Server Deployment Package (SDP) for Perforce Helix

SDP User Guide (for Unix)

Perforce Professional Services

Version v2019.3, 2020-08-05

Table of Contents

Preface

The Server Deployment Package (SDP) is the implementation of Perforce's recommendations for operating and managing a production Perforce Helix Core Version Control System. It is intended to provide the Helix Core administration team with tools to help:

- Simplify Management
- High Availability (HA)
- Disaster Recovery (DR)
- Fast and Safe Upgrades
- Production Focus
- Best Practice Configurables
- Optimal Performance, Data Safety, and Simplified Backup

This guide is intended to provide instructions of setting up the SDP to help provide users of Helix Core with the above benefits.

This guide assumes some familiarity with Perforce and does not duplicate the basic information in the Perforce user documentation. This document only relates to the Server Deployment Package (SDP) all other Helix Core documentation can be found here: Perforce Support Documentation

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.

Chapter 1. Overview

The SDP has four main components:

- Hardware and storage layout recommendations for Perforce.
- Scripts to automate critical maintenance activities
- Scripts to aid the setup and management of replication (including failover for DR/HA)
- Scripts to assist with routine administration tasks.

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

1.1. Using this Guide

Section 2 consists of what you need to know to setup Helix Core sever on a Unix platform.

Section 3 gives information around the Backup, Restoration and Replication of Helix Core.

Section 4 is about Server Maintenance, upgrades and some performance tips.

Section 5 covers the scripts used within the SDP in some more detail.

1.2. Getting the SDP

The SDP is downloaded as a single zipped tar file the latest version can be found at: https://swarm.workshop.perforce.com/projects/perforce-software-sdp/files/downloads

Chapter 2. Setting up the SDP

This section tells you how to configure the SDP to setup a new Helix Core server. Whilst the standard installation of Helix Core is fully covered in the System's Administrator Guide this section 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 Helix Core server instances.

2.1. Terminology and pre-requisites

1. The term *server* refers to a Helix Core server *instance,* unless otherwise specified.
2. The term *metadata* refers to the Helix Core database files
3. *Instance:* a separate Helix Core instantiation using its own p4d daemon/process

Pre-Requisites:

1. The Helix Core binaries (p4d, p4, p4broker, p4p) have been downloaded (see section XXXX)
2. *sudo* access is required
3. System administrator available for configuration of drives / volumes (especially if on network or SAN or similar)
4. Supported Unix version, currently these versions are fully supported - for other versions please speak with Perforce Support
    - Ubuntu 16.04 LTS (xenial)
    - Ubuntu 18.04 LTS (bionic)
    - CentOS or Red Hat 6.x
    - CentOS or Red Hat 7.x
    - SUSE Linux Enterprise Server 12

2.2. Volume Layout and Hardware

As can be expected from a version control system, good disk (storage) management is key to maximising data integrity and performance. Perforce recommend using multiple physical volumes for **each** server instance. Using three or four volumes per instance reduces the chance of hardware failure affecting more than one instance. When naming volumes and directories the SDP assumes the "hx" prefix is used to indicate Helix volumes (your own naming conventions/standards can be used instead). For optimal performance on UNIX machines, the XFS file system is recommended but not mandated.

- **Perforce metadata (database files), 1 or 2 volumes:** Use the fastest volume possible, ideally SSD or RAID 1+0 on a dedicated controller with the maximum cache available on it. These volumes default to `/hxmetadata1` and `/hxmetadata2`.

  It is fine to have these both pointing to the same physical volume, e.g. `/hxmetadata`.

- **Journals and logs:** a fast volume, ideally SSD or RAID 1+0 on its own controller with the standard amount of cache on it. This volume is normally called `/hxlogs` and should usually be backed up.

  If a separate logs volume is not available, put the logs on the `/hxmetadata1` or `/hxmetadata` 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 (NFS access is fine). This volume is the only volume that **must** be backed up. The SDP backup scripts place the metadata snapshots on this volume.

  This volume is normally called `/hxdepots`.

If multiple controllers are not available, put the `hxlogs` and `hxdepots` volumes on the same controller.

Do not run anti-virus tools or back up tools against the hxmetadata volume(s) or hxlogs volume(s), because they can interfere with the operation of the Perforce server.

On Unix/Linux platforms, the SDP will create a "convenience" directory containing links to the volumes for each instance, by default named `/p4`. The volume layout is shown in Appendix SDP Package Contents. This convenience directory enables easy access to the different parts of the file system for each instance.

For example:

- `/p4/1/root` contains the database files for instance `1`
- `/p4/1/logs` contains the log files for instance `1`
- `/p4/1/bin` contains the binaries and scripts for instance `1`
- `/p4/common/bin` contains the binaries and scripts common to all instances

Chapter 3. Installing the SDP on Unix / Linux

To install Perforce Server and the SDP, perform the steps laid out below:

- Set up a user account, file system, and configuration scripts.
- Run the configuration script.
- Start the server and configure the required file structure for the SDP.

