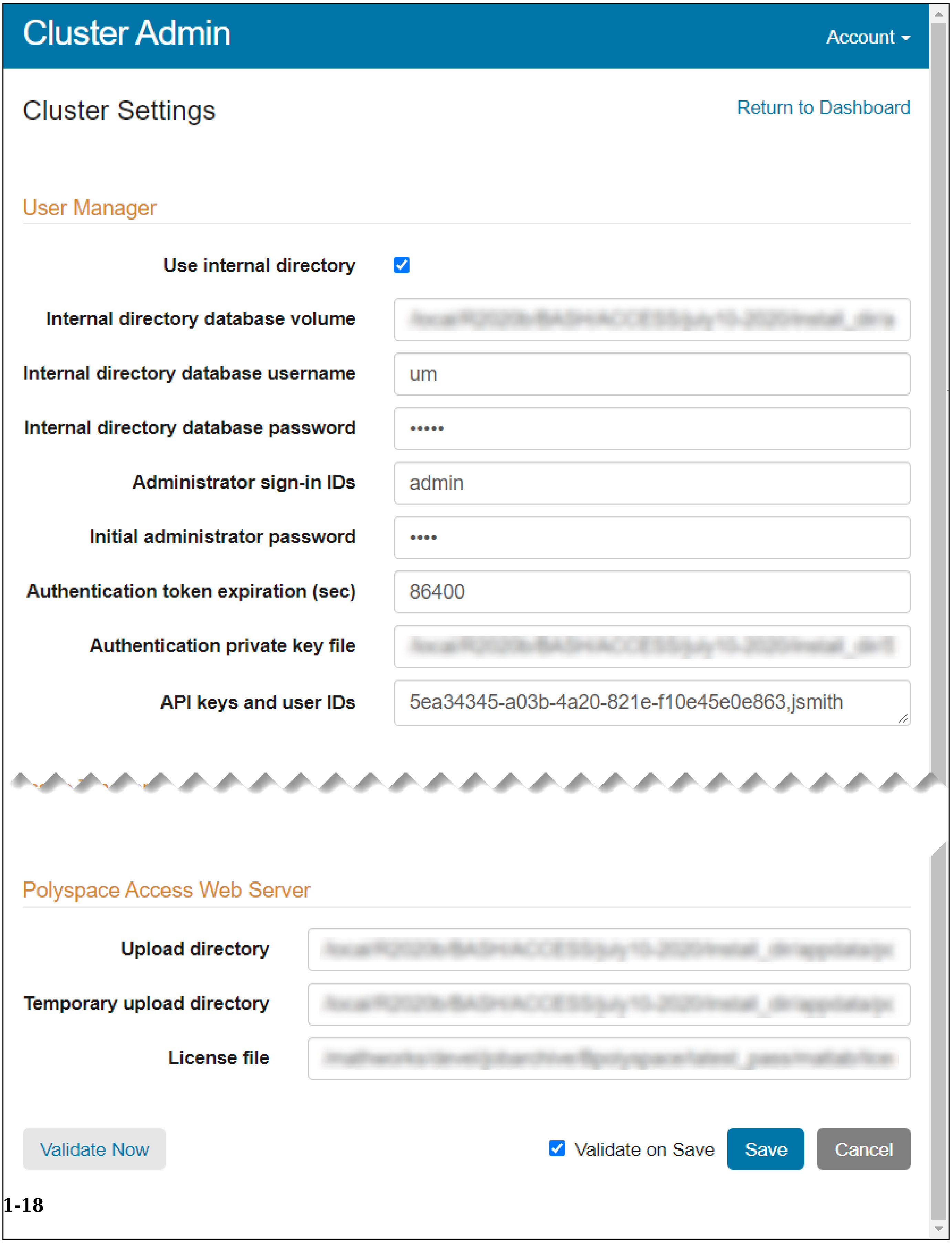


App		Status
User Manager	Manage users	● Not installed
Issue Tracker		● Not installed
Polyspace Access	Open UI	● Not installed

It is best practice to change your **Cluster Admin** password after your first login. To set a new password, click **Account** in the upper right corner of the web interface and select **Change**

password. Share the **Cluster Admin** password only with users who configure and manage the Polyspace Access services.

On the **Cluster Dashboard**, click **Configure Apps** to go to the **Cluster Settings**. Whenever you change the settings, return to the **Cluster Dashboard** and click **Restart Apps** for the changes to take effect. To save partially filled settings, clear **Validate on Save**.



Cluster Admin

Cluster Settings

User Manager

Use internal directory	☑
Internal directory database volume	[redacted]
Internal directory database username	um
Internal directory database password	•••••
Administrator sign-in IDs	admin
Initial administrator password	••••
Authentication token expiration (sec)	86400
Authentication private key file	[redacted]
API keys and user IDs	5ea34345-a03b-4a20-821e-f10e45e0e863,jsmith

Polyspace Access Web Server

Upload directory	[redacted]
Temporary upload directory	[redacted]
License file	[redacted]

☑ Validate on Save

Note On Windows systems, all the file paths that you specify must point to local drives.

See Also

`admin-docker-agent`

More About

- "Prepare Your Installation" on page 1-10
- "Configure User Manager" on page 1-20
- "Configure Issue Tracker" on page 1-30
- "Configure Polyspace Access App Services" on page 1-34

Configure User Manager

The **User Manager** manages the authentication of the Polyspace Access user logins. Depending on your configuration, user logins are authenticated by checking user names and passwords against information in your company Lightweight Directory Access Protocol (LDAP) server, or against user credentials stored in the **User Manager** internal directory. See "Authenticate Users from Your Organization LDAP Server" on page 1-20 and "Authenticate Users from Internal Directory" on page 1-24 respectively.

On the **Cluster Dashboard**, click **Configure Apps** to go to the **Cluster Settings**. Whenever you change the settings, return to the **Cluster Dashboard** and click **Restart Apps** for the changes to take effect. To save partially filled settings, clear **Validate on Save**.

Note On Windows systems, all the file paths that you specify must point to local drives.

Authenticate Users from Your Organization LDAP Server

To use the LDAP server of your organization, in the **User Manager** settings, clear **Use Internal directory**. Contact your network administrator to obtain the LDAP URL, base, and any other setting and to determine whether your LDAP server uses the Active Directory Global Catalog feature.

LDAP URL	Enter the LDAP URL as:
	`ldap://HOST:PORT`
	`HOST` is the LDAP host and *`PORT`* is the LDAP port number. The LDAP server uses different default port numbers if you configure it to use the global catalog. See "Configure User Manager for LDAP Server That Uses Global Catalog" on page 1-24.
	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 SSL" on page 1-24.
	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 who has 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 who has 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 `admin-docker-agent` binary.

Enable LDAP pagination	Enable this setting to control the rate at which your LDAP server returns results. You typically use this setting if you query a large set of users and the LDAP server has limits on the number of entries that it returns, or if the client making the query has limited resources. Before you enable this setting, ensure that: • Your LDAP server supports Simple Paged Results Control. • If you provide an **LDAP username** and **LDAP password**, pagination is not disabled for these credentials. The **User Manager** log shows LDAP result code 11 when pagination is disabled for the credentials that you provide. • The LDAP server is configured to handle a reasonable number of simultaneous paged queries. The **User Manager** log shows LDAP result code 53 if the number of simultaneous queries exceeds the LDAP server limit.
LDAP page size	Set the number of results that the LDAP server returns per page, for example 1000. This setting is available only if you select **Enable LDAP pagination**.
LDAP base	You can retrieve this parameter by using an LDAP explorer tool. For instance, connect to your LDAP server through 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 startup instead of loading 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 who have a common name (CN) attribute that starts with "test". Use parentheses to combine multiple filter expressions in an AND (&) or OR (|) clause. For instance, `(|(CN=jdoe)(department=foo))` matches all users who have CN attribute "jdoe" or department attribute "foo". The default search filter is `objectClass=organizationalPerson`. For more information about search filters, see the LDAP filters. To check whether a search filter is returning a subset of users, enter a user name from that subset in the **Administrator sign-in IDs** field and click **Validate Now** at the bottom of the page. You get a warning if the user name cannot be found.
Other LDAP settings	Leave these settings unchanged unless instructed otherwise by your LDAP administrator. Polyspace Access does not use the LDAP display name, email, and image attributes.

Administrator sign-in IDs	Enter a comma-separated list of user names to set specific users as Polyspace Access administrators. A user that has Polyspace Access administrator privileges can: • Assign or unassign other users as administrators and manage permissions for all projects from the Polyspace Access web interface. See “Manage Permissions in Polyspace Access Web Interface” on page 3-18. • Remove or add project owners. • “Manage Users in Polyspace Access” on page 1-28 The username must match an existing user in the LDAP directory. To remove a user as a Polyspace Access administrator: 1 Remove the user name from the list, save your changes, then restart the apps. 2 After the restart, a Polyspace Access administrator must unassign the user from all top-level folders in the **Project Explorer**, in the Polyspace Access 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`.
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 into 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 closes unexpectedly, 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 are 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. Use this settings to set the licensing timeout. Polyspace Access ignores the license timeout value that you set through the license manager options file (`MLM.opt`) by using the `TIMEOUT feature seconds` syntax.

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. Do not reuse the private key that you use to generate the SSL certificates, which you provide if you enable the HTTPS protocol.

API keys and user IDs	Enter an API key value and user name pair to assign an API key to a user. For example, to assign an API key to `jsmith`, enter:
	`5ea34345-a03b-4a20-821e-f10e45e0e863,jsmith`
	To assign API keys to other users, enter additional API key and user name pairs on separate lines. Each API key value must be unique.
	You pass the API key with the flag `-api-key` to these commands that require Polyspace Access credentials:
	• `polyspace-access`
	• `polyspace-results-export`
	• `polyspace-report-generator`
	The commands use the API key as a login credential for the corresponding users. If a user updates his or her password, you do not have to update the API key. If you use the API key as part of an automation script, make sure that the user associated with the key has enough permissions to perform all the operations in the script. See “Manage Project Permissions” on page 3-18.
	You can assign any combination of alphanumeric characters as an API key to a user. For example, to generate universally unique identifiers (UUID) for the API key, use these commands:
	• **Windows PowerShell**
	`[guid]::NewGuid()`
	• **Linux**
	`uuidgen`

If you add new users to your LDAP directory:

- Add those users to the `MLM.opts` file to grant the users right-to-use privileges for Polyspace Access. See “Manage Named Users for Polyspace Access” on page 2-6.
- In the **Cluster Dashboard**, click **Restart Apps** to add the users to the Polyspace Access list of users. See also “Manage Users in Polyspace Access” on page 1-28.

Configure the User Manager for LDAP over SSL

If you use an LDAP server configured over SSL (LDAPS), add the LDAPS SSL certificate to the certificate trust store file that you specify in the **CA File** field of the **Nodes** settings. To view these settings, click **Configure Nodes** on the **Cluster Dashboard**. Depending on your trust store file, the LDAP SSL certificate might already be included in the trust store.

The certificate trust store file typically corresponds to the file that you provide with `--ssl-ca-file` when you configure the **Cluster Admin** with HTTPS. See “Choose Between HTTP and HTTPS Configuration for Polyspace Access” on page 1-14.

For instance, on a Linux Debian distribution, to add LDAP certificate `ldaps_cert.pem` to trust store file `trust_store.pem`, use this command:

```
cat trust_store.pem ldaps_cert.pem > combined_cert.pem
```

The command combines the content of the two files and outputs file `combined_cert.pem`. If you use a self-signed certificate to configure HTTPS, add the LDAP certificate to the self-signed certificate.

To complete the configuration, enter the path to `combined_cert.pem` in the **CA File** field of the **Nodes** settings, save your changes, return to the dashboard, and restart the Apps.

If you did not configure the **Cluster Admin** by using HTTPS, specify the path to the LDAP SSL certificate in the **CA File** field.

Configure User Manager for LDAP Server That Uses Global Catalog

The global catalog (GC) is a mechanism that enables you to add users from different Active Directory® (AD) servers to Polyspace Access without having to provide information about those servers. For more information, see Global Catalog. If your LDAP server is configured to use the GC, you must specify a GC-specific port number when you provide the LDAP URL to the **User Manager** service. If you use secure LDAP (LDAPS), the default port for LDAP servers that use GC is 3268 or 3269. To determine whether your LDAP server is configured to use the GC, contact your LDAP administrator.

If you specify an incorrect port number, the **User Manager** service cannot communicate with the LDAP server. When you inspect the **User Manager** log, you get error messages similar to these error messages .

```
LDAP server 'ldap://dc-00.ad.mathworks.com:389' did not recognize
base DN 'DC=ad,DC=mathworks,DC=com' and search base ''.
...
Unprocessed Continuation Reference(s)
```

To save the **User Manager** log to a file `out.log`, use this command.

```
docker logs -f usermanager-server-main >> out.log 2>&1
```

The GC holds only a subset of attributes for each user from the different Active Directory (AD) servers. The LDAP ID attribute that you specify in the cluster operator settings must be available in this subset of attributes when the GC is enabled. If the LDAP ID is not available in the GC, the corresponding user is not added to Polyspace Access.

Authenticate Users from Internal Directory

If you cannot or choose not to use the LDAP server of your organization, you can create users who have custom user names and passwords in the **User Manager** interface. In the settings, select **Use**

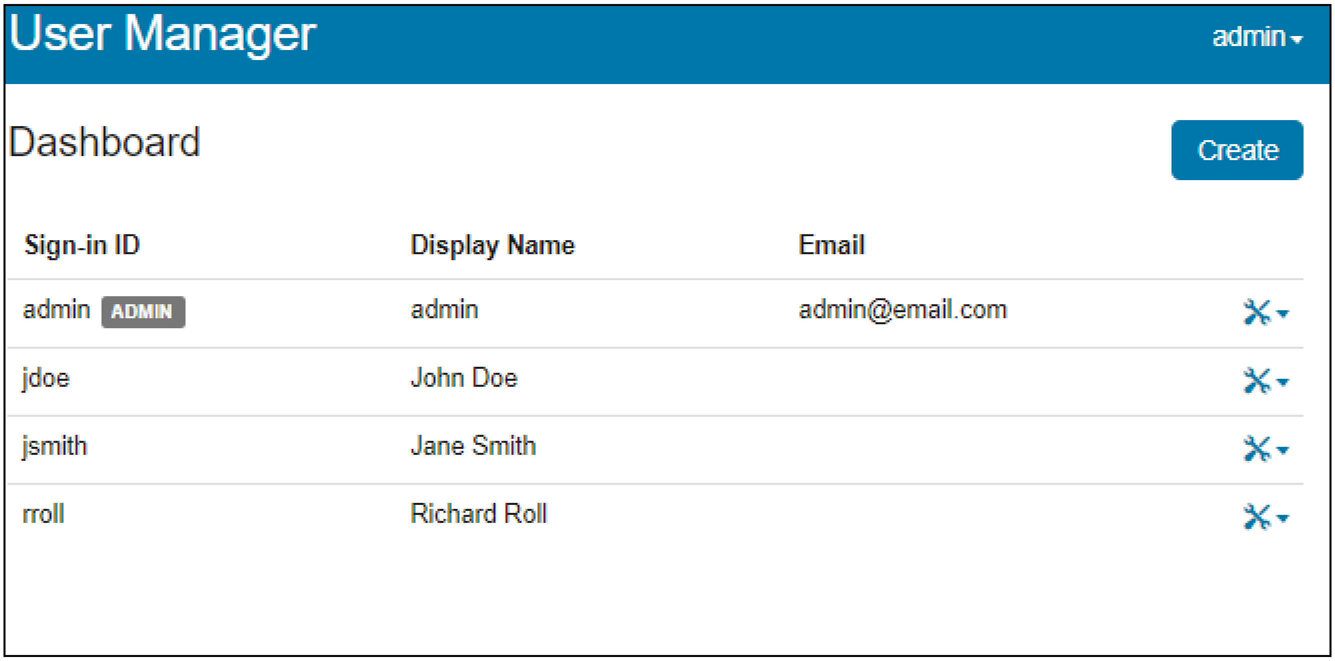
Internal directory. The **User Manager** checks user logins against the user credentials that you create. The credentials are stored in an internal directory database.

Internal directory database volume	Specify the full path to the folder where you store the database. By default, the database is stored under `appdata/usermanager` in the folder where you extracted the Polyspace Access installation image.
Internal directory database username	Use the user name and password that you specify in these two field if you need to interact with the internal database by using PostgreSQL commands. The default user name is 'UM" and the default password is "trust".
Internal directory database password	
Administrator sign-in IDs	Enter a comma-separated list of user names to set specific users as Polyspace Access administrators. A user that has Polyspace Access administrator privileges can: • Assign or unassign other users as administrators and manage permissions for all projects from the Polyspace Access web interface. See “Manage Permissions in Polyspace Access Web Interface” on page 3-18. • Remove or add project owners. • “Manage Users in Polyspace Access” on page 1-28 To remove a user as a Polyspace Access administrator: 1 Remove the user name from the list, save your changes, then restart the apps. 2 After the restart, a Polyspace Access administrator must unassign the user from all top-level folders in the **Project Explorer**, in the Polyspace Access 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`. The users that you specify in this field are also **User Manager** administrators. These users can: • Create, edit, or delete other users in the **User Manager** interface, except for other administrators. • Reset user passwords, except for other administrators. To remove a **User Manager** administrator: 1 Remove the user name from the list, save your changes, then restart the apps. 2 After the restart, an administrator must log into the **User Manager** interface to delete that user from the database.
Initial administrator password	Password that you use to log into the **User Manager** interface. This field is visible only if you select **Use Internal directory**. The default password is "`pass`".

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 into 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 closes unexpectedly, 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 are 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. Use this settings to set the licensing timeout. Polyspace Access ignores the license timeout value that you set through the license manager options file (`MLM.opt`) by using the `TIMEOUT feature seconds` syntax.
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. Do not reuse the private key that you use to generate the SSL certificates, which you provide if you enable the HTTPS protocol.

API keys and user IDs	Enter an API key value and user name pair to assign an API key to a user. For example, to assign an API key to `jsmith`, enter:
	`5ea34345-a03b-4a20-821e-f10e45e0e863,jsmith`
	To assign API keys to other users, enter additional API key and user name pairs on separate lines. Each API key value must be unique.
	You pass the API key with the flag `-api-key` to these commands that require Polyspace Access credentials:
	• `polyspace-access`
	• `polyspace-results-export`
	• `polyspace-report-generator`
	The commands use the API key as a login credential for the corresponding users. If a user updates his or her password, you do not have to update the API key. If you use the API key as part of an automation script, make sure that the user associated with the key has enough permissions to perform all the operations in the script. See “Manage Project Permissions” on page 3-18.
	You can assign any combination of alphanumeric characters as an API key to a user. For example, to generate universally unique identifiers (UUID) for the API key, use these commands:
	• **Windows PowerShell**
	`[guid]::NewGuid()`
	• **Linux**
	`uuidgen`

To create or manage users, return to the **Cluster Dashboard**, restart the **User Manager** app, then click **Manager users** to open the **User Manager** interface. Only users listed in the **Administrator sign-in IDs** field of the **User Manager** settings can open the **User Manager** interface and manage users.



Sign-in ID	Display Name	Email	
admin ADMIN	admin	admin@email.com	✕▾
jdoe	John Doe		✕▾
jsmith	Jane Smith		✕▾
rroll	Richard Roll		✕▾

After you create a user:

- Add the user sign-in ID to the `MLM.opts` file to grant that user right-to-use privileges for Polyspace Access. See “Manage Named Users for Polyspace Access” on page 2-6.
- In the **Cluster Dashboard**, click **Restart Apps** to add the user to the Polyspace Access list of users. See also “Manage Users in Polyspace Access” on page 1-28.

Manage Users in Polyspace Access

When the **Polyspace Access Web Server** service starts, Polyspace Access loads a list of users to its database from your organization's LDAP database or from the **User Manager** internal database. You can select users from only this list when you assign analysis findings or manage permission for a project or folder.

Polyspace Access periodically checks for new users from the LDAP database or the **User Manager** internal database, and adds those users to the Polyspace Access database. To add the new users to the Polyspace Access database manually instead, in the **Cluster Dashboard**, click **Restart Apps**.

Polyspace Access does not remove users from its database of users if you remove a user from the LDAP database or from the **User Manager** internal database, even if you restart the **Polyspace Access Web Server** service.

To remove users from the Polyspace Access list of users:

1. Click **Restart Apps** in the **Cluster Dashboard**.
2. In a web browser, enter the URL that you use to “Open the Polyspace Access Web Interface” on page 1-38 and append `/users/list/removed` to the URL, for example `https://access-machine.company.com:9443/users/list/removed`.

   You must be logged in as a user who has **Administrator** privileges. To set a user as **Administrator**, see “Configure User Manager” on page 1-20.

3 Select the user names that you want to remove from the Polyspace Access database and click **Confirm clean-up**. To select multiple users, press the **CTRL** key. Click the back button in your web browser to return to the Polyspace Access interface.

