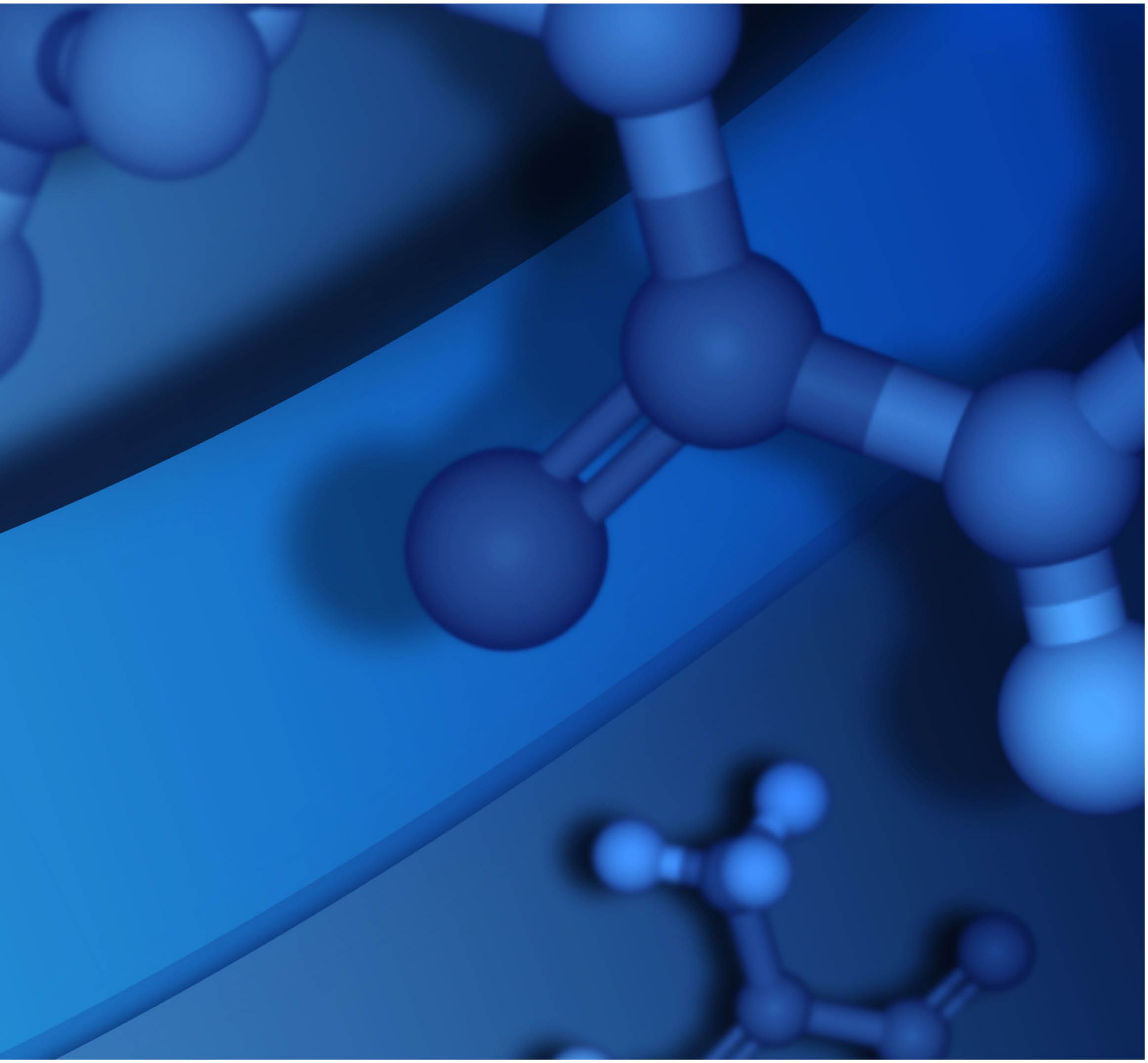
DEPLOYMENT GUIDE

PIPELINE PILOT 2021



Contents

Chapter 1: About this Document

The deployment guide provides technical details on what Pipeline Pilot 2021 is, how it can be deployed, and information used for planning purposes. This document was specifically authored with the IT Architect, IT Administrator, and Pipeline Pilot administrator personas in mind.

IMPORTANT! See the Pipeline Pilot Server Installation Guide and the Foundation Hub Installation Guide for instructions on how to deploy them.

Chapter 2: Overview

Enterprise-based deployment offers increased security, manageability, availability, and scalability. It allows you to include additional servers in your network configuration to improve performance and reduce the single points of failure throughout a system. Pipeline Pilot deployments also include additional features such as the ability to run in load-balanced and reverse-proxy environments, clustering support, and integration with batch queuing systems or grid engines.

An enterprise deployment isolates specific functionality onto separate servers instead of running all services on the same server. This isolation enables different servers to use their resources for more specific purposes, such as running applications, providing database access capabilities, and processing jobs and data.

An enterprise scale deployment means different things to different users. Choosing the best fit for your environment depends on what you want to do. If you want to deploy a set of applications to a large number of users with high availability, load balancing provides a good fit for your organization. If you are supporting a relatively small number of users doing computationally intense work that can be parallelized, clustering is a good solution. If you need to support a larger group of users doing computationally intensive work and want to manage compute resources across a large number of servers, grid integration is the best choice. Many of these options can be combined to scale the platform to fit your requirements.

Pipeline Pilot supports a number of authentication mechanisms and integrates with most enterprise scale user Authentication, Authorization, and Accounting (AAA) Systems, integration with most enterprise scale user management systems. User-level impersonation, which allows job processes to run with the privileges of the authenticated user, is also supported. In some environments, user-level impersonation is desirable for data management, security, and other concerns. Impersonation also requires user accounts to have a minimum level of system access, which can be a security concern. While impersonation is supported across all configurations, it is more applicable in some configurations and less so in others. For load-balanced configurations or high availability applications, impersonation is not recommended as it requires account-level access to the server. For grid integration, impersonation is very useful since grid engines also impersonate. The use of impersonation depends on your policy and security infrastructure.

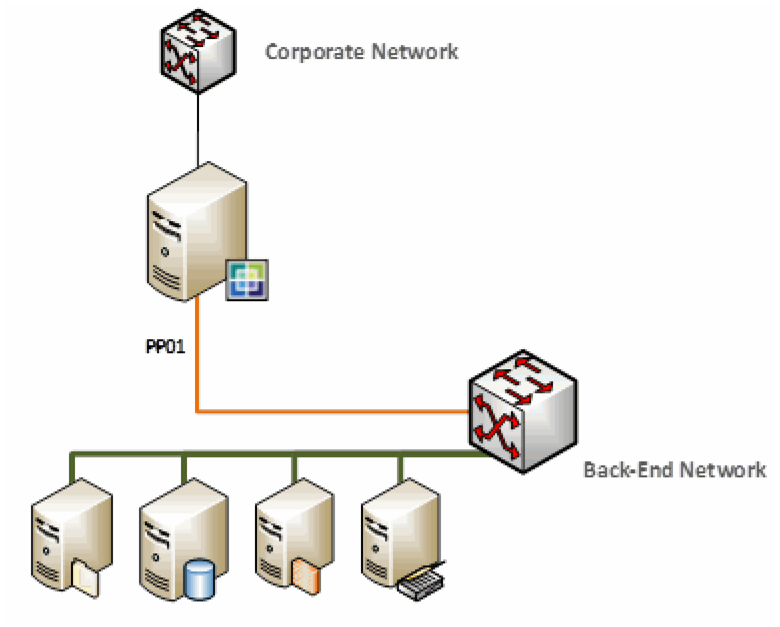
Deployment Options

There are several options for deploying Pipeline Pilot:

- Workstation
- Standalone server
- Web Farm (load balancing)
- Clustering
- Distributed grid computing

Standalone Server

The Pipeline Pilot Server can be configured to run as a standalone server for a single user or workgroup. In this mode, a single server handles all requests. A standalone server can be configured for single port operation. While this deployment can be used for most cases in team collaboration, there are instances where a single server can be quite powerful (many processing cores, large memory, and/or large and fast I/O), and can be used for heavy computing.



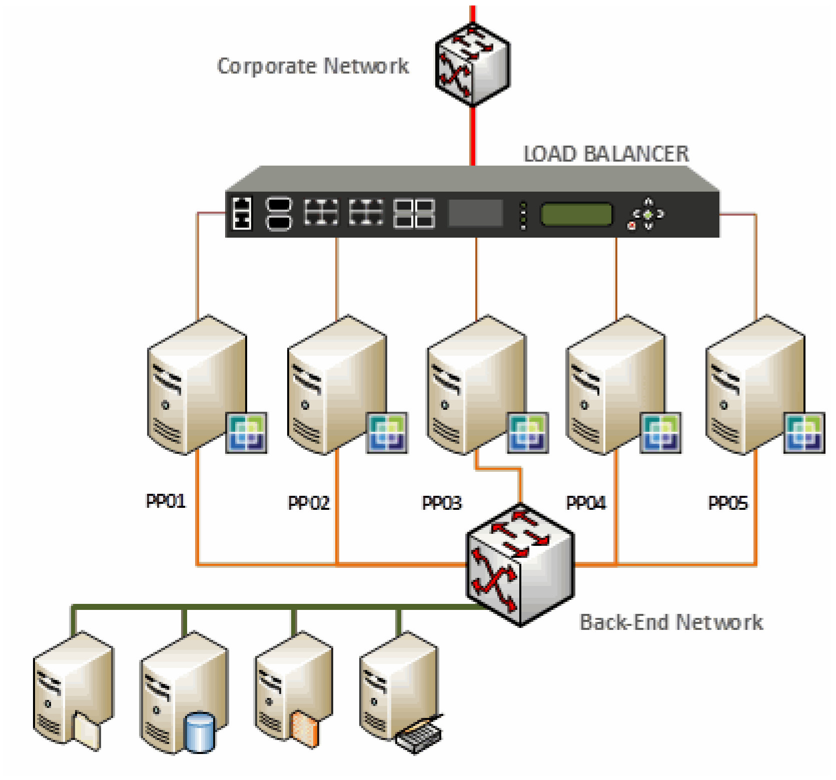
Load Balancing

A "web farm" uses load balancing to distribute incoming HTTP requests across a group of servers. This provides high availability and scalability to handle high concurrent usage. This deployment can be used to deliver services for web applications and web services that are developed for and delivered on Pipeline Pilot. In order to meet the high availability requirements in a load balanced configuration, each Pipeline Pilot Server has an independent copy of the XMLDB, which contains the server configuration. In order to provide a stateless environment for web applications and services, all of the web servers use a common and shared job folder location. Load balancing requires additional administrative involvement to:

- Manage the configuration of the servers to ensure consistent and manageable deployments by using configuration import and export.
- Deploy protocols, components, and applications using package-based publication rather than publishing using Pipeline Pilot Client.

Notes:

- Instructions for load balancing configuration for are available in the *Pipeline Pilot Server Installation Guide*, which is available in the documentation zip file on the Dassault Systèmes Download Platform or in the Pipeline Pilot Help Center on the Administrators tab.
- Load balancing is supported on both Windows and Linux. However, due to the differences in paths, only one type of operating system can be deployed in a load balanced configuration.



Clustering

Note: Clustering is only supported on Linux.

A cluster is a set of loosely connected computers that work together to function like a single system. Each computer in the cluster is known as a node and runs its own operating system. Clusters are usually deployed to improve performance and availability over that of a single computer, while typically being much more cost-effective than single computers of comparable speed or availability.

Clustering allows you to scale your server to take advantage of multiple servers for distributing jobs. The primary server, or head node, acts as a dispatcher to the other servers in the cluster. Each node in the cluster includes a Pipeline Pilot server and shares the configuration with the head node. A key element in clustering is shared storage, typically using the Network File System (NFS).

Clustering Modes of Operation

Clustering has two basic modes of operation:

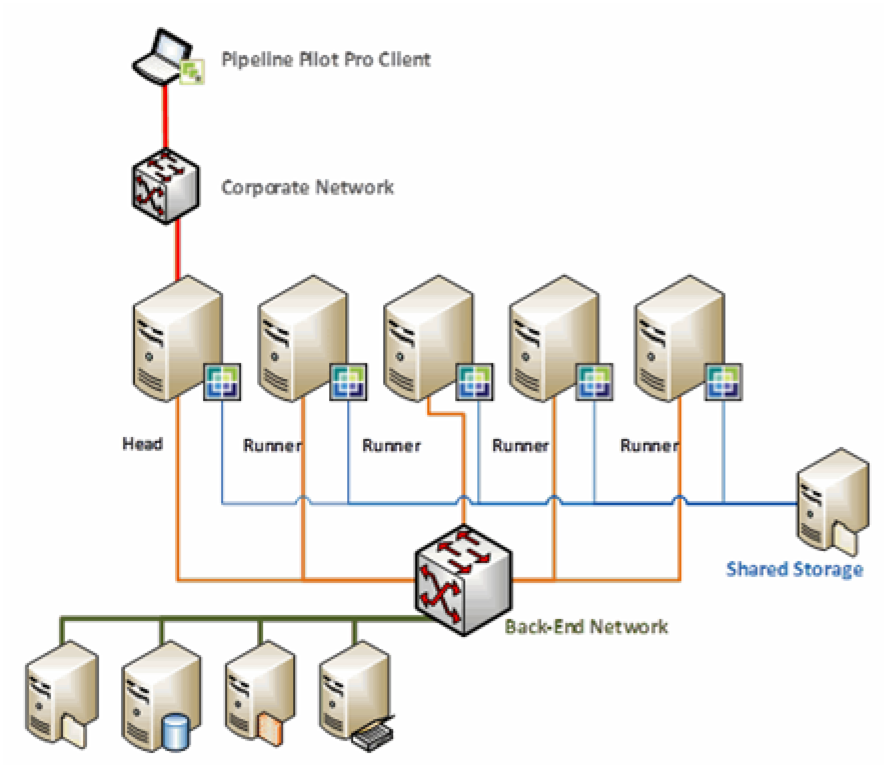
- Private mode - The client connects to the head node and all jobs are distributed to the runner nodes by the head node on a per-job basis. These requests are proxied to the runner nodes by the head node. The client does not connect to the runner nodes. The primary node requires HTTPS access from client machines.
- Public mode - The client initially logs into the head node and is assigned a new runner node for each job. The client contacts the runner nodes directly.

