

pgmoneta

Developer Guide

Contents

1	Introduction	5
1.1	Features	5
1.2	Platforms	6
2	Installation	7
2.1	Fedora	7
2.2	RHEL 9 / RockyLinux 9	7
2.3	Compiling the source	7
2.3.1	RHEL / RockyLinux	8
2.3.2	FreeBSD	9
2.3.3	Build	9
2.4	Compiling the documentation	10
2.4.1	Build	10
2.5	Extension installation	10
2.5.1	Install pgmoneta_ext	11
2.5.2	Verify success	11
2.5.3	Granting SUPERUSER Privileges	11
3	C programming	13
3.1	Debugging	13
4	Git guide	13
4.1	Basic steps	13
4.1.1	Start by forking the repository	13
4.2	Clone your repository locally	13
4.2.1	Add upstream	14
4.2.2	Do a work branch	14
4.2.3	Make the changes	14
4.2.4	Multiple commits	14
4.2.5	Rebase	14
4.2.6	Force push	14
4.2.7	Format source code	15
4.2.8	Repeat	15
4.2.9	Undo	15
5	Architecture	16
5.1	Overview	16

5.2	Shared memory	16
5.3	Network and messages	16
5.4	Memory	17
5.5	Management	17
5.5.1	Write	17
5.5.2	Read	18
5.5.3	Remote management	18
5.6	libev usage	19
5.7	Signals	19
5.8	Reload	19
5.9	Prometheus	19
5.10	Logging	20
5.11	Protocol	20
6	Encryption	21
6.1	Overview	21
6.2	Encryption Configuration	21
6.3	Encryption / Decryption CLI Commands	21
6.3.1	decrypt	21
6.3.2	encrypt	22
6.4	Benchmark	22
7	RPM	26
7.1	Requirements	26
7.2	Setup RPM development	26
7.3	Create source package	26
7.4	Create RPM package	26
8	Test	27
8.1	Container Environment	27
8.1.1	Docker	27
8.1.2	Podman	27
8.2	Test suite	28
9	WAL Reader	29
9.1	Overview	29
9.2	pgmoneta-walinfo	29
9.3	High-Level API Overview	30
9.3.1	Struct walfile	30

9.3.2	Function Overview	31
9.4	Internal API Overview	32
9.4.1	parse_wal_file	32
9.4.2	Usage Example	33
9.4.3	WAL File Structure	33
9.5	Resource Managers	34
9.5.1	Resource Manager Definitions	34
9.5.2	Resource Manager Functions	34
9.5.3	Supporting Various WAL Structures in PostgreSQL Versions 13 to 17	34
9.6	WAL Change List	36
9.6.1	xl_clog_truncate	36
9.6.2	xl_commit_ts_truncate	37
9.6.3	xl_heap_prune	37
9.6.4	xlhp_freeze_plan	38
9.6.5	spgxlogState	38
9.6.6	xl_end_of_recovery	39
9.6.7	gingxlogSplit	39
9.6.8	gistxlogDelete	40
9.6.9	gistxlogPageReuse	40
9.6.10	xl_hash_vacuum_one_page	41
9.6.11	xl_heap_prune	42
9.6.12	xl_heap_freeze_plan	42
9.6.13	xl_heap_freeze_page	43
9.6.14	xl_btree_reuse_page	43
9.6.15	xl_btree_delete	44
9.6.16	spgxlogVacuumRedirect	44
9.6.17	xl_xact_prepare	45
9.6.18	xl_xact_parsed_commit	46
9.6.19	xl_xact_parsed_abort	48
9.6.20	xlogrecord.h flags	49
9.6.21	xl_heap_prune	49
9.6.22	xl_heap_vacuum	50
9.6.23	xl_btree_metadata	50
9.6.24	xl_btree_reuse_page	51
9.6.25	xl_btree_delete	51
9.6.26	xl_btree_unlink_page	52
9.7	Additional Information	53

10 Troubleshooting	54
10.1 Could not get version for server	54
11 Acknowledgement	55
11.1 Authors	55
11.2 Committers	55
11.3 Contributing	55
12 License	57
12.1 libart	57

1 Introduction

pgmoneta is a backup / restore solution for PostgreSQL.

Ideally, you would not need to do backups and disaster recovery, but that isn't how the real World works.

Possible scenarios that could happen

- Data corruption
- System failure
- Human error
- Natural disaster

and then it is up to the database administrator to get the database system back on-line, and to the correct recovery point.

Two key factors are

- Recovery Point Objective (RPO): Maximum targeted period in which data might be lost from an IT service due to a major incident
- Recovery Time Objective (RTO): The targeted duration of time and a service level within which a business process must be restored after a disaster (or disruption) in order to avoid unacceptable consequences associated with a break in business continuity

You would like to have both of these as close to zero as possible, since RPO of 0 means that you won't lose data, and RTO of 0 means that your system recovers at once. However, that is easier said than done.

pgmoneta is focused on having features that will allow database systems to get as close to these goals as possible such that high availability of 99.99% or more can be implemented, and monitored through standard tools.

pgmoneta is named after the Roman Goddess of Memory.

1.1 Features

- Full backup
- Restore
- Compression (gzip, zstd, lz4, bzip2)
- AES encryption support
- Symlink support
- WAL shipping support

- Hot standby
- Prometheus support
- Remote management
- Offline mode
- Transport Layer Security (TLS) v1.2+ support
- Daemon mode
- User vault

1.2 Platforms

The supported platforms are

- Fedora 38+
- RHEL 9
- RockyLinux 9
- FreeBSD
- OpenBSD

2 Installation

2.1 Fedora

You need to add the PostgreSQL YUM repository, for example for Fedora 40

```
dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/F-40-x86_64/pgdg-fedora-repo-latest.noarch.rpm
```

and do the install via

```
dnf install -y pgmoneta
```

Additional information

- PostgreSQL YUM
- Linux downloads

2.2 RHEL 9 / RockyLinux 9

x86_64

```
dnf install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-9-x86_64/pgdg-redhat-repo-latest.noarch.rpm
```

aarch64

```
dnf install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-9-aarch64/pgdg-redhat-repo-latest.noarch.rpm
```

and do the install via

```
dnf install -y pgmoneta
```

2.3 Compiling the source

We recommend using Fedora to test and run **pgmoneta**, but other Linux systems, FreeBSD and MacOS are also supported.

pgmoneta requires

- clang
- cmake
- make
- libev
- OpenSSL
- zlib
- zstd
- lz4
- bzip2
- systemd
- rst2man
- libssh
- libcurl
- libarchive

```
dnf install git gcc clang clang-analyzer cmake make libev libev-devel \
            openssl openssl-devel \
            systemd systemd-devel zlib zlib-devel \
            libzstd libzstd-devel \
            lz4 lz4-devel libssh libssh-devel \
            libcurl libcurl-devel \
            python3-docutils libatomic \
            bzip2 bzip2-devel \
            libarchive libarchive-devel
```

Alternative gcc can be used.

2.3.1 RHEL / RockyLinux

On RHEL / Rocky, before you install the required packages some additional repositories need to be enabled or installed first.

First you need to install the subscription-manager

```
dnf install subscription-manager
```

It is ok to disregard the registration and subscription warning.

Otherwise, if you have a Red Hat corporate account (you need to specify the company/organization name in your account), you can register using

```
subscription-manager register --username <your-account-email-or-login> --
    password <your-password> --auto-attach
```

Then install the EPEL repository,

```
dnf install epel-release
```

Then to enable powertools

```
dnf config-manager --set-enabled codeready-builder-for-rhel-9-rhui-rpms
dnf config-manager --set-enabled crb
dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.
noarch.rpm
```

Then use the `dnf` command for **pgmoneta** to install the required packages.

2.3.2 FreeBSD

On FreeBSD, `pkg` is used instead of `dnf` or `yum`.

Use `pkg install <package name>` to install the following packages

```
git gcc cmake libev openssl libssh zlib-ng zstd liblz4 bzip2 curl \
py39-docutils libarchive
```

2.3.3 Build

2.3.3.1 Release build The following commands will install **pgmoneta** in the `/usr/local` hierarchy.

```
git clone https://github.com/pgmoneta/pgmoneta.git
cd pgmoneta
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/usr/local ..
make
sudo make install
```

See RPM for how to build a RPM of **pgmoneta**.