See Also

More About

- “Prepare Your Installation” on page 1-10
- “Configure and Start the Cluster Admin” on page 1-13
- “Configure Issue Tracker” on page 1-30
- “Configure Polyspace Access App Services” on page 1-34
- “Register Polyspace Desktop User Interface” on page 1-40
- “Start Polyspace Access and Upload Examples” on page 1-36

Configure Issue Tracker

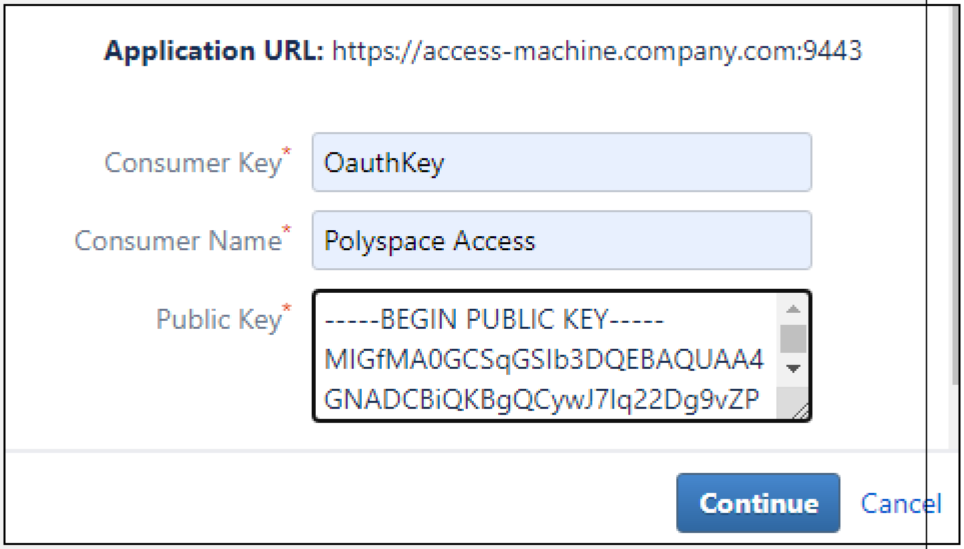
Configure the **Issue Tracker** if you want to enable the creation of tickets in your bug tracking tool (BTT) from the Polyspace Access interface. Polyspace Access supports integration with the Jira Software and Redmine BTT. If you do not want to integrate your BTT with Polyspace Access, select `none` in the **Provider** field.

On the **Cluster Dashboard**, click **Configure Apps** to go to the **Cluster Settings**. Whenever you change the settings, return to the **Cluster Dashboard** and click **Restart Apps** for the changes to take effect. To save partially filled settings, clear **Validate on Save**.

Note On Windows systems, all the file paths that you specify must point to local drives.

Configure Jira Software Bug Tracking Tool

Setting	Description
Provider	`Jira`
Jira deployment type	Specify whether your Jira instance is hosted on a local server or on a cloud service provider. This field is available only if you select Jira as a provider.
Jira URL	Specify the URL of the Jira instance for your organization, for example, `https://jira.mycompany.com`. If your Jira instance is configured by using HTTPS, see “Add BTT Instance Configured by Using HTTPS” on page 1-32. This field is available only if you select Jira as a provider.
Authentication method	Specify the method that the **Issue Tracker** uses to authenticate logins to Jira from Polyspace Access users. This field is available only if you select Jira as a provider.

Setting	Description
OAuth1 callback URL	If you select OAuth as an authentication method, your Jira administrator must first create an application link in Jira. The Jira administrator specifies an application URL (the URL for Polyspace Access) and generates an RSA public/private keypair. For more information, see step 1 on this page. The **OAuth1 callback URL** must match the application URL specified in Jira, for instance `https://access-machine.company.com:9443`.  **Application URL:** https://access-machine.company.com:9443 Consumer Key*: OauthKey Consumer Name*: Polyspace Access Public Key*: -----BEGIN PUBLIC KEY----- MIGfMA0GCSqGSIb3DQEBAQUAA4 GNADCBiQKBgQCywJ7lq22Dg9vZP This field is available only if you select Jira as a provider.
OAuth1 consumer key	Specify the consumer key value that your Jira administrator entered when configuring the application link in Jira, for instance `OauthKey`. This field is available only if you select Jira as a provider.
OAuth1 private key file	Specify the path to the private key file that your Jira administrator generated when configuring the application link in Jira, for instance `/local/polyspace_access/jira_privatekey.pem`. **Note** The host name that you specify when you create the JIRA private key must match the host name that you use in your Polyspace Access URL. If the host names do not match, for instance if you use `localhost` instead of the host name in your Polyspace Access URL, you might see a user authentication error when you attempt to create Jira tickets from Polyspace Access. This field is available only if you select Jira as a provider.

Limitations

- Polyspace Access does not support the creation of BTT tickets that have custom fields if these fields are required fields, except if the fields are:

  - All numeric values.
  - String only values.
  - Single select custom fields.
- After users log into Jira from Polyspace Access and start creating Jira tickets, the users remain logged into their Jira session until the session expires.
- In Jira Software version 8.4 and later, do not enable dark feature. See Enable Dark Feature in Jira.

Configure Redmine Bug Tracking Tool

Setting	Description
Provider	`Redmine`
Redmine URL	Specify the URL of the Redmine instance for your organization, for example, `https://redmine.mycompany.com`. If your Redmine instance is configured by using HTTPS, see “Add BTT Instance Configured by Using HTTPS” on page 1-32. This field is available only if you select Redmine as a provider.
Redmine API key	Specify the API access key of a Redmine administrator. To obtain the API key, log into your instance of Redmine as an administrator, click **My account** in the upper-right corner, then, in the right pane, click **Show** under **API access key**. The **Issue Tracker** does not validate the API key. Check periodically that the API key has not expired or become invalid. This field is available only if you select Redmine as a provider.

Limitations

- Polyspace Access does not support the creation of BTT tickets that have custom fields if these fields are required fields, except if the fields are all numeric values or string only values.
- To create a redmine ticket from Polyspace Access, the user name used to log into Polyspace Access must match the user name of a Redmine account.
- Redmine tickets that users create from Polyspace Access can be populated only with default field values. Some of the ticket field values that a user selects in Polyspace Access might not match the field values of the Redmine ticket.

Add BTT Instance Configured by Using HTTPS

If your BTT instance is configured by using HTTPS, add the BTT SSL certificate to the certificate trust store file that you specify in the **CA File** field of the **Nodes** settings. To view these settings, click **Configure Nodes** on the **Cluster Dashboard**. Depending on your trust store file, the BTT SSL certificate might already be included in the trust store.

The certificate trust store file typically corresponds to the file that you provide with `--ssl-ca-file` when you configure the **Cluster Admin** with HTTPS. See “Choose Between HTTP and HTTPS Configuration for Polyspace Access” on page 1-14.

For instance, on a Linux Debian distribution, to add BTT certificate `btts_cert.pem` to trust store file `trust_store.pem`, use this command:

```
cat trust_store.pem btts_cert.pem > combined_cert.pem
```

The command combines the content of the two files and outputs file `combined_cert.pem`. If you use a self-signed certificate to configure HTTPS, add the BTT certificate to the self-signed certificate.

To complete the configuration, enter the path to `combined_cert.pem` in the **CA File** field of the **Nodes** settings, save your changes, return to the dashboard, and restart the Apps.

If you did not configure the **Cluster Admin** by using HTTPS, specify the path to the BTT SSL certificate in the **CA File** field.

See Also

More About

- “Prepare Your Installation” on page 1-10
- “Configure and Start the Cluster Admin” on page 1-13
- “Configure User Manager” on page 1-20
- “Configure Polyspace Access App Services” on page 1-34
- “Register Polyspace Desktop User Interface” on page 1-40
- “Start Polyspace Access and Upload Examples” on page 1-36

Configure Polyspace Access App Services

Polyspace Access Database

Setting	Description
Data volume	Specify the full path to the folder where you store the database. By default, the database is stored under `appdata/polyspace-access` in the folder where you extracted the Polyspace Access installation image. Deleting the **Polyspace Access 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, manually delete the folder where you store the database.

Password	Specify a password to authenticate connections to the database. The different Polyspace Access services use this password when they communicate with the database. Use this password if you are prompted for one while performing a database backup on page 1-43 or clean up on page 1-47.
	You can specify a password only for a new database. To change the password of an existing database, use the PostgreSQL utilities instead. If you update the password of the database by using the PostgreSQL utilities, you must update the password in this field as well.
	The default password of the Polyspace Access database is "trust".
	To see the current database password, run this command.
	• **Windows**
	`docker inspect polyspace-access-db-main | FIND "POSTGRES_PASSWORD"`
	• **Linux**
	`docker inspect polyspace-access-db-main | grep POSTGRES_PASSWORD`

Polyspace Access ETL

Setting	Description
Storage directory	Specify the full path to folders with adequate write permissions. On Windows systems, the paths must point to local drives. See “Storage and Port Configuration” on page 1-5. By default, these folders are stored under `appdata/polyspace-access` in the folder where you extracted the Polyspace Access installation image. The **Upload directory** path must be the same for the **Polyspace Access Web Server** and **Polyspace Access ETL** services.
Invalid results directory	
Working directory	
Upload directory	

Polyspace Access Web Server

Setting	Description
Upload directory: **Temporary upload directory:**	Specify the full path to folders with adequate write permissions. On Windows systems, the paths must point to local drives. See “Storage and Port Configuration” on page 1-5. By default, these folders are stored under `appdata/polyspace-access` in the folder where you extracted the Polyspace Access installation image. The **Upload directory** path must be the same for the **Polyspace Access Web Server** and **Polyspace Access ETL** services.
License file:	Specify the full path to the `network.lic` license file that you created when you configured the Polyspace Access license. See “Configure Polyspace Access License” on page 2-2. On Windows systems, the paths must point to local drives.

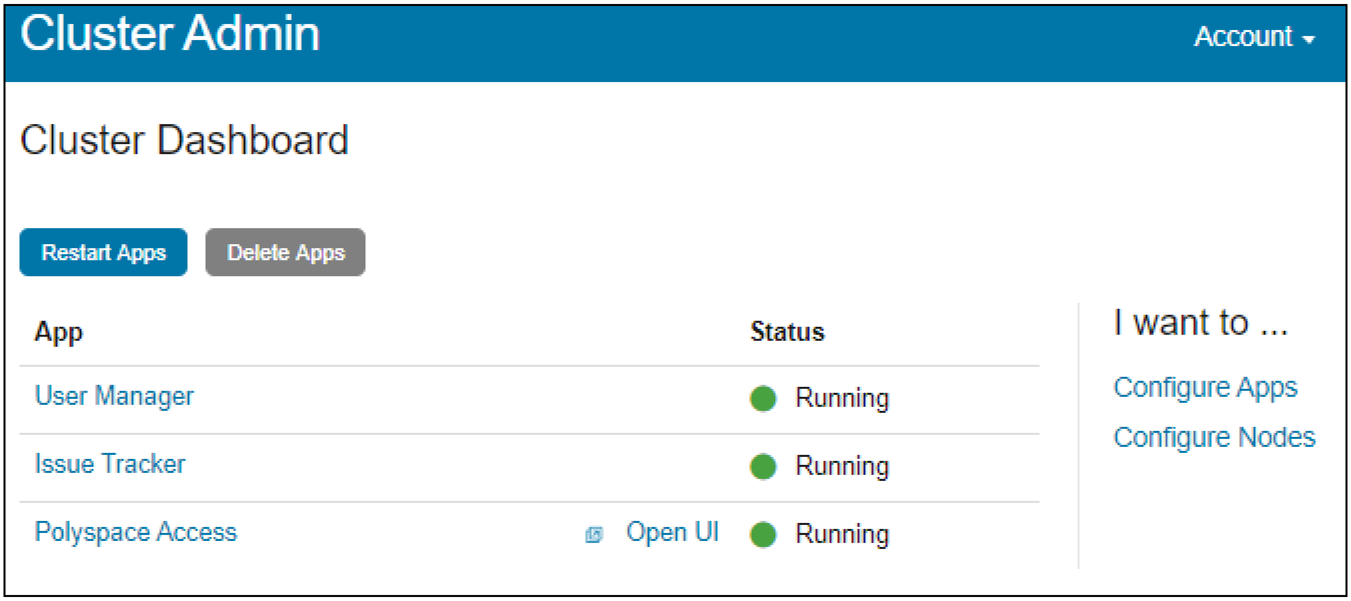
See Also

More About

- “Prepare Your Installation” on page 1-10
- “Configure and Start the Cluster Admin” on page 1-13
- “Configure User Manager” on page 1-20
- “Configure Issue Tracker” on page 1-30
- “Register Polyspace Desktop User Interface” on page 1-40
- “Start Polyspace Access and Upload Examples” on page 1-36

Start Polyspace Access and Upload Examples

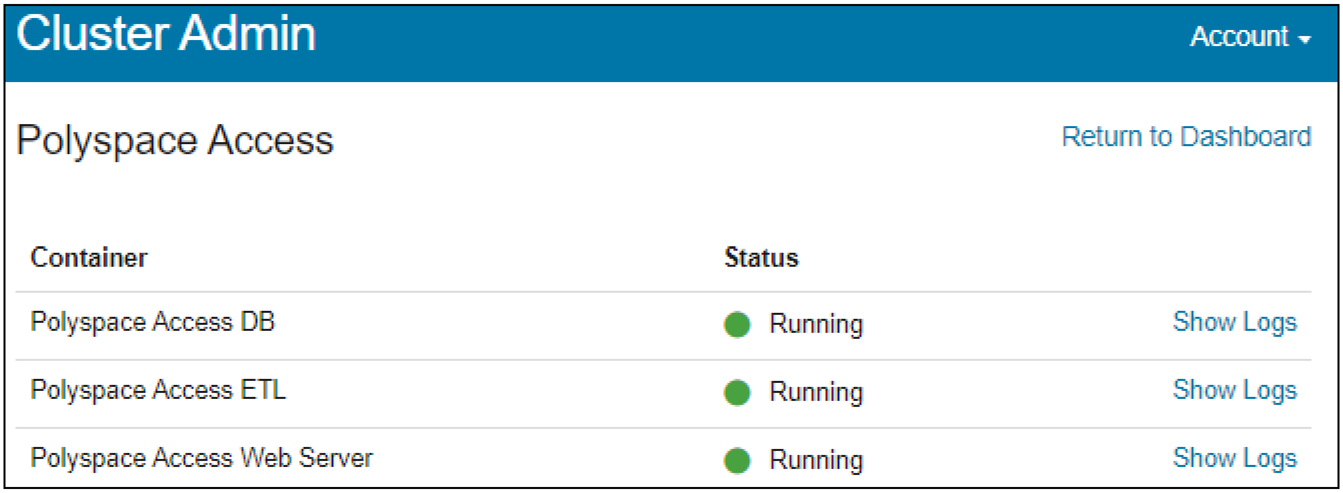
After you complete the configuration of the **User Manager**, **Issue Tracker**, and **Polyspace Access** apps, save your settings, return to the **Cluster Dashboard**, and click **Restart Apps**.



App		Status
User Manager		● Running
Issue Tracker		● Running
Polyspace Access	Open UI	● Running

The indicator turns green when an app starts. The **Polyspace Access Web Server** service might take a few moments to start.

If one or more of the apps start and stop after a short time, click the app in the **Cluster Dashboard** and then **Show Logs**.



Container	Status	
Polyspace Access DB	● Running	Show Logs
Polyspace Access ETL	● Running	Show Logs
Polyspace Access Web Server	● Running	Show Logs

Use the output log to try to identify the cause of the stopped service. If you need additional assistance, see “Contact Technical Support About Polyspace Access Issues”.

Once you complete the installation, close the **Cluster Admin** interface and stop the `admin-docker-agent` binary at the command line by pressing **Ctrl+C**. If you stop the binary before closing the interface, the app status is listed as Unknown state.

To start using Polyspace Access, upload results to the Polyspace Access database and open the web interface to view those results.

You cannot configure the Polyspace Access services to restart automatically after an unexpected system shutdown or a reboot. Always use the **Cluster Admin** interface to start the Polyspace Access services.

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
```

`polyspaceroot` is the path to your Polyspace installation. *`hostname`* is the fully qualified domain name (FQDN) of the machine that hosts Polyspace Access. *`port`* is the port number that you specified when starting the `admin-docker-agent` binary. For more information on uploading results from the command line, see “Upload Results at Command Line” on page 3-3.

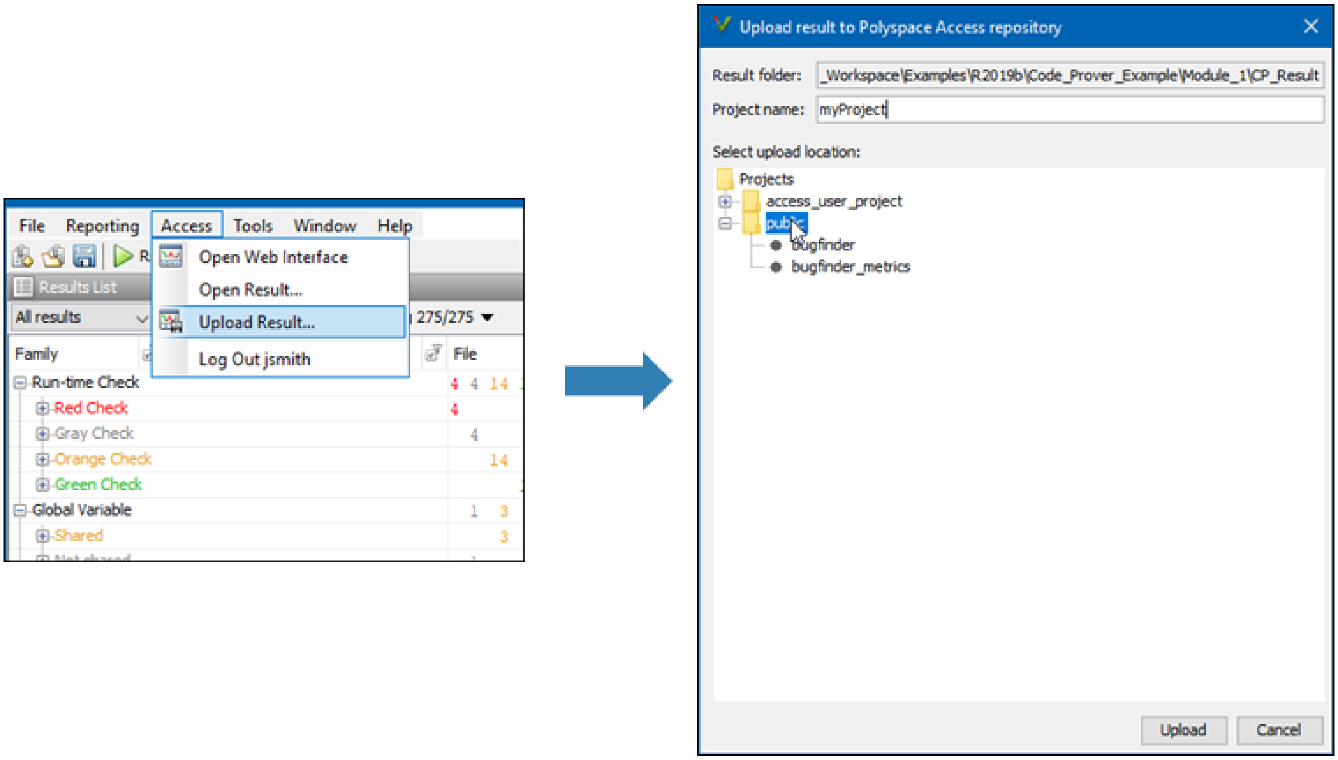
After each command, you are prompted to enter your user name and password. Enter the credentials that you use to log in to Polyspace Access.

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 1-40.

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

TO open the Polyspace Access interface, click **Open UI** in the **Admin Dashboard**. Copy the URL from the address bar, for instance `https://access-machine.company.com:9443/metrics/index.html` and share it with the Polyspace Access users. The URL allows users to open the Polyspace Access interface from any machine connected to the server that hosts Polyspace Access.

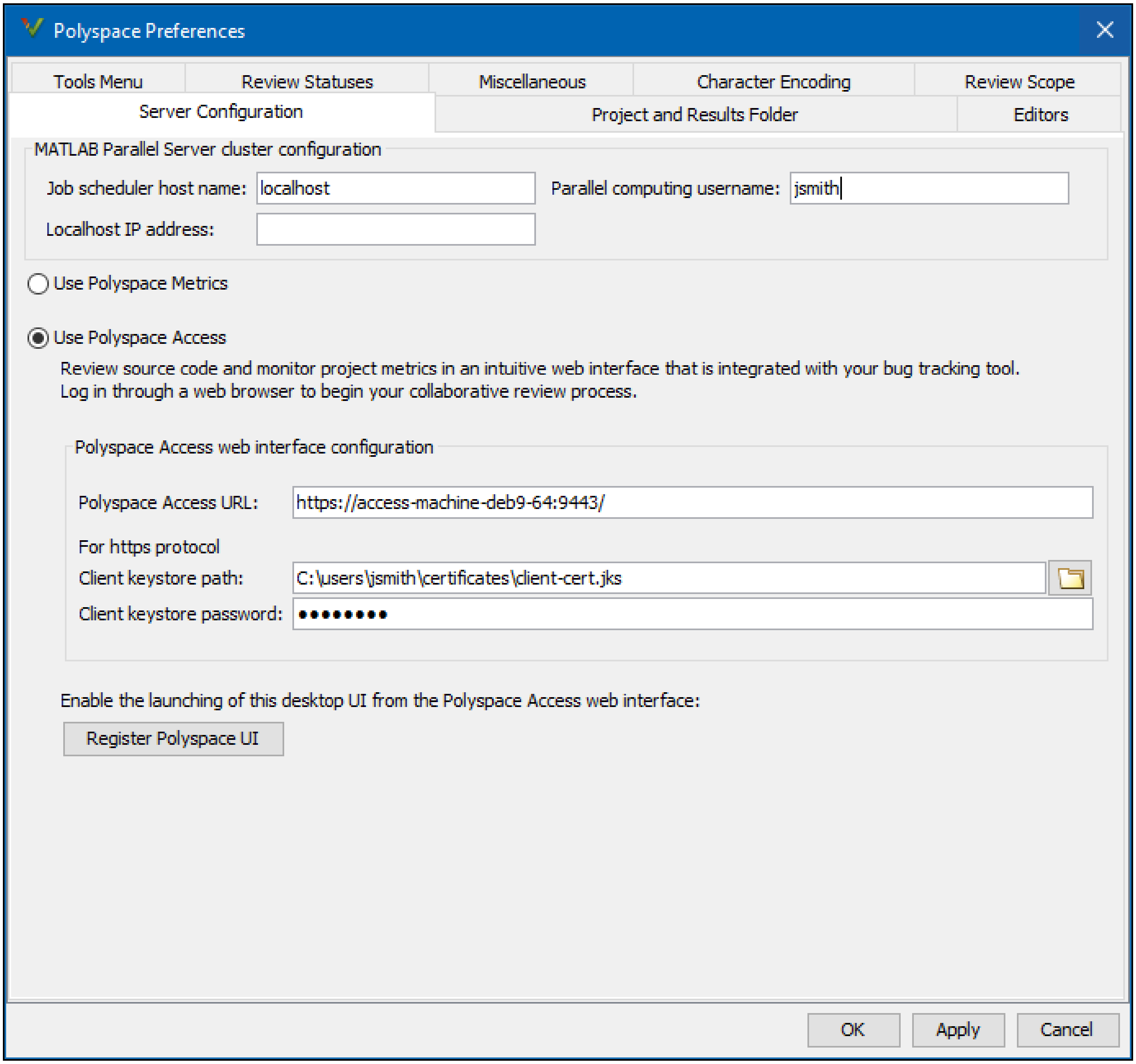
See Also

More About

- “Review Polyspace Bug Finder Results in Web Browser”

Register Polyspace Desktop User Interface

To enable interactions 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**: Specify the URL you use to log into the Polyspace Access interface as `http(s)://`*`hostName`*`:`*`port`*. If you do not know the URL, contact your Polyspace Access administrator.

- **Client keystore path**: path to the key store file where you imported the signed certificate use to configure Polyspace Access with HTTPS. See "Generate a Client Keystore" on page 1-41.

  If the Polyspace Access URL does not use HTTPS, leave this field blank.
- **Client keystore password**: The password associated with the key store file.

  If the Polyspace Access URL does not use HTTPS, 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 a Client Keystore

If you configure the Polyspace Access to use the HTTPS protocol, you must generate a Java Key Store (JKS) file to enable communications between Polyspace Access and the desktop interface, or the `polyspace-report-generator` and `polyspace-results-export` binaries. You import the signed certificate that you used to configure Polyspace Access with HTTPS to the JKS file you generate. See "Choose Between HTTP and HTTPS Configuration for Polyspace Access" on page 1-14.

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:

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

If you have a MathWorks product other than Polyspace Access installed on your machine, for example Polyspace Bug Finder Server, `keytool` is available in that product installation folder under `matlab/sys/java/jre/`*`$ARCH`*`/jre/bin`.

`$ARCH` is a platform specific folder such as `glnxa64`.

For example, if you used signed certificate `admin_cert.cer` to configure HTTPS for Polyspace Access, generate the corresponding JKS file by using this command:

```
keytool -import -trustcacerts -alias cert -file admin_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 1-37

Database Backup

To create a backup of your Polyspace Access database, use the `pg_dumpall` PostgreSQL utility. The utility creates a dump of your database. You can then restore the state of the database from when the dump was created. The `pg_dumpall` utility is available in the `polyspace-access-db-main` container of Polyspace Access.