All compute servers have identical access to a shared file system accessible from the same path on all servers.

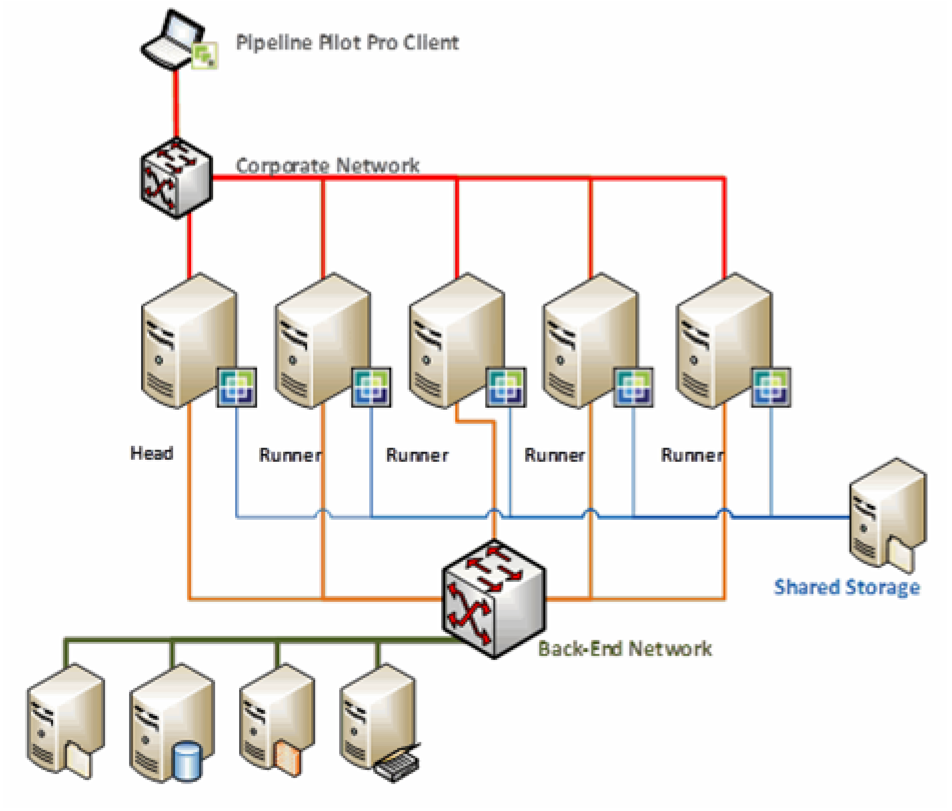
In a cluster of any significant size, the primary node should not be included in the list of job-running nodes, because it cannot delegate a job to itself. The primary node supports various services for the cluster as a whole, including secure login, job delegation, file system browsing, and XMLDB sharing. The execution of compute-intensive jobs on the primary node can compromise the performance of the cluster as a whole, and outweigh the advantages of including an extra computational node for job execution. With a small cluster, the balance favors making this node play its part both as a job runner and as a primary node.

Note: In both modes, jobs do not normally run on the head node or primary server.

Pipeline Pilot Private Cluster



Pipeline Pilot Public Cluster



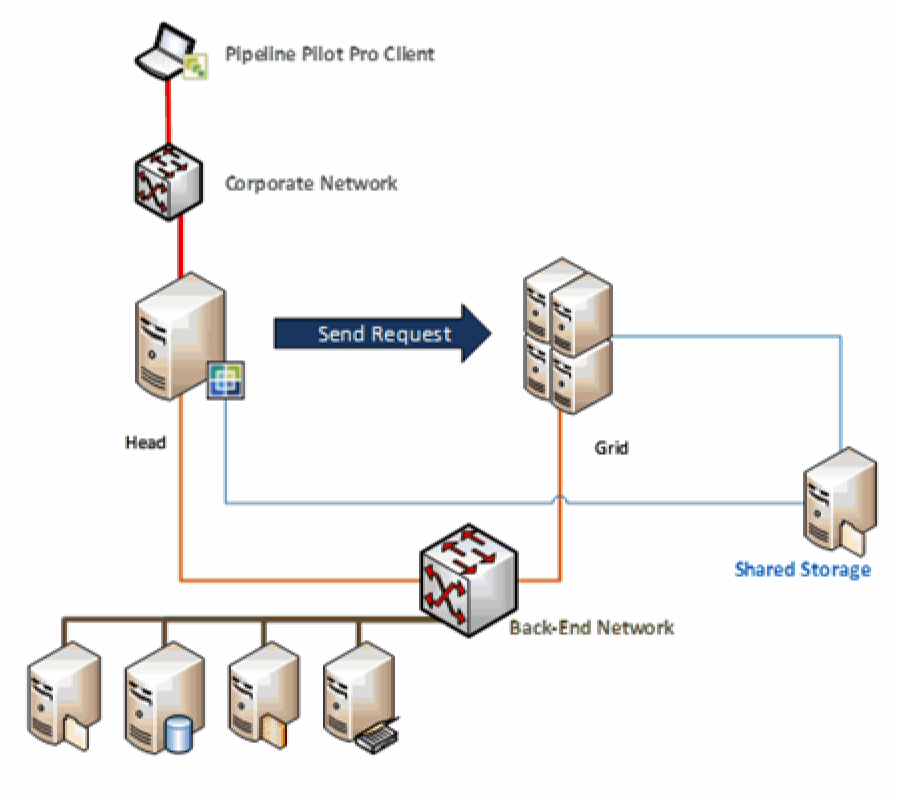
Distributed Grid Computing

Grid computing is a type of distributed computing where CPU resources are shared across a computer network, allowing all connected machines to operate as a computing cluster. You can deploy distributed grid computing technologies such as PBS Professional® and Slurm on the platform.

With distributed grid computing, you can:

- Access extra computing power to work more efficiently
- Queue long-running jobs
- Efficiently distribute protocols over cluster nodes
- Parallelize a protocol and distribute subsets of the data efficiently over cluster nodes
- Take advantage of improved capabilities for computer hardware administration and performance tuning
- Share scheduling with other HPC applications

With grid technology, a large number of jobs can be farmed out to a computing cluster, providing efficient use of enterprise computing resources. You can also queue specific long-running protocols, optimizing the turnaround for critical short-running protocols. Overall, this allows a group of clients to run more protocols with maximum efficiency for each individual job.



Connectivity

Pipeline Pilot provides a number of methods to directly communicate with the server.

Ports

Pipeline Pilot has two common public ports that it listens to and three private ports for inter-process communication. These ports are defined as part of the initial installation. If required, these ports can be modified after installation.

By default, the following are the ports configured for Pipeline Pilot:

- HTTP: 9944
- HTTPS: 9943
- Tomcat Shutdown Port: 9945
- Tomcat HTTP Port: 9946
- Derby Port: 9947

The HTTP and HTTPS port numbers are usually defined during installation. The other three are automatically selected during installation. However, you can explicitly define all port settings. All use of Tomcat specific ports is within the server node, so they should be externally blocked to increase protection.

More details are available within the Admin Portal Help Center at:
Admin > Setup and Configuration > Server Configurations > Reconfiguring Ports

For those deploying in an Internet Based Deployment, Pipeline Pilot supports front end devices such as firewalls, web application firewalls, proxy servers, and load balancers.

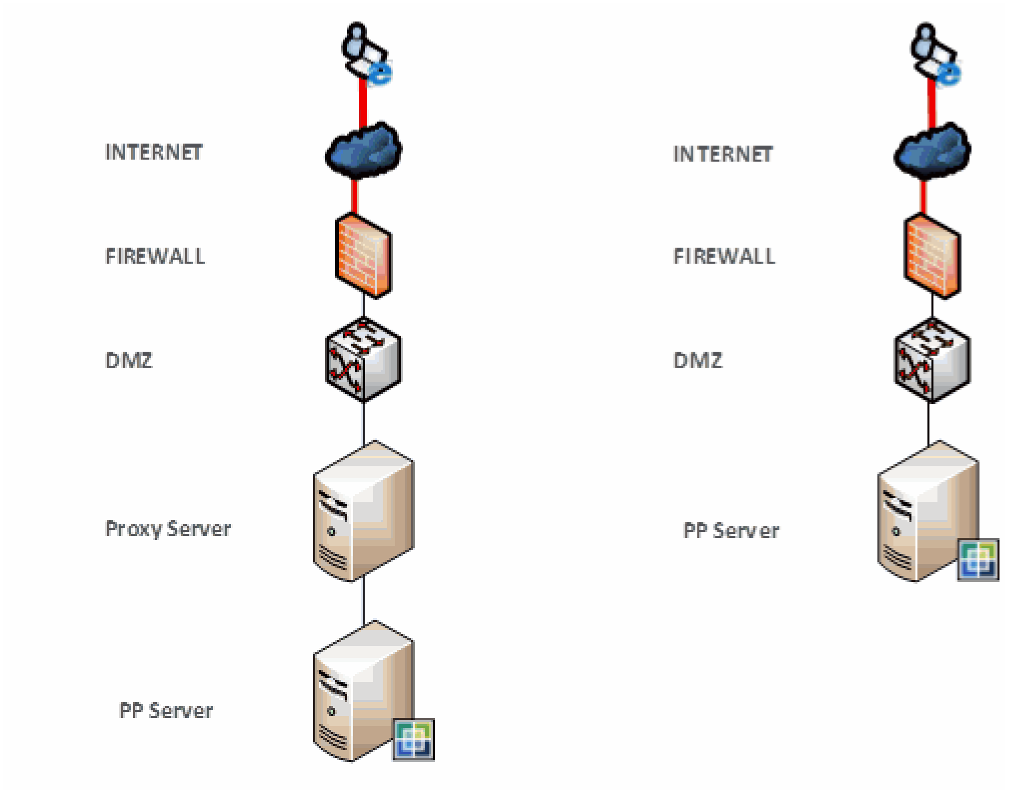
Port Forwarding

Port forwarding is used when the server is placed behind a firewall. Port forwarding is the process of intercepting traffic bound for a certain IP/port combination and redirecting it to a different IP and/or port. This redirection can be accomplished by an application running on the destination host, or it can be performed by intermediate hardware, like a router or firewall.

Reverse Proxy

A reverse proxy server is a proxy on the back-end of the browser web server chain. Instead of sitting between the browser client and internet, it sits between the web server and the internet. It acts as an intermediary for servers that need to offload the handling of the initial HTTP request. A reverse proxy can lower the server workload by terminating SSL connections, which offloads SSL processing from the primary servers, caching common HTML, and responding to or discarding invalid HTTP request. A reverse proxy also provides security by blocking known attacks against web applications. An allowlist can also be configured on the reverse proxy to restrict access to certain areas of the backend Web Application. Placing the reverse proxy in a DMZ adds an additional layer of security by not allowing direct access to the Web Server from the Internet and only allowing network connectivity from the reverse proxy internal interface. The chapter on Security provides more details on this subject.

Note: Reverse proxies and port forwarding firewalls are supported on both Windows and Linux. However, due to the differences in paths, only one type of operating system can be deployed across the enterprise.



Web Port

Note: Internet Explorer blocks cookies from a server if the server name contains characters other than those from the alphabet (A-Z), digits (0-9), minus sign (-), and period (.). Servers named using a character outside this set, such as an underscore character ("_"), may not work with Pipeline Pilot.

Endpoints

Pipeline Pilot provides a number of endpoints that can execute jobs, perform administration, find documentation, make web service calls, etc. The following endpoints list details of ideal usage. Additional details can be found in the Admin Portal Help Center, which is part of the product after it is installed on a given server.

Tip: Use RESTful services when working with Pipeline Pilot to gain provides greater flexibility and control when running protocols deployed on Pipeline Pilot.

`<ServerURI>` is a reference to the URI location of the server. This is typically `http://<hostname>:<port>`.

