Server Deployment Package

User/Admin Guide
for Windows

Perforce Software, Inc.

26 June 2019

Preface

This guide tells you how to set up a new Perforce installation using the Server Deployment Package (SDP). Recommendations for optimal system maintenance and performance are included as well. The SDP follows best practices for Perforce server configuration and administration. It consists of standard configuration settings, scripts, and tools, which provide several key features.

- A volume layout designed for maximum data integrity and server performance.
- Automated offline checkpointing and backup procedures for server metadata.
- Replication to another server.
- Easy maintenance of user accounts, labels, workspaces, and other data.
- User authentication using LDAP or Active Directory.

This guide assumes some familiarity with Perforce, and does not duplicate the basic information in the Perforce user documentation. For basic information on Perforce, consult Introducing Perforce. For system administrators, the Perforce System Administrator's Guide is essential reading. All documentation is available from the Perforce web site at http://www.perforce.com.

Please Give Us Feedback

Perforce welcomes feedback from our users. Please send any suggestions for improving this document or the SDP to consulting@perforce.com.

Overview

The SDP has four main components:

- Hardware and storage/filesystem layout recommendations for Perforce.
- Scripts to automate offline checkpoints and other critical maintenance activities.
- Scripts to replicate the Perforce journal to another volume or server.
- Scripts to assist with user account maintenance and other routine administration tasks.

Each of these components is covered in detail in this guide.

We expect you to check the SDP into a depot called *Perforce* as part of the installation process. The directory structure of the SDP is shown below in Figure 1: SDP Directory Structure.

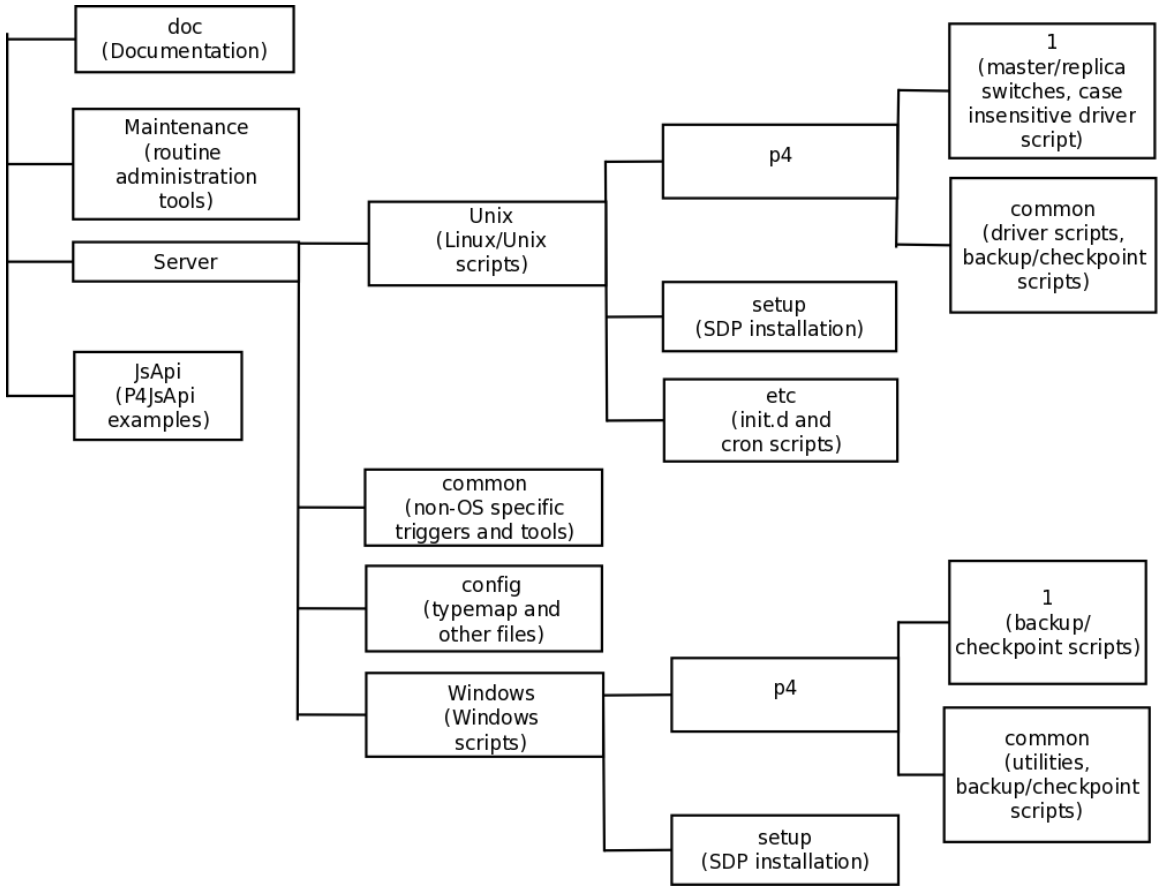


Figure 1: SDP Directory Structure

Configuring the Perforce Server

This chapter tells you how to configure a Perforce server machine and an instance of the Perforce Server. These topics are covered more fully in the System Administrator's Guide and in the Knowledge Base; this chapter covers the details most relevant to the SDP.

The SDP can be installed on multiple server machines, and each server machine can host one or more Perforce server instances. (In this guide, the term *server* refers to a Perforce server instance unless otherwise specified.) Each server instance is assigned a number. This guide uses instance number 1 in the example commands and procedures. Other instance numbers can be substituted as required.

This chapter also describes the general usage of SDP scripts and tools.

Volume Layout and Hardware

To ensure maximum data integrity and performance, use three different physical volumes for each server instance. Three volumes can be used for all instances hosted on one server machine, but using three volumes per instance reduces the chance of hardware failure affecting more than one instance. It is possible but not recommended to put all the files onto a single physical volume.

- **Perforce metadata (database files):** Use the fastest volume possible, ideally RAID 1+0 on a dedicated controller with the maximum cache available on it. This volume is normally called `/metadata`.
- **Journals and logs:** Use a fast volume, ideally RAID 1+0 on its own controller with the standard amount of cache on it. This volume is normally called `/logs`. If a separate logs volume is not available, put the logs on the `depotdata` volume.
- **Depot data, archive files, scripts, and checkpoints**: Use a large volume, with RAID 5 on its own controller with a standard amount of cache or a SAN or NAS volume. This volume is the only volume that must be backed up. The backup scripts place the metadata snapshots on this volume. This volume can be backed up to tape or another long term backup device. This volume is normally called `/depotdata`.

If three controllers are not available, put the `logs` and `depotdata` volumes on the same controller. Do not run anti-virus tools or back up tools against the `metadata` volume(s) or `logs` volume(s), because they can interfere with the operation of the Perforce server.

Back up everything on the `depotdata` volume(s). Avoid backing up the `metadata` volume directly, because doing so can interfere with the operation of a live Perforce server, potentially corrupting data. The checkpoint and journal process archive the metadata on the `depotdata` volume. Backing up the `logs` volume is optional.

The SDP assumes (but does not require) the three volumes described above.

View Figure 2: Volume Layout (below), viewed from the top down, displays a Perforce *application* administrator's view of the system, which shows how to navigate the directory structure to find databases, log files, and versioned files in the depots. Viewed from the bottom up, it displays a Perforce *system* administrator's view, emphasizing the physical volume where Perforce data is stored.

Both Unix and Windows installation of the SDP now use symlinks (on Windows this is via the `mklink` tool).