2.3.3.2 Debug build The following commands will create a `DEBUG` version of **pgmoneta**.

```
git clone https://github.com/pgmoneta/pgmoneta.git
cd pgmoneta
mkdir build
cd build
cmake -DCMAKE_C_COMPILER=clang -DCMAKE_BUILD_TYPE=Debug ..
make
```

2.4 Compiling the documentation

pgmoneta's documentation requires

- pandoc
- texlive

```
dnf install pandoc texlive-scheme-basic \
    'tex(footnote.sty)' 'tex(footnotebackref.sty)' \
    'tex(pagecolor.sty)' 'tex(hardwrap.sty)' \
    'tex(mdframed.sty)' 'tex(sourcesanspro.sty)' \
    'tex(lylenc.def)' 'tex(sourcecodepro.sty)' \
    'tex(titling.sty)' 'tex(csquotes.sty)' \
    'tex(zref-abspace.sty)' 'tex(needspace.sty)'
```

You will need the [Eisvogel](#) template as well which you can install through

```
wget https://github.com/Wandmalfarbe/pandoc-latex-template/releases/
download/2.4.2/Eisvogel-2.4.2.tar.gz
tar -xzf Eisvogel-2.4.2.tar.gz
mkdir -p $HOME/.local/share/pandoc/templates
mv eisvogel.latex $HOME/.local/share/pandoc/templates
```

where `$HOME` is your home directory.

2.4.0.1 Generate API guide This process is optional. If you choose not to generate the API HTML files, you can opt out of downloading these dependencies, and the process will automatically skip the generation.

Download dependencies

```
dnf install graphviz doxygen
```

2.4.1 Build

These packages will be detected during `cmake` and built as part of the main build.

2.5 Extension installation

When you configure the `extra` parameter in the server section of `pgmoneta.conf`, it requires the server side to have the `pgmoneta_ext` extension installed to make it work.

The following instructions can help you easily install `pgmoneta_ext`. If you encounter any problems, please refer to the more detailed instructions in the DEVELOPERS documentation.

2.5.1 Install pgmoneta_ext

After you have successfully installed `pgmoneta`, the following commands will help you install `pgmoneta_ext`:

```
dnf install -y pgmoneta_ext
```

You need to add the `pgmoneta_ext` library for PostgreSQL in `postgresql.conf` as well:

```
shared_preload_libraries = 'pgmoneta_ext'
```

And remember to restart PostgreSQL to make it work.

2.5.2 Verify success

You can use the `postgres` role to test.

1. Log into PostgreSQL

```
psql
```

2. Create a new test database

```
CREATE DATABASE testdb;
```

3. Enter the database

```
\c testdb
```

4. Follow the SQL commands below to check the function

```
DROP EXTENSION IF EXISTS pgmoneta_ext;  
CREATE EXTENSION pgmoneta_ext;  
SELECT pgmoneta_ext_version();
```

You should see

```
pgmoneta_ext_version  
-----  
0.1.0  
(1 row)
```

2.5.3 Granting SUPERUSER Privileges

Some functions in `pgmoneta_ext` require `SUPERUSER` privileges. To enable these, grant the `repl` role superuser privileges using the command below. **Please proceed with caution:** granting superuser

privileges bypasses all permission checks, allowing unrestricted access to the database, which can pose security risks. We are committed to enhancing privilege security in future updates.

```
ALTER ROLE repl WITH SUPERUSER;
```

To revoke superuser privileges from the `repl` role, use the following command:

```
ALTER ROLE repl WITH NOSUPERUSER;
```

3 C programming

pgmoneta is developed using the C programming language so it is a good idea to have some knowledge about the language before you begin to make changes.

There are books like,

- C in a Nutshell
- 21st Century C

that can help you

3.1 Debugging

In order to debug problems in your code you can use `gdb`, or add extra logging using the `pgmoneta_log_XYZ()` API

4 Git guide

Here are some links that will help you

- How to Squash Commits in Git
- ProGit book

4.1 Basic steps

4.1.1 Start by forking the repository

This is done by the “Fork” button on GitHub.

4.2 Clone your repository locally

This is done by

```
git clone git@github.com:<username>/pgmoneta.git
```

4.2.1 Add upstream

Do

```
cd pgmoneta
git remote add upstream https://github.com/pgmoneta/pgmoneta.git
```

4.2.2 Do a work branch

```
git checkout -b mywork main
```

4.2.3 Make the changes

Remember to verify the compile and execution of the code.

Use

```
[#xyz] Description
```

as the commit message where [#xyz] is the issue number for the work, and *Description* is a short description of the issue in the first line

4.2.4 Multiple commits

If you have multiple commits on your branch then squash them

```
git rebase -i HEAD~2
```

for example. It is *p* for the first one, then *s* for the rest

4.2.5 Rebase

Always rebase

```
git fetch upstream
git rebase -i upstream/main
```

4.2.6 Force push

When you are done with your changes force push your branch

```
git push -f origin mywork
```

and then create a pull request for it

4.2.7 Format source code

Use

```
./uncrustify.sh
```

to format the source code

4.2.8 Repeat

Based on feedback keep making changes, squashing, rebasing and force pushing

4.2.9 Undo

Normally you can reset to an earlier commit using `git reset <commit hash> --hard`.

But if you accidentally squashed two or more commits, and you want to undo that, you need to know where to reset to, and the commit seems to have lost after you rebased.

But they are not actually lost - using `git reflog`, you can find every commit the HEAD pointer has ever pointed to. Find the commit you want to reset to, and do `git reset --hard`.

5 Architecture

5.1 Overview

pgmoneta use a process model (`fork()`), where each process handles one Write-Ahead Log (WAL) receiver to PostgreSQL.

The main process is defined in `main.c`.

Backup is handled in `backup.h` (`backup.c`).

Restore is handled in `restore.h` (`restore.c`) with linking handled in `link.h` (`link.c`).

Archive is handled in `achv.h` (`archive.c`) backed by restore.

Write-Ahead Log is handled in `wal.h` (`wal.c`).

Backup information is handled in `info.h` (`info.c`).

Retention is handled in `retention.h` (`retention.c`).

Compression is handled in `gzip_compression.h` (`gzip_compression.c`), `lz4_compression.h` (`lz4_compression.c`), `zstandard_compression.h` (`zstandard_compression.c`), and `bzip2_compression.h` (`bzip2_compression.c`).

Encryption is handled in `aes.h` (`aes.c`).

5.2 Shared memory

A memory segment (`shmem.h`) is shared among all processes which contains the **pgmoneta** state containing the configuration and the list of servers.

The configuration of **pgmoneta** (`struct configuration`) and the configuration of the servers (`struct server`) is initialized in this shared memory segment. These structs are all defined in `pgmoneta.h`.

The shared memory segment is created using the `mmap()` call.

5.3 Network and messages

All communication is abstracted using the `struct message` data type defined in `messge.h`.

Reading and writing messages are handled in the `message.h` (`message.c`) files.

Network operations are defined in `network.h` (`network.c`).

5.4 Memory

Each process uses a fixed memory block for its network communication, which is allocated upon startup of the process.

That way we don't have to allocate memory for each network message, and more importantly free it after end of use.

The memory interface is defined in `memory.h` (`memory.c`).

5.5 Management

pgmoneta has a management interface which defines the administrator abilities that can be performed when it is running. This include for example taking a backup. The `pgmoneta-cli` program is used for these operations (`cli.c`).

The management interface is defined in `management.h`. The management interface uses its own protocol which uses JSON as its foundation.

5.5.1 Write

The client sends a single JSON string to the server,

Field	Type	Description
<code>compression</code>	uint8	The compression type
<code>encryption</code>	uint8	The encryption type
<code>length</code>	uint32	The length of the JSON document
<code>json</code>	String	The JSON document

The server sends a single JSON string to the client,

Field	Type	Description
<code>compression</code>	uint8	The compression type
<code>encryption</code>	uint8	The encryption type
<code>length</code>	uint32	The length of the JSON document

Field	Type	Description
<code>json</code>	String	The JSON document

5.5.2 Read

The server sends a single JSON string to the client,

Field	Type	Description
<code>compression</code>	uint8	The compression type
<code>encryption</code>	uint8	The encryption type
<code>length</code>	uint32	The length of the JSON document
<code>json</code>	String	The JSON document

The client sends to the server a single JSON documents,

Field	Type	Description
<code>compression</code>	uint8	The compression type
<code>encryption</code>	uint8	The encryption type
<code>length</code>	uint32	The length of the JSON document
<code>json</code>	String	The JSON document

5.5.3 Remote management

The remote management functionality uses the same protocol as the standard management method.

However, before the management packet is sent the client has to authenticate using SCRAM-SHA-256 using the same message format that PostgreSQL uses, e.g. `StartupMessage`, `AuthenticationSASL`, `AuthenticationSASLContinue`, `AuthenticationSASLFinal` and `AuthenticationOk`. The `SSLRequest` message is supported.

