

# Forum:[Hack Forums](#)

Thread: [Emoji's Bug](#)

**Omniscient** - September 30, 2026, 1:01 pm

Unfortunately, I just learned the old server dumps were in UTF8 collation and not utf8mb4. Which means Emoji's are broken. I don't plan on reverting the past 24 hours just to save emoji's. Going forward this is fixed and shouldn't happen again.

I'm fairly certain I'm going to run into many, "oh fucking hell" moments from the server move. But there is no turning back from here. I even cancelled the old server already and it expires tomorrow. This forces me to make this work.

I might be able to run something on the table to remove the bad emoji characters. I'll see what I can do to minimize the effect of this.

Sorry people. You'll just have to add new smilies yourself. ? ? ? ?

**HerrWinston** - September 30, 2026, 1:10 pm

they seem to work all just fine, very glad to see this work again, cancelling the old one atleast gives you a reason to work and work, as I said in the last thread, good luck that this will go by smoothly!

**Omniscient** - September 30, 2026, 1:16 pm

(Sep 30, 2026 01:10 PM)HerrWinston Wrote: [\[CITE\]](#) they seem to work all just fine, very glad to see this work again, cancelling the old one atleast gives you a reason to work and work, as I said in the last thread, good luck that this will go by smoothly!

Smoothly isn't the word I'd want to use. It's already not smooth. What I hope for is minimal damage and quick speedy recovery to normal. I'm already starting to wonder if my time-frame to be 100% is too optimistic. But the plan is really simple. Keep fixing bugs until there are no more bugs. And there are plenty of bugs.

**Jamika** - October 1, 2026, 3:22 am

(Sep 30, 2026 01:01 PM)Omniscient Wrote: [\[CITE\]](#) Unfortunately, I just learned the old server dumps were in UTF8 collation and not utf8mb4. Which means Emoji's are broken. I don't plan on reverting the past 24 hours just to save emoji's. Going forward this is fixed and shouldn't happen again.

I'm fairly certain I'm going to run into many, "oh fucking hell" moments from the server move. But there is no turning back from here. I even cancelled the old server already and it expires tomorrow. This forces me to make this work.

I might be able to run something on the table to remove the bad emoji characters. I'll see what I can do to minimize the effect of this.

Sorry people. You'll just have to add new smilies yourself. ? ? ? ? I'd be lying if I said I cared about emojis at all... I've always added them manually from the keyboard anyway. Maybe it will be worse on PC than on mobile?

**DemonAsylum** - October 1, 2026, 10:19 am

I'm curious, are you able to see on your end exactly how many posts nowadays use emojis? I feel like I don't see them used very often. Any given person is of course only going to see so many posts

though to be fair. It's entirely possible I just don't see the majority of the posts that do in fact use them.

**Omniscient** - October 1, 2026, 10:26 am

(Oct 1, 2026 10:19 AM)DemonAsylum Wrote: [\[CITE\]](#) I'm curious, are you able to see on your end exactly how many posts nowadays use emojis? I feel like I don't see them used very often. Any given person is of course only going to see so many posts though to be fair. It's entirely possible I just don't see the majority of the posts that do in fact use them.

It bothers me that some posts are broken. I'll lose a few hours of work to fix this but I found a way (hopefully) to do this without losing a single post.

Essentially I imported a corrected charset version of the old server into the new server on a separate DB from HF. I'll then run a python script to compare the old posts table to the new posts table looking for variations. If the edittime column matches and the post message does not, then it will update the new DB. Theoretically it should work. The question is how long will the script take to run? But I'll run it in Screen mode so I don't have to babysit it.

Also, the concept of having the old DB on the new server might come in handy for other unforeseen bugs.

**Omniscient** - October 1, 2026, 11:54 am

Fix being implemented right now.

Looking great too. I think it will be done within 30 minutes.