1. If it doesn't already exist, create a group called `perforce`:

```
sudo groupadd perforce
```

2. Create a user called perforce and set the user's home directory to `/p4` on a local disk.

```
sudo useradd -d /p4 -s /bin/bash -m perforce -g perforce
```

3. Create or mount the server file system volumes (per layout in previous section)
   - /hxdepots
   - /hxmetadata1
   - /hxmetadata2
   - /hxlogs
4. These directories should be owned by: `perforce:perforce`

```
sudo chown -R perforce:perforce /hx*
```

5. Either download the SDP directly or move the previously downloaded version to `/hxdepots`

```
cd /hxdepots
```

```
wget
https://swarm.workshop.perforce.com/downloads/guest/perforce_software/sdp/downloads
/sdp.Unix.2019.3.26571.tgz
```

   Or:

```
mv sdp.Unix.2019.3.26751.tgz /hxdepots
```

6. Untar and uncompress the downloaded sdp files:

```
tar -zxvf sdp.Unix.2019.3.26751.tgz
```

7. Set environment variable SDP, this makes certain later steps easier.

```
export SDP=/hxdepots/sdp
```

8. Make the entire $SDP (`/hxdepot/sdp`) directory writable:

```
chmod -R +w $SDP
```

9. Download the appropriate p4, p4d and p4broker binaries for your release and platform (substituting desired release for `r20.1` below)

```
cd $SDP/Server/Unix/p4/common/bin
wget http://ftp.perforce.com/perforce/r20.1/bin.linux26x86_64/p4
wget http://ftp.perforce.com/perforce/r20.1/bin.linux26x86_64/p4d
wget http://ftp.perforce.com/perforce/r20.1/bin.linux26x86_64/p4broker
```

10. make them executable

```
chmod +x p4*
```

3.1. Initial setup

The next steps highlight the setup and configuration of a new Helix Core instance using the SDP.

1. cd to `$SDP/Server/Unix/setup` and copy `mkdirs.cfg` to an instance specific version such as `mkdirs.1.cfg` and edit it, information on the variables can be found in section [Configuration Script] of this document

```
Example:
```

```
cd $SDP/Server/Unix/setup
cp mkdirs.cfg mkdirs.1.cfg
vi mkdirs.1.cfg
```

Set the following:

```
P4ADMINPASS=********
MAILFROM=perforceadmin@myDomain.com
MAILHOST=myMailServer.myDomain.com
P4DNSNAME=thisMachine.myDomain.com
P4SERVICEPASS=********
MASTER_ID=myName.$\{SDP_INSTANCE}
```

2. As the root user (or sudo), run this:

```
mkdirs.sh _<instance number/name>_
```

e.g.

```
mkdirs.sh 1
```

If you use a name for the instance you MUST modify the P4PORT variable in `mkdirs.cfg`

instance must map to the name of the cfg file or the default file will be used with potentially unexpected results.

+ e.g. `mkdirs.sh 1` requires `mkdirs.1.cfg`, or `mkdirs.sh lon` requires `mkdirs.lon.cfg`, or

1. Put the Perforce license file for the server into `/p4/1/root`

if you have multiple instances and have been provided with port-specific licenses by Perforce, the appropriate license file must be stored in the appropriate `/p4/instance/root` folder.

the license file must be renamed to `license`

Your Helix Core instance is now setup, but not running. The next steps detail how to make the Helix Core server a system service.

3.2. For Systems using systemd

RHEL 7, CentOS 7, SuSE 12, Ubuntu (>= v16.04) (and other) distributions utilize **systemd** / **systemctl** as the mechanism for controlling services, replacing the earlier init process. At present mkdirs.sh does **not** generate the systemd configuration file(s) automatically, but a sample is included in the SDP distribution in (`$SDP/Server/Unix/setup/systemd`), along with a README.md file that describes the configuration process, including for multiple instances.

We recommend that you give the OS user (perforce) sudo access, so that it can run the commands below prefixing them with sudo.

For simple installation run these commands as the root user (or prefix with `sudo`):

```
cp $SDP/Server/Unix/setup/system/p4d_1.system /etc/systemd/system/
systemctl enable p4d_1
```

|The above enables service for auto-start on boot. The following show management commands:

```
systemctl status p4d_1
systemctl start p4d_1
systemctl status p4d_1
systemctl stop p4d_1
systemctl status p4d_1
```

3.3. For (older) systems, still using init.d

The mkdirs.sh script creates a set of startup scripts in the instance-specific bin folder:

```
/p4/1/bin/p4d_1_init
/p4/1/bin/p4broker_1_init     # only created if a p4broker executable found
/p4/1/bin/p4p_1_init          # only created if a p4p executable found
```

Run these commands as the root user (or sudo): Repeat this step for all init scripts you wish to add.

```
cd /etc/init.d
ln -s /p4/1/bin/p4d_1_init
chkconfig --add p4d_1_init
chkconfig p4d_1_init on
```

Chapter 4. Upgrading an existing SDP installation

If you have an earlier version of the Server Deployment Package (SDP) installed, you'll want to be aware of the new -test flag to the SDP setup script, mkdirs.sh e.g.

```
sudo mkdirs.sh 1 -test
```

This will install into /tmp and allow you to recursively diff the installed files with your existing installation and manually update as necessary.