The remote management interface is defined in `remote.h` (`remote.c`).

5.6 libev usage

libev is used to handle network interactions, which is “activated” upon an `EV_READ` event.

Each process has its own event loop, such that the process only gets notified when data related only to that process is ready. The main loop handles the system wide “services” such as idle timeout checks and so on.

5.7 Signals

The main process of **pgmoneta** supports the following signals `SIGTERM`, `SIGINT` and `SIGALRM` as a mechanism for shutting down. The `SIGABRT` is used to request a core dump (`abort()`).

The `SIGHUP` signal will trigger a reload of the configuration.

It should not be needed to use `SIGKILL` for **pgmoneta**. Please, consider using `SIGABRT` instead, and share the core dump and debug logs with the **pgmoneta** community.

5.8 Reload

The `SIGHUP` signal will trigger a reload of the configuration.

However, some configuration settings requires a full restart of **pgmoneta** in order to take effect. These are

- `hugepage`
- `libev`
- `log_path`
- `log_type`
- `unix_socket_dir`
- `pidfile`

The configuration can also be reloaded using `pgmoneta-cli -c pgmoneta.conf conf reload`. The command is only supported over the local interface, and hence doesn’t work remotely.

5.9 Prometheus

pgmoneta has support for Prometheus when the `metrics` port is specified.

The module serves two endpoints

- / - Overview of the functionality ([text/html](#))
- /[metrics](#) - The metrics ([text/plain](#))

All other URLs will result in a 403 response.

The metrics endpoint supports **Transfer-Encoding: chunked** to account for a large amount of data.

The implementation is done in `prometheus.h` and `prometheus.c`.

5.10 Logging

Simple logging implementation based on a `atomic_schar` lock.

The implementation is done in `logging.h` and `logging.c`.

5.11 Protocol

The protocol interactions can be debugged using Wireshark or `pgprtdbg`.

6 Encryption

6.1 Overview

AES Cipher block chaining (CBC) mode and AES Counter (CTR) mode are supported in **pgmoneta**. The default setup is no encryption.

CBC is the most commonly used and considered save mode. Its main drawbacks are that encryption is sequential (decryption can be parallelized).

Along with CBC, CTR mode is one of two block cipher modes recommended by Niels Ferguson and Bruce Schneier. Both encryption and decryption are parallelizable.

Longer the key length, safer the encryption. However, with 20% (192 bit) and 40% (256 bit) extra workload compare to 128 bit.

6.2 Encryption Configuration

`none`: No encryption (default value)

`aes` | `aes-256` | `aes-256-cbc`: AES CBC (Cipher Block Chaining) mode with 256 bit key length

`aes-192` | `aes-192-cbc`: AES CBC mode with 192 bit key length

`aes-128` | `aes-128-cbc`: AES CBC mode with 128 bit key length

`aes-256-ctr`: AES CTR (Counter) mode with 256 bit key length

`aes-192-ctr`: AES CTR mode with 192 bit key length

`aes-128-ctr`: AES CTR mode with 128 bit key length

6.3 Encryption / Decryption CLI Commands

6.3.1 decrypt

Decrypt the file in place, remove encrypted file after successful decryption.

Command

```
pgmoneta-cli decrypt <file>
```

6.3.2 encrypt

Encrypt the file in place, remove unencrypted file after successful encryption.

Command

```
pgmoneta-cli encrypt <file>
```

6.4 Benchmark

Check if your CPU have AES-NI

```
cat /proc/cpuinfo | grep aes
```

Query number of cores on your CPU

```
lscpu | grep '^CPU(s):'
```

By default openssl using AES-NI if the CPU have it.

```
openssl speed -elapsed -evp aes-128-cbc
```

Speed test with explicit disabled AES-NI feature

```
OPENSSL_ia32cap=~0x2000000200000000 openssl speed -elapsed -evp aes-128-cbc
```

Test decrypt

```
openssl speed -elapsed -decrypt -evp aes-128-cbc
```

Speed test with 8 cores

```
openssl speed -multi 8 -elapsed -evp aes-128-cbc
```

```
Architecture:           x86_64
CPU op-mode(s):         32-bit, 64-bit
Address sizes:          39 bits physical, 48 bits virtual
Byte Order:             Little Endian
CPU(s):                 12
On-line CPU(s) list:    0-11
Vendor ID:              GenuineIntel
Model name:             Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
CPU family:             6
Model:                  158
Thread(s) per core:     2
Core(s) per socket:     6
Socket(s):              1
```

```

Stepping:          10
BogoMIPS:          5183.98
Flags:             fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
                  pge mca cmov pat pse36 clflush mmx fxsr sse sse2 s
                  s ht syscall nx pdpe1gb rdtscp lm constant_tsc
                  rep_good nopl xtopology cpuid pni pclmulqdq
                  vmx ssse
                  3 fma cx16 pcid sse4_1 sse4_2 movbe popcnt aes
                  xsave avx f16c rdrand hypervisor lahf_lm abm 3
                  dnowpr
                  efetch invpcid_single pti ssbd ibrs ibpb stibp
                  tpr_shadow vnmi ept vpid ept_ad fsgsbase bmi1
                  avx2 s
                  mep bmi2 erms invpcid rdseed adx smap clflushopt
                  xsaveopt xsavec xgetbv1 xsaves flush_lld
                  arch_capa
                  bilities
Virtualization features:
  Virtualization:   VT-x
  Hypervisor vendor: Microsoft
  Virtualization type: full
Caches (sum of all):
  L1d:              192 KiB (6 instances)
  L1i:              192 KiB (6 instances)
  L2:               1.5 MiB (6 instances)
  L3:               12 MiB (1 instance)
Vulnerabilities:
  Itlb multihit:    KVM: Mitigation: VMX disabled
  L1tf:             Mitigation; PTE Inversion; VMX conditional cache
                  flushes, SMT vulnerable
  Mds:              Vulnerable: Clear CPU buffers attempted, no
                  microcode; SMT Host state unknown
  Meltdown:         Mitigation; PTI
  Spec store bypass: Mitigation; Speculative Store Bypass disabled via
                  prctl and seccomp
  Spectre v1:       Mitigation; usercopy/swapgs barriers and __user
                  pointer sanitization
  Spectre v2:       Mitigation; Full generic retpoline, IBPB
                  conditional, IBRS_FW, STIBP conditional, RSB filling
  Srbds:            Unknown: Dependent on hypervisor status
  Tsx async abort:  Not affected

openssl version: 3.0.5
built on: Tue Jul  5 00:00:00 2022 UTC
options: bn(64,64)
compiler: gcc -fPIC -pthread -m64 -Wa,--noexecstack -O2 -flto=auto -ffat-
lto-objects -fexceptions -g -grecord-gcc-switches -pipe -Wall -Werror=
format-security -Wp,-D_FORTIFY_SOURCE=2 -Wp,-D_GLIBCXX_ASSERTIONS -
specs=/usr/lib/rpm/redhat/redhat-hardened-cc1 -fstack-protector-strong
-specs=/usr/lib/rpm/redhat/redhat-annobin-cc1 -m64 -mtune=generic -
fasynchronous-unwind-tables -fstack-clash-protection -fcf-protection -

```



```

02 -flto=auto -ffat-lto-objects -fexceptions -g -grecord-gcc-switches -
pipe -Wall -Werror=format-security -Wp,-D_FORTIFY_SOURCE=2 -Wp,-
D_GLIBCXX_ASSERTIONS -specs=/usr/lib/rpm/redhat/redhat-hardened-cc1 -
fstack-protector-strong -specs=/usr/lib/rpm/redhat/redhat-annobin-cc1 -
m64 -mtune=generic -fasynchronous-unwind-tables -fstack-clash-
protection -fcf-protection -Wa,--noexecstack -Wa,--generate-missing-
build-notes=yes -specs=/usr/lib/rpm/redhat/redhat-hardened-ld -specs=/
usr/lib/rpm/redhat/redhat-annobin-cc1 -DOPENSSL_USE_NODELETE -DL_ENDIAN
-DOPENSSL_PIC -DOPENSSL_BUILDING_OPENSSL -DZLIB -DDEBUG -DPURIFY -
DDEV_RANDOM="/dev/urandom\" -DSYSTEM_CIPHERS_FILE="/etc/crypto-
policies/back-ends/openssl.config"

```

The 'numbers' are in 1000s of bytes per second processed.