All it took was this simple script. :)

Code `#!/usr/bin/env python3`

```
import argparse
import logging
import os
import sys
import time
import pymysql
```

```
LIVE_DB = "NEWDB"
OLD_DB = "OLDDDB"
TABLE = "mybb_posts"
```

```
MYSQL_SOCKET = "/run/mysqld/mysqld.sock"
```

```
LOG_FILE = "/extra/post-repair.log"
STATE_FILE = "/extra/post-repair.state"
```

```
DEFAULT_BATCH = 10000
DEFAULT_SLEEP = 0.25
```

```
def connect():
    db = pymysql.connect(
        unix_socket=MYSQL_SOCKET,
        user="root",
```

```
    charset="utf8mb4",
    autocommit=False,
    cursorclass=pymysql.cursors.DictCursor
)
```

```
with db.cursor() as cur:
    cur.execute("SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED")
    cur.execute("SET SESSION innodb_lock_wait_timeout=5")
```

```
db.commit()
return db
```

```
def setup_logging():
    logging.basicConfig(
        level=logging.INFO,
        format="%(asctime)s %(message)s",
        handlers=[
            logging.FileHandler(LOG_FILE),
            logging.StreamHandler(sys.stdout)
        ]
    )
```

```
def get_state():
    try:
        with open(STATE_FILE, "r") as f:
            return int(f.read().strip())
    except (FileNotFoundError, ValueError):
        return 0
```

```
def save_state(pid):
    tmp = STATE_FILE + ".tmp"

    with open(tmp, "w") as f:
        f.write(str(pid))

    os.replace(tmp, STATE_FILE)
```

```
def single_pid(db, pid, dry_run=False):
    sql = f"""
    SELECT
        o.pid,
        o.edittime AS old_edittime,
        n.edittime AS new_edittime,
        OCTET_LENGTH(o.message) AS old_bytes,
        OCTET_LENGTH(n.message) AS new_bytes,
        SHA2(o.message, 256) AS old_hash,
        SHA2(n.message, 256) AS new_hash,
        BINARY o.message = BINARY n.message AS message_same
    FROM {OLD_DB}.{TABLE} o
```

```

JOIN {LIVE_DB}.{TABLE} n ON n.pid = o.pid
WHERE o.pid = %s
"""

with db.cursor() as cur:
    cur.execute(sql, (pid,))
    row = cur.fetchone()

if not row:
    print(f"PID {pid}: not present in both databases.")
    return

print(f"PID:      {pid}")
print(f"Edit time:  old={row['old_edittime']} new={row['new_edittime']}")
print(f"Message bytes: old={row['old_bytes']} new={row['new_bytes']}")
print(f"Old SHA256:  {row['old_hash']}")
print(f"New SHA256:  {row['new_hash']}")
print(f"Message same: {'YES' if row['message_same'] else 'NO'}")

if row["message_same"]:
    print("No repair needed.")
    return

if row["old_edittime"] != row["new_edittime"]:
    print("NOT UPDATED: edittime differs.")
    return

if dry_run:
    print("DRY RUN: would replace live message with old message.")
    return

update = f"""
UPDATE {LIVE_DB}.{TABLE} n
JOIN {OLD_DB}.{TABLE} o ON o.pid = n.pid
SET n.message = o.message
WHERE n.pid = %s
      AND n.edittime = o.edittime
      AND BINARY n.message <> BINARY o.message
"""

with db.cursor() as cur:
    cur.execute(update, (pid,))
    changed = cur.rowcount

db.commit()

if changed == 1:
    print("UPDATED successfully.")
    logging.info("REPAIRED PID %d", pid)
else:
    print("NOT UPDATED: row no longer meets repair conditions.")