See the instructions in the file README.md / README.html in the root of the SDP directory.

Chapter 5. Configuration script

The mkdirs.sh script executed above resides in `$SDP/Server/Unix/setup`. It sets up the basic directory structure used by the SDP. Carefully review the config file mkdirs.*instance*.cfg for this script before running it, and adjust the values of the variables as required. The important parameters are:

Parameter	Description
DB1	Name of the hxmetadata1 volume (can be same as DB2)
DB2	Name of the hxmetadata2 volume (can be same as DB1)
DD	Name of the hxdepots volume
LG	Name of the hxlogs volume
CN	Volume for /p4/common
SDP	Path to SDP distribution file tree
SHAREDDATA	TRUE or FALSE - whether sharing the /hxdepots volume with a replica - normally this is FALSE
ADMINUSER	P4USER value of a Perforce super user that operates SDP scripts, typically perforce or p4admin.
OSUSER	Operating system user that will run the Perforce instance, typically perforce.
OSGROUP	Operating system group that OSUSER belongs to, typically perforce.
CASE_SENSITIVE	Indicates if server has special case sensitivity settings
SSL_PREFIX	Set if SSL is required so either "ssl:" or blank for no SSL
P4ADMINPASS P4SERVICEPASS	Password to use for Perforce superuser account - can be edited later in /p4/common/config/.p4password.p4_1.admin Service User's password for replication - can be edited later - same dir as above.
P4DNSNAME	Fully qualified DNS name of the Perforce master server machine

For a detailed description of this config file it is fully documented with in-file comments, or see

5.1. Use of SSL

As documented in the comments in mkdirs.cfg, if you are planning to use SSL you need to set the value of:

```
SSL_PREFIX=ssl:
```

Then you need to put certificates in `/p4/ssl` after the SDP install or you can generate a self signed certificate as follows:

Edit `/p4/ssl/config.txt` to put in the info for your company. Then run:

```
/p4/common/bin/p4master_run <instance> /p4/<instance>/p4d_<instance> -Gc
```

For example using instance 1:

```
/p4/common/bin/p4master_run 1 /p4/1/bin/p4d_1 -Gc
```

In order to validate that SSL is working correctly:

```
source /p4/common/bin/p4_vars 1
```

Check that P4TRUST is appropriately set in the output of:

```
p4 set
```

Update the P4TRUST values (answer yes when prompted):

```
p4 -p ssl:1666 trust
```

```
p4 -p ssl:localhost:1666 trust
```

Check the stored P4TRUST values:

```
p4 trust -l
```

Check not prompted for trust:

```
p4 login
p4 info
```

5.1.1. Starting/Stopping Perforce Server Products

The SDP includes templates for initialization (start/stop) scripts, "init scripts," for a variety of Perforce server products, including:

- p4d
- p4broker
- p4p
- p4dtg
- p4ftpd
- p4web

The init scripts are named `/p4/<instance>/bin/<service>_<instance>_init`, e.g. `/p4/1/bin/p4d_1_init` or `/p4/1/bin/p4broker_1_init`.

For example, the init script for starting p4d for Instance 1 is `/p4/1/bin/p4d_1_init`. All init scripts accept at least start, stop, and status arguments. The perforce user can start p4d by calling:

```
p4d_1_init start
```

And stop it by calling:

```
p4d_1_init stop
```

Once logged into Perforce as a super user, the p4 admin stop command can also be used to stop p4d.

All init scripts can be started as the perforce user or the root user (except p4web, which must start initially as root). The application runs as the perforce user in any case. If the init scripts are configured as system services (non-systemd distributions), they can also be called by the root user using the service command, as in this example to start p4d:

```
service p4d_1_init start
```

Templates for the init scripts used by `mkdirs.sh` are stored in:

```
/p4/common/etc/init.d
```

There are also basic crontab templates for a Perforce master and replica server in:

```
/p4/common/etc/cron.d
```

These define schedules for routine checkpoint operations, replica status checks, and email reviews.

The Perforce should have a super user defined as named by the P4USER setting in mkdir.

To configure and start instance 1, follow these steps:

1. Start the Perforce server by calling

   ```
   p4d_1_init start
   ```

2. Ensure that the admin user configured above has the correct password defined in `/p4/common/config/.p4passwd.p4_1.admin`, and then run the p4login script (which calls the p4 login command using the `.p4passwd.p4_1.admin` file)
3. For new servers, run this script, which sets several recommended configurables:

   ```
   $SDP/Server/setup/configure_new_server.sh 1
   ```

For existing servers, examine this file, and manually apply the p4 configure command to set configurables on your Perforce server.

Initialize the perforce user's crontab with one of these commands:

```
crontab /p4/p4.crontab
```

and customise execution times for the commands within the crontab files to suite the specific installation.