Based on your database size and frequency of use, establish a policy for how often you create a backup. Users cannot interact with Polyspace Access while you perform a database backup or restore. Inform users before you start a backup or restore operation.

Create Database Backup

When you create a database backup, the `pg_dumpall` utility generates a list of SQL commands that you use reconstruct your database. The backup operation requires superuser privileges. The privileges are set through PostgreSQL and are separate from the user privileges on your system. For example, to generate a database dump and save it as `backup_db.sql`, open a terminal on the machine that hosts the **Polyspace Access Database** service and follow these steps. This workflow assumes that all the Polyspace Access services are running.

1. To ensure that your backup does not contain partial or corrupted data, stop the **Polyspace Access ETL** and **Polyspace Access Web Server** services before starting the backup operation. In the terminal, enter this command:

   ```
   docker stop polyspace-access-etl-main polyspace-access-web-server-main
   ```
2. Generate the database backup and save it to `backup_db.sql`.

   ```
   docker exec polyspace-access-db-main pg_dumpall -U postgres > backup_db.sql
   ```

   The `docker exec` command runs the `pg_dumpall` utility inside the `polyspace-access-db-main` container. The `-U` specifies superuser `postgres`. The output of `pg_dumpall` is then saved as `backup_db.sql`. Be aware that using `pg_dumpall` on large databases might generate files that exceed the maximum file size limit on some operating systems and can be time consuming.
3. Once you complete your backup, use this command to restart the **Polyspace Access ETL** and **Polyspace Access Web Server** services.

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

Restore Database from Backup

To recover your data from a database backup, use the `psql` utility. This utility is available in the `polyspace-access-db-main` container. The operation restores the data and the user permissions for the Polyspace Access projects. For example, you can restore your database from a backup stored in file `backup_db.sql`. You complete some steps in the **Cluster Admin** interface. Other steps require a terminal on the server that hosts the **Polyspace Access Database** service. On Linux, you might need superuser privileges to complete this operation.

1. Stop the **Polyspace Access ETL** and **Polyspace Access Web Server** services. In the terminal, enter this command:

   ```
   docker stop polyspace-access-etl-main polyspace-access-web-server-main
   ```
2. Delete the folder that stores the Polyspace Access database, and then restart the **Polyspace Access Database** service.

```
sudo rm -rf databaseFolderPath
docker restart polyspace-access-db-main
```

databaseFolderPath is the folder path that you specify in the **Data volume** field of the **Polyspace Access Database** service in the Admin **Cluster Settings**, for example:

`/local/Polyspace/R2020b/appdata/polyspace-access/db`

If you specify a volume name instead of a folder path in the **Data volume** field, for example `polyspace-data`, use these commands to stop the database service, delete the volume, create a new volume, and restart the database service:

```
docker stop polyspace-access-db-main
docker volume rm polyspace-data
docker volume create polyspace-data
docker restart polyspace-access-db-main
```

3. Restore the database from `backup_db.sql`. In the terminal, enter this command:

   ```
   docker exec -i polyspace-access-db-main psql -U postgres postgres <backup_db.sql
   ```

   If you stored your backup in a compressed file, decompress the file, and then pipe its content to the `docker exec` command. For instance, if you use `gzip`, use this command to restore the database from file `backup_db.gz`.

   ```
   gzip -cd backup_db.gz | docker exec -i polyspace-access-db-main psql -U postgres postgres
   ```

4. In the **Cluster Admin** interface, click **Restart Apps** to start all the services.

After the services start, open the Polyspace Access interface in your web browser to view the projects that were stored in the database when you created the backup.

Alternatively, you can rely on write ahead log (WAL) files to perform incremental backups and recoveries of your database. The WAL records all changes made to the database. The system 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 and Port Configuration” on page 1-5
- “Install and Manage Polyspace Access Web Server”

Database Backup for Polyspace Access Versions R2020a and Earlier

Note This topic applies to Polyspace Access versions R2020a and earlier. For versions R2020b and later, see “Database Backup” on page 1-43.

To create a backup of your Polyspace Access database, use the `pg_dumpall` PostgreSQL utility. The utility creates a dump of your database. You can then restore the state of the database from when the dump was created. The `pg_dumpall` utility is available in the `polyspace-db` container of Polyspace Access.

Based on your database size and frequency of use, establish a policy for how often you create a backup. Users cannot interact with Polyspace Access while you perform a database backup or restore. Before you start a backup or restore operation, inform your users.

Create Database Backup

When you create a database backup, the `pg_dumpall` utility generates a list of SQL commands that you use to reconstruct your database. The backup operation requires superuser privileges. The privileges are set through PostgreSQL and are separate from the user privileges on your system. For example, to generate a database dump and save it as `backup_db.sql`, open a terminal on the machine that hosts the **Database** service and follow these steps.

1 To ensure that your backup does not contain partial or corrupted data, stop the **ETL** and **Web Server** services before starting the backup operation. In the terminal, enter this command:

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

Tip Stopping the Polyspace Access services by using the Cluster Operator (COP) interface on Windows Server 2019 might result in a slow response time. Instead, use these commands to stop the services:

```
docker exec -it polyspace-etl kill 1
docker exec -it polyspace-web-server kill 1
```

2 Generate the database backup and save it to `backup_db.sql`.

```
docker exec polyspace-db pg_dumpall -U postgres > backup_db.sql
```

The `docker exec` command runs the `pg_dumpall` utility inside the `polyspace-db` container. The `-U` specifies superuser `postgres`. The output of `pg_dumpall` is then saved as `backup_db.sql`. Be aware that using `pg_dumpall` on large databases might generate files that exceed the maximum file size limit on some operating systems and can be time consuming.

3 To restart the **ETL** and **Web Server** services.

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

Restore Database from Backup

To recover your data from a database backup, use the `psql` utility. This utility is available in the `polyspace-db` container. The operation restores the data and the user permissions for the Polyspace

Access projects. For example, you can restore your database from a backup stored in file `backup_db.sql`. You complete some steps in the COP interface. Other steps require a terminal on the server that hosts the **Database** service.

1. Create a database. In the COP interface, delete all the services, then click **Settings** in the left pane. Under the **Database** settings, click **create volume** next to the **Data volume** field to create a volume, then select this volume from the **Data volume** drop-down list. If you do not see a **create volume** link, specify a new folder path in the **Data volume** field. If the folder path does not exist, the COP creates it. Save your changes.
2. Return to the **Services** tab, click **PROVISION**, then start only the **Database** service.
3. Restore the database from `backup_db.sql`. In the terminal, enter this command:

   ```
   docker exec -i polyspace-db psql -U postgres postgres <backup_db.sql
   ```

   If you stored your backup in a compressed file, decompress the file, and then pipe its content to the `docker exec` command. For instance, if you use `gzip`, to restore the database from file `backup_db.gz`, enter:

   ```
   gzip -cd backup_db.gz | docker exec -i polyspace-db psql -U postgres postgres
   ```
4. In the COP interface, click **START ALL** to start all the services.

After the services start, open the Polyspace Access interface in your web browser to view the projects that were stored in the database up to when you created the backup.

Alternatively, you can rely on write ahead log (WAL) files to perform incremental backups and recoveries of your database. The WAL records all changes made to the database. The system 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 and Port Configuration” on page 1-5
- “Install and Manage Polyspace Access Web Server”

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 your database before you perform a clean up operation. See “Create Database Backup” on page 1-43.

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 **Polyspace Access Web Server** and **Polyspace Access ETL** services. To stop the services, from a terminal on the server hosting these services, use this command and entering:

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

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

```
docker exec polyspace-access-db-main vacuumdb -U postgres prs_data
```

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-access-db-main vacuumdb -U postgres --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-access-db-main vacuumdb -U postgres --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 **Polyspace Access Web Server** and **Polyspace Access ETL** services. Use this command:

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

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

Delete Outdated Projects

When users delete projects from the **Project Explorer** of the Polyspace Access web interface, the projects move to the **ProjectsWaitingForDeletion** folder. The projects, including all the runs that you uploaded to the projects, remain in the database until you explicitly delete them.

The **ProjectsWaitingForDeletion** folder is visible only to Polyspace Access users who have the role of **Administrator**. Even users who have the **Administrator** role cannot delete projects *from the Polyspace Access interface*.

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

To remove old project runs or entire projects from your database, write a command in a text file that you save as a `.pscauto` file. Run the command by copying the `.pscauto` file to the **Storage directory** of the **Polyspace Access ETL** service. Only a user who has 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 with one of these command parameters.
  - `clean_project projectPath DATE YYYY-MM-DD`

    The command deletes project runs that were uploaded before `YYYY-MM-DD`.
  - `clean_project projectPath MAXRUNS NNN`

    `NNN` is an integer. The command keeps the `NNN` most recent runs. To delete all the project runs, use `MAXRUNS 0`.
  - `clean_project projectPath AGE DDD`

    `DDD` is the number of days. To remove recently uploaded results, use this option. The command deletes project runs that are older than `DDD` days.
- 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 `polyspace-access` binary with the `-list-project` flag. For more information, see `polyspace-access -h -list-project`.

If the path contains whitespace characters, enclose the project path in double quotes. If you use `echo` to write the commands to a file, you must also use a "\" character to escape whitespace characters in the project path.

For example, to perform a one-time cleanup of project `public/Bug_Finder_Example (Bug Finder)` and remove all results uploaded before a specific date:

1. Open a text editor, paste this command, then save the file as a `.pscauto` file, for instance `cleanup.pscauto`.

   ```
   clean_project "public/Bug_Finder_Example (Bug Finder)" DATE 2019-09-01
   ```
2. Copy the file to the **Storage directory** of the **Polyspace Access ETL** service:

```
    cp cleanup.pscauto /local/Access_install_dir/polyspace/storage
```

All analysis runs uploaded to project `public/Bug_Finder_Example (Bug Finder)` prior to September 1, 2019 are deleted from the database.

You can also perform an automatic cleanup on a specific project every time you upload a run to that project. To keep only the 10 most recent runs every time you upload a result to `public/Bug_Finder_Example (Bug Finder)`, save these commands to your `.pscauto` file.

```
assign_to_project "public/Bug_Finder_Example (Bug Finder)" AFTER_STATISTICS myScript
clean_project "public/Bug_Finder_Example (Bug Finder)" MAXRUNS 10
```

The commands that you enter after the `assign_to_project` line are stored internally in a script `myScript` that is assigned to the project `public/Bug_Finder_Example (Bug Finder)`. Use distinct names for the internal script that you assign to different projects. You specify the internal script name with the last parameter of the `assign_to_project` command. After you copy the file to the **Storage directory** of the **Polyspace Access ETL** service, the automatic cleanup starts.

To turn off the automatic cleanup, save this command to a `.pscauto` file and copy that file to the **Storage directory**:

```
unassign_project "public/Bug_Finder_Example (Bug Finder)" myScript
```

You must provide the name of the internal script that you assigned to the project by using the `assign_to_project` command. Make sure that you keep a record of the internal script names and the projects to which they are assigned.

Caution 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 and Port Configuration” on page 1-5
- “Install and Manage Polyspace Access Web Server”

Update or Uninstall Polyspace Access

You can update or uninstall Polyspace Access. Before you begin, inform Polyspace Access users of the upcoming update or uninstallation.

Update Polyspace Access

To update Polyspace Access, download and unzip the new installation image. See Download instructions on page 1-13.

When you perform an update, you can reuse your current settings and database or you can import a snapshot of the current database to a new database. If you do not reuse the settings and database, follow the standard installation instruction. See “Install Polyspace Access”.

Note If you update from Polyspace Access R2020a or earlier to Polyspace Access R2020b or later and you use the **embedded LDAP** in the older version, use the **User Manager** internal directory in the newer version. Enter the user names and passwords from the LDIF file into the **User Manager** interface. See “Authenticate Users from Internal Directory” on page 1-24.

Create Polyspace Access Snapshot

To reuse your database and settings, create a snapshot of your current instance of Polyspace Access. In your snapshot, include a backup of the:

- Polyspace Access database. See “Create Database Backup” on page 1-43. To back up a database for Polyspace Access version R2020a or earlier, see “Database Backup for Polyspace Access Versions R2020a and Earlier” on page 1-45.
- Current settings. Make a copy of the `settings.json` file. This file is typically in the same folder as the `admin-docker-agent` binary.
- User profiles that are not stored in your company LDAP.

  To back up user profiles that are stored in the **User Manager** internal directory (Polyspace Access R2020b and later), save a copy of the internal directory folder. You specify the path of this folder in the **Internal directory database volume** field of the Admin **Cluster Settings**, for example, `/local/Polyspace/R2020b/appdata/usermanager/db`.

  To back up user profiles that are stored in the **embedded LDAP** (Polyspace Access R2020a and earlier), save a copy of the LDIF file. You specify the path of this file in the **LDIF file** field of the Cluster Operator settings.

Uninstall Polyspace Access and Reuse Settings and Database with New Installation

1. To remove your current instance, see “Uninstall Polyspace Access” on page 1-51 but omit step 2.a.
2. In the new installation folder, start the `admin-docker-agent` binary and use the `--data-dir` flag to point to the folder that contains the backup `settings.json` file. For example, if `settings.json` is in folder `/local/Polyspace/R2020a_backup`, enter:

   ```
   ./admin-docker-agent --data-dir /local/Polyspace/R2020a_backup
   ```
3. Open the **Cluster Admin** web interface and click **Restart Apps**.

If you install your new Polyspace Access instance on a different machine, you cannot reuse your SSL certificates. See “Choose Between HTTP and HTTPS Configuration for Polyspace Access” on page 1-14.

Keep Current Polyspace Access and Reuse Settings and Database with New Installation

1. In the new installation folder, start the `admin-docker-agent` binary and use the `--data-dir` flag to point to the folder that contains the backup `settings.json` file.
2. Open the **Cluster Admin** web interface, click **Configure Apps**, and then enter a new folder path in the **Data volume** field to create a new database. You cannot use the same database in two different instances of Polyspace Access at the same time.
3. Start the **Polyspace Access Database** service and import the database snapshot into the new database. For example, if you stored the database snapshot in `db_backup.sql`, at the command line, enter:

   ```
   docker restart polyspace-access-db-main
   docker exec -i polyspace-access-db-main psql -U postgres postgres <backup_db.sql
   ```

   Ensure that the **Polyspace Access Web Server** and **Polyspace Access ETL** services are stopped before you import the database.
4. Return to the **Cluster Dashboard** and click **Restart Apps**.

If you install your new Polyspace Access instance on a different machine, you cannot reuse your SSL certificates. See “Choose Between HTTP and HTTPS Configuration for Polyspace Access” on page 1-14.

Reuse Database with New Polyspace Access Installation

Start the `admin-docker-agent` in the new installation folder, and then, for instructions, see “Restore Database from Backup” on page 1-43.

Check License Manager Version

To avoid any potential issues with license file operation, make sure that you run the latest license manager software version. To view the latest license manager software version available, see the FlexNet Version on this page.

To check your current license manager software version, at the command line, depending on your operating system, enter the commands listed in this table.

Windows	`cd` *`LM_Folder`*`\etc\win64`
	`lmgrd.exe -v`

Linux	`cd` *`LM_Folder`*`/etc/glnx64`
	`./lmgrd -v`

`LM_Folder` is the folder where you installed the license manager. See also “Update Network License Manager Software”.

Uninstall Polyspace Access

1. Verify that the **Cluster Admin** agent is running. Use the command:

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

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

2. Open the **Cluster Admin** web interface and click **Delete Apps**. After you delete an app, the status indicator turns gray and you see the text **Not installed** next to the indicator.

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

   a. To delete a data volume and its content, manually delete the folder where you store the database.

      ```
      sudo rm -rf databaseFolderPath
      ```

      `databaseFolderPath` is the folder path that you specify in the **Data volume** field of the **Polyspace Access Database** service in the Admin **Cluster Settings**, for example, `/local/Polyspace/R2020b/appdata/polyspace-access/db`.

      If you specify a volume name instead of a folder path in the **Data volume** field, for instance `polyspace-data`, use this command to delete the volume:

      ```
      docker volume rm polyspace-data
      ```

3. Stop the `admin-docker-agent` binary from the command line window by pressing **CTRL+C**, and then stop the remaining services:

   ```
   docker stop gateway \
   polyspace-access \
   issuetracker \
   usermanager
   ```

   To use this command in Windows PowerShell, replace the backslash "\" characters with a backtick " ` ".

4. Delete the Polyspace Access installation folder.

See Also

`admin-docker-agent`

More About

- “Configure and Start the Cluster Admin” on page 1-13
- “Configure User Manager” on page 1-20
- “Configure Issue Tracker” on page 1-30
- “Configure Polyspace Access App Services” on page 1-34
- “Database Backup” on page 1-43

2

Manage Polyspace Access License

Configure Polyspace Access License

The Polyspace Access license is a Network Named User (NNU) license that requires a license manager to manage license checkouts and an options file to specify the Named Users to whom you grant right-to-use privileges.

Prerequisites

- Install the license manager. See “Install License Manager” on page 2-5.

Follow these steps to configure the Polyspace Access license. To add or remove users, see “Manage Named Users for Polyspace Access” on page 2-6.

Note Enterprise license customers. Contact your license administrator to configure the Polyspace Access Enterprise license.

1 Copy this template file to a text editor and save it as `MLM.opt` on the machine where you installed the license manager.

Template

```
# Options file used by MATLAB vendor daemon (MLM).
# This file contains the INCLUDE lines necessary
# for a User Based license.
# If you change the user names listed here,
# you must restart the license manager
# for the changes to take effect.
# The frequency of user name changes may be limited
# by your software license agreement.
# If you have combined multiple license files into a
# single license file, you will need to change
# the INCLUDE lines to specify a particular INCREMENT
# line. You can do this using the "featurename Key=value"
# syntax in the INCLUDE line.
# See the FLEXnet Licensing End Users Guide for
# details on how to use options files.

# Make user names and host names case insensitive when
# listed in a GROUP or HOST_GROUP.  This is not
# required but it is here to prevent some common errors.
GROUPCASEINSENSITIVE ON

# Define a group of users
GROUP ACCESS_CP_users user1 user2 user3


# Grant right-to-use privileges to individual users
INCLUDE Polyspace_BF_Access USER user1
INCLUDE Polyspace_BF_Access USER user2
INCLUDE Polyspace_CP_Access USER admin
# Grant right-to-use privileges to group of users
INCLUDE Polyspace_CP_Access GROUP ACCESS_CP_users
```

- Use this file to identify the users to whom you grant right-to-use privileges for Polyspace Bug Finder Access (Polyspace_BF_Access) and Polyspace Code Prover Access (Polyspace_CP_Access).

  A user with right-to-use privileges for Polyspace Bug Finder Access has right-to-use privileges for Polyspace as You Code.
- For each user, enter the user name that the user specifies to log into Polyspace Access. The user names correspond to the user name entries in your company LDAP server or the **User Manager** internal directory. See “Configure User Manager” on page 1-20.
- For Polyspace as You Code users, the user name must also match the user name used to log into the machine where the user installs and runs Polyspace as You Code.