IMPORTANT! *`<pps_install>`* is the root of the Pipeline Pilot installation.
On Windows this is typically:
`C:\Program Files\BIOVIA\PPS`
On Linux this depends on the location of your Pipeline Pilot installation, but may be:
`[Home]/BIOVIA/PPS`

Endpoint	Location	Usage
Server Home Page	`<ServerURI>/`	
Web Port	`<ServerURI>/webport`	
Admin Portal	`<ServerURI>/clientinstall`	
Client Install	`<ServerURI>/admin`	
Install DS Builder	`<ServerURI>/dsbuilder`	
Install Draw	`<ServerURI>/draw`	
Help Center	`<ServerURI>/help/docs`	
Command Line [1]	`<ServerURI>/bin/RunProtocol.exe` (Windows) `<ServerURI>/linux_bin/runprotocol` (Linux)	
RESTful Launch Job [2]	`<ServerURI>/auth/launchjob`	
SOAP Polling[3]	`<ServerURI>/scitegic/service/async`	
SOAP Blocking Authenticated	`<ServerURI>/auth`	

[1]See the *Command-line Protocols: RunProtocol.exe User Guide*

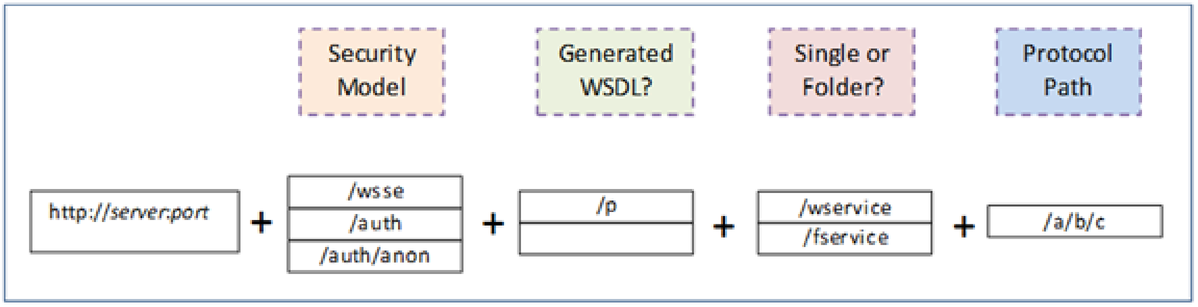
[2]See the *RESTful Web Service Guide*

[3]See the *SOAP Web Services Guide*

Endpoint	Location	Usage
SOAP Blocking Anonymous	`<ServerURI>/auth/anon`	
SOAP Blocking	`<ServerURI>/wsse`	
Query Service	`<ServerURI>/auth/queryservice`	
XMLDB endpoint	`<ServerURI>/scitegic/xmldb`	

The referenced guides are available in the Pipeline Pilot Help Center at:
Developers > Client-side Integration

When connecting to Pipeline Pilot using a URI, the following diagram shows the pattern of usage within the URI.



Folders

Pipeline Pilot makes extensive usage of file system folders to store system configuration and data. While Pipeline Pilot works out-of-the-box, the administrator should explicitly define what their policy and security definitions are.

The following table describes the default folders, global property that protocol authors use to reference a given folder (if applicable), and their associated usage.

IMPORTANT! *<pps_install>* is the root of the Pipeline Pilot installation.
On Windows this is typically:
`C:\Program Files\BIOVIA\PPS`
On Linux this depends on the location of your Pipeline Pilot installation, but may be:
`[Home]/BIOVIA/PPS`

Location	Global Property	Usage
<pps_install>/apps		Applications deployed through a Pipeline Pilot Package. Also a web location that is accessible through a URI.
<pps_install>/bin		Core libraries and utility programs
<pps_install>/config		Configuration files for some Pipeline Pilot-related services
<pps_ install>/database/solr *<pps_*		Used by Catalog indexing service Used by Insight Used by GEMS database in Derby

Location	Global Property	Usage
install>/database/Insight <*pps_ install*>/database/GEMS		
<*pps_install*>/install		Installation support files
<*pps_install*>/licensing		Licensing files
<*pps_install*>/logs		Installation and usage logs for a given deployment.
<*pps_install*>/public	SharedPublicDir	
<*pps_ install*>/public/appcatalog		Used by the Catalog indexing service
<*pps_ install*> /public/chartingservice		Used to store temporary files created while interacting with charts generated using the Reporting collection
<*pps_ install*>/public/server		Manages details on other defined servers
<*pps_install*>/public/users	UserDir	Stores protocols and data related to a given user
<*pps_install*>/temp	TempDir	Temporary location with multiple files
<*pps_install*>/web		Location of the web server configuration and parent folder of publicly accessible web content folders, such as `/apps` and `/webport`
<*pps_ install*> /web/jobs/ <*username*>/<*jobid*>	JobDir	Location of job folder which is dynamically created at runtime for a specific <*username*> and <`jobid`>
<*pps_install*>/xmldb		The XMLDB directory stores protocols and components specific to the server. Some of these are public (Components and Protocols subfolders), and some are owned by specific users (Users subfolders). Besides a backup strategy for this data, you can relocate the directory to a more appropriate location for these valuable resources.

Software Development Tools

While many interactions with Pipeline Pilot can be done using web services as listed in the previous section, there are additional tools available for Software Tools to further abstract the interactions with Pipeline Pilot.

Java, .NET, and JavaScript can talk to dedicated APIs, provided as part of the client side SDKs, which abstract the internal web services layer.

Tip: BIOVIA recommends that you utilize the web service methods in Pipeline Pilot rather than the SDK for greater performance and control when interacting with Pipeline Pilot.

IMPORTANT! *`<pps_install>`* is the root of the Pipeline Pilot installation.
On Windows this is typically:
`C:\Program Files\BIOVIA\PPS`
On Linux this depends on the location of your Pipeline Pilot installation, but may be:
`[Home]/BIOVIA/PPS`

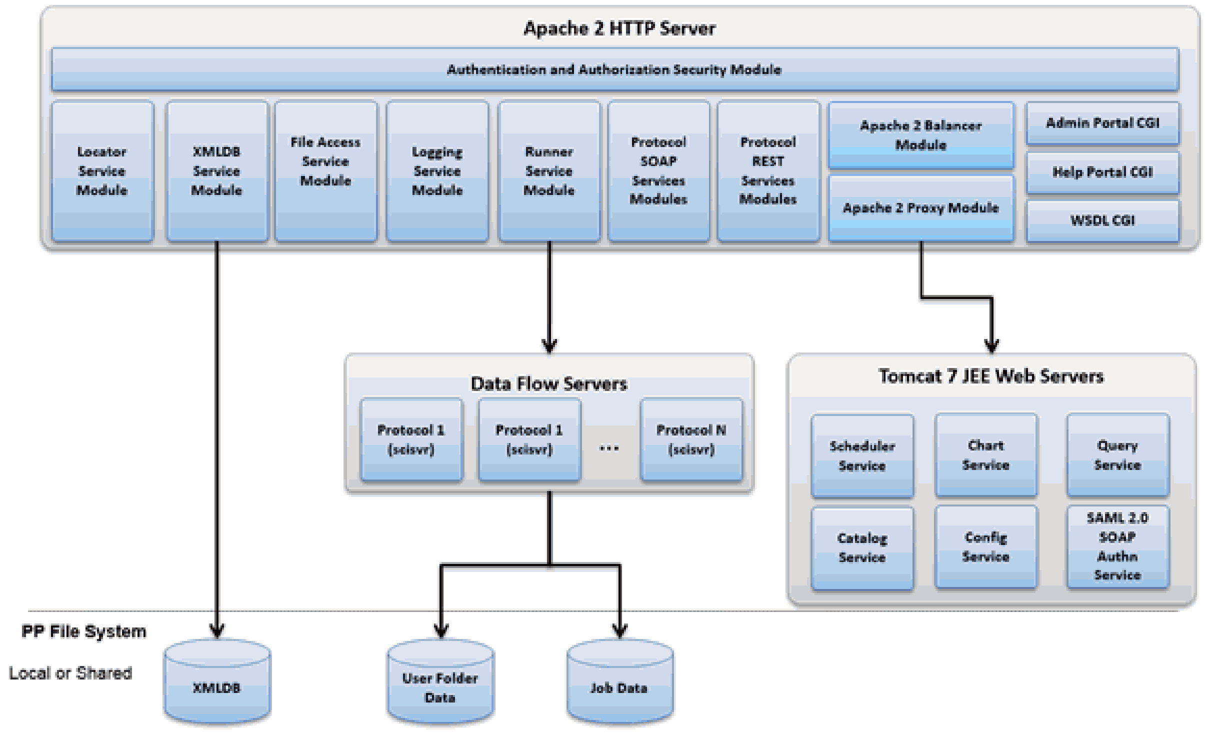
SDK	**Location**
.NET Client SDK	`<pps_install>/apps/scitegic/clientsdkdotnet`
Java Client SDK	`<pps_install>/apps/scitegic/clientsdkjava`
JavaScript SDK	`<pps_install>/apps/scitegic/clientsdkjavascript`

Chapter 3: Architecture

Pipeline Pilot provides several ways to publish and access HTTP-based web services. Both SOAP and RESTful web services can be accessed through various client channels. For example, Pipeline Pilot provides language-specific client SDKs, protocol-specific SOAP services and RESTful web services.

Pipeline Pilot uses HTTP-based web services to provide modular, portable communication between the server and the various client types. These web services run on an Apache 2 HTTP server, supported on both Windows and Linux platforms.

The Pipeline Pilot installer automatically installs, configures, and starts a Tomcat service for you. Pipeline Pilot also includes an embedded Apache Tomcat server (http://tomcat.apache.org/). Tomcat must be installed and running to support core services such as task scheduling, fast chart updates, and Query Service.



Apache

Apache 2 HTTP Server (Apache) acts as a gateway to all back-end web services. Many web services are deployed directly within Apache 2 module-based system. Module-based services support requests such as user logins, locating all available web services, navigating file systems on the server, accessing the XMLDB and running protocols. However, Apache also acts as a front-end web proxy to services hosted in other containers such as CGIs, J2EE web applications, and protocol-based services running within data flow servers. The value of Apache is that it provides a single point of access and control for all incoming requests.

Apache handles authentication and authorization for each client request, confirming the identity of the requesting user and their permission to access the back-end service or resource. Apache then routes incoming requests to the correct service container and balances these resources across all client connections. Apache can balance requests across multiple server nodes in load-balanced and clustered configurations.

Apache web server acts as a front-end filter to all Tomcat requests. Apache validates the security of all requests through the Pipeline Pilot security model.

Considerations for Deployments on Microsoft Windows

On Windows, Apache is installed as a Windows service that restarts automatically upon reboot. Windows services run, by default, as the Local System User who has limited network and domain access rights (for example, the Local System User cannot access UNC paths on many networks). Although this setup is appropriate for many services, BIOVIA recommends that you run the service under another user who has administrative rights on the server machine. If you are not planning to use impersonation for running protocols, ensure that the Apache user has appropriate network rights for performing all the tasks that the user protocols require.

Considerations for Deployments on Linux

On Linux, Apache can be configured to automatically start when the server reboots. If you are not using impersonation, all resource access rights for running protocols are based on the Apache user account specified during installation. Ensure that the Apache user has appropriate rights for carrying out all the tasks that the user protocols require (such as access to mounted drives).

Java, .NET, and JavaScript can talk to dedicated APIs, provided as part of the client-side SDKs, which abstract the internal web services layer.

Best Practice for Apache Configuration

If you have a large server, it is reasonable to increase the thread count so you do not have to deal with lock-up or blocking job timeouts. If you increase the thread count, you can also increase the blocking job timeout.

Tomcat

Pipeline Pilot is tested and tuned to work with specific applications and services. BIOVIA recommends that you do not deploy custom or third-party applications to the Tomcat server that is embedded on the platform.

More information on the Tomcat server can be found in the Pipeline Pilot Help Center at:
Admin > Introduction > Web Services Overview

Pipeline Pilot Manager

Since Pipeline Pilot leverages both Apache and Tomcat, BIOVIA Pipeline Pilot 2021 (Manager) ensures that both of these services are running properly.

More information on the Pipeline Pilot Manager can be found in the Pipeline PilotHelp Center at:
Admin > Pipeline Pilot Services > Pipeline Pilot Manager > Managing Services with Pipeline Pilot

BIOVIA Query Service

Traditional scientific database systems load data into individual schemas, each system having its own client application. As the availability of external data grows, or companies merge systems, the number of database systems and platforms multiplies, and more systems are introduced to handle the increase in data. The complexity of the infrastructure increases, and data becomes difficult to manage and access across multiple systems and platforms.

The Query Service:

- Solves the problem of searching across multiple data sources, by mapping the metadata of each data source to descriptive field aliases. Queries are built using these field aliases, and can be mapped onto any data source for which a corresponding mapping is provided. This method eliminates the need to migrate data or to modify database schemas, and provides the flexibility to modify data sources.
- Queries are written using the Unified Query Language (UQL). UQL has a similar syntax to the widely used Structured Query Language (SQL). An XML query format is also available.
- Is compatible with Oracle databases.
- Is installed with the Pipeline Pilot. Within Pipeline Pilot, the Query Service is managed by the Apache Tomcat servlet container.

More information on the Query Service can be found in the Pipeline Pilot Help Center at:
Admin > Reference Guides > Collections > Platform > Administration Guide: Query Service

Query Service RESTful API

The Query Service exposes a RESTful API that client applications can use to perform the following tasks:

- Get the metadata of a data source to determine what data can be queried
- Execute cataloged (that is, stored) and ad hoc queries
- Combine recordsets (that is, sets of full records) or search results

The Query Service can provide responses in various formats, including XML (the default format), XDfile, SDfile, JavaScript Object Notation (JSON), and several character-delimited formats. For more details about available response formats, see the Query Service Developer’s Guide.