```

```

def get_pid_range(db, last_pid, batch_size):
    sql = f"""
        SELECT pid
        FROM {OLD_DB}.{TABLE}
        WHERE pid > %s
        ORDER BY pid
        LIMIT %s
    """

    with db.cursor() as cur:
        cur.execute(sql, (last_pid, batch_size))
        rows = cur.fetchall()

    if not rows:
        return None

    return rows[0]["pid"], rows[-1]["pid"], len(rows)

def find_candidates(db, start_pid, end_pid):
    sql = f"""
        SELECT o.pid
        FROM {OLD_DB}.{TABLE} o
        JOIN {LIVE_DB}.{TABLE} n ON n.pid = o.pid
        WHERE o.pid BETWEEN %s AND %s
        AND n.edittime = o.edittime
        AND BINARY n.message <> BINARY o.message
        ORDER BY o.pid
    """

    with db.cursor() as cur:
        cur.execute(sql, (start_pid, end_pid))
        return [row["pid"] for row in cur.fetchall()]

def repair_range(db, start_pid, end_pid):
    sql = f"""
        UPDATE {LIVE_DB}.{TABLE} n
        JOIN {OLD_DB}.{TABLE} o ON o.pid = n.pid
        SET n.message = o.message
        WHERE o.pid BETWEEN %s AND %s
        AND n.edittime = o.edittime
        AND BINARY n.message <> BINARY o.message
    """

    with db.cursor() as cur:
        cur.execute(sql, (start_pid, end_pid))
        return cur.rowcount

def full_run(db, batch_size, sleep_time, dry_run, resume, start_pid):
    if start_pid is not None:
        last_pid = start_pid

```

```

elif resume:
    last_pid = get_state()
else:
    last_pid = 0

scanned = 0
repaired = 0
batches = 0

logging.info(
    "START live=%s old=%s batch=%d sleep=%.2f start_pid=%d dry_run=%s",
    LIVE_DB, OLD_DB, batch_size, sleep_time, last_pid, dry_run
)

while True:
    pid_range = get_pid_range(db, last_pid, batch_size)

    if not pid_range:
        break

    start, end, count = pid_range
    candidates = find_candidates(db, start, end)

    scanned += count
    batches += 1

    if candidates:
        logging.info(
            "RANGE %d-%d scanned=%d candidates=%d",
            start, end, count, len(candidates)
        )

        if dry_run:
            for pid in candidates:
                logging.info("DRY-RUN PID %d", pid)
        else:
            changed = repair_range(db, start, end)
            db.commit()

            repaired += changed

            logging.info(
                "UPDATED range=%d-%d candidates=%d changed=%d",
                start, end, len(candidates), changed
            )
        else:
            db.rollback()

    last_pid = end
    save_state(last_pid)

    if batches % 10 == 0:
        logging.info(

```

```

        "PROGRESS last_pid=%d scanned=%d repaired=%d",
        last_pid, scanned, repaired
    )

    if sleep_time > 0:
        time.sleep(sleep_time)

    logging.info(
        "FINISHED last_pid=%d scanned=%d repaired=%d",
        last_pid, scanned, repaired
    )

def main():
    parser = argparse.ArgumentParser(
        description="Repair HF post messages from old database."
    )

    parser.add_argument(
        "-p", "--pid",
        type=int,
        help="Check or repair a single PID, e.g. -p123456"
    )

    parser.add_argument(
        "--dry-run",
        action="store_true",
        help="Compare only; do not update."
    )

    parser.add_argument(
        "--batch",
        type=int,
        default=DEFAULT_BATCH,
        help=f"Rows per batch, default {DEFAULT_BATCH}"
    )

    parser.add_argument(
        "--sleep",
        type=float,
        default=DEFAULT_SLEEP,
        help=f"Sleep between batches, default {DEFAULT_SLEEP}"
    )

    parser.add_argument(
        "--resume",
        action="store_true",
        help="Resume from saved PID."
    )

    parser.add_argument(
        "--start-pid",
        type=int,

```

```

        help="Start scanning after this PID."
    )

    args = parser.parse_args()

    setup_logging()
    db = connect()

    try:
        if args.pid:
            single_pid(db, args.pid, args.dry_run)
        else:
            full_run(
                db,
                args.batch,
                args.sleep,
                args.dry_run,
                args.resume,
                args.start_pid
            )
    except KeyboardInterrupt:
        db.rollback()
        print("\nInterrupted. Progress preserved.")
    except Exception:
        db.rollback()
        logging.exception("FATAL ERROR")
        raise
    finally:
        db.close()

if __name__ == "__main__":
    main()

```

**Scrub** - October 1, 2026, 11:57 am

(Oct 1, 2026 11:54 AM)Omniscient Wrote: [\[CITE\]](#) Fix being implemented right now.