2 Copy your Polyspace Access license to the machine where you installed the license manager and save it as `license.dat`. Then, open the file in a text editor and insert these lines at the top of the file.

```
SERVER lmHostname HostID 27000
DAEMON MLM pathTo_MLM_bin options=pathTo_MLM.opt
```

- *`lmHostname`* is the fully qualified domain name (FQDN) of the machine where you installed the license manager. To get the FQDN, open a command-prompt window and enter:

Windows	`net config workstation | findstr /C:"Full Computer name"`
Linux	`hostname --fqdn`

- *`HostID`* is the MAC address that you provided to activate the Polyspace Access license. This MAC address must match the host ID listed for Polyspace Access in the license file. *`HostID`* must also match a MAC address on the machine where you run the license manager.
- *`pathTo_MLM_bin`* is the path to the MLM binary. You can find this binary in *`LM_Folder`*`\etc\win64` (Windows) or *`LM_Folder`*`/etc/glnx64` (Linux), where *`LM_Folder`* is the folder where you installed the license manager.
- *`pathTo_MLM.opt`* is the path to the options file that you created in step 1.
- By default, the license manager starts on port 27000. To use a different port, specify a different port number at the end of the `SERVER` line.

If you used the MATLAB® installer to install the license manager, the file `license.dat` already exists in the folder *`matlabroot`*`/etc` and the file already includes the `SERVER` and `DAEMON` lines. You may have to add the `options=`*`pathTo_MLM.opt`* instruction on the `DAEMON` line of `license.dat`. *`matlabroot`* is your MATLAB installation folder. Append the content of your Polyspace Access license to the `license.dat` file and go to step 3.

3 Copy the `SERVER` line from the `license.dat` file and paste it in a new file in a text editor. Add `USE_SERVER` below the `SERVER` line.

```
SERVER lmHostname HostID 27000
USE_SERVER
```

Save this file as `network.lic` in a location that is accessible from the machine where you installed Polyspace Access or Polyspace as You Code. This location can be on a different machine from the one where you installed the license manager.

- For the Polyspace Access web server, specify the path to this file for the **License file:** field of the **Polyspace Access Web Server** settings in the Cluster Admin web interface. See “Configure Polyspace Access App Services” on page 1-34.

Make sure that the docker engine can resolve the host name `lmHostname`. In a command-prompt window, enter:

```
docker run --rm -it alpine ping lmHostname
```

If the docker engine cannot resolve this host name, in `network.lic`, replace *`lmHostname`* with the IP address of the machine where you installed the license manager.

- For Polyspace as You Code, specify the path to `network.lic` when the installer prompts you to provide a license file path. See “Install Polyspace as You Code Using Installer” on page 4-2.

4. In a command-prompt window, navigate to the folder where you installed the license manager, and then start the license manager.

Windows	`cd LM_Folder\etc\win64`
	`lmgrd.exe -c pathToLicense -l lm_log.log`
	On Windows, you can also use lmtool.exe and go to the **Start/Stop/Reread** tab to start the license manager.

Linux	`cd LM_Folder/etc/glnx64`
	`./lmgrd -c pathToLicense -l lm_log.log`

`LM_Folder` is the folder where you installed the license manager.

`pathToLicense` is the path to the `license.dat` file that you saved on the machine where you installed the license manager. The command starts the license manager and outputs a log file `lm_log.log`. Refer to this log file for debugging purposes.

Note The license file path listed in the log and error messages of the license manager might not correspond to *`pathToLicense`*. The **Polyspace Access Web Server** service remaps *`pathToLicense`* to an internal path inside the docker container.

5. After you start the license manager, ensure that the license manager is configured to automatically start at boot time.

Windows	Use lmtool.exe and go to the **Config Services** tab, then check that **Start Server at Power Up** and **Use Services** are selected.
Linux	Refer to the documentation for your Linux distribution to configure the license manager to start automatically at boot time, for instance by adding a script to the `/etc/inti.d` folder. Configure the license manager to start at the end of the boot sequence.

Each licensed Polyspace Access user can log in to up to five concurrent sessions.

Polyspace Access ignores any license timeout value you set in the license options file (`MLM.opt`) by using the syntax `TIMEOUT feature seconds`. To set the licensing timeout, use the **Authentication token expiration** setting of the **User Manager**. See “Configure User Manager” on page 1-20.

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.

Install License Manager

The license manager is shipped with the Polyspace Access software. The license manager binaries and utilities are located in `accessRoot/lm`. `accessROOT` is the folder where you extracted your Polyspace Access installation image.

To run the license manager on a separate server from the server where you run Polyspace Access, copy the folder that corresponds to your platform from `accessRoot/lm`, for instance `accessRoot/lm/glnxa64`, to that server. The license manager folder includes these binaries:

- `lmgrd`: Core license manager binary. Use this binary to start the license manager from the command line. For a list of useful commands, enter `lmgrd -h`.
- `mlm`: The MATLAB vendor daemon.
- `lmutil`: a suite of tools for administering the license manager at the command line. For a list of useful commands, enter `lmutil -h`.
- `lmtools.exe` (Windows only): Graphical user interface for administering the license manager.
- For Linux systems, the license manager folder also includes command-line utilities. See “Using Command-Line Utilities”.

To avoid any potential issues with license file operation, make sure that you run the latest license manager software version. To view the latest license manager software version available, see the FlexNet Version on this page.

To check your current license manager software version, at the command line, depending on your operating system, enter the commands listed in this table.

Windows	`cd` *`LM_Folder`*`\etc\win64`
	`lmgrd.exe -v`

Linux	`cd` *`LM_Folder`*`/etc/glnx64`
	`./lmgrd -v`

`LM_Folder` is the folder where you installed the license manager. See also “Update Network License Manager Software”.

See Also

Related Examples

- “Install Polyspace as You Code Using Installer” on page 4-2
- “Configure Polyspace Access App Services” on page 1-34

Manage Named Users for Polyspace Access

The Polyspace Access license is a Network Named User (NNU) license. You specify the users to whom you grant right-to-use privileges in an text file typically named `MLM.opt`. See "Configure Polyspace Access License" on page 2-2. To add or remove users from the list of Named Users associated with the Polyspace Access license, edit the list of Named Users in the `GROUP` and `INCLUDE` entries of the `MLM.opt` file.

```
 Define a group of users
GROUP ACCESS_CP_users user1 user2 user3


# Grant right-to-use privileges to individual users
INCLUDE Polyspace_BF_Access USER user1
INCLUDE Polyspace_BF_Access USER user2
INCLUDE Polyspace_CP_Access USER admin
# Grant right-to-use privileges to group of users
INCLUDE Polyspace_CP_Access GROUP ACCESS_CP_users
```

For each user, enter the user name that the user specifies to log into Polyspace Access. The user names correspond to the user name entries in your company LDAP server or the **User Manager** internal directory. See "Configure User Manager" on page 1-20.

For Polyspace as You Code users, the user name must also match the user name used to log into the machine where the user installs and runs Polyspace as You Code.

Then, in a command-prompt window, navigate to the folder where you installed the license manager. Stop and restart the license manager.

Note These instructions do not apply to Enterprise license customers.

Windows	`cd` *`LM_Folder`*`\etc\win64`
	`lmutil lmdown`
	`lmgrd.exe -c` *`pathToLicense`* `-l lm_log.log`
	On Windows, you can also use lmtool.exe and go to the **Start/Stop/Reread** tab to stop and restart the license manager.

Linux	`cd` *`LM_Folder`*`/etc/glnx64`
	`./lmutil lmdown`
	`./lmgrd -c` *`pathToLicense`* `-l lm_log.log`

`LM_Folder` is the folder where you installed the license manager.

`pathToLicense` is the path to the Polyspace Access license file.

Changes to the options file might be delayed by 15 minutes before they take effect.

To view the status of the license manager and see how many licenses are currently checked out, use the `lmstat` command.

Windows	`cd LM_Folder\etc\win64`
	`lmutil.exe lmstat -c pathToLicense -a`
	On Windows, you can also use lmtool.exe and go to the **Server Status** tab to stop and restart the license manager.

Linux	`cd LM_Folder/etc/glnx64`
	`./lmutil lmdown`
	`./lmutil lmstat -c pathToLicense -a`

`LM_Folder` is the folder where you installed the license manager. *`pathToLicense`* is the path to the Polyspace Access license file.

See Also

Related Examples

- “Install Polyspace as You Code Using Installer” on page 4-2
- “Configure Polyspace Access App Services” on page 1-34

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 user interface to view statistics about the quality of your code and to triage and review individual results.

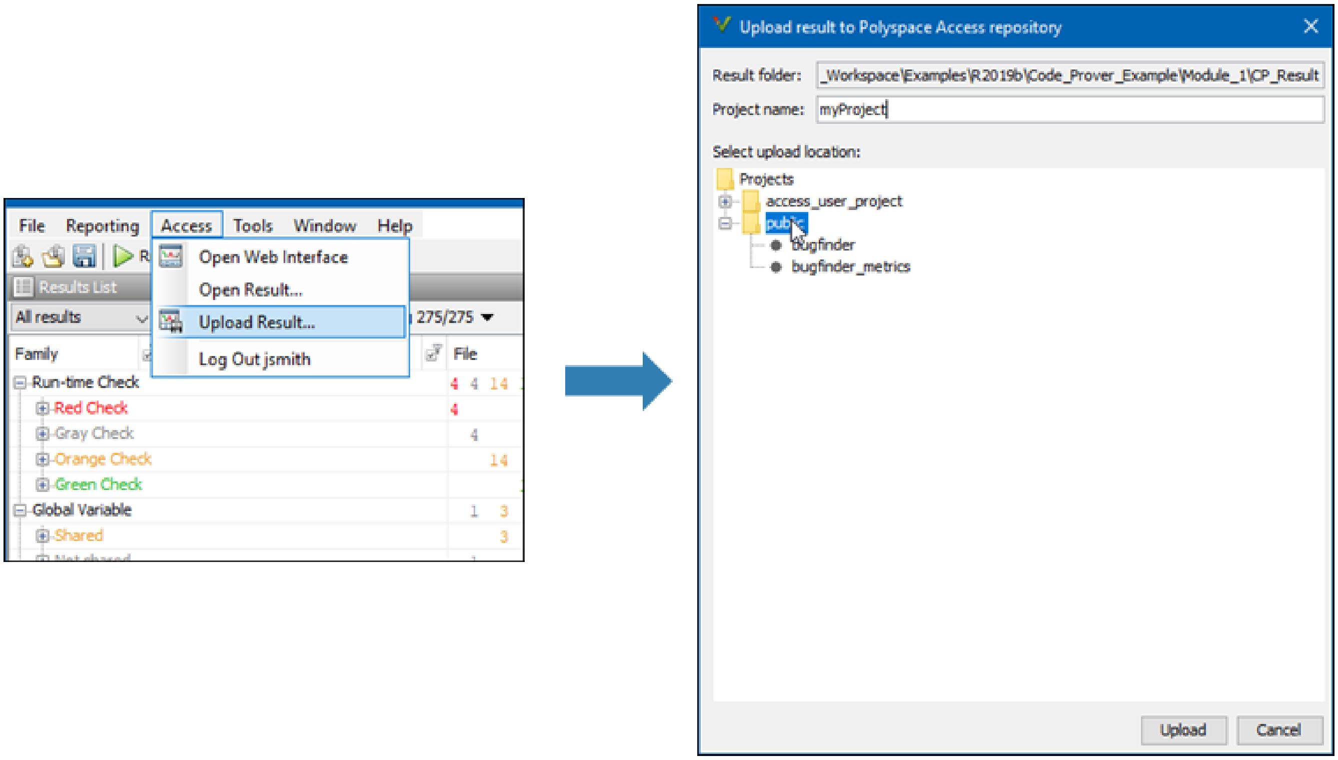
Polyspace assigns a unique run ID to each analysis run that you upload and increments the run ID with each upload to any project. If you use an automation tool such as Jenkins to upload results, the Polyspace Access run ID is not related to the tool job ID.

Note You can upload up to 2GB of results per upload to Polyspace Access.

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 1-40.

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. To create custom statuses, see "Open Polyspace Access Results in a Desktop Interface" on page 3-5.

Upload Results at Command Line

You can upload results 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. See `polyspace-access`.

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. To upload results to a different folder, use the `-parent-folder` option. *`hostName`* and *`port`* correspond to the host name and port number you specify in the URL of the Polyspace Access interface, for example `https://`*`hostName`*`:`*`port`*`/metrics/index.html`. If you are unsure about which host name and port number to use, contact your Polyspace Access administrator. Depending on your configuration, you might also have to specify the `-protocol` option in the command. See "Configure and Start the Cluster Admin" on page 1-13.

Results Upload Compatibility and Permissions

Results Compatibility

You cannot upload analysis results to a Polyspace Access version that is older than the version of the Polyspace product that generated the results. For instance, you cannot upload results generated with a Polyspace product version R2019b to a Polyspace Access version R2019a.

If you upload results generated with a Polyspace product version R2018b or earlier, you cannot view these results in the Polyspace Access **REVIEW** perspective. To review R2018b or earlier results that you uploaded to Polyspace Access, see "Open Polyspace Access Results in a Desktop Interface" on page 3-5.

User Permissions for Uploaded Results

You are the project **Owner** for all the results that you upload. The project **Owner** or an **Administrator** must add other users as **Contributor** to grant them permission to see those results, unless you upload the results to a folder that other users already have permission to see.

Results that you upload to the **public** folder are visible to all Polyspace Access users. For more information, see "Manage Project Permissions" on page 3-18.

See Also

`polyspace-access`

More About

- “Register Polyspace Desktop User Interface” on page 1-40
- “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 1-40.

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. To assign a custom **Status**, in the desktop interface:

- Go to **Tools** > **Preferences** and select the **Review Statuses** tab to create a custom status.
- In the **Result Details** pane, assign the status you created from the **Status** drop-down.

In the Polyspace Access interface, a custom status you assign in a project is not available in other projects.

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`.

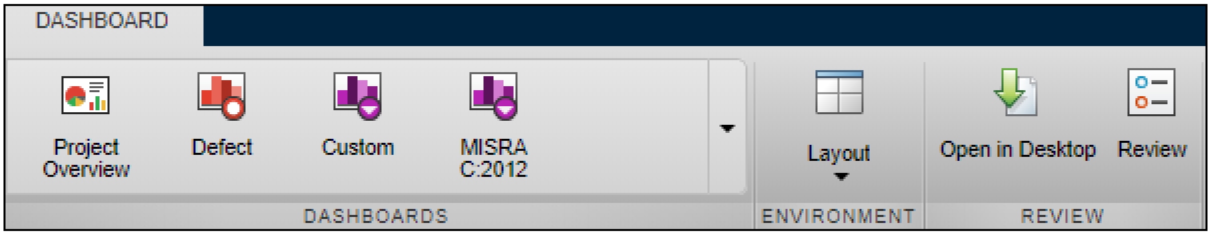
`hostName` and *`port`* correspond to the host name and port number you specify in the URL of the Polyspace Access interface, for example `https://`*`hostName`*`:`*`port`*`/metrics/index.html`. If you are unsure about which host name and port number to use, contact your Polyspace Access administrator. Depending on your configuration, you might also have to specify the `-protocol` option in the command. See “Configure and Start the Cluster Admin” on page 1-13.

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

Dashboard

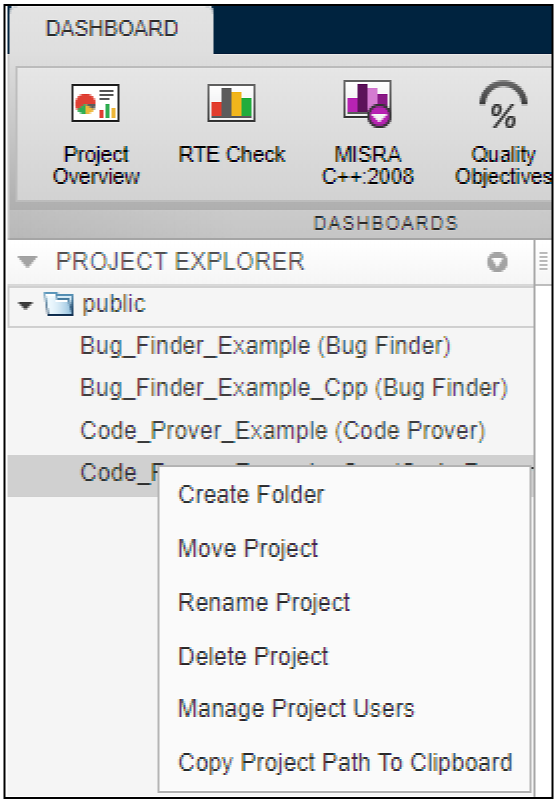
The **DASHBOARD** perspective provides an overview of the analysis results in graphical format, with clickable fields that enable you drill down into your findings by file, project, or category.

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.

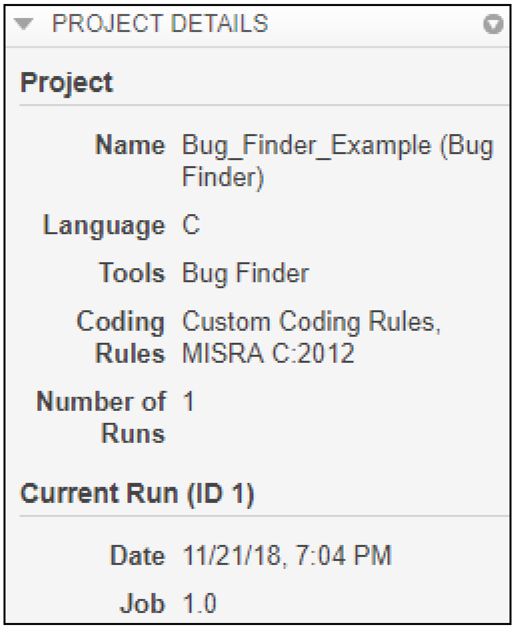
Project Explorer pane



- 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

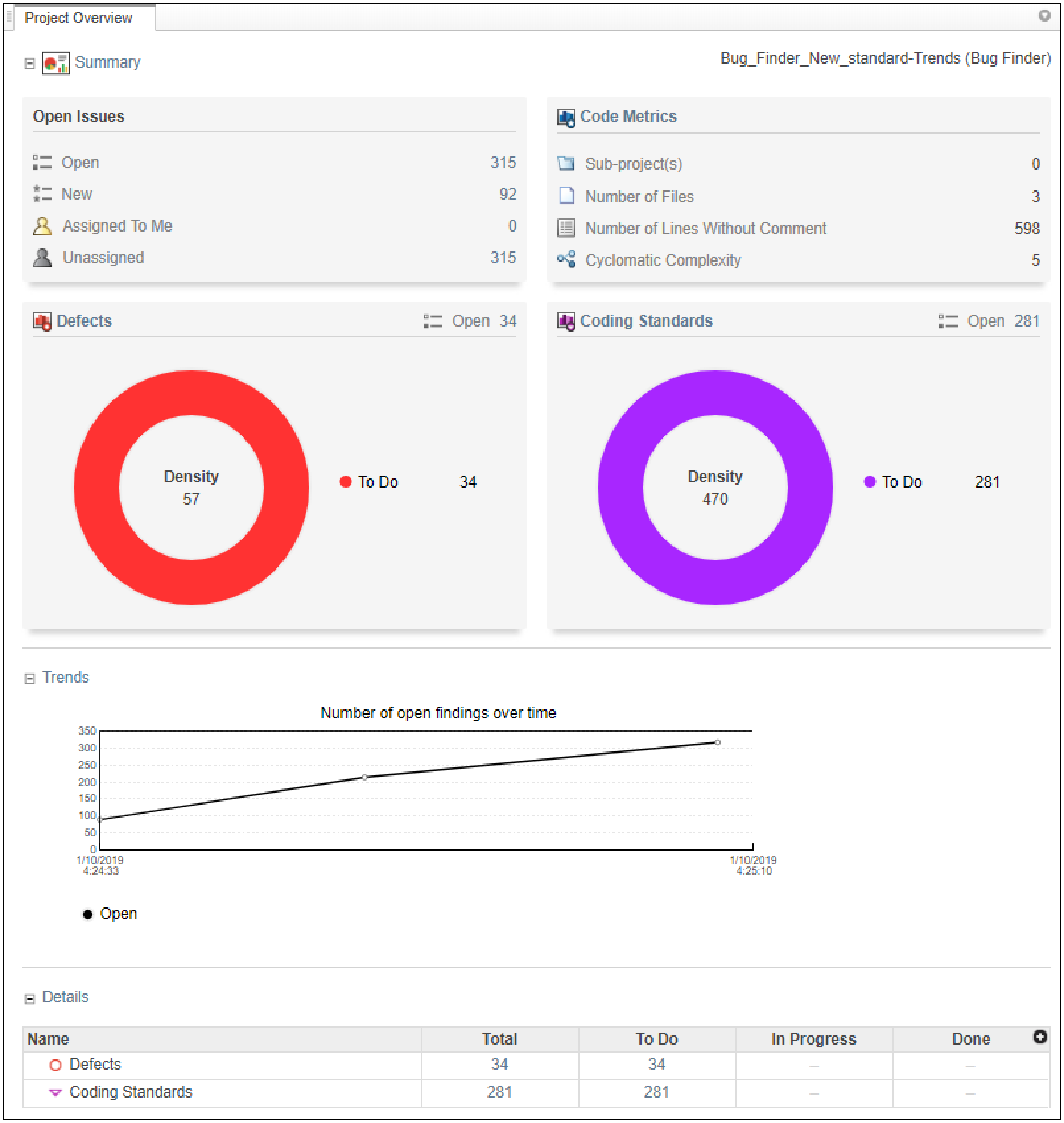
Name	Bug_Finder_Example (Bug Finder)
Language	C
Tools	Bug Finder
Coding Rules	Custom Coding Rules, MISRA C:2012
Number of Runs	1

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



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

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 that you have permission to view. If the folder contains a project for which you are not an **Administrator**, **Owner**, or **Contributor**, results for that project are not included in the aggregate calculation.

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 a section of a pie chart or the pie chart legend (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**.