More information on the Query Service can be found in the Pipeline Pilot Help Center at:
Developers > Collections > Platform > Developer’s Guide: Query Service

MongoDB

MongoDB is an open source document-oriented database system that is part of the NoSQL family of database systems. Some BIOVIA applications, such as Insight, utilize MongoDB. As part of Pipeline Pilot, a set of prototype components are available which can take advantage of the MongoDB capabilities. These prototype components are available in the folder:

`Components > Data Access and Manipulation > Utilities > Prototypes > MongoDB`

MongoDB can be configured to work with Pipeline Pilot in the following ways:

- Enterprise and medium size deployments - For enterprise and medium size deployments with over five simultaneous users, MongoDB must be installed on a separate server from Pipeline Pilot. This is because MongoDB uses large amounts of RAM, and thus competes for system resources with other Pipeline Pilot processes.
- Small or single-user deployments - For small deployments with 1-5 users on servers with sufficient RAM (8GB or more), MongoDB can be installed on the same server as Pipeline Pilot. MongoDB binaries are located in `<install_root>/apps/scitegic/mongodb/bin/<OS_VERSION>/bin`.

More information on the MongoDB deployment can be found in the Pipeline Pilot Help Center at:
Admin > Server Maintenance > MongoDB > Installing and Deploying MongoDB

Chapter 4: Hardware Guidance

While planning for your new deployment of or upgrade to Pipeline Pilot, there are areas around hardware selection and configuration that should be considered. While each deployment is different and dependent on your specific business and technical requirements, you should use the following details as guidance.

Processor Guidance

As noted in the System Requirements, Pipeline Pilot requires an x64 processing architecture (Intel or AMD based). The system requirements specify 1 processing core per 2 simultaneous jobs. This standard guidance is based using Pipeline Pilot as a collaboration server. For more information, see the *Pipeline Pilot System Requirements* document.

Processors that support hyper-threading can be leveraged by Pipeline Pilot. Operating systems make the hyper-threaded cores appear as regular cores, and Pipeline Pilot uses them as if they were physical cores. BIOVIA has not seen any degradation of performance when hyper-threading. Hyper-threading is supported.

Memory Guidance

Pipeline Pilot is architected to be extremely efficient with memory usage. As data moves along the data pipeline (as defined with a Protocol), Pipeline Pilot is working with a single data record as it passes it along. This keeps memory extremely efficient. However, there are instances where certain protocol designs and component usage can impact the usage of performance.

An example of products and collections by BIOVIA that are typically memory intensive are:

- Next Generation Sequencing collection
- BIOVIA Discovery Studio
- BIOVIA Material Studio
- Imaging

Also, the following component types within Pipeline Pilot also have the ability to consume large amounts of memory, but this is dependent on how much data is flowing through the pipeline:

- Aggregators (such as Sort, Group, or Merge Data)
- Readers

Operating System Guidance

Pipeline Pilot works on both Windows and Linux, and many of protocols and components also work on both operating systems. However, there are a few capabilities that are limited to one operating system. Some of these restrictions are due to the underlying technologies being utilized, such as the Microsoft .NET Framework which is not available on Linux. The limitations are:

- Windows only: VB Script (on server), .NET, Python, ISIS
- Linux only: Cluster, Grid, and NGS deployments

Server Virtualization Guidance

Server virtualization can offer many benefits, such as consolidating systems, workloads, and operating environments. Scenarios in which hosting Pipeline Pilot on a virtualized server might be desirable include:

- Deploying test/development servers and installing new Pipeline Pilot versions without jeopardizing production workloads
- Deploying Pipeline Pilot on multiple operating systems using one physical server

Note: Some BIOVIA products utilizing Pipeline Pilot as the core platform, such as Discovery Studio and Materials Studio are not currently supported in virtualized environments.

When creating a virtual server for deploying Pipeline Pilot, the following information must be considered:

- The virtualization hypervisor must be fully supported by its vendor.
- The virtual server must meet or exceed the specifications as stated in the *Pipeline Pilot System Requirements* document.
- The underlying physical hardware must have sufficient resources (that is, CPU, RAM, etc.) to manage the virtual server as well as any other processes such as the virtualization hypervisor that are also running on it.
- For a scenario where multiple virtual servers are managed through the same virtualization hypervisor, physical server resources must be statically allocated to the virtual server(s) hosting Pipeline Pilot and any associated database instances. This should include:
  - Dedicate physical cores to the virtual server
  - Dedicate physical memory to that virtual server
  - Dedicate a separate network adapter for that virtual server
  - Dedicate a separate disk volume or RAID array for that virtual server
  - Dedicate a separate network adapter for network attached storage (for example, ISCSI) or use a high performance I/O connection (for example, Fibre Channel)

Optimizing Grid Guidance

You can optimize your grid hardware configuration to process jobs on the grid. To alleviate memory load from the head node on the server, consider any available node on the grid as a potential head node for the protocol job. Your runner node should have the same privileges as the head node.

Chapter 5: Planning Your Server Deployment

Note: For details about hardware and software requirements for the platform, see the *Pipeline Pilot System Requirements* document.

Users access Pipeline Pilot with a variety of applications including Pipeline Pilot Web Port and the Pipeline Pilot Client. This is the typical case for Pipeline Pilot deployed as a personal productivity or work group server.

The goal of performance tuning is to balance the running time of protocol jobs versus allowing multiple users to effectively access server resources.

Other items to consider as part of your planning:

Items to consider	Done
The implementation of Windows versus Linux.	☐
Folder locations and impact on performance.	☐
Pool usage and performance tuning. See Tuning Server Settings for more information.	☐
Apache impersonation setting. This might require a new local or domain user account to be used.	☐
Security settings. See Security Considerations on page 28.	☐
Access to operating system resources: File permissions, ODBC settings. For more details, see the *Pipeline Pilot Server Installation Guide*.	☐

Load Balancing

Web requests tend to have very short run times with high frequency and they commonly access the same resources multiple times over. In this profile, most users access server resources through deployed web applications such as Insight, Biological or Chemical Registration or using the web services API. Pipeline Pilot Client use is infrequent and perhaps not permitted by security policy.

- Performance tuning in this case focuses on maximizing server throughput in terms of requests per second.

With all of the availability and concurrency advantages that come with a load balancing, there are some disadvantages:

- Pipeline Pilot authoring directly against a load balanced configuration is not supported.
- Model building protocols in a on a Pipeline Pilot Server that is part of a load balanced configuration is not supported as they write to the XMLDB. However, you can build using a Model Building protocol on a development or staging server and deploy that model to load balanced Pipeline Pilot Servers using a package.
- Parallel subprotocols are NOT distributed to multiple nodes as with cluster or grid deployments.

Other items to consider as part of your planning:

Items to consider	Done
Requires a shared storage.	☐
All web server nodes must be the same operating system.	☐
All web server nodes must have the same folder structure and configuration.	☐
Applications, protocols, and components need to be distributed across the nodes using packages rather than the XMLDB.	☐
Windows-based deployment requires disabling directory caching on the shared volume.	☐
Protocols should use the Shared Public Directory (defined on the **Setup > Folder Locations** page of the Pipeline Pilot Administration Portal) when referencing.	☐

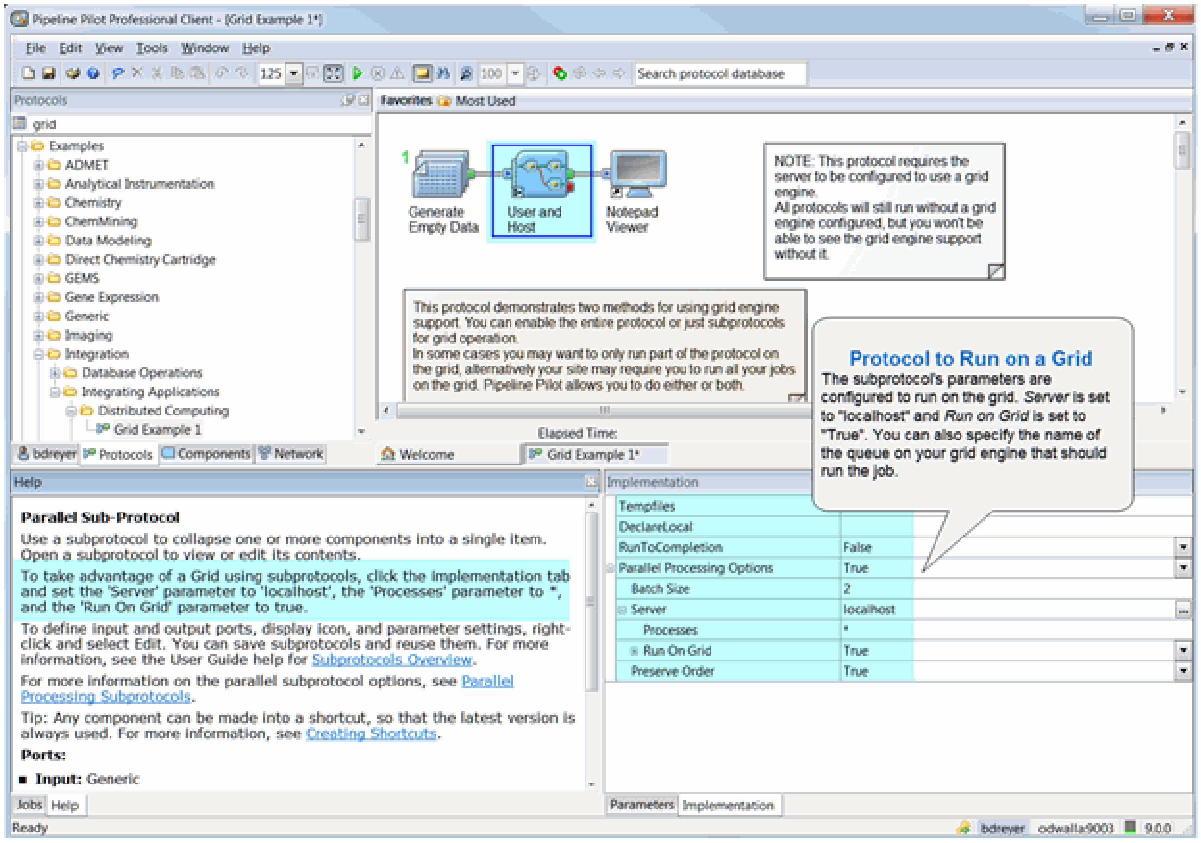
Parallel Computing on Distributed Deployments

Most server access comes from clients such as Discovery Studio, Pipeline Pilot Client, or batch jobs. In this case, a few jobs typically run for a long time and many tasks may queue. Parallel subprotocols are often used to reduce the running time of jobs by parallelizing tasks. The scheduling of jobs is often delegated to an external scheduler system. Performance tuning focuses on maximizing server performance in terms of data-record or process throughput.

With parallel computing, you can efficiently distribute individual jobs to cluster nodes, making it possible to tackle extremely large data sets that were previously off limits, enhancing the client parallel processing capabilities.

You can run any subprotocol in parallel on a specified set of remote servers and specify data record batch size, remote hosts, and number of simultaneous jobs. Optionally, you can run these parallel subprotocols under a distributed grid computing engine, giving you more options for job management and load balancing. You can also run normal protocols under the grid engine.

Note: Distributed grid computing is only supported on Linux.



NOTE: This protocol requires the server to be configured to use a grid engine.
All protocols will still run without a grid engine configured, but you won't be able to see the grid engine support without it.

This protocol demonstrates two methods for using grid engine support. You can enable the entire protocol or just subprotocols for grid operation.
In some cases you may want to only run part of the protocol on the grid, alternatively your site may require you to run all your jobs on the grid. Pipeline Pilot allows you to do either or both.

Protocol to Run on a Grid
The subprotocol's parameters are configured to run on the grid. *Server* is set to "localhost" and *Run on Grid* is set to "True". You can also specify the name of the queue on your grid engine that should run the job.

Parallel Sub-Protocol
Use a subprotocol to collapse one or more components into a single item. Open a subprotocol to view or edit its contents.
To take advantage of a Grid using subprotocols, click the implementation tab and set the 'Server' parameter to 'localhost', the 'Processes' parameter to *, and the 'Run On Grid' parameter to true.
To define input and output ports, display icon, and parameter settings, right-click and select Edit. You can save subprotocols and reuse them. For more information, see the User Guide help for Subprotocols Overview.
For more information on the parallel subprotocol options, see Parallel Processing Subprotocols.
Tip: Any component can be made into a shortcut, so that the latest version is always used. For more information, see Creating Shortcuts.
Ports:
- **Input:** Generic

Tempfiles	
DeclareLocal	
RunToCompletion	False
Parallel Processing Options	True
Batch Size	2
Server	localhost
Processes	*
Run On Grid	True
Preserve Order	True

Grid example protocol for Pipeline Pilot. The protocol "User and Host" has two implementations that allow the protocol to run on a grid. Server is set to "localhost" and "Run on Grid" is set to true.

Grid Engines

A grid installation is virtually identical to a cluster installation. The only real difference is that instead of running Pipeline Pilot on the cluster nodes, the grid engine software manages the cluster nodes.

Clusters are well suited to mixed use cases (some ad-hoc development mixed with computation), whereas, grid usage is really ideal for large-scale resource management. This is due to the performance differences with short running jobs.

Installation and Requirements

See the *Pipeline Pilot Installation and Configuration Guide* and *Pipeline Pilot System Requirements*.