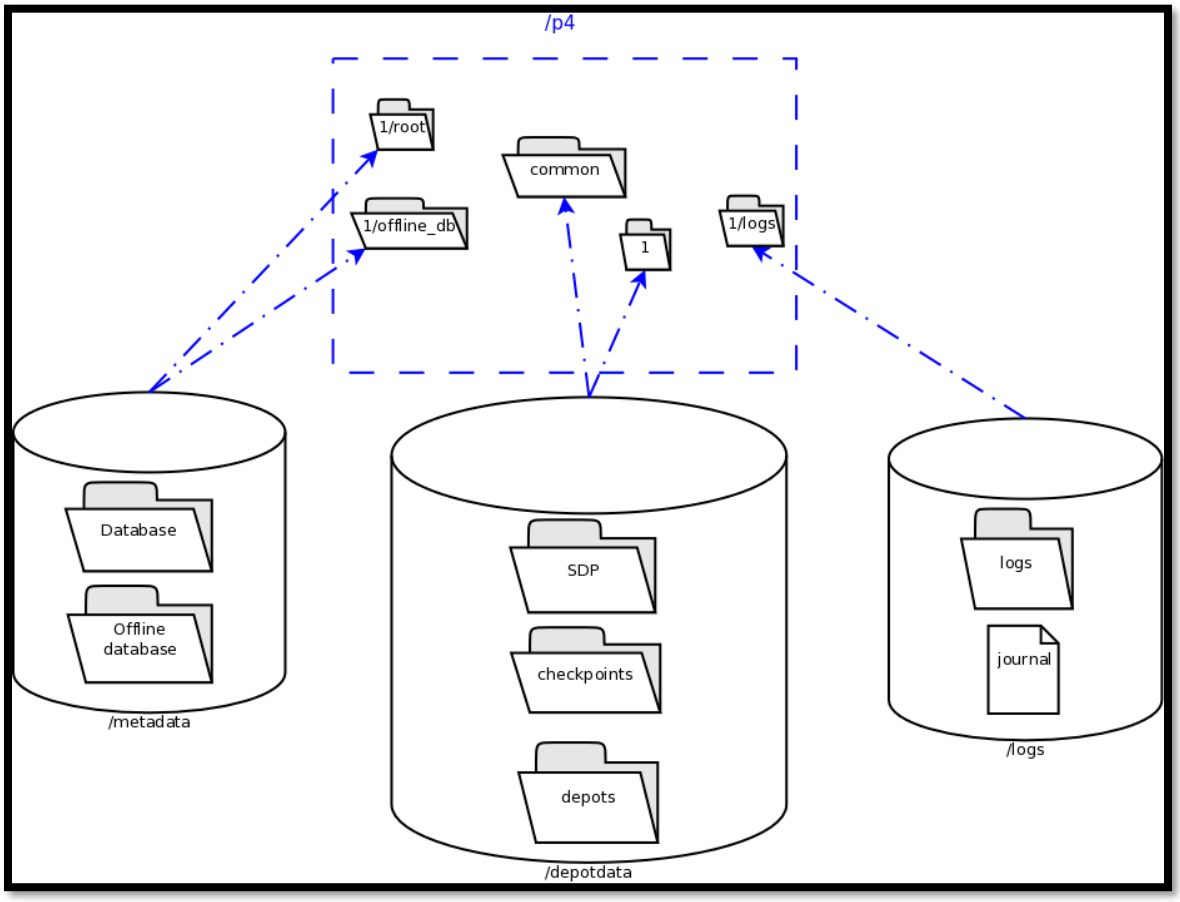


Figure 2: Volume Layout

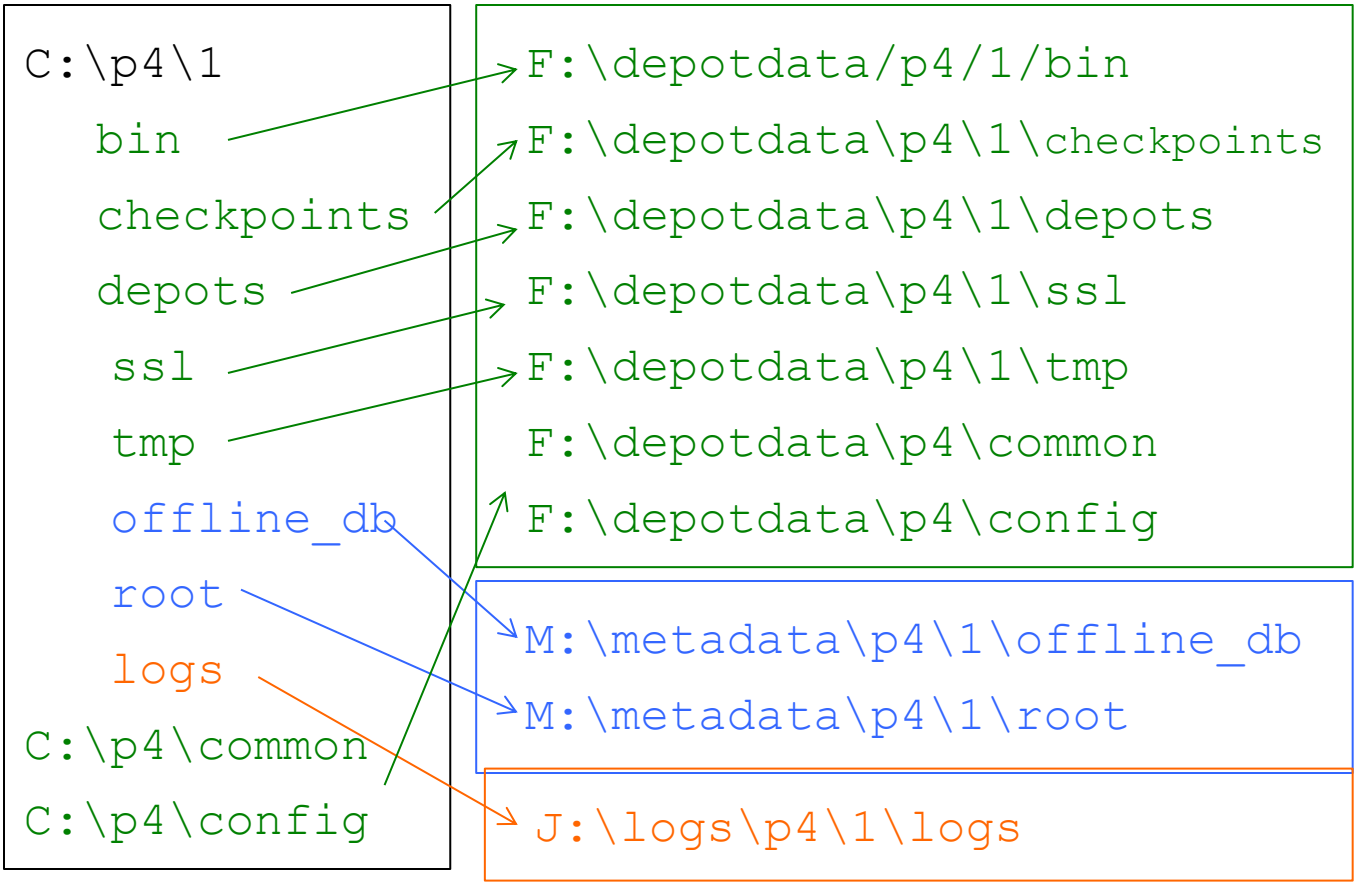


Figure 3: Logical and Physical View Mapping

The links are shown as <SYMLINKD> below on a Windows installation.

```
Directory of c:\p4
20/06/2014  15:05    <DIR>          .
20/06/2014  15:05    <DIR>          ..
20/06/2014  15:05    <SYMLINKD>     common [f:\p4\common]
20/06/2014  15:05    <SYMLINKD>     config [f:\p4\config]
20/06/2014  15:05    <SYMLINKD>     M1 [f:\p4\M1]

Directory of c:\p4\M1
20/06/2014  15:05    <DIR>          .
20/06/2014  15:05    <DIR>          ..
20/06/2014  15:05    <DIR>          bin
20/06/2014  15:05    <DIR>          checkpoints
20/06/2014  15:05    <DIR>          depots
20/06/2014  15:05    <SYMLINKD>     logs [g:\p4\M1\logs]
20/06/2014  15:05    <SYMLINKD>     offline_db [e:\p4\M1\offline_db]
20/06/2014  15:05    <SYMLINKD>     root [e:\p4\M1\root]
20/06/2014  15:05    <DIR>          ssl
20/06/2014  15:05    <DIR>          tmp
```

Instance Names

Traditionally the SDP has used integers for instance names which show up in the paths above, for example C:\p4*1*\root.

However it is increasingly the case that alphanumeric names are used for instances, e.g. C:\p4*Acme*\root. Commonly organizations strive to use a single Perforce instance, one logical data set, which may be replicated around the globe. Using a single instance optimize colloboration and simplifies code access for all development activity. When there is a single instance, the name '*1*' is as good as any. When there is more than one instance, e.g. if there are isolated silos of development activity, an alaphnumeric name may be more helpful than an integer for identifying the data set, such as *Acme* or perhaps *LegacyApps*. Another instance is sometimes to develop and test things like Perforce trigger scripts before rolling them out to the live production instance, or to provide a standing internal training data set.

In any case it is worth thinking and planning your naming, particularly if you have multiple instances including replicas of different types and these are located on different hosts.

If you are using instance numbers, then an example configuration where there are 2 master server instances, each with a replica, might be:

Server hostname	Instance ID	Port
`p4d-sfo-01`	1	1666
`sfo-p4d-01`	2	2666
`sfo-p4d-02`	1	1666
`sfo-p4d-02`	2	2666

For consistency, instances with same ID should refer to the same logical data set, they just run on different machines.