type	16 bytes	64 bytes	256 bytes	1024 bytes	8192
	bytes	bytes			
AES-128-CBC *	357381.06k	414960.06k	416301.23k	416687.10k	
	416175.45k	416268.29k			
AES-128-CBC	902160.83k	1496344.68k	1514778.62k	1555236.52k	
	1542537.22k	1569259.52k			
AES-128-CBC d	909710.79k	2941259.46k	5167110.31k	5927086.76k	
	6365967.70k	6349198.68k			
AES-128-CBC 8	3912786.36k	8042348.31k	9870507.86k	10254096.38k	
	10653332.82k	10310331.05k			
AES-128-CBC 8d	4157037.26k	12337480.36k	26613686.27k	29902703.27k	
	32306793.13k	31440366.25k			
AES-128-CTR *	146971.83k	165696.94k	574871.64k	634507.61k	
	676448.94k	668139.52k			
AES-128-CTR	887783.06k	2255074.22k	4800168.19k	5930596.01k	
	6431110.49k	6376062.98k			
AES-128-CTR d	793432.63k	2181439.06k	4541298.09k	5743022.42k	
	6480090.45k	6271221.76k			
AES-128-CTR 8	3833975.47k	10832239.55k	23757293.40k	28413146.79k	
	30514317.99k	30092356.27k			
AES-128-CTR 8d	3456838.44k	9749773.91k	22107652.18k	27229352.28k	
	30703026.18k	29387025.07k			
AES-192-CBC	853380.50k	1238507.90k	1299788.12k	1257189.03k	
	1272591.70k	1271840.77k			
AES-192-CBC d	876094.29k	2843770.82k	4523019.52k	5177496.92k	
	5442652.84k	5372559.36k			
AES-192-CTR	869039.84k	2285946.18k	4229439.91k	5049118.04k	
	5422994.77k	5309748.57k			
AES-192-CTR d	789470.51k	2177050.05k	4194812.76k	4935891.63k	
	5257865.90k	5323046.91k			
AES-256-CBC	834298.24k	1100648.64k	1117826.90k	1104301.40k	
	1130657.11k	1097285.63k			
AES-256-CBC d	843079.68k	2714917.67k	4084088.23k	4510005.59k	
	4557821.27k	4594783.57k			
AES-256-CTR	811325.74k	2222582.89k	3749333.08k	4412143.27k	
	4640549.55k	4554828.46k			

AES-256-CTR d	730844.97k	2081179.20k	3673258.15k	4346793.64k
	4515722.58k	4594335.74k		

*: AES-NI disabled; 8: 8 cores; d: decryption

7 RPM

pgmoneta can be built into a RPM for Fedora systems.

7.1 Requirements

```
dnf install gcc rpm-build rpm-devel rpmlint make python bash coreutils
diffutils patch rpmdevtools chrpath
```

7.2 Setup RPM development

```
rpmdev-setuptree
```

7.3 Create source package

```
git clone https://github.com/pgmoneta/pgmoneta.git
cd pgmoneta
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
make package_source
```

7.4 Create RPM package

```
cp pgmoneta-$VERSION.tar.gz ~/rpmbuild/SOURCES
QA_RPATHS=0x0001 rpmbuild -bb pgmoneta.spec
```

The resulting RPM will be located in `~/rpmbuild/RPMS/x86_64/`, if your architecture is `x86_64`.

8 Test

8.1 Container Environment

8.1.1 Docker

First, ensure your system is up to date.

```
dnf update
```

Install the necessary packages for Docker.

```
dnf -y install dnf-plugins-core
```

Add the Docker repository to your system.

```
sudo dnf config-manager --add-repo https://download.docker.com/linux/fedora/docker-ce.repo
```

Install Docker Engine, Docker CLI, and Containerd.

```
sudo dnf install docker-ce docker-ce-cli containerd.io
```

Start the Docker service and enable it to start on boot.

```
sudo systemctl start docker
sudo systemctl enable docker
```

Verify that Docker is installed correctly.

```
docker --version
```

If you see the Docker version, then you have successfully installed Docker on Fedora.

8.1.2 Podman

Install Podman and the Docker alias package.

```
dnf install podman podman-docker.noarch
```

Verify that Podman is installed correctly.

```
podman --version
```

If you see the Podman version, then you have successfully installed Podman on Fedora.

The `podman-docker.noarch` package simplifies the use of `Podman` for users accustomed to `Docker`.

8.2 Test suite

You can simply use `CTest` to test all PostgreSQL versions from 13 to 16. It will automatically run `testsuite.sh` to test `pgmoneta` and `pgmoneta_ext` for each version. The script will automatically create the `Docker` container, run it, and then use the `check` framework to test their functions inside it. After that, it will automatically clean up everything for you.

Go to the directory `/pgmoneta/test`, and give permission to `testsuite.sh` using:

```
chmod +x testsuite.sh
```

After you follow the `DEVELOPERS.md` to install `pgmoneta`, go to the directory `/pgmoneta/build` and run the test.

```
make test
```

`CTest` will output logs into `/pgmoneta/build/Testing/Temporary/LastTest.log`. If you want to check the specific process, you can review that log file.

`testsuite.sh` accepts three variables. The first one is `dir`, which specifies the `/test` directory location, with a default value of `./`. The second one is `dockerfile`, with a default value of `Dockerfile.rocky8`. The third one is the PostgreSQL `version`, with a default value of 13.

9 WAL Reader

9.1 Overview

This document provides an overview of the `wal_reader` tool, with a focus on the `parse_wal_file` function, which serves as the main entry point for parsing Write-Ahead Log (WAL) files. Currently, the function only parses the given WAL file and prints the description of each record. In the future, it will be integrated with other parts of the code.

9.2 pgmoneta-walinfo

`pgmoneta-walinfo` is a command line utility designed to read and display information about PostgreSQL Write-Ahead Log (WAL) files. The tool provides output in either raw or JSON format, making it easy to analyze WAL files for debugging, auditing, or general information purposes.

In addition to standard WAL files, `pgmoneta-walinfo` also supports encrypted (**aes**) and compressed WAL files in the following formats: **zstd**, **gz**, **lz4**, and **bz2**.

9.2.0.1 Usage

```
pgmoneta-walinfo
  Command line utility to read and display Write-Ahead Log (WAL) files
```

Usage:

```
pgmoneta-walinfo <file>
```

Options:

```
-c, --config CONFIG_FILE Set the path to the pgmoneta.conf file
-o, --output FILE         Output file
-F, --format              Output format (raw, json)
-L, --logfile FILE        Set the log file
-q, --quiet               No output only result
    --color               Use colors (on, off)
-v, --verbose             Output result
-V, --version             Display version information
-?, --help                Display help
```

9.2.0.2 Raw Output Format In `raw` format, the default, the output is structured as follows:

```
Resource Manager | Start LSN | End LSN | rec len | tot len | xid |
description (data and backup)
```

- **Resource Manager:** The name of the resource manager handling the log record.
- **Start LSN:** The start Log Sequence Number (LSN).

- **End LSN:** The end Log Sequence Number (LSN).
- **rec len:** The length of the WAL record.
- **tot len:** The total length of the WAL record, including the header.
- **xid:** The transaction ID associated with the record.
- **description (data and backup):** A detailed description of the operation, along with any related backup block information.

Each part of the output is color-coded:

- **Red:** Header information (resource manager, record length, transaction ID, etc.).
- **Green:** Description of the WAL record.
- **Blue:** Backup block references or additional data.

This format makes it easy to visually distinguish different parts of the WAL file for quick analysis.

9.2.0.3 Example To view WAL file details in JSON format:

```
pgmoneta-walinfo -F json /path/to/walfile
```

9.3 High-Level API Overview

The following section provides a high-level overview of how users can interact with the functions and structures defined in the `walfile.h` file. These APIs allow you to read, write, and manage Write-Ahead Log (WAL) files.

9.3.1 Struct `walfile`

The `walfile` struct represents the core structure used for interacting with WAL files in PostgreSQL. A WAL file stores a log of changes to the database and is used for crash recovery, replication, and other purposes. Each WAL file consists of pages (each 8192 bytes by default), containing records that capture database changes.

9.3.1.1 Fields:

- **magic_number:** Identifies the PostgreSQL version that created the WAL file. You can find more info on supported magic numbers [here](#).
- **long_phd:** A pointer to the extended header (long header) found on the first page of the WAL file. This header contains additional metadata.

- **page_headers:** A deque of headers representing each page in the WAL file, excluding the first page.
- **records:** A deque of decoded WAL records. Each record represents a change made to the database and contains both metadata and the actual data to be applied during recovery or replication.

9.3.2 Function Overview

The `walfile.h` file provides three key functions for interacting with WAL files: `pgmoneta_read_walfile`, `pgmoneta_write_walfile`, and `pgmoneta_destroy_walfile`. These functions allow users to read from, write to, and destroy WAL file objects, respectively.