Server Performance Tuning

As a general platform for scientific informatics, Pipeline Pilot supports a wide variety of uses cases. Proper performance tuning depends heavily on the typical usage your server must handle.

Usage Profiles

- General Profile - Users access Pipeline Pilot with a variety of applications including Pipeline Pilot Client and Web Port. This is typical for personal productivity or work group settings. The goal of performance tuning is to balance the running time of protocol jobs versus allowing many users to use the system effectively.

- Web Profile - Pipeline Pilot is more frequently or exclusively accessed by web applications and services. Web requests usually have short running times but occur frequently and commonly access the same resources, like database connections. Example web applications include Database Search, Biological or Chemical Registration or SDK clients. Direct access to Pipeline Pilot through the server is infrequent and usually only permitted for administration. Performance tuning in this case focuses on maximizing server throughput in terms of requests per second.
- HPC Profile - Most access to Pipeline Pilot comes in from clients like Discovery Studio, Pipeline Pilot Client, or batch jobs. In this case, a few jobs typically run for a long time and many tasks can enter the queue. Parallel sub-protocols often reduce the running time of jobs by parallelizing tasks across several processes. Performance tuning focuses on maximizing server performance in terms of data-record or process throughput.

Job Modes

Pipeline Pilot has two basic job modes for running jobs:

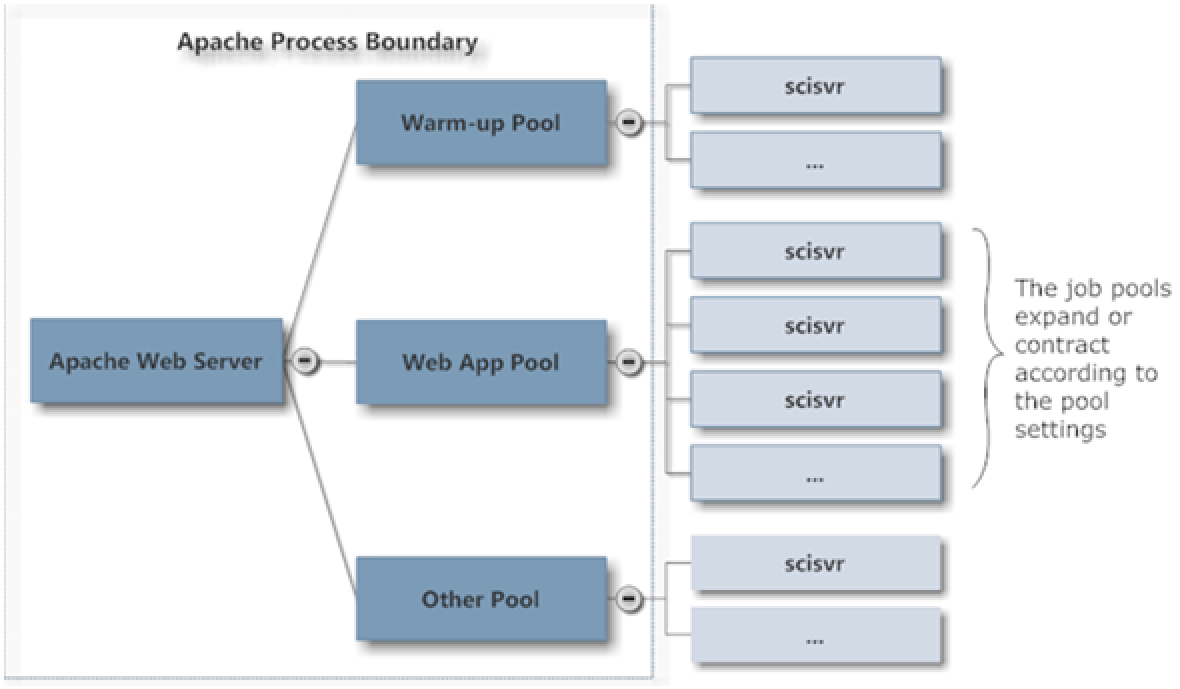
- Polling - The client connects to the head node, creates a job, launches the job and then polls for status. Polling is synonymous with asynchronous execution, which is the default mode for all jobs launched from the Pipeline Pilot Client.
- Blocking - The client connects to server and makes a request. The server responds with the final job status and results only when the job completes or times out. In this case, the server "blocks" until the job execution is finished. Blocking is synonymous with synchronous execution. Clients, such as Web Port or SDK clients, can choose to launch jobs in blocking mode by making appropriate calls. Blocking jobs are the same as typical web server requests.

The following table summarizes each profile in terms of the user load and jobs types:

Feature / Profile	Client Examples & Usage	Job Types	Concurrent Users	Job Running Time
General	■ Pipeline Pilot Client ■ Web Port	Mostly polling jobs	< 10	< 30 seconds
Web	■ Database Search ■ Web port ■ Biological Registratio n ■ Chemical Registratio n ■ SDK Clients	Mostly blocking jobs	10 or more	< 5 seconds
HPC	■ Pipeline Pilot Client ■ Discovery Studio ■ Batch Jobs	Mostly polling jobs	< 5	Mostly polling

Job Pools

Pipeline Pilot can reuse the `scisvr` process daemons in named pools. Job pools apply to both polling and blocking jobs, but their performance impact is much clearer for blocking jobs with fast running times. While Pipeline Pilot manages these named pools for you, background information is helpful to understand performance tuning.



Pooled process daemons can cache resources to reduce latency and increase the server scalability. Named pools help associate process daemons with expensive resources like the Java Virtual Machine, database connections, and even metadata.

Pipeline Pilot automatically expands and contracts process pools to accommodate the load and to keep the server operating at peak efficiency. Job pools start out with a minimum size and expand as needed to handle more concurrent job requests to each named pool. Conversely, process pools also contract under several possible scenarios:

- A process daemon consumes too much memory
- A process daemon idles for too long
- The server's total physical memory becomes low

Built-in Job Pools

Each server can have many different job pools setup and active at the same time. Some job pools run with pre-defined configurations such as:

- Warm-up Pool - The "warm-up" pool prepares a daemon processes to handle one incoming request. Each process handles exactly one request and then exits. This decreases the latency for jobs that do not specify a specific job pool – such as jobs launched from the Pipeline Pilot Client application.
- Keep-warm Pool - The "keep-warm" pool runs scheduled tasks. The sole purpose of these tasks is to keep the server binaries loaded into operating system memory and page cache. This improves process start-up times on systems that see long idle periods or support a mixture of different programs.
- Default Pool - This pool simply defines the default settings for named pools that might be created at runtime.

Job Pools and Impersonation

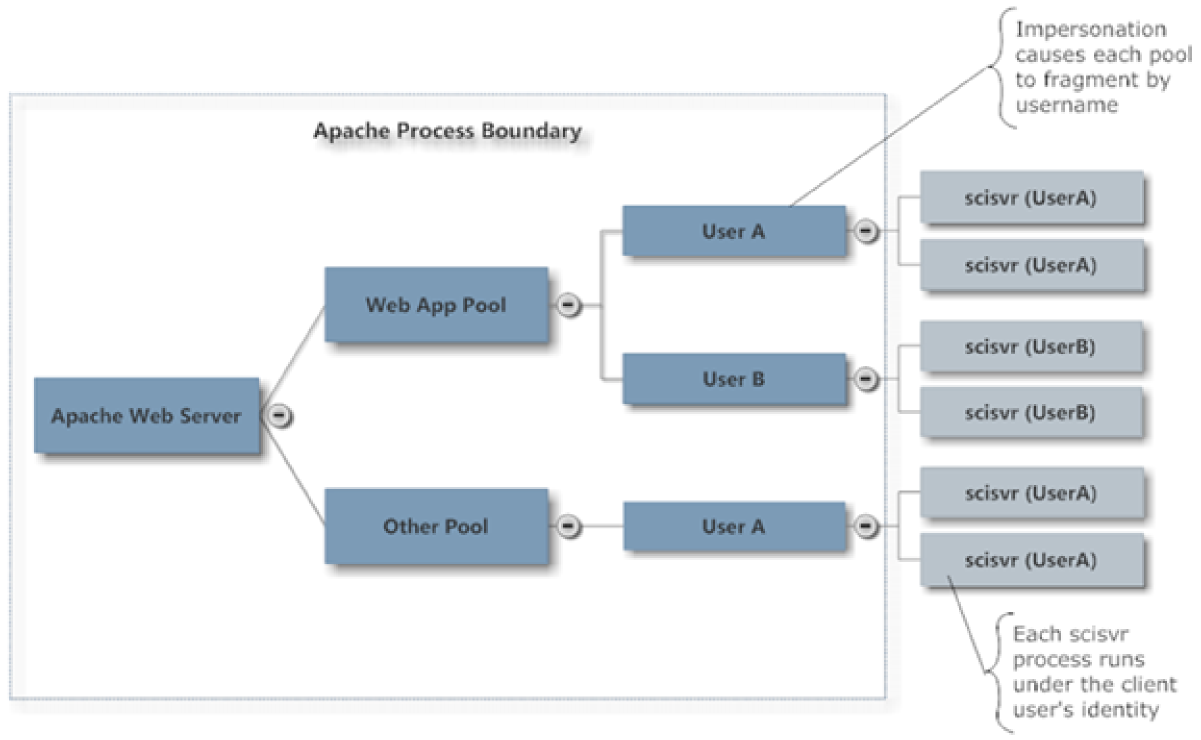
When Pipeline Pilot runs with authentication but not impersonation, pooled scisvr processes run under the Apache web server user account. In this case, Pipeline Pilot efficiently shares job pools across multiple client users. Impersonation is useful for client-user authorization in the general and HPC profiles. However, impersonation adversely affects scalability in the web profile.

With impersonation enabled, each pooled process runs under identity of the client user. The net effect is that the pool settings apply on a per-user basis. Job pools cannot scale as well under impersonation because unique client users must have their own unique processes. This reduces the effectiveness of resource caching and can consume extra system resources, like memory.

Tips:

- If possible, turn-off impersonation when you deploy Pipeline Pilot in a web profile environment.
- If you must run with impersonation on, divide the pool configuration recommendations given below by the number of expected concurrent users. If you do not know the number of concurrent users, start with a pool size of 1 - 2 and adjust upward if needed.
- The currently running job pools are listed at:

```
http://<server><port>/scitegic/managepools?action=debug
```



Tuning Server Settings

Several server parameters are available for tuning on Pipeline Pilot servers. You can find a simple set of tuning parameters in the Pipeline Pilot Help Center at:
Admin > Setup and Configuration > Jobs Settings.

In most cases, the default settings are sufficient. You can also perform more advanced tuning by editing the pool configuration files in the Apache web service configuration.

Admin Portal Job Settings

The job settings defined in the Admin Portal act as global defaults for all daemon pools and warm-up processes. The default settings installed out-of-thSe-box are tuned for a general profile server running on a system with eight logical cores. Tune these settings to fit your server and your target usage profile.

Setting	Description	General Profile	Web Profile	HPC Profile
Running Job Limit	Maximum number of polling jobs that can simultaneously run on the server. Excess jobs are placed in a job queue. Default setting is 10 jobs. To disable the limit on running jobs, set to 0 or less.	Set this value between 1 - 2 times the number of logical cores.	Set this value to 2 - 4 times the number of logical cores.	Set this value between 1 - 2 times the number of logical cores.
Blocking Job Timeout	Length of time (in seconds) before blocking jobs time out and return an error. The default is 10 seconds.	Use the default value (10).	Set this value between 30 and 60 seconds. Web clients will receive an error message when jobs time out.	Use the default value (10).
Job Priority Switching	When enabled, job processes are downgraded to a lower priority after 10 seconds. This feature is enabled by default. Normally, this setting does not affect performance significantly.	Use the default value (On).	Use the default value (On).	Use the default value (On).
Maximum Number of Simultaneous Parallel Processing Subprotocol Jobs	Maximum number of simultaneous jobs that can be spawned to support each individual parallel processing subprotocol. When the server hosts a parallel subprotocol operation, this setting is used by the running protocol to handle x-number of subprotocol jobs for a specific subprotocol. The default number of jobs is 4. To disable this feature, set the value to 0.	Use the default value (4) or less to provide more even load balancing across users.	Set this value to 0 or 1 to provide better load balancing across users.	Use the default value (4) or more depending on your specific needs.

Setting	Description	General Profile	Web Profile	HPC Profile
Maximum Number of Persistent Daemons per Job Pool	Maximum number of persistent daemons associated with a particular job pool. A single job or a set of related jobs can use daemons from the same job pool. After a pooled job finishes running, a daemon remains resident in memory for a configurable period of time waiting for new job requests. Running a job in a daemon is much faster because it skips initialization of many internal data structures. When the server is using a high amount of memory, no new daemons are created. The default maximum number of persistent daemons per job pool is 16. No daemons are created when the setting is set to 0.	Set this value between 1 - 2 times the number of logical cores.	Set this value to 2 - 4 times the number of logical cores.	Set this value to 0. HPC servers do not usually need job pools. Avoiding job pools will also free systems resources and allow job scheduling software to manage these systems more effectively.
Persistent Daemon Timeout (seconds)	Maximum time (in seconds) that an idle daemon will remain resident in memory before it is shut down. The default timeout is 300 seconds (or 5 minutes).	Use the default value (300).	Use the default value (300).	Use the default value (300).
Job Readiness Refresh Rate (seconds)	To keep the server warm during periods of inactivity, a single protocol runs periodically to keep system files in memory and improve performance when starting new jobs. This manifests as an additional scisvr process. The configuration time, in seconds, sets the amount of time in between each protocol execution. The default is 300 seconds (or 5 minutes). To disable this feature, set the value to 0.	Use the default value (300).	Use the default value (300).	Set to 0 to disable this feature.
Pre-started Daemons for Non-Pooled Jobs	When set to a number > 0, a pre-initialized daemon is used to run jobs that are not assigned to any job pool.	Set to 2 or more.	Set to 2 or less.	Set to 0.