Alternatively, alphanumeric names can be clearer and easier:

Server hostname	Instance ID	Port
`sfo-p4d-01`	Acme	5000
`sfo-p4d-01`	Test	5999
`sfo-p4d-02`	Acme	5000
`sfo-p4d-02`	Test	5999

Some sites apply a convention to the port number to identify whether the P4PORT value is that of a master server, a broker, replica, edge server, or proxy. In such cases the first digit is reserved to identify the instance, and the remaining 3 digits identify the target service, e.g. 666 for a broker, 999 for a master server, 668 for a proxy.

Host naming conventions vary from site to site, and often have local naming preferences or constraints. These examples the the code of the nearest major airport, `sfo` in this case, as a

location code. Using location in the hostname is merely an example of a site preference, not necessarily a best practice.

End user **P4PORT** values typically do not reference the actual machine names. Instead they reference an alias, e.g. `perforce` or `sfo-p4d` (without the `-01`). This helps make failover operations more transparent.

Memory and CPU

Maximum performance is obtained if the server has enough memory to keep all of the database files in memory. Make sure the server has enough memory to cache the **db.rev** database file and to prevent the server from paging during user queries.

Below are some approximate guidelines for allocating memory.

- 1.5 kilobyte of RAM per file stored in the server.
- 32 MB of RAM per user.

Use the fastest processors available with the fastest available bus speed. Faster processors with a lower number of cores provide better performance for Perforce. Quick bursts of computational speed are more important to Perforce's performance than the number of processors, but have a minimum of two processors so that the offline checkpoint and back up processes do not interfere with your Perforce server.

General SDP Usage

This section presents an overview of the SDP scripts and tools. Details about the specific scripts are provided in later sections.

Most tools reside in `c:\p4\common\bin`. The directory `c:\p4\`***`instance`***`\bin` contains scripts and executables that are specific to a server instance, such as the `p4.exe` client. The scripts in `c:\p4\`***`instance`***`\bin` generally set the environment for an instance correctly, then invoke the corresponding script in `c:\p4\common\bin`.

Run important administrative commands using the scripts in `c:\p4\`***`instance`***`\bin`, when available. Then, use the `p4.exe` executable located in c:\p4***`instance`***`\bin`.

Below are some usage examples for instance 1 or instance master.

Example	Remarks
`c:\p4\common\bin\live-checkpoint.ps1 1`	Take a checkpoint of the live database on instance 1
`c:\p4\common\bin\daily-backup.ps1 master`	A daily checkpoint of the *master* instance.

Monitoring SDP activities

The important SDP maintenance and backup scripts generate email notifications when they complete.

For further monitoring, you can consider options such as:

- Making the SDP log files available via a password protected HTTP server.
- Directing the SDP notification emails to an automated system that interprets the logs.

Installing the Perforce Server and the SDP

This chapter tells you how to install a Perforce server instance in the SDP framework. For more details about server installation, refer to the Perforce System Administrator's Guide.

Many companies use a single Perforce Server to manage their files, while others use multiple servers. The choice depends on network topology, the geographic distribution of work, and the relationships among the files being managed. If multiple servers are run, assign each instance a number and use that number as part of the name assigned to depots, to make the relationship of depots and servers obvious. See the discussion above on Instance Names.

To install the SDP, you must have Administrator access to the Windows server machine(s) as you will be installing services.

Clean Installation

In this section we describe the server and SDP installation process on Windows. The process consists of:

1. Initial setup of the file system and configuration files.
2. Running the SDP configuration script.
3. Starting the server and performing initial configuration.

Do not use the Windows installer to install and configure the Perforce server. It will configure the environment for a single Perforce server.

Pre-requisites

The following is required (details mentioned below):

- Administrator account on the server
- Python installed (see below)
- Perforce Helix executables (p4/p4d – see below)
- Powershell 2.x or greater

Optional (but recommended):

- Perforce Helix Visual client (P4V – optional but very useful)
- An editor (Notepad will do, but Download Notepad++)
- GOW (Gnu on Windows) – optional but very useful for parsing log files etc.

Configuring Powershell

The scripts now use Powershell rather than .BAT files due to improved error handling and options, and code re-use (via a single included module rather than duplication of functionality in every script). This also allows us to keep the scripts more closely aligned with the functionality of the Unix scripts.

It is important to enable local scripts to be run. The following command must be run within an Powershell Administrator prompt:

For Windows 7, Windows 8, Windows Server 2008 R2 or Windows Server 2012, run the following commands as Administrator:

x86
Open `C:\Windows\SysWOW64\cmd.exe`
Run the command `powershell Set-ExecutionPolicy RemoteSigned`
x64
Open `C:\Windows\system32\cmd.exe`
Run the command `powershell Set-ExecutionPolicy RemoteSigned`
Use "get-executionpolicy" to check the policy has been updated. You may need to ensure that the various scripts are not "blocked" – right click in Windows Explorer and check Properties options.

Initial setup

Prior to installing the Perforce server, perform the following steps.

1. Mount the volumes for the three-volume configuration described in Volume Layout and Hardware. The procedure assumes the drives are mapped as follows:
    - Metadata on `e:`
    - Depotdata on `f:`
    - Logs on `g:`[1]
2. Copy the SDP to the `f:\sdp` directory (`%SDP%`).
3. Customize the following for your environment. It requires you to identify the master server and all replicas that we need to setup for the SDP, including instance names, hostnames, etc. This information is all in a single file:

    a. `%SDP%\Server\Windows\setup\sdp_master_config.ini`
4. Customize the following for your environment:

    a. `%SDP%\Server\Windows\p4\common\bin\run_p4review.cmd`
5. Download and install Python, e.g. from www.python.org. We use Python 2.7.x (latest) and 3.4.x (latest). Usually we will install 32 bit even on 64 bit OS for ease of compilation of extras, but it is your choice. Typically we install to default dir, e.g. c:\python27. For initial installation we only require base Python. For subsequent scripting you may wish to install P4Python (e.g. using pip).
6. Other tools we find useful: Notepad++ and GOW (Gnu on Windows - Unix command line utilities such as `wc, head, tail`). These are recommended but not strictly required.
7. Download to directory `%SDP%\Server\Windows\setup` the desired release of `p4.exe` and `p4d.exe`. For example, for Helix Core for release 2019.1 on 64 bit Windows, use this URL:

    `http://ftp.perforce.com/perforce/r19.1/bin.ntx64`

    From that directory listing, select `p4.exe` and then `p4d.exe` to download each of those files. If you are using 32 bit Windows (unusual these days), substitute bin.ntx86 for bin.ntx64 in the URL above..

Running Configuration script

The `SDPEnv.py` script, available in `%SDP%\Server\Windows\setup`, sets up the basic directory structure used by the SDP. It creates .bat files to register the Perforce service as Windows

---

[1] If you do not have a logs volume, put the logs on the depot data volume.

services. It parses and validates the `sdp_master_config.ini` file in the same directory.

You need to customize this `sdp_master_config.ini` file. It contains lots of comments as to how to set the various configuration values.

Review the contents of `template_configure_new_server.bat` file which defines the recommended default configurable values for any server, and make any desired changes. This file will be parsed and used to create instance specific configuration files.

After updating the configuration file, run `SDPEnv.py` from the same directory.

You must run this command from a CMD window which has administrator rights.

```
cd %SDP%\Server\Windows\setup
```

Edit and save changes:

```
notepad sdp_master_config.ini
```

Run the command to create the environment (by default it looks for the config file sdp_master_config.ini but this can be changed with –c option):

```
Create_env.py
```

If the output looks correct then re-run the script with `-y` parameter to actually perform the copying of files and creation of directories and links:

```
Create_env.py -y
```

Installing service(s)

The above command will create a couple of files in that directory. The first is `install_services_<hostname>.bat`, so on a machine where the hostname is `svrp4master`, it will be `install_services_svrp4master.bat`

Validate the contents of this file and run it if it looks appropriate – this installs the service(s) with appropriate parameters. Please note that it is specific to the **hostname** that you specified inside `sdp_master_config.ini` – so it will only run on the correct host server.

Note that if you have defined multiple instances in `sdp_master_config.ini` to run on this same hostname, then they will all be installed by this .bat file.

Start the server to test

Having installed the service, we now test that it will start: "net start p4_<instance name>", e.g.

```
net start p4_1
```

Or

```
net start p4_Master
```

If the service fails to start, then examine the log file for the reasone (e.g. missing license file) in `c:\p4\instance_name\logs`.

Ensure the server is running (specify appropriate port):

```
p4 -p 1666 info
```

Use "net stop p4_<instance>" to stop the service if required.