Looking great too. I think it will be done within 30 minutes.

All it took was this simple script. :)

Code `#!/usr/bin/env python3`

```

import argparse
import logging
import os
import sys
import time
import pymysql

LIVE_DB = "NEWDB"
OLD_DB = "OLDDDB"
TABLE = "mybb_posts"

```

```
MYSQL_SOCKET = "/run/mysqld/mysqld.sock"
```

```
LOG_FILE = "/extra/post-repair.log"
```

```
STATE_FILE = "/extra/post-repair.state"
```

```
DEFAULT_BATCH = 10000
```

```
DEFAULT_SLEEP = 0.25
```

```
def connect():
```

```
    db = pymysql.connect(
        unix_socket=MYSQL_SOCKET,
        user="root",
        charset="utf8mb4",
        autocommit=False,
        cursorclass=pymysql.cursors.DictCursor
    )
```

```
    with db.cursor() as cur:
```

```
        cur.execute("SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED")
```

```
        cur.execute("SET SESSION innodb_lock_wait_timeout=5")
```

```
    db.commit()
```

```
    return db
```

```
def setup_logging():
```

```
    logging.basicConfig(
        level=logging.INFO,
        format="%(asctime)s %(message)s",
        handlers=[
            logging.FileHandler(LOG_FILE),
            logging.StreamHandler(sys.stdout)
        ]
    )
```

```
def get_state():
```

```
    try:
```

```
        with open(STATE_FILE, "r") as f:
```

```
            return int(f.read().strip())
```

```
    except (FileNotFoundError, ValueError):
```

```
        return 0
```

```
def save_state(pid):
```

```
    tmp = STATE_FILE + ".tmp"
```

```
    with open(tmp, "w") as f:
```

```
        f.write(str(pid))
```

```
    os.replace(tmp, STATE_FILE)
```

```

def single_pid(db, pid, dry_run=False):
    sql = f"""
    SELECT
        o.pid,
        o.edittime AS old_edittime,
        n.edittime AS new_edittime,
        OCTET_LENGTH(o.message) AS old_bytes,
        OCTET_LENGTH(n.message) AS new_bytes,
        SHA2(o.message, 256) AS old_hash,
        SHA2(n.message, 256) AS new_hash,
        BINARY o.message = BINARY n.message AS message_same
    FROM {OLD_DB}.{TABLE} o
    JOIN {LIVE_DB}.{TABLE} n ON n.pid = o.pid
    WHERE o.pid = %s
    """

    with db.cursor() as cur:
        cur.execute(sql, (pid,))
        row = cur.fetchone()

    if not row:
        print(f"PID {pid}: not present in both databases.")
        return

    print(f"PID: {pid}")
    print(f"Edit time: old={row['old_edittime']} new={row['new_edittime']}")
    print(f"Message bytes: old={row['old_bytes']} new={row['new_bytes']}")
    print(f"Old SHA256: {row['old_hash']}")
    print(f"New SHA256: {row['new_hash']}")
    print(f"Message same: {'YES' if row['message_same'] else 'NO'}")

    if row["message_same"]:
        print("No repair needed.")
        return

    if row["old_edittime"] != row["new_edittime"]:
        print("NOT UPDATED: edittime differs.")
        return

    if dry_run:
        print("DRY RUN: would replace live message with old message.")
        return

    update = f"""
    UPDATE {LIVE_DB}.{TABLE} n
    JOIN {OLD_DB}.{TABLE} o ON o.pid = n.pid
    SET n.message = o.message
    WHERE n.pid = %s
        AND n.edittime = o.edittime
        AND BINARY n.message <> BINARY o.message
    """

```

```

with db.cursor() as cur:
    cur.execute(update, (pid,))
    changed = cur.rowcount

db.commit()

if changed == 1:
    print("UPDATED successfully.")
    logging.info("REPAIRED PID %d", pid)
else:
    print("NOT UPDATED: row no longer meets repair conditions.")