Note: These guidelines refer to the number of logical cores. The value is normally equal to the number of physical cores available on your server. If hyper-threading is enabled, the number of logical cores is double the number of physical cores. For example, a quad-core processor with hyper-threading will present eight logical cores. The platform multi-process architecture works well with hyper-threading.

Job Pool Configuration

In most cases, the settings available in the Admin Portal are sufficient to configure and tune job pools on your server. However, more detailed configuration options are available through specific configuration files.

The main `SciSvrPolConfig.xml` file found in <pps_ install>/apps/scitegic/core/xml/Objects/ provides the default server pool configuration.

These settings can be customized by editing the Admin Portal or by adding specific configuration pool files in your <package>/xml/Objects folder or in the <pps_install>/xml/Objects directory.

IMPORTANT! *<pps_install>* is the root of the Pipeline Pilot installation.
On Windows this is typically:
`C:\Program Files\BIOVIA\PPS`
On Linux this depends on the location of your Pipeline Pilot installation, but may be:
`[Home]/BIOVIA/PPS`

This list provides details for each configuration option:

```
<sci:mpropval name="_default_pool" type="StringType">
<sci:value>1</sci:value>                    <!-- format -->
<sci:value>scitegic/core</sci:value>        <!-- package provenance -->
<sci:value>0</sci:value>                    <!-- start servers -->
<sci:value>0</sci:value>                    <!-- min spare servers -->
<sci:value>-1</sci:value>                   <!-- max spare servers -->
<sci:value>5</sci:value>                    <!-- max spare servers trim time-->
<sci:value>16</sci:value>                   <!-- max servers -->
<sci:value>32</sci:value>                   <!-- max queue depth -->
<sci:value>-1</sci:value>                   <!-- max requests per server -->
<sci:value>300</sci:value>                  <!-- idle timeout -->
<sci:value></sci:value>                     <!-- warmup -->
<sci:value>0</sci:value>                    <!-- rerun warmup time -->
<sci:value>80</sci:value>                   <!-- memory threshold -->
<sci:value>15</sci:value>                   <!-- individual usage threshold -->
</sci:mpropval>
```

Job Pool Settings

Setting	Description
Format	A flag used to version the job pool configuration details. Always set this to 1 unless otherwise directed.
Package Provenance	The name of the package that defines the job pool. Set this to the name of your package or to "scitegic/core" if the pool settings are defined in the global XMLDB location.
Start Servers	The number of `scisvr` processes to launch for the pool when the Pipeline Pilot server starts up (default is 0).
Minimum Spare Servers	The minimum number of processes to keep in the pool when the pool is contracting (default to 0).
Maximum Spare Servers	The maximum number of `scisvr` processes that should be available at one time (default is -1). The default value of -1 indicates that the process available state is not checked and the pool will increase to its maximum total size and contract only when idle processes timeout.
Maximum	The number of seconds to wait before pruning `scisvr` processes down to the Maximum Spare Servers.

Setting	Description
Spare Servers Trim Time	
Maximum Servers	The total maximum number of `scisvr` processes that can exist in the job pool at any point in time. This value should be equal to or greater than the value specified in Maximum Spare Servers.
Maximum Queue Depth	The total number of requests to hold in the queued state while waiting for `scisvr` processes to handle the requests. Pipeline Pilot will reject additional job requests when the queue reaches its maximum depth. A value of -1 indicates that any jobs in excess of Maximum Servers should still spawn new `scisvr` processes to handle the request. A value of -2 indicates that the pool should use the default maximum (default is 256).
Maximum Requests Per Server	Each `scisvr` process will handle up to this many requests before exiting. A value of -1 indicates there is no limit.
Idle Timeout	The number of seconds a `scisvr` process may be idle before it becomes available for pruning. A value of -1 indicates no idle timeout.
Warm-up Protocol	The fully qualified path or GUID of a protocol to run when a new `scisvr` processes starts and is not immediately assigned a user-submitted job. Use warm-up protocols to pre-load binaries, component, or create connections that improve response time for user jobs.
Rerun Warm-up Protocol	The number of seconds to wait before rerunning the warm-up protocol. This keeps resources loaded in memory and ready to serve requests.
Memory Threshold	Pipeline Pilot will pool `scisvr` processes until the host operating system exceeds this overall physical memory threshold. Pipeline Pilot will prune pooled `scisvr` processes to prevent or reduce swapped-to virtual memory in order to improve the overall system performance and throughput.
Individual Usage Threshold	The maximum percentage of physical memory that any individual process can consume before pruning.

Chapter 6: Security Considerations

Pipeline Pilot supports various configuration options and settings that can help protect your data from security vulnerabilities. When planning your deployment, consider the following recommendations for internet-accessible deployments and other environments that require a high level of security. Additionally, your deployment configuration should be validated by security experts and comply with policies within your organization.

Server Security

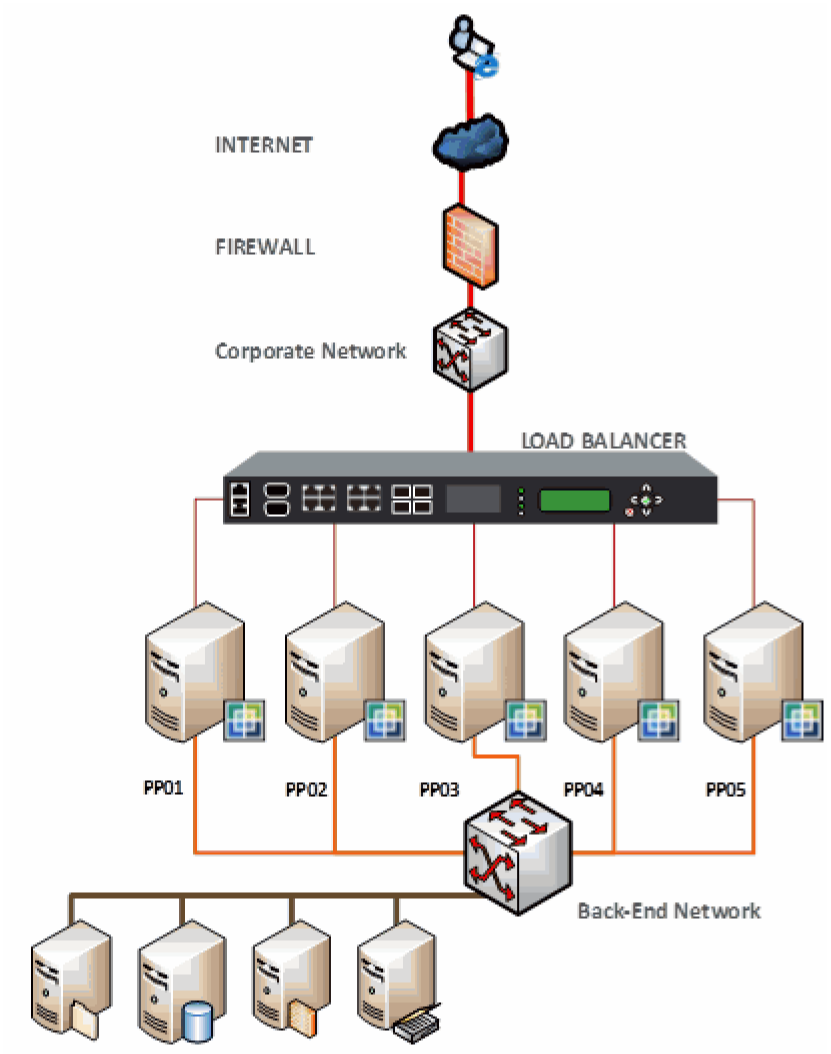
Secure deployment, whether on a corporate network or over the internet, requires proper configuration. Pipeline Pilot deployments can be internet-accessible without a requirement for a reverse proxy or virtual private network. However, this configuration requires some protection by using third-party networking, firewall, and additional security precautions on the server itself.

Server Infrastructure

Set up your server infrastructure with security in mind. Implement security layers, such as firewalls, proxy servers, and a DMZ, that adhere to your company policies, good security practices, and the following guidelines. Be sure to keep all hardware up to date with the latest patches. Consider the physical security of your hardware by using secure, limited-access facilities.

Use a Firewall

An internet-accessible Pipeline Pilot server should always be deployed behind a stateful firewall that accepts only HTTPS traffic.



Implement SSL Termination

Consider implementing SSL termination for an internet-accessible Pipeline Pilot server deployment. Firewalls, reverse proxies, and load balancers can provide SSL termination, which offloads encryption and decryption from the primary server and provides a single certificate entity for a set of servers. Placing the endpoint device in a DMZ adds an additional layer of security by not allowing direct access to the web server from the internet and only allowing network connectivity from the reverse proxy internal interface.

Reduce Attack Surface and Endpoints

When deploying an internet-accessible Pipeline Pilot server, it is important to reduce the amount of "attack surfaces" and endpoints that are exposed. Many networking devices (firewall, proxy, load balancers, web application firewalls) provide features, such as allowlists, that allow only certain URLs to pass through the device and onto the server. Consider using these features to prevent outside requests from reaching sensitive locations on the server (for example, Admin Portal /admin). Endpoint devices can also be deployed to protect the Pipeline Pilot server from known web application attacks, such as cross-site scripting and directory traversal.

SSL Certificates

For your production deployment of Pipeline Pilot, you must obtain and install a trusted SSL certificate from a recognized Certificate Authority to secure communication between the server and any client connecting through SSL.

For development or test systems, you can generate a self-signed certificate for your server host computer. However, self-signed certificates will cause some clients, typically web browsers, to prompt the user to ensure they want to connect to a "non-trusted sources".

For more information, see Managing SSL Certificates in the *Pipeline Pilot Help Center*.

Encryption

Encryption in Transit

To ensure that sensitive data is communicated securely between the server and client applications, Pipeline Pilot server supports secure connectivity using SSL over HTTPS. For performance and security reasons, you might want to leverage a networking device to handle the SSL termination. In this case, you can configure Pipeline Pilot for unencrypted HTTP-Only connectivity, so that communication between the SSL termination device and the Pipeline Pilot server uses HTTP. This setup requires that the network between these two devices are physically and logically secure to prevent "man in the middle" attacks and packet capturing. Unencrypted HTTP connectivity between the endpoint device and server allows for visibility for troubleshooting and security monitoring.

It is recommended that you configure the server or the SSL termination device, if applicable, to allow only traffic encrypted with TLS 1.2.

Encryption at Rest

Ensure that sensitive information, such as passwords and account information, is stored in a secure way. For example, it is important to ensure that developers do not store passwords in plain text or hard-code credentials into protocols and components that could be used in unauthorized ways. For more secure, encrypted credential storage, an administrator can configure access to shared data resources in the Pipeline Pilot Admin portal, and only authorized users can leverage the credentials from data access components. For more information, see Data Sources and Tagged Resources in the *Pipeline Pilot Help Center*.

Application Configuration (Server)

For internet-accessible deployments and other environments that require a high level of security, use the following recommended settings to reduce the server's vulnerability to common attacks.

Notes:

- Depending on the security requirements for your organization, you might want to change some the default settings.
- This list highlights some important settings for server security. For information about all available configuration settings, see the *Pipeline Pilot Help Center*.

To access the following settings, go to the Pipeline Pilot Admin Portal, and go to **Setup > Server Configuration**.

Setting	Default	Recommended for high security	Explanation
File Browsing	Restricted	User Folder Only	Users can browse only their own user data folder. When set to Restricted, users can browse only within their own data folder, the public directory tree, and shortcuts defined by packages and by administrators. **Note:** If impersonation is enabled, the operating system settings control a user's file access privileges, instead of the **File Browsing** and **File Editing** settings.
File Editing	Restricted	Restricted	Users can modify only files and folders within their own user folder.
Unrestricted File Download	[empty]	[empty]	Only authenticated users can download files from the server. If folders are listed in this setting, files in those folders can be downloaded without authentication.
Job Directory Web Access	Unrestricted	Restricted	When set to Restricted, only the user that ran a job can access its results, by default. The job owner may then grant access to specific people or groups.
Expiration of Single Sign-on Credentials	1d	60m	For web applications, a short session timeout is recommended to reduce the possibility of unintended system access.
Retain Session Cookie beyond Web Browser Session	Yes	No	When this setting is disabled, the web browser deletes the session cookie on shutdown. Users must provide their credentials again when opening a new browser session.
Enable HttpOnly flag on Session Cookies	Yes	Yes	Session cookies are created using the HttpOnly flag. This feature mitigates the effect of cross-site scripting attacks by preventing Javascript-based applications from accessing the cookie value and using it to perform unintended actions on behalf of the client user. When enabled, this feature might negatively affect some legacy clients.
Validate Redirect URLs	No	Yes	Redirect URLs are validated before authentication. This feature ensures that users are not forwarded to untrusted sites and only trusted sites receive security tokens. Enabling this feature might affect backward compatibility and requires you to

Setting	Default	Recommended for high security	Explanation
			configure the list of trusted URLs. See Configure Trusted URLs on page 33.
Enable Server Info Pages	Yes	No	When this setting is disabled, users cannot access sensitive information about the server installation from outside the Admin Portal.
Enable Full Server Configuration Listing	No	No	When this setting is disabled, there is a restriction on which configuration settings can be retrieved by querying the server.
Allow Content Framing by Non-Origin Hosts	Yes	No	When this setting is disabled, an anti-framing HTTP header is set for every response, which prevents embedding content from other sites. This feature defends against clickjacking attacks.
Server Port Usage	Default	SSL Only	When set to SSL Only, incoming traffic is allowed only over a secure SSL/TLS connection, reducing the possibility that an attacker could intercept and misuse session cookies or security tokens. Session cookies are created with the "secure" attribute, which tells web browser to transmit cookies only over secure connections. The Default setting allows both SSL and HTTP traffic.
SSL/TLS Security Level	Intermediate	Intermediate	The SSLCipherSuite and SSLProtocol values of the Pipeline Pilot HTTPS port are configured based on the recommendations from https://wiki.mozilla.org/Security/Server_Side_TLS. The Intermediate setting is recommended because it is highly secure and supports a wide range of modern clients, including web browsers.
Allow Remote Administration	Yes	No	When this setting is disabled, the Admin Portal cannot be accessed from a browser on a remote machine.
Session Salt	[empty]	[custom password]	An additional string can be configured to use for session encryption. Using a custom string prevents the same session from being used to connect to multiple Pipeline Pilot servers.