Applying configurables to the server instance

For each instance defined in `sdp_master_config.ini`, a configuration .bat file will be created, called `configure_<instance>.bat`, so for instance `master`, it will be `configure_master.bat`.

Review the contents of the file and make any desired changes.

You will only be able to run the .bat file if you have started the server instance as per previous section.

If an instance is a replica (or similar), then you should apply the configurables to the master server and then checkpoint it before seeding the replica – see the Distributing Perforce guide.

Configuring the server

To configure the server, perform the following steps:

1. Make sure your server is running (specify appropriate port below):
   ```
   p4 -p 1666 info
   ```
2. Create your Perforce administrator account within the Perforce repository, using the user name and password specified in `sdp_master_config.ini`.

3. Optional. To create a Perforce stream depot called `PerforceSDP` and load the SDP, issue the following commands:

```
p4 depot -t stream -o PerforceSDP | p4 depot -i

p4 stream -t mainline -o //PerforceSDP/main | p4 stream -i

cd /d C:\sdp

p4 client -S //Perforce/main -o PerforceSDP_ws | p4 client -i

p4 -c PerforceSDP_ws reconcile

p4 -c PerforceSDP_ws submit -d "Added SDP."
```

4. Optional. To create a Perforce spec depot, issue the following commands:

```
p4 depot -t spec -o spec | p4 depot -i
```

   Then add the following to the Protections table, near the bottom (about super user entries), to hide specs which could have security implications:

```
list user * * -//spec/protect.p4s

list user * * -//spec/triggers.p4s
```

   Then update specs in the depot with this command:

```
p4 admin updatespecdepot -a
```

5. Optional. To create an unload depot, issue the following command:

```
p4 depot -t unload -o unload | p4 depot -i
```

6. Optional. To delete the default Perforce depot naned `depot`, issue the following command: `p4 depot -d depot`. Create one or more depots as required to store your files, following your site's directory naming conventions.

Verifying your server installation

To verify your installation, perform these steps:

1. Issue the `p4 info` command, after setting appropriate environment variables. If the server is running, it will display details about its settings.
2. Create a client workspace and verify that it is archived in the spec depot and written to the `c:\p4\Master\depots\specs\client` directory.
3. Add a file to the server and verify that the archive file gets created in the corresponding directory under `c:\p4\Master\depots\Perforce`.

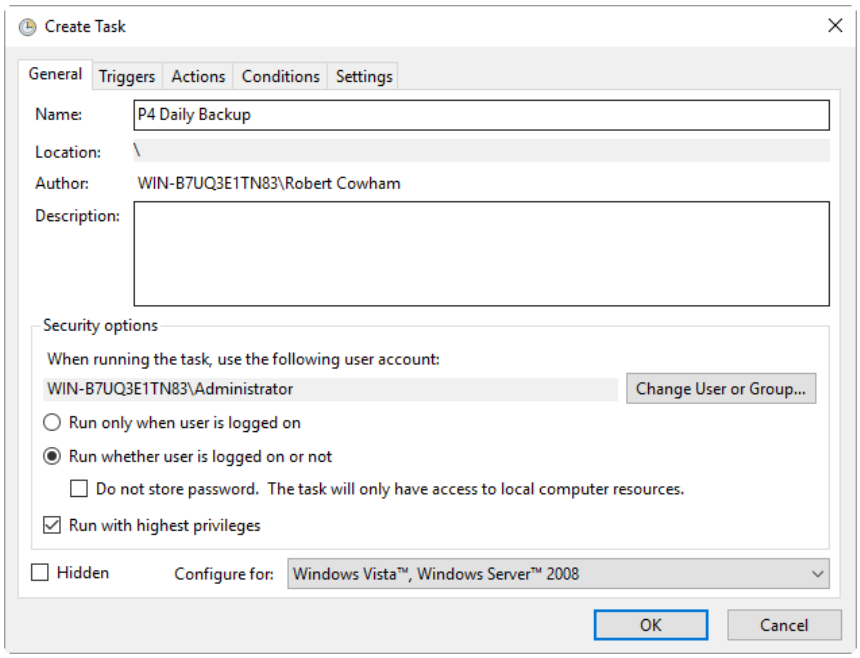
Scheduling maintenance scripts

In Windows 2012 you should use the Task Scheduler. We recommend that you create a folder called Perforce at the top level in which to create your tasks (otherwise they can be hard to find when you next look in Task scheduler!).

Note that the "schtasks" command can be useful from command line, or "taskschd.msc" to get the control panel equivalent.

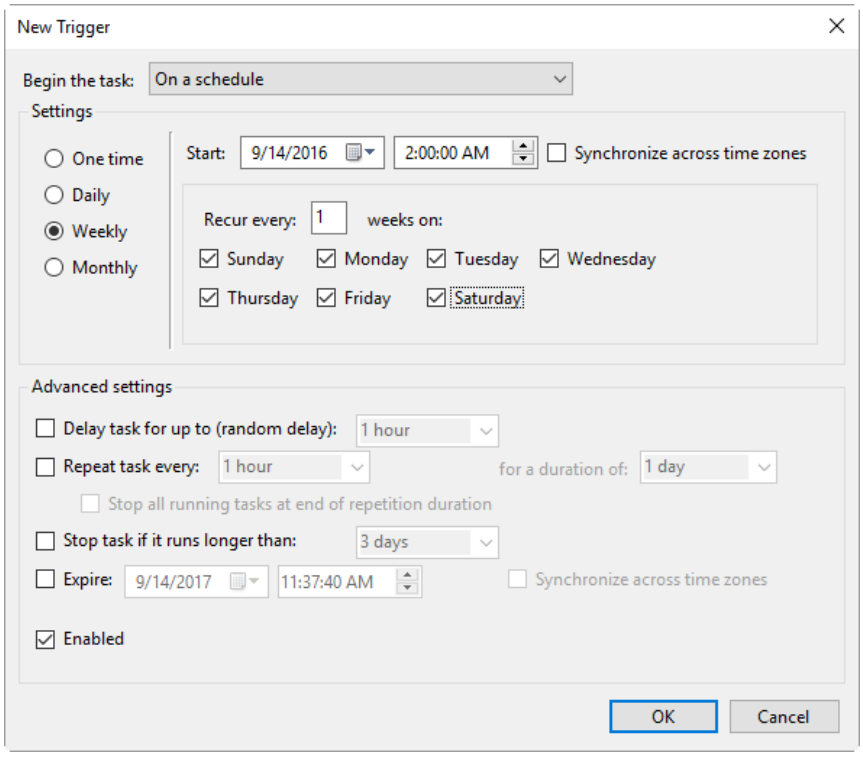
The recommendation is to run daily-backup.ps1 every day at say 2:00 am (or similar time).

Task Basics



Trigger screen

Set the time to run like this:



Action

Program: c:\p4\master\bin\daily-backup.bat

Where <master> is your instance name.

Setup the review script

Use Windows task scheduler to create a p4_Master_review task that runs once per day from 00:00 to 23:50 every 5 minutes. The task should run this command:

c:\p4\common\bin\run_p4review.cmd Master

Saving your configuration files in Perforce

It is sensible to create a Perforce workspace and to store the configuration files in Perforce.

Typically the depot root might be something like //perforce/sdp. If you have many machines, then you might use //perforce/sdp/<machine> using either a physical or a logical name for the machine.

A typical workspace view (e.g. for workspace called `p4admin.sdp` and for instance `master`), might be:

```
Root:  c:\p4
View:
//perforce/sdp/p4/1/bin/... //p4admin.sdp/1/bin/...
//perforce/sdp/p4/common/bin/... //p4admin.sdp/common/bin/...
//perforce/sdp/p4/config/... //p4admin.sdp/config/...
```

You would have appropriate workspaces for each machine, and appropriate lines for each instance on that machine.

Archiving configuration files

Now that the server is running properly, copy the following configuration files to the `depotdata` volume for backup:

- The scheduler configuration.
- Cluster configuration scripts, failover scripts, and disk failover configuration files.

Configuring a New Instance on an existing machine

It is possible to add a new instance to an existing machine.

Edit the sdp_master_config.ini and add a new section for the new instance.

The run SDPEnv.py, specifying to just create the new instance.

```
cd %SDP%\Server\Windows\setup
```



```
notepad sdp_master_config.ini

SDPEnv.py -c sdp_master_config.ini --instance Replica2
```

If the output looks correct then re-run the script with –y parameter to actually perform the copying of files and creation of directories and links.