9.3.2.1 `pgmoneta_read_walfile`

```
int pgmoneta_read_walfile(int server, char* path, struct walfile** wf);
```

9.3.2.1.1 Description: This function reads a WAL file from a specified path and populates a `walfile` structure with its contents, including the file's headers and records.

9.3.2.1.2 Parameters:

- **server:** The index of the Postgres server in Pgmoneta configuration.
- **path:** The file path to the WAL file that needs to be read.
- **wf:** A pointer to a pointer to a `walfile` structure that will be populated with the WAL file data.

9.3.2.1.3 Return:

- Returns 0 on success or 1 on failure.

9.3.2.1.4 Usage Example:

```
struct walfile* wf = NULL;
int result = pgmoneta_read_walfile(0, "/path/to/walfile", &wf);
if (result == 0) {
    // Successfully read WAL file
}
```

9.3.2.2 `pgmoneta_write_walfile`

```
int pgmoneta_write_walfile(struct walfile* wf, int server, char* path);
```

9.3.2.2.1 Description: This function writes the contents of a `walfile` structure back to disk, saving it as a WAL file at the specified path.

9.3.2.2.2 Parameters:

- **wf**: The `walfile` structure containing the WAL data to be written.
- **server**: The index or ID of the server where the WAL file should be saved.
- **path**: The file path where the WAL file should be written.

9.3.2.2.3 Return:

- Returns 0 on success or 1 on failure.

9.3.2.2.4 Usage Example:

```
int result = pgmoneta_write_walfile(wf, 0, "/path/to/output_walfile");
if (result == 0) {
    // Successfully wrote WAL file
}
```

9.3.2.3 pgmoneta_destroy_walfile

```
void pgmoneta_destroy_walfile(struct walfile* wf);
```

9.3.2.3.1 Description: This function frees the memory allocated for a `walfile` structure, including its headers and records.

9.3.2.3.2 Parameters:

- **wf**: The `walfile` structure to be destroyed.

9.3.2.3.3 Usage Example:

```
struct walfile* wf = NULL;
int result = pgmoneta_read_walfile(0, "/path/to/walfile", &wf);
if (result == 0) {
    // Successfully read WAL file
}
pgmoneta_destroy_walfile(wf);
```

9.4 Internal API Overview

9.4.1 parse_wal_file

This function is responsible for reading and parsing a PostgreSQL Write-Ahead Log (WAL) file.

9.4.1.1 Parameters

- **path**: The file path to the WAL file that needs to be parsed.
- **server_info**: A pointer to a `server` structure containing information about the server.

9.4.1.2 Description The `parse_wal_file` function opens the WAL file specified by the `path` parameter in binary mode and reads the WAL records. It processes these records, handling various cases such as records that cross page boundaries, while ensuring correct memory management throughout the process.

9.4.2 Usage Example

```
parse_wal_file("/path/to/wal/file", &my_server);
```

9.4.3 WAL File Structure

The image illustrates the structure of a WAL (Write-Ahead Logging) file in PostgreSQL, focusing on how XLOG records are organized within WAL segments.

Source: <https://www.interdb.jp/pg/pgsql09/03.html>

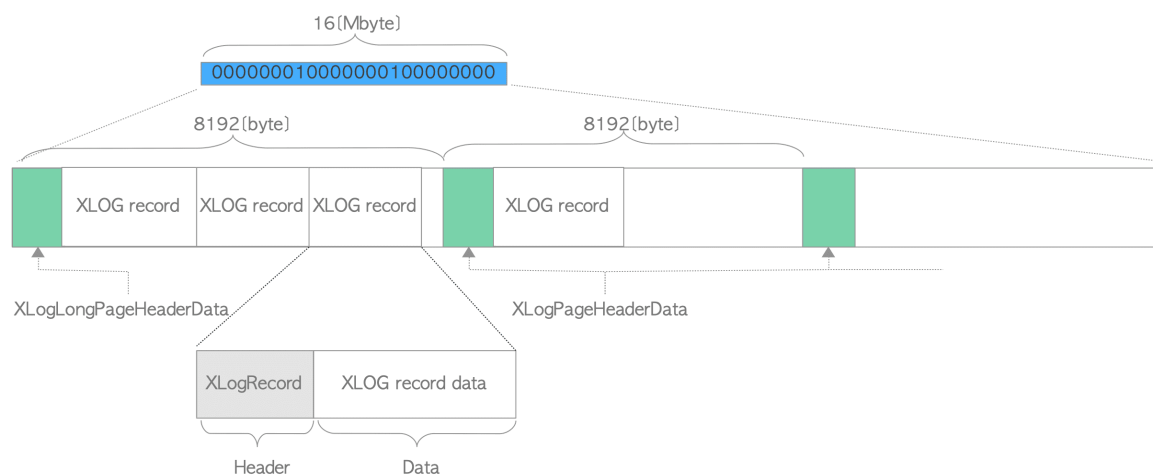


Figure 1: WAL File Structure

A WAL segment, by default, is a 16 MB file, divided into pages of 8192 bytes (8 KB) each. The first page contains a header defined by the `XLogLongPageHeaderData` structure, while all subsequent pages

have headers described by the `XLogPageHeaderData` structure. XLOG records are written sequentially in each page, starting at the beginning and moving downward.

The figure highlights how the WAL ensures data consistency by sequentially writing XLOG records in pages, structured within larger 16 MB WAL segments.

9.5 Resource Managers

In the context of the WAL reader, resource managers (rm) are responsible for handling different types of records found within a WAL file. Each record in the WAL file is associated with a specific resource manager, which determines how that record is processed.

9.5.1 Resource Manager Definitions

Each resource manager is defined in the `rm_[name].h` header file and implemented in the corresponding `rm_[name].c` source file.

In the `rmgr.h` header file, the resource managers are declared as an enum, with each resource manager having a unique identifier.

9.5.2 Resource Manager Functions

Each resource manager implements the `rm_desc` function, which provides a description of the record type associated with that resource manager. In the future, they will be extended to implement the `rm_redo` function to apply the changes to another server.

9.5.3 Supporting Various WAL Structures in PostgreSQL Versions 13 to 17

The WAL structure has evolved across PostgreSQL versions 13 to 17, requiring different handling for each version. To accommodate these differences, we have implemented a wrapper-based approach, such as the factory pattern, to handle varying WAL structures.

Below are the commit hashes for the officially supported magic values in each PostgreSQL version:

1. PostgreSQL 13 - 0xD106: <https://github.com/postgres/postgres/commit/c6b92041d38512a4176ed76ad06f713d2e>
2. PostgreSQL 14 - 0xD10D: <https://github.com/postgres/postgres/commit/08aa89b326261b669648df97d4f2a6edba>

3. PostgreSQL 15 - 0xD110: <https://github.com/postgres/postgres/commit/8b1dccd37c71ed2ff016294d8f9053a32b0>
4. PostgreSQL 16 - 0xD113: <https://github.com/postgres/postgres/commit/6af1793954e8c5e753af83c3edb37ed326>
5. PostgreSQL 17 - 0xD116: <https://github.com/postgres/postgres/commit/402b586d0a9caae9412d25fcf1b91dae45>

`xl_end_of_recovery` is an example of how we handle different versions of structures with a wrapper struct and a factory pattern.

```
struct xl_end_of_recovery_v16 {
    timestamp_tz end_time;
    timeline_id this_timeline_id;
    timeline_id prev_timeline_id;
};

struct xl_end_of_recovery_v17 {
    timestamp_tz end_time;
    timeline_id this_timeline_id;
    timeline_id prev_timeline_id;
    int wal_level;
};

struct xl_end_of_recovery {
    int pg_version;
    union {
        struct xl_end_of_recovery_v16 v16;
        struct xl_end_of_recovery_v17 v17;
    } data;
    void (*parse)(struct xl_end_of_recovery* wrapper, const void* rec);
    char* (*format)(struct xl_end_of_recovery* wrapper, char* buf);
};

xl_end_of_recovery* create_xl_end_of_recovery(int pg_version) {
    xl_end_of_recovery* wrapper = malloc(sizeof(xl_end_of_recovery));
    wrapper->pg_version = pg_version;

    if (pg_version >= 17) {
        wrapper->parse = parse_v17;
        wrapper->format = format_v17;
    } else {
        wrapper->parse = parse_v16;
        wrapper->format = format_v16;
    }

    return wrapper;
}

void parse_v16(xl_end_of_recovery* wrapper, const void* rec) {
    memcpy(&wrapper->data.v16, rec, sizeof(struct xl_end_of_recovery_v16))
```

```
    ;
}

void parse_v17(xl_end_of_recovery* wrapper, const void* rec) {
    memcpy(&wrapper->data.v17, rec, sizeof(struct xl_end_of_recovery_v17))
    ;
}

char* format_v16(xl_end_of_recovery* wrapper, char* buf) {
    struct xl_end_of_recovery_v16* xlrec = &wrapper->data.v16;
    return pgmoneta_format_and_append(buf, "tli %u; prev tli %u; time %s",
                                      xlrec->this_timeline_id, xlrec->
                                      prev_timeline_id,
                                      pgmoneta_wal_timestampz_to_str(
                                      xlrec->end_time));
}

char* format_v17(xl_end_of_recovery* wrapper, char* buf) {
    struct xl_end_of_recovery_v17* xlrec = &wrapper->data.v17;
    return pgmoneta_format_and_append(buf, "tli %u; prev tli %u; time %s;
        wal_level %d",
                                      xlrec->this_timeline_id, xlrec->
                                      prev_timeline_id,
                                      pgmoneta_wal_timestampz_to_str(
                                      xlrec->end_time),
                                      xlrec->wal_level);
}
```

9.6 WAL Change List

This section lists the changes in the WAL format between different versions of PostgreSQL.