The SDP uses wrapper scripts in the crontab: `run_if_master.sh`, `run_if_edge.sh`, `run_if_replica.sh`. We suggest you ensure these are working as desired, e.g.

```
/p4/common/bin/run_if_master.sh 1 echo yes
```

The above should output `yes` if you are on the master (commit) machine, but otherwise nothing. Any issues with the above indicate incorrect values for `$MASTER_ID`. You can debug this with:

```
bash -xv /p4/common/bin/run_if_master.sh 1 echo yes
```

If in doubt contact support.

To verify that your server installation is working properly:

1. Issue the p4 info command, after setting appropriate environment variables by running:

```
source /p4/common/bin/p4_vars 1
```

2. If the server is running, it will display details about its settings.

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

- Any init scripts used in `/etc/init.d`.
- A copy of the crontab file, obtained using `crontab -l`.
- Any other relevant configuration scripts, such as cluster configuration scripts, failover scripts, or disk failover configuration files.

5.2. Archiving configuration files

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

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

5.3. 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 typemap entries to override Perforce's default behavior.
5. 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`

6. To make your changelists default to restricted (for high security environments):

```
p4 configure set defaultChangeType=restricted
```

5.4. Other server configurables

There are various configurables that you should consider setting for your server.

Some suggestions are in the file: `$SDP/Server/setup/configure_new_server.sh`

Review the contents and either apply individual settings manually, or edit the file and apply the newly edited version. If you have any questions, please see the configurables section in Command Reference Guide appendix (get the right version for your server!). You can also contact support regarding questions.

5.5. General SDP Usage

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

5.5.1. Linux

Most scripts and tools reside in `/p4/common/bin`. The `/p4/instance/bin` directory (e.g. `/p4/1/bin`) contains scripts or links that are specific to that instance such as wrappers for the p4d executable.

Older versions of the SDP required you to always run important administrative commands using the `p4master_run` script, and specify fully qualified paths. This script loads environment information from `/p4/common/bin/p4_vars`, the central environment file of the SDP, ensuring a controlled environment. The `p4_vars` file includes instance specific environment data from `/p4/common/config/p4_instance.vars` e.g. `/p4/common/config/p4_1.vars`. The `p4master_run script` is still used when running p4 commands against the server unless you set up your environment first by sourcing p4_vars with the instance as a parameter (for bash shell: `source /p4/common/bin/p4_vars 1`). Administrative scripts, such as `daily_backup.sh`, no longer need to be called with `p4master_run` however, they just need you to pass the instance number to them as a parameter.

When invoking a Perforce command directly on the server machine, use the p4*instance wrapper that is located in /p4/_instance*/bin. This wrapper invokes the correct version of the p4 client for the instance. The use of these wrappers enables easy upgrades, because the wrapper is a link to the correct version of the p4 client. There is a similar wrapper for the p4d executable, called p4d_*instance*.

This wrapper is important to handle case sensitivity in a consistent manner, e.g. when running a Unix server in case-insensitive mode. If you just execut `p4d` directly when it should be case-insenstive, then you may cause problems, or commands will fail.

Below are some usage examples for instance 1.

Example	*Remarks*
`/p4/common/bin/p4master_run 1 /p4/1/bin/p4_1 admin stop`	Run `p4 admin stop` on instance 1
`/p4/common/bin/live_checkpoint.sh 1`	Take a checkpoint of the live database on instance 1
`/p4/common/bin/p4login 1`	Log in as the perforce user (superuser) on instance 1.

Some maintenance scripts can be run from any client workspace, if the user has administrative access to Perforce.

5.5.2. 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.

5.6. P4V Performance Settings

These are covered in: https://community.perforce.com/s/article/2878

Chapter 6. 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 hxdepots 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.

6.1. Typical Backup Procedure

The SDP's maintenance scripts, run as `cron` tasks, periodically back up the metadata. The weekly sequence is described below.

Seven nights a week, perform the following tasks:

1. Truncate the active journal.
2. Replay the journal to the offline database. (Refer to Figure 2: SDP Runtime Structure and 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 depot files.

Once every few months, perform the following tasks:

1. Stop the live server.
2. Truncate the active journal.
3. Replay the journal to the offline database. (Refer to Figure 2: SDP Runtime Structure and Volume Layout for more information on the location of the live and offline databases.)
4. Archive the live database.
5. Move the offline database to the live database directory.
6. Start the live server.
7. Create a new checkpoint from the archive of the live database.
8. Recreate the offline database from the last checkpoint.
9. Verify all depots.

This normal maintenance procedure puts the checkpoints (metadata snapshots) on the hxdepots volume, which contains the versioned files. Backing up the hxdepots 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 hxdepots 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.

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.

6.2. 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. 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. See the Distributing Perforce Guide for details on setting up replica servers.

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 or you can use a forwarding replica. The broker can do load balancing to a pool of replicas if you need more than one replica to handle your load. Use of the broker may require use of a P4AUTH server for authentication.

6.2.1. Replication Setup

To configure a replica server, first configure a machine identically to the master server (at least as regards the link structure such as /p4, /p4/common/bin and /p4/*instance*/*), then install the SDP on it to match the master server installation. Once the machine and SDP install is in place, you need to configure the master server for replication.