```

```

def get_pid_range(db, last_pid, batch_size):

```

```

    sql = f"""
        SELECT pid
        FROM {OLD_DB}.{TABLE}
        WHERE pid > %s
        ORDER BY pid
        LIMIT %s
    """

```

```

    with db.cursor() as cur:
        cur.execute(sql, (last_pid, batch_size))
        rows = cur.fetchall()

```

```

    if not rows:
        return None

```

```

    return rows[0]["pid"], rows[-1]["pid"], len(rows)

```

```

def find_candidates(db, start_pid, end_pid):

```

```

    sql = f"""
        SELECT o.pid
        FROM {OLD_DB}.{TABLE} o
        JOIN {LIVE_DB}.{TABLE} n ON n.pid = o.pid
        WHERE o.pid BETWEEN %s AND %s
        AND n.edittime = o.edittime
        AND BINARY n.message <> BINARY o.message
        ORDER BY o.pid
    """

```

```

    with db.cursor() as cur:
        cur.execute(sql, (start_pid, end_pid))
        return [row["pid"] for row in cur.fetchall()]

```

```

def repair_range(db, start_pid, end_pid):

```

```

    sql = f"""
        UPDATE {LIVE_DB}.{TABLE} n
        JOIN {OLD_DB}.{TABLE} o ON o.pid = n.pid

```

```
SET n.message = o.message
WHERE o.pid BETWEEN %s AND %s
AND n.edittime = o.edittime
AND BINARY n.message <> BINARY o.message
.....
```

```
with db.cursor() as cur:
    cur.execute(sql, (start_pid, end_pid))
    return cur.rowcount
```

```
def full_run(db, batch_size, sleep_time, dry_run, resume, start_pid):
    if start_pid is not None:
        last_pid = start_pid
    elif resume:
        last_pid = get_state()
    else:
        last_pid = 0

    scanned = 0
    repaired = 0
    batches = 0

    logging.info(
        "START live=%s old=%s batch=%d sleep=%.2f start_pid=%d dry_run=%s",
        LIVE_DB, OLD_DB, batch_size, sleep_time, last_pid, dry_run
    )

    while True:
        pid_range = get_pid_range(db, last_pid, batch_size)

        if not pid_range:
            break

        start, end, count = pid_range
        candidates = find_candidates(db, start, end)

        scanned += count
        batches += 1

        if candidates:
            logging.info(
                "RANGE %d-%d scanned=%d candidates=%d",
                start, end, count, len(candidates)
            )

            if dry_run:
                for pid in candidates:
                    logging.info("DRY-RUN PID %d", pid)
            else:
                changed = repair_range(db, start, end)
                db.commit()
```

```

        repaired += changed

        logging.info(
            "UPDATED range=%d-%d candidates=%d changed=%d",
            start, end, len(candidates), changed
        )
    else:
        db.rollback()

    last_pid = end
    save_state(last_pid)

    if batches % 10 == 0:
        logging.info(
            "PROGRESS last_pid=%d scanned=%d repaired=%d",
            last_pid, scanned, repaired
        )

    if sleep_time > 0:
        time.sleep(sleep_time)

    logging.info(
        "FINISHED last_pid=%d scanned=%d repaired=%d",
        last_pid, scanned, repaired
    )

def main():
    parser = argparse.ArgumentParser(
        description="Repair HF post messages from old database."
    )

    parser.add_argument(
        "-p", "--pid",
        type=int,
        help="Check or repair a single PID, e.g. -p123456"
    )

    parser.add_argument(
        "--dry-run",
        action="store_true",
        help="Compare only; do not update."
    )

    parser.add_argument(
        "--batch",
        type=int,
        default=DEFAULT_BATCH,
        help=f"Rows per batch, default {DEFAULT_BATCH}"
    )

    parser.add_argument(
        "--sleep",