9.6.1 xl_clog_truncate

17

```
struct xl_clog_truncate
{
    int64 pageno;                /**< The page number of the clog to truncate
        */
    transaction_id oldestXact;   /**< The oldest transaction ID to retain */
    oid oldestXactDb;           /**< The database ID of the oldest transaction
        */
};
```

16

```
struct xl_clog_truncate
{
    int64 pageno;                /**< The page number of the clog to truncate
        */
    transaction_id oldestXact;    /**< The oldest transaction ID to retain */
    oid oldestXactDb;            /**< The database ID of the oldest transaction
        */
};
```

9.6.2 xl_commit_ts_truncate

17:

```
typedef struct xl_commit_ts_truncate
{
    int64 pageno;
    TransactionId oldestXid;
} xl_commit_ts_truncate;
```

16:

```
typedef struct xl_commit_ts_truncate
{
    int pageno;
    TransactionId oldestXid;
} xl_commit_ts_truncate;
```

9.6.3 xl_heap_prune

17:

```
typedef struct xl_heap_prune
{
    uint8 reason;
    uint8 flags;

    /*
     * If XLHP_HAS_CONFLICT_HORIZON is set, the conflict horizon XID
     * follows,
     * unaligned
     */
} xl_heap_prune;
#define SizeOfHeapPrune (offsetof(xl_heap_prune, flags) + sizeof(uint8))
```

16:

```
typedef struct xl_heap_prune
{
    TransactionId snapshotConflictHorizon;
    uint16        nredirected;
    uint16        ndead;
    bool          isCatalogRel; /* to handle recovery conflict during
                                logical
                                * decoding on standby */
    /* OFFSET NUMBERS are in the block reference 0 */
} xl_heap_prune;
#define SizeOfHeapPrune (offsetof(xl_heap_prune, isCatalogRel) + sizeof(
    bool))
```

9.6.4 xlhp_freeze_plan

Removed `xl_heap_freeze_page`

17:

```
typedef struct xlhp_freeze_plan
{
    TransactionId xmax;
    uint16        t_infomask2;
    uint16        t_infomask;
    uint8         frzflags;

    /* Length of individual page offset numbers array for this plan */
    uint16        ntuples;
} xlhp_freeze_plan;
```

9.6.5 spgxlogState

(Doesn't need to be changed)

17:

```
typedef struct spgxlogState
{
    TransactionId redirectXid;
    bool         isBuild;
} spgxlogState;
```

16:

```
typedef struct spgxlogState
{
    TransactionId myXid;
```

```
    bool        isBuild;  
} spgxlogState;
```

9.6.6 xl_end_of_recovery

```
typedef struct xl_end_of_recovery  
{  
    TimestampTz end_time;  
    TimeLineID  ThisTimeLineID; /* new TLI */  
    TimeLineID  PrevTimeLineID; /* previous TLI we forked off from */  
    int         wal_level;  
} xl_end_of_recovery;
```

16:

```
typedef struct xl_end_of_recovery  
{  
    TimestampTz end_time;  
    TimeLineID  ThisTimeLineID; /* new TLI */  
    TimeLineID  PrevTimeLineID; /* previous TLI we forked off from */  
} xl_end_of_recovery;
```

16 → 15

9.6.7 ginxlogSplit

16: same for gin_xlog_update_meta

```
typedef struct ginxlogSplit  
{  
    RelFileLocator locator;  
    BlockNumber rrlink;          /* right link, or root's blocknumber if  
    root                        * split */  
    BlockNumber leftChildBlkno; /* valid on a non-leaf split */  
    BlockNumber rightChildBlkno;  
    uint16      flags;          /* see below */  
} ginxlogSplit;
```

15:

```
typedef struct ginxlogSplit  
{
```



```
    RelFileNode node;
    BlockNumber rlink;          /* right link, or root's blocknumber if
                                root
                                * split */
    BlockNumber leftChildBlkno; /* valid on a non-leaf split */
    BlockNumber rightChildBlkno;
    uint16      flags;          /* see below */
} ginxlogSplit;
```

9.6.8 gistxlogDelete

16:

```
typedef struct gistxlogDelete
{
    TransactionId snapshotConflictHorizon;
    uint16      ntodelete;      /* number of deleted offsets */
    bool        isCatalogRel;   /* to handle recovery conflict during
                                logical
                                * decoding on standby */

    /* TODOLETE OFFSET NUMBERS */
    OffsetNumber offsets[FLEXIBLE_ARRAY_MEMBER];
} gistxlogDelete;
#define SizeOfGistxlogDelete    offsetof(gistxlogDelete, offsets)
```

15:

```
typedef struct gistxlogDelete
{
    TransactionId latestRemovedXid;
    uint16      ntodelete;      /* number of deleted offsets */

    /*
     * In payload of blk 0 : todelete OffsetNumbers
     */
} gistxlogDelete;
#define SizeOfGistxlogDelete    (offsetof(gistxlogDelete, ntodelete) +
    sizeof(uint16))
```

9.6.9 gistxlogPageReuse

16:

```
typedef struct gistxlogPageReuse
{
    RelFileLocator locator;
```

```
    BlockNumber block;
    FullTransactionId snapshotConflictHorizon;
    bool          isCatalogRel; /* to handle recovery conflict during
        logical
                                * decoding on standby */
} gistxlogPageReuse;
#define SizeOfGistxlogPageReuse (offsetof(gistxlogPageReuse, isCatalogRel)
    + sizeof(bool))
```

15:

```
typedef struct gistxlogPageReuse
{
    RelFileNode node;
    BlockNumber block;
    FullTransactionId latestRemovedFullXid;
} gistxlogPageReuse;

#define SizeOfGistxlogPageReuse (offsetof(gistxlogPageReuse,
    latestRemovedFullXid) + sizeof(FullTransactionId))
```

9.6.10 xl_hash_vacuum_one_page

16:

```
typedef struct xl_hash_vacuum_one_page
{
    TransactionId snapshotConflictHorizon;
    uint16        ntuples;
    bool          isCatalogRel; /* to handle recovery conflict during
        logical
                                * decoding on standby */

    /* TARGET OFFSET NUMBERS */
    OffsetNumber offsets[FLEXIBLE_ARRAY_MEMBER];
} xl_hash_vacuum_one_page;
#define SizeOfHashVacuumOnePage offsetof(xl_hash_vacuum_one_page, offsets)
```

15:

```
typedef struct xl_hash_vacuum_one_page
{
    TransactionId latestRemovedXid;
    int          ntuples;

    /* TARGET OFFSET NUMBERS FOLLOW AT THE END */
} xl_hash_vacuum_one_page;
#define SizeOfHashVacuumOnePage \
    (offsetof(xl_hash_vacuum_one_page, ntuples) + sizeof(int))
```

9.6.11 xl_heap_prune

16:

```
typedef struct xl_heap_prune
{
    TransactionId snapshotConflictHorizon;
    uint16        nredirected;
    uint16        ndead;
    bool          isCatalogRel; /* to handle recovery conflict during
                                logical
                                * decoding on standby */
    /* OFFSET NUMBERS are in the block reference 0 */
} xl_heap_prune;
#define SizeOfHeapPrune (offsetof(xl_heap_prune, isCatalogRel) + sizeof(
    bool))
```

15:

```
typedef struct xl_heap_prune
{
    TransactionId latestRemovedXid;
    uint16        nredirected;
    uint16        ndead;
    /* OFFSET NUMBERS are in the block reference 0 */
} xl_heap_prune;
#define SizeOfHeapPrune (offsetof(xl_heap_prune, ndead) + sizeof(uint16))
```