To access the following settings, go to the Pipeline Pilot Admin Portal, and go to **Setup > Job Settings**.

Setting	Default	Recommended	Explanation
Job Directory Web Access	Unrestricted	Restricted	This setting is available on both the **Job Settings** and **Server Configuration** pages. For more information, see the explanation in the previous table.
Maximum Job Age Based on Job Folder Size	[empty]	1d	When a value is set, jobs without explicit expiration dates are cleaned up automatically, ensuring that sensitive information is not retained indefinitely on the server.

Configure Trusted URLs

Pipeline Pilot can perform validation of redirect URLs prior to user authentication. This feature ensures that users are not forwarded to untrusted sites and only trusted sites receive security tokens.

After you enable the **Validate Redirect URLs** configuration setting, Pipeline Pilot server trusts only itself by fully-qualified domain name or as localhost. You must configure the list of trusted URLs by editing Pipeline Pilot configuration files or by connecting Pipeline Pilot with BIOVIA Foundation Hub.

To add trusted URLs manually:

1. Log in to the Pipeline Pilot server host.
2. Go to the Pipeline Pilot server installation folder (`<pps_install>`).
3. Go to the `<pps_install>/xmldb/Objects` folder.
4. If the `TrustedURIs.2.xml` file does not exist, create it.
5. Edit the `TrustedURIs.2.xml` file by adding a new line for each trusted URL, including aliases to the same URL. See `TrustedURIs.xml` for an example.

Note: The server overwrites the `TrustedURIs.xml` file when synchronizing content. Add your custom settings only to the `TrustedURIs.2.xml` to prevent data loss.

Connect to BIOVIA Foundation Hub

When you configure Pipeline Pilot as part of a larger BIOVIA Foundation deployment, you must connect Pipeline Pilot to an instance of Foundation Hub. In this configuration, Foundation Hub provides master data management for various data types including user identity and application settings. Pipeline Pilot automatically synchronizes trusted URLs from Foundation Hub when connected. Therefore, Pipeline Pilot automatically trusts all applications registered with Foundation Hub.

Application Security

Use application settings to limit who can access the application and what data they can see and functions they can perform.

Authentication

Authentication allows you to control access to your server by verifying the identity of your network users. Pipeline Pilot offers several authentication methods, which you should implement based on your organization's security policies and how you want to control access. Authentication is not enabled by default.

This list highlights some important considerations when configuring authentication for your Pipeline Pilot deployment. For information about all available authentication methods, see Managing Authentication in the *Pipeline Pilot Help Center*.

- **Anonymous access:** Anonymous access allows an unidentified user to execute protocols by using a generic account, the credentials of which are stored in the Admin Portal. When it is enabled, an unauthenticated user can access specific endpoints that allow anonymous access, without being prompted for credentials. Instead, the user name and password of the generic account are used.

  To configure anonymous access, you must enter the user name and password of the generic account in the Admin Portal.
- **Local:** You can use local computer user accounts to define permissions for a deployment. This option is not recommended for an enterprise deployment because it makes overall administration of increasingly challenging with additional servers. However, it might be appropriate for a development or test system.
- **External user directory:** You can implement authentication methods that are not maintained by Pipeline Pilot itself. Options include using Foundation Hub as the authentication server and configuring domain authentication for Windows servers.
- **SAML web single sign-on:** Pipeline Pilot can act as a service provider that federates with an identity provider based on the SAML 2.0 Web Browser SSO Profile specification.

Update Administrator Accounts and Default Passwords

Pipeline Pilot ships with a default account called "scitegicadmin" that you use to log in to the Admin Portal after the initial installation. The default password for this account is the same for all installations and well known to users of Pipeline Pilot. Therefore, you should change the password for this account or replace it by setting up domain-based or file-based accounts and adding them to the Platform/Administrators group.

IMPORTANT! Before disabling the scitegicadmin account, test the login for your new administrator account.

Impersonation

Under impersonation, jobs are run as if the user launched them from the command line with the privileges only granted to the user account. This allows users and groups to keep their data private at the operating system level. However, impersonation assumes that users have a system account on the servers that run Pipeline Pilot and are able to log in to access their data. For some applications, this might not be desirable (for example, a load-balanced application that does not require users to have a login ID). If impersonation is not used, protocols run by users have the same access rights as the server process.

Impersonation can be used with grid engines and clustering. Load-balanced configurations or public deployments might want to avoid using impersonation.

For more information, see Managing Impersonation in the *Pipeline Pilot Help Center Administrators tab*.

Authentication Examples

- **Corporate directory plus file-based authentication:** You might need to extend access beyond the set of network accounts. For example, you might create a set of administrator accounts that have unique passwords for the Admin Portal. In this case, an administrator can define a list of users and passwords in addition to configuring authentication through the corporate user directory. However, if the additional users need to run protocols, impersonation must be turned off.

- **Corporate directory authentication plus anonymous access:** A server might expose some functionality that requires authentication (for example, protocol development or Web Port use), but also some functionality that supports anonymous access. The anonymous access can be used to run protocols from other web pages, such as those that update reports with real-time data. In this example, an administrator would configure authentication through an external user directory and enter the credentials for anonymous access.
- **Windows server with domain authentication and impersonation:** A Windows server on which protocol developers and users need access to restricted network resources generally requires domain authentication and impersonation. If the impersonation must cascade to child processes of the protocol, then impersonation must be set to **Full**. Otherwise, **Restricted** impersonation will suffice. In this scenario, users must have, at a minimum, the **Logon as a Batch Job** permission on the server. If anonymous access is enabled, the credentials must be a valid Windows domain account.
- **High-security corporate directory authentication:** For the strictest level of security with corporate directory authentication, you can disable both user list authentication and anonymous access, since these provide alternatives to the directory authentication. You can also disable impersonation, since this requires a (modest) level of user access to the server machine itself. When configuring authentication, select the option to **Accept passwords via SSL only**. Finally, deploy (generally as packages) only protocols developed and validated on other servers, which is required on load-balanced installations. This process allows administrators to keep tight control over which protocols can be run on the server.

Authorization

Pipeline Pilot provides robust methods for managing authorization both within Pipeline Pilot and for other applications developed by BIOVIA, BIOVIA Partners, or internally within your organization.

Secure Protocols and Components

Protocols and components can be secured to limit access and use to specific users or groups. In addition to preventing unauthorized use, this can provide users with streamlined set of choices by removing functionality that applies to other teams, projects, or job definitions.

To manage access rights for protocols and components, log in to the Admin Portal, and go to **Security > Access Rights**. For example, you can mark all protocols and components as read only and allow read/write access only for administrators. In this configuration, only administrators can make changes to shared components and protocols, which might be preferred for environments that require a high level of security. For more information, see XMLDB Access Rights Overview in the *Pipeline Pilot Help Center*.

There are alternative ways to secure protocols and components. To limit use of Web Port protocols, you could create custom Web Port instances for each set of users. Protocols can also include security checks. Putting protocols and components in a package can secure them against unwanted use or development. Components can be restricted to allow other developers and end users to run the component without having read or write access to the sensitive implementation details.

For recommended operational policies for publishing protocols and managing access rights, see Suggested Processes and Policies on page 43.

Configure Groups and Assign Permissions

A group is a collection of users that have the same set of permissions to perform tasks in Pipeline Pilot. Pipeline Pilot includes a set of predefined groups, and you can create custom groups to help manage user access rights.

Pipeline Pilot includes a set of default system permissions. Other BIOVIA-provided packages and collections can define their own system permissions and groups, so the available system permissions can vary depending on your installation of products.

For more information, see Groups and Permissions Overview in the *Pipeline Pilot Help Center*.

Troubleshooting

Administrator Lockout Problems

The Admin Portal is accessible only to users with the *Platform/Administration/Logon* permission. To ensure that your system security and access settings are up to date, an administrator with this permission must be able to log in to the Admin Portal at all times.

If an administrator is locked out of the system, follow the Support for Security Issues troubleshooting guide in the *Pipeline Pilot Help Center* or contact Dassault Systèmes Customer Support.

Dassault Systèmes Support Resources

For additional resources or to contact Dassault Systèmes Customer Support, visit the Support portal:

https://www.3ds.com/support/

From this portal, you can:

- Call or email Dassault Systèmes Customer Support
- Submit a request
- Download installers
- Access hardware and software requirements
- Access Knowledge Base
- Access Communities and Twitter feeds

Chapter 7: Planning for Operation

This section provides guidance on what an administrator should plan for after Pipeline Pilot is deployed for development or production.

Backup Procedures

Backing up a server is a standard practice within IT infrastructure services. Within a Pipeline Pilot deployment, the following areas require backup:

- XMLDB using either the Admin portal, DBUtil command line tool, or use a standard backup method for the file system pointing to the XMLDB folder. See the Pipeline Pilot Help Center at:
  Admin > Server Maintenance > XMLDBs > Backup Up an XMLDB
- Folders, as defined previously in this document, using a standard backup method on the file system.
  - /public/users
  - /logs/usage
  - /xmldb/Users
  - /xmldb/Components
  - /xmldb/Protocols
  - /xmldb/Objects
  - /apps
  - /web/apps
- Configuration using Import/Export within Pipeline Pilot Help Center, see:
  Admin > Server Maintenance > Exporting and Importing Server Configurations
  - Server configuration changes should be very infrequent
  - As changes are made to the server, you should periodically export the configuration and put that in a location that is both backed up and available to "undo" any changes previously made.

Restore Procedures

While backup procedures are important, testing your restore process is critical.

In the Pipeline Pilot Help Center, see:

Admin > Server Maintenance > XMLDBs > Restoring an XMLDB

Keep the following points in mind when planning a restore:

- XMLDB - Backup files are stored in a folder specified in your Admin Portal Settings page and use the extension .xmldb.zip. The backup is a compressed copy of the entire XMLDB folder. Restoring converts all files into the XML file formats saved on the server for your components and protocols. When you restore XML files from a backup copy, all changes to protocols and components since your last backup are overwritten. Use the backup that contains the latest versions to minimize the amount of rework.
- Database Drivers - JDBC and ODBC drivers and configuration
- Files and Folders - The folders and files that were defined previously in this document

Note: Do not restore `<pps_install>/xmldb/Objects`, whose files can contain server, path, and port-specific details to a different server.

Keeping Your Server Healthy

Depending on your use of Pipeline Pilot within development or production, keep following areas in mind.

Managing Versions

When an author using Pipeline Pilot Client saves a protocol or component that they are working on, Pipeline Pilot will make this a new "version" to ensure the author can revert to a previously saved point. The impact to this is it can bloat the size of the XMLDB since each version is stored as a snapshot. Within the Admin Portal, you can purge versions and decide which versions to keep (for example, the last 6 versions). This practice and policy should be well communicated to the authors using the server.

For more information, in the Admin Portal Help Center, see:
Admin > Server Maintenance > XMLDBs > Cleaning Up an XMLDB

Job Folders

Each time a protocol is executed, a job folder is created in the user directory. That folder can contain temporary data and results data depending on how the protocol author configured the protocol. Depending on the usage and number of users that progressively runs protocols on that server, this could result in growing consumption of the disk volume.

To manage the job folder:

- Purge Job Folder - It is recommended that you purge job folders on a regular basis (for example, weekly) based on a given threshold (for example, keep only job folders that have been updated within the past two weeks).
- Job Folder Location - It is recommended that you configure the job folder to a different location that supports high read/write performance such as a RAID stripped volume or other high performance volume storage.

Upload Folder

Within the Admin Portal, there is a folder setting to a directory that stores files uploaded to the server. This is an optional directory that administrators can configure if necessary. If an Upload directory is not defined (and the setting is left blank), User folders are used for this purpose. When users upload content to Pipeline Pilot, as the default is to store the content within the User folder for that user.

This uploaded data is typically short lived and can consume a large amount of disk space on your production volumes. By allowing you to redirect the uploaded content to a different location, you can both periodically purge older or unneeded data and keep it separate from other user data.