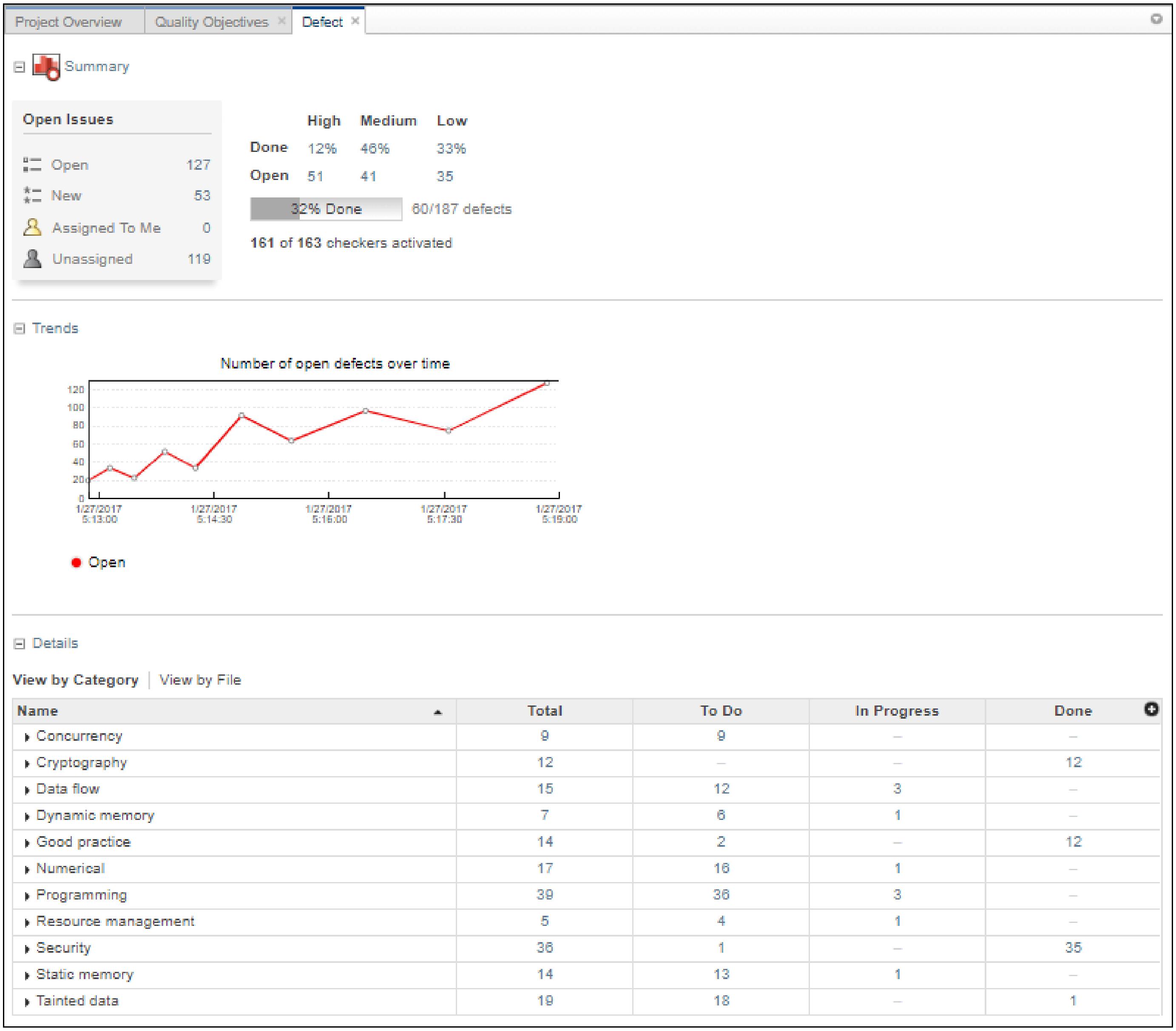
  The card also shows the density, the number of defects or coding standard violation per one thousand lines of code without comments. To view the density you must enable **Code Metrics** in your analysis configuration.
- **Trends**

  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.

  Green run-time checks, green shared variables, not shared variable, and code metrics do not count toward the number of **To Do**, **In Progress**, and **Done** 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

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**

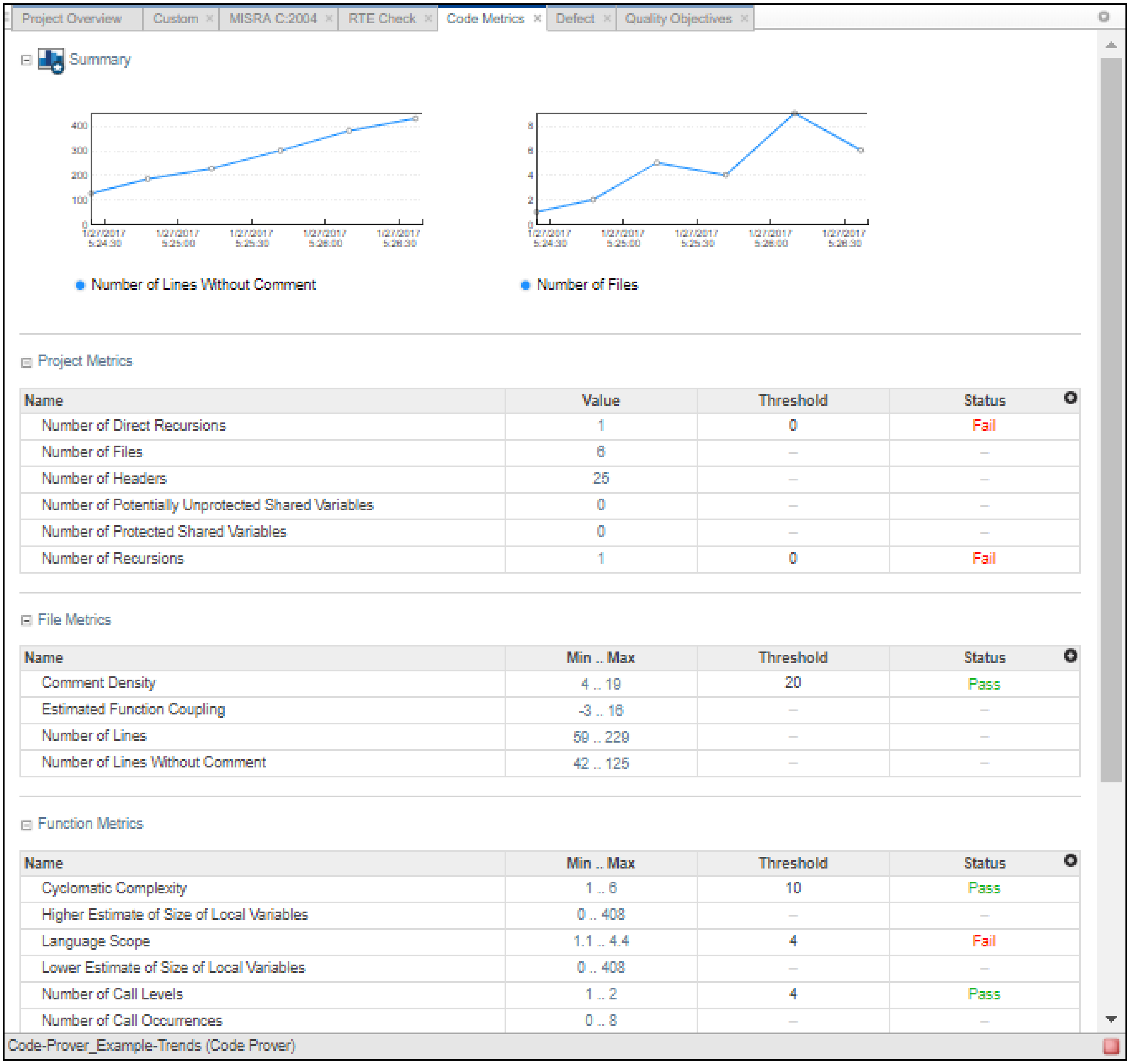
  Displays a table with information about open issues and the progress in addressing these issues. Click the number of findings in the card (when applicable) to open a list filtered to this set of findings.
- **Trends**

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	–	–

Code-Prover_Example-Trends (Code Prover)

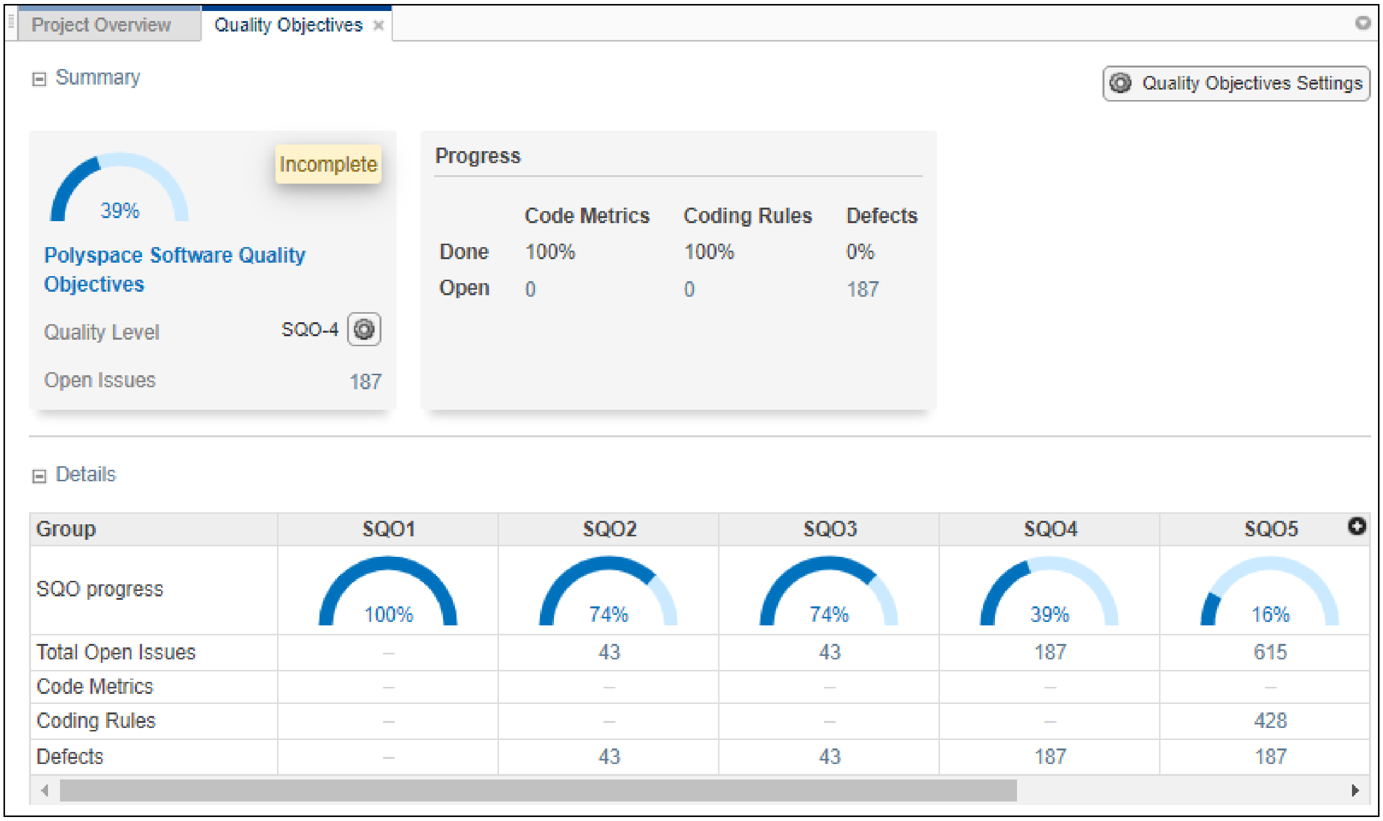
This dashboard contains four collapsible sections.

- **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



Polyspace Software Quality Objectives

Quality Level	SQO-4
Open Issues	187

Progress

	Code Metrics	Coding Rules	Defects
Done	100%	100%	0%
Open	0	0	187

Group	SQO1	SQO2	SQO3	SQO4	SQO5
SQO progress	100%	74%	74%	39%	16%
Total Open Issues	–	43	43	187	615
Code Metrics	–	–	–	–	–
Coding Rules	–	–	–	–	428
Defects	–	43	43	187	187

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.

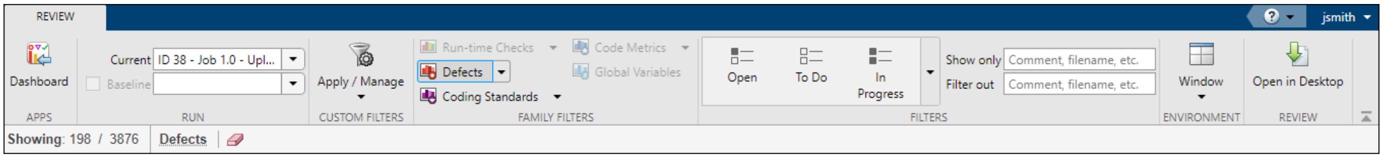
Review

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.

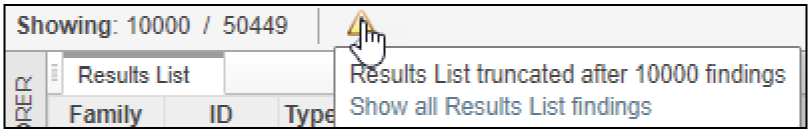
Note The **REVIEW** perspective is only available for analysis results generated with a Polyspace product version R2019a or later.

REVIEW toolstrip



- 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.

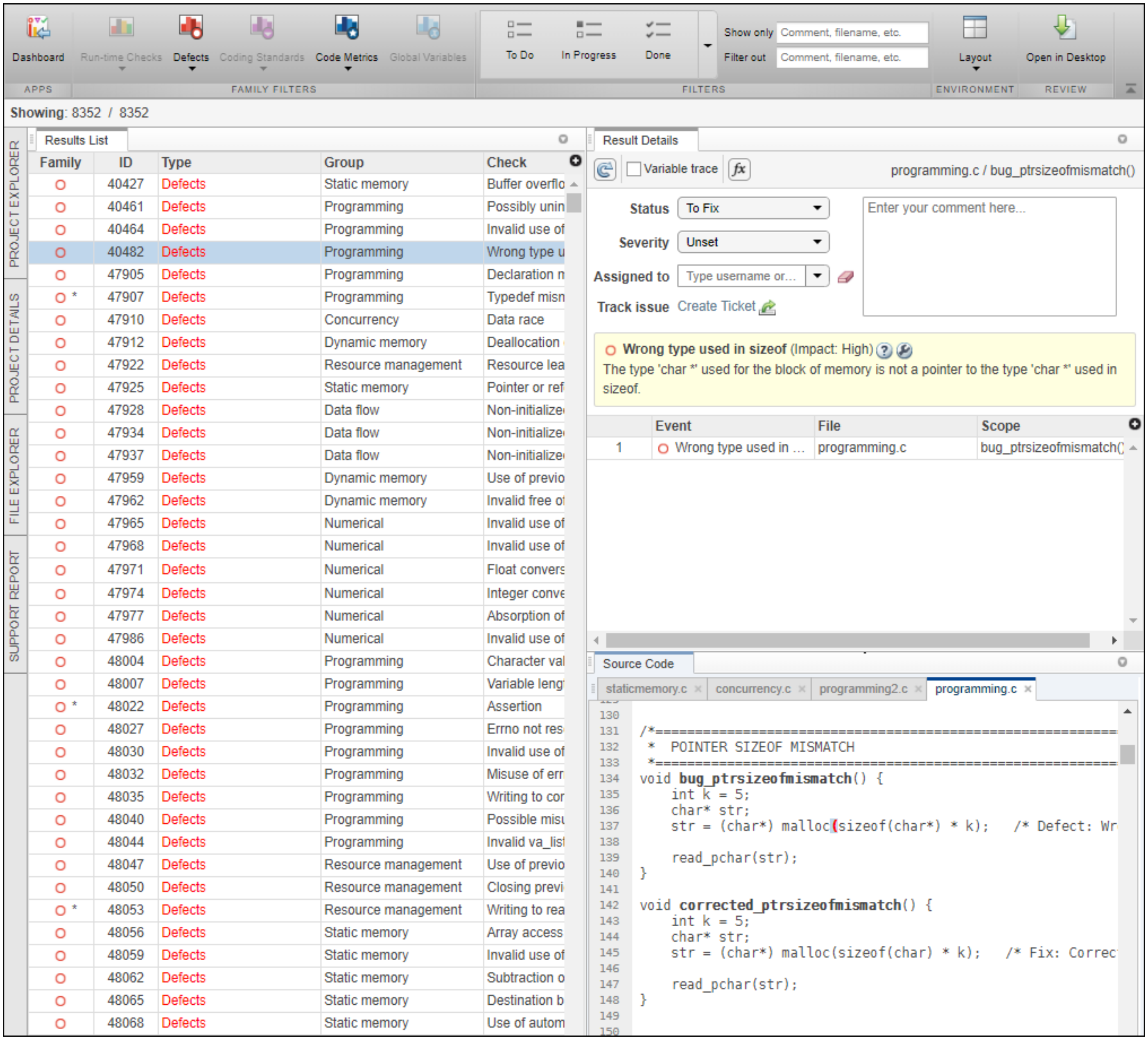
  Results List truncated after 10000 findings

  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.

  When you select the **To Do**, **In Progress**, or **Done** filters, the filtered **Results List** does not show green run-time checks, green shared variables, not shared variables, or code metrics findings.

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



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 m
○ *	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

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()

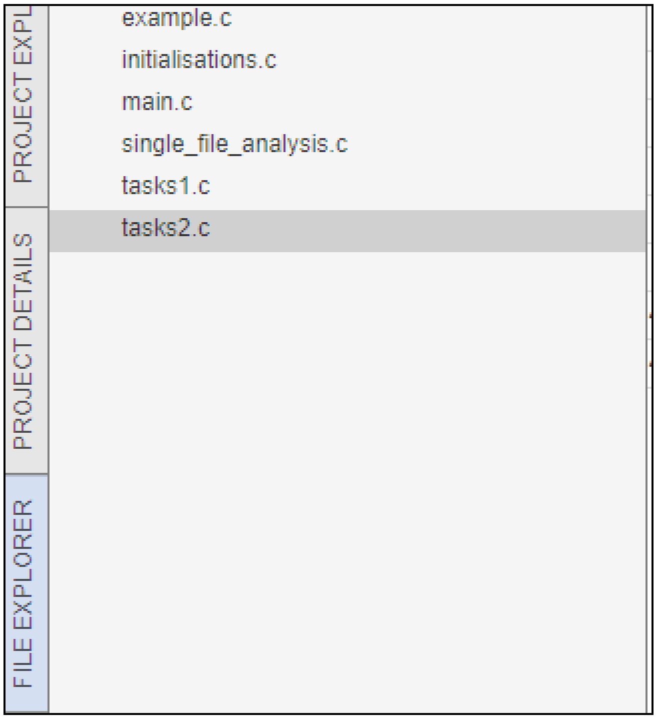
```
/*==============================================================
 *  POINTER SIZEOF MISMATCH
 *==============================================================
void bug_ptrsizeofmismatch() {
    int k = 5;
    char* str;
    str = (char*) malloc(sizeof(char*) * k);   /* Defect: Wr

    read_pchar(str);
}

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

    read_pchar(str);
}
```

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

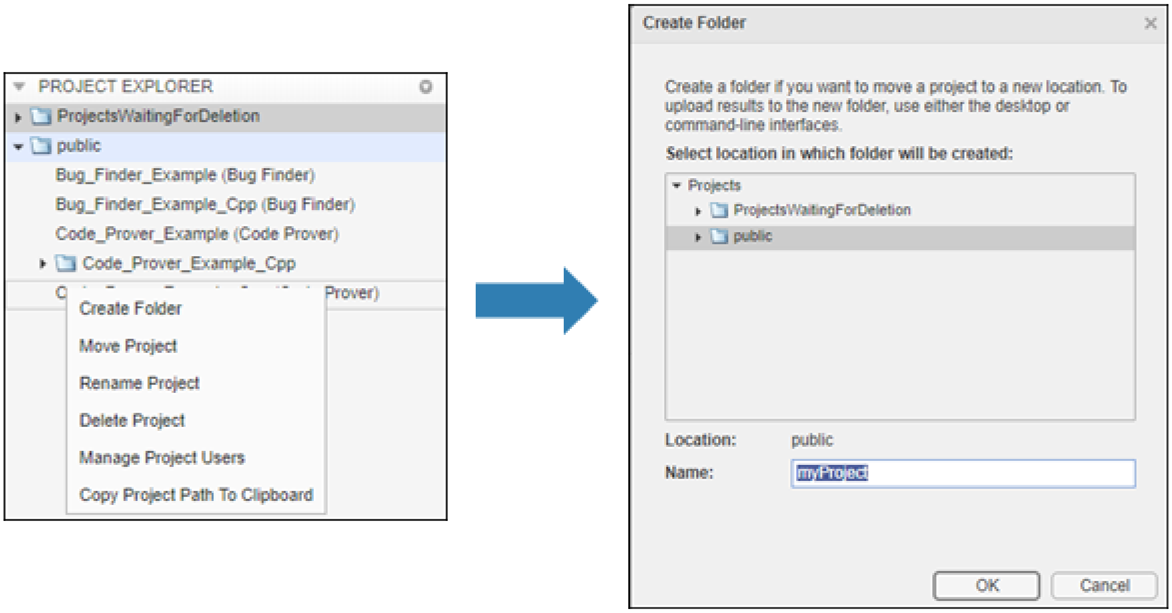
Before you start reviewing the overall quality of a project and investigating findings in your code, create project folders and set permissions to allow or restrict team members access to your projects.

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 folder `myRelease`, use this command:

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

`hostName` and *`port`* correspond to the host name and port number you specify in the URL of the Polyspace Access interface, for example `https://`*`hostName`*`:`*`port`*`/metrics/index.html`. If you are unsure about which host name and port number to use, contact your Polyspace Access administrator. Depending on your configuration, you might also need to specify the `-protocol` option 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 are the permissions that correspond to each role.

Role	Permission
Administrator	Move, rename, or delete specified folders or projects and review their content. Assign roles **Administrator**, **Owner**, **Contributor**, or **Forbidden**. View and manage contents of **ProjectsWaitingForDeletion** folder. See “Delete Outdated Projects” on page 1-48. 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.
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 to the specified folder or project. Set this role to restrict the access to a project inside a folder that is accessible to the user.