9.6.12 xl_heap_freeze_plan

16:

```
typedef struct xl_heap_freeze_plan
{
    TransactionId xmax;
    uint16        t_infomask2;
    uint16        t_infomask;
    uint8         frzflags;

    /* Length of individual page offset numbers array for this plan */
    uint16        ntuples;
} xl_heap_freeze_plan;
```

15:

```
typedef struct xl_heap_freeze_tuple
{
    TransactionId xmax;
    OffsetNumber offset;
```

```
uint16    t_infomask2;
uint16    t_infomask;
uint8     frzflags;
} xl_heap_freeze_tuple;
```

9.6.13 xl_heap_freeze_page

16:

```
typedef struct xl_heap_freeze_page
{
    TransactionId snapshotConflictHorizon;
    uint16        nplans;
    bool          isCatalogRel; /* to handle recovery conflict during
        logical                      * decoding on standby */

    /*
     * In payload of blk 0 : FREEZE PLANS and OFFSET NUMBER ARRAY
     */
} xl_heap_freeze_page;
```

15:

```
typedef struct xl_heap_freeze_page
{
    TransactionId cutoff_xid;
    uint16        ntuples;
} xl_heap_freeze_page;
```

9.6.14 xl_btree_reuse_page

16:

```
typedef struct xl_btree_reuse_page
{
    RelFileLocator locator;
    BlockNumber block;
    FullTransactionId snapshotConflictHorizon;
    bool          isCatalogRel; /* to handle recovery conflict during
        logical                      * decoding on standby */
} xl_btree_reuse_page;
```

15:

```
typedef struct xl_btree_reuse_page
```

```
{
    RelFileNode node;
    BlockNumber block;
    FullTransactionId latestRemovedFullXid;
} xl_btree_reuse_page;
```

9.6.15 xl_btree_delete

16:

```
typedef struct xl_btree_delete
{
    TransactionId snapshotConflictHorizon;
    uint16      ndeleted;
    uint16      nupdated;
    bool        isCatalogRel; /* to handle recovery conflict during
                               logical                               * decoding on standby */

    /*-----
     * In payload of blk 0 :
     * - DELETED TARGET OFFSET NUMBERS
     * - UPDATED TARGET OFFSET NUMBERS
     * - UPDATED TUPLES METADATA (xl_btree_update) ARRAY
     *-----
     */
} xl_btree_delete;
```

15:

```
typedef struct xl_btree_delete
{
    TransactionId latestRemovedXid;
    uint16      ndeleted;
    uint16      nupdated;

    /* DELETED TARGET OFFSET NUMBERS FOLLOW */
    /* UPDATED TARGET OFFSET NUMBERS FOLLOW */
    /* UPDATED TUPLES METADATA (xl_btree_update) ARRAY FOLLOWS */
} xl_btree_delete;
```

9.6.16 spgxlogVacuumRedirect

16:

```
typedef struct spgxlogVacuumRedirect
{
```

```
uint16      nToPlaceholder; /* number of redirects to make
                             placeholders */
OffsetNumber firstPlaceholder; /* first placeholder tuple to remove
                             */
TransactionId snapshotConflictHorizon; /* newest XID of removed
                             redirects */
bool         isCatalogRel; /* to handle recovery conflict during
                             logical
                             * decoding on standby */

/* offsets of redirect tuples to make placeholders follow */
OffsetNumber offsets[FLEXIBLE_ARRAY_MEMBER];
} spgxlogVacuumRedirect;
```

15:

```
typedef struct spgxlogVacuumRedirect
{
    uint16      nToPlaceholder; /* number of redirects to make
                             placeholders */
    OffsetNumber firstPlaceholder; /* first placeholder tuple to remove
                             */
    TransactionId newestRedirectXid; /* newest XID of removed redirects
                             */

    /* offsets of redirect tuples to make placeholders follow */
    OffsetNumber offsets[FLEXIBLE_ARRAY_MEMBER];
} spgxlogVacuumRedirect;
```

15 → 14

9.6.17 xl_xact_prepare

15:

```
ctypedef struct xl_xact_prepare
{
    uint32      magic; /* format identifier */
    uint32      total_len; /* actual file length */
    TransactionId xid; /* original transaction XID */
    Oid          database; /* OID of database it was in */
    TimestampTz prepared_at; /* time of preparation */
    Oid          owner; /* user running the transaction */
    int32        nsubxacts; /* number of following subxact XIDs */
    int32        ncommitrels; /* number of delete-on-commit rels */
    int32        nabortrels; /* number of delete-on-abort rels */
}
```

```
int32      ncommitstats; /* number of stats to drop on commit */
int32      nabortstats;  /* number of stats to drop on abort */
int32      ninvalmsgs;   /* number of cache invalidation messages
 */
bool       initfileinval; /* does relcache init file need
invalidation? */
uint16     gidlen;       /* length of the GID - GID follows the
header */
XLogRecPtr origin_lsn;   /* lsn of this record at origin node */
TimestampTz origin_timestamp; /* time of prepare at origin node */
} xl_xact_prepare;
```

14:

```
typedef struct xl_xact_prepare
{
    uint32      magic;      /* format identifier */
    uint32      total_len;  /* actual file length */
    TransactionId xid;      /* original transaction XID */
    Oid         database;   /* OID of database it was in */
    TimestampTz prepared_at; /* time of preparation */
    Oid         owner;      /* user running the transaction */
    int32       nsubxacts;  /* number of following subxact XIDs */
    int32       ncommitrels; /* number of delete-on-commit rels */
    int32       nabortrels; /* number of delete-on-abort rels */
    int32       ninvalmsgs; /* number of cache invalidation messages
 */
    bool       initfileinval; /* does relcache init file need
invalidation? */
    uint16     gidlen;      /* length of the GID - GID follows the
header */
    XLogRecPtr origin_lsn;   /* lsn of this record at origin node */
    TimestampTz origin_timestamp; /* time of prepare at origin node */
} xl_xact_prepare;
```

9.6.18 xl_xact_parsed_commit

15:

```
typedef struct xl_xact_parsed_commit
{
    TimestampTz xact_time;
    uint32      xinfo;

    Oid         dbId;      /* MyDatabaseId */
    Oid         tsId;      /* MyDatabaseTableSpace */

    int         nsubxacts;
    TransactionId *subxacts;
```

```
    int         nrels;
    RelFileNode *xnodes;

    int         nstats;
    xl_xact_stats_item *stats;

    int         nmsgs;
    SharedInvalidationMessage *msgs;

    TransactionId twophase_xid; /* only for 2PC */
    char         twophase_gid[GIDSIZE]; /* only for 2PC */
    int         nabortrels; /* only for 2PC */
    RelFileNode *abortnodes; /* only for 2PC */
    int         nabortstats; /* only for 2PC */
    xl_xact_stats_item *abortstats; /* only for 2PC */

    XLogRecPtr  origin_lsn;
    TimestampTz origin_timestamp;
} xl_xact_parsed_commit;
```

14:

```
typedef struct xl_xact_parsed_commit
{
    TimestampTz xact_time;
    uint32      xinfo;

    Oid         dbId; /* MyDatabaseId */
    Oid         tsId; /* MyDatabaseTableSpace */

    int         nsubxacts;
    TransactionId *subxacts;

    int         nrels;
    RelFileNode *xnodes;

    int         nmsgs;
    SharedInvalidationMessage *msgs;

    TransactionId twophase_xid; /* only for 2PC */
    char         twophase_gid[GIDSIZE]; /* only for 2PC */
    int         nabortrels; /* only for 2PC */
    RelFileNode *abortnodes; /* only for 2PC */

    XLogRecPtr  origin_lsn;
    TimestampTz origin_timestamp;
} xl_xact_parsed_commit;
```


9.6.19 xl_xact_parsed_abort

15:

```
typedef struct xl_xact_parsed_abort
{
    TimestampTz xact_time;
    uint32      xinfo;

    Oid          dbId;          /* MyDatabaseId */
    Oid          tsId;          /* MyDatabaseTableSpace */

    int          nsubxacts;
    TransactionId *subxacts;

    int          nrels;
    RelFileNode *xnodes;

    int          nstats;
    xl_xact_stats_item *stats;

    TransactionId twophase_xid; /* only for 2PC */
    char          twophase_gid[GIDSIZE]; /* only for 2PC */

    XLogRecPtr   origin_lsn;
    TimestampTz  origin_timestamp;
} xl_xact_parsed_abort;
```