Perforce supports many types of replicas suited to a variety of purposes, such as:

- Real-time backup,
- Providing a disaster recovery solution,
- Load distribution to enhance performance,
- Distributed development,
- Dedicated resources for automated systems, such as build servers, and more.

We always recommend first setting up the replica as a read-only replica and ensuring that everything is working. Once that is the case you can easily modify server specs and configurables to change it to a forwarding replica, or an edge server etc.

6.2.2. Using mkrep.sh

This script automates the following:

- creation of all the configurables for a replica appropriate to its type (e.g. forwarding-replica, forwarding-standby, edge-server etc).
- standard naming conventions are used for server ids, service user names etc. This simplifies managing multiple server/replica topologies and understanding the intended use of a replica (e.g. that it is intended for HA - high availability)
- creation of service user account, password, and with appropriate permissions
- creation of server spec
- detailed instructions to follow in order to create a checkpoint and restore on the replica server

Prerequisites:

- You must have a server spec for your master server, typically defined with Services: "commit-server" ("standard" is fine if no edge servers are to be created, but it is not a problem to use commit-server even without any edge servers) - use the serverid (output of "p4 serverid") as the name.
- You should be running p4d 2018.2 or later
- You should have a configuration file which defines site tags - this is part of naming.

Server Types

These are:

- ha - High Availability
- ham - High Availability (Metadata only)
- ro - Read only replica
- rom - Read only replica (Metadata only)
- fr - Forwarding replica
- fs - Forwarding standby
- frm - (Metadata only)
- fsm - (Metadata only)
- ffr - Filtered forwarding replica
- edge - Edge server

Replicas with 'standby' are always unfiltered, and use the 'journalcopy' method of replication, which copies a byte-for-byte verbatim journal file rather than one that is merely logically equivalent.

Example

An example run is:

```
/p4/common/bin/mkrep.sh -i 1 -t fs -s bos -r replica1 -skip_ssh
```

The above will:

- Create a replica for instance 1
- Of type `fs` (forwarding standby) - with appropriate configurables
- For site `bos` (e.g. Boston)
- On host name `replica1`
- Without checking that passwordless ssh is possible to the host replica1

The tag has several purposes:

- Short Hand. Each tag represents a combination of 'Type:' and fully qualified 'Services:' values used in server specs.
- Distillation. Only the most useful Type/Services combinations have a shorthand form.
- For forwarding replicas, the name includes the critical distinction of whether any replication filtering is used; as filtering of any kind disqualifies a replica from being a potential failover target. (No such distinction is needed for edge servers, which are filtered by definition).

Mkrep.sh output

The output (which is also written to a log file in `/p4/<instance>/logs/mkrep.*`) describes a number of steps required to continue setting up the replica, e.g.

- Rotate the current live journal (to save the configuration parameters required)
- Copy across latest checkpoint and the subsequent rotated journals to the replica host machine
- Restore the copied checkpoints/journals into `/p4/<instance>/root` (and `offline_db`)
- Create a password file for service user
- Create appropriate server.id files
- Login the service user to the upstream server (usually commit server)
- Start the replica process
- Monitor that all is well with `p4 pull -lj`

More details on these steps can be found in the manual process below as well as the actualy mkrep.sh output.

6.2.3. Manual Steps

In the sample below, the replica name will be `replica1`, it is instance 1 on a particular host, the service user name is `svc_replica1`, and the master server's hostname is `svrmaster`.

The following sample commands illustrate how to setup a simple read-only replica.

First we ensure that journalPrefix is set appropriately for the master server (in this case we assume

instance 1 rather than a named instance):

```
p4 configure set master#journalPrefix=/p4/1/checkpoints/p4_1
```

Then we set values for the replica itself:

```
p4 configure set replica1#P4TARGET=svrmaster:1667
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=svc_replica1
```

Then the following also need to be setup:

- Create a service user for the replica (Add the Type: service field to the user form before saving):

  ```
  p4 user -f svc_replica1
  ```

- Set the service user's password:

  ```
  p4 passwd svc_replica1
  ```

- Add the service user svc_replica1 to a specific group ServiceUsers which has a timeout value of unlimited:

  ```
  p4 group ServiceUsers
  ```

- Make sure the ServiceUsers group has super access in protections table:

  ```
  p4 protect
  ```

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

```
/p4/common/bin/daily_checkpoint.sh 1
```

When the checkpoint finishes, rsync the checkpoint plus the versioned files over to the replica:

```
rsync -avz /p4/1/checkpoints/p4_1.ckp.###.gz perforce@replica:/p4/1/checkpoints/.
```

```
rsync -avz /p4/1/depots/ perforce@replica:/p4/1/depots/
```

(Assuming perforce is the OS user name and replica is the name of the replica server in the commands above, and that # is the checkpoint number created by the daily backup.)