Upgrading an existing (non SDP) Windows installation

The easiest way to upgrade a service instance is:

1. Create new sdp_master_config.ini file to describe the existing installations.
2. Run SDPEnv.py to create the new environment
3. Run install_services_<hostname>.bat to create new services.
    a. Stop existing services
    b. Manually move the following files from their existing to new locations:
        i. db.* files
        ii. license
        iii. log file(s)
        iv. journal
        v. checkpoints and archived journals
    c. Start new service and check it runs successfully
    d. Adjust the depot root paths (with 2014.1 or greater use the configurable `server.depot.root`, otherwise manually edit depot specs and install the appropriate trigger for new depot specs)

Simple reporting commands to compare before/after include:

- p4 changes –m20 –l –t > changes.txt
- p4 depots > depots.txt
- p4 verify –q //…@yyyy/mm/dd,#head (specifying a few days before the cutover)

Upgrading an older Windows SDP installation

Older versions of the Windows SDP (pre June 2014) stored configuration values for each instance in a p4env.bat file within the p4\common\bat directory.

They also didn't link all directories from c:\p4, but instead used drives such as E:, F: and G: and paths on those drives.

The easiest way to upgrade (most of the work can be done without stopping the service) is:

1. Ensure that existing instance files are checked in to Perforce (instance\bin and common\bin files), for example in workspace p4admin.sdp.orig (use a root directory of %DEPOTDATA% - see next step).
2. Extract existing values from p4env.bat such as mailfrom, mailhost, mailto, and also

METADATA, LOGDATA and DEPOTDATA

3. Edit sdp_master_config.ini and set the appropriate values using extracted ones, and appropriate instance specific values.
4. Set the values for METADATA_ROOT, DEPOTDATA_ROOT and LOGDATA_ROOT to the same (dummy) value, e.g. c:\p4assets
5. Run SDPENV.py to generate the new structure.
6. Manually edit c:\p4assets\p4\config\sdp_config.ini and set the ROOT values to the existing values taken from step 2.
7. Using a different but similar workspace p4admin.sdp.new, which has a root directory of c:\p4, run "p4 sync –k", then do a "p4 reconcile" to identify all the changed files – this will include most of the .bat files, but it shouldn't include p4d.exe or p4s.exe as we are not updating these files.
8. Submit the new changes.
9. In workspace p4admin..sdp.orig, carefully check the updated files that need to be synced (recommend you review diffs one by one), and then sync them.
10. Manually remove and recreate the links (using "del" and "mklink /d") for directories under c:\p4 so that they point to the existing directories on e:, f: or g: (the original DEPOTDATA).
11. Review existing configurables and adjust as appropriate.
12. Setup the scheduled tasks for daily/weekly backup and verify as appropriate. Validate that the daily backup works (typically wait until the next day)
13. At an appropriate point, stop the existing service, adjust the service paths to use the new paths starting from c:\p4.

Configuring protections, file types, monitoring and security

After the server is installed and configured, most sites will want to modify server permissions (protections) and security settings. Other common configuration steps include modifying the file type map and enabling process monitoring. To configure permissions, perform the following steps:

1. To set up protections, issue the `p4 protect` command. The protections table is displayed.
2. Delete the following line:
   ```
   write user * * //depot/...
   ```
3. Define protections for your server using groups. Perforce uses an inclusionary model. No access is given by default, you must specifically grant access to users/groups in the protections table. It is best for performance to grant users specific access to the areas of the depot that they need rather than granting everyone open access, and then trying to remove access via exclusionary mappings in the protect table even if that means you end up generating a larger protect table.
4. To set the server's default file types, run the p4 typemap command and define your typemap to override Perforce's default behavior.

Add any file type entries that are specific to your site. Suggestions:

- For already-compressed file types (such as `.zip, .gz, .avi, .gif`), assign a file type

of `binary+Fl` to prevent the server from attempting to compress them again before storing them.

- For regular binary files, add binary+l to make so that only one person at a time can check them out.
- A sample file is provided in $SDP/Server/config/typemap

5. For large, generated text files (e.g. postscript files), assign the `text+C` file type, to avoid causing server memory issues.
6. If authentication against LDAP or Active Directory is required[2], use the authentication script `AD_auth.pl`, available in `$SDP/Server/common/p4/common/bin/triggers`, as a starting point. (There is also a version of the script called `AD_auth_debug.pl`, which contains additional debugging information.)
   There is also a **PowerShell authentication script** which may be easier to configure than the Perl one. This powershell script is in `$SDP/Server/Windows/p4/common/bin/triggers/ad-auth-check.ps1`
   Note: Perforce provides most IT required password management practices internally. It is recommend to use internal passwords over LDAP/AD to avoid exposing LDAP/AD passwords to the Perforce admin via the auth trigger.

Backup, Replication, and Recovery

Perforce servers maintain *metadata* and *versioned files*. The metadata contains all the information about the files in the depots. Metadata resides in database (`db.*`) files in the server's root directory (`P4ROOT`). The versioned files contain the file changes that have been submitted to the server. Versioned files reside on the `depotdata` volume.

This section assumes that you understand the basics of Perforce backup and recovery. For more information, consult the Perforce System Administrator's Guide and the Knowledge Base articles about replication.

Typical Backup Procedure

The SDP's maintenance scripts, run as *cron* tasks on Unix/Linux or as Windows *scheduled tasks*, periodically back up the metadata. The weekly sequence is described below.

---

[2] The trigger must be customized for your LDAP/AD environment, and then installed in the trigger table; instructions are contained in the script header. Carefully consider the implications of using external authentication, including the fact that users will have to authenticate with the `p4 login` command. Optionally, the script can be modified to recognize purely local accounts. Ask Perforce Consulting for assistance in this area if necessary.

Seven nights a week, perform the following tasks.

1. Rotate/truncate the active journal.
2. Replay the journal to the offline database. (Refer to Figure 2: Volume Layout for more information on the location of the live and offline databases.)
3. Create a checkpoint from the offline database.
4. Recreate the offline database from the last checkpoint.

Once a week, perform the following tasks.

1. Verify all depots.

This normal maintenance procedure puts the checkpoints (metadata snapshots) on the `depotdata` volume, which contains the versioned files. Backing up the `depotdata` volume with a normal backup utility like *robocopy* or *rsync* provides you with all the data necessary to recreate the server.

To ensure that the backup does not interfere with the metadata backups (checkpoints), coordinate backup of the `depotdata` volume using the SDP maintenance scripts.

The preceding maintenance procedure minimizes server downtime, because checkpoints are created from offline or saved databases while the server is running.

Be sure to back up the entire `depotdata` volume using a normal backup utility.

With no additional configuration, the normal maintenance prevents loss of more than one day's metadata changes. To provide an optimal Recovery Point Objective (RPO), the SDP provides additional tools for replication.

Full One-Way Replication

Perforce supports a full one-way replication of data from a master server to a replica, including versioned files. The `p4 pull` command is the replication mechanism, and a replica server can be configured to know it is a replica and use the replication command. The `p4 pull` mechanism requires very little configuration and no additional scripting.

To use full one-way replication, you must:

- Set configuration parameters in the Perforce database for each replica instance. The SDP startup scripts use the replica hostname as the identifying server name. The configuration includes defining the number of `p4 pull` workers, and can be set up such that versioned files are only replicated on demand.
- Configure a service user for replica authentication using `p4 configure`. Note that you will need to execute the `p4 login` command (against the master server) for this service user on the replica server.

- Initialize the replica. Configure the replica as a duplicate of the SDP, and initialize it with a checkpoint and depots from the master server.
- Start the replica and verify that it is operating correctly. Note that `login` commands are forwarded to the master server, so be sure to use `p4 login -a`.

Each of these steps is explained in detail in the server documentation.

As this replication mechanism is simple and effective, we recommend it as the preferred replication technique. Replica servers can also be configured to only contain metadata, which can be useful for reporting or offline checkpointing purposes.

If you wish to use the replica as a read-only server, you can use the P4Broker to direct read-only commands to the replica. Use of the broker may require use of a P4AUTH server for authentication.

Replication handles all server metadata and versioned file content, but not the SDP installation itself or other external scripts such as triggers. Use tools such as `robocopy` or `rsync` to replicate the rest of the `depotdata` volume.

Replication Setup

To configure a replica server, first configure a machine identical to the master server, then install the SDP on it to match the master server installation. We assume that there is a broker in use in this setup. The broker is listening on port 1666 and master server is on port 1667.