14:

```
typedef struct xl_xact_parsed_abort
{
    TimestampTz xact_time;
    uint32      xinfo;

    Oid          dbId;          /* MyDatabaseId */
    Oid          tsId;          /* MyDatabaseTableSpace */

    int          nsubxacts;
    TransactionId *subxacts;

    int          nrels;
    RelFileNode *xnodes;

    TransactionId twophase_xid; /* only for 2PC */
    char          twophase_gid[GIDSIZE]; /* only for 2PC */

    XLogRecPtr   origin_lsn;
    TimestampTz  origin_timestamp;
} xl_xact_parsed_abort;
```

9.6.20 xlogrecord.h flags

15:

```
#define BKPIMAGE_APPLY          0x02    /* page image should be restored
                                         * during replay */

/* compression methods supported */
#define BKPIMAGE_COMPRESS_PGLZ 0x04
#define BKPIMAGE_COMPRESS_LZ4  0x08
#define BKPIMAGE_COMPRESS_ZSTD 0x10

#define BKPIMAGE_COMPRESSED(info) \
    ((info & (BKPIMAGE_COMPRESS_PGLZ | BKPIMAGE_COMPRESS_LZ4 | \
             BKPIMAGE_COMPRESS_ZSTD)) != 0)
```

14:

```
#define BKPIMAGE_IS_COMPRESSED 0x02    /* page image is compressed */
#define BKPIMAGE_APPLY        0x04    /* page image should be restored
                                         during
                                         * replay */
```

14 → 13

9.6.21 xl_heap_prune

14:

```
typedef struct xl_heap_prune
{
    TransactionId latestRemovedXid;
    uint16        nredirected;
    uint16        ndead;
    /* OFFSET NUMBERS are in the block reference 0 */
} xl_heap_prune;
```

13:

```
typedef struct xl_heap_clean
{
    TransactionId latestRemovedXid;
    uint16        nredirected;
    uint16        ndead;
    /* OFFSET NUMBERS are in the block reference 0 */
} xl_heap_clean;
```

9.6.22 xl_heap_vacuum

14:

```
typedef struct xl_heap_vacuum
{
    uint16      nunused;
    /* OFFSET NUMBERS are in the block reference 0 */
} xl_heap_vacuum;
```

13:

```
typedef struct xl_heap_cleanup_info
{
    RelFileNode node;
    TransactionId latestRemovedXid;
} xl_heap_cleanup_info;
```

9.6.23 xl_btree_metadata

14:

```
typedef struct xl_btree_metadata
{
    uint32      version;
    BlockNumber root;
    uint32      level;
    BlockNumber fastroot;
    uint32      fastlevel;
    uint32      last_cleanup_num_delpages;
    bool        allequalimage;
} xl_btree_metadata;
```

13:

```
typedef struct xl_btree_metadata
{
    uint32      version;
    BlockNumber root;
    uint32      level;
    BlockNumber fastroot;
    uint32      fastlevel;
    TransactionId oldest_btpo_xact;
    float8      last_cleanup_num_heap_tuples;
    bool        allequalimage;
} xl_btree_metadata;
```

9.6.24 xl_btree_reuse_page

14:

```
typedef struct xl_btree_reuse_page
{
    RelFileNode node;
    BlockNumber block;
    FullTransactionId latestRemovedFullXid;
} xl_btree_reuse_page;
```

13:

```
typedef struct xl_btree_reuse_page
{
    RelFileNode node;
    BlockNumber block;
    TransactionId latestRemovedXid;
} xl_btree_reuse_page;
```

9.6.25 xl_btree_delete

14:

```
typedef struct xl_btree_delete
{
    TransactionId latestRemovedXid;
    uint16      ndeleted;
    uint16      nupdated;

    /* DELETED TARGET OFFSET NUMBERS FOLLOW */
    /* UPDATED TARGET OFFSET NUMBERS FOLLOW */
    /* UPDATED TUPLES METADATA (xl_btree_update) ARRAY FOLLOWS */
} xl_btree_delete;
```

13:

```
typedef struct xl_btree_delete
{
    TransactionId latestRemovedXid;
    uint32      ndeleted;

    /* DELETED TARGET OFFSET NUMBERS FOLLOW */
} xl_btree_delete;
```

9.6.26 xl_btree_unlink_page

14:

```
typedef struct xl_btree_unlink_page
{
    BlockNumber leftsib;          /* target block's left sibling, if any */
    BlockNumber rightsib;         /* target block's right sibling */
    uint32      level;            /* target block's level */
    FullTransactionId safexid;    /* target block's BTPageSetDeleted() XID
        */

    /*
     * Information needed to recreate a half-dead leaf page with correct
     * topparent link. The fields are only used when deletion operation's
     * target page is an internal page. REDO routine creates half-dead
     * page
     * from scratch to keep things simple (this is the same convenient
     * approach used for the target page itself).
     */
    BlockNumber leafleftsib;
    BlockNumber leafrightsib;
    BlockNumber leaftopparent;    /* next child down in the subtree */

    /* xl_btree_metadata FOLLOWS IF XLOG_BTREE_UNLINK_PAGE_META */
} xl_btree_unlink_page;
```

13:

```
typedef struct xl_btree_unlink_page
{
    BlockNumber leftsib;          /* target block's left sibling, if any */
    BlockNumber rightsib;         /* target block's right sibling */

    /*
     * Information needed to recreate the leaf page, when target is an
     * internal page.
     */
    BlockNumber leafleftsib;
    BlockNumber leafrightsib;
    BlockNumber topparent;        /* next child down in the branch */

    TransactionId btpo_xact;      /* value of btpo.xact for use in recovery
        */
    /* xl_btree_metadata FOLLOWS IF XLOG_BTREE_UNLINK_PAGE_META */
} xl_btree_unlink_page;
```

9.7 Additional Information

For more details on the internal workings and additional helper functions used in `parse_wal_file`, refer to the source code in `wal_reader.c`.

10 Troubleshooting

10.1 Could not get version for server

If you get this `FATAL` during startup check your PostgreSQL logins

```
psql postgres
```

and

```
psql -U repl postgres
```

And, check the PostgreSQL logs for any error.

Setting `log_level` to `DEBUG5` in `pgmoneta.conf` could provide more information about the error.

11 Acknowledgement

11.1 Authors

pgmoneta was created by the following authors:

```
Jesper Pedersen <jesper.pedersen@comcast.net>  
David Fetter <david@fetter.org>  
Will Leinweber <will@bitfission.com>  
Luca Ferrari <fluca1978@gmail.com>  
Nikita Bugrovsky <nbugrovs@redhat.com>  
Mariam Fahmy <mariamfahmy66@gmail.com>  
Jichen Xu <kyokitisin@gmail.com>  
Saurav Pal <resyfer.dev@gmail.com>  
Bokket <bokkett@gmail.com>  
Haoran Zhang <andrewzhr9911@gmail.com>  
Hazem Alrawi <hazemalrawi7@gmail.com>  
Shahryar Soltanpour <shahryar.soltanpour@gmail.com>  
Shikhar Soni <shikharish05@gmail.com>  
Nguyen Cong Nhat Le <lenguyencongnhat2001@gmail.com>  
Chao Gu <chadraven369@gmail.com>  
Luchen Zhao <lucian.zlc@gmail.com>  
Joan Jeremiah J <joanjeremiah04@gmail.com>  
Iury Santos <iuryroberto@gmail.com>  
Palak Chaturvedi <palakchaturvedi2843@gmail.com>  
Jakub Jirutka <jakub@jirutka.cz>  
Mario Rodas
```

11.2 Committers

```
Jesper Pedersen <jesper.pedersen@comcast.net>  
Haoran Zhang <andrewzhr9911@gmail.com>
```

11.3 Contributing

Contributions to **pgmoneta** are managed on GitHub

- Ask a question
- Raise an issue
- Feature request
- Code submission

Contributions are most welcome!

Please, consult our Code of Conduct policies for interacting in our community.

Consider giving the project a star on GitHub if you find it useful. And, feel free to follow the project on Twitter as well.

12 License

Copyright (C) 2025 The pgmoneta community

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, **this** list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, **this** list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from **this** software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

BSD-3-Clause

12.1 libart

Our adaptive radix tree (ART) implementation is based on The Adaptive Radix Tree: ARTful Indexing for Main-Memory Databases and libart which has a 3-BSD license as

Copyright (c) 2012, Armon Dadgar
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistributions of source code must retain the above copyright notice, **this** list of conditions and the following disclaimer.
- * Redistributions in binary form must reproduce the above copyright notice, **this** list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- * Neither the name of the organization nor the names of its contributors may be used to endorse or promote products derived from **this** software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL ARMON DADGAR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.