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

```
/p4/1/bin/p4d_1 -r /p4/1/root -jr -z /p4/1/checkpoints/p4_1.ckp.###.gz
```

Login as the service user (specifying appropriate password when prompted), and making sure that the login ticket generated is stored in the same place as specified in the P4TICKETS configurable value set above for the replica (the following uses bash syntax):

```
source /p4/common/bin/p4_vars 1
/p4/1/bin/p4_1 -p svrmaster:1667 -u svc_replica1 login
```

Start the replica instance (either using _init script or systemctl if on system):

```
/p4/1/bin/p4d_1_init start
```

Now, you can log into the replica server itself and run p4 pull -lj to check to see if replication is working. If you see any numbers with a negative sign in front of them, replication is not working. The most likely cause of this is that the service user is not logged in. Rerun the steps above to login the service user and check again. If replication still is not working, check /p4/1/logs/log on the replica, and also look for authentication failures in the log for the master instance on svrmaster.

The final steps for setting up the replica server are to set up the crontab for the replica server.

To configure the ssh trust:

On both the master and replica servers, go to the perforce user's home directory and run:

```
ssh-keygen -t rsa
```

Just use the defaults for the questions it asks.

Now from the master, run:

```
rsync -avz ~/.ssh/id_rsa.pub perforce@replica:~/.ssh/authorized_keys
```

and from the replica, run:

```
rsync -avz ~/.ssh/id_rsa.pub perforce@master:~/.ssh/authorized_keys
```

The crontab (/p4/p4.crontab) contains several lines which are prefixed by /p4/common/bin/run_if_replica.sh or run_if_edge.sh or run_if_master.sh

These can be tested to make sure all is valid with:

```
/p4/common/bin/run_if_replica.sh 1 echo yes
```

If "yes" is output then SDP thinks the current hostname with instance 1 is a replica server. Similarly for edge/master.

The log files will be in /p4/1/logs, so you can check for any errors from each script.

6.3. 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 hxdepots volume using your normal backup utilities.

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

6.3.1. Recovering a master server from a checkpoint and journal(s)

The checkpoint files are stored in the /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 /p4/*instance*/checkpoints and the journal file from /p4/instance/logs.

To recover the server database manually, perform the following steps from the root directory of the server (/p4/instance/root).

Assuming instance 1:

1. Stop the Perforce Server by issuing the following command:

```
/p4/1/bin/p4_1 admin stop
```

2. Delete the old database files in the `/p4/1/root/save` directory
3. Move the live database files (db.*) to the save directory.
4. Use the following command to restore from the most recent checkpoint.

```
/p4/1/bin/p4d_1 -r /p4/1/root -jr -z /p4/1/checkpoints/p4_1.ckp.####.gz
```

5. To replay the transactions that occurred after the checkpoint was created, issue the following command:

```
/p4/1/bin/p4d_1 -r /p4/1/root -jr /p4/1/logs/journal
```

6. Restart your Perforce server.

If the Perforce service starts without errors, delete the old database files from `/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 backup and recovery, refer to the Perforce System Administrator's Guide.

6.3.2. Recovering a replica from a checkpoint

This is very similar to creating a replica in the first place as described above.

If you have been running the replica crontab commands as suggested, then you will have the latest checkpoints from the master already copied across to the replica.

See the steps in the script weekly_sync_replica.sh for details (note that it deletes the state and rdb.lbr files from the replica root directory so that the replica starts replicating from the start of a journal).

Remember to ensure you have logged the service user in to the master server (and that the ticket is stored in the correct location as described when setting up the replica).

6.3.3. 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 hxdepots 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 hxdepots volume from your backup tape.
2. Create the `/p4` convenience directory on the OS volume.
3. Create the directories `/metadata/p4/instance/root/save` and `/metadata/p4/instance/offline_db`.
4. Change ownership of these directories to the OS account that runs the Perforce processes.
5. Switch to the Perforce OS account, and create a link in the `/p4` directory to `/depotadata/p4/instance`.
6. Create a link in the `/p4` directory to `/hxdepots/p4/common`.
7. As a super-user, reinstall and enable the `init.d` scripts
8. Find the last available checkpoint, under `/p4/instance/checkpoints`
9. Recover the latest checkpoint by running:

```
/p4/_instance_/bin/p4d__instance_ -r /p4/_instance_/root -jr -z _last_ckp_file_
```

10. Recover the checkpoint to the offline_db directory (assuming instance 1):

```
/p4/1/bin/p4d_1 -r /p4/1/offline_db -jr -z <last_ckp_file>
```

11. Reinstall the Perforce server license to the server root directory.
12. Start the perforce service by running 1/p4/1/bin/p4d_1_init start `
13. Verify that the server instance is running.
14. Reinstall the server crontab or scheduled tasks.
15. Perform any other initial server machine configuration.
16. Verify the database and versioned files by running the `p4verify.sh` 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.

6.3.4. Failover to a replicated standby machine

See DR-Failover-Steps-Linux.docx

Chapter 7. Server Maintenance

This section describes typical maintenance tasks and best practices for administering server machines.

7.1. 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 for your OS from ftp.perforce.com and place them in `/p4/common/bin`
- Run:

  ```
  /p4/common/bin/upgrade.sh <instance>
  ```

  e.g.

  ```
  /p4/common/bin/upgrade.sh 1
  ```

- If you are running replicas, upgrade the replicas first, and then the master (outside → in)

7.2. Database Modifications

Occasionally modifications are made to the Perforce database from one release to another. 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:

```
/p4/common/bin/live_checkpoint.sh 1
```

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 should 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.