Note, it is required that you set P4TICKETS for the service and for the users on the machine to a common location for replication to work. To set this up, run the following on both the master and the replica:

p4 set -s P4TICKETS=c:\p4\1\p4tickets.txt

p4 set -S p4_1 P4TICKETS=c:\p4\1\p4tickets.txt

Once the machine and SDP install is in place, you need to configure the master server for replication. We will assume the following for the setup:

The replica name will be `replica1`, the service user name is `replica1`, and the master server's name is `master`, and the metadata volume is e:, the depotdata volume is f:, and the logs volume is g:. You will run the following commands on the master server:

```
p4 configure set P4TICKETS=c:\p4\1\p4tickets.txt
p4 configure set replica1#P4PORT=1667
p4 configure set replica1#P4TARGET=master:1667
p4 configure set replica1#journalPrefix=c:\p4\1\checkpoints\p4_1
p4 configure set replica1#server=3
p4 configure set "replica1#startup.1=pull -i 1"
```

```
p4 configure set "replica1#startup.2=pull -u -i 1"
p4 configure set "replica1#startup.3=pull -u -i 1"
p4 configure set "replica1#startup.4=pull -u -i 1"
p4 configure set "replica1#startup.5=pull -u -i 1"
p4 configure set "replica1#db.replication=readonly"
p4 configure set "replica1#lbr.replication=readonly"
p4 configure set replica1#serviceUser=replica1
```

The following commands will also need to be run:

- `p4 user -f replica1` (You need to add the Type: service field to the user form before saving)
- `p4 passwd replica1` (Set the service user's password)
- `p4 group service.g` (Add the service user to the Users section and set the timeout to unlimited.)
- `p4 protect` (Give the `replica1` user super user rights to //…)

Now that the setttings are in the master server, you need to create a checkpoint to seed the replica. Run:

c:\p4\commond\bin\daily-backup.ps1 1

When the checkpoint finishes, copy the checkpoint plus the versioned files over to the replica server. You can use xcopy or something like robocopy for this step.

```
xcopy c:\p4\1\checkpoints\p4_1.ckp.###.gz replica_f_drive:\p4\1\checkpoints

xcopy c:\p4\1\depots replica_f_drive:\p4\1\depots /S
```

(### is the checkpoint number created by the daily backup)

Once the copy finishes, go to the replica machine run the following:

- c:\p4\1\bin\p4d -r c:\p4\1\root -jr -z c:\p4\1\checkpoints\p4_1.ckp.###.gz
- c:\p4\1\bin\p4 -p master:1667 -u replica1 login (enter the service user's password)
- net start p4_1

Now, you check the log on the master server (`c:\p4\1\logs\log`) to look for the rmt-Journal entries that show you the replication is running. If you see those entries, then you can make some changes on the master server, and then go to the replica server and check to see that they changes were replicated across. For example, you can submit a change to the master server, then go to the replica server and check to see that the change was replicated over to the replica by running p4 describe on the changelist against the replica server.

The final steps for setting up the replica server are to set up the task scheduler to run the replica sync scripts. This has to be done via task scheduler running as a regular AD user so that the scripts can access the network in order to get to the drives on the replica machine.

You need to configure a task to run `c:\p4\common\bin\sync-replica.ps1 <instance>` every

day. The task should be set up to run after the master server finishes running `daily-backup.ps1`. Be sure to give it some buffer for the length of time it takes the master to run that script is likely to become gradually longer over time.

Recovery Procedures

There are three scenarios that require you to recover server data:

Metadata	Depotdata	Action required
lost or corrupt	intact	Recover metadata as described below
Intact	lost or corrupt	Call Perforce Support
lost or corrupt	lost or corrupt	Recover metadata as described below. Recover the depotdata volume using your normal backup utilities.

Restoring the metadata from a backup also optimizes the database files.

Recovering from a checkpoint and journal(s)

The checkpoint files are stored in the `c:\p4\instance\checkpoints` directory, and the most recent checkpoint is named `p4_instance.ckp.number.gz`. Recreating up-to-date database files requires the most recent checkpoint, from `c:\p4\instance\checkpoints`, and the journal file from `c:\p4\instance\logs`.

To recover the server database manually, perform the following steps from the root directory of the server (`c:\p4\instance\root`).

1. Stop the Perforce Server by issuing the following command:
   ```
   c:\p4\instance\bin\p4_1 admin stop
   ```
2. Delete the old database files in `c:\p4\instance\root\save` directory (note there may not be any files there as they will typically be cleared out **after** successful completion of the previous invocation of the recovery process – see below).
3. Move the live database files (`db.*`) to the save directory.
4. Use the following command to restore from the most recent checkpoint.
   ```
   c:\p4\instance\bin\p4d_instance -r c:\p4\instance\root -jr -z
   c:\p4\instance\checkpoints\p4_instance.ckp.most recent_#.gz
   ```
5. To replay the transactions that occurred after the checkpoint was created, issue the following command:
   ```
   c:\p4\instance\bin\p4d_instance -r c:\p4\instance\root -jr
   c:\p4\instance\logs\journal
   ```
6. Restart your Perforce server.

If the Perforce service starts without errors, delete the old database files from

`c:\p4\instance\root\save`.

If problems are reported when you attempt to recover from the most recent checkpoint, try recovering from the preceding checkpoint and journal. If you are successful, replay the subsequent journal. If the journals are corrupted, contact Perforce Technical Support. For full details about back up and recovery, refer to the Perforce System Administrator's Guide.

Recovering from a tape backup

This section describes how to recover from a tape or other offline backup to a new server machine if the server machine fails. The tape backup for the server is made from the `depotdata` volume. The new server machine must have the same volume layout and user/group settings as the original server. In other words, the new server must be as identical as possible to the server that failed.

To recover from a tape backup, perform the following steps.

1. Recover the `depotdata` volume from your backup tape.
2. As a super-user, reinstall and enable the Windows services that run the Perforce instance.
3. Find the last available checkpoint, under `c:\p4\instance\checkpoints`.
4. Recover the latest checkpoint by running:

```
c:\p4\instance\bin\p4d_instance -r c:\p4\instance\root -jr -z last_ckp_file
```

5. Recover the checkpoint (as shown in the preceding step) into the `offline_db` directory rather than the `root` directory.

```
c:\p4\instance\bin\p4d_instance -r c:\p4\instance\offline_db -jr -z
last_ckp_file
```

6. Reinstall the Perforce server license to the server `root` directory.
7. Start the Perforce service.
8. Verify that the server instance is running.
9. Reinstall the server crontab or scheduled tasks.
10. Perform any other initial server machine configuration.
11. Verify the database and versioned files by running the p4verify script. Note that files using the +k file type modifier might be reported as BAD! after being moved. Contact Perforce Technical Support for assistance in determining if these files are actually corrupt.

Failover to a replicated standby machine

See DR-Failover-steps-windows.docx

Server Maintenance

This section describes typical maintenance tasks and best practices for administering server machines. The directory `c:\p4\sdp\Maintenance` contains scripts for several common maintenance tasks.

The user running the maintenance scripts must have administrative access to Perforce for most activities. All of these scripts can be run from any client machine.

Server upgrades

Upgrading a server instance in the SDP framework is a simple process involving a few steps.

- Download the new `p4` and `p4d` executables from `ftp.perforce.com` and place them in `<depotdata>:\p4\common\bin`
- Run `<depotdata>:\p4\common\bin\upgrade.ps1 <instance>` (Please note that the upgrade to and thru version 2013.3 requires different processing, and the `2013.3-upgrade.ps1` script must instead be used).

Database Modifications

Occasionally modifications are made to the Perforce database. For example, server upgrades and some recovery procedures modify the database.

When upgrading the server, replaying a journal patch, or performing any activity that modifies the `db.*` files, you must restart the offline checkpoint process so that the files in the `offline_db` directory match the ones in the live server directory. The easiest way to restart the offline checkpoint process is to run the `live-checkpoint` script after modifying the `db.*` files, as follows:

`C:\p4\common\bin\live-checkpoint.ps1` *`instance`*

This script makes a new checkpoint of the modifed database files in the live root directory, then recovers that checkpoint to the `offline_db` directory so that both directories are in sync. This script can also be used anytime to create a checkpoint of the live database.

This command must be run when an error occurs during offline checkpointing. It restarts the offline checkpoint process from the live database files to bring the offline copy back in sync. If the live checkpoint script fails, contact Perforce Consulting at consulting@perforce.com.

Listing inactive specifications

To list branch specifications, clients, labels and users that have been inactive for a specified number of weeks, run `accessdates.py`. This script generates four text files listing inactive specifications:

- `branches.txt`

- `clients.txt`
- `labels.txt`
- `users.txt`

Unloading and Reloading labels

Archiving labels is a best practice for large installations, with hundreds of users and Perforce checkpoints that are gigabytes in size. Smaller sites need not necessarily concern themselves with archiving labels to maintain performance, though doing so will minimize database size if labels are used extensively.