```

```
    type=float,  
    default=DEFAULT_SLEEP,  
    help=f"Sleep between batches, default {DEFAULT_SLEEP}"  
)
```

```
parser.add_argument(  
    "--resume",  
    action="store_true",  
    help="Resume from saved PID."  
)
```

```
parser.add_argument(  
    "--start-pid",  
    type=int,  
    help="Start scanning after this PID."  
)
```

```
args = parser.parse_args()
```

```
setup_logging()  
db = connect()
```

```
try:  
    if args.pid:  
        single_pid(db, args.pid, args.dry_run)  
    else:  
        full_run(  
            db,  
            args.batch,  
            args.sleep,  
            args.dry_run,  
            args.resume,  
            args.start_pid  
        )  
except KeyboardInterrupt:  
    db.rollback()  
    print("\nInterrupted. Progress preserved.")  
except Exception:  
    db.rollback()  
    logging.exception("FATAL ERROR")  
    raise  
finally:  
    db.close()
```

```
if __name__ == "__main__":  
    main()
```

High quality when things work out easy.

Omniscient - October 1, 2026, 12:17 pm

2026-10-01 12:15:11,885 FINISHED last\_pid=63009630 scanned=35866296 repaired=2279371

Done. LMK if you see any post bugs.

Also, turns out that it was about 6% of all posts have an emoji. That's significant and I'm glad to have done this.

**Scrub** - October 1, 2026, 1:29 pm

(Oct 1, 2026 12:17 PM)Omniscient Wrote: [\[CITE\]](#) 2026-10-01 12:15:11,885 [FINISHED](#)  
last\_pid=63009630 scanned=35866296 repaired=2279371

Done. LMK if you see any post bugs.

Also, turns out that it was about 6% of all posts have an emoji. That's significant and I'm glad to have done this.

Will we ever have an option to add custom emojis to the side?

**DemonAsylum** - October 1, 2026, 3:33 pm

(Oct 1, 2026 12:17 PM)Omniscient Wrote: [\[CITE\]](#) 2026-10-01 12:15:11,885 FINISHED  
last\_pid=63009630 scanned=35866296 repaired=2279371

Done. LMK if you see any post bugs.

Also, turns out that it was about 6% of all posts have an emoji. That's significant and I'm glad to have done this.

Congrats on the fix and thanks for sharing the percentage insight as well. Clearly was worth it.

**Terry A. Davis** - October 1, 2026, 4:12 pm

I miss the old school :blackhat: mybb emote :(

**DemonAsylum** - October 1, 2026, 4:20 pm

(Oct 1, 2026 04:12 PM)Terry A. Davis Wrote: [\[CITE\]](#) I miss the old school :blackhat: mybb emote :(  
That the one where he takes the hat off and tips it forward or whatever?

**Terry A. Davis** - October 1, 2026, 10:26 pm

(Oct 1, 2026 04:20 PM)DemonAsylum Wrote: [\[CITE\]](#) That the one where he takes the hat off and tips it forward or whatever?

Damn, you stalking me without convo, but yeah.

**DemonAsylum** - October 1, 2026, 10:46 pm

(Oct 1, 2026 10:26 PM)Terry A. Davis Wrote: [\[CITE\]](#) Damn, you stalking me without convo, but yeah.  
Your new posts can't hide from me ?

As for the emoji, I thought so and I was a fan of that one as well.

**Terry A. Davis** - October 1, 2026, 11:20 pm

(Oct 1, 2026 10:46 PM)DemonAsylum Wrote: [\[CITE\]](#) Your new posts can't hide from me ?