Chapter 8. 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 Knowledge Base.

8.1. 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. This normally only needs to be done very few months.

To minimize the size of back up files and maximize server performance, minimize the size of the db.have and db.label files.

8.2. Proactive Performance Maintenance

This section describes some things that can be done to proactively to enhance scalability and maintain performance.

8.2.1. 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.

8.2.2. 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. To launch P4P on Unix machines, copy the `/p4/common/etc/init.d/p4p_1_init script` to `/p4/1/bin/p4p_1_init`. Then review and customize the script to specify your server volume names and directories.

P4P does not require special hardware but it can be quite CPU intensive if it is working with binary files, which are CPU-intensive to attempt to compress. 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.

Chapter 9. Tools and Scripts

This section describes the various scripts and files provided as part of the SDP package. To run main scripts, the machine must have Python 2.7, and a few scripts require Perl 5. The Maintenance scripts can be run from the server machine or from client machines.

The following various scripts.

9.1. Core Scripts

The core SDP scripts are those related to checkpoints and other scheduled operations, and all run from /p4/common/bin.

9.1.1. p4_vars

Defines the environment variables required by the Perforce server. This script uses a specified instance number as a basis for setting environment variables. It will look for and open the respective p4_<instance>.vars file (see next section).

This script also sets server logging options and configurables.

Location: /p4/common/bin

9.1.2. p4_<instance>.vars

Defines the environment variables for a specific instance, including P4PORT etc.

Location: /p4/common/config

9.1.3. p4master_run

This is the wrapper script to other SDP scripts. This ensures that the shell environment is loaded from p4_vars. It provides a '-c' flag for silent operation, used in many crontab so that email is sent from the scripts themselves.

Location: /p4/common/bin

9.1.4. recreate_offline_db

Recovers the offline_db database from the latest checkpoint and replays any journals since then. If you have a problem with the offline database then it is worth running this script first before running live_checkpoint, as the latter will stop the server while it is running which can take hours.

Run this script if an error occurs while replaying a journal during weekly or daily checkpoint process.

Location: /p4/common/bin

9.1.5. live_checkpoint

Stops the server, creates a checkpoint from the live database files, recovers the offline_db database 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. Be aware it locks live database for the duration of the checkpoint which can take hours.

Location: /p4/common/bin

9.1.6. daily_checkpoint

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: /p4/common/bin

9.1.7. p4verify.sh

Verifies the integrity of the depot files. This script is run by crontab on a regular basis.

Location: /p4/common/bin

9.1.8. 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 /p4/common/config, called p4_<instance>.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.

This is not required if you have installed Swarm which also performs notification functions and is easier for users to configure.

Location: /p4/common/bin

9.1.9. p4login

Executes a p4 login command, using the administration password configured in `mkdirs.cfg` and subsequently stored in a text file: `/p4/common/config/.p4passwd .p4_<instance>.admin`

Location: /p4/common/bin

9.1.10. p4d_<instance>_init

Starts the Perforce server. Do not use if you have configured systemctl for systemd Linux distributions such as CentOS 7.x

This script sources `/p4/common/bin/p4_vars`, then `/p4/common/bin/p4d_base`.

In clustered environments, put this script in the `/p4/<instance>/bin` directory and configure your cluster software to launch it from this location.

Location: `/p4/<instance>/bin` with a symlink to it from `/etc/init.d` (or a copy in `/etc/init.d` in a clustered environments). Templates for init scripts for other Perforce server products exist in `/p4/common/etc/init.d`

9.2. More Server Scripts

These scripts are helpful components of the SDP that run on the server, but are not included in the default crontab schedules.

9.2.1. upgrade.sh

Runs a typical upgrade process, once new p4 and p4d binaries are available in `/p4/common/bin`.

This script will:

- Rotate the journal (for clean recovery point)
- Apply all necessary journals to offline_db
- Stop the server
- Create an appropriately versioned link for new p4/p4d/p4broker etc
- Link those into `/p4/1/bin` (per instance)
- Run `p4d -xu` on live and offline_db to perform database upgrades
- Restart server

Location: /p4/common/bin

9.2.2. p4.crontab

Contains crontab entries to run the server maintenance scripts.