To use the `p4 unload` and `p4 reload` commands for archiving clients and labels, you must first create an unload depot using the `p4 depot` command. Run:

```
p4 depot unload
```

Set the type of the depot to `unload` and save the form.

After the depot is created, you can use the following command to archive all the clients and labels that have been accessed since the given date:

```
p4 unload -f -L -z -a -d <date>
```

For example, to unload all clients and labels that haven't been accessed since Jan. 1, 2013, you would run:

```
p4 unload -f -L -z -a -d 2013/01/01
```

Users can reload their own clients/labels using the reload command. They can run:

```
p4 reload -c <clientname>
```

or

```
p4 reload -l <labelname>
```

As a super user, you can reload and unloaded item by adding the -f flag to the reload command as follows:


```
p4 reload -f -c|l <specname>
```

In addition, you can avoid having to unload/reload labels by creating a trigger to set the autoreload option as the default on all new labels. That will cause the server to use the unload depot for storing the labels rather than storing them in db.label. This helps with performance of the server by not increasing the size of the database for label storage.

You can automate these tasks with `$SDP/Maintenance/unload_clients.py` and `$SDP/Maintenance/unload_labels.py`

Archiving client workspaces and branch specifications

To delete clients and branch specs that have not been accessed recently, run the `archive_clients_and_branches.py` script. This script deletes clients and branch specifications that have not been accessed in the number of weeks specified by the variable `weeks`.

Deleting users

To delete users, run `python p4deleteuser.py`, specifying the users to be deleted. The script deletes the users, any workspaces they own, and removes them from any groups they belong to.

To delete all users that have not accessed the server in the past 12 weeks, run `python delusers.py`. To change the number of weeks to a value other than 12, change the `weeks` variable in the `p4deleteuser.py` module.

To remove a specified user from all groups it belongs to, run `python removeuserfromgroups.py`. Specify the user name or the name of a text file containing a list of users to be removed.

Listing users

To display a list of users that are in a group but do not have an account on the server, run `python checkusers.py`.

To display the number of users defined in your server, run `python count_users.py`.

Group management

To duplicate a specified user's group entries on behalf of another user, run `python mirroraccess.py`. The script adds the target user to all groups that the source user belongs to. Invoke the script as follows:

```
python mirroraccess.py sourceuser targetuser
```

To add users to a group, run

```
python addusertogroup.py user group
```

where:

- *`user`* = user name or a file containing a list of user names, one per line.

- *group* = name of Perforce group to which the users are added.

Adding users

To add users to a server:

1. Create a text file, such as `/temp/user.txt`, containing the users to add, specifying one user name per line.
2. In the `createusers.py` script, replace `yourcompany.com` with the company name.
3. In the `setpass.py` script, set the default password to assign to users.
4. Verify that the Perforce group called `limits` exists and has the desired `maxscanrows`, `maxresults`, `maxlocktime`, and `timeout` settings.
5. Run `addusers`.

The script creates the users, sets their initial passwords, and adds the users to the `limits` group.

To add users without adding them to a group:

1. Modify the `createusers.py` script to specify the correct address for your company.
2. Invoke the script as follows:

```
python createusers.py user_or_file
```

On the command line, specify the user name or a text file containing one user name per line.

To set the password for a Perforce user, set the `password` variable in the `setpass.py` script. Then run the script, specifying the user to be changed, as follows:

```
python setpass.py username
```

Email functions

To list the email addresses of your Perforce users, run `python make_email_list.py`.

Workspace management

The form-out trigger `$SDP/Server/common/p4/common/bin/triggers/SetWsOptions.py` contains default workspace options, such as `leaveunchanged` instead of `submitunchanged`.

To use the trigger, first copy it to `/p4/common/bin/triggers`

To enable the trigger, first modify the `OPTIONS` variable in the script, providing the set of desired options. Then insert an entry in the trigger table like the following:

```
setwsopts form-out client "python /p4/common/bin/triggers/SetWsOptions.py %formfile%"
```

The form-save trigger `$SDP/Server/common/p4/common/bin/triggers/PreventWsNonAscii.py` enforces the policy that no workspaces may contain non-ASCII characters.

To use the trigger, first copy it to `/p4/common/bin/triggers`

To enable the trigger, insert an entry in the trigger table like the following:


```
nowsascii form-save client "python
/p4/common/bin/triggers/PreventWsNonAscii.py %formfile%"
```

Removing empty changelists

To delete empty pending changelists, run `python remove_empty_pending_changes.py`.

Maximizing Server Performance

The following sections provide some guidelines for maximizing the performance of the Perforce Server, using tools provided by the SDP. More information on this topic can be found in the System Administrator's Guide and in the Knowledge Base.

Optimizing the database files

The Perforce Server's database is composed of b-tree files. The server does not fully rebalance and compress them during normal operation. To optimize the files, you must checkpoint and restore the server. The weekly checkpoint script used as part of the normal server maintenance automates this task.

To minimize the size of back up files and maximize server performance, minimize the size of the `db.have` and `db.label` files. The scripts described in Unloading and Reloading labels, Deleting users, and Unloading and Reloading labels

Archiving labels is a best practice for large installations, with hundreds of users and Perforce checkpoints that are gigabytes in size. Smaller sites need not necessarily concern themselves with archiving labels to maintain performance, though doing so will minimize database size if labels are used extensively.

To use the `p4 unload` and `p4 reload` commands for archiving clients and labels, you must first create an unload depot using the `p4 depot` command. Run:

```
p4 depot unload
```

Set the type of the depot to `unload` and save the form.

After the depot is created, you can use the following command to archive all the clients and labels that have been accessed since the given date:

```
p4 unload -f -L -z -a -d <date>
```

For example, to unload all clients and labels that haven't been accessed since Jan. 1, 2013, you would run:

```
p4 unload -f -L -z -a -d 2013/01/01
```


Users can reload their own clients/labels using the reload command. They can run:

```
p4 reload -c <clientname>
```

or

```
p4 reload -l <labelname>
```

As a super user, you can reload and unloaded item by adding the -f flag to the reload command as follows:

```
p4 reload -f -c|l <specname>
```

In addition, you can avoid having to unload/reload labels by creating a trigger to set the autoreload option as the default on all new labels. That will cause the server to use the unload depot for storing the labels rather than storing them in db.label. This helps with performance of the server by not increasing the size of the database for label storage.

You can automate these tasks with `$SDP/Maintenance/unload_clients.py` and `$SDP/Maintenance/unload_labels.py`

Archiving client workspaces and branch specifications help achieve this goal. For best server performance, run these scripts frequently.

Managing server load

Limiting large requests

To prevent large requests from overwhelming the server, you can limit the amount of data and time allowed per query by setting the `maxresults`, `maxscanrows` and `maxlocktime` parameters to the lowest setting that does not interfere with normal daily activities. As a good starting point, set `maxscanrows` to `maxresults` * 3; set `maxresults` to slightly larger than the maximum number of files the users need to be able to sync to do their work; and set `maxlocktime` to 30000 milliseconds. These values must be adjusted up as the size of your server and the number of revisions of the files grow. To simplify administration, assign limits to groups rather than individual users.

To prevent users from inadvertently accessing large numbers of files, define their client view to be as narrow as possible, considering the requirements of their work. Similarly, limit users' access in the protections table to the smallest number of directories that are required for them to do their job.

Finally, keep triggers simple. Complex triggers increase load on the server.

Offloading remote syncs

For remote users who need to sync large numbers of files, Perforce offers a proxy server. P4P, the Perforce Proxy, is run on a machine that is on the remote users' local network. The Perforce Proxy caches file revisions, serving them to the remote users and diverting that load from the main server.

P4P is included in the Windows installer.

P4P does not require special hardware because it doesn't use much processing power, and it doesn't need to be backed up. If the P4P instance isn't working, users can switch their port back to the main server and continue working until the instance of P4P is fixed.

P4V performance settings

At large sites with hundreds or thousands of simultaneous users, the P4V data retrieval settings can help prevent P4V requests from impacting server performance. As of the 2010.1 release, P4V settings that affect performance can be centrally managed for all users or specific groups of users, using the JavaScript API (P4JsApi).

The SDP includes a sample P4V settings file, along with the P4JsApi `centralsettings` file that enables it. These files are located in `//Perforce/sdp/JsApi`.

Follow these steps to provide P4V performance settings for your users.