More information can be found in the Pipeline Pilot Help Center at:
Admin > Setup and Configuration > Folder Locations > Upload Folder

Managing Your Log Files

The following logs provide useful information about your deployment.

Log of Job Execution Events

This log is maintained in the file `<pps_install>/logs/usage/PipelinePilot.log`.

This log:

- Is a structured, delimited file with one line per completed job
- Includes information on the name of the protocol executed, the time and duration, the client user, and which packages were employed

For more information, see the Pipeline Pilot Help Center at:
Admin > Reports > Completed Jobs

Log of User Authentication Events

This log is maintained in the file `<pps_install>/logs/usage/access.log`.

This log:

- Is a structured, delimited file with one line per authentication event
- Includes initial user login events and revalidation of the user credentials
- Includes information of the event time, the login name, the client program type, the authentication result (0=failed, 1=success, 2=reset), and the client IP address

Error and Diagnostic Messages

There are a number of files used for this purpose, all under the `<pps_install>/logs/messages` folder.

There are three categories of file:

- `scitegicerror_NNN.log`
- `javaserver_NNN.log`
- Files in the java subfolder

scitegicerror_NNN.log

NNN identifies a process name. Possible names are:

- `cgiadmin`
- `cgifile`
- `cgicache`
- `cgihelp`
- `scisvr`
- `runprotocol`
- `regression`
- `dbutil`
- `client`
- `pkgutil`
- `apache`
- `aepmanager`

The format of these files is a time stamp, followed by a process and thread id, a severity indicator, and then a message, which can be a single entry or a multi-line entry such as a stack trace. Some messages might contain new lines, which appear to flow over multiple lines when viewed in a text editor. Messages can be errors, warnings, or trace (informational) in nature.

javaserver_NNN.log

NNN identifies a java application name. Possible names are:

- `catalog`
- `derby`
- `platform`
- `securityfilter`
- `tomcat_shared`

Each file follows a standard java log file format with a time stamp and a message, although the formatting details can vary.

java Subfolder

The java subfolder contains server error dump files, named `jvm_error_<process-id>.log`.

Managing BIOVIA Catalog Indexes

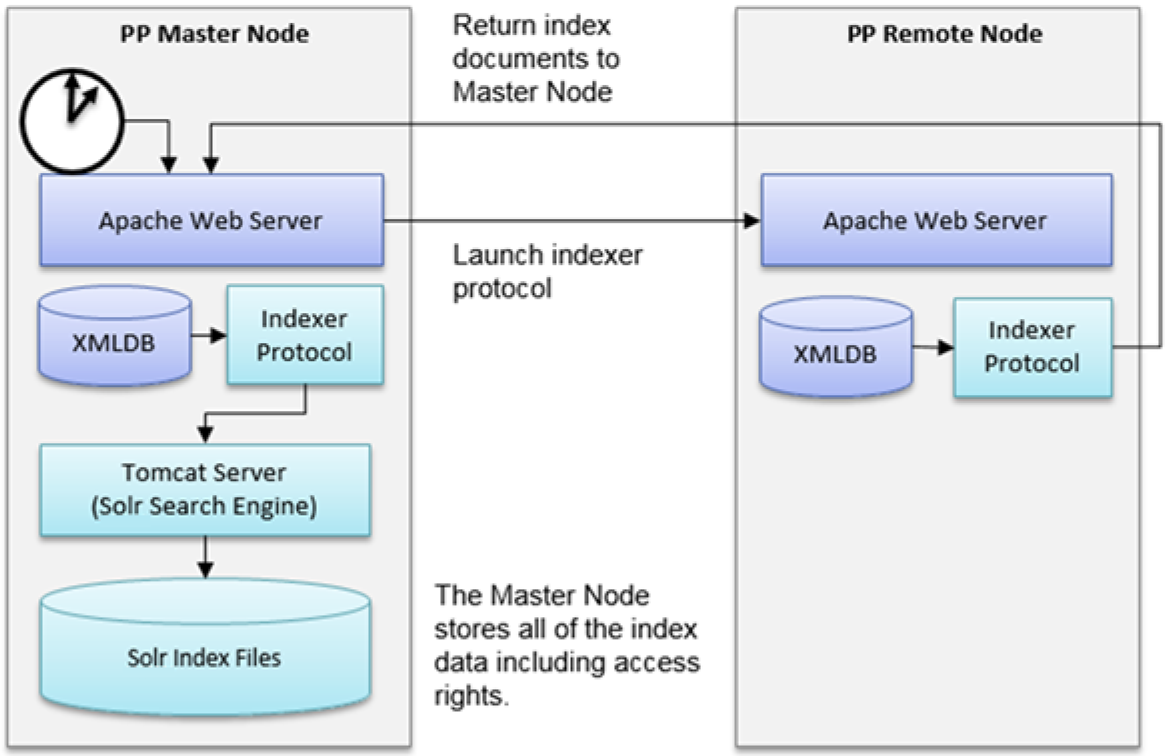
You can use the Catalog Search feature to explore protocol and component database contents and usage patterns. For example, when preparing for an upgrade, you can identify most or least frequently used protocols and find protocols that are authored by specific users.

For more information, see the Pipeline Pilot Help Center at:
Admin > Reports > Catalog Search

As a general server administration tool, you can query for references about specific servers, authors, and protocol parameters. Catalog Search supports constructing ad hoc queries that result in a set of protocol names with associated protocol and user information. You can search the current server or other servers that are indexed as part of the Catalog, including pre-9.0 servers.

Catalog Architecture

The Catalog Search service uses the Apache Solr 4.0 search engine to perform full-text searching and field-based searches over components and protocols. The Catalog Search is typically setup on one Master Node. A scheduled task on the Master Node triggers the indexing protocol process.



The indexer protocol extracts information from the local file system. The extracted information includes the content of components, protocols, access rights, and usage log information. This data is organized into documents that describe each component or protocol and is then added to the search index through Solr.

Local versus Remote Indexing

Catalog Search indexes the local SMLDG by default. You can add one or more remote XMLDBs to the Catalog Search Index by configuring the Server Registry in the Admin Portal. Catalog Search will index remote XMLDBs by running a protocol on the Pipeline Pilot node.

Security Considerations

End-users search the catalog index through a variety of client applications like Pipeline Pilot, Web Port, and the Admin Portal. However, all clients access the search index using a central web service hosted on the Master Node. The common search service ensures that all clients support the same search syntax, return the same result sets and check security consistently. The search service applies the following security checks:

- The logged on user is authenticated and has a valid session token
- The logged on user has permission to either search or update the catalog index
- The logged on user has permission to read the components and protocols in the result set

Catalog search automatically removes any components and protocols in the result set that the user is not permitted to read. For example, users can never see protocols stored on another user’s tab. However, administrators can find it helpful to search across user tabs. In this case, administrators with the **Platform | Catalog | SeeAllIndex** permission are permitted to search across all component and protocols in the index.

For more information, see the Pipeline Pilot Help Center at:
Setup and Configruation > Catalog Settings
Admin > Reports > Catalog Search

Indexer Configuration and Scheduling

The catalog indexer provides many configuration options for you to tune. For example, you can select which top-level folders to index, which servers to index, and whether to perform full or partial rebuilds.

Indexing Folder Types

You can choose which folders to index on the Admin Portal page, including Protocols, Components, and User Folders. It is common for the Users folder to consume the most space and take-up the most indexing time. Consider disabling user folder indexing when you:

- have a very large XMLDB with many user protocols
- have low available disk space in the Pipeline Pilot installation folder
- need to update the index more frequently

Partial Rebuild

As of Pipeline Pilot 9.1, the Partial Rebuild option performs an incremental update by comparing the contents of the local XMLDB against data stored in the master index. Partial rebuilds capture all protocols that were created, updated, deleted, or run for all selected folders. This process is much faster and normally completes in a few minutes for typical usage.

Full Rebuild

If no index exists for a given server, then the indexer will automatically perform a full build of the index. Occasionally you need to rebuild the catalog index. You can manually rebuild the index by selecting **Rebuild Index** on the **Admin Portal > Catalog Settings** page and then clicking on **Update Index Now**. A full build or rebuild of the index can take several hours for a large XMLDB (>10,000 protocols).

Consider performing a full rebuild of the catalog index when you:

- install many new packages
- update the server to a new version
- schedule frequent partial rebuilds and notice query performance degrading

Scheduling the Indexer

The index picks up recent changes the next time the index job runs. Therefore, you need to balance update frequency against the indexer over-head. Partial rebuilds typically takes less than 1 minute to complete on small to medium sized databases. This means, you can update the index several times per day. However, you should update less frequently if you have a large XMLDB. Once per day during off-peak hours is sufficient in most cases.

Note: You should not typically configure the indexer schedule task to rebuild the index unless you have a small XMLDB (<5,000 protocols).

Disk Usage

Catalog Search stores index files in the Pipeline Pilot server is installation folder. It is located at: `<pps_install>/database/solr/appcatalog`. This location is not configurable.

The data size varies depending on the number, size, complexity, and folder types selected on the **Catalog Settings** page. With the standard Pipeline Pilot installation, the hard disk needs at most 10MB per 1000 protocols. Larger databases can consume upwards of 10 GB of disk space.

Backup and Recovery

The index database is not backed up automatically unless you automatically back-up the entire Pipeline Pilot installation folder. However, the index files can be easily regenerated by performing a full rebuild after recovering the XMLDB contents.

Suggested Processes and Policies

Publication of Protocols

Protocols are typically considered published when made available within the Protocols or Components area on the production server. This can be done by either using the client or deployed as part of a package.

For Web Port publication of interactive forms and scheduled tasks, the best practice is to have moderate controls with defined ownership and documentation.

There are standard validation tools to ensure protocols meet specific standard. You can extend them to address deployment-specific needs such as adherence to corporate coding standards, or to prevent publication of a protocol that lacks access to a key resource. Validation is typically invoked manually by the developer from the Pipeline Pilot Client, but it can be configured to be run automatically for a given server when a protocol is moved or saved.

For more information, see the Pipeline Pilot Help Center at:
Admin > Reports > Validation Reports

Authors building protocols on development servers should manually run the validation to ensure they are conforming to corporate policies and best practices. BIOVIA strongly recommends that validation be automatically run on any saved protocol on production servers.

While validation does not fully replace peer review, it helps ensure that published protocols are not missing key properties. The standard validation protocol looks for DSNs that are referenced but not defined for your server, hard-coded file paths that are not found from the current server, etc.

XMLDB Folder Structure, Access Rights, and Roles

As described previously, having a core shared folder structure is important for the following reasons:

- Defining ownership for development where one protocol can refer to another by path where it can be known and the path is common to all servers
- Shared settings for access rights

A proposed structure is described in the Pipeline Pilot Help Center at:
Admin > Security > XMLDB Access Rights > XMLDB Access Rights

For all but the most locked-down application needs, project owners have access rights set to give full read-write access for their project folders on all servers. Access rights settings for other users are defined by the project owner and implemented by the server administrator. End-users access is further defined by their assigned role.

For example, an end-user is typically assigned WEBPORT role to allow Web Port login on the production server, but is denied this login on the Development server to prevent bypassing the publication process. After login, the folder access rights setting determines which protocols they can view from the interface.

Publication of Results

Results can be published almost anywhere in most formats. Common uses include updating a database or producing a static report, such as a PDF document. One feature of the Interactive Reporting collection is the ability to build complex dynamic HTML-based reports and typically use functionality that tie it to the server. Since job results folders are intended to be temporary storage locations, a common request by developers is for access to a location to which they can publish interactive reports available to end-users via a hyperlink.

The preferred location to write results that you want to keep long-term and URLs that you can share with others is into the path `<pps_install>/../web/apps` folder on your server. An alias on the Pipeline Pilot Apache Web server named `webapps` points to this `web/apps` directory, and you can create and reference subfolders to help keep your files organized.

For example, an HTML page written or copied to `<pps_install>/../web/apps/myReport/report1.html` can be accessed by all users at a URL such as `http://<servername>:<port>/webapps/myReport/report1.html`.

While `webapps` is a convenient alias, it is hard-coded to reside on the server collocated with your Pipeline Pilot server installation. You can create a corporate-specific alias to point to an alternate location. However, if left unchecked, these reports are written to any location that specified by the protocol, and there is no requirement that you define the report owner or the longevity be defined.

Data Longevity and Size

Some users have historically experienced disk space usage pressures. This can be mitigated by implementing policies, processes, and resource suggestions help limit consumption of space.

When a protocol is run a job folder is created, and has a defined lifetime that depends on the invoking client. Since interfaces like Web Port do not do explicit automatic job folder cleanup, some sites institute regular job folder cleanup as a scheduled task and notify their users of this policy.

For more information, see the Pipeline Pilot Help Center at:
Admin > Status and Monitoring > Job Management > Job Folder Maintenance

Packaging and Regressions

Stricter publication controls and testing requirements should be imposed for protocols whose mission is larger than a simple deployment as a scheduled task or through Web Port, such as a protocol being used as a versioned component in a critical deployment of a larger system.

Regressions and/or manual test plans for published protocols should be expected for any published protocol but should be mandatory for projects that have this type of larger scope or are of a business-critical nature.

For more information on regression, see the Pipeline Pilot Help Center at:
Developers > Deployment Guides > Component Development Regression Test Guide

For more information on packaging:
Developers > Deployment Guides > Application Packaging Guide