Location: /p4/sdp/Server/Unix/p4/common/etc/cron.d

9.3. 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.
backup_functions.sh	`/p4/common/bin`	Unix/Linux only. Utilities for maintenance scripts.
p4admin_verify_client.bat	`/p4/common/bin`	Unix/Linux only. Used by p4verify.sh.
p4d_base	`/p4/common/bin`	Unix/Linux only. Template for Unix/Linux init.d scripts.
template.(pl	sh)	`/p4/common/bin`
Sample script templates for Bash and Perl scripts.	mirror_ldap_groups.pl	`/p4/common/bin`
Script to mirror selected groups from an LDAP server (e.g. Active Directory).	Perl Modules (*.pm files)	`/p4/common/lib`

9.3.1. SDP Package Contents

The directory structure of the SDP is shown below in Figure 1 - SDP Package Directory Structure. This includes all SDP files, including documentation and maintenance scripts. A subset of these files are deployed to server machines during the installation process.

```
sdp
    doc
    Server (Core SDP Files)
        Unix
            setup (unix specific setup)
            p4
                common
                    bin (Backup scripts, etc)
                        triggers (Example triggers)
                    config
                    etc
                        cron.d
                        init.d
                        lib
                        test
    setup (cross platform setup - typemap, configure, etc)
    test (automated test scripts)
```

Figure 1 - SDP Package Directory Structure

9.3.2. Volume Layout and Server Planning

Figure 2: SDP Runtime Structure and Volume Layout, 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.

Memory and CPU

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

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 are typically more desirable than a greater number of cores and provide better performance since quick bursts of computational speed are more important to Perforce's performance than the number of processors. Have a minimum of two processors so that the offline checkpoint and back up processes do not interfere with your Perforce server. There are log analysis options to diagnose underperforming servers and improve things (contact support/consulting for details).

9.3.3. Directory Structure Configuration Script for Linux/Unix

This script describes the steps performed by the mkdirs.sh script on Linux/Unix platforms. Please review this appendix carefully before running these steps manually. Assuming the three-volume configuration described in the Volume Layout and Hardware section are used, the following directories are created. The following examples are illustrated with "1" as the server instance number.

Directory	*Remarks*
`/p4`	Must be under root (`/`) on the OS volume
`/hxdepots/p4/1/bin`	Files in here are generated by the mkdirs.sh script.
`/hxdepots/p4/1/depots`	
`/hxdepots/p4/1/tmp`	
`/hxdepots/p4/common/config`	Contains p4_<instance>.vars file, e.g. `p4_1.vars`
`/hxdepots/p4/common/bin`	Files from `$SDP/Server/Unix/p4/common/bin`.
`/hxdepots/p4/common/etc`	Contains `init.d` and `cron.d`.
`/hxlogs/p4/1/logs/old`	

Directory	*Remarks*
`/hxmetadata2/p4/1/db2`	Contains offline copy of main server databases (linked by `/p4/1/offline_db`.
`/hxmetadata1/p4/1/db1/save`	Used only during running of `recreate_db_checkpoint.sh` for extra redundancy.

Next, `mkdirs.sh` creates the following symlinks in the `/hxdepots/p4/1` directory:

Link source	***Link target***	***Command***
`/hxmetadata1/p4/1/db1`	`/p4/1/root`	`ln -s /hxmetadata1/p4/1/root`
`/hxmetadata2/p4/1/db2`	`/p4/1/offline_db`	`ln -s /hxmetadata1/p4/1/offline_db`
`/hxlogs/p4/1/logs`	`/p4/1/logs`	`ln -s /hxlogs/p4/1/logs`

Then these symlinks are created in the /p4 directory:

Link source	***Link target***	***Command***
`/hxdepots/p4/1`	`/p4/1`	`ln -s /hxdepots/p4/1 /p4/1`
`/hxdepots/p4/common`	`/p4/common`	`ln -s /hxdepots/p4/common /p4/common`

Next, mkdirs.sh renames the Perforce binaries to include version and build number, and then creates appropriate symlinks.

The structure is shown in this example, illustrating values for two instances, with instance #1 using Perforce 2018.1 and instance #2 using 2018.2.

In `/p4/common/bin`:

```
p4_2018.1.685046
p4_2018.1_bin -> p4_2018.1.685046
p4d_2018.1.685046
p4d_2018.1_bin -> p4d_2018.1.685046
p4d_2018.2.700949
p4_2018.2_bin -> p4_2018.2.700949
p4d_2018.2_bin -> p4d_2018.2.700949
p4_1_bin -> p4_2018.1_bin
p4d_1_bin -> p4d_2018.1_bin
p4_2_bin -> p4_2018.2_bin
p4d_2_bin -> p4d_2018.2_bin
```

In /p4/1/bin:

```
p4_1 -> /p4/common/bin/p4_1_bin
p4d_1 -> /p4/common/bin/p4d_1_bin
```

In /p4/2/bin:

```
p4_2 -> /p4/common/bin/p4_2
p4d_2 -> /p4/common/bin/p4d_2
```

9.3.4. Frequently Asked Questions/Troubleshooting

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. Because the scripts will replay all outstanding journals, this error is much less likely to occur. This error can be fixed by running the live_checkpoint.sh script. Alternatively, if you know that the checkpoints created from previous runs of daily_checkpoint.sh are correct, then restore the offline_db from the last known good checkpoint.

Unexpected end of file in replica daily sync

Check the start time and duration of the daily_checkpoint.sh cron job on the master. If this overlaps with the start time of the sync_replica.sh cron job on a replica, a truncated checkpoint may be rsync'd to the replica and replaying this will result in an error.

Adjust the replica's cronjob to start later to resolve this.

Default cron job times, as installed by the SDP are initial estimates, and should be adjusted to suit your production environment.