1. Determine whether you want P4V settings common to all users, or different settings for different groups. If the latter, make a unique copy of `//Perforce/sdp/JsApi/p4vsettings.xml` for each group of users. For example, you may create `//Perforce/sdp/JsApi/p4vsettings_dev.xml` for developers and `//Perforce/sdp/JsApi/p4vsettings_qa.xml` for QA.
2. Review and set the performance limits in `//Perforce/sdp/JsApi/p4vsettings.xml`, or in each copy of this file. (The file contains suggested default values.) The available settings are:
    a. The `ServerRefresh` interval in minutes, which defines how often P4V attempts to get updated information from the server.
    b. The `MaxFiles` that P4V will retrieve for one fetch command.
    c. The `MaxFilePreviewSize` in kilobytes.
    d. The `FetchCount`, which affects the number of forms fetched for some operations.
3. If using common settings for all users, proceed with this step; otherwise proceed to the next step. Install the `centralsettings` file by adding a line to the protections table like:

```
list group All.G centralsettings //Perforce/sdp/JsApi/centralsettings.js
```

(This line assumes that you have a group called `All.G` that represents all users.)

4. (Skip this step if using common settings for all users.) If using different settings for different groups, create a copy of `//Perforce/sdp/JsApi/centralsettings.js` for each group of users. For example, you may create `//Perforce/sdp/JsApi/centralsettings_dev.js` for

developers and `//Perforce/sdp/JsApi/centralsettings_qa.js` for QA. Modify the line that references `p4vsettings.xml` to reference the copy for the group.

5. (Skip this step if using common settings for all users.) Install each copy of `centralsettings.js` in the protections table. In our example with separate copies for developers and QA, we would use lines like:

```
list group Dev.G centralsettings //Perforce/sdp/JsApi/centralsettings_dev.js

list group QA.G centralsettings //Perforce/sdp/JsApi/centralsettings_qa.js
```

6. Each P4V user must follow the instructions in the P4JsApi manual to enable P4V extensions.

Of course, the P4JsApi provides many other valuable features. If you choose to use these features, you can use the same `centralsettings` files for your groups to enable them. Refer to the P4JsApi manual for details.

Tools and Scripts

This section describes the various scripts and files provided as part of the SDP package. To run these scripts, the machine must have Python, grep, and Perforce installed. The server scripts are specific to your server platform (Unix/Linux or Windows) The maintenance scripts can be run from the server machine or from client machines.

Server maintenance scripts reside in the depot in the following locations, according to platform:

- `c:\p4\sdp\Server\Windows`

The following sections describe the scripts in detail.

Standard scripts

These scripts are provided for both Windows and Unix. Unix scripts are typically shell scripts (`.sh` files) and Windows scripts are typically Powershell scripts (`.ps1` files) – usually with a simple wrapper of a `.bat` which just shows how to call the corresponding `.ps1`.

Please note that not all scripts are available for Windows – contact **consulting@perforce.com** if you want to know more.

Daily-backup

This script is configured to run six days a week using crontab or the Windows scheduler. The script truncates the journal, replays it into the `offline_db` directory, creates a new checkpoint from the resulting database files, then recreates the `offline_db` directory from the new checkpoint.

This procedure rebalances and compresses the database files in the `offline_db` directory, which are rotated into the live database directory once a week by the `weekly_checkpoint` script.

Location:

- `c:\p4\common\bin`

live-checkpoint

Stops the server, creates a checkpoint from the live database files, recovers from that checkpoint to rebalance and compress the files, then recovers the checkpoint in the `offline_db` directory to ensure that the database files are optimized.

Run this script when creating the server and if an error occurs while replaying a journal during the off-line checkpoint process.

Location:

- `c:\p4\common\bin`

p4review.py

Sends out email containing the change descriptions to users who are configured as reviewers for affected files (done by setting the `Reviews:` field in the user specification). This script is a version of the `p4review.py` script that is available on the Perforce Web site, but has been modified to use the server instance number. It relies on a configuration file in the same location, called `p4review.cfg`. On Windows, a driver called `run_p4review.cmd`, located in the same directory, allows you to run the review daemon through the Windows scheduler.

Location:

- `c:\p4\common\bin`

p4verify

Verifies the integrity of the depot files. This script is run by `crontab or Windows scheduler`. It emails the resulting report – please check for any errors contained.

Location:

- `c:\p4\common\bin`

p4verify-incremental

Verifies the integrity of the depot files. This version incrementally verifies one directory at a time, which may reduce the load that verification places on the server at large sites. It uses the `p4verify.pl` script.

Location:

- `c:\p4\common\bin`

create-offline-db-from-checkpoint

On occasion you may wish to recreate the offline_db from the latest checkpoint. This script does that looking for the checkpoint with the most recent date/time. Specifically, this script does the following:

1. Removes db.* from offline_db
2. Finds and replays latest checkpoint to offline_db
3. Replays any later journals to `offline_db`.

Location:

- `c:\p4\common\bin`

Recreate-live-from-offline-db

This script can be scheduled to run every few months – it used to be run weekly, but that is no longer best practice since database files are not fragemented as they used to be. It will move the db.* files from offline to live root (so requires stopping the service).

Location:

- `c:\p4\common\bin`

Rotate-log-files

This script is intended to be run nightly on a replica which may not have any other scheduled tasks running. It ensures that the log files are appropriately rotated and old logs (and journals) are deleted according the settings of KEEP_CKPS in sdp_config.ini

Location:

- `c:\p4\common\bin`

Create-filtered-edge-checkpoint

This creates a checkpoint from the offline_db files, filtered for use with an edge server. Please see the script header for more details and for limitations of filter handling. See also partner script recover-edge which replays the checkpoint on the edge server machine.

Recover-edge

This recreates an edge server from create-filtered-edge-checkpoint, maintaining local data such as workspaces (in edge specific db.have table) plus the other 6 edge tables.

See the script for more details.

P4login

Executes a `p4 login` command, using the password configured in `sdp_master_config.ini` and stored in a text file.

Location:

- `c:\p4\common\bin`

totalusers.py

Calculates the total number of users in Perforce. The script works across multiple servers, removing duplicates and accounting for non-human accounts. This script is particularly useful for large sites with several server instances. Options are specified in totalusers.cfg.

Location: `c:\p4\sdp\Maintenance`

remove_jobs.py

Deletes any jobs (and their fixes) listed in a text file passed as an argument.

Location: `c:\p4\sdp\Maintenance`

p4lock.py and p4unlock.py

Lock or unlock a label.

- **Location**: `c:\p4\sdp\Maintenance`

isitalabel.py

Determines if a label exists.

- **Location**: `c:\p4\sdp\Maintenance`

email_pending_user_deletes.py and email_pending_client_deletes.py

Notifies users via email that their user account or client workspace will be deleted soon, due to inactivity.

- **Location**: `c:\p4\sdp\Maintenance`

delclients.py

Deletes workspaces listed in a file. Used in conjunction with `accessdates.py`.

- **Location**: `c:\p4\sdp\Maintenance`

del_shelve.py

Deletes workspaces that contain shelved files and have not been accessed for a period of time.

- **Location**: `c:\p4\sdp\Maintenance`

countrevs.py

Counts the total number of revisions for a set of files.

- **Location**: `c:\p4\sdp\Maintenance`

Maintain_user_from_groups.py

Synchronizes user and group membership.

- **Location**: `c:\p4\sdp\Maintenance`

The following tools are located in `c:\p4\common\bin`.

instsrv.exe

Executable that create services on Windows.

srvany.exe

Executable that runs an application that wasn't designed to be a service as a service on Windows. Can be used to run `p4review.py` as a service.

Other Files

The following table describes other files in the SDP distribution. These files are usually not invoked directly by you; rather, they are invoked by higher-level scripts.

File	Location	Remarks
dummy_ip.txt	$SDP/Server/config	Instructions for using a license on more than one machine. Typically used to enable a standby server. Contact Perforce Licensing before using.
blat.exe, blat.dll	C:\p4\common\bin	Windows only. Email utility.
grep.exe, e.exe, gzip.exe	C:\p4\common\bin	Windows only. Various utilities.
SDP-functions.ps1	/p4/common/bin	Windows only. Utility functions for maintenance scripts.
change.txt	$SDP/Maintenance	Template for new pending changelist.

Appendix A – Frequently Asked Questions

This appendix lists common questions and problems encountered by SDP users. Do not hesitate to contact consulting@perforce.com if additional assistance is required.

Journal out of sequence

This error is encountered when the offline and live databases are no longer in sync, and will cause the offline checkpoint process to fail. This error can be fixed by running the `create-offline-db-from-checkpoint` (or if that doesn't work then `live-checkpoint` script – which blocks live server), as described in Server upgrades.