Only **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 **Administrator** roles can assign other users as administrators or as owners to a project or folder. To set a user as **Administrator**, see “Manage Permissions in Polyspace Access Web Interface” on page 3-18 or “Configure Polyspace Access App Services” on page 1-34.

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`. The **Administrator** roles applies to all projects.

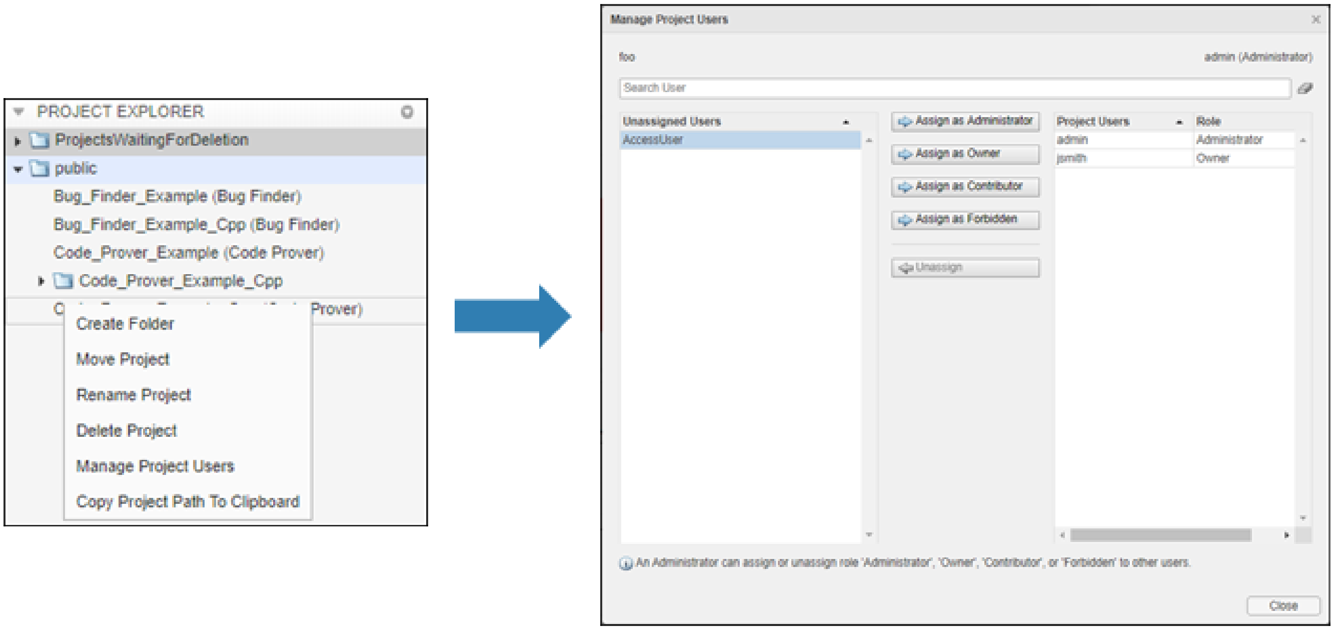
By default, all users have **Contributor** privileges for the **public** folder.

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.

- **Administrator** role can assign other users as **Administrator**, **Owner**, **Contributor**, or **Forbidden**.
- **Owner** role can assign other users as **Contributor** or **Forbidden**.

- **Contributor** and **Forbidden** roles cannot assign roles to other users.



Project Users	Role
admin	Administrator
jsmith	Owner

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

The **Assign as Administrator** button is visible only for users with role **Administrator**. You must assign at least one user as administrator in the cluster operator settings on page 1-34 before you can manage administrators from the web interface.

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:

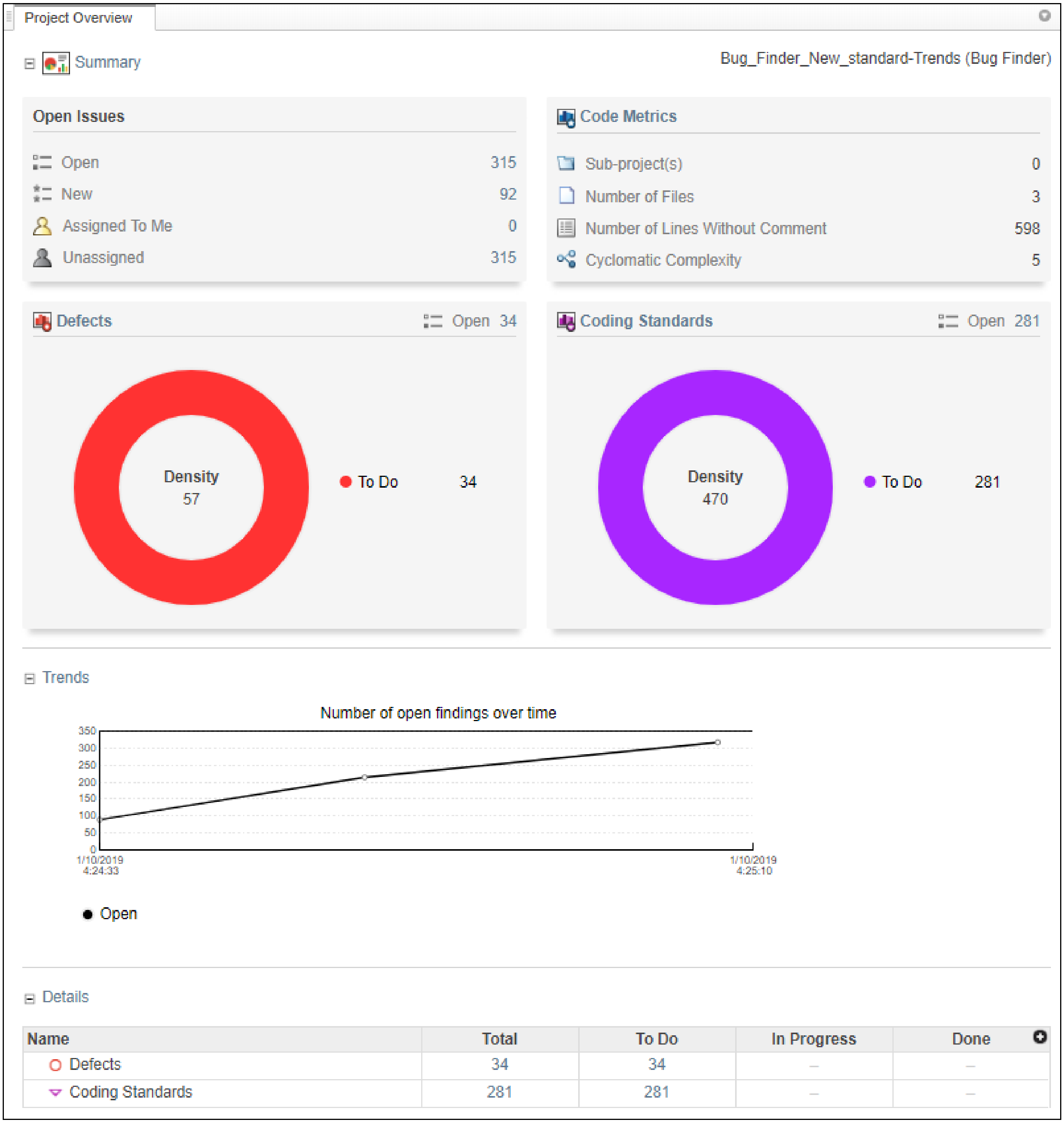
```
polyspace-access -host hostName ^
-set-role contributor -user jsmith ^
-project-path myProject
```

`hostName` and *`port`* correspond to the host name and port number you specify in the URL of the Polyspace Access interface, for example `https://hostName:port/metrics/index.html`. If you are unsure about which host name and port number to use, contact your Polyspace Access administrator. Depending on your configuration, you might also need to specify the `-protocol` option in the migration command.

You cannot assign the **Administrator** role to a user from the command line.

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 that includes multiple projects, the dashboard displays an aggregate of results for all the projects that you have permission to view. If the folder contains a project for which you are not an **Administrator**, **Owner**, or **Contributor**, results for that project are not included in the aggregate calculation.

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`.

Green run-time checks, green shared variables, non-shared variables, and code metrics do not count toward the number of **To Do**, **In Progress**, and **Done** findings.

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 3-22.

See Also

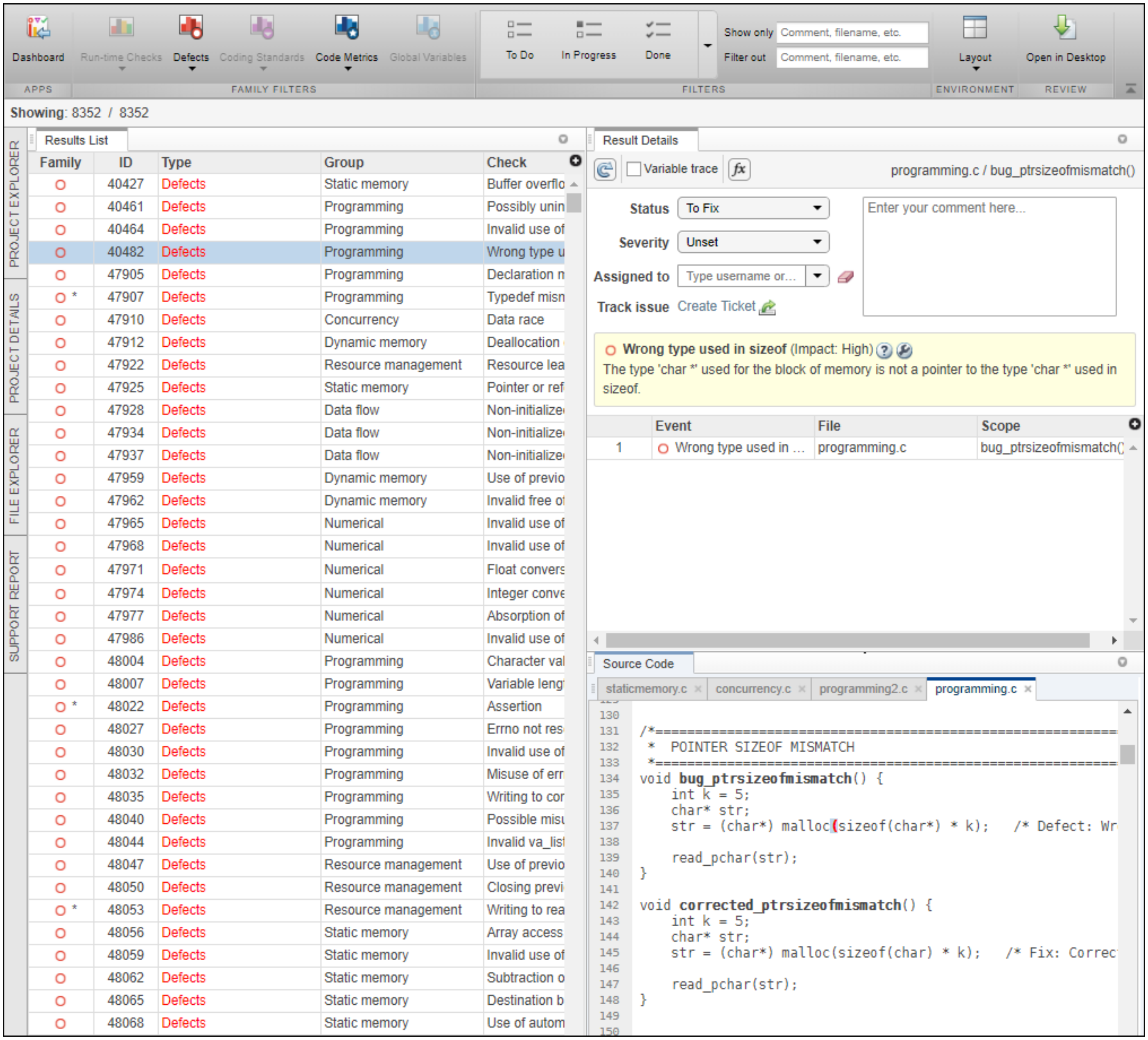
More About

- “Upload Results to Polyspace Access” on page 3-2

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 3-17.

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**.



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 lengt
○ *	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()

□ Variable trace

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 bug tracking tool (BTT) ticket and keep track of the workflow that addresses a finding from an existing BTT project, click **Create** in the **Results Details** pane. Creating a BTT ticket is available only if Polyspace Access is configured to create BTT tickets. The ticket entry is populated with details of the finding and a URL to open the finding in Polyspace Access. See “Track Issue in Bug Tracking Tool”.

See Also

More About

- “Interpret Results”
- “Manage Results”

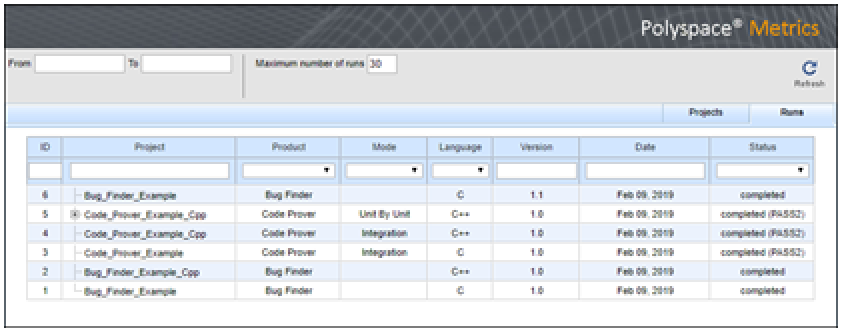
Migrate Results from Polyspace Metrics to Polyspace Access

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 details. After you narrow down the set of findings that you want to address, open them in the **REVIEW** perspective to start reviewing individual results.

Note The **REVIEW** perspective is only available for analysis results generated with a Polyspace product version R2019a or later. To review R2018b or earlier results that you migrated to Polyspace Access, see “Open Polyspace Access Results in a Desktop Interface” on page 3-5.

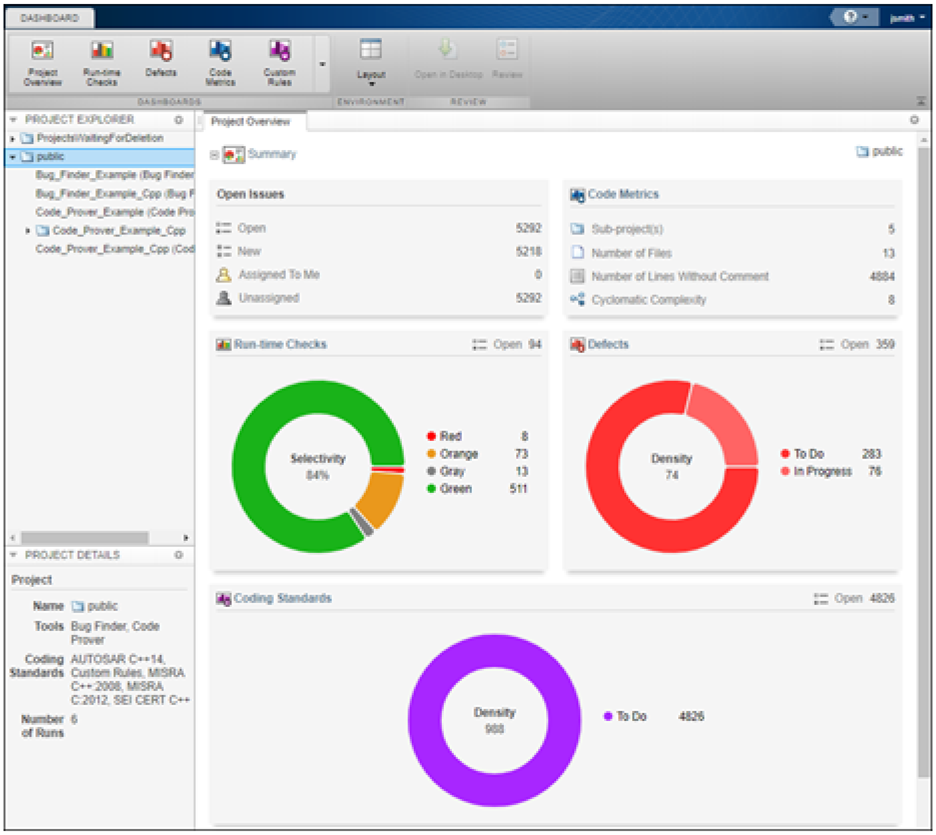
You can also review results from Polyspace Access by opening them in the Polyspace desktop interface. You do not need to download a local copy of Polyspace Access results to view those results in the desktop interface. The edits that you make to the results are saved directly in Polyspace Access and 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.
-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 -port port  ^
-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` and *`port`* correspond to the host name and port number you specify in the URL of the Polyspace Access interface, for example `https://`*`hostName`*`:`*`port`*`/metrics/index.html`. If you are unsure about which host name and port number to use, contact your Polyspace Access administrator. Depending on your configuration, you might also need to specify the `-protocol` option 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 1-40
- "Upload Results to Polyspace Access" on page 3-2

Quick Start Guide for Polyspace Server and Access Products

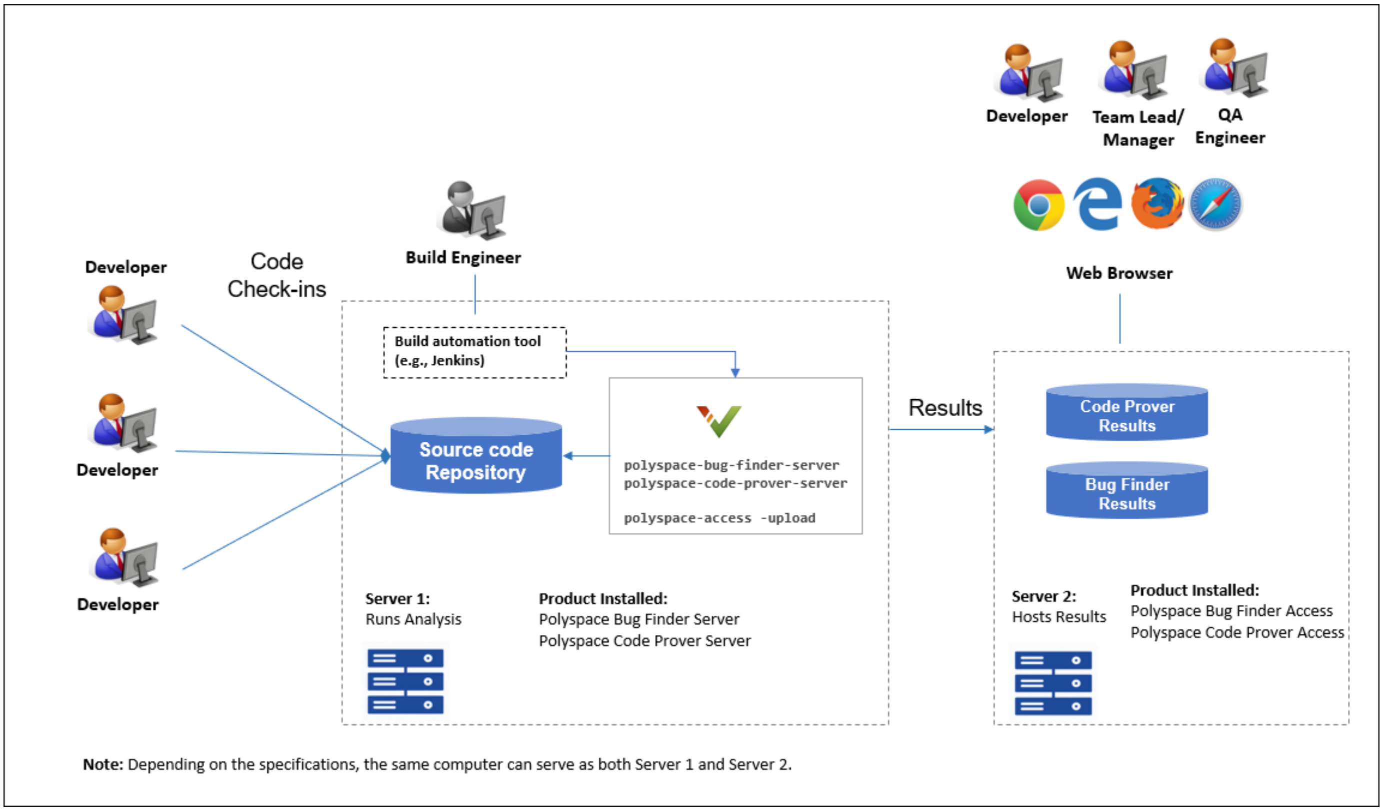
To avoid finding bugs late in the development process, run static analysis by using Polyspace products.

- **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.



```
polyspace-bug-finder-server
polyspace-code-prover-server

polyspace-access -upload
```

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 user manager to authenticate user logins, an issue tracker to integrate your bug tracking tool with Polyspace, a database to manage results, 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 Admin.