As for the emoji, I thought so and I was a fan of that one as well.  
I'll start posting more and we can get into a post off

**DemonAsylum** - October 2, 2026, 12:21 am

(Oct 1, 2026 11:20 PM)Terry A. Davis Wrote: [\[CITE\]](#) I'll start posting more and we can get into a post off

Better if we save the epic post off for when convo makes it's triumphant return.

**Terry A. Davis** - October 2, 2026, 1:10 am

(Oct 2, 2026 12:21 AM)DemonAsylum Wrote: [\[CITE\]](#) Better if we save the epic post off for when convo makes it's triumphant return.

But you're always there but not present, just like my dad.

**DemonAsylum** - October 2, 2026, 1:17 am

(Oct 2, 2026 01:10 AM)Terry A. Davis Wrote: [\[CITE\]](#) But you're always there but not present, just like my dad.

If that's what I have you thinking then everything is going according to plan ?

**Omniscient** - October 2, 2026, 10:39 am

(Oct 1, 2026 01:29 PM)Scrub Wrote: [\[CITE\]](#) Will we ever have an option to add custom emojis to the side?

Emoji's are just characters, not images. You can post any emoji you want. That box in the reply is just to make it faster for you. And you should be easily able to alter the emoji's with a userscript.

Also, it could be time I update the emoji set we have. That's easy enough.

**Omniscient** - October 2, 2026, 10:42 am

? loading encrypted image...

btw, I just realized that signatures might be effected by this. I'll run a script to fix those too. Give me an hour or two.

**Terry A. Davis** - October 2, 2026, 12:49 pm

(Oct 2, 2026 10:42 AM)Omniscient Wrote: [\[CITE\]](#) ? loading encrypted image...

btw, I just realized that signatures might be effected by this. I'll run a script to fix those too. Give me an hour or two.

Holy shit I didn't expect to actually be taken seriously with that byte comment last night hahah.

I'll be throwing them right into slots!

**Scrub** - October 2, 2026, 1:03 pm

(Oct 2, 2026 10:39 AM)Omniscient Wrote: [\[CITE\]](#) Emoji's are just characters, not images. You can post any emoji you want. That box in the reply is just to make it faster for you. And you should be easily able to alter the emoji's with a userscript.

Also, it could be time I update the emoji set we have. That's easy enough.

Adding my plug as an emoji would be littttty, then I won't have to try and use the janky looking iPhone one since you know I'm mainly mobile lol

**Terry A. Davis** - October 2, 2026, 1:18 pm

(Oct 2, 2026 01:03 PM)Scrub Wrote: [\[CITE\]](#) Adding my plug as an emoji would be littttty, then I won't have to try and use the janky looking iPhone one since you know I'm mainly mobile lol

They're Unicode, they'll display differently for any device your on unless Omners starts forking out converting to IMG and CDNING them out.

So become one of them and get an Android.

**DemonAsylum** - October 2, 2026, 1:30 pm

(Oct 2, 2026 12:49 PM)Terry A. Davis Wrote: [\[CITE\]](#) Holy shit I didn't expect to actually be taken seriously with that byte comment last night hahah.

I'll be throwing them right into slots!

He must not have wanted to see you keep bringing it up in the future as much as you do shit talking Stanley ?

**Terry A. Davis** - October 2, 2026, 2:25 pm

(Oct 2, 2026 01:30 PM)DemonAsylum Wrote: [\[CITE\]](#) He must not have wanted to see you keep bringing it up in the future as much as you do shit talking Stanley ?

So what you're saying is, that I should keep going and double, if not triple down?

**DemonAsylum** - October 2, 2026, 3:17 pm

(Oct 2, 2026 02:25 PM)Terry A. Davis Wrote: [\[CITE\]](#) So what you're saying is, that I should keep going and double, if not triple down?

You might have to now that I see you've lost all of your bytes again. Speaking of which and like I mentioned on the other thread would happen if you did that.. ?