To install Polyspace Access:

- Download the installer as a zip file.
- Unzip the file and start the Cluster Admin. From the Cluster Admin 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
```

Install Network License Manager

Both Polyspace Server and Polyspace Access use licenses that require communication with a network license manager for license checkouts.

- To install, configure and start the network license manager for Polyspace Server, see “Administer Network Licenses”.
- To install, configure and start the network license manager for Polyspace Access, see “Manage Polyspace Access License”.

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/R2021a`, see also “Installation Folder” (Polyspace Bug Finder Server)). 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 Server 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 in Polyspace Access Web Interface”
  - “Address Results in Polyspace Access 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.

Install Polyspace as You Code

Install Polyspace as You Code Using Installer

Polyspace as You Code checks your code for bugs and coding standard violations while you work in a code editor or in these supported IDEs:

- Visual Studio 2017 and 2019
- Visual Studio Code (versions 1.49.3 and later[1])
- Eclipse (versions 4.11 to 4.18)[2]

Polyspace as You Code uses the Polyspace Access Network Named Users (NNU) license and requires a license manager to manage license checkouts.

Prepare Polyspace as You Code Installation

The Polyspace as You Code installer is available in the Polyspace Access installation image.

In a typical installation:

1. An administrator downloads Polyspace Access, installs the license manager on a server machine, and configures the Polyspace Access NNU license.

   a. To download the Polyspace Access installation image:

      i. Go to the MathWorks download page and click the **Download Rxxxxy** button. You might have to log into your MathWorks account to complete this step.

      ii. On the next page, select the Polyspace Access link under Additional Rxxxxy Product Downloads.

      Rxxxxy corresponds to a release number, for instance R2021a.

   b. To install the license manager and configure the Polyspace Access license. See “Configure Polyspace Access License” on page 2-2.

      Provide end users with the path to the license file that you configure in this step. The license is typically named `network.lic` and contains these lines:

      ```
      SERVER lmHostname HostID 27000
      USE_SERVER
      ```

      where *`lmHostname`* is the host name of the machine where you install the license manager. *`HostID`* is the MAC address that you provided to activate the Polyspace Access license.

2. An administrator or an end user installs Polyspace as You Code on client machines that are on the same network as the license manager server machine.

   This table describes additional required steps before you begin your installation, depending on your role and the type of installation.

[1] It is likely that Polyspace as You Code will run successfully with versions much later than 1.49.3 (compared to the release date of Polyspace as You Code), however, there might be certain issues.

[2] If you install the Polyspace as You Code Eclipse plugin on Linux, see “Configure Eclipse for Supported Java Version on Linux” on page 4-15.

Type of Installation	Role	Steps
Download installer from Polyspace Access interface and install interactively	Administrator	**a** Install Polyspace Access. See “Install Polyspace Access”. **b** Provide end users with the Polyspace Access URL.
	End User	**a** Obtain the path to the license file and Polyspace Access URL from your administrator. **b** Download and unzip the Polyspace as You Code installation folder. Log into the Polyspace Access interface and click ? > **Download Polyspace as You Code**. **c** For installation, see “Install Polyspace as You Code Interactively” on page 4-3. If you install Visual Studio Code or Eclipse IDE extensions, you must provide the installation path of the IDEs.
Start installer from shared location and install interactively	Administrator	**a** Unzip the Polyspace as You Code installation folder to a location that is accessible from end user client machines. The installation folders for Windows and Linux are available under the `download` folder in the location where you unzipped the Polyspace Access installation image. **b** Provide end users with the path to the installation folder that you unzipped in step a.
	End User	**a** Obtain the path to the license file and location of the installer from your administrator. **b** For installation, see “Install Polyspace as You Code Interactively” on page 4-3. If you install Visual Studio Code or Eclipse IDE extensions, you must provide the installation path of the IDEs.
Install noninteractively	Administrator or End User	**a** Obtain the path to the license file and location of the installer. **b** Modify the installer properties file: • Provide the path for the Polyspace as You Code installation folder. • Determine whether you want to install IDE extensions. For Visual Studio Code and Eclipse IDEs, you must provide the installation path of the IDEs. See “Install Polyspace as You Code Noninteractively” on page 4-5.

Install Polyspace as You Code Interactively

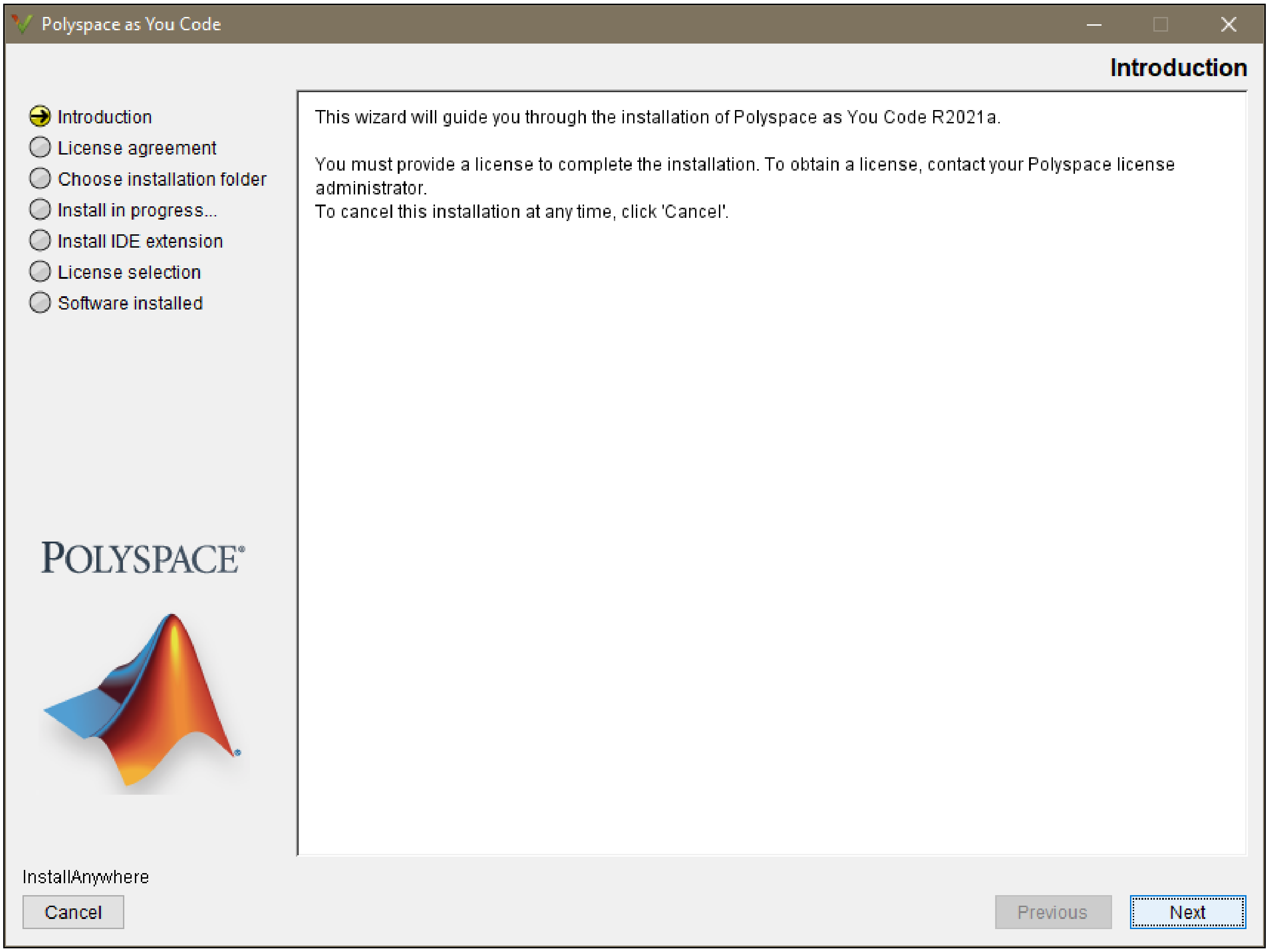
Prerequisites

— See “Prepare Polyspace as You Code Installation” on page 4-2.

To install Polyspace as You Code interactively, at the command line, navigate to the folder that contains the Polyspace as You Code installer and enter the commands listed in this table, depending on your operating system.

Windows	`setup.exe` You can also start the installer by double-clicking the `setup.exe` binary.
Linux	`./install.sh`

Note If you install the Polyspace as You Code Eclipse plugin on Linux, see "Configure Eclipse for Supported Java Version on Linux" on page 4-15.



This wizard will guide you through the installation of Polyspace as You Code R2021a.

You must provide a license to complete the installation. To obtain a license, contact your Polyspace license administrator.
To cancel this installation at any time, click 'Cancel'.

In the Polyspace as You Code installation wizard, click **Next** and follow the prompts to complete the installation. If you install the IDE extensions:

- The installer might take some time before you can proceed to the next installation step.

- Before you start using Polyspace as You Code, you must complete the configuration of the extensions. See Configure Polyspace as You Code IDE Extensions.

Uninstall Polyspace as You Code Interactively

1. Go to the `Uninstall` folder located in your Polyspace as You Code installation folder, for instance `C:\Program Files\Polyspace as You Code\R2021a\Uninstall`.
2. Start the `Uninstall.exe` binary (Windows) or the `Uninstall` script (Linux).

The operation uninstalls Polyspace as You Code and the IDE extensions.

Install Polyspace as You Code Noninteractively

Prerequisites

— See “Prepare Polyspace as You Code Installation” on page 4-2.

When installing noninteractively, you can either:

- Start the installer by using a properties file where you specify your installation configuration.
- Start the installer by using the `-silent` option to install only the Polyspace as You Code analysis engine.

Install Polyspace as You Code by Using Installer Properties File

To specify your installation configuration in a properties file and install Polyspace as You Code noninteractively:

1. Edit the installer properties file. A template installer properties file is located in *`installerRoot`*`/Docs/installer.properties`, where *`installerRoot`* is the folder that contains the Polyspace as You Code installer.
2. At the command line, navigate to the folder that contains the Polyspace as You Code installer. Depending on your operating system, enter one of the commands listed in this table.

Windows	`setup.exe -f` *`installerPropertiesFile`*
Linux	`./install.sh -f` *`installerPropertiesFile`*

`installerPropertiesFile` is the full file path to the installer properties file.

For example, to install Polyspace as You Code in folder `C:\Program Files\Polyspace as You Code\R2021a` with license file `C:\Polyspace\licenses\network.lic`, specify the following installer properties file. The line `INSTALL_VS_PLUGIN=true` enables the installation of the Visual Studio extension.

```
# Installer properties file
# Uncomment the line below to launch the installer in silent mode.
INSTALLER_UI=silent

# Uncomment the line below and specify an installation folder path or
# leave line commented to install in default installation folder.
# Enter '\' characters in the folder path as '\\', for instance:
# C:\\Program Files\\Polyspace as You Code\\R2021a
USER_INSTALL_DIR=C:\\Program Files\\Polyspace as You Code\\R2021a

##### Begin - Windows only options #####
# Set to true to install the Visual Studio extension.
INSTALL_VS_PLUGIN=true
##### End   - Windows only options #####
```

```
# Set to true to install the Visual Studio Code extension.
INSTALL_VSCODE_PLUGIN=false
# If INSTALL_VSCODE_PLUGIN is set to "true", provide the path to the
# Visual Studio Code installation folder, for instance /usr/share/code
VSCODE_INSTALL_FOLDER=

# Set to true to install the Eclipse extension.
INSTALL_ECLIPSE_PLUGIN=false
# If INSTALL_ECLIPSE_PLUGIN is set to "true", provide the path to the
# Eclipse installation folder
ECLIPSE_INSTALL_FOLDER=

# Provide the path to the license file. The file will be copied to <install_root>/licenses.
LICENSE_FILE=C:\\Polyspace\\licenses\\network.lic
```

Note If you install the Polyspace as You Code Eclipse plugin on Linux, see “Configure Eclipse for Supported Java Version on Linux” on page 4-15.

Install Only Polyspace as You Code Analysis Engine

To install only the Polyspace as You Code analysis engine without opening the graphical interface, start the installer binary by using the `-silent` flag. The installer installs Polyspace as You Code to these default locations, based on your operating system:

Windows	`C:\Program Files\Polyspace as You Code\R2021a`
Linux	`/usr/local/PolyspaceAsYouCode/R2021a`

To specify a different installation path, use the `-install-dir` flag, for instance `./install.sh -silent -install-dir /local/myFolder`. To complete the installation, copy your Polyspace Access license to the `licenses` folder located in the Polyspace as You Code installation folder.

Uninstall Polyspace as You Code Noninteractively

To uninstall Polyspace as You Code without opening the graphical interface:

1. Go to the `Uninstall` folder located in your Polyspace as You Code installation folder, for instance `C:\Program Files\Polyspace as You Code\R2021a\Uninstall`.
2. Start the `Uninstall.exe` binary (Windows) or the `Uninstall` script (Linux or Mac OS) by using the `-i silent` flag.

The uninstall operation removes Polyspace as You Code and the IDE extensions.

See Also

Related Examples

- “Configure Polyspace Access License” on page 2-2
- “Install Polyspace as You Code Extension in Visual Studio” on page 4-7
- “Install Polyspace as You Code Extension in Visual Studio Code” on page 4-9
- “Install Polyspace as You Code Plugin in Eclipse” on page 4-12

Install Polyspace as You Code Extension in Visual Studio

The Polyspace as You Code extension in the Visual Studio IDE allows you to run Polyspace on the file that you are currently viewing and see analysis results such as bugs and coding standard violations. You must install Polyspace as You Code separately to run the analysis. The extension allows you to point to this installation from Visual Studio and show results produced by the Polyspace analysis.

You can install the extension in one of two ways:

- While installing the Polyspace as You Code analysis engine using the installer that you download from Polyspace Access on the web.

  See "Install Polyspace as You Code Using Installer" on page 4-2.
- After installing the Polyspace as You Code analysis engine, using a Visual Studio extension installer or in short, a VSIX file, provided with your Polyspace as You Code installation.

The rest of this topic describes the second approach where you install the extension separately from installing the Polyspace as You Code analysis engine.

You can install the extension in Visual Studio 2017 and 2019.

Interactive Installation

To install the extension interactively:

1. Double-click the `VSIX` file in the folder *`polyspaceroot`*`\polyspace\plugin\visual_studio`.

   Here, *`polyspaceroot`* is the Polyspace as You Code installation folder, for instance, `C:\Program Files\Polyspace as You Code`.
2. Follow the prompts on the screen.

   If you see a message that indicates that your Visual Studio IDE does not satisfy the prerequisites for the plugin and you are using Visual Studio 2017 or Visual Studio 2019, install the latest updates to these versions and try re-installing the plugin.

After installation, open the Visual Studio IDE and check that the extension has been installed. For instance, in Visual Studio 2019, select **Extensions > Manage Extensions**. You should see Polyspace in the list of extensions installed. You can also disable or uninstall the extension right from this list.

Command-Line Installation

If you want to install the extension without opening a graphical user interface, you can run the VSIX installer at the command line with the `/q` flag.

- To install the extension in all Visual Studio versions on the machine, enter:

  ```
  cd VSIXInstallerpath
  VSIXInstaller.exe /q polyspaceroot\polyspace\plugin\visual_studio\POLYSPACE_FOR_VS.vsix
  ```

  Here, *`VSIXInstallerpath`* is the path to the VSIX installer file. For instance, in a Visual Studio 2019 installation, the path can be `C:\Program Files (x86)\Microsoft Visual Studio \2019\Professional\Common7\IDE` .
- To install the extension on a specific Visual Studio versions installed on the machine, enter:

```
cd VSIXInstallerpath
VSIXInstaller.exe /q /s:name /v:version ^
polyspaceroot\polyspace\plugin\visual_studio\POLYSPACE_FOR_VS.vsix
```

Here, *name* is name of the Visual Studio application, for instance, `Pro` for the Visual Studio Professional edition, and *version* is the version number in the form *major.minor*, for instance, `16.0` for the major version of Visual Studio 2019.

To uninstall the extension silently, use the `/u` flag.

For more information on the flags, search for the online VSIXInstaller documentation or simply enter:

```
cd VSIXInstallerpath
VSIXInstaller.exe
```

See Also

More About

- “Install Polyspace as You Code Using Installer” on page 4-2

Install Polyspace as You Code Extension in Visual Studio Code

The Polyspace as You Code extension in the Visual Studio Code IDE allows you to run Polyspace on the file that you are currently viewing and see analysis results such as bugs and coding standard violations. You must install Polyspace as You Code separately to run the analysis. The extension allows you to point to this installation from Visual Studio Code and show results produced by the Polyspace analysis.

You can install the extension in one of two ways:

- While installing the Polyspace as You Code analysis engine using the installer that you download from Polyspace Access on the web.

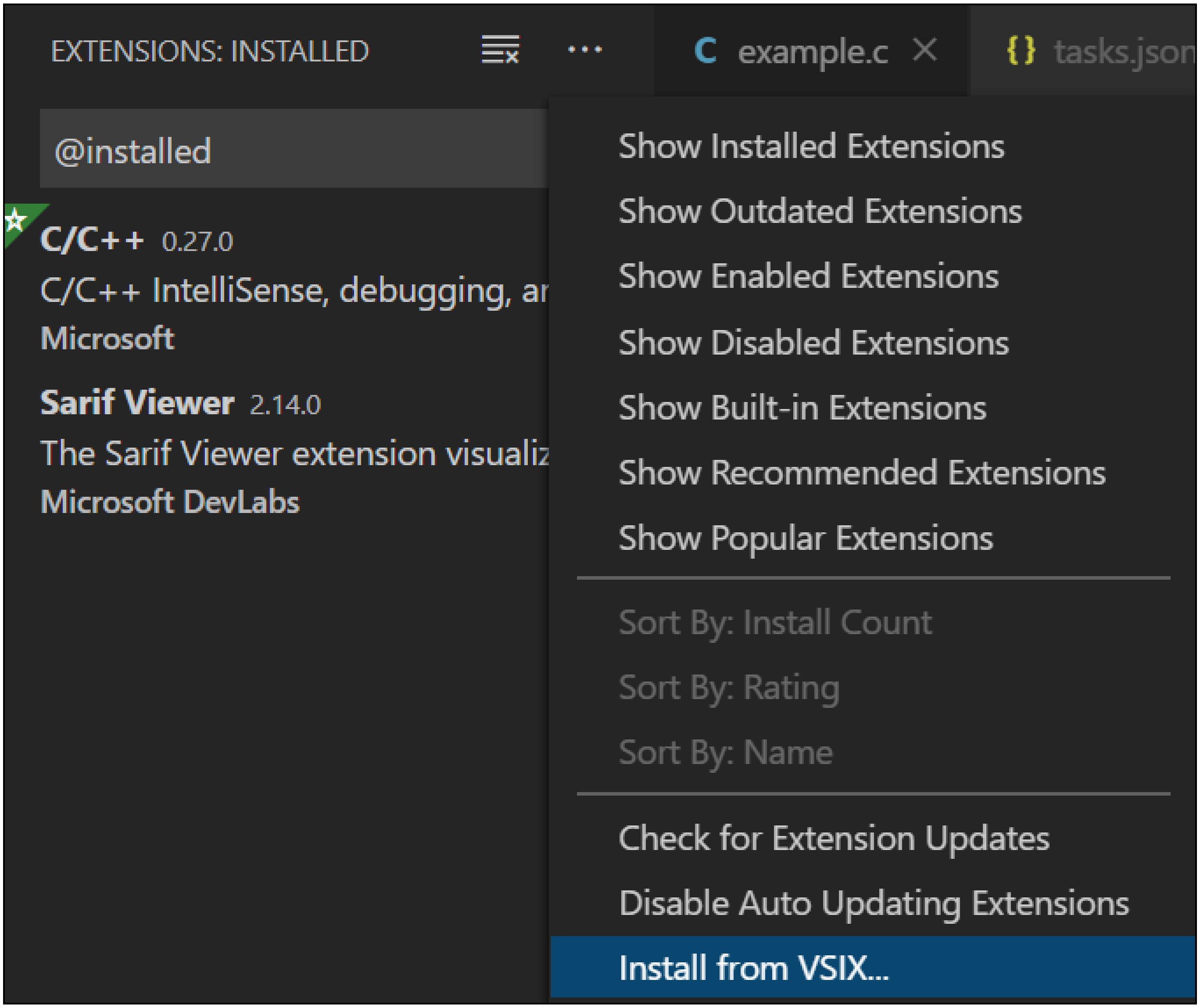
  See “Install Polyspace as You Code Using Installer” on page 4-2.
- After installing the Polyspace as You Code analysis engine, using a Visual Studio Code extension installer or in short, a VSIX file, provided with your Polyspace as You Code installation.

The rest of this topic describes the second approach where you install the extension separately from installing the Polyspace as You Code analysis engine.

Interactive Installation

To install the extension interactively:

1. In the Visual Studio Code IDE, select **View** > **Extensions**.
2. On the **EXTENSIONS** pane, click the ellipsis on the upper right and select **Install from VISIX**.



3 Navigate to the VSIX file in the folder `*polyspaceroot*\polyspace\plugin\visual_studio_code`.

   Here, *`polyspaceroot`* is the Polyspace as You Code installation folder, for instance, `C:\Program Files\Polyspace as You Code`.

After installation, you can see the extension in the **EXTENSIONS** pane.

To uninstall the extension, on the **EXTENSIONS** pane, click the ⚙ icon and select **Uninstall**.

Command-Line Installation

You can also install the extension using the VSIX file at the command line.

- To install the extension, in a command window, enter:

```
code --install-extension polyspaceroot\polyspace\plugin\visual_studio\POLYSPACE_FOR_VS.vsix
```

- To uninstall the extension, enter:

```
code --uninstall-extension polyspaceroot\polyspace\plugin\visual_studio\POLYSPACE_FOR_VS.vsix
```

See Also

More About

- “Install Polyspace as You Code Using Installer” on page 4-2

Install Polyspace as You Code Plugin in Eclipse

This topic describes how to install the Polyspace as You Code plugin in Eclipse. For Polyspace desktop products such as Polyspace Bug Finder, see the topic "Polyspace Analysis in Eclipse" in the Polyspace Bug Finder documentation.

The Polyspace as You Code plugin in the Eclipse IDE allows you to run Polyspace on the file that you are currently viewing and see analysis results such as bugs and coding standard violations. You must install Polyspace as You Code separately to run the analysis. The plugin allows you to point to this installation from Eclipse and show results produced by the Polyspace analysis.

Note If you install the Polyspace as You Code Eclipse plugin on Linux, see “Configure Eclipse for Supported Java Version on Linux” on page 4-15.

You can install the plugin in one of two ways:

- While installing the Polyspace as You Code analysis engine using the installer that you download from Polyspace Access on the web.

  See “Install Polyspace as You Code Using Installer” on page 4-2.
- After installing the Polyspace as You Code analysis engine.

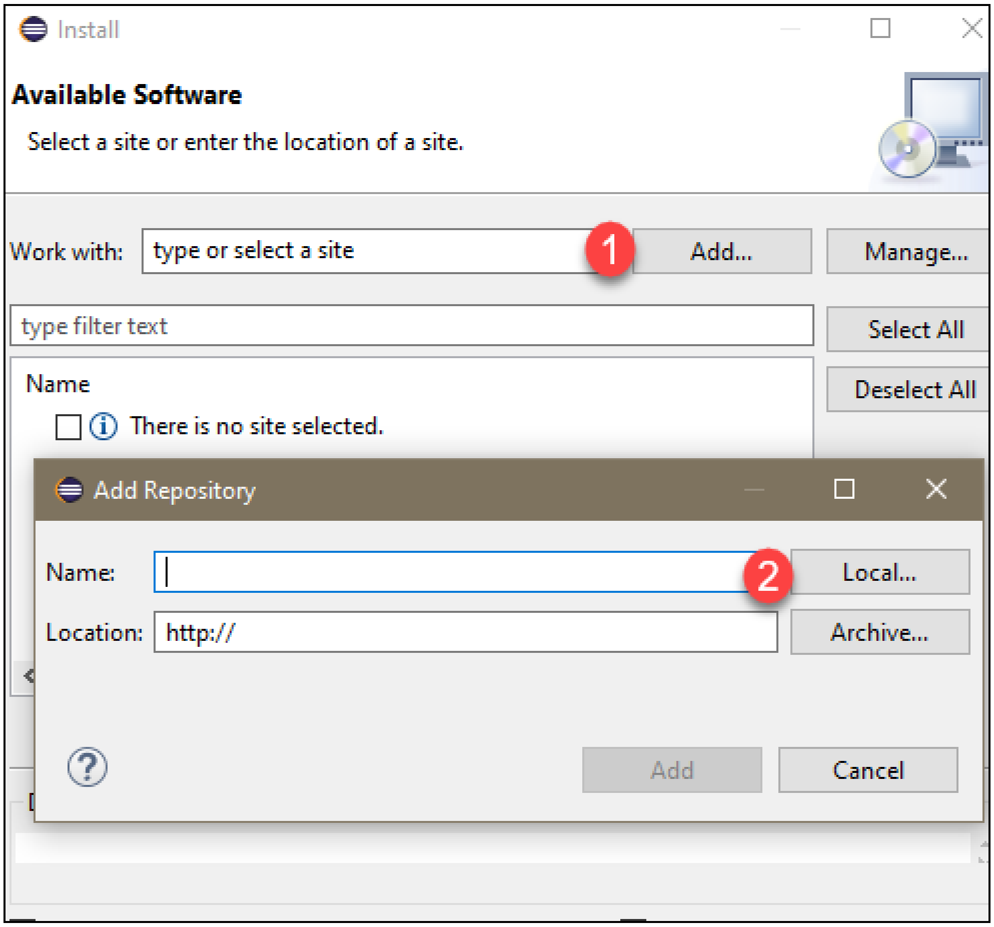
  To install the Polyspace as You Code plugin, use the contents of the `*polyspaceroot*\polyspace\plugin\eclipse` folder provided with your installation. *`polyspaceroot`* is the Polyspace as You Code installation folder, for instance, `C:\Program Files\Polyspace as You Code`.

The rest of this topic describes the second approach where you install the plugin separately from installing the Polyspace as You Code analysis engine.

Interactive Installation

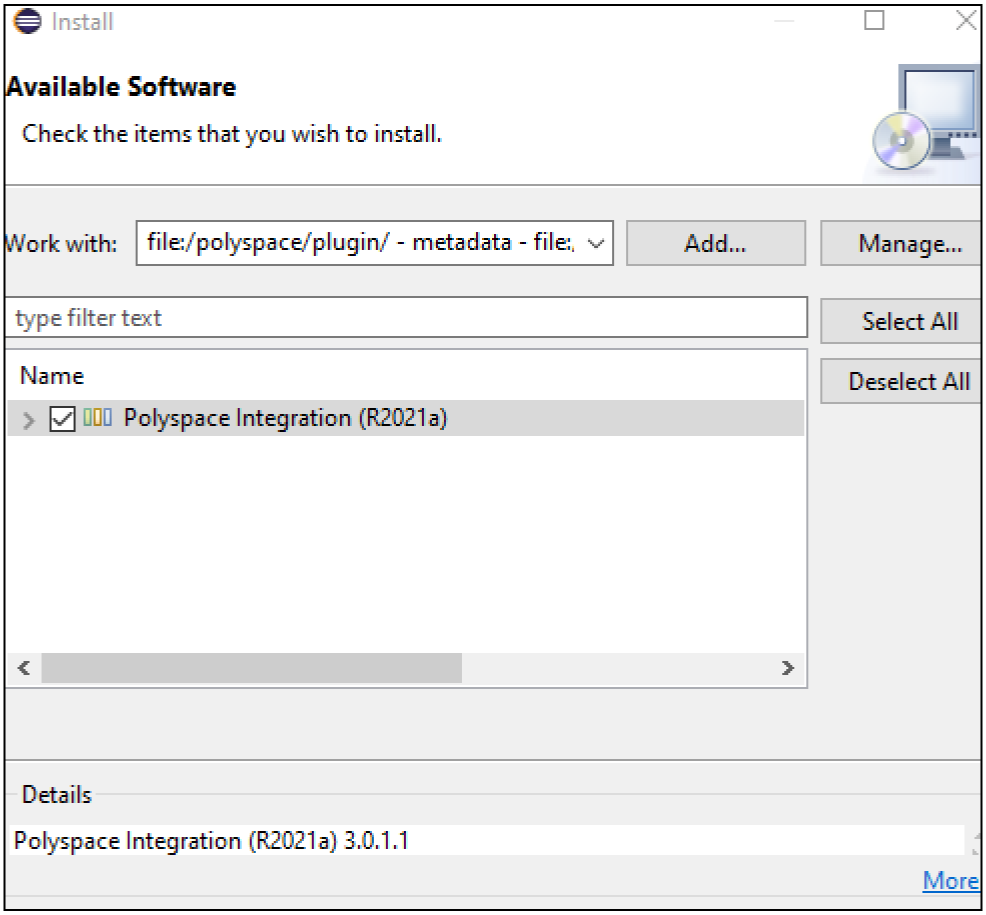
To install the plugin interactively:

1. In the Eclipse IDE, select **Help** > **Install New Software**.
2. In the **Install** window, click **Add**, and then click **Local** in the **Add Repository** popup.



3 Navigate to the *`polyspaceroot`*`\polyspace\plugin\eclipse` folder and click **Select Folder**, then add the folder to the repository.

4 Make sure that the Polyspace plugin is selected, then click **Next** and follow the prompts to complete the installation.



Available Software

Check the items that you wish to install.

Work with: file:/polyspace/plugin/ - metadata - file:,

☑ Polyspace Integration (R2021a)

Polyspace Integration (R2021a) 3.0.1.1

To uninstall the plugin, go to **Help** > **About Eclipse IDE**. In the **About Eclipse IDE** window, click **Installation Details**, select the Polyspace plugin on the **Installed Software** tab, and then click **Uninstall** and follow the prompts.

Command-line Installation

To install the plugin, open a terminal and navigate to your Eclipse installation folder, for instance `C:\Program Files\eclipse`, and enter:

```
eclipsec.exe -application org.eclipse.equinox.p2.director ^
-repository file:"/polyspaceroot/polyspace/plugin/eclipse/" ^
-installIU com.polyspace.eclipse.feature -nosplash
```

where *`polyspaceroot`* is the Polyspace as You Code installation folder, for instance, `C:/Program Files/Polyspace as You Code`. The `-repository` path must be specified with forward slashes `"/"`.

To uninstall the plugin, enter:

```
eclipsec.exe -application org.eclipse.equinox.p2.director ^
-uninstallIU com.polyspace.eclipse.feature -nosplash
```

Configure Eclipse for Supported Java Version on Linux

The Polyspace as You Code Eclipse plugin does not support Java versions 13 and later. If your Eclipse IDE uses an unsupported Java version, you see an error message when you start the IDE.

To configure your IDE to use a supported Java version:

1. Go to https://jdk.java.net/archive, download the OpenJDK version **12 GA (build 12+33)** for Linux, and then right-click the downloaded TAR file to extract its content.

   Alternatively, at the command line, enter these commands:

   ```
   wget https://download.java.net/java/GA/jdk12/33/GPL/openjdk-12_linux-x64_bin.tar.gz
   tar xzvf openjdk-12_linux-x64_bin.tar.gz
   ```
2. Open file *`eclipseRoot`*`/eclipse.ini` in a text editor and replace the path below the `-vm` line with the path to the `bin` folder for the openJDK installation that you downloaded in step 1. The *`eclipseRoot`* folder is your Eclipse IDE installation folder.

   For example, if you extracted the download from step 1 into folder `/local/tools`, the section of your `eclipse.ini` file around the `-vm` line looks similar to this section:

   ```
   --launcher.defaultAction
   openFile
   --launcher.appendVmargs
   -vm
   /local/tools/jdk-12/bin
   ```
3. Restart your IDE to apply the changes.

See Also

More About

- “Install Polyspace as You Code Using Installer” on page 4-2

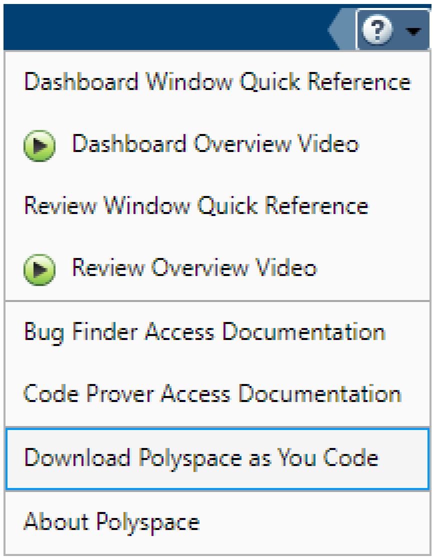
Get Started with Polyspace as You Code

Quick Start Guide for Polyspace as You Code

Polyspace as You Code is a static code analysis software meant for regular use by C/C++ developers within their Integrated Development Environments (IDEs). Polyspace as You Code can find bugs and coding standard violations on the file that is currently active in the IDE. (For full integration analysis of a project, use Polyspace Bug Finder or Polyspace Bug Finder Server.)

Install Polyspace as You Code Analysis Engine and IDE Extensions

Polyspace as You Code comes bundled with a Polyspace Bug Finder Access installation meant for teams or organizations. Once the Polyspace Access web server is set up, any of the licensed users can download the Polyspace as You Code installer as a zipped file from the Polyspace Access web interface.


Dashboard Window Quick Reference
Dashboard Overview Video
Review Window Quick Reference
Review Overview Video
Bug Finder Access Documentation
Code Prover Access Documentation
Download Polyspace as You Code
About Polyspace

Conceptually, Polyspace as You Code consists of these parts:

- An analysis engine
- An IDE extension that allows you to launch an analysis and view results in your IDE

  IDE extensions are provided for these IDEs: Visual Studio, Visual Studio Code and Eclipse. If you use another IDE, you can still install the analysis engine and run the analysis from the command line or IDE console.

Unzip and start the installer and follow the on-screen instructions. After the analysis engine is installed, you have a choice to install one or more IDE extensions. For more information, see “Install Polyspace as You Code Using Installer” on page 4-2.

Alternatively, you can install the IDE extensions later. For more information, see:

- “Install Polyspace as You Code Extension in Visual Studio” on page 4-7
- “Install Polyspace as You Code Extension in Visual Studio Code” on page 4-9
- “Install Polyspace as You Code Plugin in Eclipse” on page 4-12

Run Polyspace as You Code and Review Results

After installation, each time open your IDE, the Polyspace as You Code extension is ready to start an analysis. If you open a C or C++ file, make some changes and save the file, an analysis starts automatically. (You can disable the automatic analysis and choose to launch an analysis explicitly instead.)

To start an analysis, in your IDE, open the project or workspace that you are currently working on, and open a file in the project. Alternatively, copy the following function into a `.c` or `.cpp` file and open the file in your IDE (using a project or otherwise). The function contains bugs such as array access out of bounds, unnecessary code, and use of assignment operator instead of equality.

```
#define MAXBUF 20
int buf[MAXBUF];

int saturateAndShift(int limit, int* stream, int size) {
    int i;
    int numMax = 0;

    if(size > MAXBUF) {
        return -1;
    }

    if(size <= MAXBUF) {
        for(i=0; i<size; i++) {
            if(stream[i] > limit || stream[i] < 0) {
                buf[i+1] = 0;
            }
            else if(stream[i] = limit){
                buf[i+1] = stream[i];
                numMax ++;
            }
            else {
                buf[i+1] = stream[i];
            }
        }
    }
    return numMax;
}
```

After a Polyspace as You Code analysis, you can see the results (bugs and coding standard violations) as source code markers on the currently active file. You can also see the results in a separate list in the IDE. For more information, see:

- “Run Polyspace as You Code in Visual Studio and Review Results”
- “Run Polyspace as You Code in Visual Studio Code and Review Results”
- “Run Polyspace as You Code in Eclipse and Review Results”

You can also export the results on a command line terminal or IDE console. For richer results, you can export the results to a JSON format and manipulate them further before display. For more information, see “Run Polyspace as You Code from Command Line and Export Results”.

Configure Polyspace as You Code IDE Extension

The default analysis is preconfigured to work on small projects in IDEs. In practice, you might have to configure the IDE extension settings further to emulate your build closely, to enable or disable checkers, to see new results only or for other adjustments.

For instance, by default, Polyspace as You Code runs each time you save your code. You can disable the automatic runs using an extension setting (and run the analysis explicitly with right-click options on the source code). For the complete list of extension settings and how to open them, see:

- “Configure Polyspace as You Code Extension in Visual Studio”
- “Configure Polyspace as You Code Extension in Visual Studio Code”
- “Configure Polyspace as You Code Plugin in Eclipse”

The extension settings fall into three major groups:

- *Build options*:

  Using these settings, you specify whether to extract build information from existing artifacts in IDEs such as a Visual Studio solution or a Visual Studio Code build task, or to manually enumerate build-related Polyspace options in an options file. For more information, see “Analyzing Build in Polyspace as You Code”.
- *Checkers*:

  Using these settings, you can enable or disable checkers. For more information, see “Setting Checkers in Polyspace as You Code”.
- *Baselining options*:

  Using these settings, you can connect your Polyspace as You Code installation with a Polyspace Access instance, and baseline your results using a project in Polyspace Access. Baselining allows you to focus only on new results because of recent code changes. See “Baselining in Polyspace as You Code”.

Polyspace Products for Code Analysis and Verification

In this section...

Polyspace products use static analysis to check code for run-time errors, coding standard violations, security vulnerabilities, and other issues:

- A static analysis tool such as Polyspace Code Prover can cover all possible execution paths through a program and track data flow along these paths following certain mathematical rules. The exhaustive control and data flow analysis can complement dynamic testing and expose potential run-time errors that might not be otherwise found in regular robustness testing.
- A static analysis tool such as Polyspace Bug Finder can scan a program for more obvious run-time errors, security vulnerabilities, coding standard violations and other issues that potentially lead to run-time errors or unexpected results.

Using Polyspace Products in Software Development

The Polyspace suite of products supports all phases of a software development process:

- *Prior to code submission*:

  Developers can run the Polyspace desktop or IDE-focused products to check their code during development or right before submission to meet predefined quality goals.

  The products can be integrated into IDEs such as Visual Studio Code, Visual Studio, or Eclipse, or run with scripts during compilation. The analysis results can be reviewed in the IDEs or in the graphical user interface of the desktop products.

  Polyspace provides the following products for desktop usage. These products are meant to run on complete projects or smaller code modules (upto a single source file).

  - **Polyspace Bug Finder** to check code for semantic errors that a compiler cannot detect (such as use of = instead of ==), concurrency issues, security vulnerabilities and other defects in C and C++ source code. The analysis can also detect some run-time errors.
  - **Polyspace Code Prover** to perform a much deeper check and prove absence of overflow, divide-by-zero, out-of-bounds array access and other run-time errors in C and C++ source code.
- *After code submission*:

  The Polyspace server products can run automatically on newly committed code as a build step in a continuous integration process (using tools such as Jenkins). The analysis runs on a server and the results are uploaded to a web interface for collaborative review.

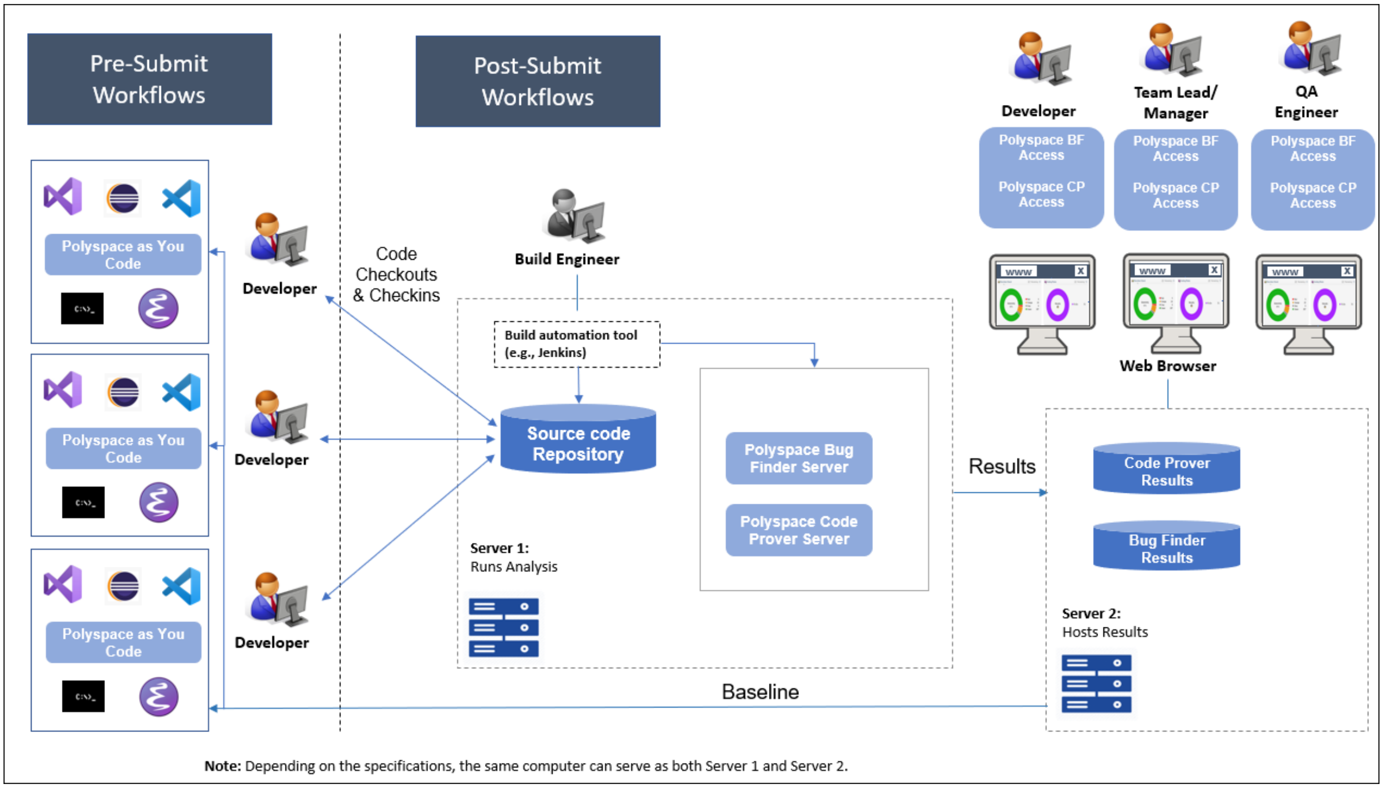
  Polyspace provides these products for server usage:

- **Polyspace Bug Finder Server** to run Bug Finder automatically on a server and upload the results to a web interface for review, and **Polyspace Bug Finder Access** to review the uploaded results.
- **Polyspace Code Prover Server** to run Code Prover automatically on a server and upload the results to a web interface for review, and **Polyspace Code Prover Access** to review the uploaded results.

Typically, Polyspace Bug Finder Server (or Polyspace Code Prover Server) runs on a few build servers and checks newly committed code as part of software build and testing. Each reviewer (developer, quality assurance engineer or development manager) has a Polyspace Bug Finder Access (or Polyspace Code Prover Access) license to review the uploaded analysis results.

In addition, if developers have access to **Polyspace Bug Finder Access** for web review of post-submission results, they can also install **Polyspace as You Code** in their IDEs for pre-submission analysis. When installed as an IDE extension, Polyspace as You Code performs a file-scope Bug Finder-like analysis and provides near-instant feedback to developers while coding.

This diagram shows *one possible deployment* of Polyspace products.



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

When you use both the desktop and server products, your pre-submission workflow can transition smoothly to the post-submission workflow.

Coordinating Pre-Submit and Post-Submit Usage of Polyspace

When you run more than one Polyspace products at separate stages in your software development workflow, the later runs can benefit from the earlier usage, and vice versa. In particular:

- Developers using Polyspace as You Code in their IDEs can easily fix defects and coding standard violations that can be found and resolved within a single file. A later Polyspace Bug Finder Server analysis after code submission no longer shows these issues.
- The results of a Polyspace Bug Finder Server analysis can act as a baseline for Polyspace as You Code runs. Developers using the latest Polyspace Bug Finder Server result as baseline for their IDE runs can focus only on issues that result from their code changes.

Polyspace Products for Ada Code

Polyspace provides these products for verifying Ada code:

- **Polyspace Client™ for Ada** to check Ada code for run-time errors on a desktop.
- **Polyspace Server for Ada** to check Ada code for run-time errors on a server.

You can either use the desktop product to run the analysis on your desktop, or a combination of the desktop and server products to run the analysis on a server. The analysis results are downloaded to your desktop for review.

If you have a Polyspace Code Prover Access license and have set up the web interface of Polyspace Code Prover Access, you can upload each individual Ada result from the Ada desktop products to the web interface for collaborative review.

See also:

- https://www.mathworks.com/products/polyspace-ada.html
- https://www.mathworks.com/products/polyspaceserverada/

See Also

Related Examples

- “Install Polyspace Desktop Products” (Polyspace Bug Finder)
- “Install and Manage Polyspace as You Code in IDEs”
- “Install Polyspace Server and Access Products” (Polyspace Bug Finder Server)